Proceedings of the Workshop on

Logics in Security

Organised as part of the European Summer School on
Logic, Language and Information (ESSLLI)
August 2010, Copenhagen, Denmark

Dov M. Gabbay and Leendert van der Torre (eds.)

Table of Contents

Secure communication of local states in multi-agent systems

Michael Albert[1], Andrés Cordón–Franco[2], Hans van Ditmarsch[2], David Fernández–Duque[2], Joost J. Joosten[2], and Fernando Soler–Toscano[2]

[1] University of Otago, New Zealand
`malbert@cs.otago.ac.nz`
[2] University of Sevilla, Spain
{`acordon,hvd,dfduque,jjoosten,fsoler`}`@us.es`

Abstract. Given a deal of cards over three agents, we investigate ways for two agents to communicate secrets by public announcements. The problem to keep all of your cards a secret (i) can be distinguished from the problem to keep some of your cards a secret (ii). For (i): we characterize a novel class of protocols consisting of two announcements, for the case where two agents both hold n cards and the third agent a single card; the communicating agents announce the sum of their cards modulo $2n + 1$. For (ii): we show that the problem to keep at least one of your cards a secret is equivalent to the problem to keep your local state (hand of cards) a secret; we provide a large class of card deals for which exchange of secrets is possible; and we give an example for which there is no protocol of less than three announcements.

1 Introduction

Alice and Bob, draw a and b cards from a deck of $a + b + c$ cards, and Eve, the eavesdropper, receives the remaining c cards. Alice and Bob wish to communicate their cards to each other by way of public announcements, without informing Eve of any of their cards. The investigation of the generalized problem with card deal size parameters (a, b, c) was inspired by its $(3, 3, 1)$ instance that was coined in [12] the *Russian Cards Problem*, and that originates with Kirkman [9]. A standard solution for $(3, 3, 1)$ is as follows. Suppose Alice holds 0, 1, and 2, Bob holds 3, 4, and 5, and Eve holds 6. Alice announces that her hand of cards is one of $012, 034, 056, 135, 146, 236, 245$, i.e., one of the seven hands $\{0, 1, 2\}$, etc., after which Bob announces that Eve holds 6. Another solution is that Alice announces that she holds one of the five hands $012, 034, 056, 135, 246$, again followed by Bob announcing that Eve holds 6. We can view such solutions as the execution sequences of an underlying protocol. Some general patterns and special cases of card deal sizes (a, b, c) for which two-announcement solutions exist are found in [2], but a complete characterization is not known.

We can relax the constraints for secrecy in the Russian Cards Problem somewhat. Suppose that the eavesdropper may learn single card ownership for Alice and Bob, but just not their entire holding, i.e., the eavesdropper may not learn the card deal. In that case, simpler protocols suffice. In terms of interpreted systems, Alice and Bob attempt to communicate their local state to each other, without Eve learning their local states. Note that, if Eve were to learn the local state of Alice or the local state of Bill, she would learn the entire deal of cards.

A simple way for Alice to communicate her local state to Bob, in the $(3, 3, 1)$ case, is to announce that she holds one of 012, 034, and 056. In other words, she gives away that she holds card 0, but this does not disclose her whole hand. After that announcement, Bob as before responds that Eve holds 6. We may call Alice's announcement *state safe*, as opposed to *card safe*, above. There is a relation to *bit exchange* problem: is it possible for Alice and Bob to share a secret bit (i.e., the value of a proposition) by public communication. The seminal publication for the latter is [7].

Motivation We are motivated in our investigations by the ground separating unconditionally secure (also known as information-based) from conditionally secure protocols. The security of the latter are based on computational features: the intractability of some computation, a one-way function between some cryptographic primitives, etc. It is tempting to say that unconditionally secure protocols are abstractions of conditionally secure protocols. But this high and dry ground seems very poor: abstracting away from keys and one-way functions seems to remove the essence from reasoning about protocols, and it is therefore unclear what results for the abstraction have to bear on practical security matters and protocol design. We do not bridge that gap. But anything we do, aims to bridge that gap.

Our slightly less ulterior motive is to design fast unconditionally secure protocols for information exchange in multi-agent systems. Within the more specific bounds that we have set, such as card secure protocol or state secure protocols, we aim to find minimum-length protocols, to find the maximum number of bits that can be exchanged, and to analyze multi-agent versions of protocols ('multi-party' in security jargon; with 'multi' as 'more than two') where the intention to securely exchange information about the ignorance and knowledge of other agents (also known as higher-order preconditions in protocol execution) inevitably draws in dynamic epistemic methodology. An additional challenge in that setting is the reconciliation of what may be called more embedded methods with the more abstract logical and combinatorial approaches. Somewhere on the far horizon remains a link with conditional security.

Results This work contains the following contributions. For the card exchange problem for card deal size $(n, n, 1)$ we characterize a novel class of protocols consisting of two announcements. In that case, we treat the set of cards not as a set of (interchangable) labels as in design theory [11], but as set of consecutive numbers $0, 1, ..., 2n$ and employ number theoretical methods and brute force (Haskell). The protocol is simple: both A and B announce the sum of their cards modulo $2n + 1$. The method has promising generalizations. Further, we show that state safe is equivalent to bit safe, and provide a large class of card deal sizes (a, b, c) for which bit exchange is possible (this should be seen as a special case of results in [6]). These protocols typically consist of various announcements, without a claim that these are minimal. We also give an example of a bit exchange protocol consisting of three announcements where no solution of two announcements exists.[3]

2 Card deal terminology and known results

The three *agents* Alice, Bob, and Eve are abbreviated as $\mathsf{A}, \mathsf{B}, \mathsf{C}$. Given a set/*deck* D of $d = a + b + c$ *cards*, their hands of cards A, B, C consist of a, b, c cards. The *card deal* (A, B, C) is the triple of the three hands of cards, and we call this a card deal of *size* (a, b, c). The cards in the deck may be called anything whatsoever, but it is customary to name them $0, 1, ..., d - 1$.

Given that cards are numbers, and that our examples use small numbers, we allow ourselves some abus de langage. Consider size $(3, 3, 1)$. For hand of cards $\{0, 1, 2\}$ we write 012 (and the cards in a hand always in this ascending order), and for deal $(\{0, 1, 2\}, \{3, 4, 5\}, \{6\})$ we write 012.345.6.

We can distinguish the information requirement—what A and B are supposed to learn from each other—from the safety requirement—what C is not supposed to learn from the communications taking place between A and B. The information requirement is for A and B to learn all of their cards (and therefore the entire deal of cards). We call an information state satisfying that requirement *state informative*. The *card safe* requirement is for C to remain ignorant of the ownership of all of A's and B's cards; whereas in *state safe*, the requirement is for C to remain ignorant of the ownership of at least one of those cards (and therefore ignorant about the hand of cards of the other agents, their *local state*).

Protocols to solve these problems consist of a finite number of alternating truthful public announcements by A and B, all of which are informative (trivial announcements are not allowed), and where each announcement consists of a number of alternatives for the hand of cards of the announcing agent. These are not truly restrictive conditions: for a finite number of cards, there are only a finite number of possible card deals, and each informative announcement results in a reduction of these alternatives.

An information state is represented by a Kripke model for what agents know, 'informative announcement' can be defined as one resulting in a proper model restriction, any complex logical statement that is announced is equivalent to an announcement of a number of alternatives for the actual hand of cards, and all states in (a bisimulation contraction of) that Kripke model are about different card deals [12]. (Results we do not explain in technical detail here.) The various safety and information requirements are formulas that can be checked in such a model. In this work we only consider protocols of length two where first A and then B makes an announcement, and protocols of length three where the announcements are made by A, then B, and then A again.

[3] This is interesting, because at the time of submission no such protocol was known for the card exchange problem, although we have now found one for parameters $(4, 4, 2)$, and no other such example is known for the state exchange problem.

All the following should hold for any deal of cards for which a given sequence of announcements can be made truthfully. An announcement is *card safe* / *state safe* if it preserves ignorance of C of all cards / some card (the safety requirement). We will normally call them safe, and let the context determine if card safe or state safe is intended. A sequence of announcements is a protocol.[4] A protocol is safe if it consists of safe announcements. A protocol is state informative if A and B know the card deal after termination, i.e., if the information state reached is state informative. (This implies that the last announcement in the protocol informs the agent addressed by that announcement of the hand of cards of the announcing agent, and that the second-to-last announcement informs the other agent.) A protocol is good if it is safe and state informative.

In [2] some sizes (a, b, c) are listed for which good protocols consisting of two announcements exist, e.g., (a, b, c) such that $a + b + c = p^2 + p + 1$ for any prime $p \leq a - 1$, and $(3, b, 1)$ if $b \geq 3$, and $(a, 2, 1)$ if $a = 0, 4 \mod 6$. In a two-announcement protocol the second announcement is always equivalent to B announcing the cards of C. There may be protocols for (a, b, c) but *not* for (b, a, c), e.g., there is a protocol for $(4, 2, 1)$ but not for $(2, 4, 1)$.

2.1 Subgroup common knowledge or public knowledge?

The role of common knowledge in protocols is of logical interest. The solution requirements discussed in [12] and [2] are formulated for an actual deal of cards, and *not*, as above, for any deal of cards for which the announcements can be truthfully made. Given that, they are required to be commonly known. (The contribution of [12] is to show what can go wrong if that does not hold, by an analysis in public announcement logic.) However, there is a subtlety: it must be common knowledge among all three agents, i.e. *public* knowledge, that the safety requirement is satisfied, and it must be common knowledge among A and B (also known as subgroup common knowledge) that the information requirement is satisfied. An open question remained if the information requirement should also be publicly known. If not, Eve is uncertain whether Alice and Bob have terminated the protocol. In all known cases of this kind, announcing that the protocol is finished then results in Eve learning some of Alice's or Bob's cards. We investigated the matter thoroughly but not exhaustively, and were also greatly helped by the program DEMO [13] for model checking dynamic epistemics. An open question remains:

> Are there protocols after which it is common knowledge to A and B that they know each other's cards (and thus the card deal), and where this is not public knowledge, and where the announcement is safe that the protocol is terminated (after which it is public knowledge that A and B know the card deal)?

[4] Strictly, it is only an execution sequence of an underlying protocol; see the section 'Further Research' and the example on page 13.

Open questions are nice, but they should be relevant. The literature on security protocols suggests that this is not an interesting question:

A time-honoured principle by Kerckhoffs [8] states that the safety of a protocol should not depend on whether the protocol is public or not, with the exception of the 'private' keys of the agents performing in the protocol. For card deal protocols the role of the private keys is played by actual hand of cards of A and B. Protocols ending in common knowledge between A and B that the secret has been exchanged, but where this is not publicly known, are therefore ruled out.

If we do not reason from the perspective of an actual deal of cards, but from the perspective of all cards deals consistent with the announcements made so far, the discussion on subgroup or public common knowledge evaporates: the information and security requirements should then be model validities, from which it follows that they are publicly known. From now on, we assume that the requirements should be met for any card deal consistent with the announcements.

3 Card safe protocols for size $(n, n, 1)$

The five hand solution for the $(3, 3, 1)$ case is also known under the form of the 'sum modulo number of cards' [10]. For example, when Alice holds 012, she announces that the sum of her cards modulo 7 is 3. There are five hands of cards having that sum: 012, 046, 136, 145, 235. Not all hands in the five hand announcement 012, 034, 056, 135, 246 in the introductory section have the same sum, but subject to the permutation of cards $s(0) = 1, s(1) = 0, s(2) = 2, s(3) = 4, s(4) = 5, s(5) = 6, s(6) = 3$ it can be transformed in the modulo 7 solution. And instead of responding by announcing Eve's card, Bob could equivalently have announced the sum of his cards modulo 7.

In [2] an 18 hand solution for $(4, 4, 1)$ and a 66 hand solution for $(5, 5, 1)$ are given, but no general method was known for $(n, n, 1)$. In this section we will present conditions for (a, b, c) for which the announcement by Alice and Bob of the sum of their cards is card safe and state informative. For $(n, n, 1)$ the answer will be: always, if $n \geq 3$.

It should be noted that the sum announcement is not always safe. For example, take card deal size $(4, 2, 1)$. Assume that A holds 0123. It is not (card) safe for A to announce that the sum of her cards is 6. The quadruples summing to 6 are: 0123 0346 0256 1246 1345. If C holds 4, then she learns that A holds 0.

Let $\sum A$ denote the sum of A's cards modulo d, and similarly for other agents, and for the deck $D = 0, 1, ..., d-1$. For our purposes we can equate D with the ring $\mathbb{Z}_d$ of d elements, and $+$ to the sum operation defined on that ring. The announcement by an agent of the sum modulo the total number of cards is called the *sum announcement*, and the protocol consisting of A and then B announcing their sum is called the *sum announcement protocol*.

First let us note that if $c = 1$, the sum announcement informs the other agent of your cards.

Proposition 1. *If $c = 1$ and* A *announces the sum of her cards, then* B *knows* A*'s cards.*

Proof. Let x denote C's only card. Then B can easily compute

$$\sum \mathbb{Z}_d = \frac{d(d-1)}{2}$$

This sum is actually 0 when d is odd, but this is unimportant. We then have the equation

$$\sum A + \sum B + x = \sum \mathbb{Z}_d.$$

Clearly, after A's announcement, B knows all the terms in this equation, and thus can easily solve it for x. Agent A must then have the remaining cards.

The same argument applies if B announces the sum of his cards, so that:

Corollary 1. *For $(a, b, 1)$, the protocol where first* A *announces the sum of her cards and then* B *announces the sum of his cards is state informative.*

A direct result from the proof of Proposition 1 is that

Corollary 2. *A good sum announcement protocol for (a, b, c) is also good for (b, a, c).*

As we have seen in Section 2, this is not necessarily the case for other than sum announcement protocols. Now, let us characterize (card) safety. Consider the 'pair swap' property:

Pair swap (for A)
For every $x_0 \in \mathbb{Z}_d$ and every deal (A, B, C) such that $x_0 \in A$, there exist $x_1 \in A$ and $y_0, y_1 \in B$ with $x_0 \neq x_1$, $y_0 \neq y_1$, and $x_0 + x_1 = y_0 + y_1$. (1)

Proposition 2. *Suppose that the triple (a, b, c) satisfies pair swap for* A*. Then,* C *does not know any of* A*'s cards after $\sum A$ is announced.*

Proof. Let $x_0 \in \mathbb{Z}_d$. Suppose that pair swap for A holds and consider any assignment (A, B, C) with $x_0 \in A$. We will produce a new assignment (A', B', C') such that C cannot distinguish between (A, B, C) and (A', B', C'), even after the announcement of $\sum A$, and $x_0 \notin A'$. This means that C cannot know that A has x_0, and since x_0 is arbitrary, C cannot know any of A's cards.

Pick x_1, y_0 and y_1 satisfying pair swap for A and set

$$\begin{aligned} A' &= (A \setminus \{x_0, x_1\}) \cup \{y_0, y_1\} \\ B' &= (B \setminus \{y_0, y_1\}) \cup \{x_0, x_1\} \\ C' &= C. \end{aligned}$$

Then, C cannot distinguish between (A, B, C) and (A', B', C') because her cards are unchanged, and

$$\begin{aligned} \sum A' &= \sum A - (x_0 + x_1) + (y_0 + y_1) \\ &= \sum A \end{aligned}$$

(because $x_0 + x_1 = y_0 + y_1$). Thus A would have made the same announcement in both cases, and C cannot distinguish the two deals, hence cannot know that A has x_0.

Here we must note that it does not matter whether A announces $\sum A$ or B announces $\sum B$ as far as C is concerned, since she can compute one using the other. Hence for C to remain truly ignorant, we would want not only pair swap to hold for A but also the analogous property we obtain when switching A and B:

Pair swap (for B)
For every $x_0 \in \mathbb{Z}_d$ and every deal (A, B, C) such that $x_0 \in B$, there exist $x_1 \in B$ and $y_0, y_1 \in A$ with $x_0 \neq x_1$, $y_0 \neq y_1$, and $x_0 + x_1 = y_0 + y_1$. (2)

It is clear that an announcement is (card) safe if (1) and (2) hold.[5] We will now investigate when they hold. For this we need a combinatorial theorem, conjectured by Erdös and Heilbronn in [4] and proven by Dias da Silva and Hamidoune in [3]:

Proposition 3 ([3]). *Let d be a prime. For a set $A \subseteq \mathbb{Z}_d$, denote $S^n(A)$ as the set of all sums $x_1 + ... + x_n$ of n distinct elements of A. Then,*

$$|S^n(A)| \geq \min\left\{d, n|A| - n^2 + 1\right\}.$$

In particular for a prime d, any set A defines at least $2|A| - 2^2 + 1 = 2|A| - 3$ sums of pairs, if not all of $\mathbb{Z}_d$. This gives us the following:

Proposition 4. *If d is prime and both*

$$\begin{aligned} 2a - 3 + (b-1) &\geq d + 1 \\ (a-1) + 2b - 3 &\geq d + 1, \end{aligned}$$

then announcing $\sum A$ (or $\sum B$) is card safe.

Proof. We must prove that (1) holds, as well as (2). The situation is symmetric and we shall only prove the former.

Given a card deal (A, B, C), pick $x_0 \in A$. Then,

$$|x_0 + (A \setminus \{x_0\})| = a - 1$$

since $x_0 + x_1 = x_0 + x_2$ would imply that $x_1 = x_2$ and hence we get one distinct value for each sum $x_0 + x_1$. On the other hand, by Proposition 3, $|S^2(B)| \geq 2b - 3$. Since by assumption

$$(a-1) + 2b - 3 \geq d + 1,$$

we see that

$$|x_0 + (A \setminus \{x_0\})| + |S^2(B)| \geq d + 1.$$

[5] The two conditions also imply CA2 and CA3, respectively, in [2].

Now, there are at most d different sums modulo d. Therefore, by the pigeonhole principle, two pairs must have the same sum, and we can find

$$z \in (x_0 + (A \setminus \{x_0\}) \cap S^2(B)$$

satisfying (1).

In the case of $(n, n, 1)$ we do not need $d = 2n + 1$ to be prime, due to the following proposition.

Proposition 5. *If $|A| = n \geq 9$ and $A \subseteq \mathbb{Z}_{2n+1}$, then $|S^2(A)| \geq n + 3$.*

The proof of Proposition 5 is found in the appendix. This gives us the following

Corollary 3. *For any $n \geq 9$, announcing $\sum A$ is card safe in the $(n, n, 1)$ case.*

Proof. For $n \geq 9$, we note that given a deal (A, B, C) and $x_0 \in A$ we have $n - 1$ different sums of the form $x_0 + x_1$ with $x_0 \neq x_1$ and $x_1 \in A$. Further, by Proposition 5, $|S^2(B)| \geq n + 3$, and since

$$n - 1 + |S^2(B)| > 2n + 1,$$

there must be an element of $\mathbb{Z}_{2n+1}$ which can be written both in the form $x_0 + x_1$ with $x_1 \in A$ and $y_0 + y_1$ with $y_0, y_1 \in B$. These elements then satisfy pair swap for A. Once again, pair swap for B follows by symmetry.

We also have that

Lemma 1. *For any $3 \leq n \leq 8$, announcing $\sum A$ is card safe in the $(n, n, 1)$ case.*

Proof. The case for 3 started this section. We have used a simple Haskell script to check that the sum announcement is safe for $4 \leq n \leq 8$. (And we also note that, independently, the cases $(5, 5, 1)$ and $(6, 6, 1)$ are treated in [2].) Indeed, we have checked not only that the security result remains true for $4 \leq n \leq 8$, but also that the method of proof employed in Proposition 2 applies equally well. Namely, for each $4 \leq n \leq 8$, in every card deal of the $(n, n, 1)$ distribution, each of A's cards can be interchanged in a pair with a pair from B's cards with the same sum (modulo $2n + 1$). Pair swap for B follows by symmetry. The Haskell script and some further explanations are found in the appendix.

From Corollary 1, Corollary 3 and Lemma 1 we now obtain that

Theorem 1. *For $n \geq 3$, the sum announcement protocol is a good protocol for size $(n, n, 1)$.*

Protocols for one announcement Alice and Bob can announce their sum at the same time, and this is card safe and state informative. So we can shorten the sum announcement protocol into a single announcement protocol. This is an elementary observation, but still remarkable: for the protocols in [2] (and for all other card protocols that we know off) Bob can only make a specific response *after* Alice's announcement.

Protocols for more than two announcements For (a, b, c) where $c > 1$, the two announcement protocol of both agents announcing the sum does not work. From A's announcement, B still learns the sum of C's cards, but two cards that are held by A instead of C may also have that sum. It is conceivable that B then makes some other informative response (other than announcing *his* sum of cards!), from which A learns his cards, and may then make yet another announcement informing B of C's cards. In other words, number theory may assist us to find good protocols consisting of more than two announcements. For that, we also need to be more general than just swapping pairs.

From swapping pairs to swapping n-tuples Interestingly, in the original Russian cards problem for parameters $(3, 3, 1)$ the swapping pairs argument for showing safety fails. Let us consider the card deal 013.245.6. There is no pair of cards from 013 with the same sum as a pair of cards from 245, for otherwise the remaining cards in each hand would be equal since $0+1+3 = 2+4+5$ modulo 7. Observe that, however, safety can be easily shown by a swapping triples argument: it suffices to interchange the whole players' hands. Indeed, this is a general fact. Given a card deal of the $(3, 3, 1)$ case, if the sum of A's cards is different from the sum of B's cards, the swapping pairs argument works. Otherwise, safety can be shown by exchanging the whole hand of both players.

Employing Haskell, we have encountered several other examples (than $(3, 3, 1)$) where card safety can be shown by a swapping n–tuples argument. Given parameters (a, b, c), for each deal (A, B, C) of that size that may be a different n. This is for instance also the case for deals of size $(4, 4, 2)$, $(4, 4, 3)$, $(5, 5, 2)$, $(5, 5, 3)$, $(5, 5, 4)$, $(6, 6, 2)$, $(6, 6, 4)$, $(6, 6, 5)$, $(7, 7, 2)$, and $(7, 7, 4)$. (As C holds more than one card, none of these are state informative.) It can be even worse: for parameters $(5, 5, 9)$ the sum announcement is still card safe (checked in Haskell), but for a given deal (A, B, C) of that size, n may even vary for different cards $x \in A$. This suggests that other methods of proof for showing card safety should also be investigated.

Finally, back to swapping pairs of cards, we conjecture a strengthening of Proposition 5 that may help us find good protocols consisting of more than two announcements: *In $\mathbb{Z}_{2n+1}$, any set of size n defines at least $2n - 3$ different sums of pairs.* From this conjecture follows that (straightforward proof omitted): *Given is card deal size $(a, b, 1)$. If $a + b$ is even, $b \geq a$ and $a > 5$, then after the announcement of $\sum A$ agent C does not know a single of A's cards.* For card safety, we would also need that C does not know a single of B's cards, but as a may be different from b, this now requires a separate proof.

4 Communicating local states

The models encoding what agents know in a card deal can also be seen as an interpreted system [5], namely where each processor/agent only knows his local state (namely his hand of cards), and where there is public knowledge among all agents of the set of possible global states of the system, where a global state is an n-tuple of local states (given n agents). That a local state consists of several cards is somewhat less relevant from this perspective. The concern of the agents communicating to each other may simply be to keep their local state a secret, but they may not care about each and every of their cards. That is, the protocols should be *state safe*, but not necessarily *card safe.*

In works like [7] the basic building block for secrecy is not a card, or a state, but a *bit.* A bit may be any proposition that the communicating agents wish to share while keeping it a secret from intruders. Given a card deal of size (a, b, c), 'A and B share a secret' means that there is a proposition p such that it is public knowledge (i.e., common knowledge to A, B, and C) that A and B commonly know the value of p but that C remains ignorant of the value of p. A protocol can be called *bit safe* and *bit informative* (or 'a good protocol for bit exchange') if for each initial state of information a sequence of A, B announcements results in an information state with a shared secret.[6] We note that p typically is some factual proposition p (such as 'A holds card 0', 'the deal of cards is 012.345.6', ...), but it can be any proposition, also an epistemic statement; but this is not the situation typically considered in information theory, nor in security protocol analysis. From this perspective, *state informative is bit informative for the proposition describing the deal of cards*; and we note that this is a different proposition in every different state. There are also less obvious correspondences:

Proposition 6. *State safe is bit safe.*

Proof. Assume a state safe protocol. Let $\mathcal{L}$ be a sequence of announcements after which A and B know the card deal, but not C. Then C considers at least two deals (A, B, C) and (A', B', C') possible. As $C = C'$, $A \neq A'$. Let p be the proposition 'Agent A holds A'. Then A and B have common knowledge of p, but C does not know if p. Therefore, the protocol is bit safe.

Now, assume a bit safe protocol, with $\mathcal{L}$ the sequence of announcements realizing the shared bit p such that C does not know the value of p. (The next part of the proof refers to modal logical semantics not explained in detail, and results from [12].) From the last follows, directly from the semantics of the epistemic modal operator, that C considers at least two different possible states in the Kripke model encoding what agents know about each other in card deals. As different states are about different card deals (see Section 2), C considers at least two deals (A, B, C) and (A', B', C') possible. As $C = C'$, $A \neq A'$. Therefore, the protocol is state safe.

Similarly, one might wonder if bit informative is state informative. As said, state informative is bit informative: the description of a state is a bit. But it is quite possible to share a secret bit without disclosing all your cards. But, if it is possible to share a secret bit, is there then also another protocol to safely disclose all of your cards? We think the answer is yes, but we do not know the answer.

[6] A logical analogue exists in group announcement logic [1] with common knowledge where this corresponds to the property that there is a φ, not necessarily a propostional variable, such that $\langle \mathsf{AB} \rangle ((C_{\mathsf{AB}}\varphi \vee C_{\mathsf{AB}}\neg\varphi) \wedge \neg K_{\mathsf{C}}\varphi \wedge \neg K_{\mathsf{C}}\neg\varphi)$ is valid for initial models of card deals.

We continue by showing for a large class of (a, b, c) that they are bit safe.

4.1 Bit exchange protocols

Lemma 2 (type $>$). *If $a, b > c$,* A *and* B *can share a secret after public communication.*

Proof. The proof is by induction on c.

$c = 0$:
A and B already share a secret. Because C does not hold any cards, A knows that B holds the cards A does not hold, and vice versa. The entire deal of cards is common knowledge between A and B. E.g., let x be any card, then A and B now share the secret of the value of 'A holds x'.

$c > 0$:
Suppose that $a \leq b$ (or else, swap the roles of A and B). A chooses one of her own cards, say x, and two of the remaining cards, say y, z. A announces: "I hold exactly one of $\{x, y, z\}$." (Given set notation, there is no order among the three. Else, assume that A randomizes the order before the announcement.) B has either two, one, or zero of these cards. We proceed by these cases.

- If B holds both y and z, B says: "I hold two of these cards."
  A and B now share the secret of (e.g.) the value of 'A holds x'.
- If B holds one of y and z, say y, B says: "I hold one of these cards, namely y. What is your card?" A responds by saying "My card is x." It is now public knowledge that C must have card z. Continue by repeating the procedure for $(a-1, b-1, c-1)$. Note this case is also (type $>$). By induction hypothesis, A and B can share a secret as a result.
- If B holds neither y nor z, B says: "I hold none of those. Which one was yours?"; after which A responds by saying "My card is x." It is now public knowledge that A must have card x. Note this is only possible if $c \geq 2$. Continue by repeating the procedure for $(a-1, b, c-2)$. Note this case is also (type $>$). By induction hypothesis, A and B can share a secret as a result.

Lemma 3 (type $=$). *If $a > b = c > 0$ or $b > a = c > 0$,* A *and* B *can share a secret after public communication.*

Proof. Assume that $a > b = c > 0$ (or else, swap the roles of A and B). (Note that, if $c = b = 0$, no secret can be exchanged, as it is then public knowledge that A holds all cards.) Agent A chooses a card x from her own cards and a card y from the remaining cards. Agent A now announces: "I hold exactly one of $\{x, y\}$." (Again, assume there is no order among the two, or else A should randomize before the announcement.) If B holds y, B responds: "So do I." A and B now share a secret, e.g. that A holds x. Otherwise, B responds: "I hold neither x nor y." (It is now common knowledge to A and C that A holds x and that C holds y.) There are now two cases. (It is not an induction.)

$c = 1$:
A continues by saying: "C holds y." A and B now know the card deal.

$c > 1$:
A continues by saying: "I hold x and C holds y." Proceed with case $(a-1, b, c-1)$ of (type $>$). (If $a > b = c > 0$, then $a-1, b > c-1 > 0$.) In Lemma 2 we proved that A and B can then share a secret after public communication.

We combine Lemmas 2 and 3 in

Theorem 2. *Let $a, b > c$, or $a > b = c > 0$, or $b > a = c > 0$. Then A and B can share a secret after public communication.*

Theorem 2 follows from [6, Theorem 2.1] (also cited in [7]) of which the special case for two agents sharing a secret is that $a + b \geq c + 2$. We note that this involves cases where either a or b is smaller than c, unlike our conditions, so their results are more general. Like ours, the bound in [6] employs a specific construction. It is therefore unclear if that bound is sharp and if for all other card deal size (a, b, c), no secret can be shared between A and B.

At least, also a negative result is found in [7]. The special case for three agents of the matter treated in Section 6 of [7] shows that no bit exchange is possible between two given agents for card deal size $(1, 1, 1)$.

For 'can be shared' we should of course read 'can be *guaranteed* to be shared'. For any (a, b, c) with $a, b \geq 1$, if B responds: "So do I." to A's announcement "I hold exactly one of $\{x, y\}$," then A, B share a secret bit. This observation leads to (we think) a somewhat strange result—strange because it cannot be used to design safe protocols between two *given* agents A, B, and because this observation seems not to have been made in [7], a paper that is otherwise pretty comprehensive on such matters. (We write 'seems' because their terminology is different from ours and hides many implicit consequences, such as the cited [6, Theorem 2.1].) Note that for any (a, b, c): $a, b, c \geq 1$ iff there is uncertainty about the card deal.

Proposition 7. *Given (a, b, c) where there is uncertainty about the card deal, two agents can share a secret.*

Proof. Take any agent i. Let i announce: "I hold exactly one of $\{x, y\}$. The (single!) other agent j for which this also holds now responds: "So do I." Now, i and j share a secret bit. (Namely, the value of the proposition 'i holds card x'.)

4.2 A protocol for $(2, 2, 1)$ of length strictly larger than two

The bit exchange protocols above may consist of more than two announcements. But it is not proved that no shorter protocol to exchange a secret exists in those cases. For card safe protocols there are no known cases (a, b, c) for which only protocols of three or more announcements exist. In this section we present a state informative and state safe protocol of length 3.

Consider card deal size $(2, 2, 1)$. Let the actual card deal be 01.23.4. Now consider the sequence of announcements

> Alice says: "I have one of 01 12 23 34 40," after which Bob says: "Eve has card 4 or card 1," after which Alice says: "Eve has card 4."

We show that this sequence is state safe and state informative. Initially, there are $\binom{5}{2} \cdot \binom{3}{2} = 30$ possible card deals. After Alice's announcement there are 15 remaining deals. In their informational setting they are:

01.23.4	01.24.3	01.34.2		
12.03.4	12.04.3			12.34.0
23.01.4			23.04.1	23.14.0
		34.01.2	34.02.1	34.12.0
	04.12.3	04.13.2	04.23.1	

The lines stand for A-equivalence classes and the columns for C-equivalence classes. There is no visual equivalent for B-classes in this two-dimensional representation. All A-classes and all C-classes consist of three card deals, whereas some B-classes have size 2 and other B-classes have size 1. For example, if the actual deal is 01.23.4, Bob has not learnt Alice's cards, as he cannot distinguish that deal from 04.23.1. After Bob's announcement, the remaining deals are

01.23.4	
12.03.4	
	34.02.1
	04.23.1

It is essential that Bob's announcement is not merely 'Eve has card 1 or 4' but 'I know that Eve has card 1 or 4.' Therefore a deal like 23.01.4 is now eliminated from consideration: although Eve has card 4 in that deal, Bob cannot distinguish it from the other deal 34.01.2. Therefore, Bob considers it possible that 'Eve has card 1 or 4' is false (namely when she holds 2), and therefore he does not *know* that Eve has card 1 or 4.

Bob (of course) still doesn't know Alice's cards, Alice now knows the card deal, and again Eve remains ignorant of Alice's and Bob's hands, although she now has learnt that Alice hold card 1 and Bob holds card 3. After Alice's final announcement we retain

```
01.23.4
12.03.4
```

and we are done. For a change, let us give the protocol underlying this sequence of three announcements:

> **Protocol**
> *Alice*: Let ij be my own cards. Let klm be the remaining cards. My announcement is a random order of the hands ij jk kl lm mi. *Bob*: Let ij be my own cards. If after Alice's announcement I do not know the card deal and (thus) consider it possible that Eve's card is k or l, then I announce that Eve's card is k or l. If after Alice's announcement I know

the card deal, and (thus) that Eve's card is k, then I choose a card l from Alice's cards, and I announce (in random order) that Eve's card is k or l. *Alice*: I announce Eve's card.

It should be noted that for other card deals Bob would already have learnt the entire card deal from Alice's first announcement, but that he should not disclose that, because Eve would then learn the entire card deal, whatever her card was. This is because the card deals remaining if Bob announces that in response to Alice are

```
23.14.0  34.02.1  04.13.2  01.24.3  12.03.4
```

after which Eve always knows the card deal. So even when Bob knows the card deal after Alice announcement, he 'has to feign' not knowing it, by continuing to execute the protocol above.

We still have to show that there is no protocol of length two for parameters $(2, 2, 1)$. This is easy. First, observe that there is no way for Alice to inform Bob of her cards in an announcement where all hands have empty intersection: that would restrict the announcements to two hands only, e.g. 01 23. Therefore, consider an announcement wherein two hands have a card in common. If Alice were to have one of those hands (comprising three of the five cards), she considers it possible that Bob holds the remaining two cards, and thus would not be able to learn her hand of cards. Eve knows that too, and thus eliminates such hands from her consideration. Then at most one hand will remain in the announcement, so that Eve learns the card deal. For example, suppose Alice announces 01 23 04. Her hand cannot be 01 nor 04 for the reasons above. But then Eve concludes she must hold 23! In any announcement consisting of more than three 2-hands, all hands have non-empty intersection with at least one other hand.

Proposition 8. *There are (a, b, c) for which good protocols satisfying state safety always require more than two announcements.*

Although an elementary result, it is a remarkable result: no other case is known to us, and (despite a lot of effort) no such case is known to us for card safety.

5 Further research

We are still investigating generalizations of the sum announcement method, using n-tuple swap instead of pair swap (such as the conjectured results in Section 3). It is already clear that sum announcements are good protocols for far more (a, b, c) than just $(n, n, 1)$, and that this also goes beyond the results in [2].

We mentioned some specific open questions. Are there protocols of length three or more and that cannot be reduced to protocols of shorter length, so that A and B inform each other of their cards and C does not learn any card? Are there protocols wherein A and B inform each other of their cards, and where making this termination public keeps their secret safe? Are there card deals where you can share a bit but not your hand of cards?

Of further logical interest is a language of protocols, and a logic to check protocol properties. As known, in dynamic epistemic logics one can refer to sequences of announcements, and thus to protocols as sets of sequences of announcements. This extensional view of protocols goes a long way, but a more intensional modelling that sees a protocol as a function from agent's local states histories to announcements, would, we think, be progress. A promising logic having such features is found in [14].

Acknowledgement

Hans van Ditmarsch is also affiliated to the University of Otago. We thank Marco Vervoort for first proving an $n + 1$ lower bound version of Proposition 5. We thank the LiS anonymous reviewers for their comments.

References

1. T. Ågotnes, P. Balbiani, H. van Ditmarsch, and P. Seban. Group announcement logic. *Journal of Applied Logic*, 8:62–81, 2010.
2. M.H. Albert, R.E.L. Aldred, M.D. Atkinson, H. van Ditmarsch, and C.C. Handley. Safe communication for card players by combinatorial designs for two-step protocols. *Australasian Journal of Combinatorics*, 33:33–46, 2005.
3. J.A. Dias da Silva and Y.O. Hamidourne. Cyclic spaces for Grassmann derivatives and additive theory. *Bull. London Math. Soc.*, 26:140–146, 1994.
4. P. Erdös and H. Heilbronn. On the addition of residue classes modulo *p*. *Acta Arithmetica*, 9:149–159, 1964.
5. R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge.* MIT Press, Cambridge MA, 1995.
6. M.J. Fischer and R.N. Wright. Multiparty secret key exchange using a random deal of cards. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 141–155. Springer-Verlag, 1992.
7. M.J. Fischer and R.N. Wright. Bounds on secret key exchange using a random deal of cards. *Journal of Cryptology*, 9(2):71–99, 1996.
8. A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–38 and 161–191, 1883.
9. T. Kirkman. On a problem in combinations. *Camb. and Dublin Math. J.*, 2:191–204, 1847.
10. K.S. Makarychev and Yu.S. Makarychev. The importance of being formal. *Mathematical Intelligencer*, 23(1):41–42, 2001.
11. D.R. Stinson. *Combinatorial Designs – Constructions and Analysis.* Springer, 2004.
12. H. van Ditmarsch. The russian cards problem. *Studia Logica*, 75:31–62, 2003.
13. J. van Eijck. DEMO — a demo of epistemic modelling. In J. van Benthem, D. Gabbay, and B. Löwe, editors, *Interactive Logic — Proceedings of the 7th Augustus de Morgan Workshop*, pages 305–363. Amsterdam University Press, 2007. Texts in Logic and Games 1.
14. Y. Wang, L. Kuppusamy, and J. van Eijck. Verifying epistemic protocols under common knowledge. In *TARK '09: Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 257–266, New York, NY, USA, 2009. ACM.

Appendix: Proof of Proposition 5 on page 8

Let us denote the elements of $\mathbb{Z}_{2n+1}$ by $0, 1, \ldots, 2n$. Throughout this proof we shall sometimes numbers as elements of $\mathbb{Z}_{2n+1}$ or just as integers so that we can, for example speak of an order. The context will always clarify which of the two is meant.

We shall consider any subset A of $\{0, 1, \ldots, 2n\}$ of size n. We are interested in the amount of different sums of pairs of two different numbers from A that we can form. This amount is invariant under shifting the whole set A along some distance. That is, $|S^2(w + A)| = |S^2(A)|$: we are interested in the number of different sums and if $x+y \neq u+v$ then $x+y+2w \neq u+v+2w$, whence shifting all points along a distance w does not affect the number of different sums of pairs. Thus, we may always assume that $0 \in A$.

We call two points $x, y \in A$ *consecutive* when either $x < y$ and for no $z \in A$ we have $x < z < y$ or when x is the largest point in A and y is the smallest point in A. Let us look at the *minimal distance between two consecutive points* in A and call this *dist*.

What can the value of *dist* be? Clearly it can be 1 but it cannot be larger than 2. For, if we fix the first point, say, at 0, then there are $2n$ points left to allocate the remaining $n-1$ points. But $3(n-1) > 2n$ for $n > 3$ thus the remaining points cannot be all at distance 3 from each other. It is easy to see that *dist* $= 2$ is possible. Thus our proof has two cases.

dist $= 2$

What about distance 2? The first point after 0 would be at 2 and the last point at $2n + 1 - 3$. Hence we see that necessarily we fill the remaining space with one distance of 3 and the remaining distances of 2. Without loss of generality we may assume that our first point is at 0 and the second point is at 3 and the $n-2$ remaining points all are of the form $2m+3$. It is now clear that the element 0 defines $n-1$ sums, namely $3, 5, \ldots$ Moreover, it is clear that the element 3 together with each of the $n-2$ elements of the form $3+2m$ defines a new sum. Thus, we count $(n-1)+(n-2) = 2n-3$ which is certainly at least $n+3$ for $n \geq 6$.

dist $= 1$

This case is rather more involved and needs some case distinctions. We define a *gap* to be a pair (x, y) of consecutive elements $\pmod{2n+1}$ such that $y-x \neq 1$. For example, if $n = 3$ and we consider the set $\{2, 3, 5\}$, then $(2, 3)$ are considered consecutive and not a gap, while $(5, 2)$ are considered also consecutive (given that we are looking at these as elements of the cyclic group $\mathbb{Z}_7$) but are a gap, since $2 - 5 \equiv 4 \pmod 7$.

An *interval* is a set of consecutive elements (without gaps), for example $\{2, 3, 4\}$ but not $\{2, 4\}$. Let $E \subseteq \mathbb{Z}_{2n+1}$. By SE we mean the set of sums of pairs of elements in E. Clearly $E \setminus \{0\}$ is always a subset of SE, since 0 is an element of E; we will use this fact throughout the following. We consider five cases, according to the number of gaps.

1. If we have only one gap, that means we have an interval $\{0,1,...,n\}$ and this gives us $2n-3$ sums of pairs. Of course we cannot have zero gaps.
2. If we have two gaps, then E consists of two intervals, one of which has at least five elements (we are using the assumption that $n \geq 9$). We can then assume without loss of generality that $\{0,...,4\} \subseteq E$. Now, if we add the differences between consecutive elements this must be $2n+1$; since we have only two gaps of width (say) g_1, g_2 this becomes $n-2+g_1+g_2$. Here the $n-2$ comes from the pairs of consecutive elements which do not form gaps. So we must have
$$(n-2)+g_1+g_2 = 2n+1$$
which becomes
$$g_1+g_2 = n+3,$$
and thus one of the two must be at least $\left\lfloor \frac{n+3}{2} \right\rfloor$, which because $n \geq 9$ is at least 6. Then, let (x_1, y_1) and (x_2, y_2) be the two gaps, with $y_1 - x_1 \geq 6$. $E \setminus \{0\} \subseteq SE$ gives us $n-1$ elements. Further, $x_1 + i \in SE$ with $i = 1,2,3$ and $x_2 + 1 \in SE$, adding up to $(n-1)+3+1 = n+3$ elements.
3. If we have three gaps, S consists of three intervals, and at least one has three elements. Further, one gap has width at least 3, because if no gaps had width 3 we would have that the sums of differences of consecutive elements is $(n-3)+6 = n+3$, which is smaller than $2n+1$ provided that $n > 2$. We now consider two subcases:
   (a) E contains an interval with four elements. Then, we can assume without loss of generality that $\{0,1,2,3\} \subseteq E$. Now, if (x,y) is a gap of width at least 3, $x+1, x+2 \in SE$, while the other two gaps contribute at least one element each, plus $E \setminus \{0\} \subseteq SE$ giving us a total of at least $2+2+(n-1) = n+3$ elements.
   (b) E contains no interval with four elements. Then, E contains more than two intervals with three elements. This can be shown as follows. Let (x,y) be a gap of width at least 3. There exist three consecutive elements $z, z+1, z+2$ such that $z+2 \neq x$ (since we have at least two such intervals we can pick one or the other accordingly), and thus we can assume without loss of generality that $\{0,1,2\} \subseteq S$ and there is a gap (x,y) of width ≥ 3 such that $x \neq 2$. But then once again $x+1, x+2 \in SE$, and the same computation as above gives us the desired bound.
4. If we have four gaps, S consists of four intervals, and at least one of them must have three elements. Thus we can assume $\{0,1,2\} \subseteq E$. As always $E \setminus \{0\} \subseteq SE$, and if (x,y) is a gap, then $x+1 \in E$ giving us an extra four elements, for a total of $(n-1)+4 = n+3$.
5. If there are at least five gaps, then assume without loss of generality that $\{0,1\} \subseteq E$. In that case, at least four of the gaps are of the form (x,y) with x not equal to 1, and hence we have that $x+1$ is an element of SE, giving us four new elements for a total of $(n-1)+4 = n+3$.

```
import Data.List
-- (subsets n xs) outputs the list of all the subsets of xs of n elements.
subsets :: Int -> [Int] -> [[Int]]
subsets 0 _ = [[]]
subsets _ [] = []
subsets (n+1) (x:xs) = [x:ys | ys <- subsets n xs] ++ subsets (n+1) xs
-- (subsetSum m n xs) outputs the list of all the sums (modulo m) of the subsets of xs of n elements.
subsetSum :: Int -> Int-> [Int] -> [Int]
subsetSum m n xs =  nub [mod (sum ys) m | ys <- subsets n xs]
-- (deals a b c) generates all the deals in an (a,b,c) card distribution.
deals a b c = [[xs,ys,zs] | xs <- subsets a [0..(a+b+c-1)],
                            ys <- subsets b ([0..(a+b+c-1)] \\ xs),
                            zs <- [([0..(a+b+c-1)] \\ xs) \\ ys]]
-- (check m n as bs) checks whether each card of as can be interchanged in an n-tuple with an
-- n-tuple of elements of bs with the same sum (modulo m).
check m n as bs = and [ or [elem (mod (x+y) m) ys | y <- subsetSum m (n-1) (as \\ [x])]| x <- as]
                  where ys = subsetSum m n bs
-- (secure a b c n) checks whether for each deal of an (a,b,c) card distribution, [as,bs,cs],
-- each card of as can be interchanged in an n-tuple with an n-tuple of elements of bs with
-- the same sum (modulo a+b+c).
secure a b c n = and [check (a+b+c) n as bs | [as,bs,_] <- deals a b c]
-- (secure2 a b c) checks whether for each deal of an (a,b,c) card distribution, [as,bs,cs],
-- there exists some n <= min(a,b) such that each card of as can be interchanged in an n-tuple
-- with an n-tuple of elements of bs with the same sum (modulo a+b+c)"
secure2 a b c = and [or [check (a+b+c) n as bs | n <- [2..min a b] ]| [as,bs,_] <- deals a b c]
```

Fig. 1. The Haskell script `subsets.hs`

Appendix: Haskell script used in Lemma 1 on page 8

Figure 1 shows the Haskell script `subsets.hs`. The implemented algorithm is the natural brute force one. In the general setting of an (a, b, c) card distribution, we firstly define a function

```
deals :: Int -> Int -> Int ->  [[Int]]
```

so that `deals a b c` generates all the $\binom{a+b+c}{a} \cdot \binom{b+c}{b}$ possible card deals in an (a, b, c) card distribution. Observe that for parameters $(8, 8, 1)$ this amounts to generating 218790 card deals. Next, we define an auxiliary predicate

```
check :: Int -> Int -> [Int] -> [Int] ->  Bool
```

so that `check d k as bs` checks whether each card of A's hand *as* can be interchanged in an k–tuple with a k–tuple of elements of B's hand *bs* with the same sum (modulo d). Finally, combining `deals` and `check` we define the main *generate and test* predicate

```
secure :: Int -> Int -> Int -> Int ->  Bool
```

so that `secure a b c k` checks whether for each card deal of an (a, b, c) card distribution each card of A's hand can be interchanged in a k–tuple with a k–tuple of elements of B's hand with the same sum (modulo $a + b + c$). (For *card safety* we also have to check `secure b a c k`, namely that each card of B's hand can be interchanged in a k–tuple with a k–tuple of elements of A's hand with the same sum – but when $a = b$ this holds by symmetry, as in the case $(n, n, 1)$ below.)

Therefore, the following Haskell evaluation gives us the proof that the announcement of the sum of A's cards is also secure for $4 \leq n \leq 8$ (as well as the additional information that this fact can be established by a *swapping pairs* argument):

```
Main> and [secure n n 1 2 | n <- [4..8]] True
```

In view of the remarks in Section 3 on generalizing pair swap to swap of n-tuples, it is natural to add to our Haskell script a nonuniform version of the predicate `secure`

```
secure2 ::  Int -> Int -> Int ->  Bool
```

so that `secure2 a b c` checks whether for each card deal of an (a, b, c) card distribution there is some $k \leq min(a, b)$ such that each card of A's hand can be interchanged in a k–tuple with a k–tuple of elements of B's hand.

A preference logic-based approach for Alert correlation

Salem Benferhat and Karima Sedki

CRIL CNRS UMR 8188, Université d'Artois,
Rue Jean Souvraz SP 18 62307 Lens
Cedex, France,
`{benferhat,sedki}@cril.univ-artois.fr`
`http://www.cril.univ-artois.fr`

Abstract. Alert correlation consists in reducing the large amount of alerts that intrusion detection systems (IDS) produce. In this paper, we present a new alert correlation approach based on knowledge and preferences of security operators. Our logic-based approach allows to rank-order produced alerts on the basis of a security operator knowledge about the system, used IDS and his preferences about alerts that he wants to analyze or to ignore. It is based on the development of a new non-classical logic for representing preferences, called *FO-MQCL* (First Order - Minimal Qualitative Choice Logic). Our logic extends a fragment of the first order logic by adding a new logical connective. The general idea is to present only alerts that fully fit security operator's preferences and knowledge. And if needed, less preferred alerts can also be presented. Experimental results shows how our approach is useful for reducing the set of reported alerts.

Key words: Preferences logic, Alert correlation, Security Operator's Knowledge and Preferences

1 Introduction

Intrusion Detection Systems (IDS) are important tools to detect abnormal and malicious activities that attempt to compromise the integrity, confidentiality and availability of resources on the system [1, 16]. Currently, there are two main intrusion detection approaches. Misuse detection [18] based on signatures of known attacks. These approaches achieve very high detection rates on known attacks. However, they are not capable to detect new attacks that do not follow predefined patterns. Anomaly approach [5] is based on a definition of the normal activity profile. Any activity or behavior that deviates from the normal profile is considered as anomaly or possible intrusion. These approaches have the advantage to detect unknown and new attacks. However, they generate a large amount of alerts due to deviations or changes of authorized user's behaviors or actions. Both of these approaches are important to protect the system. However, they present several problems. The main problem concerns the large amount of alerts that IDS produce. The security operator who analyzes and takes appropriate decisions is often overwhelmed. Several alert correlation approaches have been proposed in recent years to address this problem.
Alert correlation tools [7, 10, 14, 20, 22, 15, 13, 19, 23] are important for reducing the large volume of alerts that are generated by IDS. Three kinds of alert correlation techniques can be distinguished.

- Alerts correlation based on similarity between attributes: These techniques exploit similarities between alerts attributes. In [22], a probabilistic method that considers feature overlap, feature similarity, minimum similarity, and expectation of similarity was used to group or correlate alerts based on the similarities between some selected attributes, such as source and destination IP addresses of the victim and attacker. In [20], a similar approach that is based on the port-scan detector tool Spice (Stealthy Probing and Intrusion Correlation Engine), was used to detect stealthy port-scans. Other techniques that belong to the same category are proposed in [10, 6].

- Alert correlation based on known attacks scenarios: This class of approaches [21, 12] is based on known attacks scenarios (i.e., steps that attackers use in their attacks). Examples of attack languages to specify attack scenarios can be found in [11, 8]. This family of correlation approaches give satisfactory results for well-defined and known attacks sequences, but they cannot discover novel attack scenarios.

- Alert correlation based on prerequisites and consequences relationship [7, 14]: The main idea for these approaches is that earlier stages of attacks are often used in a preparation for later ones. The prerequisites of attack is the necessary conditions for the attack to be successful, while the consequences of an attack are the possible outcomes of an attack. Such approaches have the potential of discovering unknown attack pattern.

Existing alert correlation techniques are definitely useful to reduce initial set of reported alerts. However, the number of remaining alerts to be manually handled by the security operator is still high. Clearly, a security operator uses his knowledge about system, reliability of IDS and also his preferences (for instance, on the basis on how risky is an action) to select a subset of alerts to be first analyzed. The aim of this paper is to first develop a logic that represents security operator's knowledge and preferences. Then, we develop an inference mechanism that allows to select a subset of alerts to be first presented to the security operators.

We propose a logic is called First Order - Minimal Qualitative Choice Logic (*FO-MQCL*) and it is an extension of a fragment of first order logic. It is also an extension of a basic part of QCL (Qualitative Choice Logic) logic [4]. QCL adds to classical propositional logic a new connective, called *ordered disjunction*, used to express preferences between alternatives. One of the important limitation of QCL is that it does not correctly deal with negated and conditional preferences of the form (If A is preferred to B then C is preferred to D). Conditional rules that involve preferences are expressed using propositional implication.

The rest of this paper is organized as follows: Section 2 provides formal definition of our new logic. Section 3 presents our alert correlation approach and Section 4 presents some experimental results. Section 5 concludes the paper.

2 *FO-MQCL*: A new Logic for Handling Preferences

This section presents our new logic to be used to represent knowledge and preferences of a security operator. The starting point is an existing logic called *Qualitative Choice Logic* (QCL) [4]. The non-standard part of QCL logic is a new logical connective $\times$, called *ordered disjunction*, which is fully embedded in the logical language. Intuitively, if A and B are propositional formulas then A $\times$ B means: if possible A, but if A is impossible then at least B. Existing logics for representing preferences are not fully satisfactory because most of theses logics, particularly those that are based on the propositional language, cannot express generic knowledge that involve variables and conditional rules of the form "if A is preferred to B then C is preferred to D". For example, QCL logic inference relation is not satisfactory when one deals with complex preferences [4]. This is due to the way rules are handled, and where ordered disjunction is lost when preferences are associated with negation. Indeed, preferences of the form (A $\times$ B)$\Rightarrow$ C is equivalent to ($\neg$ A $\vee$ $\neg$B)$\Rightarrow$ C where ordered disjunction is replaced by standard disjunction (see [2, 17] for more details). On the other hand propositional logic is not appropriate to express complex knowledge. It can only deal with pieces of information regarding particular situations or properties, and cannot express generic knowledge that involve variables. The extensions of QCL proposed in [2] are not satisfactory for our application, in particular they are defined within propositional language. What is needed is a richer language, based on a fragment of first order logic, to express general pieces of information. Our new logic is called *FO-MQCL*, and will be formally described in next section.

2.1 *FO-MQCL* language

The language of *FO-MQCL* logic is a fragment of first order logic. It is composed of three types of formulas: i) universally quantified first order formulas which will be used to express knowledge of a security operator, ii) universally quantified basic choice formulas which will be used to express simple preferences of a security operator and iii) universally quantified general choice formulas which will be used to express complex and general preferences. The *FO-MQCL* language is given by the following five definitions:

Definition 1 (Terms). *Let $X = \{x, y, z, ...\}$ be a set of variables. Let $C = \{c_1, c_2, ..., c_n\}$ be a set of constants. We define a term as either a constant of C or a variable of X.*

This is a restricted definition of the notion of terms used in first order logic since no function symbol is used in this paper.

Definition 2. *Let us denote by PS a set of predicate symbols. The language of unquantified formulas, denoted $PROP_{PS}$ is defined as follows:*

1. *If $p \in PS$ is a predicate symbol of arity n, and $t_1, t_2, ..., t_n$ are terms, then $p(t_1, t_2, ..., t_n)$ is an unquantified first order logic formula (i.e. $p \in PROP_{PS}$).*
2. *If $p \in PROP_{PS}$, $q \in PROP_{PS}$, then $p \wedge q$, $p \vee q$, $\neg p$ are unquantified first order logic formulas.*
3. *Unquantified first order logic formulas are only obtained by applying items (1) and (2) a finite number of times.*

Clearly, $PROP_{PS}$ language is close to propositional language and does not use all the expressive power of the first order logic. However, it is sufficient for our application. In fact, this language will be used to express a security operator's preferences. However, from reasoning point of view and for implementation purposes, it is better to work on propositional level in order to exploit existing inference tools such as SAT-Solvers (SAT for satisfiability tests) which are publicly available. Namely, it is important to instantiate first order knowledge's bases to propositional ones. Let us now define the concept of a basic choice formulas and a general choice formulas to represent simple and complex preferences respectively. Given a set of unquantified first order formulas $a_1, a_2, ..., a_n$, the formula $a_1 \times a_2 \times \ldots \times a_n$ is used to express and order a list of alternatives: some a_i must be true, preferably a_1, but if this is not possible then a_2, if this is not possible a_3, etc.

Definition 3 (Basic Choice Formulas). *Basic Choice Formulas (BCF) are ordered disjunctions of unquantified first order formulas. They offer a simple way to order available alternatives. The language composed of BCF formulas is denoted by BCF_{PS} and is the smallest set of sentences defined as follow:*
1. *If $\phi \in PROP_{PS}$ then $\phi \in BCF_{PS}$*
2. *If ϕ, $\psi \in BCF_{PS}$ then $(\phi \times \psi) \in BCF_{PS}$*
3. *Every* basic choice formula *is only obtained by applying the two rules above a finite number of times.*

Definition 4 (General Choice Formulas). *General Choice Formulas (GCF) represent any formula that can be obtained from PS using connectors $\times$, $\wedge$, $\vee$, $\neg$. The language composed of GCF formulas, denoted GCF_{PS}, is defined as follows:*
1. *If $\phi \in BCF_{PS}$ then $\phi \in GCF_{PS}$*
2. *If ϕ, $\psi \in GCF_{PS}$ then $(\phi \wedge \psi)$,$\neg(\psi)$, $(\phi \vee \psi)$, $(\phi \times \psi) \in GCF_{PS}$.*
3. *The language of GCF_{PS} is only obtained by applying the two rules above a finite number of times.*

Definition 5 (FO-$MQCL$ formulas). *The set of universally quantified formulas is defined as follows:*
- *If $\phi \in PROP_{PS}$ is an unquantified first order formula , and $\{x_1, x_2,..., x_n\}$ are the set of variables used in ϕ, then ($\forall$ x_1,..., $\forall$ x_n ϕ) is a universally quantified first order formula (i.e. FO-$PROP$ formula). The language of universally quantified first order formulas is symbolized by FO-$PROP_{PS}$.*
- *If $\phi \in BCF_{PS}$ is a basic choice formula, and $\{x_1, x_2,..., x_n\}$ are the set of variables used in ϕ, then ($\forall$ x_1,..., $\forall$ x_n ϕ) is a universally quantified basic choice formula (i.e. FO-BCF formula). The language of universally quantified basic choice formulas is symbolized by FO-BCF_{PS}.*
- *If $\phi \in GCF_{PS}$ is a general choice formula, and $\{x_1, x_2,..., x_n\}$ are the set of variables used in ϕ, then ($\forall$ x_1,...,$\forall$ x_n ϕ) is a universally quantified general choice formula (i.e. FO-GCF formula). The language of universally quantified general choice formulas is symbolized by FO-GCF_{PS}.*

Example 1. Assume that a security operator prefers to analyze alerts that are issued by Bro IDS to those that are issued by Snort IDS. We use the FO-$MQCL$ language to define this preference, which is written using the following universally quantified basic choice formula:

$$\phi = \forall x, \forall y\ (\textit{Show-To-Security-Operator-Alert}(x) \wedge \textit{IDS-Bro}(x)) \times \\ (\textit{Show-To-Security-Operator-Alert}(y) \wedge \textit{IDS-Snort}(y)).$$

This formula means that "a situation where an analysis (by a security operator) of an alert issued by Bro IDS is preferred to a situation where an analysis of an alert issued by Snort". $\textit{Show-To-Security-Operator-Alert}(x)$ is the predicate which means that x will show (to be analyzed) to the security operator.
Assume that we have a general rule saying that if a security operator prefers to analyze suspected inbound alerts to suspected outbound ones then he wants to analyze Web alerts. This preference is encoded by the following universally quantified general choice formula:

$$\forall x, \forall y, \forall z, (\textit{Suspected-Inbound-Alerts}(x) \times \textit{Suspected-Outbound-Alerts}(y)) \Rightarrow \\ \textit{Show-To-Security-Operator-Web-Alerts}(z).$$

If a security operator wants to analyze all suspected inbound alerts, the FO-$MQCL$ language allows to encode this knowledge by using the following universally quantified first order formula (which does not involve ordered disjunction $\times$):

$$\psi = \forall x\ \textit{Show-To-Security-Operator-Suspected-Inbound-Alerts}(x).$$

In the above formulas: x, y, z denotes variables where their values are the identification of alerts.

2.2 Inference relation of *FO-MQCL* formulas

The semantics of *FO-MQCL* formulas (*FO-PROP*, *FO-BCF* or *FO-GCF*) is based on the degree of satisfaction of a formula in a particular model I. As in standard propositional logic, an interpretation I is an assignment of the classical truth values T(True), F(False) to the atoms in PS. I will be represented by the set of its satisfied atoms.
The symbol $\models$ represents the propositional inference relation. $I \models \phi$ means that the interpretation I satisfies a formula ϕ (in this case, the satisfaction degree of the formula is always equal to 1). So, we can also write $\models_1$.
$I \not\models \phi$ means that the interpretation I does not satisfy a formula ϕ.
The symbol $\models_k$ represents the non classical inference. For example, $I \models_k \psi$ means that the interpretation I satisfies a formula ψ to degree k (k $\geq$ 1).

Inference relation of universally quantified basic choice formulas The inference relation of *FO-BCF* formulas is given in the following definition:

Definition 6. *1. Let $\phi = \forall\ x_1, \forall x_2,..., \forall\ x_n$ $a_1(x_1)\times a_2(x_2)\times \ldots \times a_n(x_n)$.*
Let $Inst(\phi) = a_1(c_1)\times a_2(c_2)\times \ldots \times a_n(c_n)$ be an instance of ϕ where c_1 ..., c_n are constants. Then $I \models_k a_1(c_1)\times a_2(c_2)\times ... \times a_n(c_n)$ if and only if $I \models a_1(c_1) \vee a_2(c_2) \vee ... \vee a_n(c_n)$, and $k = min(j\ |I \models a_j(c_j))$.
2. Let $\phi \in FO\text{-}PROP_{PS}$, and $Inst(\phi)$ be an instance of ϕ. $I \models_1 Inst(\phi)$ if and only if $I \models Inst(\phi)$.

Namely, given a basic choice formula
$a_1(c_1)\times a_2(c_2)\times \ldots \times a_n(c_n)$, an interpretation or a solution I satisfies ϕ to a degree k, if it satisfies the k^{th} option of ϕ (namely $a_k(c_k)$) and falsifies the first (k-1) options of ϕ. Interpretation I is said to be preferred to an interpretation I', with respect to a given formula ϕ, if the degree of satisfaction of ϕ by I is less than the degree of satisfaction of ϕ by I'.
Note that contrary to the inference relation of *FO-BCF* formulas where the satisfaction degree can be greater than 1, the satisfaction degree of satisfied *FO-PROP* formulas is equal to 1.

Example 2. Assume that we have two alerts A_1 and A_2. Let ϕ = *Alert-Snort*(A_1) $\times$ *Alerts-Bro*(A_2). The satisfaction degree of each interpretation with respect to this preference is given in the following table.

Interpretation	*Alert-Snort*(A_1)	*Alerts-Bro*(A_2)	ϕ
I_1	F	F	∞
I_2	F	T	2
I_3	T	F	1
I_4	T	T	1

Hence, from this table, the interpretations where *Alert-Snort*(A_1) is true is preferred to interpretations where *Alert-Bro*(A_2) is true. Indeed, the satisfaction degree of *Alert-Snort*(A_1) is 1 in its models I_3 and I_4, while it is greater than 1 in I_1 and I_2.

Inference from universally quantified general choice formulas This section proposes our inference relation for universally quantified general choice formulas which simply reuses Definition 6 after a normalization step that transforms a *FO-GCF* formula into a *FO-BCF* formula. The starting point is that a general choice formula is a combination of basic choice formulas using usual connectors $\vee$, $\wedge$, $\neg$ and the new connector $\times$. We argue that these connectors can be distributed, and hence any complex choice formulas can be equivalently transformed into basic choice formulas. The idea is then to provide an equivalent *FO-BCF* formula for each *FO-GCF* formula.
The following introduces the notion of normal form function, which associates with each *FO-GCF* formulas (complex form of preferences), its corresponding *FO-BCF* formulas (simple form of preferences). This normal form function is denoted by $\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}$.

Definition 7. *We define a normal function denoted by* $\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}$*, a function from FO-GCF*$_{PS}$ *to FO-BCF*$_{PS}$*, such that:*

1) **Case of universally quantified basic choice formulas**
   - *Normal form of universally quantified basic choice formulas are these formulas themselves:*
     $\forall\ \phi \in$ *FO-BCF*$_{PS}$,$\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\phi) = \phi$.

2) **Case of combination of universally quantified basic choice formulas**
   - *Normal form of negated, conjunction and disjunction of FO-BCF formulas are:*
     Let $\phi = a_1 \times \ldots \times a_n$, *and* $\psi = b_1 \times \ldots \times b_m$ *such that* $a_i \in$ *FO-PROP*$_{PS}$, $b_i \in$ *FO-PROP*$_{PS}$,

     (a) $\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}((a_1 \times \ldots \times a_n) \wedge (b_1 \times \ldots \times b_m)) \equiv c_1 \times \ldots \times c_k$, *where* $k = max(m, n)$, *and*

$$c_i = \begin{cases} [(a_1 \vee \ldots \vee a_i) \wedge b_i] \vee [a_i \wedge (b_1 \vee \ldots \vee b_i)] \\ \quad \text{if} \quad i \leq min(m,n) \\ ((a_1 \vee \ldots \vee a_n) \wedge b_i) \quad \text{if} \quad n < i \leq m \\ (a_i \wedge (b_1 \vee \ldots \vee b_m)) \quad \text{if} \quad m < i \leq n \end{cases}$$

     (b) $\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\phi \vee \psi) \equiv c_1 \times c_2 \times \ldots \times c_k$, *where* $k = max(m, n)$, *and*

$$c_i = \begin{cases} (a_i \vee b_i) & \text{if} \quad i \leq min(m,n) \\ a_i & \text{if} \quad m \leq i \leq n \\ b_i & \text{if} \quad n \leq i \leq m \end{cases}$$

     (c) $\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}\neg(\ a_1 \times\ a_2 \times \ldots \times\ a_n) \equiv \neg\ a_1 \times \neg a_2 \times \ldots \times \neg a_n$.

   The conjunctive and disjunctive operators have the same definitions as the one proposed in original QCL logic [4], but the negation fully departs from the one given in [4].

3) **Case of combination of universally quantified general choice formulas**
   - The normal form with respect to negated formulas:

   (a) $\forall\ \phi \in$ *FO-GCF*$_{PS}$ *and* $\phi \notin$ *FO-BCF*$_{PS}$,
   $\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\neg\phi) \equiv \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\neg\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\phi))$.

   - The normal form is decomposable with respect to conjunction, disjunction and ordered disjunction of FO-GCF formulas:

(b) $\forall\ \phi, \psi \in$ *FO-GCF*$_{PS}$ *and* ($\phi \notin$ *FO-BCF*$_{PS}$ *or* $\psi \notin$ *FO-BCF*$_{PS}$),
$\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\phi \wedge \psi) \equiv \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\phi) \wedge \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\psi))$.

(c) $\forall\ \phi, \psi \in$ *FO-GCF*$_{PS}$ *and* ($\phi \notin$ *FO-BCF*$_{PS}$ *or* $\psi \notin$*FO-BCF*$_{PS}$),
$\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\phi \vee \psi) \equiv \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\phi) \vee \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\psi))$.

(d) $\forall\ \phi, \psi \in$*FO-GCF*$_{PS}$ *and* ($\phi \notin$*FO-BCF*$_{PS}$ *or* $\psi \notin$ *FO-BCF*$_{PS}$),
$\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\phi \times \psi) \equiv \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\phi) \times \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\psi))$.

Property 1 (Case of universally quantified basic choice formulas) says that the normal form of a basic choice formula ϕ, is the formula ϕ. Property 2 ((a), (b), (c)) gives the definition of conjunction, disjunction and negation applied to basic choice formulas. Property 2-(a) confirms the meaning of conjunction of preferences. Indeed, assume $a_1 \times a_2 \times \ldots \times a_n$ denotes a preference of a user A, and $b_1 \times b_2 \times \ldots \times b_m$ denotes a preference of B. Applying, conjunction of preferences allows to select solutions that privilege A and B. For instance, $a_1 b_1$ (which represents the best choice for A and the best choice for B) is preferred to ($a_2 b_2$) (which represents the best second choice for A and the second choice for B).
Property 3 ((a), (b), (c), (d)) expresses that the normal form function applied to general choice formulas is decomposable with respect to negation, conjunction, disjunction and ordered disjunction.

The above definition in fact provide a recursive way to normalize a general choice formulas. The stopping criteria is when the formula itself is a simple (basic) choice formula (case 1). If ψ is a combination of two simple choice formulas, then case 2 of Definition 7 provides a direct way to get a simple choice formula. If ψ is in fact a combination of two general choice formulas, then first normalize recursively each formula (case 3) and then apply case 2 of Definition 7.

Example 3. Let the following general choice formula ψ be such that: $\psi = ((a_1 \times\ a_2) \vee (a_3 \times a_4)) \wedge\ (b_1 \times\ b_2)$.
To give the satisfaction degree of the formula ψ for each interpretation, we first normalize ψ (i.e. we transform ψ into equivalent basic choice formula) as indicated in Definition 7. After the normalization step, we use Definition 6 to compute the satisfaction degree of obtained basic choice formula.

1. **Normalization of ψ:**
   ψ is a combination (conjunction) of two formulas, the first formula $((a_1 \times\ a_2) \vee (a_3 \times\ a_4))$ is a general choice formula and the second one $(b_1 \times\ b_2)$ is a basic choice formula.
   Using item 3-(b) of Definition 7, we have
   $\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(((a_1 \times\ a_2) \vee (a_3 \times\ a_4)) \wedge (b_1 \times b_2))$
   $\equiv \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}\ (\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}((a_1 \times\ a_2) \vee (a_3 \times\ a_4)) \wedge \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(b_1 \times b_2))$.

   We first use item 2-(b) of Definition 7 to normalize $(a_1 \times\ a_2) \vee (a_3 \times\ a_4)$.
   So, $\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}((a_1 \times\ a_2) \vee (a_3 \times\ a_4)) \equiv (a_1 \vee a_3) \times\ (a_2 \vee a_4)$.
   Then, we use item 1 of Definition 7 to normalize $(b_1 \times\ b_2)$ which is already a basic choice formula. Namely,
   $\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}((b_1 \times\ b_2)) \equiv b_1 \times b_2$.

   Thus, $\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(((a_1 \times\ a_2) \vee (a_3 \times a_4)) \wedge (b_1 \times b_2))$
   $\equiv \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}\ (\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}((a_1 \times\ a_2) \vee (a_3 \times\ a_4)) \wedge \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(b_1 \times b_2))$.
   $\equiv \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(((a_1 \vee a_3) \times\ (a_2 \vee a_4)) \wedge (b_1 \times b_2))$

At this step, we have to normalize a conjunction of two basic choice formulas, so, we use item 2-(a) of Definition 7. The final result is:
$\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\psi) \equiv ((a_1 \vee a_3) \wedge b_1) \times (((a_1 \vee a_2 \vee a_3 \vee a_4) \wedge b_2) \vee ((a_2 \vee a_4) \wedge b_1))$.

2. **Computing satisfaction degree of ψ:**
   - Let $I_1 = \{a_2, b_2\}$. Applying item 2 of Definition 6, we have:
     $I_1 \not\models (a_1 \vee a_3) \wedge b_1$, and $I_1 \models ((a_1 \vee a_2 \vee a_3 \vee a_4) \wedge b_2) \vee ((a_2 \vee a_4) \wedge b_1)$.
     Using item 1 of Definition 6, we have: $I_1 \models_2 \mathcal{N}_{\mathcal{MQCL}}(\psi)$. So, I_1 satisfies ψ to degree 2.

   - If $I_2 = \{b_2\}$, then $I_2 \not\models \mathcal{N}_{\mathcal{MQCL}}(\psi)$. I_2 does not satisfy ψ.

   - If $I_3 = \{a_1, b_1\}$, then $I_3 \models (a_1 \vee a_3) \wedge b_1$ and $I_3 \not\models ((a_1 \vee a_2 \vee a_3 \vee a_4) \wedge b_2) \vee ((a_2 \vee a_4) \wedge b_1)$. Using item 1 of Definition 6, we have $I_3 \models_1 \mathcal{N}_{\mathcal{MQCL}}(\psi)$. I_3 satisfies ψ to degree 1.
     This means that ψ is totally satisfied by I_3.

The above definitions (Definition 6 and Definition 7) provide a satisfactory degree of one basic choice formula or general choice formula with respect to a given interpretation. This allow to rank-order different interpretations and define the notion of preferred models when we have a set of pieces of knowledge and a set of preferences. Let K be a set of universally quantified first order formulas which represents knowledge or integrity constraints, and let T be a set of preferences that contains only simple form preferences (universally quantified basic choice formulas). We suppose that all complex form preferences (general choice formulas) are first transformed into simple form preferences using Definition 7.

Definition 8. *Let $M^k(T)$ denote the subset of universally quantified basic choice formulas of T satisfied by a model M to a degree k (using Definition 6). A model M_1 is $\{K \cup T\}$-preferred over a model M_2 if $\sum_{k=1}^{n} | M_1^k(T)| > \sum_{k=1}^{n} | M_2^k(T)|$, where n is the maximum number of satisfaction degrees that can be associated with a formula ($| M_1^k(T)|$ indicates the number of formulas in T that are satisfied to degree k by a model M_1). M is a preferred model of $\{K \cup T\}$ if and only if:*
1. M is model of K,
2. M is maximally $\{K \cup T\}$-preferred.

Again Definition 8 differs from the lexicographic ordering given in [4]. Our definition allows compensations between satisfactions degrees.

It is important to be precise, that there is no priorities between the different formulas and the inference relation of any formula is given independently of other formulas, but the preferred models are computed from the inference relation of each formula. This means that if a given interpretation does not satisfy one formula, it can not be a preferred model. For example, if the security operator gives two preferences: inbound alerts are preferred to outbound ones, and TCP alerts are preferred to UDP alerts, then these two preferences have the same priorities.

Example 4. Assume that K contains one formula

$$\phi_1 = \forall x \neg \textit{Show-To-Security-Operator-Ping-alerts}(x),$$

and T contains one preference represented by the formula ϕ_2 as follows:

$\phi_2 = \forall x, \forall y, \forall z$ $\textit{Show-To-Security-Operator-Web-alerts}(x) \vee (\textit{Show-To-Security-Operator-Scan-alerts}(y) \times \textit{Show-To-Security-Operator-Ping-alerts}(z))$.

ϕ_1 is a universally first order quantified formula that allows to encode the knowledge of a security operator who considers that alerts *Ping* have not to be presented for analysis. ϕ_2 is a universally quantified general choice formula that indicates that either a security operator wants to analyze alerts *Web*, or he prefers analyzing *Scan* alerts to *Ping* ones.
We first compute the satisfaction degree (inference relation) of formulas ϕ_1 and ϕ_2. The satisfaction degree of the formula ϕ_1 is given directly using item 2 of Definition 6. Concerning the inference relation of the formula ϕ_2, we first use Definition 7 to normalize it (for short we use *Show* instead of *Show-To-Security-Operator* in table 1).

Show-Web-alerts(A'_1)	*Show-Scan-alerts*(A'_2)	*Show-Ping-alerts*(A'_3)	(1)	(2)
True	True	True	∞	1
True	**True**	**False**	**1**	**1**
True	False	True	∞	1
True	**False**	**False**	**1**	**1**
False	True	True	∞	1
False	**True**	**False**	**1**	**1**
False	False	True	∞	2
False	False	False	1	∞

Table 1. Satisfaction degree of formulas of $K \cup T$

- **Normalization of the formula ϕ_2:**
Let $\phi'_2 = \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\phi_2)$, using item 2-(b) of Definition 7, we have:

  $\phi'_2 = \mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}(\forall x, \forall y, \forall z\ \textit{Show-To-Security-Operator-Web-alerts}(x)$
  $\vee(\textit{Show-To-Security-Operator-Scan-alerts}(y) \times \textit{Show-To-Security-Operator-Ping-alerts}(z)))$
  $\equiv \forall x, \forall y, \forall z\ (\textit{Show-To-Security-Operator-Web-alerts}(x) \vee \textit{Show-To-Security-Operator-Scan-alerts}(y)) \times \textit{Show-To-Security-Operator-Ping-alerts}(z)$.

  After normalization of the formula ϕ_2, $K \cup T$ contains a universally quantified first order formula ϕ_1 and a universally quantified basic choice formula ϕ'_2.

- **Instances of formulas ϕ_1 and ϕ_2:** in this example, we suppose that we have three different alerts: (A'_1, A'_2 and A'_3) such that A'_1 is a *Web* alert, A'_2 is a *Scan* and A'_3 is a *Ping*, then *Inst*(K) and *Inst*(T) which are respectively instances of ϕ_1 and ϕ'_2 are given in the following:

  $\textit{Inst}(K) = \textit{Inst}(\phi_1) = \neg\ \textit{Show-To-Security-Operator-Ping-alerts}(A'_3)$ - (1)
  and
  $\textit{Inst}(T) = \textit{Inst}(\phi'_2) = (\textit{Show-To-Security-Operator-Web-alerts}(A'_1) \vee \textit{Show-To-Security-Operator-Scan-alerts}(A'_2)) \times \textit{Show-To-Security-Operator-Ping-alerts}(A'_3)$ - (2)

- **Computing satisfaction degrees:** in Table 1, we give for each formula (1) in *Inst*(K) and (2) in *Inst*(T) associated satisfaction degree (1, 2) if they are satisfied or not (∞) given an interpretation.

- **Preferred models**: Three interpretations (bold line) satisfy the formulas (1) and (2) and *Show-Web-alerts*(A'_1) and *Show-Scan-alerts*(A'_2) are true in two models, thus preferred alerts are $Web\text{-}alerts(A'_1)$ and $Scan\text{-}alerts(A'_2)$. This means that the security operator will analyze in the first Web and Scan alerts.

3 Application of $\boldsymbol{FO\text{-}MQCL}$ to alert correlation

Intrusion Detection Systems use a specific format to report alerts in order to facilitate the exploitation (analysis, explication, etc.) of these alerts. IDMEF (Intrusion Detection Message Exchange Format) is one of the well-known used format [9]. IDMEF allows to define common data formats and exchange procedures for sharing important information to intrusion detection and response systems and those that may need to interact with them. Using this format, each generated alert is characterized by a set of attributes.

3.1 Description of inputs

The inputs of our approach contain four elements: a group of alerts generated by IDS, a set of facts which gathers information describing each alert, a knowledge base and a preference base of a security operator.

1. **A group of alerts G produced by IDS:** Each alert is characterized by a set of attributes called *basic attributes*. Examples of basic attributes are: Timestamp, Signature Identifier (SID), messages associated with alerts, Protocol, IP source and IP destination addresses, source port and destination port, TTL (Time To Live), identification field (ID), etc.
2. **Alert Facts:** Each alert's attribute will be represented by a predicate symbol. The set of predicates (or facts) containing values of alert's attributes of G will be represented by K_1.

   Example 5. Assume that G contains one alert identified by id_1. Assume that the attributes concerning this alert are: IDS identity is Snort, the used protocol is TCP and the class of attack is DoS. These facts will be represented by $K_1 = \{IDS(id_1, Snort)$, $Protocol(id_1, TCP)$, $Class(id_1, DoS)\}$. Note that in general, some attributes may not be informed (known) by an IDS.

   We distinguish two kinds of facts:

   (a) **Alert facts:** These facts are directly defined on basic attributes of alerts. $Protocol(A_1, TCP)$ is an example of alert facts which indicates that the attribute protocol of alert A_1 takes a value TCP.

   (b) **Other facts:** These facts concern attributes that are not directly informed by the IDS from which the alerts are issued. *Direction* of an alert is an example of this kind of facts. It is based on source and target IP address. This information allows to know the suspected direction of concerned alerts on the system (*suspected-inbound*, *suspected-outbound*, *suspected-inside*).

3. **Knowledge of a security operator:** The security operator can provide some knowledge or beliefs on networks, on system, etc. This knowledge base is denoted by K_2, it contains a set of universally quantified first order formulas (namely, formulas that do not involve $\times$). An example of simple knowledge which means that a security operator involves that ftpwrite attack should be presented is the following rule: $\forall x\ Sid(x, 553) \Rightarrow Show\text{-}To\text{-}Security\text{-}Operator\text{-}alert(x)$.

4. **Preferences of a security operator:** The security operator can express his preferences according to what he wants to first analyze and what he would like to ignore. This will be represented with a set of *FO-MQCL* formulas T. T contains the set of universally quantified basic choice formulas (*FO-BCF* formulas) and a universally quantified general choice formulas (*FO-GCF* formulas) which represent the preferences of the security operator.

   Example 6. Let ϕ be a universally quantified basic choice formula:

   $\phi = \forall x, \forall y\ IDS(x, Snort) \wedge Type(x, Web) \wedge IDS(y, Bro) \wedge Type(y, Scan) \Rightarrow$ *Show-To-Security-Operator-alert*(x) $\times$ *Show-To-Security-Operator-alert*(y).
   Intuitively, this formula means that if an alert x is provided by the IDS Snort and concerns an attack Web, and if an alert y is provided by the IDS Bro and concerns an attack Scan, then the security operator prefers first to analyze the alert x, then the alert y.

3.2 Output of our model

The output of our model is a subset $G' \subseteq G$. More precisely, the subset of alerts in G to be first presented to a security operator.
To select alerts to be first presented to a security operator according to his preferences, we need to preprocess available alerts and encode them in our logical framework. The result of our model is given by the following algorithm:

Algorithm 1 (Define a set of preferred alerts)

```
• Inputs:
    • $G$: The group of alerts in IDMEF format
    • $K_2$: The set of propositional formulas
    • T: The set of preferences
• Transform each attribute in IDMEF format into a predicate fact to be added in $K_1$
• Use $\mathcal{N}_{\mathcal{FO}-\mathcal{MQCL}}$ (Definition 7) to normalize the set of universally quantified general choice formulas
  from $T$ if exists.
• Select only instantiated formulas that concern considered facts.
• Compute the satisfaction degrees of the instantiated formulas (Definition 6).
• Compute the preferred models (Definition 8)
• Finally, define $G'$ as a subset of alerts $x$, where Show-To-Security-Operator-Alert$(x)$ is true in most
  of preferred models. Namely, if one denotes $|x|$ the number of preferred models that satisfy $x$, then
  $G' = \{ x: x \in G, \neg\exists\, y, |y| > |x|\}$.
```

4 Experimental results on DADDi's data

We present in this section some results on alerts produced by Snort IDS concerning DADDi[1] data set (Dependable Anomaly Detection with Diagnosis) which are describing in the following subsection.

[1] The authors would like to thank the participants to the DADDi (http://www.rennes.supelec.fr/DADDi/) project for providing this real data.

4.1 Characteristics of DADDi data

Rough real network traffic concerning this data is collected on university campus in the DADDi project (Dependable Anomaly Detection with Diagnosis) during spring 2007 (between May 24 and June 11 2007).
The traffic contains normal and some abnormal connections (Portscan, slammer (worm propagation), bibtexRawHTTP, write-fs-HTTP3, login-http, etc.). Note that this traffic includes inbound and outbound TCP, UDP and ICMP connections. Some information on DAADi's data are given in the following:

- Duration: 18 days of traffic.
- Size: 100 Giga octets.
- IP protocols: TCP (65.84%), UDP (19.66%), ICMP (14.50%).
- Services: HTTP (9.63%), SSH (1.11%), FTP (0.22%), auth (0.31%), DNS/Domain (15.02%), etc.

The volume of collected data is important (100 files numbered 1 to 100, and where the size of each file is around 1 Giga octets. These data are stored in files of 1 Giga octets each one because we have used them for other operations such as processing the rough network traffic into formated connection records described by useful features using our developed tool [3].) and their analysis with Snort IDS have generated a large amount of alerts. For experimentations studies, only the 15 first files (files #1-#5, #6-#10 and #11-#15) that are available for all members of the project.

4.2 Expressing preferences and knowledge regarding DADDi data

Preferences and knowledge presented below are inspired from discussions between some members of DADDi project, expressing what is desirable and what alerts should be discarded for a security operator.

Security operator knowledge

- $\forall x, \forall y\ SameIPsrc(x,y) \wedge SameIPdst(x,y) \wedge SamePortsrc(x,y) \wedge SamePortdst(x,y) \wedge SameSid(x,y) \wedge Timestamp\text{-}Smaller\text{-}than(x,y) \wedge Differ(x,y) \Rightarrow Show\text{-}To\text{-}Security\text{-}Operator\text{-}Alert(x) \wedge \neg Show\text{-}To\text{-}Security\text{-}Operator\text{-}Alert(y)$.

  By this knowledge, the security operator wants to delete all redundant alerts. More precisely, if two alerts have the same source and destination IP addresses, same source and destination ports and same signature, then only earlier alert is preserved.

- $\forall x\ Protocol(x, ICMP) \wedge Type(x, 8) \wedge Code(x, 0) \Rightarrow Show\text{-}To\text{-}Security\text{-}Operator\text{-}alert(x)$.
  The security operator wants analyze alerts where ICMP type is equal to "8" and the code is "0" (Ping alerts).

- $\forall x\ Protocol(x, UDP) \wedge Port(x, 1434) \Rightarrow Show\text{-}To\text{-}Security\text{-}Operator\text{-}alert(x)$.

  The security operator wants to analyze all "UDP" alerts that concern packets sent to port number 1334. These alerts represent "SQL Slammer" vulnerability that is based on buffer overflow in resolution service of SQL server.

Security operator preferences

- $\forall x, \forall y, Service(x, http) \wedge Service(y, other) \wedge Differ(x, y) \rightarrow$ *Show-To-Security-Operator-alert*$(x) \times$ *Show-To-Security-Operator-alert*(y).
  This formula means that the security operator prefers to analyze "http" alerts than others.

- $\forall x, \forall y, \forall z$ $Direction(x, suspected\text{-}inbound) \wedge Direction(y, suspected\text{-}outbound) \wedge Direction(z, suspected\text{-}inside) \wedge Differ(x, y, z)$
  $\Rightarrow$ (*Show-To-Security-Operator-alert*$(x) \vee$ *Show-To-Security-Operator-alert*$(y)) \times$ *Show-To-Security-Operator-alert*(z).
  This formula indicates that the security operator prefers to analyze in the first "suspected inbound" or "suspected outbound" alerts, and then " suspected inside" alerts.

- $\forall x, \forall y$ $Host(x, suspected\text{-}Spider) \wedge Host(y, Other) \wedge Message(x,$ "WEBMISC robots.txt access") $\wedge$ $Message(y,$ "WEBMISC robots.txt access") $\wedge$ $Differ(x, y) \Rightarrow$ *Show-To-Security-Operator-alert*$(y) \times \neg$ *Show-To-Security-Operator-alert*(x).

  This preference concerns alerts reporting to robots programs[2]. Let us briefly recall the description of this attack. To specify which pages to be not indexed, Web administrators indicate these pages in a specific file `robot.txt`. This file is located at the root of the website. Intruders try access to files
  `robot.txt` where they can find important information concerning hidden files, trees directory secret or any other information that administrators have left by accident or neglect. So, Snort raises an alert whenever the file `robot.txt` is accessed. The number of alerts that are generated daily on accesses to the file `robot.txt` is very important (in the files #1-#5 for example, we found 295 alerts).
  To reduce the number of these alerts, a security operator can group these alerts according to source host that accessed to the file `robot.txt`, and he can ignore alerts where source hosts are known and trusted (such as known search engines). The preference above aims to representing suspected alerts that represent accesses to the file `robot.txt` where sources hosts do not belong to the group *Spider* (i.e. known search engines such as Google and Yahoo).

Results of applying our model to different alerts are summarized in table 2.

	Number of alerts	Number of preferred alerts	reduction rate
Files #1-#5	76380	24953	67.33%
Files #6-#10	77825	31090	60.05%
Files #11-#15	97788	36615	62.55%
Total	**251993**	**92658**	**63.31%**

Table 2. Preferred alerts in DADDi data

[2] Programs indexing web pages.

We observe that the reduction rate of alerts is important. The set of preferred alerts contains several suspicious alerts requiring analysis such as "SQL Slammer", "Ping", "Access to the `robot.txt` file". Note that there is significant number of redundant Ping alerts, about (48592 alerts in the files #1-#5 for example). Of course, the security operator can reduce more the number of alerts by considering that "Ping" alerts are less important than the others. In this case, the number of preferred alerts will be equal to 9671 in files #1-#5.

5 Conclusion

This paper has proposed a new logic called *FO-MQCL* (First Order-Minimal Qualitative Choice Logic) which extends a fragment of the first order logic by adding a new logical connective called ordered disjunction. This logic is rich because it has many advantages like a representation of simple and complex preferences and knowledge of a security operator, computing satisfaction degrees and preferred models of different knowledge and preferences, normalization of preferences, etc. Since security operators cannot analyze a large number of alerts produced by IDS, our logic is useful to filter alerts by presenting only alerts that fit security operator knowledge and preferences. Experimental results on real data (provided by DADDi project) show how useful and important our proposed logic contributes to the reduction rate of reported alerts.

Acknowledgments

This works is supported by the SETIN06 project PLACID (Probabilistic graphical models and Logics for Alarm Correlation in Intrusion Detection).

References

1. James P. Anderson, *Computer security threat monitoring and surveillance*, Anderson Company, Pennsylvania, 1980.
2. Salem Benferhat and Karima Sedki, *A revised qualitative choice logic for handling prioritized preferences*, Ninth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, oct 2007, pp. 635–647.
3. Salem Benferhat, Karima Sedki, and Karim Tabia, *Reprocessing rough network traffic for intrusion detection purposes*, IADIS: International Conference Telecommunications, Networks and Systems (Portugal), 2007.
4. Gerard Brewka, Salem Benferhat, and Daniel Le Berre, *Qualitative choice logic*, Artificial Intelligence Journal(AIJ) **157** (2004), no. 1-2, 203–237.
5. Banerjee Arindam Chandola Varun and Kumar Vipin, *Anomaly detection: A survey*, In *ACM Computing Surveys Journal* (2009).
6. Frédéric Cuppens and Fabien Autrel, *Using an intrusion detection alert similarity operator to aggregate and fuse alerts*, The 4th Conference on Security and Network Architectures, 2005, pp. 6–10.
7. Frédéric Cuppens and Alexandre Miège, *Alert correlation in a cooperative intrusion detection framework*, Proceedings of the 2002 IEEE Symposium on Security and Privacy (USA), 2002, p. 202.
8. Frédéric Cuppens and Rodolphe Ortalo, *Lambda: A language to model a database for detection of attacks*, RAID'00: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection (London, UK), 2000, pp. 197–216.

9. D. Curry and Herv Debar, *Intrusion detection message exchange format data model and extensible markup language (xml) document type definition, draft-itetfidwg- idmef-xml-03.txt*, 2001.
10. Klaus Julisch, *Clustering intrusion detection alarms to support root cause analysis*, Journal ACM Transactions on Information and System Security **6** (2003), 443–471.
11. Cédric Michel and Ludovic Mé, *Adele: an attack description language for knowledge-based intrusion detection*, PIFIP/SEC, 2001, pp. 353–365.
12. Benjamin Morin and Hervé Debar, *Correlation of intrusion symptoms: an application of chronicles*, Proceedings of the 6th International Conference on Recent Advances in Intrusion Detection (RAID'03), 2003, pp. 94–112.
13. Benjamin Morin, Ludovic Mé, Hervé Debar, and Mireille Ducassé, *A logic-based model to support alert correlation in intrusion detection*, Inf. Fusion **10** (2009), no. 4, 285–299.
14. Peng Ning, Yun Cui, and Douglas S. Reeves, *Constructing attack scenarios through correlation of intrusion alerts*, CCS '02: Proceedings of the 9th ACM conference on Computer and communications security (New York, NY, USA), ACM, 2002, pp. 245–254.
15. Xi Peng, Yugang Zhang, Shisong Xiao, Zheng Wu, JianQun Cui, Limiao Chen, and Debao Xiao, *An alert correlation method based on improved cluster algorithm*, Proceedings of the 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application (USA), 2008, pp. 342–347.
16. F. Sabahi and A. Movaghar, *Intrusion detection: A survey*, In Third International Conference on Systems and Networks Communications, 2008, pp. 23–26.
17. Benferhat Salem and Sedki Karima, *Alert correlation based on a logical handling of administrator preferences and knowledge*, International Conference on Security and Cryptography (SECRYPT'08) (Porto, Portugal), jul 2008, pp. 50–56.
18. Hanaoka Miyuki Shimamura Makoto and Kono Kenji, *Filtering false positives based on server-side behaviors*, In IEICE Transactions on Information and Systems Journal **2** (2008), 264–276.
19. Dondo Maxwell Smith Reuben, Japkowicz Nathalie and Mason Peter, *Using unsupervised learning for network alert correlation*, In *Advances in Artificial Intelligence Journal* (2008), 308–319.
20. Stuart Staniford, James A. Hoagland, and Joseph M. McAlerney, *Practical automated detection of stealthy portscans*, Journal of Computer Security **10** (2002), no. 1-2, 105–136.
21. Eric Totel, Bernard Vivinis, and Ludovic Mé, *A language driven intrusion detection system for events and alerts correlation*, In Proceedings of the IFIP International Information Security Conference, 2004, pp. 209–224.
22. Alfonso Valdes and Keith Skinner, *Probabilistic alert correlation*, Recent Advances in Intrusion Detection (RAID 2001), Lecture Notes in Computer Science, no. 2212, Springer-Verlag, 2001, pp. 54–68.
23. Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera, *Decentralized multi-dimensional alert correlation for collaborative intrusion detection*, Journal of Network and Computer Applications (2009).

Trust in complex actions

Julien Bourdon[1], Guillaume Feuillade[2], Andreas Herzig[2], and Emiliano Lorini[2]

1. Kyoto University, Department of Social Informatics, Japan
2. Université de Toulouse, CNRS, IRIT, France

Abstract. Current formal models of trust are limited since they only consider an agent's trust in the atomic action of another agent and therefore do not apply to trust in complex actions where the elements in the complex action are atomic actions of different agents. Our aim is to present a logical formalization of trust in complex actions, and to show that this formalization can be useful for the formal characterization of trust in composite services, where trust in a composed service is defined in a compositional way from trust in the components of that service.

1 Introduction

According to Castelfranchi and Falcone (C&F henceforth), the trust relation involves a truster i, a trustee j, an action a that is performed by j and a goal φ of i [5, 7]. They defined the predicate `Trust` as a goal together with a particular configuration of beliefs of the trustee. Precisely, i trusts j to do a in order to achieve φ if and only if:

1. i has the goal that φ and
2. i believes that:
   (a) j is capable to perform a,
   (b) j is willing to perform a,
   (c) j has the power to achieve φ by doing a.

C&F distinguish external from internal conditions in trust assessment: j's capability to perform a is an external condition, while j's willingness to perform a is an internal condition (being about the trustee's mental state). Finally, j's power to achieve φ by doing a relates internal and external conditions: if j performs a then φ will result. Observe that in the power condition the result is conditioned by the execution of a; therefore the power condition is independent from the capability condition. In particular, j may well have the power to achieve φ without being capable to perform a: for example, right now I have the power to lift a weight of 50kg, but I am not capable to do this because there is no such weight at hand.[1]

C&F did not investigate further how goals, capabilities, willingness and power have to be defined; their definition might therefore be called semi-formal. Recently Herzig,

[1] Together, capability to perform a and power to achieve φ by doing a amount to having a strategy to achieve φ. Similar modalities were studied in Coalition Logic CL [17], Alternating-time Temporal Logic ATL [1], and STIT theory [2]. However, these logics focus on game-theoretic situations where an agent has the power to achieve φ *whatever the other agents choose to do.* While this latter aspect will not be captured in our analysis here, we have shown in [12, 15] how it could be integrated into our logical framework.

Lorini et al. analysed these predicates in more detail in [13, 14, 16]. First of all they defined the predicate $\mathtt{Trust}_0$ as follows:

$$\mathtt{Trust}_0(i, j{:}a, \varphi) \overset{\text{def}}{=} \mathtt{Goal}(i, \varphi) \wedge \mathtt{Bel}_i(\mathtt{CExt}(i{:}a) \wedge \mathtt{CInt}(i{:}a) \wedge \mathtt{Res}(i{:}a, \varphi))$$

where $\mathtt{Goal}(i, \varphi)$ corresponds to item 1 and $\mathtt{CExt}(i{:}a)$, $\mathtt{CInt}(i{:}a)$ and $\mathtt{Res}(i{:}a, \varphi)$ respectively correspond to items 2a, 2b and 2c in C&F's definition (and $\mathtt{CExt}$ and $\mathtt{CInt}$ stand for the external and the internal conditions in trust assessment).[2]

They then defined the predicates $\mathtt{CExt}(j{:}a)$, $\mathtt{CInt}(j{:}a)$ and $\mathtt{Res}(j{:}a, \varphi)$ in terms of the concepts of belief, choice, action and time. They draw a distinction between occurrent trust —i's trust that j is going to perform a here and now— and dispositional trust: i trusts that j is going to perform a whenever suitable conditions obtain. The above definition is that of occurrent trust.

Both C&F and Herzig, Lorini et al. only considered trust in the atomic action of another agent and did not consider trust in complex actions where the elements in the complex action are atomic actions of different agents. Our aim in this article is to extend their definition to complex actions, and to show that a definition of trust in complex actions is extremely important in the context of composite services.

In the context of services architecture, some provider agents publish atomic services; however, when a client agent needs to implement a more complex business process, it must chain service calls, according to a specific workflow structure. Automating the service calls is called service composition: given the business process to implement, the control flow has to be computed in order to guarantee that the goal of the service caller is satisfied.

Since services are provided by agents, users may trust some agents for certain actions but not for other actions which they deem critical, usually depending on the nature of the information they have to send to this agent for the service action to perform the action.

In current literature, for example in [18], service selection for composition assumes the existence of a central authority guaranteeing the non-functional properties of the services. In practice, such an authority might not exist, for example in P2P networks [20], or may not itself be trustworthy.

To resolve the aforementioned problem, trust in complex action, supported by the introduction of beliefs and trust in the description of services, could be used. With a model for trust in the services world, one may express composition objectives as dynamic logic formulas with trust component. The service composition problem would then reduce to the satisfaction of such a formula. This method would ensure that the composition is correct and compatible with the beliefs of the user, thus ensuring a trustworthy sequence of service call for achieving the goal of the user.

The rest of the paper is organized as follows. In Section 2 we introduce a modal logic of belief, goal, time, and complex actions. In Section 3 we define the trust predicate for complex actions and study its constituents. In Section 4 we relate trust in complex

[2] They used a 4-ary predicate $\mathtt{Trust}(i, j, a, \varphi)$ instead of our ternary $\mathtt{Trust}_0(i, j{:}a, \varphi)$. Moreover, their trust definition was in terms of the predicates $\mathit{Capable}(j{:}a)$, $\mathit{Willing}(j{:}a)$ and $\mathit{Power}(j{:}a, \varphi)$ instead of our $\mathtt{CExt}(j{:}a)$, $\mathtt{CInt}(j{:}a)$ and $\mathtt{Res}(j{:}a, \varphi)$. We preferred our terms and notations because they better generalize to complex actions.

actions to trust in atomic actions, providing thus a way to construct trust in complex actions. In Section 5 we then discuss how this applies to service composition.

2 Background

We recall the logical framework of [13, 14, 16], that we extend towards complex actions.

2.1 A logical language with complex actions

Suppose given three countable sets: a set of propositional variables *Atm*, a set of agents *Agt* and a set of atomic actions *Act*. Complex formulas φ and complex actions α are defined by the following BNF:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathtt{Bel}_i\varphi \mid \mathtt{Ch}_i\varphi \mid \mathtt{Feasible}_\alpha\varphi \mid \mathtt{Happens}_\alpha\varphi \mid \mathtt{F}\varphi$$
$$\alpha ::= i{:}a \mid \alpha;\alpha \mid \alpha{+}\alpha \mid \varphi? \mid \alpha^*$$

where p ranges over *Atm*, i ranges over *Agt*, and a ranges over *Act*. $\mathtt{Bel}_i\varphi$ reads "i believes that φ"; $\mathtt{Ch}_i\varphi$ reads "i chooses that φ"; $\mathtt{Feasible}_\alpha\varphi$ reads "there is a possible execution of α after which φ is true"; $\mathtt{Happens}_\alpha\varphi$ reads "α happens, and φ is true afterwards"; and $\mathtt{F}\varphi$ reads "φ will eventually be true". Both $\mathtt{Feasible}_\alpha$ and $\mathtt{Happens}_\alpha$ are modal operators of the possible kind, and could be written $\langle\alpha\rangle$ as in dynamic logic. Note that they are different: $\mathtt{Feasible}_\alpha\top$ means that α is execut*able*, while $\mathtt{Happens}_\alpha\top$ means that α is execut*ed*.

Operators $\mathtt{Ch}_i$ are used to denote an agent's current chosen goals, that is, the goals that the agent has decided to pursue here and now. We do not consider how an agent's chosen goals originate through deliberation from more primitive motivational attitudes, called desires, and from moral attitudes, such as ideals and imperatives. Since the chosen goals of an agent result from the its deliberation, they must satisfy two fundamental rationality principles: chosen goals have to be consistent (i.e., a rational agent cannot decide to pursue an inconsistent state of affairs); chosen goals have to be compatible with the agent's beliefs (i.e., a rational agent cannot decide to pursue something that it believes to be impossible). These two principles will be formally expressed in Section 2.2.

Remark 1. In [13, 14, 16] $\mathtt{Happens}_{i:a}$ was noted $Does_{i:a}$ and read "a is going to be performed by i". We preferred $\mathtt{Happens}$ in order to allow for complex actions such as $i{:}a;\, j{:}b$ that are performed by more than one agent.

The atomic action $i{:}a$ reads "i performs a"; the complex action $\alpha_1;\alpha_2$ reads "do α_1 and then α_2"; $\alpha_1{+}\alpha_2$ reads "choose nondeterministically between α_1 and α_2", where the choice is understood to be up to the environment (i.e., the other agents and nature), and not up to the agents performing α_1 and α_2; $\varphi?$ reads "if φ is true then continue, else fail"; and finally, α^* reads "do α an arbitrary number of times".

We define $\mathtt{After}_\alpha\varphi$ to be an abbreviation of $\neg\mathtt{Feasible}_\alpha\neg\varphi$, which therefore has to be read "if the execution of α is possible then φ holds afterwards". Moreover, the following standard program constructions are defined as follows:

$$\begin{aligned}
\texttt{skip} &\overset{\text{def}}{=} \top? \\
\texttt{fail} &\overset{\text{def}}{=} \bot? \\
\texttt{if}\,\varphi\,\texttt{then}\,\alpha_1\,\texttt{else}\,\alpha_2 &\overset{\text{def}}{=} (\varphi?;\alpha_1)+(\neg\varphi?;\alpha_2) \\
\texttt{while}\,\varphi\,\texttt{do}\,\alpha &\overset{\text{def}}{=} (\varphi?;\alpha)^*;\neg\varphi?
\end{aligned}$$

In our application the actions seem never to be joined actions (which here would be something like translating a text together). For that reason we define parallel composition as interleaving, i.e. $\alpha\|\beta \overset{\text{def}}{=} (\alpha;\beta)+(\beta;\alpha)$. In this way we can avoid introducing $\|$ as a primitive.

2.2 Semantics

We take over the semantics of [14] and extend it to complex actions, whose semantics we take over from Propositional Dynamic Logic PDL. We call the resulting logic $\mathcal{L}$. The semantics of $\mathcal{L}$ is in terms of a class of frames that has to satisfy several constraints.

Frames A *frame* is a tuple $M = \langle W, A, B, C, D\rangle$ that is defined as follows.

- W is a nonempty set of possible worlds or states.
- $A : Agt \times Act \longrightarrow 2^{W\times W}$ maps every agent i and action a to a relation $A_{i:a}$ between possible worlds in W.
- $B : Agt \longrightarrow 2^{W\times W}$ maps every agent i to a serial, transitive and Euclidean[3] relation B_i between possible worlds in W.
- $C : Agt \longrightarrow 2^{W\times W}$ maps every agent i to a serial relation C_i between possible worlds in W.
- $D : Agt \times Act \longrightarrow 2^{W\times W}$ maps every agent i and action α to a deterministic relation (alias a partial function) $D_{i:a}$ between possible worlds in W.[4]

It is convenient to view relations on W as functions from W to 2^W; therefore we write $D_{i:a}(w)$ for the set $\{w' \mid (w, w') \in D_{i:a}\}$, etc.

When $w' \in A_{i:a}(w)$ then *if* at w agent i performs α then this *might* result in w'. $B_i(w)$ is the set of worlds that are compatible with agent i's beliefs at w; the conditions of seriality, transitivity and Euclideanity are those of the standard logic of belief KD45. $C_i(w)$ is the set of worlds that are compatible with agent i's choices at w; seriality corresponds to consistency of choices, which is the only condition that is generally imposed on choices. $D_{i:a}(w)$ is the set of worlds w' that can be reached from w through the occurrence of agent i's action a. If $(w, w') \in D_{i:a}$ then w' is the unique actual *successor* world of w, that will be reached from w through the occurrence of agent i's action a at w: at w agent i performs an action a, resulting in the next state w'. (We might also say that $D_{i:a}$ is a partial function.) If $D_{i:a}(w) \neq \emptyset$ then we say that $D_{i:a}$ is defined at w.

[3] A relation B_i on W is Euclidean if and only if, if $(w, w') \in B_i$ and $(w, w'') \in B_i$ then $(w', w'') \in B_i$.

[4] A relation $D_{i:a}$ is deterministic iff, if $(w, w') \in D_{i:a}$ and $(w, w'') \in D_{i:a}$ then $w' = w''$.

Constraints on frames Frames will have to satisfy some constraints in order to be legal $\mathcal{L}$-frames. For every $i, j \in Agt$, $\alpha, \beta \in Act$ and $w \in W$ we suppose:

C1 if $D_{i:a}$ and $D_{j:b}$ are defined at w then $D_{i:a}(w) = D_{j:b}(w)$.

Constraint **C1** says that if w' is the *next* world of w which is reached from w through the occurrence of agent i's action α and w'' is also the *next* world of w which is reached from w through the occurrence of agent j's action β, then w' and w'' denote the same world. Indeed, we suppose that one agent acts at a time, and that every world can only have one *next* world. Note that **C1** implies determinism of every $D_{i:a}$ (so we might have omitted that from the above constraints on D).

Therefore, when $w' \in A_{i:a}(w)$ but $D_{i:a}(w) = \emptyset$ then at w agent i does not perform α, but if it did so it might have produced another outcome world w'.

Moreover, for every $i \in Agt$, $\alpha \in Act$ we suppose:

C2 $D_{i:a} \subseteq A_{i:a}$.

The constraint **C2** says that if w' is the *next* world of w which is reached from w through the occurrence of agent i's action α, then w' must be a world which is *reachable* from w through the occurrence of agent i's action α.

The next constraint **C3** links the agents' choices with what they do: if a is executable and i chooses to do a then a is going to happen.

C3 if $A_{i:a}$ is defined at w and $D_{i:a}$ is defined at w' for all $w' \in C_i(w)$ then $D_{i:a}$ is defined at w.

The following semantic constraint **C4** is also about the relationship between an agent i's choices (i.e., chosen worlds) and the actions performed by i. For every $i \in Agt$, $\alpha \in Act$ and $w \in W$, we suppose that:

C4 if $w' \in C_i(w)$ and $D_{i:a}$ is defined at w, then $D_{i:a}$ is defined at w'.

In other words, if it is not the case that i performs a in all of i's chosen worlds then i is not going to perform a.

The next constraint relates worlds that are compatible with agent i's beliefs and worlds that are compatible with i's chosen goals: as motivated in the beginning of Section 2.1, they should not be disjoint. For every $i \in Agt$ and $w \in W$:

C5 $C_i(w) \cap B_i(w) \neq \emptyset$.

The next constraint on $\mathcal{L}$-frames is one of introspection w.r.t. choices. For every $i \in Agt$ and $w \in W$:

C6 if $w' \in B_i(w)$ then $C_i(w) = C_i(w')$.

The next two constraints on $\mathcal{L}$-frames are what is called 'no learning' and 'no forgetting' for beliefs in the literature [6]. For every $i, j \in Agt$, $a \in Act$ and $w \in W$:

C7 if $(w, v) \in A_{j:a} \circ B_i$ and there is u such that $(w, u) \in B_i \circ A_{j:a}$ then $(w, v) \in B_i \circ A_{j:a}$
C8 if $(w, v) \in B_i \circ A_{j:a}$ and there is u such that $(w, u) \in A_{j:a}$ then $(w, v) \in A_{j:a} \circ B_i$,

where $\circ$ is the standard composition operator between two binary relations. Thus, we suppose that events are always uninformative, in the sense: *i should not forget* anything about the particular effects of j's action a that starts at a given world w, and *i should not learn* anything new due to the occurrence of j's action a that starts at a given world w (except the occurrence of that very action). In other words, what an agent i believes at a world v after the occurrence of j's action a, only depends on what i believed at the previous world w and on the action which has occurred and which was responsible for the transition from w to v. Note that the 'no forgetting' and 'no learning' constraints rely on an additional assumption that actions are public: it is supposed that j's action a occurs if and only if every agent is informed of this fact.

We have similar principles of no learning and no forgetting for the relations $D_{i:a}$. For every $i, j \in Agt$, $a \in Act$ and $w \in W$:

C9 if $(w, v) \in D_{j:a} \circ B_i$ and there is u such that $(w, u) \in B_i \circ A_{j:a}$ then $(w, v) \in B_i \circ D_{j:a}$;
C10 if $(w, v) \in B_i \circ D_{j:a}$ and there is u such that $(w, u) \in D_{j:a}$ then $(w, v) \in D_{j:a} \circ B_i$.

Models and truth conditions A *model* is a tuple $M = \langle W, A, B, C, D, V\rangle$ where the tuple $\langle W, A, B, C, D\rangle$ is a frame and $V : Atm \to 2^W$ is a valuation.

Formulas and events are interpreted according to the following clauses.

$$\begin{aligned}
R_{\alpha;\beta} &= R_\alpha \circ R_\beta \\
R_{\alpha+\beta} &= R_\alpha \cup R_\beta \\
R_{\varphi?} &= \{\langle v, v\rangle \mid v \in W \text{ and } M, v \models \varphi\} \\
R_{\alpha^*} &= (R_\alpha)^* \\
M, w \models p &\text{ iff } w \in V(p) \\
M, w \models \mathtt{Bel}_i\varphi &\text{ iff } M, w' \models \varphi \text{ for every } w' \in B_i(w) \\
M, w \models \mathtt{Ch}_i\varphi &\text{ iff } M, w' \models \varphi \text{ for every } w' \in C_i(w) \\
M, w \models \mathtt{Feasible}_\alpha\varphi &\text{ iff } M, w' \models \varphi \text{ for some } w' \in A_\alpha(w) \\
M, w \models \mathtt{Happens}_\alpha\varphi &\text{ iff } M, w' \models \varphi \text{ for some } w' \in D_\alpha(w) \\
M, w \models \mathtt{F}\varphi &\text{ iff } M, w' \models \varphi \text{ for some } w' \text{ such that } w(\textstyle\bigcup_{a\in Act} D_a)^* w'
\end{aligned}$$

The clauses for the Boolean operators are as usual. The last clause is based on the hypothesis that time flow is determined by the actions that are performed (where the D_a and $\bigcup_{a\in Act} D_a$ are understood as relations). $\bigcup_{a\in Act} D_a(w)$ is the set of worlds w' that are in the future of w: w' can be attained from w by some D-chain, i.e. by some sequence of actions.

2.3 Some useful validities

We now state some validities of our logic that will be useful later.[5]

Proposition 1. *The following formulas are valid:*

[5] We do not give a completeness result: there is such a result (albeit for a simpler language) in [14], which should be extended in order to account for complex actions; in particular the Kleene star "$*$" requires a fixpoint axiom and a least fixpoint axiom, which makes that the completeness proof is not straightforward.

1. $\texttt{After}_{\varphi?}\psi \leftrightarrow (\varphi \rightarrow \psi)$
2. $\texttt{Happens}_{\varphi?}\psi \leftrightarrow (\varphi \wedge \psi)$
3. $\texttt{Happens}_{\alpha;\beta}\varphi \leftrightarrow (\texttt{Happens}_{\alpha}\top \wedge \neg\texttt{Happens}_{\alpha}\neg\texttt{Happens}_{\beta}\varphi)$
4. $\texttt{Happens}_{\alpha+\beta}\varphi \leftrightarrow (\texttt{Happens}_{\alpha}\varphi \vee \texttt{Happens}_{\beta}\varphi)$
5. $\texttt{Happens}_{\alpha^*}\varphi \leftrightarrow (\varphi \wedge \texttt{Happens}_{\alpha}\texttt{Happens}_{\alpha^*}\varphi)$
6. $(\texttt{F}\neg\varphi \wedge \texttt{After}_{\alpha^*}(\varphi \rightarrow \texttt{Happens}_{\alpha}\top)) \rightarrow \texttt{Happens}_{\texttt{while}\ \varphi\ \texttt{do}\ \alpha}\top$
7. $\texttt{Happens}_{\alpha}\varphi \rightarrow \texttt{Feasible}_{\alpha}\varphi$
8. $\texttt{Feasible}_{\alpha}\varphi \rightarrow \texttt{F}\varphi$
9. $(\texttt{Happens}_{\alpha}\varphi \wedge \texttt{After}_{\alpha}\psi) \rightarrow \texttt{Happens}_{\alpha}(\varphi \wedge \psi)$
10. $\neg(\texttt{Ch}_i\varphi \wedge \texttt{Bel}_i\varphi)$
11. $(\texttt{Feasible}_{i:a}\varphi \wedge \texttt{Ch}_i\texttt{Happens}_{i:a}\top) \rightarrow \texttt{Happens}_{i:a}\top$
12. $(\neg\texttt{Bel}_i\neg\texttt{Feasible}_{\alpha}\top \wedge \texttt{Bel}_i\texttt{After}_{\alpha}\varphi) \rightarrow \texttt{After}_{\alpha}\texttt{Bel}_i\varphi$
13. $(\texttt{Feasible}_{\alpha}\top \wedge \texttt{After}_{\alpha}\texttt{Bel}_i\varphi) \rightarrow \texttt{Bel}_i\texttt{After}_{\alpha}\varphi$
14. $(\neg\texttt{Bel}_i\neg\texttt{Happens}_{\alpha}\top \wedge \texttt{Bel}_i\neg\texttt{Happens}_{\alpha}\neg\varphi) \rightarrow \neg\texttt{Happens}_{\alpha}\neg\texttt{Bel}_i\varphi$
15. $(\texttt{Happens}_{\alpha}\top \wedge \neg\texttt{Happens}_{\alpha}\neg\texttt{Bel}_i\varphi) \rightarrow \texttt{Bel}_i\neg\texttt{Happens}_{\alpha}\neg\varphi$

Formula 11 is a principle of intentional action **IntAct**. The last four are principles of no forgetting (**NF**, alias perfect recall) and no learning (**NL**, alias no miracles) for beliefs. Similar principles have been studied in [8, 19, 11].

3 Trust about complex actions

We now generalize the definition of (occurrent) trust about atomic actions of [13, 14, 16] to trust about complex actions and study its constituents. Among all possible complex actions we here only consider deterministic actions [9]: actions built with "`skip`", "`fail`", ";", "`if` φ `then` α_1 `else` α_2", and "`while` φ `do` α". Their BNF is:

$$\alpha ::= i{:}a \mid \texttt{skip} \mid \texttt{fail} \mid \alpha;\alpha \mid \texttt{if}\ \varphi\ \texttt{then}\ \alpha\ \texttt{else}\ \alpha \mid \varphi? \mid \texttt{while}\ \varphi\ \texttt{do}\ \alpha$$

Tests $\varphi?$ can be defined as `if` φ `then skip else fail`. In our analysis of trust in complex actions we do not consider the other program operators of PDL, viz. nondeterministic composition and iteration.

Let us first recall the definition of the original trust predicate in [13, 14, 16]. There, the goal condition $\texttt{Goal}(i,\varphi)$ was defined as $\texttt{Ch}_i\texttt{F}\varphi$, i.e. as i's choice of futures where φ holds. The external condition $\texttt{CExt}(j{:}a)$ was defined as $\texttt{Feasible}_{j:a}\top$ ($j{:}a$ is executable), and the internal condition $\texttt{CInt}(j{:}a)$ as $\texttt{Ch}_j\texttt{Happens}_{j:a}\top$ (j chooses that $j{:}a$ is going to occur). Finally, the power condition $\texttt{Res}(j{:}a,\varphi)$ was defined as $\texttt{After}_{j:a}\varphi$ (φ will hold immediately after every possible performance of $j{:}a$).

It turns out that our move from trust in atomic actions to trust in complex actions requires some adjustments.

3.1 Definition of trust

First of all, here is our official definition of trust in a complex action:

$$\texttt{Trust}(i,\alpha,\varphi) \overset{\text{def}}{=} \texttt{Goal}(i,\varphi) \wedge \texttt{Bel}_i(\texttt{CExt}(\alpha) \wedge \texttt{CInt}(\alpha) \wedge \texttt{Res}(\alpha,\varphi))$$

where i is an agent, α is a deterministic action, and φ is a formula. As before, $\mathtt{CExt}$ and $\mathtt{CInt}$ stand for the external and the internal conditions in trust assessment; they will be defined in the sequel. We have thus simply replaced the atomic actions in our definition of Section 1 by complex actions.

Observe that trust in atomic actions involved a single trustee j. Here we have to account for trust in complex actions that may be performed by several agents; we therefore consider trust in a group of agents.

Note also that before, the trustee j —which here would be a set of agents J— appeared explicitly in the definition of the predicate $\mathtt{Trust}$. However, one may consider that J is implicitly already there: it is the set of agents occurring in α. Therefore the agent argument need not appear as a separate argument in the definition.

It remains to explain the predicates on the right hand side of the definition of trust.

3.2 Defining the ingredients of trust

We now reduce the predicates on the right hand side of the definition of trust.

Goal The definition of the $\mathtt{Goal}$ predicate transfers straightforwardly because no action occurs in it:

$$\mathtt{Goal}(i, \varphi) \overset{\text{def}}{=} \mathtt{Ch}_i \mathtt{F}\varphi$$

So it remains to define $\mathtt{CExt}$, $\mathtt{CInt}$ and $\mathtt{Res}$.

Result The original power condition $\mathtt{Bel}_i\mathtt{After}_{j:a}\varphi$ stipulated that i believes φ immediately results from j's performance of atomic action a. However, consider i's trust in j_1 and j_2 to perform the sequence of actions $j_1{:}a_1; j_2{:}a_2$ in order to achieve i's goal φ. With respect to which goal should i trust j_1? The truster i typically does not bother about the direct effect of j_1's action a_1 and is only interested in the overall effect φ of the complex action $j_1{:}a_1; j_2{:}a_2$. In other words, we have to account for the case where φ is not achieved immediately, but only at some time point in the future. We therefore redefine

$$\mathtt{Res}(\alpha, \varphi) \overset{\text{def}}{=} \mathtt{After}_\alpha \mathtt{F}\varphi$$

Under the other definitions to come, the original $\mathtt{Trust}_0(i, j{:}a, \varphi)$ will be equivalent to our $\mathtt{Trust}(i, j{:}a, \mathtt{F}\varphi)$.

External and internal condition Up to now, all our definitions were directly in terms of well-defined formulas of our logic. Things are not as simple for the external condition $\mathtt{CExt}$ and for the internal condition $\mathtt{CInt}$.

In [13, 14, 16], using axiom **IntAct** it was proved that

$$(\mathtt{CExt}(i{:}a) \wedge \mathtt{CInt}(i{:}a)) \rightarrow \mathtt{Happens}(i{:}a)$$

is valid. That is, if both the external condition and the internal condition for the execution of action a by agent i obtain —i.e., i is capable to perform action a and is willing (has the intention) to perform a— then i performs a. We would like to keep this principle of intentional action, and therefore need a definition of the $\mathtt{CExt}$ and $\mathtt{CInt}$ predicates validating

$$(\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)) \rightarrow \mathtt{Happens}(\alpha)$$

In particular, we will have to include a condition guaranteeing that while-loops are exited (because $\mathtt{Happens}_{\mathtt{while}\ \psi\ \mathtt{do}\ \alpha}\top$ implies that $\mathtt{F}\neg\psi$).

As to the external condition, $\mathtt{CExt}(\alpha)$ means that the complex action α is executable *whatever the other agents and nature choose to do*. This means that the preconditions of α must obtain at every step of every execution of α. It follows that while $\mathtt{CExt}(\alpha)$ implies $\mathtt{Feasible}_\alpha\top$, it should not be equivalent to it. For example, the complex action $(i{:}a+i{:}a');i{:}b$ cannot be said to be executable (in the above sense) when just $\mathtt{Feasible}_{i:a+i:a';i:b}\top$ holds. Indeed, a situation where $\mathtt{Feasible}_{i:a'}\mathtt{After}_{i:b}\top$ is compatible with the latter formula, and if nature chooses $i{:}a'$ when executing the nondeterministic $i{:}a+i{:}a'$ then it cannot be said that $\mathtt{Feasible}_{i:a+i:a';i:b}\top$ is executable.

Given these considerations we recursively define $\mathtt{CExt}(\alpha)$ as follows:

$$\begin{aligned}
\mathtt{CExt}(i{:}a) &\overset{\text{def}}{=} \mathtt{Feasible}_{i:a}\top \\
\mathtt{CExt}(\mathtt{skip}) &\overset{\text{def}}{=} \top \\
\mathtt{CExt}(\mathtt{fail}) &\overset{\text{def}}{=} \bot \\
\mathtt{CExt}(\alpha;\beta) &\overset{\text{def}}{=} \mathtt{CExt}(\alpha) \wedge \mathtt{After}_\alpha\mathtt{CExt}(\beta) \\
\mathtt{CExt}(\mathtt{if}\ \varphi\ \mathtt{then}\ \alpha_1\ \mathtt{else}\ \alpha_2) &\overset{\text{def}}{=} (\varphi \wedge \mathtt{CExt}(\alpha_1)) \vee (\neg\varphi \wedge \mathtt{CExt}(\alpha_2)) \\
\mathtt{CExt}(\mathtt{while}\ \psi\ \mathtt{do}\ \alpha) &\overset{\text{def}}{=} \mathtt{F}\neg\psi \wedge \mathtt{After}_{(\psi?;\alpha)^*;\psi?}\mathtt{CExt}(\alpha)
\end{aligned}$$

It is the clause for ";" that makes that $\mathtt{CExt}(\alpha)$ stronger than $\mathtt{Feasible}_\alpha\top$.

As to the (internal) willingness condition, it is tempting to define $\mathtt{CInt}(\alpha)$ as

$$\textstyle\bigwedge_{j\in Agt(\alpha)} \mathtt{Ch}_j\mathtt{Happens}_\alpha\top,$$

where $Agt(\alpha)$ is the set of agent names occurring in α: every agent involved in the complex action α chooses that α happens. However, this would be too strong. Indeed, consider the scenario where $j_1{:}a_1$ is j_1's action of requesting j_2 to do a_2, and where j_2 initially prefers not to be asked by j_1, i.e. $\mathtt{Ch}_{j_2}\neg\mathtt{Happens}_{j_1:a_1}\bot$, but intends to perform $j_2{:}a_2$ after j_1's request. In symbols, we have a situation where $\mathtt{Happens}_{j_1:a_1;j_2:a_2}\top$ and $\neg\mathtt{Ch}_{j_2}\mathtt{Happens}_{j_1:a_1;j_2:a_2}\top$ is true.

Such considerations lead to the following recursive definition of the predicate $\mathtt{CInt}$.

$$\begin{aligned}
\mathtt{CInt}(i{:}a) &\overset{\text{def}}{=} \mathtt{Ch}_i\mathtt{Happens}_{i:a}\top \\
\mathtt{CInt}(\mathtt{fail}) &\overset{\text{def}}{=} \top \\
\mathtt{CInt}(\mathtt{skip}) &\overset{\text{def}}{=} \top \\
\mathtt{CInt}(\alpha;\beta) &\overset{\text{def}}{=} \mathtt{CInt}(\alpha) \wedge \mathtt{After}_\alpha\mathtt{CInt}(\beta) \\
\mathtt{CInt}(\mathtt{if}\ \varphi\ \mathtt{then}\ \alpha_1\ \mathtt{else}\ \alpha_2) &\overset{\text{def}}{=} (\varphi \wedge \mathtt{CInt}(\alpha_1)) \vee \\
&\phantom{\overset{\text{def}}{=}}\ (\neg\varphi \wedge \mathtt{CInt}(\alpha_2)) \\
\mathtt{CInt}(\mathtt{while}\ \psi\ \mathtt{do}\ \alpha) &\overset{\text{def}}{=} \mathtt{After}_{(\psi?;\alpha)^*;\psi?}\mathtt{CInt}(\alpha)
\end{aligned}$$

3.3 A principle of intentional action for complex actions

We are now going to relate the predicates CExt and CInt with the modal operator Happens. We prove that when α is complex then one half of the axiom **IntAct** remains valid.

Proposition 2. *The formula* $\mathtt{CExt}(\alpha) \to \mathtt{Feasible}_\alpha\top$ *is valid.*

Proof. The proof is by induction on the structure of α.

As mentioned above, the other direction $\mathtt{Feasible}_\alpha\top \to \mathtt{CExt}(\alpha)$ is guaranteed to be valid only when α is atomic.

Proposition 3. *The formula* $(\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)) \to \mathtt{Happens}_\alpha\top$ *is valid.*

Proof. We use induction on the structure of α. The base cases are ensured by the axiom **IntAct** and by Proposition 1. For the induction step we have:

$$\begin{array}{ll}
\mathtt{CExt}(\alpha;\beta) \wedge \mathtt{CInt}(\alpha;\beta) & \\
\leftrightarrow \mathtt{CExt}(\alpha) \wedge \mathtt{After}_\alpha\mathtt{CExt}(\beta) \wedge \mathtt{CInt}(\alpha) \wedge \mathtt{After}_\alpha\mathtt{CInt}(\beta) & \\
\to \mathtt{Happens}_\alpha\top \wedge \mathtt{After}_\alpha\mathtt{Happens}_\beta\top & \text{(by I.H.)} \\
\to \mathtt{Happens}_{\alpha;\beta}\top & \text{(by Prop. 1)}
\end{array}$$

$$\begin{array}{ll}
\mathtt{CExt}(\mathtt{if}\ \psi\ \mathtt{then}\ \alpha\ \mathtt{else}\ \beta) \wedge \mathtt{CInt}(\mathtt{if}\ \psi\ \mathtt{then}\ \alpha\ \mathtt{else}\ \beta) & \\
\leftrightarrow (\psi \to (\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)) \wedge (\neg\psi \to (\mathtt{CExt}(\beta) \wedge \mathtt{CInt}(\beta)) & \\
\to (\psi \to \mathtt{Happens}_\alpha\top) \wedge (\neg\psi \to \mathtt{Happens}_\beta\top) & \text{(by I.H.)} \\
\to \mathtt{Happens}_{\mathtt{if}\ \psi\ \mathtt{then}\ \alpha\ \mathtt{else}\ \beta}\top & \text{(by Prop. 1)}
\end{array}$$

$$\begin{array}{ll}
\mathtt{CExt}(\mathtt{while}\ \psi\ \mathtt{do}\ \alpha) \wedge \mathtt{CInt}(\mathtt{while}\ \psi\ \mathtt{do}\ \alpha) & \\
\leftrightarrow \mathtt{F}\neg\psi \wedge \mathtt{After}_{(\psi?;\alpha)^*;\psi?}(\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)) & \\
\to \mathtt{F}\neg\psi \wedge \mathtt{After}_{(\psi?;\alpha)^*}(\psi \to \mathtt{Happens}_\alpha\top) & \text{(by I.H.)} \\
\to \mathtt{F}\neg\psi \wedge \mathtt{After}_{(\psi?;\alpha)^*}(\psi \to \mathtt{Happens}_{\psi?;\alpha}\top) & \\
\to \mathtt{Happens}_{\mathtt{while}\ \psi\ \mathtt{do}\ \alpha}\top & \text{(by Prop. 1)}
\end{array}$$

As said above, the other direction of Proposition 3

$$\mathtt{Happens}_{i_1:a_1;i_2:a_2}\top \to (\mathtt{CExt}(i_1{:}a_1; i_2{:}a_2) \wedge \mathtt{CInt}(i_1{:}a_1; i_2{:}a_2))$$

is invalid because i_1's performance of a_1 may cause i_2's performance of a_2.

We finally observe that when the truster's goal is $\top$ then trust in α amounts to the conjunction of external and internal condition.

Proposition 4. *The formula* $\mathtt{Trust}(i, \alpha, \top) \leftrightarrow \mathtt{Bel}_i(\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha))$ *is valid.*

4 Properties of trust

In this section we state the properties of trust in complex actions, alias workflow constructs.

First of all and as announced in Section 3.2 we observe that our and the original definition coincide for atomic actions, except that we have relaxed the result condition:

for us it suffices that the result φ obtains at some point in the future, and not immediately after the action. We therefore have $\mathtt{Trust}(i, j{:}a, \varphi) \leftrightarrow \mathtt{Trust}_0(i, j{:}a, \mathtt{F}\varphi)$.

For complex actions we are going to have reductions in terms of equivalences for the cases of `skip`, `fail`, if-then-else conditionals and while loops. For trust in sequential compositions we only give a sufficient condition. We only give some of the proofs.

4.1 Atomic actions

Theorem 1. *The formulas* $\mathtt{Trust}(i, \mathtt{fail}, \varphi) \leftrightarrow \bot$ *and*

$$\mathtt{Trust}(i, \mathtt{skip}, \varphi) \leftrightarrow (\mathtt{Goal}(i, \varphi) \wedge \mathtt{Bel}_i \mathtt{F}\varphi)$$

are valid.

4.2 Sequential composition

Our first theorem allows to construct trust in a sequence $\alpha;\beta$ from trust in α and trust in β.

Theorem 2. *The formula*

$$(\mathtt{Trust}(i, \alpha, \varphi) \wedge \mathtt{Bel}_i\mathtt{After}_\alpha\mathtt{Trust}(i, \beta, \varphi)) \rightarrow \mathtt{Trust}(i, (\alpha;\beta), \varphi)$$

is valid.

Proof. We have:

$$\begin{aligned}
& \mathtt{Trust}(i, \alpha, \varphi) \wedge \mathtt{Bel}_i\mathtt{After}_\alpha\mathtt{Trust}(i, \beta, \varphi) \\
\rightarrow\ & \mathtt{Goal}(i, \varphi) \wedge \mathtt{Bel}_i(\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)) \wedge \\
& \qquad \mathtt{Bel}_i\mathtt{After}_\alpha\mathtt{Bel}_i(\mathtt{CExt}(\beta) \wedge \mathtt{CInt}(\beta) \wedge \mathtt{After}_\beta\mathtt{F}\varphi) \\
\rightarrow\ & \mathtt{Goal}(i, \varphi) \wedge \mathtt{Bel}_i(\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)) \wedge \\
& \qquad \mathtt{Bel}_i(\mathtt{After}_\alpha\bot \vee \mathtt{Bel}_i\mathtt{After}_\alpha(\mathtt{CExt}(\beta) \wedge \mathtt{CInt}(\beta) \wedge \mathtt{After}_\beta\mathtt{F}\varphi)) \\
& \qquad \text{(by } \mathbf{NL} \text{ of Prop. 1)} \\
\rightarrow\ & \mathtt{Goal}(i, \varphi) \wedge \mathtt{Bel}_i(\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)) \wedge \\
& \qquad \mathtt{Bel}_i\mathtt{Bel}_i\mathtt{After}_\alpha(\mathtt{CExt}(\beta) \wedge \mathtt{CInt}(\beta) \wedge \mathtt{After}_\beta\mathtt{F}\varphi) \\
\rightarrow\ & \mathtt{Goal}(i, \varphi) \wedge \mathtt{Bel}_i(\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)) \wedge \mathtt{Bel}_i\mathtt{After}_\alpha(\mathtt{CExt}(\beta) \wedge \mathtt{CInt}(\beta) \wedge \mathtt{After}_\beta\mathtt{F}\varphi) \\
\leftrightarrow\ & \mathtt{Goal}(i, \varphi) \wedge \mathtt{Bel}_i(\mathtt{CExt}(\alpha) \wedge \mathtt{After}_\alpha\mathtt{CExt}(\beta)) \wedge \\
& \qquad \mathtt{Bel}_i(\mathtt{CInt}(\alpha) \wedge \mathtt{After}_\alpha\mathtt{CInt}(\beta)) \wedge \mathtt{Bel}_i\mathtt{After}_\alpha\mathtt{After}_\beta\mathtt{F}\varphi \\
\leftrightarrow\ & \mathtt{Goal}(i, \varphi) \wedge \mathtt{Bel}_i(\mathtt{CExt}(\alpha;\beta) \wedge \mathtt{CInt}(\alpha;\beta) \wedge \mathtt{After}_{\alpha;\beta}\mathtt{F}\varphi) \\
=\ & \mathtt{Trust}(i, (\alpha;\beta), \varphi)
\end{aligned}$$

The next two theorems are about the consequences of trust in a sequence of actions.

Theorem 3. *The formula*

$$\mathtt{Trust}(i, (\alpha;\beta), \varphi) \rightarrow \mathtt{Trust}(i, \alpha, \varphi)$$

is valid.

Proof. We have:

$$\begin{aligned}
&\mathtt{Trust}(i,(\alpha;\beta),\varphi)\\
&\leftrightarrow \mathtt{Goal}(i,\varphi) \wedge \mathtt{Bel}_i(\mathtt{CExt}(\alpha) \wedge \mathtt{After}_\alpha\mathtt{CExt}(\beta))\wedge\\
&\qquad\qquad \mathtt{Bel}_i(\mathtt{CInt}(\alpha) \wedge \mathtt{After}_\alpha\mathtt{CInt}(\beta)) \wedge \mathtt{Bel}_i\mathtt{After}_{\alpha;\beta}\mathrm{F}\varphi\\
&\leftrightarrow \mathtt{Goal}(i,\varphi) \wedge \mathtt{Bel}_i(\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha))\wedge\\
&\qquad\qquad \mathtt{Bel}_i\mathtt{After}_\alpha(\mathtt{CExt}(\beta) \wedge \mathtt{CInt}(\beta) \wedge \mathtt{After}_\beta\mathrm{F}\varphi)\\
&\to \mathtt{Goal}(i,\varphi) \wedge \mathtt{Bel}_i(\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)) \wedge \mathtt{Bel}_i\mathtt{After}_\alpha(\mathtt{Happens}_\beta\top \wedge \mathtt{After}_\beta\mathrm{F}\varphi)\\
&\qquad\qquad \text{(by Prop. 3)}\\
&\to \mathtt{Goal}(i,\varphi) \wedge \mathtt{Bel}_i(\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)) \wedge \mathtt{Bel}_i\mathtt{After}_\alpha\mathtt{Happens}_\beta\mathrm{F}\varphi \quad \text{(by Prop. 1)}\\
&\leftrightarrow \mathtt{Goal}(i,\varphi) \wedge \mathtt{Bel}_i(\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)) \wedge \mathtt{Bel}_i\mathtt{After}_\alpha\mathrm{F}\varphi \quad \text{(by Prop. 1)}\\
&\leftrightarrow \mathtt{Trust}(i,\alpha,\varphi)
\end{aligned}$$

Our last theorem says that trust persists under the condition that the goal persists.

Theorem 4. *The formula*

$$\mathtt{Trust}(i,(\alpha;\beta),\varphi) \to \mathtt{After}_\alpha(\neg\mathtt{Goal}(i,\varphi) \vee \mathtt{Trust}(i,\beta,\varphi))$$

is valid.

Proof. First, observe that

$$\mathtt{Trust}(i,(\alpha;\beta),\varphi) \to \mathtt{Bel}_i\mathtt{Feasible}_\alpha\top \ (*)$$

is valid by Proposition 3 and Proposition 1. Now:

$$\begin{aligned}
&\mathtt{Trust}(i,(\alpha;\beta),\varphi)\\
&\to \mathtt{Bel}_i\mathtt{After}_\alpha(\mathtt{CExt}(\beta) \wedge \mathtt{CInt}(\beta) \wedge \mathtt{After}_\beta\mathrm{F}\varphi)\\
&\to \mathtt{Bel}_i\mathtt{After}_\alpha\bot \vee \mathtt{After}_\alpha\mathtt{Bel}_i(\mathtt{CExt}(\beta) \wedge \mathtt{CInt}(\beta) \wedge \mathtt{After}_\beta\mathrm{F}\varphi) \quad \text{(by } \mathbf{NF} \text{ of Prop. 1)}\\
&\to \mathtt{After}_\alpha\mathtt{Bel}_i(\mathtt{CExt}(\beta) \wedge \mathtt{CInt}(\beta) \wedge \mathtt{After}_\beta\mathrm{F}\varphi) \quad \text{by } (*)
\end{aligned}$$

4.3 If-then-else

Theorem 5. *The formula*

$$\begin{aligned}
\mathtt{Trust}(i,\mathtt{if}\ \psi\ \mathtt{then}\ \alpha\ \mathtt{else}\ \beta,\varphi) \leftrightarrow \mathtt{Bel}_i(\ &(\psi \to \mathtt{Trust}(i,\alpha,\varphi))\wedge\\
&(\neg\psi \to \mathtt{Trust}(i,\beta,\varphi))\ )
\end{aligned}$$

is valid.

Proof. We have:

$$\begin{aligned}
&\mathtt{Trust}(i,\mathtt{if}\ \psi\ \mathtt{then}\ \alpha\ \mathtt{else}\ \beta,\varphi)\\
&\leftrightarrow \mathtt{Goal}(i,\varphi)\wedge\\
&\quad \mathtt{Bel}_i((\psi \to \mathtt{CExt}(\alpha)) \wedge (\neg\psi \to \mathtt{CExt}(\beta)))\wedge\\
&\quad \mathtt{Bel}_i((\psi \to \mathtt{CInt}(\alpha)) \wedge (\neg\psi \to \mathtt{CInt}(\beta))))\wedge\\
&\quad \mathtt{Bel}_i((\psi \to \mathtt{After}_\alpha\mathrm{F}\varphi) \wedge (\neg\psi \to \mathtt{After}_\beta\mathrm{F}\varphi))\\
&\leftrightarrow \mathtt{Bel}_i\mathtt{Goal}(i,\varphi)\wedge\\
&\quad \mathtt{Bel}_i(\psi \to (\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha) \wedge \mathtt{After}_\alpha\mathrm{F}\varphi))\wedge\\
&\quad \mathtt{Bel}_i(\neg\psi \to (\mathtt{CExt}(\beta) \wedge \mathtt{CInt}(\beta) \wedge \mathtt{After}_\beta\mathrm{F}\varphi))\\
&\leftrightarrow \mathtt{Bel}_i((\psi \to \mathtt{Trust}(i,\alpha,\varphi)) \wedge (\neg\psi \to \mathtt{Trust}(i,\beta,\varphi)))
\end{aligned}$$

4.4 While

Theorem 6. *The formula*

$$\begin{aligned}\mathtt{Trust}(i,(\mathtt{while}\ \psi\ \mathsf{do}\ \alpha),\varphi) \leftrightarrow (\ &\mathtt{Bel}_i\mathtt{After}_{(\psi?;\alpha)^*}(\psi \rightarrow (\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)))\wedge\\ &\mathtt{Goal}(i,\varphi) \wedge \mathtt{Bel}_i\mathtt{After}_{\mathtt{while}\ \psi\ \mathsf{do}\ \alpha}\mathrm{F}\varphi \wedge \mathtt{Bel}_i\mathrm{F}\neg\psi\ )\end{aligned}$$

is valid.

Proof. We have:

$$\begin{aligned}\mathtt{Trust}(i,(\mathtt{while}\ \psi\ \mathsf{do}\ \alpha),\varphi) \leftrightarrow\ &\mathtt{Goal}(i,\varphi) \wedge \mathtt{Bel}_i\mathtt{After}_{\mathtt{while}\ \psi\ \mathsf{do}\ \alpha}\mathrm{F}\varphi\wedge\\ &\mathtt{Bel}_i(\mathtt{CExt}(\mathtt{while}\ \psi\ \mathsf{do}\ \alpha) \wedge \mathtt{CInt}(\mathtt{while}\ \psi\ \mathsf{do}\ \alpha))\\ \leftrightarrow\ &\mathtt{Goal}(i,\varphi) \wedge \mathtt{Bel}_i\mathtt{After}_{\mathtt{while}\ \psi\ \mathsf{do}\ \alpha}\mathrm{F}\varphi \wedge \mathtt{Bel}_i\mathrm{F}\neg\psi\wedge\\ &\mathtt{Bel}_i\mathtt{After}_{(\psi?;\alpha)^*}(\psi \rightarrow (\mathtt{CExt}(\alpha) \wedge \mathtt{CInt}(\alpha)))\end{aligned}$$

5 Application

Services-oriented architectures (SOA) allow to develop dynamic business processes and agile applications spanning across organisations and computing platforms to quickly adapt to ever changing requirements. By their modular nature, services can be composed to implement processes of various complexities.

Actors of SOA are divided into two rules, the client, having specific requirements, and the provider advertising its services. Non-functional parameters, such as quality of service (QoS) become important when selecting among a range of functionally equivalent services. However, in certain cases, discrepancies between advertised and observed QoS can occur, either because of temporary failures or voluntary over-rating from the provider. When facing such uncertainties, trust mechanisms should be used to select services matching the goals of the clients and providers.

Trust becomes even more crucial in composite services, where not only the client must trust the composite service but also where each provider involved in the composition must trust its partners [4]. Composite services can be modelled as a set of workflow patterns [10], which are equivalent to the complex action framework described in Section 4. Indeed, trust in a composite service depends on the services involved but also on the structure workflow. For example, a provider might agree to participate in a composite service if only its service is used at the end of a sequence, notably for data privacy concerns [3].

In the aforementioned paper, a multi-agent protocol is developed to entice providers to take part in composite web services. This protocol is centered around data privacy in composite services. Basically, and according to Theorem 4, a provider is willing to enter a composite service if and only if it trusts the providers of subsequent services to not mishandle its data. In other words the goal *"not mishandle the data"* only holds after its won service invocation thus fostering the need for trust.

6 Conclusion

We have presented in this work a logical formalization of trust in complex actions, and have sketched how this formalization could be useful for the formal characterization of trust in composite services, where trust in a composed service is defined in a compositional way from trust in the components of that service. Directions of future research are manifold. In the present article we only gave a semantics for a logic of complex actions. On the one hand, future works will be devoted to find a complete axiomatization of the logic of Section 2 and to study the computational properties of this logic (decidability and complexity). On the other hand, we plan to extend the PDL-based formalism of Section 2 by parallel actions in order to be able to formalize services whose components might work in parallel.

7 Acknowledgements

We would like to thank the anonymous reviewers of LIS'2010 whose comments (hopefully) helped to improve the paper.

References

1. Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. In *Proc. 38th IEEE Symposium on Foundations of Computer Science*, Florida, October 1997.
2. Nuel Belnap, Michael Perloff, and Ming Xu. *Facing the Future: Agents and Choices in Our Indeterminist World*. Oxford University Press, Oxford, 2001.
3. Julien Bourdon and Toru Ishida. Trust chaining for provider autonomy in composite services. In *Joint Agent Workshop and Symposium (JAWS'09)*, 2009.
4. Julien Bourdon, Laurent Vercouter, and Toru Ishida. A multiagent model for provider-centered trust in composite web services. In *The 12th International Conference on Principles of Practice in Multi-Agent Systems (PRIMA 2009)*, number 5925 in LNAI, pages 216–228. Springer Verlag, 2009.
5. Cristiano Castelfranchi and Rino Falcone. Principles of trust for MAS: Cognitive anatomy, social importance, and quantification. In *Proceedings of the Third International Conference on Multiagent Systems (ICMAS'98)*, pages 72–79, 1998.
6. Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.
7. Rino Falcone and Cristiano Castelfranchi. Social trust: A cognitive approach. In C. Castelfranchi and Y. H. Tan, editors, *Trust and Deception in Virtual Societies*, pages 55–90. Kluwer, 2001.
8. J. Halpern and M. Vardi. The complexity of reasoning about knowledge and time. *Journal of Computer and System Sciences*, 38:195–237, 1989.
9. J.Y. Halpern and J. H. Reif. The propositional dynamic logic of deterministic, well-structured programs. *Theoretical Computer Science*, 27:127–165, 1983.
10. Q. He, J. Yan, H. Jin, and Y. Yang. Adaptation of web service composition based on workflow patterns. *Proceedings of the 6th International Conference on Service-Oriented Computing (ICSOC'08)*, Jan 2008.
11. Andreas Herzig and Dominique Longin. C&L intention revisited. In *Proc. 9th International Conference on Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 527–535. AAAI Press, 2004.
12. Andreas Herzig and Emiliano Lorini. A dynamic logic of agency I: STIT, abilities and powers. *Journal of Logic, Language and Information*, 19:89–121, 2010.

13. Andreas Herzig, Emiliano Lorini, Jomi F. Hübner, Jonathan Ben-Naim, Olivier Boissier, Cristiano Castelfranchi, Robert Demolombe, Dominique Longin, Laurent Perrussel, and Laurent Vercouter. Prolegomena for a logic of trust and reputation. In *3rd International Workshop on Normative Multiagent Systems (NorMAS 2008), Luxembourg, 15/07/2008-16/07/2008*, pages 143–157, http://wwwen.uni.lu/fdef/luxembourg_business_academy/press, 2008. University of Luxembourg Press. ISBN: 2919940481.
14. Andreas Herzig, Emiliano Lorini, Jomi F. Hübner, and Laurent Vercouter. A logic of trust and reputation. *Logic Journal of the IGPL*, 18(1):214–244, February 2010. Special Issue "Normative Multiagent Systems".
15. Emiliano Lorini. A dynamic logic of agency II: deterministic DLA, Coalition Logic, and game theory. *Journal of Logic, Language and Information*, 19(3):327–351, 2010.
16. Emiliano Lorini and Robert Demolombe. Trust and norms in the context of computer security. In *Proc. Ninth International Conference on Deontic Logic in Computer Science (DEON'08)*, number 5076 in LNCS, pages 50–64. Springer-Verlag, 2008.
17. Marc Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.
18. E. Sirin, B. Parsia, and J. Hendler. Composition-driven filtering and selection of semantic web services. In *AAAI Spring Symposium on Semantic Web Services*, pages 129–138, 2004.
19. J. van Benthem and E. Pacuit. The tree of knowledge in action: Towards a common perspective. In G. Governatori, I. Hodkinson, and Y. Venema, editors, *Proc. of Advances in Modal Logic Volume 6 (AiML 2006)*, pages 87–106. College Publications, 2006.
20. Y. Wang and J. Vassileva. Trust and reputation model in peer-to-peer networks. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, pages 150–157, Linkoeping, Sweden, 2003. IEEE Press.

New decidability result for ground entailment problems and application to security protocols

Yannick Chevalier Mounira Kourjieh

IRIT, Université de Toulouse, France

Abstract. Basin and Ganzinger [3] have proved that for finite sets of clauses saturated for ordered resolution, the ground entailment problem is order local. This implies decidability of ground entailment problems when the employed term ordering has a finite complexity. We present in this paper an extension of that result that does not assume finite complexity of the term ordering, and we show an application of this result on the security protocols.

1 Introduction

Resolution is an inference rule introduced by Robinson [17] for theorem proving in first-order logic. It consists of saturating a theory represented by a finite set of disjunctions, called clauses, with all its consequences; It is correct and complete for refutation [2]: for every unsatisfiable set of clauses, resolution will eventually lead to the empty clause, *i.e.* a clause that can never be satisfied.

Since the seminal work of Robinson lot of efforts have been devoted to finding strategies that limit the possible inferences but still are complete for refutation. The correctness of resolution implies the correctness of these strategies. Among these, there is the ordered resolution which is refutationally complete [1]. Later, it was proved in [3] that if a set S of clauses is saturated for ordered resolution with respect to a term ordering of finite complexity then the *ground entailment problem for* S – *i.e.* deciding whether a clause without variables is a consequence of S – is decidable. We recall that a term ordering is said to be with finite complexity if every term t has a finite number of terms smaller or equal than t with respect to the given ordering. We present in this paper another decidability result for the ground entailment problem which does not assume a finite complexity for the term ordering, and we show an application of such result on the analysis of security protocols.

Outline of this paper. In Section 2 we introduce briefly some of the basic notions we employ in this paper. In Section 3, we introduce our notions of redundancy and saturation, and we give our decidability result. In Section 4, we show an application of our result on cryptographic protocols.

2 Formal setting

2.1 Basic notions

Syntax. Let $\mathcal{X}$ be an infinite set of variables, $\mathcal{C}$ an infinite set of constant symbols, $\mathcal{P}$ a set of predicate symbols with arities, and $\mathcal{F}$ a set of function symbols with arities. The arity of a function symbol (respectively predicate symbol) indicates the number of arguments that the function symbol (respectively the predicate symbol) expects. We define the set of *terms* $\mathcal{T}(\mathcal{F},\mathcal{X})$ as follows: $\mathcal{X},\mathcal{C} \subseteq \mathcal{T}(\mathcal{F},\mathcal{X})$, and for each function symbol $f \in \mathcal{F}$ with arity $n \geq 0$, for each terms $t_1,\ldots,t_n \in \mathcal{T}(\mathcal{F},\mathcal{X})$, we have $f(t_1,\ldots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{X})$; And we denote by $\mathcal{T}(\mathcal{F})$ the set of terms of $\mathcal{T}(\mathcal{F},\mathcal{X})$ that does not contain variables. We define *atoms* as follows: if I is a predicate symbol in $\mathcal{P}$ with arity $n \geq 0$, and $t_1,\ldots,t_n$ are terms in $\mathcal{T}(\mathcal{F},\mathcal{X})$, then $I(t_1,\ldots,t_n)$ is an atom. A *literal* L is either positive literal A or negative literal $\neg A$ where A is an atom, and $\neg$ denotes the *negation.* A *(full) clause* is a formula of the form $\Gamma \rightarrow \Delta$, where Γ and Δ are two sets of atoms representing respectively the *negative literals (or the antecedent)*, and the *positive literals (or the succedent)* of the clause. For example, the set of literals of the clause $C = A_1,\ldots,A_m \rightarrow B_1,\ldots,B_n$ is $\{\neg A_1,\ldots,\neg A_m,B_1,\ldots,N_n\}$. A clause $\Gamma \rightarrow \Delta$ is *Horn* when Δ is a singleton or empty. A clause $\Gamma \rightarrow \Delta$ is *unit* when it has only one literal, *i.e.* either Γ is a singleton and $\Delta = \emptyset$, or $\Gamma = \emptyset$ and Δ is a singleton. A clause $\Gamma \rightarrow \Delta$ is *positive* when $\Gamma = \emptyset$ and is *negative* when $\Delta = \emptyset$. We will write Γ_1,Γ_2 to indicate the union of sets and usually omit braces. If C is a clause, by $\neg C$ we denote the set of unit clauses $\neg L$, with L a literal in C, for example, $\neg C = \{A_1,\ldots,A_m,\neg B_1,\ldots,\neg B_n\}$ for $C = A_1,\ldots,A_m \rightarrow B_1,\ldots,B_n$.

Substitutions and unifiers. A substitution is a partial mapping from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F},\mathcal{X})$ such that no variable in its domain occurs in its range; It is said to be ground if its range is included in $\mathcal{T}(\mathcal{F})$. If M is an expression (i.e. a term, an atom, a clause, or a set of such objects), and σ is a substitution, then $M\sigma$ is obtained by applying σ to M as usually defined; We say that $M\sigma$ is an *instance* of M, and if σ is ground we say that $M\sigma$ is a ground instance of M. We recall that an expression M is ground if it does not contain variables. We refer the reader to [3] for more details.

A substitution σ is a *unifier* of two terms or atoms e_1,e_2 whenever $e_1\sigma = e_2\sigma$. We denote $\mathrm{Unif}(e_1,e_2)$ the set of unifiers of the two expressions e_1 and e_2. We say that a substitution σ is *more general* than a substitution τ if there exists a substitution θ such that $\sigma\theta = \tau$, and we note $\sigma \leq \tau$. Equivalent substitutions, *i.e.* substitutions σ and τ such that $\sigma \leq \tau$ and $\tau \leq \sigma$ are said to be equal up to renaming since in that case the substitution θ is a bijective mapping from variables to variables. It is well-known that whenever the set $\mathrm{Unif}(e_1,e_2)$ is not empty it has a unique minimal element up to renaming, called the *most general unifier* of e_1 and e_2, and denoted $\mathrm{mgu}(e_1,e_2)$.

Orderings. Let E be a set of elements and let $\succ$ be an ordering on E, $\succ$ is *well-founded* if there is no infinite descending chain $e \succ e_1 \succ \ldots$ for any element $e \in E$; is *monotone* if $e \succ e'$ implies $e\sigma \succ e'\sigma$ for any elements $e,e' \in E$ and any substitution σ; *stable* if $e \succ e'$ implies $u[e] \succ u[e']$ for any elements

$u, e, e' \in E$, and $u[e]$ means that u contains e as sub-element; *subterm* if $e[e'] \succ e'$ for any elements $e, e' \in E$; *complete* if it is total over ground elements of E; and *simplification ordering* if it is well-founded, monotone, stable, and subterm. We define an *atom ordering* $\prec_a$ (respectively *term ordering* $\prec_a$) to be an arbitrary ordering on atoms (respectively on terms). An atom ordering $\prec_a$ is said to be compatible with a term ordering $\prec_t$ if it satisfies the following condition: $p(t_1, \ldots, t_k) \prec_a q(s_1, \ldots, s_n)$ if and only if for any j with $1 \leq j \leq n$ there exists an i with $1 \leq i \leq k$ such that $t_i \prec_t s_j$. We remark that any ordering $\succ$ on a set of elements E can be extended to an ordering $\succ^{set}$ on finite sets over E as follows: if η_1 and η_2 are two finite sets over E, we have $\eta_1 \succ^{set} \eta_2$ if (i) $\eta_1 \neq \eta_2$ and (ii) whenever for every $e \in \eta_2 \setminus \eta_1$ then there is $e' \in \eta_1 \setminus \eta_2$ such that $e' \succ e$; We also remark that the set ordering extension of a complete (respectively well-founded) ordering is also complete (respectively well-founded) [11]. Now, we extend the atom ordering on clauses; To this end, we identify clauses by the sets of their respective atoms, and we pre-order clauses with respect to the order on the sets of their respective atoms. For example the clauses $A_1, A_2 \rightarrow B$ and $A_1 \rightarrow B$ are identified respectively by the two sets of atoms $\{A_1, A_2, B\}$ and $\{A_1, B\}$; The second set is strictly smaller than the first one with respect to the set ordering extension of $\prec_a$, and hence the second clause is strictly smaller than the first one. We refer reader to [11] (Chapter 6) for more details. We remark that the ordering we define on clauses is crucial for our decidability proof for the ground entailment problem. We show in the next Lemma some properties on term and atom orderings.

Lemma 1. *Let $\prec_t$ be a complete simplification ordering over terms, and let $\prec_a$ be a complete atom ordering compatible with $\prec_t$. Then $\prec_a$ is:* (1) *well-founded,* (2) *monotone, and* (3) *$B \prec_a A$ implies* $\mathrm{Var}(B) \subseteq \mathrm{Var}(A)$.

Proof. 1. Let us prove that $\prec_a$ is well-founded by contradiction. Assume that $\prec_a$ is not well-founded. By definition, there is an infinite descending chain of atoms $A_0 \prec_a A_1 \prec_a \ldots$. By the compatibility of $\prec_a$ with $\prec_t$, we deduce that there is an infinite descending chain of terms $t_0 \prec_t t_1 \prec_t \ldots$ where t_i is a term of the atom A_i. That implies $\prec_t$ is not well-founded which contradicts the fact that $\prec_t$ is a complete simplification ordering.

2. Let A, B be two atoms such that $B \prec_a A$. Suppose that $A = I(t_1, \ldots, t_n)$ and $B = I'(s_1, \ldots, s_m)$. By the compatibility of $\prec_a$ with $\prec_t$, for all $i \in \{1, \ldots, m\}$, there is $j \in \{1, \ldots, n\}$ such that $s_i \prec_t t_j$, and then, by monotonicity of $\prec_t$, $s_i\sigma \prec_t t_j\sigma$ for any substitution σ. Again by the compatibility of $\prec_a$ with $\prec_t$, we deduce that $B\sigma \prec_a A\sigma$ for any σ and then the monotonicity of $\prec_a$.
3. Let A, B be two atoms such that $B \prec_a A$. By the compatibility of $\prec_a$ with $\prec_t$, each term in B is smaller than a term in A for the ordering $\prec_t$ and as $Var(t) \subseteq Var(t')$ for all terms t, t' and $t \prec_t t'$, we deduce that $Var(B) \subseteq Var(A)$.

In the remainder of this paper, we assume only that the atom ordering $\prec_a$ is monotone, well-founded, and is such that $B \prec_a A$ implies $\mathrm{Var}(B) \subseteq \mathrm{Var}(A)$ for every atoms A, B.

2.2 Ordered resolution

The resolution is one of the most successful methods for automated proof search, it was developed by Robinson [17]. After that, several resolution strategies have been proposed in the literature that limit the possible inferences, and these proposed strategies are still complete for refutation and correct. A complete presentation on resolution and its different strategies can be found in [2]. In this paper, we will use the *ordered resolution* that we briefly present next.

The ordered resolution is described by the *ordered factoring inference rule* and the *ordered resolution inference rule*, which are given below:
Ordered factoring rule:

$$\frac{\Gamma \rightarrow \Delta, A, A'}{(\Gamma \rightarrow \Delta, A)\sigma} \; \sigma = \mathrm{mgu}(A, A')$$

where $A\sigma$ is strictly maximal with respect to $\Gamma\sigma$, and maximal with respect to $\Delta\sigma$.

We will distinguish here between two types of *ordered resolution inference rules:* the *a posteriori ordered resolution* and the *a priori ordered resolution* inference rules.
A posteriori ordered resolution rule:

$$\frac{\Gamma \rightarrow \Delta, A \qquad A', \Gamma' \rightarrow \Delta'}{(\Gamma, \Gamma' \rightarrow \Delta, \Delta')\sigma} \; \sigma = \mathrm{mgu}(A, A')$$

where $A\sigma$ is strictly maximal with respect to $\Gamma\sigma,\ \Delta\sigma$ and $A\sigma$ is maximal with respect to $\Gamma'\sigma,\ \Delta'\sigma$.

A priori ordered resolution inference rule:

$$\frac{\Gamma \rightarrow \Delta, A \qquad A', \Gamma' \rightarrow \Delta'}{(\Gamma, \Gamma' \rightarrow \Delta, \Delta')\sigma} \; \sigma = \mathrm{mgu}(A, A')$$

where A is strictly maximal with respect to $\Gamma,\ \Delta$ and A' is maximal with respect to $\Gamma',\ \Delta'$.

We remark that the two previous inference rules coincide on ground clauses. It has been remarked [7] that on non-ground clauses there were inferences possible in *a priori ordered resolution* but impossible in the *a posteriori* version, and hence, the termination of the saturation when considering *a posteriori ordered resolution* does not implies its termination when considering the *a priori* version. We prove in Lemma 4 that such problematic cases are eliminated and that thanks to the notion of redundancy we employ. We also remark that both versions of ordered resolution are complete and sound [11].

Ground entailment problem.
Given a set of clauses S, the *ground entailment problem for S* is defined as follows:
Input : a ground clause C
Output : entailed if and only if $S \models C$ (*i.e.* C is true in every model of S).

The main contribution of this paper is a new decidability result for the ground entailment problem. This result can be stated by the following theorem:

Theorem 1. *Let $\prec_a$ be a well-founded, monotone atom ordering such that $A \prec_a B$ implies $\mathrm{Var}(A) \subseteq \mathrm{Var}(B)$ for every atoms A and B. Let S be a set of clauses, and assume that saturation on S terminates using the atom ordering $\prec_a$. Then the ground entailment problem for S is decidable.*

The rest of this paper is devoted to this result.

3 Decidability Result

We introduce in this section our saturation algorithm, our notion of redundancy, and then give our decidability result.

Definition 1. *(Rewriting systems on atoms) A* rewriting system $\mathcal{R}$ on atoms *based on $\prec_a$ is a set of pairs (L, R) where L and R are two atoms with $R \prec_a L$. Each pair (L, R) is called a* rewriting rule *and is denoted $L \to R$.*

We remark that rewriting systems on atoms behave as rewriting systems on terms. Let $\mathcal{R}$ be a rewriting system on atoms, $A \to_{\mathcal{R}} B$ means that the atom A rewrites to the atom B by $\mathcal{R}$. Let A be an atom, we denote $A \downarrow_{\mathcal{R}}$ the set of atoms reachable from A when applying rules in $\mathcal{R}$, this notion is extended to sets of atoms in a traditional way. We let $A \downarrow_{\mathcal{R}}^{-}$ be the set $A \downarrow_{\mathcal{R}} \setminus \{A\}$. We denote $A \prec_{\mathcal{R}} B$ whenever $A \in B \downarrow_{\mathcal{R}}^{-}$. By definition of $\to_{\mathcal{R}}$, $\prec_a$ and $\prec_{\mathcal{R}}$, it is easy to see that $A \prec_{\mathcal{R}} B$ implies $A \prec_a B$, and $A \to_{\mathcal{R}} B$ implies $B \prec_a A$ for every atoms A and B.

Lemma 2. *If $\mathcal{R}$ is a finite rewriting system on atoms based on $\prec_a$ then for every ground atom C the set $C \downarrow_{\mathcal{R}}$ is finite.*

Proof. Consider the (infinite) directed graph whose vertices are ground atoms, and there is an edge from A to B whenever $A \to_{\mathcal{R}} B$. First we note that since in every rewrite rule $L \to R$ we have $\mathrm{Var}(R) \subseteq \mathrm{Var}(L)$ then for every atom A there is most $|\mathcal{R}|$ successors. Second we note that $A \to_{\mathcal{R}} B$ implies $B \prec_a A$, and thus this graph is acyclic. Also, the fact that $\prec_a$ is well-founded implies that this graph does not contain any infinite path. Consider its (potentially infinite) tree build from the vertice C by considering the possible paths to all other nodes. We note that this tree is of finite branching and every path in it is finite. Thus by König's lemma this graph has only a finite number of vertices. Since all atoms in $C \downarrow_{\mathcal{R}}$ must be by definition vertices in this tree, we have that $C \downarrow_{\mathcal{R}}$ is finite.

Definition 2. *(Rewriting system based on a set of clauses) Let S be a finite set of clauses. The rewriting system $\mathcal{R}(S)$ based on S is a rewriting system on atoms defined as the set of all rewriting rules $L \to R$ such that there exists a clause $C \in S$ with:* (1) L, R *are two distinct atoms of* C*, and* (2) $R \prec_a L$*.*

First let us remark that since S is finite we also have that $\mathcal{R}(S)$ is finite. We also remark that if $S \subseteq S'$, then $\mathcal{R}(S) \subseteq \mathcal{R}(S')$.

Definition 3. *(Local DAG proofs) Let S be a set of clauses, C be a clause and $\mathcal{A}$ be a set of ground atoms. We say that C is a* $\mathcal{A}$-local *DAG proof of S, and we write $S \vdash_{\mathcal{A}} C$, whenever there exists a set T of ground clauses and a well-founded partial ordering $<_T$ on T such that* (1) *for any clause $t \in T$, we have either t is a ground instance of a clause in $S \cup \neg C$, or there exists $t_1, t_2 \in T$ with $t_1, t_2 <_T t$ and $t_1, t_2 \rightsquigarrow t$ is an instance of resolution inference rule;* (2) *T contains the empty clause; and* (3) *for every atom $A \in T$ we have $A \in \mathcal{A}$.*

The ordering $<_T$ is not to be confused with an ordering on clauses; It is the ordering on the nodes of a proof tree in DAG representation. Now, we can deduce the following lemma.

Lemma 3. *The problem consisting in determining, given a finite set S of clauses, a ground clause C and a finite rewriting system on atoms $\mathcal{R}$, whether $S \vdash_{C\downarrow_{\mathcal{R}}} C$ is decidable.*

Proof. It suffices to remark that, seeing that $C \downarrow_{\mathcal{R}}$ is finite by Lemma 2, the set of all instances of clauses in S with atoms occurring in $C \downarrow_{\mathcal{R}}$ is finite.

Definition 4. *Given a set of clauses S and a clause C. The notation $\sigma_{S,C}$ means that σ is a substitution grounding of C for S, that is σ is a substitution satisfying the following:* (1) *the domain of σ is the set of variables occurring in C; and* (2) *σ is injective and maps each variable x to a constant c_x that does not occur in S or C.*

Definition 5. *(Redundancy) Let $\mathcal{R}$ be a finite rewriting system on atoms.*

- *A ground clause C is $\mathcal{R}$-redundant in a set of clauses S if $S \vdash_{C\downarrow_{\mathcal{R}}} C$.*
- *A non-ground clause C is $\mathcal{R}$-redundant in a set of clauses S if all its ground instances are $\mathcal{R}$-redundant in S;*
- *An inference $C', C" \rightsquigarrow C$ by ordered resolution is $\mathcal{R}$-redundant in the set of clauses S if either C' or $C"$ is $\mathcal{R}$-redundant in S or $S \vdash_{C\sigma_{S,C}\downarrow_{\mathcal{R}} \cup A\sigma_{S,C}\downarrow^{-}_{\mathcal{R}}} C\sigma_{S,C}$, with $A\sigma$ the resolved atom.*

We remark that if an inference by ordered resolution is redundant with respect to the notion of redundancy given in [3] then it is also redundant with respect to our notion of redundancy. Using this notion of redundancy, we show next how to relate *a priori* and *a posteriori* ordered resolution inference rules.

Lemma 4. *Let C_1, C_2 be two clauses and let $C_1, C_2 \rightsquigarrow C$ be an inference by a priori ordered resolution with $A\sigma$ the resolved atom. Let $\mathcal{R} = \mathcal{R}(C_1\sigma) \cup \mathcal{R}(C_2\sigma)$. Then either this inference is $\mathcal{R}$-redundant or is an inference by a posteriori ordered resolution.*

Proof. Assume this is not an inference for *a posteriori* ordered resolution. Then the resolved atom $A\sigma$ is not maximal for $\prec_a$ in the set of atoms of C. Thus there exists in C_1 or C_2 an atom B with $A\sigma \prec_a B\sigma$. By definition we thus have $B\sigma \to A\sigma \in \mathcal{R}$. As a consequence, all the atoms in $C_1\sigma, C_2\sigma$ are in $C\downarrow_{\mathcal{R}}$, and hence, every atom in $C_1\sigma, C_2\sigma$ is smaller than an atom in C with respect to the atom ordering $\prec_a$. By definition this inference is $\mathcal{R}$-redundant in $\{C_1, C_2\}$.

We define next our notion of *saturation* for ordered resolution.

Definition 6. *(Saturated sets of clauses) Let $\mathcal{R}$ be a rewriting system on atoms. We say that a set S of clauses is $\mathcal{R}$-saturated up to redundancy under ordered resolution, if:* (1) *any inference by ordered resolution from premises in S is $\mathcal{R}$-redundant in S;* (2) $\mathcal{R}(S) \subseteq \mathcal{R}$*; and* (3) *for each a priori ordered resolution inference $C_1, C_2 \rightsquigarrow C$ with $C_1, C_2 \in S$, if the resolved atom $A\sigma$ is not maximal in $C_1\sigma, C_2\sigma$ then $\mathcal{R}(C_1\sigma, C_2\sigma) \subseteq \mathcal{R}$.*

3.1 Saturation

We present now a procedure that, providing it terminates, constructs from a finite set S of clauses the pair $(S', \mathcal{R})$ such that: (1) : S' is a finite set of clauses; (2) : $\mathcal{R}$ is a rewriting system on atoms; and (3) : for every ground clause C, we have $S \models C$ iff C is a $C\downarrow_{\mathcal{R}}$-local DAG proof of S' (*i.e.* $S' \vdash_{C\downarrow_{\mathcal{R}}} C$).

Saturation procedure.

Input:
A finite set S of clauses.
Initialisation:
Let $(S_1, \mathcal{R}_1) = (S, \mathcal{R}(S))$, and $i = 1$.
Transformation step.
We construct the couple $(S_{i+1}, \mathcal{R}_{i+1})$ from the couple $(S_i, \mathcal{R}_i)$ as follows: Let $C_1, C_2 \to C$ be an inference by ordered resolution with $C_1, C_2 \in S_i$, and $A\sigma$ the resolved atom; One of the following three cases will be applied:

- *Non-maximality:* If $A\sigma$ is not maximal for $\prec_a$ in the atoms of $C_1\sigma, C_2\sigma$ then $S_{i+1} = S_i$, $\mathcal{R}_{i+1} = \mathcal{R}_i \cup \mathcal{R}(\{C_1\sigma, C_2\sigma\})$, and $i = i + 1$;
- *Redundancy:* Otherwise, if $S_i \vdash_{C\downarrow_{\mathcal{R}_i}} C$, then $S_{i+1} = S_i$, $\mathcal{R}_{i+1} = \mathcal{R}_i$, and $i = i + 1$;
- *Discovery:* Otherwise a new clause useful for establishing local proofs has been discovered, and hence $S_{i+1} = S_i \cup \{C\}$, $\mathcal{R}_{i+1} = \mathcal{R}_i \cup \mathcal{R}(C)$, and $i = i + 1$.

Iteration
We repeat the **Transformation step** until a fixed point is reached.

We remark that a sequence of Transformation steps in the previous procedure is *fair* if every possible inference by *a priori* ordered resolution is eventually performed [2].

Definition 7. *(Output of the saturation procedure) Given a finite set S of clauses and an atom ordering $\prec_a$, $\mathrm{Sat}_{\prec_a}(S)$ denotes the pair $(S', \mathcal{R})$ obtained by a fair sequence of Transformation steps by the saturation procedure in case it terminates.*

We prove next that the saturation procedure actually constructs a saturated set of clauses.

Proposition 1. *Let S be a finite set of clauses and $\prec_a$ be a monotone, well-founded atom ordering such that $A \prec_a B$ implies $\mathrm{Var}(A) \subseteq \mathrm{Var}(B)$ for every atoms A and B. If the saturation procedure terminates on S and outputs $\mathrm{Sat}_{\prec_a}(S) = (S', \mathcal{R})$ then S' is $\mathcal{R}$-saturated.*

Proof. Assume that there exists two clauses $C_1, C_2 \in S'$ such that the inference by ordered resolution $C_1, C_2 \rightsquigarrow C$ is not $\mathcal{R}$-redundant. In the saturation algorithm it thus falls into the case *non-maximality* or the case *discovery.*

non-maximality: Assume that the resolved atom $A\sigma$ is not maximal in the atoms of $C_1\sigma, C_2\sigma$. Then this inference is not an inference by *a posteriori* ordered resolution. It is thus $\mathcal{R}(C_1\sigma) \cup \mathcal{R}(C_2\sigma)$-redundant. Since it is not redundant we must have $\mathcal{R}(C_1\sigma) \cup \mathcal{R}(C_2\sigma) \not\subseteq \mathcal{R}$. This implies that $(S', \mathcal{R})$ is not a result of the saturation algorithm.

discovery: If $(S', \mathcal{R})$ is a result of the saturation algorithm we would have $C \in S'$, then it is trivial that the inference is redundant in S'.

As a consequence every inference between two clauses of S' must be $\mathcal{R}$-redundant, and thus the set S' is $\mathcal{R}$-saturated by Definition 6.

3.2 Decidability of the ground entailment problem

We consider in this section an $\mathcal{R}$-saturated set S of clauses.

Proposition 2. *Let S be an $\mathcal{R}$-saturated set of clauses, and C be a ground clause. We have that $S \models C$ implies $S \vdash_{C\downarrow_{\mathcal{R}}} C$*

Proof. Assume that $S \models C$, and let $\mathcal{T}$ be the set of ground DAG proofs of $S \cup \neg C$ that contain the empty clause. By completeness of resolution we know that $\mathcal{T} \neq \emptyset$. For each set of atoms $\mathcal{A}$ let $\mathcal{T}_{\mathcal{A}}$ be the set of DAG proofs T such that $T\downarrow_{\mathcal{R}} \setminus \mathrm{atoms}(C)\downarrow_{\mathcal{R}} = A$ and let $\mathcal{A}_{min}$ be minimal such that $\mathcal{T}_{\mathcal{A}_{min}}$ is not empty. If $\mathcal{A}_{min}$ is empty then we are done as each $T \in \mathcal{T}_{\mathcal{A}_{min}}$ is then a DAG proof of $S \cup \neg C$ in which all atoms are in $C\downarrow_{\mathcal{R}}$ and that contains the empty clause, and thus $S \vdash_{\downarrow_{\mathcal{R}}} C$. Otherwise let us derive a contradiction. For any $T \in \mathcal{T}_{\mathcal{A}_{min}}$ the set of atoms in T is finite and therefore $T\downarrow_{\mathcal{R}}$ is also finite by Lemma 2. Thus we can consider a maximal element A in $\mathcal{A}_{min}$ (the same for all T in $\mathcal{T}_{\mathcal{A}_{min}}$). Since A is maximal we also have that A is an atom occurring in T for each $T \in \mathcal{T}_{\mathcal{A}_{min}}$.

claim For any $T \in \mathcal{T}_{\mathcal{A}_{min}}$ the atom A is maximal in atoms(T) for the ordering $\prec_{\mathcal{R}}$.

Proof of the claim. *By contradiction if this were not the case there would exist $B \in T$ with $A \prec_{\mathcal{R}} B$. Since A is maximal in $\mathcal{A}_{min}$ we would have that B is not in this set. Since $B \in T$ this would imply $B \in C \downarrow_{\mathcal{R}}$, which implies that $A \in C \downarrow_{\mathcal{R}}$, which would contradict $A \in T \downarrow_{\mathcal{R}} \setminus C \downarrow_{\mathcal{R}}$.*

Let T be in $\mathcal{T}_{\mathcal{A}_{min}}$, and let Leaves_A^+ be the set of minimal clauses (for $<_T$) that contain the atom A, and Leaves_A^- be the subset of minimal clauses (for $<_T$) that do not contain A. Since T is a DAG proof of $\mathit{Leaves}_A^+ \cup \mathit{Leaves}_A^-$ that contains the empty clause, the correctness of resolution implies that the set of clauses $\mathit{Leaves}_A^+ \cup \mathit{Leaves}_A^-$ is unsatisfiable. Let us consider the set Leaves' of all possible conclusions of resolution on A between clauses in Leaves_A^+. The set of ground clauses $\mathit{Leaves}' \cup \mathit{Leaves}_A^-$ is also unsatisfiable.

claim Each clause $C_A \in \mathit{Leaves}_A^+$ is an instance with a substitution σ of a clause $C_A^s \in S$ that has a maximal atom A^s for $\prec_a$ with $A^s\sigma = A$.

Proof of the claim. *By definition of DAG proofs the minimality for $<_{T_0}$ of C_A implies that C_A is either an instance of a clause in S or of a clause in $\neg C$. Since A is not an atom occurring in C the later case is excluded. Thus there exists $C_A^s \in S$, an atom $A^s \in C_A^s$, and a substitution σ such that $A^s\sigma = A$ and $C_A^s\sigma = C_A$. Finally if A^s is not maximal for $\prec_a$ in C_A^s then it is not maximal for $\prec_{\mathcal{R}}$ and thus A cannot be maximal for $\prec_{\mathcal{R}}$ in the atoms of C_A. This would contradict the fact that A is maximal for $\prec_{\mathcal{R}}$ among the atoms occurring in T.*

Thus every resolution on A between clauses in Leaves_A^+ is an instance with substitution σ of an *a priori* ordered resolution inference between two clauses C_1 and C_2 of S. Let $C_3 \in \mathit{Leaves}'$ be its conclusion. Since S is $\mathcal{R}$-saturated each such inference is redundant. We note that A maximal in atoms(T) for $\prec_{\mathcal{R}}$ and the fact that S is saturated (second point of the ordering condition) for $\mathcal{R}$ imply that A cannot be smaller for $\prec_{\mathcal{R}}$ than an atom in C_3. Thus for each conclusion C_3 we can define a set $\mathcal{S}(C_3)$ which is either:

- the singleton $\{C_3\}$ if C_3 is an instance of a clause $C_3^g \in S$;
- or a set $S_{C_3}^g$ of instances of clauses of S whose atoms are in $C_3 \downarrow_{\mathcal{R}} \cup A \downarrow_{\mathcal{R}}^-$ that entails C_3

The set of ground clauses $S^g = \mathit{Leaves}_A^- \cup \bigcup_{C_3 \in \mathit{Leaves}'} \mathcal{S}(C_3)$ is unsatisfiable. By completeness of resolution there exists a DAG proof T_m of S^g that contains the empty clause. By construction we have $T_m \downarrow_{\mathcal{R}} \subseteq (\text{atoms}(T) \setminus \{A\}) \downarrow_{\mathcal{R}} \cup A \downarrow_{\mathcal{R}}^-$. Since A is maximal in atoms(T) for $\prec_{\mathcal{R}}$ and A is not in $C \downarrow_{\mathcal{R}}$ this implies that $T_m \downarrow_{\mathcal{R}} \setminus C \downarrow_{\mathcal{R}} \prec_a T \downarrow_{\mathcal{R}} \setminus C \downarrow_{\mathcal{R}}$. This contradicts the fact that $\mathcal{A}_{min}$ is minimal such that $\mathcal{T}_{\mathcal{A}_{min}}$ is not empty.

We note that $S \vdash_{C\downarrow_{\mathcal{R}}} C$ trivially implies $S \models C$ by correctness of resolution. As a consequence of Lemma 3 and of Proposition 2 we thus have the following proposition.

Proposition 3. *If S is an $\mathcal{R}$-saturated set of clauses then the ground entailment problem for S is decidable.*

The main theorem of this paper is a self-contained re-formulation of the above proposition using the initial set of clauses.

Theorem 1. *Let $\prec_a$ be a well-founded, monotone atom ordering such that $A \prec_a B$ implies $\mathrm{Var}(A) \subseteq \mathrm{Var}(B)$ for every atoms A and B. Let S be a set of clauses, and assume that saturation on S terminates using the atom ordering $\prec_a$. Then the ground entailment problem for S is decidable.*

4 Applications: From cryptographic protocols to logic of clauses

Cryptographic protocols are programs designed to ensure secure electronic communications between participants using an insecure network, they use cryptographic primitives to obtain the basic building bricks. However, even if these bricks are secure, the way they are combined in the protocol is very important. Indeed, several protocols which were believed to be correct were late found to have attacks; The most relevant example is the bug, *man-in-the-middle* attack, of the Needham-Schroeder public key protocol [13] found 17 years after the publication of the protocol. This situation shows that one actually needs to formally verify the protocols.

Among several methods proposed in the literature to analyse security protocols, for instance the methods that use theorem provers, or process algebra, or tree automata, there is a method that uses clauses and Horn clauses to analyse protocols. This method has been widely studied in the literature, and several models and decidability results have been given [6, 5, 18]. In the most common models of the last method, intruder behaviour, protocol description and security properties are encoded as Horn clauses; And, the insecurity problem of the protocol, *i.e.* the problem that inputs a protocol and a security property and outputs yes when the protocol does not preserve the property, is reduced to the satisfiability problem for a class of Horn clauses.

In this section, we present a new model based on the use of clauses and Horn clauses to analyse cryptographic protocols, and hence show how to apply the decidability result obtained in Section 3 to the analysis of protocols.

4.1 The model

Now, we present our model to analyse security protocols using Horn clauses. We remark that Horn Clauses, and not full clauses, are sufficient to analyse security protocols, and that we actually need one unary predicate symbol, let I be this symbol.

Intruder clauses. We show now how to represent the intruder behaviour by Horn clauses. Let $\mathcal{I}$ be an intruder and let $\mathcal{L}_\mathcal{I}$ be its deduction system ($\mathcal{L}_\mathcal{I}$ represents the intruder capacities). $\mathcal{L}_\mathcal{I}$ is a set of rules of the form $u_1, \ldots, u_n \to v$ where $u_1, \ldots, v_n, v$ are terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. We recall that the rule $u_1, \ldots, u_n \to v \in \mathcal{L}_\mathcal{I}$ means that the intruder is able to construct the term v if he knows the terms $u_1, \ldots, u_n$. The set of clauses associated to $\mathcal{I}$ is

$$C_\mathcal{I} \overset{def}{=} \{I(u_1), \ldots, I(u_n) \to I(v) \textit{ such that } u_1, \ldots, u_n \to v \in \mathcal{L}_\mathcal{I}\}$$

For example, the clause $I(x), I(y) \to I(\{x\}_y^s)$ represents the capacity of the intruder to compute the symmetric encryption of a term t using a key k is he knows t and k.

Insecurity problem of cryptographic protocols. We show here how we analyse security protocols using Horn clauses. In [6, 5, 18], the rules describing the protocol, and the security properties are encoded by Horn clauses, and the insecurity problem of the protocol is reduced to the satisfiability problem for a class of Horn clauses. Our model differs from the previous models by the fact that we do not encode neither protocol description rules, nor security properties by Horn clauses. Actually, we reduce the insecurity problem of protocols to the ground entailment problem using two steps instead of one step:

- First, following the approach based on the resolution of constraint systems to analyse protocols, which is widely studied in the literature [12], we reduce the insecurity problem to the intruder reachability problem;
- An then, we reduce the intruder reachability problem to the ground entailment problem.

We will describe next our method, and we recall that we are concerned only by the secrecy property, protocols with bounded number of sessions, and passive intruder.

Constraint systems are quite common in modelling cryptographic protocols for a bounded number of sessions [12]. Actually, many protocol security properties can be characterised as *reachability problems* which are converted to *constraint solving problems*. In this approach, a constraint system is built from each execution of the protocol, and we say that an execution is not secure (or does not preserve the secrecy) if the corresponding constraint system is satisfiable modulo the intruder deduction system. And hence, the insecurity problem of the protocol can be reduced to the satisfiability problem of constraint systems modulo the intruder deduction system, which is also called intruder reachability problem. Since the intruder is passive, the constraint systems corresponding to the executions, the satisfiability problem, and the intruder reachability problem are actually ground.

As shown in the previous paragraph, the intruder capacities are represented by a set of Horn clauses. We also represent the initial knowledge of the intruder, denoted IK, by a set of clauses C_{IK}: $C_{IK} = \{I(m) \textit{ such that } m \in IK\}$. For instance the set of clauses $C_{IK} = \{I(a), I(Pk_a)\}$ means that the intruder knows the messages a, Pk_a. Now we show how we reduce the intruder ground reachability problem to the ground entailment problem.

Let the ground constraint system $\mathfrak{C} = (E_1 \rhd t_1, \ldots, E_n \rhd t_n)$ where $E_i = \{e^i_1, \ldots, e^i_n\}$, t_i, e^i_j are terms in $\mathcal{T}(\mathcal{F})$. For each constraint $E_i \rhd t_i \in \mathfrak{C}$, we associate the following Horn clause $C_{E_i} \to I(t_i)$ where $C_{E_i} = \{I(e^i_1), \ldots, I(e^i_n)\}$. For example, the clause $I(a), I(b) \to I(< a, b >)$ corresponds to the constraint $\{a, b\} \rhd \; < a, b >$. The constraint system $\mathfrak{C}$ will then be associated to the set of ground Horn clauses $C_{\mathfrak{C}} = \{C_{E_1} \to I(t_1), \ldots, C_{E_n} \to I(t_n)\}$. It is easy to see that $\mathfrak{C}$ is satisfiable if and only if $C_{\mathfrak{C}}$ is entailed by $C_{\mathcal{I}}$. Thus, an execution $\mathfrak{exec}$ does not preserve the secrecy if and only if the set of Horn clauses $C_{\mathfrak{C}_{\mathfrak{exec}}}$ is entailed by the set of Horn clauses $C_{\mathcal{I}}$. Finally, we conclude that in presence of passive intruder, each execution of the protocol is associated to a ground constraint system, and then, the insecurity problem of an execution and hence of a cryptographic protocol under a bounded number of sessions is reduced to the ground entailment problem for $C_{\mathcal{I}}$.

4.2 Example: Needham-Schroeder symmetric key protocol

Presentation of the protocol. We consider the Needham-Schroeder symmetric key protocol [13] as an example. This protocol intends to permit *Alice* to establish a shared key (session key) with *Bob* and to obtain mutual conviction of the possession of the key by each other. The session key is created by a *trusted server* which shares a secret key with each party. The protocol can be described as follows:

$$\mathfrak{P}_{NS} : \begin{cases} 1 \;\; A \Rightarrow S : & A, B, N_A \\ 2 \;\; S \Rightarrow A : & \{N_A, B, K_{AB}, \{K_{AB}, A\}^s_{K_{BS}}\}^s_{K_{AS}} \\ 3 \;\; A \Rightarrow B : & \{K_{AB}, A\}^s_{K_{BS}} \\ 4 \;\; B \Rightarrow A : & \{N_B\}^s_{K_{AB}} \\ 5 \;\; A \Rightarrow B : & \{N_B - 1\}^s_{K_{AB}} \end{cases}$$

N_A (respectively N_B) represents the nonce freshly created by A (respectively B), K_{AS} (respectively K_{BS}) represents the secret key shared between A (respectively B) and the *trusted server*, and K_{AB} the session key shared between A and B. The functions symbols $\{-\}^s_-, \{-\}^{-1s}_-$ denotes respectively the symmetric encryption and symmetric decryption.

In this protocol, we have three roles, the *trusted server*, *Alice* and *Bob*. In the same spirit as in [11], we describe the role server by the following rule:
$S1$:

$$v_1 \Rightarrow \{\pi_2(\pi_2(v_1)), \pi_1(\pi_2(v_1)), K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, \{K_{\pi_1(v_1)\pi_1(\pi_2(v_1))}, \pi_1(v_1)\}^s_{K_{\pi_1(\pi_2(v_1))S}}\}^s_{K_{\pi_1(v_1)S}};$$
$$v_1 \overset{?}{=} X_1, Y_1, Z_1$$

The role *Alice* by the following set of rules:

$$\begin{array}{ll} A1: & \emptyset \Rightarrow A, X_2, N_A; \emptyset \\ A2: & v_2 \Rightarrow \pi_2(\pi_2(\pi_2(\{v_2\}^{-1s}_{K_{AS}}))); \ v_2 \overset{?}{=} \{N_A, X_2, Y_2, Z_2\}^s_{K_{AS}} \\ A3: & v_3 \Rightarrow \{\{v_3\}^{-1s}_{\pi_1(\pi_2(\pi_2(\{v_2\}^{-1s}_{K_{AS}})))} - 1\}^s_{\pi_1(\pi_2(\pi_2(\{v_2\}^{-1s}_{K_{AS}})))}; \\ & \qquad v_3 \overset{?}{=} \{X_2'\}^s_{\pi_1(\pi_2(\pi_2(\{v_2\}^{-1s}_{K_{AS}})))} \end{array}$$

And the role *Bob* by the following set of rules:

$$\begin{array}{ll} B1: & v_4 \Rightarrow \{N_B\}^s_{\pi_1(\{v_4\}^{-1s}_{K_{BS}})}; v_4 \overset{?}{=} \{X_3, Y_3\}^s_{K_{BS}} \\ B2: & v_5 \Rightarrow \emptyset; v_5 \overset{?}{=} \{N_B - 1\}^s_{\pi_1(\{v_4\}^{-1s}_{K_{BS}})} \end{array}$$

Attack on the protocol. Many attacks have been discovered on this protocol, Denning and Sacco [8] considered that communication keys may be compromised, and showed that the protocol is vulnerable to replay attack. Here we concentrate on the key exchange goal rather than on the authentication of the two parties. The key exchange goal can be expressed by the secrecy of the nonce N_B. Intuitively, if N_B remains secret than the key K_{AB} has also been kept secret. We will present the attack discovered by Pereira and Quisquater [16], this attack described in Figure 1 allows, as we will see below, the intruder to break the secrecy of N_B. This attack is possible if the encryption scheme used in the implementation of the protocol is used in the cipher-block-chaining (CBC) mode. In the case of CBC encryption, the intruder may be able to get from any encrypted message the encryption of any of its prefixes, and that without knowing the encryption key: from the message $\{< x, y >\}^s_z$, the intruder can deduce the message $\{x\}^s_z$. This property, called *prefix property*, is encoded by the deduction rule

$$\{< x, y >\}^s_z \to \{x\}^s_z$$

and hence by the clause

$$C_{pre} = I(\{< x, y >\}^s_z) \to I(\{x\}^s_z)$$

In a first session (1), the intruder can listen to the message $\{N_A, B, K_{AB}, \{K_{AB}, A\}^s_{K_{BS}}\}_{K_{AS}}$ and then, using the prefix property, computes the message $\{N_A, B\}_{K_{AS}}$. This message is of the form *Alice* might expect to receive as third message of a later session of the protocol where *Bob* is considered to play initiator's role. Then once the message $\{N_A, B\}^s_{K_{AS}}$ computed, the intruder starts another session of the protocol by sending to *Alice* the message $\{N_A, B\}_{K_{AS}}$. *Alice* thinks that *Bob* has started a session (2) with her and thus, *Alice* can be fooled into accepting the publicly known N_A as a secret key shared with *Bob*. Let us consider an instance of the protocol $\mathfrak{P}_{NS}$ where *Alice* instantiates once the role A and once the role B, *Bob* instantiates only once the role B and the agent s instantiates only once the role S, K_{bs} denotes the secret

$$
\begin{array}{ll}
(1).1 & A \Rightarrow S: \quad A, B, N_A \\
(1).2 & S \Rightarrow A: \quad \{N_A, B, K_{AB}, \{K_{AB}, A\}^s_{K_{BS}}\}_{K_{AS}} \\
(2).3 & I(B) \Rightarrow A: \quad \{N_A, B\}_{K_{AS}} \\
(2).4 & A \Rightarrow I(B): \quad \{N'_A\}_{N_A}
\end{array}
$$

Fig. 1. Attack on the Needham-Schroeder protocol

key shared between *Bob* and s, and K_{as} denotes the secret key shared between *Alice* and s.

The attack described above is a possible execution of this instance of $\mathfrak{P}_{NS}$. This execution is given by the set of rules $\{A_1(Alice), S_1(s), B_1(Alice)\}$. At the end of this execution, the intruder breaks the secrecy of the nonce n'_a. This execution is then given by the following set of rules:

$$
\begin{array}{l}
\emptyset \Rightarrow Alice, Bob, n_a \\
v_1 \Rightarrow \{n_a, Bob, K_{ab}, \{K_{ab}, Alice\}^s_{K_{bs}}\}_{K_{as}}; v_1 \stackrel{?}{=} Alice, Bob, n_a \\
v_2 \Rightarrow \{n'_a\}_{n_a}; v_2 \stackrel{?}{=} \{n_a, Bob\}_{K_{as}}
\end{array}
$$

We associate to this execution the ground constraint system $\mathfrak{C}$, $\mathfrak{C} = (E_1 \rhd Alice, E_1 \rhd Bob, E_1 \rhd n_a, E_1 \cup \{n_a, Bob, K_{ab}, \{K_{ab}, Alice\}^s_{K_{bs}}\}^s_{K_{as}} \rhd \{n_a, Bob\}^s_{K_{as}}, E_1 \cup \{n_a, Bob, K_{ab}, \{K_{ab}, Alice\}^s_{K_{bs}}\}^s_{K_{as}} \cup \{n'_a\}^s_{n_a} \rhd n'_a)$ and E_1 is the initial knowledge of the intruder, $E_1 = \{Alice, Bob, s, K_{is}, n_a\}$.
We associate to this ground constraint system the following set of clauses $C_{\mathfrak{C}}$:

$$
\begin{array}{l}
\mathcal{C}_{E_1} \to I(Alice) \\
\mathcal{C}_{E_1} \to I(Bob) \\
\mathcal{C}_{E_1} \to I(n_a) \\
\mathcal{C}_{E_1} \cup I(\{n_a, Bob, K_{ab}, \{K_{ab}, Alice\}_{K_{bs}}\}^s_{K_{as}}) \to I(\{n_a, Bob\}^s_{K_{as}}) \\
\mathcal{C}_{E_1} \cup I(\{n_a, Bob, K_{ab}, \{K_{ab}, Alice\}^s_{K_{bs}}\}^s_{K_{as}}) \cup I(\{n'_a\}^s_{n_a}) \to I(n'_a)
\end{array}
$$

where $C_{E_1} = \{I(Alice), I(Bob), I(s), I(K_{is}), I(n_a)\}$. It is easy to see that $C_{\mathfrak{C}}$ is entailed by $C_{\mathcal{I}}$ where $C_{\mathcal{I}} = C_{DY} \cup C_{pre}$, and C_{DY} denotes the set of clauses representing the Dolev-Yao intruder system with explicit destructors.

5 Conclusion

We have presented in this paper an extension of the result of Basin and Ganzinger. Such extension may lead to a further extension for resolution modulo an equational theory [10, 15, 19]. We believe that the technique employed can be extended to add reflexivity or transitivity axiom to an already saturated theory, and that a consequence of our proof is that saturated theories are complete for contextual deduction [4, 14], which may help in the resolution of [9]. Another contribution of this paper is the reduction of the insecurity problem of cryptographic protocols to a ground entailment problem in the first order logic. While the application on protocols is limited to search of proofs, we believe that this application can be extended to proof of correctness, and that by including the clauses describing the protocol to the set of clauses. We believe also that one can apply this result to analyse control access policies.

References

1. Leo Bachmair and Harald Ganzinger. Completion of first-order clauses with equality by strict superposition (extended abstract). In *CTRS*, volume 516 of *Lecture Notes in Computer Science*, pages 162–180, 1991.
2. Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 19–99. Elsevier and MIT Press, 2001.
3. David Basin and Harald Ganzinger. Automated complexity analysis based on ordered resolution. *J. ACM*, 48(1):70–109, 2001.
4. Francois Bronsard and Uday S. Reddy. Conditional rewriting in focus. In M. Okada, editor, *Proceedings of the Second International Workshop on Conditional and Typed Rewriting Systems*, volume 516 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.
5. H. Comon-Lundh and V. Cortier. Security properties: Two agents are sufficient. In *ESOP*, pages 99–113, 2003.
6. H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *R. Nieuwenhuis editor, RTA, volume 2709 of Lecture Notes in Computer Science*, pages 148–164, Springer, 2003.
7. Hans de Nivelle. Using resolution as a decision procedure. Habilitation Thesis, available at `http://www.ii.uni.wroc.pl/~nivelle/publications/index.html`, 2006.
8. D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Commun. ACM*, 24(8):533–536, 1981.
9. Nachum Dershowitz and Ralf Treinen. Rta list of open problems, problem 37. `http://rtaloop.mancoosi.univ-paris-diderot.fr/problems/summary.html`, 1998.
10. Gérard Huet. *Constrained Resolution: A Complete Method for Higher Order Logic*. PhD thesis, Case Western Reserve University, 1972.
11. Mounira Kourjieh. *Logical analysis and verification of cryptographic protocols*. PhD thesis, Université Paul Sabatier Toulouse 3, dec 2009.
12. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 166–175, 2001.
13. R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.
14. Robert Nieuwenhuis and Fernando Orejas. Clausal rewriting. In *CTRS*, pages 246–258, 1990.
15. Robert Nieuwenhuis and Albert Rubio. Ac-superposition with constraints: No ac-unifiers needed. In *CADE*, pages 545–559, 1994.
16. O. Pereira and J.J. Quisquater. On the perfect encryption assumption. *Proc. of the 1st Workshop on Issues in the Theory of Security (WITS'00)*, pages 42–45, Geneva (Switzerland), 2000.
17. John Alan Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.
18. H. Lin S. Delaune and Ch. Lynch. Protocol verification via rigid/flexible resolution. In N. Dershowitz and A. Voronkov, editors, *LPAR*, volume 4790 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2007.
19. Laurent Vigneron. Associative-commutative deduction with constraints. volume 814 of *Lecture Notes in Computer Science*, pages 530–544, 1994.

Obligations with deadlines: a formalization in Dynamic Deontic Logic

Robert Demolombe[1]

Institut de Recherche en Informatique de Toulouse
France
robert.demolombe@orange.fr

Abstract. Segerberg's Dynamic Deontic Logic is a dynamic logic where among the set of all possible histories those fulfilling the norms are distinguished. An extension of this logic to obligations (respectively permissions and prohibitions) to do an action before a given deadline or during a given time interval is defined. These temporal constraints are defined by events which may have several occurrences (like the obligation to update a given file before midnight). Violations of these kinds of norms are defined in this logical framework.

1 Introduction

An important issue in computer security is to have clear definitions of the norms which are intended to influence the behavior of interacting agents (human agents and software agents).

One of the most significant concepts involved in these norms is the concept of obligation and the related concepts of prohibition and permission.

For instance, in the context of data management an information system user may have the right to access his private data in order to check its validity. That means that, by performing some request, he can create the obligation to the information system to deliver these data. However, this kind of obligation is not completely defined if we do not specify a deadline to fulfill this obligation. Indeed, without specification of this deadline it is impossible at a given moment to check whether the obligation has been violated. Moreover, it is worth noting that the definition of the deadline may be more complex than, for instance, a fixed number of days. It may be something like: data must be delivered before the end of the month where the request has occurred.

In other contexts deadlines may be involved in the definitions of prohibitions. For instance, it may be prohibited to disseminate contractors' names before a contract has been signed.

There are many papers in the literature about the formalization of regulations, or norms in general [2, 18, 11]. Some of them have a static point of view and investigate properties (for instance, the obligation for a car driver to have a licence) that ought to be fulfilled in a given fixed situation. They are usually called "*obligations to be*". Others have a dynamic point of view. According to that point of view it is important not to confuse obligation change [10, 8] on the one hand, and obligations to change a situation, that is obligations to do some actions, on the other hand. The latter are usually called "*obligations to do*".

Obligations to do combine deontic logic and dynamic logic, and there are several proposals for their formalization in the framework of modal logics [21, 23, 17, 12]. However, as far as we know, Dignum et al., in [15, 14, 13], and Broersen et al., in [5, 6], are the only researchers who have taken into account the notion of deadline.

The objective of this paper is to present new formal definitions of obligations to do and related concepts with deadlines[1]. These formal definitions are needed in the field of computer security because these concepts may be quite complex and when a team of computer scientists has to specify pieces of software to check norm violations they have to agree on the definitions.

For that purpose, we have adopted the formal framework that has been proposed by Segerberg in [23, 22, 21, 20] for obligations to do without deadlines. This framework is presented in section 2. Then it is extended to deadlines (these extensions are the new contribution of the paper). We consider two cases: the case where a deadline is specified by the occurrence of a proposition (section 3), and the case where it is specified by two propositions defining an interval for the performance of the action (section 4). In both cases the circumstances where the obligations are violated are formally defined. Finally, we compare our proposal with other similar works (section 5) and possible future research directions are mentioned in the conclusion.

2 Segerberg's dynamic deontic logic

Obligations to do raise a particular formalization problem with regard to obligations to be. The problem comes from the fact that the formula in the scope of an obligation, just as any formula in the scope of a modal operator, must denote a proposition, and actions are denoted by terms, not by propositions (see [21]). Then, the argument of an obligation to do operator cannot just be an action.

The solution proposed by Segerberg in [23, 22, 21] is to interpret an obligation to do an action as an obligation to have done this action. Since the fact that an action has been done is represented by a proposition, this fact can be in the scope of an obligation operator. This fits well our intuitions: an obligation to do an action is fulfilled when the action has been done.

The logic is defined by its semantics. The primitive notion is the notion of **point**, which is similar to a possible world in a Kripke model (sometimes we use the term "**instant**" to show that we are talking about the world at a given moment). A **path** is a sequence of points that can be understood as a given evolution of the world.

An **event type** is a finite set of paths (when there is no ambiguity we will use the term "event"). Each path can be viewed as a particular realization of the event type.

[1] A preliminary version of this work has been presented in a short paper [9].

An **individual action** is a tuple $< i, e, p >$, where i denotes an agent, e an event type and p a path in the set of paths denoted by e. Intuitively, i is the agent who realized the event type e along the path p.

A **history** is a sequence of individual actions, which can be finite or infinite, of the form:

$< i_0, e_0, p_0 >< i_1, e_1, p_1 > \ldots < i_n, e_n, p_n >$

where for every i in $[0, n-1]$, the last point of p_i is the first point of p_{i+1}. A history with no individual action is denoted by: *empty*.

We say that "i does e in h", if p is some path in e and $< i, e, p >$ is an individual action of the history h. That is, in formal terms: $\exists h', h''$ such that $h = h' < i, e, p > h''$ (where h' and h'' may be *empty*).

We use the following notations:

- ef is the complex event consisting of e immediately followed by f,
- pq is the path made up by p immediately followed by q.

We say that the histories h and h' are equivalent, and that is denoted by $h \approx h'$, iff $\exists g, g'$ [2]

$h = g < i, e_0, p_0 > \ldots < i, e_{n-1}, p_{n-1} > g'$ and

$h' = g < i, e_0 \ldots e_{n-1}, p_0 \ldots p_{n-1} > g'$

It is assumed that for any given history there is a definite set of possible continuations, where a continuation is an history. The set of complete continuations of h is denoted by: $cont(h)$, where a complete continuation is an infinite history. This set represents the set of possible futures when we are at the last point of h.

It is also assumed that for every h the set $cont(h)$ can be partitioned into two categories: the set of continuations that conform to the Norm, denoted by $norm(h)$, and those that do not. A continuation conforms the Norm if none of the obligations and none of the prohibitions are violated along this continuation.

Sometimes the histories in $norm(h)$ are called the "*ideal histories*". In formal terms it is assumed that we have:

$\forall h \; \exists norm(h) \; such \; that \; norm(h) \subseteq cont(h)$

If there is no dilemma inside the Norm we have:

$cont(h) \neq \emptyset \; \Rightarrow \; norm(h) \neq \emptyset$

which intuitively means that there is always a future continuation where the Norm is completely fulfilled.

The formal language is the language of a propositional multimodal logic [7] with the following additional modal operators:

$[H]\phi$: it is historically necessary that ϕ.

$[D]\phi$: it is deontically necessary that ϕ.

$[F]\phi$: it will always be the case that ϕ.

$[P]\phi$: it always was the case that ϕ.

The intuitive distinction between the operators H and F is that $[H]\phi$ holds iff ϕ holds for all the continuations of a given history, while $[F]\phi$ holds iff ϕ holds at every point of a given continuation. The meaning of the operator P is similar to that of the operator F except that it is evaluated for a past history.

[2] $\exists g, g'$ abbreviates $\exists g \exists g'$, and $\forall g, g'$ abbreviates $\forall g \forall g'$. Also, $e_0 \ldots e_{n-1}$ and $p_0 \ldots p_{n-1}$ respectively abbreviate $< e_0, \ldots, e_{n-1} >$ and $< p_0, \ldots, p_{n-1} >$.

The meaning of the operator D is similar to that of the operator H except that it is evaluated only for all the continuations that conform the Norm.

The satisfiability conditions are defined using the relation:

$(h, g) \models \phi$

whose intuitive meaning is: ϕ is true at a point which is the last point of the past history h, and the first point of the future history g.

If ϕ is an atomic formula A, we have $(h, g) \models A$ iff the last point of h is in the set of points that interprets A. The satisfiability conditions for logical connectives are defined as usual. For the modal operators we have:

$(h, g) \models [H]\phi$ iff $\forall g' \in cont(h)((h, g') \models \phi)$

$(h, g) \models [D]\phi$ iff $\forall g' \in norm(h)((h, g') \models \phi)$

$(h, g) \models [F]\phi$ iff $\forall g_0, g_1(g = g_0g_1 \ \Rightarrow \ (hg_0, g_1 \models \phi)$

$(h, g) \models [P]\phi$ iff $\forall h_0, h_1(h = h_0h_1 \ \Rightarrow \ (h_0, h_1g) \models \phi)$

A formula ϕ is consistent iff there exists a set of histories such that for some h and g in this set we have $(h, g) \models \phi$.

For any modal operator $[O]$, such that $O \in \{H, D, F, P\}$, the dual operator is denoted by $< O >$.

Actions are represented by terms built up with atomic actions and the constructors of sequence and non deterministic choice which have the same meaning as in Harel's Dynamic Logic [16]. The constructor of sequence is denoted by ";" and the constructor of non deterministic choice is denoted by "|".

We adopt the following notations.

$|\alpha|$: event type which is the interpretation of the action α.

$|\alpha_1; \alpha_2| \overset{\text{def}}{=} \{p_1p_2 \ : \ p_1 \in |\alpha_1| \ and \ p_2 \in |\alpha_2|\}$

$|\alpha_1|\alpha_2| \overset{\text{def}}{=} \{p \ : \ p \in |\alpha_1| \ or \ p \in |\alpha_2|\}$

For each agent i we have the event-to-proposition operators:

$does_i(\alpha)$: i is just about to do α,

$done_i(\alpha)$: i has just finished doing α.

These operators play a fundamental role since they allow to talk about actions in terms of propositions.

Their formal semantics is defined by:

$(h, g) \models does_i(\alpha)$ iff $\exists g', e, p(p \in e \wedge e = |\alpha| \wedge g \approx < i, e, p > g')$

$(h, g) \models done_i(\alpha)$ iff $\exists h', e, p(p \in e \wedge e = |\alpha| \wedge h \approx h' < i, e, p >)$

We say that: "i does α in h" iff $\exists h', h'', e, p(p \in e \wedge e = |\alpha| \wedge h \approx h' < i, e, p > h'')$.

Now it is possible to introduce the two operators that represent obligation or prohibition to do an action:

$ob_i(\alpha)$: it is obligatory for agent i to have done action α.

$fb_i(\alpha)$: it is forbidden for agent i to have done action α.

Their satisfiability conditions are:

$(h, g) \models ob_i(\alpha)$ iff $\forall g' \in cont^0(h)(\neg(i \ does \ \alpha \ in \ g') \ \Rightarrow$
$\forall f \in norm(hg') \ (i \ does \ \alpha \ in \ f))$

$(h, g) \models fb_i(\alpha)$ iff $\forall g' \in cont^0(h)(\forall f \in norm(hg') \ \neg(i \ does \ \alpha \ in \ f))$

where $cont^0(h)$ denotes the set of h finite continuations and $i \ does \ \alpha \ in \ k$ is an abbreviation for: $\exists k_1, k_2(k \approx k_1k_2 \wedge (k_1, k_2) \models does_i(\alpha))$.

The intuition of the definition of the operator $ob_i(\alpha)$ is that if it is not the case that i did α along the history g', then in all the hg' continuations conforming the norm the Norm i does α. The intuition of the definition of the operator $fb_i(\alpha)$ is that there is no continuation of the history h that conform the Norm where i does α.

The *until* operator is assigned the intuitive meaning[3]:

$(until\ \phi)\psi$: ψ holds from the point where we are until the point where ϕ holds.

Its formal definition is:

$(h, g) \models (until\ \phi)\psi$ iff $\forall g', g''(g \approx g'g'' \Rightarrow$
$((hg', g'') \models \psi \vee \exists g_0, g_1(g' \approx g_0g_1 \wedge (hg_0, g_1g'') \models \phi)))$

If there are several points in g where ϕ holds, $(until\ \phi)\psi$ guarantees that ψ holds until the point that is just before the first point where ϕ holds.

If ϕ never holds in the future, it is always the case in the future that ψ holds. Therefore we have the valid formula:

$(until\ false)\psi \leftrightarrow [F]\psi$

The operator $(before\ \phi)\psi$ is assigned the intuitive meaning:

$(before\ \phi)\psi$: ψ will hold, and it will hold before ϕ holds.

If ϕ and ψ have several occurrences in the future, $(before\ \phi)\psi$ guarantees that the first occurrence of ψ will hold before the first occurrence of ϕ. The formal definition of *before* is:

$(before\ \phi)\psi \overset{\text{def}}{=} ((until\ \psi)\neg\phi) \wedge (< F > \psi)$

Therefore we have the valid formula:

$(before\ false)\psi \leftrightarrow < F > \psi$

Then, it can be shown that we have the following valid formulas:

$ob_i(\alpha) \leftrightarrow [H](until\ done_i(\alpha))[D] < F > done_i(\alpha)$

$fb_i(\alpha) \leftrightarrow [H][F][D][F]\neg done_i(\alpha)$

From these properties the intuitive meaning of the obligation operator and of the prohibition operator can be reformulated in the following way:

$ob_i(\alpha)$: for every future history, until α has been done, α must have been done at some future point.

$fb_i(\alpha)$: for every future history, at every point of this history, α must not have been done at any future point.

3 Extension to norms with deadlines

The definitions which have been presented in the previous section are now extended to norms with deadlines.

Obligation.

We define the operator $ob_i\ (\alpha < d)$ whose intuitive meaning is:

$ob_i(\alpha < d)$: it is obligatory for agent i to have done the action α between the present point and the first occurrence of the proposition d.

Its formal definition is:

[3] Here we have slightly modified Segerberg's definition of the *until* operator.

$ob_i(\alpha < d) \overset{\text{def}}{=} [H](until\ (done_i(\alpha) \vee d))[D](before\ d)done_i(\alpha)$

According to this definition Segerberg's definition can be found as a special case where the deadline never happens, i.e. where d is equivalent to $false$. Indeed, we have the valid formula:

$ob_i(\alpha < false) \leftrightarrow [H](until\ done_i(\alpha))[D] < F > done_i(\alpha)$

Therefore, we have the valid formula:

$ob_i(\alpha < false) \leftrightarrow ob_i(\alpha)$

Prohibition.

Prohibition to do an action until a given deadline occurs is defined in a similar way through the operator $fb_i(\alpha < d)$ whose intuitive meaning is:

$fb_i(\alpha < d)$: it is forbidden for agent i to have done the action α between the present point and the first occurrence of the proposition d.

Its formal definition is:

$fb_i(\alpha < d) \overset{\text{def}}{=} [H](until\ d)[D](until\ d)\neg done_i(\alpha)$

Segerberg's definition can also be found as a special case. Indeed, we have the valid formula:

$fb_i(\alpha < false) \leftrightarrow [H][F][D][F]\neg done_i(\alpha)$

Therefore, we have the valid formula:

$fb_i(\alpha < false) \leftrightarrow fb_i(\alpha)$

Permission.

Segerberg did not propose a definition for permission. Such a definition can easily be derived from the definitions of obligation and prohibition. The permission operator is denoted by: $pm_i(\alpha < d)$, and it is intuitively defined by:

$pm_i(\alpha < d)$: it is permitted for agent i to have done the action α between the present point and the first occurrence of the proposition d.

It is formally defined by:

$pm_i(\alpha < d) \overset{\text{def}}{=} [H](until\ d) < D > (before\ d)done_i(\alpha)$

In this definition we do not have the condition: $(until\ (done_i(\alpha) \vee d))$. Instead of this condition we have: $(until\ d)$. The reason is that agent i is permitted to have done the action α several times before d.

For instance, if it is permitted to access a file before the end of the day, after the file has been accessed it remains permitted to access this file before the end of the day.

In the case where the deadline never occurs, that is, if d is $false$, we have the valid formula:

$pm_i(\alpha < false) \leftrightarrow [H][F] < D >< F > done_i(\alpha)$

The comparison of $pm_i(\alpha < false)$ with $ob_i(\alpha)$ shows that, for every future continuation, in the case of a permission, the property $< F > done_i(\alpha)$ holds at every future point for some normal continuation, while for an obligation it holds until α has been done at some future point for every normal continuation. This is consistent with our intuition.

It is worth noting that $ob_i\ (\alpha < d) \wedge fb_i(\alpha < d)$ is not an inconsistent formula (it is satisfied when $norm(hg') = \emptyset$), though it is impossible to fulfil both norms.

Finally, we can easily show that we have the valid formulas:

$ob_i\ (\alpha < d) \to [H](until\ (done_i(\alpha) \vee d))ob_i\ (\alpha < d)$

$fb_i\ (\alpha < d) \rightarrow [H](until\ d) fb_i\ (\alpha < d)$

$pm_i\ (\alpha < d) \rightarrow [H](until\ d) pm_i\ (\alpha < d)$

These properties show how norms persist.

Violations.

An obligation of the form $ob_i(\alpha < d)$ has been violated at the present instant t, if there is some instant t_1 in the past where:

- d has occurred, and

- before t_1 there was an instant t_2 where we had $ob_i(\alpha < d)$, and the action α has not been done between t_2 and the first instant after t_2 where d has occurred.

If $ob.viol(i, \alpha, d)$ is used to denote the formula that characterizes the violation of the obligation $ob_i(\alpha < d)$, we have:

$$ob.viol(i, \alpha, d) \overset{\text{def}}{=} \ < P > (d \wedge < P > (ob_i(\alpha < d) \wedge (until\ d) \neg done_i(\alpha)))$$

In the case where d is logically equivalent to $false$ the above formula can never be true, because $< P > (d)$ is always false. That means that this obligation can never be violated, and that fits our intuition.

It is worth noting that after the first occurrence of d the obligation $ob_i(\alpha < d)$ imposes no more constraint. Nevertheless, after d we can have $ob.viol(i, \alpha, d)$ if a violation has occurred in the past (before d).

Note also that, if α has been done before the instant where the obligation $ob_i(\alpha < d)$ has been created, but α has not been done after that instant, the obligation is violated. We can find real scenarios where this position is, in a first approach, questionable.

Let's imagine, for instance, a customer who has the intention to buy some good and sends a check to pay it before sending the order to buy it. Suppose that the regulation states that a good has to be paid after the good has been ordered and no later than one month after this order. According to our formal definition of a violation, the obligation has been violated. This is rather counter intuitive.

In fact this is more a theoretical problem than a practical one. The problem is not that the good has not been paid, but rather that it has been paid "too early". A formal solution to avoid this oddity could be to create the obligation to pay only if the good has not been already paid.

A prohibition of the form $fb_i(\alpha < d)$ has been violated at the present instant t if there is some instant t_1 in the past where:

- we had $fb_i(\alpha < d)$, and

- between t_1 and the first occurrence of d, α has been done.

If $fb.viol(i, \alpha, d)$ is used to denote the formula that characterizes the violation of the forbiddance $fb_i(\alpha < d)$, we have:

$$fb.viol(i, \alpha, d) \overset{\text{def}}{=} \ < P > (done_i(\alpha) \wedge < P > (fb_i(\alpha < d) \wedge (before\ d) done_i(\alpha)))$$

Note that, in the case of a prohibition, a violation may occur before d has occurred.

In the case of permissions there is no violation. Indeed, a permission does not impose to do an action, it just offers the possibility to do an action without any violation of the norms.

4 Extension to norms that apply during an interval

There are many practical examples where a norm applies during an interval which is defined by two events. For instance, it may be obligatory to update a file between 11 p.m. and 5 a.m.

In the case where d or d' may have several occurrences, it is assumed that the norm applies between the first occurrence of d (say instant t) after the instant where the norm has taken place, and the first occurrence of d' after t[4].

Finally, it is assumed that the norm does not apply at the instants where d or d' occur.

To characterize violations we have to be able to characterize the instant of the first future occurrence of d. For that purpose the operator $< F1, \phi > \psi$ is introduced. Its intuitive meaning is:

$< F1, \phi > \psi$: there exists some instant t in the future where we have ψ, and t is the first instant, since the present instant, where we have ϕ.

It is formally defined by:

$(h, g) \models < F1, \phi > \psi$ iff $\exists g_1 \exists g_2 (g \approx g_1 g_2 \wedge (hg_1, g_2) \models \phi \wedge \neg \exists g_1' \exists g_1'' (g_1 \approx g_1' g_1'' \wedge \neg (g_1'' = empty) \wedge (hg_1', g_1'' g_2) \models \phi) \wedge (hg_1, g_2) \models \psi)$

Obligation.

We introduce the obligation operator $ob_i(d < \alpha < d')$ whose intuitive meaning is:

$ob_i(d < \alpha < d')$: it is obligatory for agent i to have done the action α between the first occurrence of d and the first following occurrence of d'.

Its formal definition is:

$ob_i(d < \alpha < d') \stackrel{\text{def}}{=} [H] < F1, d > ($
$(until\ (done_i(\alpha) \vee d'))[D](before\ d')done_i(\alpha))$

This definition can be read:
for every future history:
- there exists some instant t in the future where:
– d has occurred for the first time, and
– until we have $done_i(\alpha) \vee d'$:
— for every ideal continuation: α has been done before d'.

The reason why in this definition we need the operator $< F1, d >$ instead of $< F >$, is that we have to refer to the instant which is the first future occurrence of d, and not just to some future instant.

Prohibition.

We introduce the prohibition operator $fb_i(d < \alpha < d')$ whose intuitive meaning is:

$fb_i(d < \alpha < d')$: it is forbidden for agent i to have done the action α between the first occurrence of d and the first following occurrence of d'.

Its formal definition is:

[4] Since we have to consider "the first occurrence of d' after t" we cannot trivially define norms that apply in an interval from norms that apply *after* the first occurrence of d and *before* the first occurrence of d' because it may happen that the first occurrence of d' occurs before the first occurrence of d.

$fb_i(d < \alpha < d') \overset{\text{def}}{=} [H] < F1, d > ($
$(until\ d')[D](until\ d')\neg done_i(\alpha))$

This definition can be read as:
for every future history:
- there exists some instant t in the future where:
– d has occurred for the first time, and
– until we have d':
— for every ideal continuation: α has not been done before d'.

Permission.

We introduce the permission operator $pm_i(d < \alpha < d')$ whose intuitive meaning is:

$pm_i(d < \alpha < d')$: it is permitted for agent i to have done the action α between the first occurrence of d and the first following occurrence of d'.

Its formal definition is:
$pm_i(d < \alpha < d') \overset{\text{def}}{=} [H] < F1, d > ($
$(until\ d') < D > (before\ d')done_i(\alpha))$

This definition can be read:
for every future history:
- there exists some instant t in the future where:
– d has occurred for the first time, and
– until d':
— there exists a future ideal continuation where α has been done before d'.

Violations.

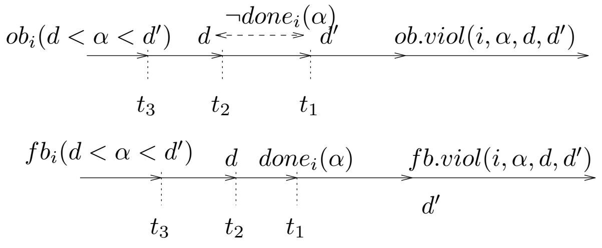

Fig. 1. Violations of obligation to do and prohibition to do.

A violation of the obligation $ob_i(d < \alpha < d')$ is denoted by: $ob.viol(i, \alpha, d, d')$. We formally have:

$ob.viol(i, \alpha, d, d') \overset{\text{def}}{=} < P > (d' \wedge < P > (d \wedge$
$< P > (ob_i(d < \alpha < d') \wedge < F1, d > ((until\ d')\neg done_i(\alpha)))))$

This characterization of a violation can be read as (see figure 1):
- there exists an instant t_1 in the past where we had d', and
- before t_1 we had d at the instant t_2, and
- before t_2 we had $ob_i(d < \alpha < d')$ at the instant t_3, and

- since the first occurrence of d after t_3, until the first occurrence of d', α has never been done.

A violation of the prohibition $fb_i(d < \alpha < d')$ is denoted by: $fb.viol(i, \alpha, d, d')$. We formally have:

$$fb.viol(i, \alpha, d, d') \overset{\text{def}}{=} \; < P > (done_i(\alpha) \wedge$$
$$< P > (d\wedge < P > (fb_i(d < \alpha < d')\wedge < F1, d > ((before\ d')done_i(\alpha)))))$$

This characterization of a violation can be read (see figure 1):
- there exists an instant t_1 in the past where we had $done_i(\alpha)$, and
- before t_1 we had d at the instant t_2, and
- before t_2 we had $fb_i(d < \alpha < d')$ at the instant t_3, and
- after the first occurrence of d after t_3 we had $done_i(\alpha)$ before d'.

5 Comparison with other works

To compare our work with other approaches that can be found in the literature, we can mention Pörn [19], who defines formulas of the form: $OE_i\phi$, whose meaning is: it is obligatory for agent i to bring it about that ϕ. However, the operator E_i does not mention which action has been done and there is no formal means to express that the effect ϕ should be obtained before a given deadline. Then, the obligation is violated as soon as we have $\neg E_i\phi$, and the agent i has no delay to perform some appropriate action.

In [17], Horty and Belnap present a logic where histories have a tree structure, which has some similarities with Segerberg's structures. The main difference is that, like for Pörn, actions are not explicitly represented. They define the operator $[i\ dstit : \phi]$ whose intuitive meaning is rather close to $E_i\phi$, but its semantics makes explicit reference to histories and instants. Deontic notions are represented via the deontic modal operator O, and $O[i\ dstit : \phi]$ means that it is obligatory for agent i to see to it that ϕ. Nevertheless, the concept of deadline is ignored.

As mentioned in the introduction, Dignum et al.[15] are among the few people who have explicitly investigated norms with deadlines. They have defined a temporal logic with tree structures and the operator $E_i\phi$. However, the formal semantics given to this operator is quite different from Pörn's semantics, and, in our view, it is not perfectly clear. Norms are formalized according to Anderson's reductionist approach [1] thanks to a specific proposition that means: "a violation has occurred". Formulas that express obligations to do with deadlines have the form: $O_i(\phi < d)$. Their meaning is: for all future histories, either, until d has occurred, agent i has brought about that ϕ and there is no violation, or, at the next instant, we have d and there is a violation. The strong limitation of this formalization is that in the reductionist approach there is no explicit representation of ideal histories. Moreover deontic modalities cannot be nested with other modal operators. For instance, in the context of a psychiatric hospital, we would like to be able to represent the fact that for some patients it is prohibited to know that it is prohibited to access their own file.

In Broersen et al. [5], formulas of the form $O_i(\phi, d)$ mean that it is obligatory for agent i that ϕ holds before the deadline d. That could be reformulated in Segerberg's framework as: $[H](until\ (\phi \vee d))O_i\phi$, where O_i obeys a KD logic. But there is no explicit reference to the actions, and we see this operator as a representation of obligation to be (rather than obligation to do an action) with deadlines. In [6], there is no deontic modality and norms are formalized by their violation à la Anderson. We have seen above the limitations of this approach.

Finally, in [3], Äqvist presents a logic that combines temporal and deontic operators. In [4], he suggests to extend it with deadlines. The idea is, for example, to represent that it is obligatory to have ϕ before t_3 by a formula that means that is it obligatory to have ϕ either in t_1 or t_2 or t_3. This solution may be interesting for some particular applications but it lacks generality. Moreover there is no explicit representation of actions, and, again, the resulting logic would be closer to a logic for obligations to be than to a logic for obligations to do.

6 Conclusion

We have extended Segerberg's logical framework to obligations, prohibitions and permissions to do an action before a proposition comes to be true or within an interval defined by two propositions. In each case, we have also formally characterized the circumstances where these norms are violated.

The resulting definitions of norms and of their violations are far from being simple and that is why, as mentioned in the introduction, we need formal definitions to prevent misunderstanding between people who have to design computer systems which have to fulfil the norms about security.

Nevertheless these results have to be considered as a preliminary step which clarify semantical issues but should be completed by further works in several directions.

The first direction is about the semantics of other notions involved in norm definitions. In this direction we should extend the definitions to *obligations to be* with deadlines (for example, the obligation to have digital data for identification on your passport before the end of the year). Another direction is the obligation to have started the performance of an action. For example, the obligation, for a driver, to start to cross the road when the light turns to green. It seems that this kind of norms could be formalized thanks to formulas of the form: $does_i(\alpha)$.

The second direction is the automatization of reasoning about norms with deadlines. This requires to define a sound and complete axiomatics of the semantics which has been presented, to analyze its formal properties, like decidability, to design strategies for automated reasoning and to analyze their complexity in the perspective of potential implementations.

Acknowledgement. We would like to thank Lennart Aqvist, Krister Segerberg, Andreas Herzig and an anonymous referee for their fruitful comments. If there are any mistakes they are the only responsibility of the author.

References

1. A. R. Anderson. A reduction of deontic logic to alethic modal logic. *Mind*, 67, 1958.
2. L. Aqvist. Deontic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 2. Reidel, 1984.
3. L. Aqvist. Combination of tense and deontic modality. In A. Lomuscio and D. Nute, editors, *Proceedings of the 7th International Workshop on Deontic Logic in Computer Science*. Springer, LNAI 3065, 2004.
4. L. Aqvist. Private communication, 2004.
5. J. Broersen, M. Dastani, and L. van der Torre. BDIO-CTL: obligations and the specification of agent behaviour. In *International Joint Conference on Artificial intelligence*, 2003.
6. J. Broersen, F. Dignum, V. Dignum, and J-J. C. Meyer. Designing a deontic logic of deadlines. In A. Lomuscio and D. Nute, editors, *Proceedings of the 7th International Workshop on Deontic Logic in Computer Science*. Springer, LNAI 3065, 2004.
7. B. F. Chellas. *Modal Logic: An introduction*. Cambridge University Press, 1988.
8. R. Demolombe. De l'évolution des croyances à l'évolution des obligations dans le calcul des situations. In A. Herzig, B. Chaib-draa, and P. Mathieu, editors, *Actes des secondes Journées Francophones sur les Modèles Formels de l'Interaction*. Cépaduès-Editions, 2003.
9. R. Demolombe, P. Bretier, and V. Louis. Norms with deadlines in Dynamic Deontic Logic (short paper). In G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, editors, *Proceedings of the 17th European Conference on Artificial Intelligence*. IOS Press, 2006.
10. R. Demolombe and A. Herzig. Obligation change in Dependence Logic and Situation Calculus. In A. Lomuscio and D. Nute, editors, *Proceedings of the 7th International Workshop on Deontic Logic in Computer Science*. Springer, LNAI 3065, 2004.
11. R. Demolombe and R. Hilpinen. Special issue on Deontic Logic in Computer Science. *Fundamenta Informaticae*, 48(2-3), 2001.
12. R. Demolombe and A.J. Jones. Actions and normative positions. A modal-logical approach. In D. Jacquette, editor, *Companion to Philosophical Logic*. Blackwell, 2002.
13. F. Dignum and R. Kuiper. Combining dynamic deontic logic and temporal logic for the specification of deadlines. In *Thirties HICSS*, 1997.
14. F. Dignum, H. Weigand, and E. Verharen. Meeting the deadline: on the formal specification of temporal deontic constraints. In *International Symposium of Management of Intelligent Systems*, 1996.
15. V. Dignum, J-J. Meyer, F. Dignum, and H. Weigand. Formal specification of interaction in agent societies. In *Second Goddard workshop on Formal Approaches to Agent-Based Systems*, 2002.
16. D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 2. Reidel, 1984.
17. J.F. Horty and N. Belnap. The deliberative STIT: a study of action, omission, ability, and obligation. *Journal of Philosophical Logic*, 24:583–644, 1995.
18. A. Lomuscio and D. Nute. *Deontic Logic in Computer Science*. Springer, LNAI 3065, 2004.

19. I. Pörn. Action Theory and Social Science. Some Formal Models. *Synthese Library*, 120, 1977.
20. K. Segerberg. Private communication, 2003.
21. K. Segerberg. Some Meinong/Chisholm thesis. In K. Segerberg and K. Sliwinski, editors, *Logic, Law, Morality. A festschrift in honor of Lennart Aqvist*, volume 51, pages 67–77. Uppsala Philosophical Studies, 2003.
22. K. Segerberg. Intension, Intention. In R. Kahle, editor. CSLI Publications, 2004.
23. K. Segerberg. A blueprint for deontic logic in three (not necessarily easy) steps. In Giacomo Bonanno, James Delgrande, Jérôme Lang, and Hans Rott, editors, *Formal Models of Belief Change in Rational Agents*, number 07351 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.

Logics for Access Control: A Conditional Approach

Valerio Genovese[1], Laura Giordano[2], Valentina Gliozzi[3], and Gian Luca Pozzato[3]

[1] University of Luxembourg and Università degli Studi di Torino - Italy
valerio.genovese@uni.lu
[2] Dipartimento di Informatica - Università del Piemonte Orientale - Alessandria, Italy
laura@mfn.unipmn.it
[3] Dipartimento di Informatica - Università degli Studi di Torino - Torino, Italy
{gliozzi,pozzato}@di.unito.it

Abstract. In this paper we provide a reconstruction of access control logics within constructive conditional logics, by regarding the assertion A **says** ϕ, whose intended meaning is that *principal* A *says that* ϕ, as a conditional implication. We identify the conditional axioms needed to capture the basic properties of the "says" operator and to provide a proper definition of boolean principals. Most of these axioms are standard axioms of conditional logics. We provide a Kripke model semantics for the logic and we prove that the axiomatization is sound and complete with respect to the semantics. Also, we define a sound, complete and cut-free labelled sequent calculus for it.

1 Introduction

Access control is concerned with the decision of accepting or denying a request from a *principal* (e.g., user, program) to do an operation on an object. In practice, an access control system is a product of several, often independent, distributed entities with different policies that interact in order to determine access to resources. In order to specify and reason about such systems, many formal frameworks have been proposed [1–5].

A common feature of most well-known approaches is the employment of constructive logics enriched with formulas of the form A **says** φ, intuitively meaning that the principal A *asserts* or *supports* φ to hold in the system. In [6] it is shown that an intuitionistic interpretation of the modality "says" allows to avoid unexpected conclusions that are derivable when **says** is given an axiomatization in classical logic.

In [7] an access control logic, ICL, is defined as an extension of intuitionistic propositional logic, in which the operator **says** is given a modal interpretation in the logic S4. The treatment of the operator **says** as a modality can be also found in [8], which introduces a logical framework, FSL, based on multi-modal logic methodology.

Even if there is some agreement on looking at the says construct as a modal operator, the *correspondence theory* between its axiomatizations and the underlying (Kripke-syle) semantics is left unexplored. By correspondence theory we mean identifying canonical properties for well-known access control axioms, i.e., first-order conditions of Kripke structures that are *necessary* and *sufficient* for the corresponding axiom to hold. This approach raise several challenges because axiom of access control are not standard in modal literature and their correspondence with the underlying semantics is mainly unexplored. Identifying canonical properties for well-known axioms for access control permits to study them separately and naturally yields completeness for logics that adopt *any* combination of them. This methodology is significant if we want logic to be employed to compare different access control models, because different systems adopts different axioms depending on the specific application domain.

In this paper we show that conditional logics [9] can provide a natural framework to define axiomatization, semantics and proof methods for access control logics. We present an intuitionistic logic, $\mathsf{Cond}_{\mathrm{ACL}}$, which integrates access control logics with conditional logics. We formalize the **says** operator as a conditional normal modality so that A **says** ϕ is regarded as a conditional implication $A \Rightarrow \phi$, meaning that proposition ϕ holds in all the preferred worlds for the principal A. The generality of this approach allows a natural formalization of boolean principals [7], that is, principals which are formed by boolean combination of atomic principals.

From the access control point of view, the **says** operator satisfies some basic axioms of access control logics [7, 10]. We define a sound and complete Kripke semantics for $\mathsf{Cond}_{\mathrm{ACL}}$ as well as a sound and complete cut-free sequent calculus for it.

The paper is structured as follows. In Section 2 we introduce the axiomatization and the semantics for the intuitionistic conditional logic $\mathsf{Cond}_{\mathrm{ACL}}$ and we compare it with existing approaches. In Section 3 we show that the axiomatization is sound and complete with respect to the semantics. In Section 4 we define a cut-free sequent calculus for $\mathsf{Cond}_{\mathrm{ACL}}$. Section 6 contains the conclusions and a discussion of related work.

2 The logic $\mathsf{Cond}_{\mathrm{ACL}}$

In this section, we introduce the conditional intuitionistic logic $\mathsf{Cond}_{\mathrm{ACL}}$ for access control by defining its axiomatization and Kripke semantics. The formulation of the "says" modality as a conditional operator allows boolean principals to be modelled in a natural way, since in a conditional formula A **says** ϕ, both A and ϕ are arbitrary formulas. For instance, we can write, $A \wedge B$ **says** ϕ to mean that principals A and B jointly say that ϕ, and $A \vee B$ **says** ϕ to mean that principals A and B independently say that ϕ. Indeed, conditional logics provide a natural generalization of multimodal logics to the case when modalities are labelled by arbitrary formulas.

2.1 Axiom System

We define the language $\mathcal{L}$ of the logic $\mathsf{Cond}_{\mathrm{ACL}}$. Let ATM be a set of atomic propositions. The formulas of $\mathcal{L}$ are defined inductively as follows: if $P \in ATM$, then $P \in \mathcal{L}$; $\bot \in \mathcal{L}$, where $\bot$ is a proposition which is always false; if A, φ, φ_1 and φ_2 are formulas of $\mathcal{L}$, then $\neg\varphi$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \rightarrow \varphi_2$, and A **says** φ are formulas of $\mathcal{L}$.

The intended meaning of the formula A **says** φ, where A and φ are arbitrary formulas, is that *principal A says that φ*, namely, "the principal A asserts or supports φ" [7]. Although the principal A is an arbitrary formula, in order to stress the fact that a formula is playing the role of a principal, we will denote it by $A, B, C, \ldots$ while we will use greek letters for arbitrary formulas.

The axiomatization of $\mathsf{Cond}_{\mathrm{ACL}}$ contains few basic axioms for access control logics [7, 6], as well as few additional axioms governing the behavior of boolean principals.

Basic Axioms. The *axiom system* of the logic $\mathsf{Cond}_{\mathrm{ACL}}$ contains the following axioms and inference rules, which are intended to capture the basic properties of the **says** operator.

(TAUT)	all tautologies of intuitionistic logic
(K)	A **says** $(\alpha \rightarrow \beta) \rightarrow (A$ **says** $\alpha \rightarrow A$ **says** $\beta)$
(UNIT)	$\alpha \rightarrow (A$ **says** $\alpha)$
(C)	A **says** $(A$ **says** $\alpha \rightarrow \alpha)$
(MP)	If $\vdash \alpha$ and $\vdash \alpha \rightarrow \beta$ then $\vdash \beta$
(RCEA)	If $\vdash A \leftrightarrow B$ then $\vdash (A$ **says** $\gamma) \leftrightarrow (B$ **says** $\gamma)$
(RCK)	If $\vdash \alpha \rightarrow \beta$ then $\vdash (A$ **says** $\alpha) \rightarrow (A$ **says** $\beta)$

We say that a formula α is a theorem of the logic, and write $\vdash \alpha$ if there is a derivation of α from the above axioms and rules. We say that α can be derived from a set of formulas Γ, and write $\Gamma \vdash \alpha$, if there are $\gamma_1, \ldots \gamma_n$ in Γ such that $\vdash \gamma_1 \wedge \ldots \wedge \gamma_n \rightarrow \alpha$. The rule (MP) is modus ponens. (RCEA) and (RCK) are standard inference rules for conditional logics. (RCK) plays the role of the rule of Necessitation (if $\vdash \phi$ then $\vdash \Box\phi$) in modal/multimodal logic. The axiom (K) belongs to the axiomatization of all normal modal logics and it is derivable in "normal" conditional logics. (UNIT) and (K) are the characterizing axioms of the access control logics ICL [7], while (C) has been included in the axiomatization of the logic DTL_0 in [10].

Axioms for boolean principals. The axioms introduced above do not enforce by themselves any intended property of boolean principals. In this subsection, we discuss the properties that are intended for boolean principals and we introduce axioms which capture such properties. Specifically, we focus on the intended meaning of conjunctions and disjunctions among principals.

Our interpretation of the statement $A \wedge B$ **says** ϕ is that A *and* B *jointly (combining their statements) say that* ϕ. It comes from the interpretation of the statement as a conditional implication: A and B (jointly) conditionally prove ϕ. Instead, our interpretation of the statement $A \vee B$ **says** ϕ is that A *and* B *disjointly (independently) say that* ϕ, which comes from the reading of the conditional formula as A and B (disjointly) conditionally prove ϕ.

Concerning the statement $A \vee B$ **says** ϕ, we expect that if both A says ϕ and B says ϕ, then A and B disjointly (independently) say that ϕ. This property can be captured by the following axiom:

$$A \textbf{ says } \phi \wedge B \textbf{ says } \phi \rightarrow A \vee B \textbf{ says } \phi$$

which corresponds to the well known axiom (CA) of conditional logics [9]. Similarly, we can expect that the converse axiom

$$A \vee B \textbf{ says } \phi \rightarrow A \textbf{ says } \phi \wedge B \textbf{ says } \phi$$

holds. The two axioms together enforce the property that A and B disjointly say that ϕ if and only if A says that ϕ and B says that ϕ .

Concerning the statement $A \wedge B$ **says** ϕ, we expect that A and B jointly say that ϕ when either A or B says that ϕ. This condition can be enforced by introducing the axiom

$$A \textbf{ says } \phi \rightarrow A \wedge B \textbf{ says } \phi$$

which, although is a very controversial axiom of conditional logics, called monotonicity[4], is proved to be armless in this intuitionistic setting. We would like to have the property that if $A \wedge B$ **says** ϕ then, by combining the statements of A and B, ϕ can be concluded. This is not equivalent to saying that either A says ϕ or B says ϕ. Indeed, this last property would correspond to axiom $(A \wedge B \textbf{ says } \phi) \rightarrow (A \textbf{ says } \phi) \vee (B \textbf{ says } \phi)$, which is too strong and not wanted. In the following we show that this property can be captured in a first order axiomatization.

Although we do not put any restriction to the language, in the following we will limit our consideration to principals obtained by boolean combination (conjunction and disjunction) of atomic principals and of the principal $\perp$. Although a principal can be an arbitrary propositional formula (including negation and implication), no specific properties are intended for such formulas, and no specific axioms are introduced for them.

The axiomatization of $\mathsf{Cond}_{\mathrm{ACL}}$ includes (in addition) the following axioms:

(CA)	$A \textbf{ says } \phi \wedge B \textbf{ says } \phi \rightarrow A \vee B \textbf{ says } \phi$
(CA-conv)	$A \vee B \textbf{ says } \phi \rightarrow A \textbf{ says } \phi$
(Mon)	$A \textbf{ says } \phi \rightarrow A \wedge B \textbf{ says } \phi$
(DT)	$A \wedge B \textbf{ says } \phi \rightarrow (A \textbf{ says } (B \rightarrow \phi))$
(ID)	$A \textbf{ says } A$

The first three axioms are those introduced above. (DT) and (ID) are used together to enforce the property that if $A \wedge B$ **says** ϕ then, by combining the statements of A and B, ϕ can be concluded. The two axioms allow propositions representing principals to occur on the right of the **says** modality. The intended meaning of (DT) is that, if $A \wedge B$ **says** ϕ, then A says that ϕ holds in all B worlds (worlds visible to the principal B). The meaning of (ID) is that "A says that principal A is visible". We will come back to the meaning of these axioms when describing the semantic conditions associated with the axioms.

It can be shown that:

Theorem 1. *The above axiomatization is consistent.*

Proof. Consistency immediately follows from the fact that, by replacing A **says** B with the intuitionistic implication $A \rightarrow B$, we obtain axioms which are derivable in intuitionistic logic.

□

[4] In general, conditional logics only allow weaker forms of monotonicity, encoded, for instance, by the axiom (CV) of Lewis' logic VC.

Let us observe that the above interpretation of conjunction and disjunction between principals is different from the one given in the logic ICL$^{\mathcal{B}}$ [7], which actually adopts the opposite interpretation of $\wedge$ and $\vee$: in Garg and Abadi's logic ICL$^{\mathcal{B}}$, $A \wedge B$ **says** ϕ is the same as A **says** $\phi \wedge B$ **says** ϕ, while $A \vee B$ **says** ϕ means that, by combining the statements of A and B, ϕ can be concluded. Due to this, let us say, superficial difference, the properties of the principal $A \wedge B$ in our logic are properties of the principal $A \vee B$ in their logic, and vice-versa, the properties of the principal $A \vee B$ in our logic are properties of the principal $A \wedge B$ in their logic.

Observe that the axioms, (trust), (untrust) and (cuc') of the logic ICL$^{\mathcal{B}}$ are not derivable from our axiomatization. Also, the addition of the axiom (untrust) $\top$ **says** $\bot$ to our axiomatization would entail that for all principals A, A **says** $\bot$, which is an unwanted property.

2.2 Semantics

The semantics of the logic $\mathsf{Cond}_{\mathrm{ACL}}$ is defined as follows.

Definition 1. *A* $\mathsf{Cond}_{\mathrm{ACL}}$ *model has the form* $\mathcal{M} = (S, \leq, \{R_A\}, h)$ *where:* $S \neq \emptyset$ *is a set of items called worlds;* $\leq$ *is a partial order over S;* R_A *is a binary relation on* S *associated with the formula A;* h *is an evaluation function* $ATM \longrightarrow Pow(S)$ *that associates to each atomic proposition* P *the set of worlds* x *in which* P *is true.*

We define the truth conditions of formulas with respect to worlds in a model $\mathcal{M}$*, by the relation* $\mathcal{M}, x \models \phi$*, as follows. We use* $[|\phi|]$ *to denote* $\{y \in S \mid \mathcal{M}, y \models \phi\}$.

1. $\mathcal{M}, t \models P \in ATM$ *iff, for all s such that* $t \leq s$, $s \in h(P)$
2. $\mathcal{M}, t \models \varphi \wedge \psi$ *iff* $\mathcal{M}, t \models \varphi$ *and* $\mathcal{M}, t \models \psi$
3. $\mathcal{M}, t \models \varphi \vee \psi$ *iff* $\mathcal{M}, t \models \varphi$ *or* $\mathcal{M}, t \models \psi$
4. $\mathcal{M}, t \models \varphi \rightarrow \psi$ *iff for all s such that* $t \leq s$ *(if* $\mathcal{M}, s \models \varphi$ *then* $\mathcal{M}, s \models \psi$*)*
5. $\mathcal{M}, t \models \neg\varphi$ *iff, for all* s *such that* $t \leq s$, $\mathcal{M}, s \not\models \varphi$
6. $\mathcal{M}, t \not\models \bot$
7. $\mathcal{M}, t \models A$ **says** ψ *iff, for all* s *such that* tR_As, $\mathcal{M}, s \models \psi$.

We say that ϕ *is valid in a model* $\mathcal{M}$ *if* $\mathcal{M}, t \models \phi$ *for all* $t \in S$*. We say that* ϕ *is* valid tout court *(and write* $\models \phi$*) if* ϕ *is valid in every model. We extend the notion of validity to a set of formulas* Γ *in the obvious way: for all* t, $\mathcal{M}, t \models \Gamma$ *if* $\mathcal{M}, t \models \psi$ *for all* $\psi \in \Gamma$*. Last, we say that* ϕ *is a* logical consequence *of* Γ *(and write* $\Gamma \models \phi$*) if, for all models* $\mathcal{M}$*, for all worlds* t*, if* $\mathcal{M}, t \models \Gamma$*, then* $\mathcal{M}, t \models \phi$.

The relations $\leq$ and R_A must satisfy the following conditions:

(S-Int) $\forall t, s, z \in S$, if $s \leq t$ and tR_Az then sR_Az;
(S- UNIT) $\forall t, s \in S$, if sR_At, then $s \leq t$;
(S-C) $\forall t, s, z \in S$, if sR_At and $t \leq z$, then zR_Az;
(S-CA) $R_{A \vee B}(t) = R_A(t) \cup R_B(t)$.
(S-Mon) $\forall t, s, z \in S$, if $sR_{A \wedge B}t$, then sR_At and sR_Bt;
(S-DT) $\forall t, s, z \in S$, if sR_At and $t \leq z$, and $z \in [|B|]$, then $sR_{A \wedge B}z$;
(S-ID) $\forall t, s \in S$, if sR_At, then $t \in [|A|]$;

(S-RCEA) if $[|A|] = [|B|]$, then $R_A = R_B$.

Condition (S-Int) enforces the property that when a formula A **says** ϕ true in a world t, it is also true in all worlds reachable from s by the relation $\leq$ (i.e., in all worlds s such that $t \leq s$). All the other semantic conditions are those associated with the axioms of the logic, apart from condition (S-RCEA), which is the well-known condition for normality in conditional logics, claiming that the accessibility relation R_A is associated with the semantic interpretation of A. (S-CA) is the semantic condition for both axioms (CA) and its converse.

Observe that, in the semantics above, the binary relation R_A plays the role of the selection function f, which is used in most formulations of conditional logic semantics. In particular, sR_At corresponds to $t \in f([|A|], s)$, and the conditions above are indeed conditions on the selection function f, as usual in conditional logics. Note also that the semantic conditions for some of the axioms, as for instance (DT), slightly departs from the semantic condition usually given to these axioms in conditional logic. This is due to the fact that $\mathsf{Cond}_{\mathrm{ACL}}$ is an intuitionistic conditional logic and the implication occurring within axioms is intuitionistic implication.

Concerning the interpretation of boolean conditionals and, in particular, of the conjunction between principals, it can be proved that, from the semantic conditions of (Monotonicity), (ID) and (DT) it follows that:

$$R_{A \wedge B}(t) = R_A(t) \cap R_B(t).$$

It is worth noticing that the notion of logical consequence defined above can be used to verify that a request φ of a principal A is compliant with a set of policies. Intuitively, given a set of formulas Γ representing policies, we say that A is compliant with Γ iff Γ, A **says** $\phi \models \phi$. For instance, if Γ contains the following formulas:

$Admin1$ **says** $(SU\_user1 \rightarrow write\_perm\_user1)$
$Admin2$ **says** SU_user1
$((Admin1 \wedge Admin2)$ **says** $delete\_file1) \rightarrow delete\_file1$
$Admin1 \wedge Admin2$ **says** $((write\_perm\_user1 \wedge user1$ **says** $delete\_file1) \rightarrow delete\_file1)$
$user1$ **says** $delete\_file1$

we obtain that $\Gamma, user1$ **says** $deletefile1 \models delete\_file1$.

3 Soundness and Completeness

In this section we prove that the axiomatization of the logic $\mathsf{Cond}_{\mathrm{ACL}}$ given above is sound and complete with respect to the semantics of Definition 1.

Theorem 2 (Soundness). *Given a formula $\varphi \in \mathcal{L}$, if $\vdash \varphi$, then $\models \varphi$.*

Proof. It is easy to prove that each axiom is a valid formula and, for each inference rule, if the antecedent of the rule is a valid formula, the consequence of the rule is also a valid formula.

□

The completeness proof we present is based on the proof of completeness for the Kripke semantics of intuitionistic logic in [11] and extends it to deal with the modalities **says** in the language and, more precisely, with the interplay between the relation $\leq$ and the accessibility relations R_A associated with the modalities.

Definition 2 (Consistency). *Let Γ be a set of well formed formulas. Γ is consistent iff $\Gamma \nvdash \bot$. If Γ has an infinite number of formulas, we say that Γ is consistent iff there are no finite $\Gamma_0 \subset \Gamma$ such that $\Gamma_0 \vdash \bot$.*

Definition 3 (Saturation). *Let Γ be a set of well formed formulas, we say that Γ is saturated iff 1. Γ is consistent; 2. if $\Gamma \vdash \varphi$, then $\varphi \in \Gamma$; 3. if $\Gamma \vdash \varphi \vee \psi$, then $\Gamma \vdash \varphi$ or $\Gamma \vdash \psi$.*

Lemma 1 (Saturated Extensions). *Let Γ be a set of well formed formulas. Suppose $\Gamma \nvdash \varphi$, then there is a saturated extension Γ^* such that $\Gamma^* \nvdash \varphi$.*

Lemma 2. *Let Γ be a set of formulas and let $\Delta = \{\varphi : A$ **says** $\varphi \in \Gamma\}$. If $\Delta \vdash \psi$, then $\Gamma \vdash A$ **says** ψ.*

Proof. Suppose there is a derivation of ψ from Δ. Then, there must be a finite set of formulas $\{\varphi_1, \ldots, \varphi_n\} \subseteq \Delta$ such that $\{\varphi_1, \ldots, \varphi_n\} \vdash \psi$. By definition of $\vdash$, $\vdash \varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \psi$. By (RCK) and (K), $\vdash A$ **says** $\varphi_1 \wedge \ldots \wedge A$ **says** $\varphi_n \rightarrow A$ **says** ψ, and from definition of $\vdash$ (and since A **says** $\varphi_i \in \Gamma$ for all $i = 1, \ldots, n$) we conclude that $\Gamma \vdash A$ **says** ψ.

□

Definition 4 (Canonical model construction). *Let Γ_0 be any saturated set of formulas. Then we define $\mathbf{M} = (S, \leq, \{R_A\}, h)$ such that: S is the set of all saturated $\Gamma \supseteq \Gamma_0$; $\Gamma_1 \leq \Gamma_2$ iff $\Gamma_1 \subseteq \Gamma_2$; $\Gamma_1 R_A \Gamma_2$ iff $\{\alpha \mid A$ **says** $\alpha \in \Gamma_1\} \subseteq \Gamma_2$; for all $P \in ATM$, $h(P) = \{\Gamma \in S \mid P \in \Gamma\}$.*

Lemma 3. *For all $\Gamma \in S$ and each wff formula φ, we have that $\mathbf{M}, \Gamma \models \varphi$ iff $\varphi \in \Gamma$.*

Proof. By induction on the complexity of φ. In case φ is an atomic formula, the lemma holds by definition. For $\varphi \equiv \phi \wedge \psi$ the proof is easy and left to the reader. For $\varphi \equiv \phi \vee \psi$, then $\Gamma \models \phi \vee \psi \Leftrightarrow (\Gamma \models \phi$ or $\Gamma \models \psi) \Leftrightarrow (\phi \in \Gamma$ or $\psi \in \Gamma) \Leftrightarrow \phi \vee \psi \in \Gamma$ (by the saturation of Γ). For $\varphi \equiv \phi \rightarrow \psi$, suppose $\Gamma \models \phi \rightarrow \psi$. Then for all saturated $\Gamma' \supset \Gamma$ we have that if $\Gamma' \models \phi$, then $\Gamma' \models \psi$. Assume $\Gamma \nvdash \phi \rightarrow \psi$, then $\Gamma \cup \{\phi\} \nvdash \psi$; let Γ' be a saturated extension of $\Gamma \cup \{\phi\}$ such that $\Gamma' \nvdash \psi$, then $\Gamma' \models \phi$ but not $\Gamma' \models \psi$ (induction hypothesis); this contradicts $\Gamma \models \phi \rightarrow \psi$, hence $\Gamma \vdash \phi \rightarrow \psi$. As Γ is saturated, by condition 2 in Definition 3, $\phi \rightarrow \psi \in \Gamma$. The converse is trivial. For $\varphi \equiv A$ **says** ϕ, suppose $\Gamma \models A$ **says** ϕ. Hence, for all Γ' such that $\Gamma R_A \Gamma'$, $\Gamma' \models \phi$. By inductive hypothesis, $\phi \in \Gamma'$. Let $\Delta = \{\alpha : A$ **says** $\alpha \in \Gamma\}$. By construction, $\Gamma' \supseteq \Delta$. Assume, for a contradiction, that A **says** $\phi \notin \Gamma$. By condition 2 in Definition 3, $\Gamma \nvdash A$ **says** ϕ. Then, by Lemma 2, $\Delta \nvdash \phi$. By Lemma 1, there is a saturated extension Δ^* of Δ such that $\Delta^* \nvdash \phi$, i.e. $\phi \notin \Delta^*$. By definition of R_A, $\Gamma R_A \Delta^*$. This contradicts the fact that, for all Γ' such that $\Gamma R_A \Gamma'$, $\phi \in \Gamma'$. The converse is trivial.

□

Lemma 4. *Let* $\mathbf{M}$ *be the canonical model as defined in Definition 4.* $\mathbf{M}$ *satisfies the semantic conditions (S-Int), (S-UNIT), (S-C), (S-CA), (S-Mon), (S-DT), (S-ID), and (S-RCEA).*

Proof. We have to prove that

(S-Int) $\forall \Gamma, \Gamma', \Gamma'' \in S$, if $\Gamma \leq \Gamma'$ and $\Gamma' R_A \Gamma''$ then $\Gamma R_A \Gamma''$
(S-UNIT) $\forall \Gamma, \Gamma' \in S$, if $\Gamma R_A \Gamma'$ then $\Gamma \leq \Gamma'$.
(S-C) $\forall \Gamma, \Gamma', \Gamma'' \in S$, if $\Gamma R_A \Gamma'$, and $\Gamma' \leq \Gamma''$, then $\Gamma'' R_A \Gamma''$
(S-CA) $\forall \Gamma, \Gamma' \in S$, $\Gamma R_A \Gamma'$ or $\Gamma R_B \Gamma'$, iff $\Gamma R_{A \vee B} \Gamma'$
(S-Mon) $\forall \Gamma, \Gamma' \in S$, if $\Gamma R_{A \wedge B} \Gamma'$, then $\Gamma R_A \Gamma'$ and $\Gamma R_B \Gamma'$
(S-DT) $\forall \Gamma, \Gamma', \Gamma'' \in S$, if $\Gamma R_A \Gamma'$ and $\Gamma' \leq \Gamma''$, and $\Gamma'' \in [|B|]$, then $\Gamma R_{A \wedge B} \Gamma''$;
(S-ID) $\forall \Gamma, \Gamma' \in S$, if $\Gamma R_A \Gamma'$, then $\Gamma' \in [|A|]$
(S-RCEA) $\forall \Gamma, \Gamma' \in S$, if $\vdash A \leftrightarrow B$, then $\Gamma R_A \Gamma'$ if and only if $\Gamma R_B \Gamma'$.

The proof is straightforward. As an example, let us prove (S-DT). We have to show that if $\Gamma R_A \Gamma'$, $\Gamma' \leq \Gamma''$, and $\Gamma'' \in [|B|]$, then $\Gamma R_{A \wedge B} \Gamma''$, i.e. $\{\phi$ such that $A \wedge B$ **says** $\phi \in \Gamma\} \subseteq \Gamma''$. For all such ϕ, by (DT), A **says** $(B \rightarrow \phi) \in \Gamma$, hence by definition of R_A, $B \rightarrow \phi \in \Gamma'$, and by definition of $\leq$, $B \rightarrow \phi \in \Gamma''$. Furthermore, also $B \in \Gamma''$ by Lemma 3. By deductive closure of Γ'', we conclude that $\phi \in \Gamma''$.

□

By the above lemmas, we can conclude that the axiomatization of the logic $\mathsf{Cond}_{\mathrm{ACL}}$ given in Section 2.1 is complete with respect to the semantics in Definition 1:

Theorem 3 (Completeness). *Given a formula* $\varphi \in \mathcal{L}$*, if* $\models \varphi$*, then* $\vdash \varphi$*.*

Proof. For a contradiction, suppose $\not\vdash \varphi$. Then by Lemma 1 there is a saturated extension Γ^* such that $\Gamma^* \not\vdash \varphi$, hence $\varphi \notin \Gamma^*$. By Definition 4 and Lemmas 3 and 4, we conclude that there is a (canonical) model $\mathbf{M} = (S, \leq, \{R_A\}, h)$, with $\Gamma^* \in S$, such that $\mathbf{M}, \Gamma^* \not\models \varphi$. It follows that φ is not logically valid, i.e. $\not\models \varphi$.

□

4 A sequent calculus for $\mathsf{Cond}_{\mathrm{ACL}}$

In this section we present a cut-free sequent calculus for $\mathsf{Cond}_{\mathrm{ACL}}$. Our calculus is called $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ and it makes use of labels to represent possible worlds, following the line of SeqS, a sequent calculus for standard conditional logics introduced in [12]. The completeness of the calculus is an immediate consequence of the admissibility of cut.

In addiction to the language $\mathcal{L}$ of the logic $\mathsf{Cond}_{\mathrm{ACL}}$, we consider a denumerable alphabet of labels $\mathcal{A}$, whose elements are denoted by $x, y, z, \ldots$. There are three types of labelled formulas:

1. *world formulas*, denoted by $x : \alpha$, where $x \in \mathcal{A}$ and $\alpha \in \mathcal{L}$, used to represent that the formula α holds in a world x;

2. *transition formulas*, denoted by $x \xrightarrow{A} y$, representing that xR_Ay;
3. *order formulas* of the form $y \geq x$ representing the partial order relation $\leq$.

A *sequent* is a pair $\langle \Gamma, \Delta \rangle$, usually denoted with $\Gamma \Rightarrow \Delta$, where Γ and Δ are multisets of labelled formulas. The intuitive meaning of a sequent $\Gamma \Rightarrow \Delta$ is: every model that satisfies all labelled formulas of Γ in the respective worlds (specified by the labels) satisfies at least one of the labelled formulas of Δ (in those worlds). This is made precise by the notion of *validity* of a sequent given in the next definition:

Definition 5 (Sequent validity). *Given a model $\mathcal{M} = (S, \leq, \{R_A\}, h)$ for $\mathcal{L}$, and a label alphabet $\mathcal{A}$, we consider a* mapping $I : \mathcal{A} \to S$. *Let F be a labelled formula, we define $\mathcal{M} \models_I F$ as follows:*

– $\mathcal{M} \models_I x : \alpha$ *iff* $\mathcal{M}, I(x) \models \alpha$
– $\mathcal{M} \models_I x \xrightarrow{A} y$ *iff* $I(x)R_AI(y)$
– $\mathcal{M} \models_I y \geq x$ *iff* $I(x) \leq I(y)$

We say that $\Gamma \Rightarrow \Delta$ is valid *in $\mathcal{M}$ if, for every mapping $I : \mathcal{A} \to S$, if $\mathcal{M} \models_I F$ for every $F \in \Gamma$, then $\mathcal{M} \models_I G$ for some $G \in \Delta$. We say that $\Gamma \Rightarrow \Delta$ is valid in* $\mathsf{Cond}_{\mathrm{ACL}}$ *if it is valid in every $\mathcal{M}$.*

In Figure 1 we present the rules of the calculus $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ for $\mathsf{Cond}_{\mathrm{ACL}}$. As usual, we say that a sequent $\Gamma \Rightarrow \Delta$ is *derivable* in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ if it admits a *derivation*. A derivation is a tree whose nodes are sequents. A branch is a sequence of nodes $\Gamma_1 \Rightarrow \Delta_1, \Gamma_2 \Rightarrow \Delta_2, \ldots, \Gamma_n \Rightarrow \Delta_n, \ldots$ Each node $\Gamma_i \Rightarrow \Delta_i$ is obtained from its immediate successor $\Gamma_{i-1} \Rightarrow \Delta_{i-1}$ by applying *backward* a rule of $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$, having $\Gamma_{i-1} \Rightarrow \Delta_{i-1}$ as the conclusion and $\Gamma_i \Rightarrow \Delta_i$ as one of its premises. A branch is closed if one of its nodes is an instance of axioms, namely (AX), $(AX_{\geq})$, and $(AX_{\bot})$, otherwise it is open. We say that a tree is closed if all its branches are closed. A sequent $\Gamma \Rightarrow \Delta$ has a derivation in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ if there is a closed tree having $\Gamma \Rightarrow \Delta$ as a root.
The rule (EQ) is used in order to support the rule (RCEA): if a sequent $\Gamma, x \xrightarrow{A} y \Rightarrow \Delta, x \xrightarrow{B} y$ has to be proved, then the calculus $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ checks whether A and B are equivalent, i.e. $A \leftrightarrow B$. To this aim, the (EQ) rule introduces a branch in the backward derivation, trying to find a proof for both sequents $u : A \Rightarrow u : B$ and $u : B \Rightarrow u : A$.

As an example, in Figure 2 we show a derivation in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ of an instance of the axiom (UNIT). In order to show that the formula $\alpha \to (A$ **says** $\alpha)$ is valid, we build a derivation in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ for the sequent $\Rightarrow u : \alpha \to (A$ **says** $\alpha)$.

As a further example, in Figure 3 we show a derivation in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ of an instance of the axiom (C).
The calculus $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ is sound and complete for the logic $\mathsf{Cond}_{\mathrm{ACL}}$, that is to say a formula $\psi \in \mathcal{L}$ is valid in $\mathsf{Cond}_{\mathrm{ACL}}$ if and only if the sequent $\Rightarrow u : \psi$ is derivable in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$. In order to prove this, we first need to show some basic structural properties of the calculus. First, we introduce the notion of complexity of a labelled formula:

Definition 6 (Complexity of a labelled formula). *We define the complexity of a labelled formula F as follows: cp $(x : \varphi) = 2 * \mid \varphi \mid$; cp $(x \xrightarrow{A} y) = 2 * \mid A \mid +1$; cp $(y \geq x) = 2$, where $\mid \phi \mid$ is the number of symbols occurring in the string representing the formula ϕ.*

$$(AX)\ \ \Gamma, F \Rightarrow \Delta, F \quad {\scriptstyle F \text{ either } x:P, P \in ATM \text{ or } y \geq x} \qquad (AX_{\perp})\ \ \Gamma, x : \perp \Rightarrow \Delta \qquad (AX_{\geq})\ \ \Gamma \Rightarrow \Delta, x \geq x$$

$$\frac{\Gamma, x : P \Rightarrow \Delta, y \geq x \qquad \Gamma, x : P, y : P \Rightarrow \Delta}{\Gamma, x : P \Rightarrow \Delta}(ATM) \quad {\scriptstyle P \in ATM}$$

$$\frac{\Gamma, y \geq x \Rightarrow \Delta, y : \alpha \qquad \Gamma, y \geq x \Rightarrow \Delta, y : \beta}{\Gamma \Rightarrow \Delta, x : \alpha \wedge \beta}(\wedge R) \quad {\scriptstyle y \text{ new}}$$

$$\frac{\Gamma, x : \alpha \wedge \beta \Rightarrow \Delta, y \geq x \qquad \Gamma, x : \alpha \wedge \beta, y : \alpha, y : \beta \Rightarrow \Delta}{\Gamma, x : \alpha \wedge \beta \Rightarrow \Delta}(\wedge L)$$

$$\frac{\Gamma, y \geq x \Rightarrow \Delta, y : \alpha, y : \beta}{\Gamma \Rightarrow \Delta, x : \alpha \vee \beta}(\vee R) \quad {\scriptstyle y \text{ new}}$$

$$\frac{\Gamma, x : \alpha \vee \beta \Rightarrow \Delta, y \geq x \qquad \Gamma, x : \alpha \vee \beta, y : \alpha \Rightarrow \Delta \qquad \Gamma, x : \alpha \vee \beta, y : \beta \Rightarrow \Delta}{\Gamma, x : \alpha \vee \beta \Rightarrow \Delta}(\vee L)$$

$$\frac{\Gamma, y \geq x, y : \alpha \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, x : \neg\alpha}(\neg R) \quad {\scriptstyle y \text{ new}}$$

$$\frac{\Gamma, x : \neg\alpha \Rightarrow \Delta, y \geq x \qquad \Gamma, x : \neg\alpha \Rightarrow \Delta, y : \alpha}{\Gamma, x : \neg\alpha \Rightarrow \Delta}(\neg L)$$

$$\frac{\Gamma, y \geq x, y : \alpha \Rightarrow \Delta, y : \beta}{\Gamma \Rightarrow \Delta, x : \alpha \rightarrow \beta}(\rightarrow R) \quad {\scriptstyle y \text{ new}}$$

$$\frac{\Gamma, x : \alpha \rightarrow \beta \Rightarrow \Delta, y \geq x \qquad \Gamma, x : \alpha \rightarrow \beta \Rightarrow \Delta, y : \alpha \qquad \Gamma, x : \alpha \rightarrow \beta, y : \beta \Rightarrow \Delta}{\Gamma, x : \alpha \rightarrow \beta \Rightarrow \Delta}(\rightarrow L)$$

$$\frac{\Gamma, y \geq x, y \xrightarrow{A} z \Rightarrow \Delta, z : \alpha}{\Gamma \Rightarrow \Delta, x : A \textbf{ says } \alpha}(\textbf{ says } R) \quad {\scriptstyle y \text{ and } z \text{ new}}$$

$$\frac{\Gamma, x : A \textbf{ says } \alpha \Rightarrow \Delta, y \geq x \qquad \Gamma, x : A \textbf{ says } \alpha \Rightarrow \Delta, y \xrightarrow{A} z \qquad \Gamma, x : A \textbf{ says } \alpha, z : \alpha \Rightarrow \Delta}{\Gamma, x : A \textbf{ says } \alpha \Rightarrow \Delta}(\textbf{ says } L)$$

$$\frac{u : A \Rightarrow u : B \qquad u : B \Rightarrow u : A}{\Gamma, x \xrightarrow{A} y \Rightarrow \Delta, x \xrightarrow{B} y}(EQ) \qquad \frac{\Gamma, z \geq x, z \geq y, y \geq x \Rightarrow \Delta}{\Gamma, z \geq y, y \geq x \Rightarrow \Delta}(Trans)$$

$$\frac{\Gamma, y \geq x, x \xrightarrow{A} y \Rightarrow \Delta}{\Gamma, x \xrightarrow{A} y \Rightarrow \Delta}(Unit) \qquad \frac{\Gamma, x \xrightarrow{A} y, y : A \Rightarrow \Delta}{\Gamma, x \xrightarrow{A} y \Rightarrow \Delta}(ID)$$

$$\frac{\Gamma, z \geq y, x \xrightarrow{A} y, z \xrightarrow{A} z \Rightarrow \Delta}{\Gamma, z \geq y, x \xrightarrow{A} y \Rightarrow \Delta}(C) \qquad \frac{\Gamma, x \xrightarrow{A\vee B} y, x \xrightarrow{A} y \Rightarrow \Delta \qquad \Gamma, x \xrightarrow{A\vee B} y, x \xrightarrow{B} y \Rightarrow \Delta}{\Gamma, x \xrightarrow{A\vee B} y \Rightarrow \Delta}(CA)$$

$$\frac{\Gamma, x \xrightarrow{A\vee B} y, x \xrightarrow{A} y \Rightarrow \Delta}{\Gamma, x \xrightarrow{A} y \Rightarrow \Delta}(CA - conv)$$

$$\frac{\Gamma, x \xrightarrow{A} y \Rightarrow \Delta, z \geq y \qquad \Gamma, x \xrightarrow{A} y \Rightarrow \Delta, z : B \qquad \Gamma, x \xrightarrow{A} y, x \xrightarrow{A\wedge B} z \Rightarrow \Delta}{\Gamma, x \xrightarrow{A} y \Rightarrow \Delta}(DT)$$

$$\frac{\Gamma, x \xrightarrow{A\wedge B} y, x \xrightarrow{A} y, x \xrightarrow{B} y \Rightarrow \Delta}{\Gamma, x \xrightarrow{A\wedge B} y \Rightarrow \Delta}(MON)$$

Fig. 1. The sequent calculus $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$.

The following properties hold in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$:

Lemma 5 (Height-preserving admissibility of weakening). *If a sequent $\Gamma \Rightarrow \Delta$ has a derivation of height h, then $\Gamma \Rightarrow \Delta, F$ and $\Gamma, F \Rightarrow \Delta$ have a derivation of height $h' \leq h$.*

Lemma 6 (Height-preserving label substitution). *If a sequent $\Gamma \Rightarrow \Delta$ has a derivation of height h, then $\Gamma[x/y] \Rightarrow \Delta[x/y]$ has a derivation of height $h' \leq h$, where $\Gamma[x/y] \Rightarrow \Delta[x/y]$ is the sequent obtained from $\Gamma \Rightarrow \Delta$ by replacing all occurrences of the label x by the label y.*

Lemma 7 (Height-preserving invertibility of rules). *Let $\Gamma \Rightarrow \Delta$ be an instance of the conclusion of a rule R of $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$, with R different from (EQ). If $\Gamma \Rightarrow \Delta$ is derivable, then the premise(s) of R is (are) derivable with a derivation of (at most) the same height.*

Lemma 8 (Height-preserving admissibility of contraction). *If a sequent $\Gamma \Rightarrow \Delta, F, F$ is derivable in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$, then there is a derivation of no greater height of $\Gamma \Rightarrow \Delta, F$, and if a sequent $\Gamma, F, F \Rightarrow \Delta$ is derivable in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$, then there is a derivation of no greater height of $\Gamma, F \Rightarrow \Delta$.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\ldots, z \geq x \Rightarrow z : \alpha, z \geq x}\,(AX) \qquad \dfrac{}{\ldots, x : \alpha, z : \alpha \Rightarrow z : \alpha}\,(AX)}{z \geq x, z \geq y, y \geq x, x \geq u, x : \alpha, y \xrightarrow{A} z \Rightarrow z : \alpha}\,(ATM)}{z \geq y, y \geq x, x \geq u, x : \alpha, y \xrightarrow{A} z \Rightarrow z : \alpha}\,(Trans)}{y \geq x, x \geq u, x : \alpha, y \xrightarrow{A} z \Rightarrow z : \alpha}\,(Unit)}{x \geq u, x : \alpha \Rightarrow x : A \textbf{ says } \alpha}\,(\textbf{ says } R)}{\Rightarrow u : \alpha \rightarrow (A \textbf{ says } \alpha)}\,(\rightarrow R)$$

Fig. 2. A derivation in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ for (UNIT).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Rightarrow w \geq w}\,(AX_{\geq}) \quad \dfrac{}{w \xrightarrow{A} w \Rightarrow w \xrightarrow{A} w}\,(AX) \quad \dfrac{}{w : \alpha \Rightarrow w : \alpha}\,(AX)}{w \geq y, x \geq u, x \xrightarrow{A} y, w \xrightarrow{A} w, w : A \textbf{ says } \alpha \Rightarrow w : \alpha}\,(\textbf{ says } L)}{w \geq y, x \geq u, x \xrightarrow{A} y, w : A \textbf{ says } \alpha \Rightarrow w : \alpha}\,(C)}{x \geq u, x \xrightarrow{A} y \Rightarrow y : (A \textbf{ says } \alpha) \rightarrow \alpha}\,(\rightarrow R)}{\Rightarrow u : A \textbf{ says } ((A \textbf{ says } \alpha) \rightarrow \alpha)}\,(\textbf{ says } R)$$

Fig. 3. A derivation in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ for (C).

Lemma 9. *A sequent* $\Rightarrow x : A \rightarrow B$ *is derivable in* $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ *if and only if the sequent* $x : A \Rightarrow x : B$ *is derivable in* $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ *.*

We now consider the cut rule:

$$\dfrac{\Gamma \Rightarrow \Delta, F \qquad\qquad \Gamma, F \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\,(cut)$$

where F is any labelled formula. We prove that this rule is admissible in the calculus $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$. The standard proof of admissibility of cut proceeds by a double induction over the complexity of F and the sum of the heights of the derivations of the two premises of (cut), in the sense that we replace one cut by one or several cuts on formulas of smaller complexity, or on sequents derived by shorter derivations. However, in our calculus $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ the standard proof does not work in case the cutting formula F is a transition formula $x \xrightarrow{A} y$ derived by an application of (EQ) in the left premise, and by an application of one of the following rules: $(C), (CA), (CA-conv), (DT), (MON)$ in the right premise. In order to prove the admissibility of cut for $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$, we proceed as follows. First of all, we represent with $\Gamma[x_i \xrightarrow{A} y_i] \Rightarrow \Delta[u_j \xrightarrow{A} v_j]$ a sequent containing *any* number of transitions labelled with the formula A; moreover, if $u : A \Rightarrow u : A'$ and $u : A' \Rightarrow u : A$ are derivable, we denote with $\Gamma^{\star} \Rightarrow \Delta^{\star}$ the sequent obtained by replacing *any* number of transitions labelled with A with the same transitions labelled with A' in $\Gamma[x_i \xrightarrow{A} y_i] \Rightarrow \Delta[u_j \xrightarrow{A} v_j]$. We prove that cut is admissible by “splitting” the notion of cut in two propositions:

Theorem 4. *In $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$, the following propositions hold:*

- *(A) If $\Gamma \Rightarrow \Delta, F$ and $\Gamma, F \Rightarrow \Delta$ are derivable, so is $\Gamma \Rightarrow \Delta$, i.e. the rule (cut) is admissible in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$;*
- *(B) if (I) $\Gamma[x_i \xrightarrow{A} y_i] \Rightarrow \Delta[u_j \xrightarrow{A} v_j]$ is derivable with a derivation of height h, (II) $u : A \Rightarrow A'$ and (III) $u : A' \Rightarrow A$ are derivable, then $\Gamma^\star \Rightarrow \Delta^\star$ is derivable with a derivation of height $h' \leq h$.*

Theorem 5 (Soundness of $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$). *If $\Gamma \Rightarrow \Delta$ is derivable, then $\Gamma \Rightarrow \Delta$ is valid in the sense of Definition 5.*

Theorem 6 (Completeness of $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$). *If $\Gamma \Rightarrow \Delta$ is valid in the sense of Definition 5, then $\Gamma \Rightarrow \Delta$ is derivable.*

Completeness of $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$ with respect to $\mathsf{Cond}_{\mathrm{ACL}}$ models of Definition 1 immediately follows from the completeness of the axiomatization of $\mathsf{Cond}_{\mathrm{ACL}}$ with respect to the semantics, shown in Theorem 3. We have that a formula $\varphi \in \mathcal{L}$ is valid if and only if the sequent $\Rightarrow u : \varphi$ has a derivation in $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$.

5 Related Work

The formal study of properties of access control logics is a recent research trend. As reported in [13], constructive logics are well suited for reasoning about authorization, because constructive proofs preserve the justification of statements during reasoning and, therefore, information about accountability is not lost. Classical logics, instead, allows proofs that discard evidence. For example, we can prove G using a classical logic by proving $F \rightarrow G$ and $\neg F \rightarrow G$, since from these theorems we can conclude $(F \vee \neg F) \rightarrow G$, hence $\top \rightarrow G$.

Abadi in [14] presents a formal study about connections between many possible axiomatizations of the says, as well as higher-level policy constructs such as delegation (speaks-for) and control. Abadi provides a strong argument to use constructivism in logic for access control, in fact he shows that from a well-known axiom like Unit in a classical logic we can deduce K says $\varphi \rightarrow (\varphi \vee K$ says $\psi)$. The axiom above is called *Escalation* and it represents a rather degenerate interpretation of says, i.e., if a principal says φ then, either φ is permitted or the principal can say *anything*. On the contrary, if we interpret the says within an intuitionistic logic we can avoid Escalation.

Even if there exist several authorization logics that employ the says modality, a limited amount of work has been done to study the formal logical properties of says, speaks-for and other constructs. In the following, we report the three different approaches adopted to study access control logics themselves.

Garg and Abadi [15] translate existing access control logics into S4 by relying on a slight simplification of Gödel's translation from intuitionistic logic to S4, and extending it to formulas of the form A says φ.

Garg [10] adopts an ad-hoc version of constructive S4 called DTL$_0$ and embeds existing approaches into it. Constructive S4 has been chosen because of its intuitionistic Kripke semantics which DTL$_0$ extends by adding *views* [10], i.e., a mapping from worlds to sets of principals.

Boella et al. [8] define a logical framework called FSL[5], based on Gabbay's Fibring semantics [16] by looking at says as a (fibred) modal operator.

However, adopting a fixed semantics like S4 does not permit to study the *correspondence theory* between axioms of access control logics and Kripke structures. Suppose we look at says as a principal indexed modality $\Box_K$, if we rely on S4 we would have as an axiom $\Box_K \varphi \to \varphi$, which means: *everything* that K says is permitted. To overcome this problem, both in [10, 15], Kripke semantics is sweetened with the addition of *views* which relativize the reasoning to a subset of worlds. Although this approach provides sound and complete semantics, it breaks the useful bound between modal axioms and semantic relations of Kripke structures.

6 Conclusion

We defined an intuitionistic conditional logic for Access Control called $\mathsf{Cond}_{\mathrm{ACL}}$.

The major contribution of our conditional approach w.r.t. works in [10, 15] is the identification of canonical properties for axioms of the logic (in particular Unit and C), i.e., first-order conditions on Kripke structures that are *necessary* and *sufficient* for the corresponding axiom to hold. In [8, 17, 18] we identify canonical properties for other access control axioms (e.g., C4, speaks-for, hand-off[6]).

We believe that this methodology has several advantages. First, the formalization of first-order constraints on Kripke structures shows that we do not need the full power and complexity of second-order quantification, this result has been proved for speaks-for relationship in [15] but our approach also applies to all other second-order well-known axioms for which we identify canonical properties. The fact that, from a semantic viewpoint, modal logic axiom schemas are not really second-order is a well-known result of modern modal logic and, by looking at **says** as a conditional modality, we managed to apply the same methodology to access control logics.

Second, the identification of canonical properties for access control axioms provides a natural deconstruction of access control logics. By deconstruction we intend the possibility to craft access control logics that adopt *any* combination of axioms for which there exists canonical properties. Garg and Abadi in [15] provide a translation into S4 of an access control logic with Unit and C4, but they do not define the semantic properties of the axioms with respect to the view based semantics. This limits the flexibility of the adopted semantics. For instance, although not all access control systems adopt Unit as an axiom [4, 19, 3], the translation in [15] does not provide an embedding in S4 for an access control logic without Unit. In particular, the approach in [15] does not provide a general methodology to deconstruct access control logics. In our approach, instead, we can formalize a logic and a calculus without Unit which is still sound and complete, by dropping the semantic condition S-UNIT and the corresponding rule $(Unit)$ in the calculus.

In this work, we have proven that the axiomatization of the intuitionistic conditional logic $\mathsf{Cond}_{\mathrm{ACL}}$ is sound and complete with respect to the semantics. Moreovoer, we

[5] Fibred Security Language.

[6] For a detailed discussion about these axioms see [14].

have provided a cut-free, labelled, sequent calculus for this logic. In $\mathsf{Cond}_{\mathrm{ACL}}$, principals are defined as arbitrary formulas. The generality of the language makes it possible to formalize, for instance, the so called boolean principals [7], that is, principals which are formed by boolean combinations of atomic principals. For the time being, $\mathsf{Cond}_{\mathrm{ACL}}$ only includes few axioms of access control logics but it can be extended in order to cope with richer axioms, as well as with the well known notion of "speaks for". We believe that choosing axioms for access control logics depends on the needs of security practitioners. With $\mathsf{Cond}_{\mathrm{ACL}}$ we show that, by looking at **says** as a conditional modality, we can offer a formal framework to study axioms of access control via canonical properties on the semantics and to build calculi to carry out automated deduction. Other issues to be tackled are the complexity of the logic $\mathsf{Cond}_{\mathrm{ACL}}$ and the termination and complexity of the sequent calculus $\mathcal{S}_{\mathsf{Cond}_{\mathrm{ACL}}}$. This is what we plan to do for future work.

Acknowledgements. The work has been partially supported by Regione Piemonte, Project "ICT4Law - *ICT Converging on Law: Next Generation Services for Citizens, Enterprises, Public Administration and Policymakers*". Valerio Genovese is supported by the National Research Fund, Luxembourg.

References

1. Abadi, M., Burrows, M., Lampson, B.W., Plotkin, G.D.: A calculus for access control in distributed systems. In: CRYPTO 91. 1–23
2. Bertolissi, C., Fernández, M., Barker, S.: Dynamic event-based access control as term rewriting. In: DBSec07. 195–210
3. Gurevich, Y., Roy, A.: Operational semantics for DKAL: Application and analysis. In: TrustBus 2009. 149–158
4. Lesniewski-Laas, C., Ford, B., Strauss, J., Morris, R., Kaashoek, M.F.: Alpaca: extensible authorization for distributed services. In: Proc. of ACM CCS 2007. 432–444
5. Li, N., Grosof, B.N., Feigenbaum, J.: Delegation logic: A logic-based approach to distributed authorization. ACM Trans. Inf. Syst. Secur. **6**(1) (2003) 128–171
6. Abadi, M.: Variations in access control logic. In: DEON08. 96–109
7. Garg, D., Abadi, M.: A modal deconstruction of access control logics. In: FoSSaCS 08, Budapest, Hungary 216–230
8. Boella, G., Gabbay, D., Genovese, V., van der Torre, L.: Fibred security language. Studia Logica **92**(3) (2009) 395–436
9. Nute, D.: Topics in Conditional Logic. Reidel, Dordrecht (1980)
10. Garg, D.: Principal centric reasoning in constructive authorization logic. In: Informal Proc. of IMLA. (2008)
11. Troelstra, A., van Dalen, D.: Constructivism in Mathematics: An Introduction. North-Holland Publishing, Amsterdam
12. Olivetti, N., Pozzato, G.L., Schwind, C.B.: A Sequent Calculus and a Theorem Prover for Standard Conditional Logics. ACM Transactions on Computational Logics **8**(4) (2007)
13. Garg, D., Pfenning, F.: Non-interference in constructive authorization logic. In: CSFW-19. (2006) 283–296
14. Abadi, M.: Variations in access control logic. In: 9th International Conference on Deontic Logic in Computer Science (DEON). (2008) 96–109
15. Garg, D., Abadi, M.: A modal deconstruction of access control logics. In: FoSSaCS 08, Budapest, Hungary 216–230

16. Gabbay, D.M.: Fibring logics. Oxford University Press (1999)
17. Genovese, V., Giordano, L., Gliozzi, V., Pozzato, G.: A constructive conditional logic for access control: a preliminary report. In: ECAI 2010 (to appear)
18. Modal Access Control Logic: Axiomatization, S., Proving, F.T.: D. m. gabbay and v. genovese and d. rispoli and l. van der torre. In: (to appear)
19. Becker, M.Y., Fournet, C., Gordon, A.D.: Design and semantics of a decentralized authorization language. In: 20th IEEE Computer Security Foundations Symposium (CSF). (2007) 3–15

Networks of Trust and Distrust: Towards Logical Reputation Systems

W. T. Harwood, J. A. Clark, J. L. Jacob

University of York
Department of Computer Science

Abstract. We introduce the notion of a network of trust and distrust relations between individuals and take an argumentation approach to the assessment of whether one individual should trust another.

. . . good decision is based on knowledge and not on numbers”

Plato - Early Dialogues - Laches

1 Introduction

This paper reports ongoing work in creating a logical foundation for reasoning about trust and trustworthiness in networks of individuals that may recommend one another as trustworthy or untrustworthy. One solution is to adopt some form of voting or counting scheme as in commonly done in reputations systems [10]. But in many circumstances, when the stakes are sufficiently high, e.g. deciding to trust a root certificate or disclose confidential information, weight of numbers does not constitute a good argument. As Plato puts it ". . . good decision is based on knowledge and not on numbers"[1].

One of the ultimate goals of this work is to provide the foundations for a logically well founded trust management system, or *Logical Reputation System*, where ‘reputation’ is computed by maintaining some notion of consistency between trust assertions made by trusted individuals. This approach contrasts

[1] For those without a classical education, or more relevantly today, an Internet connection, this is part of a general argument that Plato directs against amalgamating opinions as a basis for reaching a good decision. This is actually a cornerstone of Plato’s arguments against democracy. Today we take a more liberal view and regard some decisions as being appropriately arrived at by amalgamating individual opinions (such as who should rule the country, or what colour should we paint the school) and other decisions as arrived at by knowledge. It is certainly the contention of this paper that trust is best arrived at through knowledge rather than opinion.

with trust models, such as those of Coleman[6] or Marsh[12], that appeal to probabilities of trustworthiness or any similar numeric notions of degree of trustworthiness. Rather, in the approach considered here, a trust judgment is a purely logical resolution of possibly conflicting trust arguments. The intent is to use such a system to automatically make trust judgements in social network applications based on relational information gathered from users. This paper aims at setting out a logical framework based on argumentation theory to achieve this goal.

Our starting point is to consider networks of individuals that assert that they trust some individuals and distrust others.

Trust and distrust[2] are statements about the relationship between two individuals in relation to some action, such as, *information disclosure*, that holds in some context, such as, *today, in this building* (see, for example, Hardin's discussion in [9]). Throughout this paper we will consider the action and context as fixed so that we may talk of trust and distrust as binary relations. It should be apparent that we can put the additional dimensions back into the picture by considering families of relations parameterized by action and context.

If we only had information to the effect that certain individuals were trustworthy we would have a *web of trust* model (see, e.g. Zimmermann [13]) in which one individual trusts another if there is a trusted path between them. Here we consider how such models may be extended in the presence of additional negative assertions to the effect that certain individuals distrust one another. This allows the possibility of a trust path being undermined by a *distrust path*. Here we present a model of such systems in three stages of increasing complexity.

The first stage, *simple trust systems*, captures the idea that an individual trusts another if there is a trust path between them that is not undermined by distrust. Simple trust systems are modeled after argumentation theory[3, 4, 7]. Essentially, the approach is to assess the soundness of the argument that an individual, x_0 say, can trust an individual x_n. In our case the argument for trust is the existence of a trust path between x_0 and x_n in a network of trust relations. However, this argument may be undermined by an attack on it. An attack is an argument that some link in the chain of trust from x_0 to x_n is untrustworthy. In our case, such an argument is the existence of a path of trust from x_0 to some node y_m such that y_m *distrusts* some node connecting x_0 and x_n (including x_n itself). The existence of such an attack would make the original argument unsound, unless, of course, the attack itself was attacked in a similar manner, etc. etc.

[2] The relationship between trust and distrust is far from uncontroversial, see, for example, the discussions in the collection of articles [8]. We take distrust as more than the mere absence of trust. That is, distrust is not simply the complement of trust. Rather, trust and distrust are two relations that can exist between individuals and it is even possible for an individual to trust and distrust another individual simultaneously about the same topic. In such cases, although the individual is conflicted about trust, they are not logically inconsistent about trust.

The argumentation theory approach to resolving the set of attacks and counterattacks is to say that the original argument is sound if it is possible to partition the set, S, containing the original argument and the closure of all the attacks and counter attacks possible based on the initial argument, into two distinct sets, which we call S^+ and S^-, such that: S^+ is consistent in that no paths in S^+ attack one another; S^+ contains the original trust path; and for every path in S^- that attacks a path in S^+, S^+ contains a path that counter attacks that path.

Although formally straightforward, simple trust systems fail to capture an important aspect of trust: that when faced with a choice over conflicting recommendations of who to trust we have preferences over the choices. This leads to the formulation of the second stage, *preferential trust systems*, which introduces the notion that individuals may rank the other individuals into a partial ordering indicating their relative efficacy at making trust or distrust recommendations. This relative ranking is then extended to a partial order on paths which is used to measure the relative strength of paths. A distrust path can only undermine another path if it is sufficiently strong when compared to the path it is attacking (up to the point of attack). This second form of system is formalized by revising the notion of attack between paths.

The final stage *asymmetric preferential trust systems* addresses the fact that, in many situations, individuals have an asymmetric attitude to trust and distrust in that they are more willing to accept an argument that leads them to distrust than they are to accept one that leads them to trust. In the approach considered here, individuals require stronger arguments to make them trust than they do to make them distrust.

In order to directly describe the relationship between individuals, individuals' efficacy assessments, trust paths and distrust paths, trust systems are described relatively concretely. Of course these systems may be considered more abstractly using Dung's abstract argumentation systems framework. The connection between trust systems and Dung's framework is sketched in section 7.

2 Trust Systems

First we set out the framework of trust systems that we use throughout the paper.

A trust system is a collection of individuals I each of which may assert some collection of propositions, P_i for $i \in I$, and two binary relations $Trust : I \Leftrightarrow I$ and $Distrust : I \Leftrightarrow I$. If an individual, say x_0, trusts another individual, say x_n, then x_0 accepts P_n as true. If however x_0 distrusts x_n then x_0 neither accepts P_n as true nor rejects P_0 as false.

Informally, a trust system is a collection of individuals each of which may make assertions about the state of the world. In particular, each individual may assert whether or not they regard some other individuals as trustworthy or untrustworthy. If an individual i regards an individual j as trustworthy we will say that i trusts j. If, on the other hand, i regards j as untrustworthy we will say that i distrusts j. It is also possible for i to neither trust nor distrust j. If i trusts j then i is willing to accept j's assertions as true. In particular i accepts j's assertions about the trustworthiness of others as true. If i accepts a trust assertion of j as true e.g. if j trusts k, then i accepts there is an argument for trusting k, specifically i trusts j and j trusts k. If, however, j distrusts k then i accepts there is an argument that k is untrustworthy i.e. j whose assertions i trusts, distrusts k. It should be clear at this point that trust arguments can be extended (i.e. if i trusts j, j trusts k and k trusts l, then there is an argument for i trusting l) but distrust arguments cannot (i.e. if i trusts j, j distrusts k and k trusts l, then, since j does not accept k's assertions there is neither a trust argument nor a distrust argument, derivable from these facts alone, linking i and l).

Formally a trust system is a collection of individuals I and two binary relations $Trust : I \Leftrightarrow I$ and $Distrust : I \Leftrightarrow I$. Arguments for the trustworthiness and untrustworthiness of individuals will be modeled as trust paths and distrust paths between individuals. A trust path from x_0 to x_n is a sequence $<x_0, x_1, \ldots, x_{n-1}, x_n>$ such that every pair (x_i, x_{i+1}) is in $Trust$. A distrust path from x_0 to x_n is a sequence $<x_0, x_1, \ldots, x_{n-1}, x_n>$ such that every pair (x_i, x_{i+1}) for $i < n-1$ is in $Trust$ and (x_{n-1}, x_n) is in $Distrust$. That is, the path $<x_0, x_1, \ldots, x_{n-1}>$ is a trust path and the final step $<x_{n-1}, x_n>$ is distrusting.

The set of trust paths will be called TP and the set of distrust paths will be called DP.

Given a path, p, (either a trust path or a distrust path) then *range* p is the set of all individuals in the path i.e. if $p = <x_0, \ldots, x_n>$ then $range\ \ p = \{x_0, \ldots, x_n\}$. We will also say that $first\ \ p = x_0$ and $last\ \ p = x_n$, and, for later use, $front\ \ p = <x_0, \ldots, x_{n-1}>$.

A distrust path, q, *attacks* a path if it attacks the trust supporting the path, meaning it either attacks any point of a trust path (including its last node) or it attacks any point on the front of a distrust path (i.e. the trust path part of the distrust path).

Let $tr\ \ p$ be the *trust part* of a path p, defined by

$$tr\,p = \begin{cases} p & \textbf{if } p \in TP \\ front\,p & \textbf{if } p \in DP \end{cases}$$

Then *attacks* relation between paths is defined by:

$$q\ \ attacks\ \ p \equiv q \in DP \wedge first(q) = first(p) \wedge last(q) \in range(tr\,p)$$

An attack is admissible if it satisfies an *admissibility condition* that varies between the three forms of trust system considered. For simple trust systems all attacks are admissible. For preferential trust systems an attack is admissible only if it is of adequate strength. For asymmetric trust systems the strength condition varies depending on the way the attack will affect the overall outcome. The following section illustrates the effect of the different admissibility conditions with a short example.

3 A Short Example

To illustrate the three systems consider the following network of trust and distrust.

- Alice trusts Bob,
- Bob trusts Carol,
- Carol trusts Dan,
- Dan distrusts Bob,
- Alice trusts Elizabeth,
- Elizabeth distrusts Dan.
- Does Alice trust Carol?

Under simple trust systems, where we have no other information, Dan's distrust of Bob defeats the chain of trust connecting Alice and Carol but Elizabeth's attack on Dan defeats it, and so cancels its effect, leaving Alice having trust in Carol.

If additionally we know:

- Alice rates herself, Bob and Carol strictly higher in ability to make trust judgements than she does Dan.

Then, under a preferential trust system, Alice will trust Carol because Dan's distrust of Bob will lead to an attack path that is weaker than the trust path between Alice and Carol. If, however,

- Alice rates herself, Bob, Carol and Dan equally in ability to make trust judgements

then we would need to consider the exact formulation of the preference system: does an attack from a path of equally strength defeat the attacked path or not? Below we differentiate between *conservative* systems in which attacks must be strictly stronger to defeat a path and *paranoid* systems in which attacks from paths of equal strength, or attacks from incomparable paths, can defeat the attacked path.

Finally, to illustrate asymmetric trust systems, which are *conservative* if the consequence is trust and *paranoid* if the consequence is distrust, we consider two situations

1. Alice rates everyone equally in their ability to make judgements.
2. Alice rates Elizabeth higher than everyone else in her ability to make trust judgements.

Under assumption 1, Dan's distrust of Bob will lead to Alice not trusting Carol even though Elizabeth distrusts Dan, because the distrusting outcome is favored over the trusting outcome. Whereas under assumption 2, Elizabeth's distrust of Bob can cancel the attack and lead to Alice having trust in Carol.

The rest of the paper provides the technical details of each of the systems.

4 Simple Trust Systems

As mentioned above, in simple trust systems all attacks are admissible.

We wish to define notion of a *sound* trust path, p, between two individuals as a path which is either not attacked or only attacked by distrust paths that are themselves defeated by other attacks. To do this we first define the attack closure set of the path p to be p^a , the least set closed under:

– $p \in p^a$,
– $q \in p^a$ and $r \in attacks(q) \implies r \in p^a$.

and say p is sound if and only if p^a can be partitioned into two sets S^+ and S^- such that:

– $p \in S^+$,
– S^+ is *consistent* in that no path in S^+ attacks any other path in S^+, i.e. $S^+ \cap attacks^{\exists}(S^+) = \emptyset$,
– S^+ *defends itself* against S^- in that every path in S^- that attacks a path in S^+ is itself attacked by a path in S^+, i.e. $\overset{\smile}{attacks}{}^{\exists}(S^+) \subseteq attacks^{\exists}(S^+)$.

where, for a relation R, $R^{\exists}X$ is the forward image of X under R i.e. $\{y | \exists x \in S.xRy\}$.

We will call a set, S, a *support*, if it is consistent and defends itself (i.e. it can support some trust path p).

We say an individual x_0 *trusts* an individual x_n iff[3] there is a sound trust path between x_0 and x_n.

Given a simple trust system $T = (I, \{P_i\}_{i \in I}, Trust, Distrust)$ its set of sound trust paths (STP) is defined by:

$$STP = \{(x_0, x_n) \in I \times I \mid \exists p \in TP.first(p) = x_0 \wedge last(p) = x_n \wedge sound(p)\}$$

Two simple trust systems S and T are *trust equivalent* iff they have the same set of individuals and the same set of sound trust paths.

5 Preferential Trust Systems

Preferential trust systems restrict admissible attacks using a notion of relative strength between the attacked path and the attacking path. The particular notion that we use is that the strength of the path is derived from the competence, trustworthiness or reliability of the individuals in the path in making judgments about other individuals. We will settle on the neutral term *efficacy* for any of the terms competence, trustworthiness or reliability (or any other such notion).

In the above, all individuals have been regarded as of equal efficacy in rating the trustworthiness of other individuals. We will now consider what

[3] Here, and throughout, we will adopt the convention of writing *iff* for *if and only if*.

happens when individuals are partially ordered by their efficacy in performing such rating. We will assume every individual i has available their own assessment of the relative efficacy of all other individuals at rating the trustworthiness of others. Formally we take this to be a family of partial orders (reflexive, anti-symmetric and transitive binary relations) over the set of individuals I, one for each member $i \in I$, denoted $\succeq_i$ reflecting i's view of the relative efficacy of individuals. Our goal is that, given a path p which is attacked by a path q, we wish to compare the strength of p up to the point of the attack, $last(q)$, with the strength of q. To do this we need to derive a partial ordering of paths from the partial ordering of the efficacy of the individuals in the paths.

We will call the segment of the path p up to the attack, $p\ |_{last(q)}$. If we were to use a strict total ordering to compare paths then we would say that one path, say q, was weaker than another, say p, when $range(q)$ contained an element less than any element in $range(p)$. We generalize this idea to partial orders by considering minimal elements in the ranges of the paths.

First we define an extension of a partial order over a set to a partial order over subsets of that set.

Given a partial order, $\succeq$, over a set S, we say that a subsets P and Q of S are comparable[4], written $P \sim Q$ iff:

$$(\forall x \in P.\ \exists y \in Q.\ x \succeq y) \vee (\forall y \in Q.\ \exists x \in P.\ x \succeq y)$$

The set of minimal elements of a set P i s defined as:

$$minimal(P) = \{x \in P \mid \forall y \in P.\ (y \succeq x) \implies y = x\}$$

A set, $P \subseteq S$, is at-least-as-strong-as a set, $Q \subseteq S$, written $P \sqsupseteq Q$, iff

$$P \sim Q \wedge \forall x \in minimal(P).\ \exists y \in minimal(Q).\ x \succeq y$$

A set P subset of S is stronger than a set Q subset of S, written $P \sqsupset Q$, iff

$$P \sqsupseteq Q \wedge Q \not\sqsupseteq P$$

All this amounts to is that subsets are ordered by comparing the least elements of the chains and if one of the subsets has strictly smaller elements for any of its chains (and the other does not) then it is the smaller set.

A path p is *stronger than* q, also written $p \sqsupset q$, iff

$$\begin{gathered} first(p) = first(q)\ \wedge last(p) = last(q) \wedge \\ range(p) \setminus \{last(p)\} \sqsupset range(q) \setminus \{last(q)\} \end{gathered}$$

The removal of the last elements of the paths is due to the fact that we derive the efficacy of the individuals on the path that make the trust recommendations.

[4] **Warning**: For those familiar with the notation $x \parallel y$ for x incomparable with y under the partial order $\preceq$. The notion defined here is over subsets of the ordering, not elements of the ordering. So $P \sim Q \equiv \exists p \in P, q \in Q. \neg(p \parallel q)$.

We now modify the definition of attack to take account of the relative strength of paths. There are two possible views of relative strength that correspond to whether the individual x_0 takes a *conservative* or a *paranoid* stance with respect to attacks. If x_0 takes a conservative stance, then a path is only defeated by a strictly stronger attack. If, on the other hand, x_0 takes a paranoid stance, then a path is defeated if the attacking path is incomparable or is at-least-as-strong-as the attacked path. The paranoid position allows attacks to defeat other attacks if x_0 is not in a position to positively assert that the attacked path is the stronger of the two.

That is, if x_0 has a conservative stance, then an attack, q, on a path, p, only succeeds if $q \sqsupset p\,|_{last(q)}$:

$$q\ attacks_C\ p \equiv q \in DP \wedge first(q) = first(p) \wedge last(q) \in range\ p \wedge q \sqsupset p\,|_{last(q)}$$

and if x_0 has a paranoid stance, then an attack, q, on a path, p, only succeeds if $p\,|_{last(q)} \not\sqsupset q$:

$$q\ attacks_P\ p \equiv q \in DP \wedge first(q) = first(p) \wedge last(q) \in range\ p \wedge p\,|_{last(q)} \not\sqsupset q$$

Preferential trust systems are formulated by replacing the definition of attack in simple trust systems with either the conservative or the paranoid definition of attack[5].

6 Asymmetric Preferential Trust Systems

In practice individuals are often asymmetric in their attitude to trust and distrust. That is, they are paranoid about trust and conservative about distrust. This means that the admissibility of an attack changes according to the overall role it plays in determining the outcome, introducing an asymmetry between paths which ultimately lead to a trust decision and paths which ultimately lead to a distrust decsion. We capture this asymmetry by redefining the conditions for forming S and forming the partitions S^+ and S^-:

– The attack closure set S is the least closed set of paranoid attacks based on a trust path p as above.
– S^+ is restricted to only containing the initial trust path, p, and conservative attacks.
– Since conservative attacks are a subset of paranoid attacks, S^- may contain both types of attack.

Trust path p is sound iff it is possible to form a partition of S such that:

[5] A system may also be formulated where the stance varies from individual to individual which is essentially a simple trust system with an indexed family of *attacks* operators.

– $p \in S^+$.
– S^+ is *consistent* in that no path in S^+ attacks any other path in S^+.
– S^+ is *conservative* in that every path in S^+ is either p or a member of $attacks_C(x)$ for some x.
– S^+ *defends itself* against S^- in that every path in S^- that attacks a path in S^+ is itself attacked by a path in S^+.

7 Connecting Trust and Dung's Abstract Argumentation

Dung [7] defines an abstract argumentation system as a pair $(AR, Attacks)$ where AR is a set of arguments and $Attacks$ is a binary relation over AR called the *attacks* relation. We write $x \ Attacks \ y$ for x attacks y. A set, $S \subseteq D$, attacks an argument, $x \in D$, if some argument in S attacks x (we will say that $S \ ATTACKS \ x$ for $\exists y \in S.\ y \ Attacks \ x$).

Dung then goes on to define the notions of:

– *Conflict free*: A set of arguments $S \subseteq AR$ is *conflict free* iff there is no pair of arguments $x \in S$ and $y \in S$ such that $x \ Attacks \ y$.
– *Acceptable*: An argument $x \in AR$ is *acceptable with respect to S* iff for every argument $y \in AR$ if $y \ Attacks \ x$ then $S \ ATTACKS \ y$. Following [2] we will also say that S *defends* x when x is acceptable with respect to S.
– *Admissible*: A set $S \subseteq AR$ is *admissible* iff S is *conflict free* and each argument in S is *acceptable with respect to S*.

Dung then goes on to discuss various notions of semantics that further restrict the notion of admissibility which are not used in our current semantics.

To translate the above into Dung's framework we consider an individual a and the set of trust paths, P, rooted at a. For the simple trust systems:

– The set of arguments AR is the set P.
– The attacks relation between holds $q, p \in P$, i.e. $q \ Attacks \ p$, iff there exists a distrust path $d = {<}x_0, x_1, \ldots, x_{n-1}, x_n{>}$ with $q = {<}x_0, x_1, \ldots, x_{n-1}{>}$ and $d \ attacks \ p$.

Note that this definition of *Attacks* loses information by conflating multiple distinct attacks from q to different points on p.

A path p is $sound_D$ iff P can be partitioned into two sets S^+ and S^- such that $p \in S^+$ and S^+ is admissible.

Clearly the above notions of consistency and conflict freeness are the same (albeit on different domains):

Proposition 1. $S \cap R^{\exists}(S) = \emptyset \equiv S \subseteq \overline{R^{\exists}(S)}$

Likewise, S defends itself and S is acceptable are essentially the same as demonstrated by the following two propositions.

First we introduce the dual of the forward image operator on binary relations over a set S: Given a binary relation $R : S \leftrightarrow S$, the function $R^{\forall} : \mathcal{P}S \to \mathcal{P}S$ is defined by:

$$R^{\forall}Y = \{x \mid \forall y.xRy \implies y \in Y\}$$

$R^{\exists}X$ is the forward image of X and $R^{\forall}Y$ is the set of elements in the inverse image of R that only result in elements in Y. [6].

$R^{\exists}$ and $R^{\forall}$ form a (covariant) galois connection, or axiality, over S. This means that $R^{\exists} \circ R^{\forall}$ is an interior operator on S and $R^{\forall} \circ R^{\exists}$ is a closure operator on S. Letting $\overset{\smile}{R}$ represent the converse of R (i.e. $x \overset{\smile}{R} y \equiv yRx$) then

Proposition 2. *S is acceptable iff* $S \subseteq (\overset{\smile}{Attacks})^{\forall}(Attacks^{\exists}(S))$

Proof Sketch. This follows from $R^{\forall}X = \overline{(\overset{\smile}{R})^{\exists}(\overline{X})}$ and Amgoud & Cayrol's theorem, quoted in [2], which rendered in our notation is S is *acceptable* iff : S is acceptable iff $S \subseteq \overline{Attacks^{\exists}(\overline{Attacks^{\exists}S})}$.

Proposition 3. *S is acceptable iff* $(\overset{\smile}{Attacks})^{\exists}S \subseteq Attacks^{\exists}S$

Proof Sketch.

$\implies$

$S \subseteq (\overset{\smile}{Attacks})^{\forall}(Attacks^{\exists}(S))$

by proposition 2

$(\overset{\smile}{Attacks})^{\exists}S \subseteq (\overset{\smile}{Attacks})^{\exists}((\overset{\smile}{Attacks})^{\forall}(Attacks^{\exists}(S)))$

by $(\overset{\smile}{Attacks})^{\exists}$ preserves order

$(\overset{\smile}{Attacks})^{\exists}S \subseteq Attacks^{\exists}(S)$

by $(\overset{\smile}{Attacks})^{\exists} \circ (\overset{\smile}{Attacks})^{\forall}$ being an interior operator

$\impliedby$

$(\overset{\smile}{Attacks})^{\forall}((\overset{\smile}{Attacks})^{\exists}S) \subseteq (\overset{\smile}{Attacks})^{\forall}(Attacks^{\exists}S)$

by $(\overset{\smile}{Attacks})^{\forall}$ preserves order

$S \subseteq (\overset{\smile}{Attacks})^{\forall}(Attacks^{\exists}S)$

by $(\overset{\smile}{Attacks})^{\forall} \circ (\overset{\smile}{Attacks})^{\exists}$ being an closure operator

Proposition 4. $sound_D(p) \equiv sound(p)$

Proof Sketch. Since the definitions of S being a support and S being admissible are essentially the same between the two definitions of soundness, the major work falls on showing that the existence of a suitable partition of p^a is equivalent to the existence of a suitable partition of P.

[6] $R^{\forall}Y$ is closely related to the weakest precondition operator in programming languages semantics. The exact relation depending on the particular relational theory of programs and termination used.

Recall

$$tr(p) = \begin{cases} p & \textbf{if } p \in TP \\ front(p) & \textbf{if } p \in DP \end{cases}$$

Let p be a trust path, then $tr^{\exists}(p^a) \subseteq P$. Assume the pair S^+, S^- form a suitable partition of p^a then the pair $tr^{\exists}(S^+), P \setminus tr^{\exists}(S^+)$ form a suitable partition of P.

Conversely, if the pair S^+, S^- form a suitable partition of P then the pair $\overset{\smile}{tr}{}^{\exists}(S^+) \cap p^a, p^a \setminus (\overset{\smile}{tr}{}^{\exists}(S^+) \cap p^a)$ form a suitable partition of p^a.

To obtain the corresponding Dungian systems for preferential and asymmetric trust systems we modify the definition of the *Attacks* relation. Given the conflating of attacks mentioned above we must ensure that the potential multiplicity of attacks is correctly dealt with when comparing strength.

For two paths $p, q \in P$ such that q *Attacks* p we define:

$$q \sqsupset_C p \equiv \forall x \in range(p).(last(q), x) \in Distrust \implies q \sqsupset p|_x$$

$$q \sqsupset_P p \equiv \forall x \in range(p).(last(q), x) \in Distrust \implies p|_x \not\sqsupset q$$

And given a partition of P into S and $\overline{S}$ (= $P \setminus S$):

$$\sqsupset_A^S \equiv ((S \times \overline{S}) \cap \sqsupset_C) \cup ((\overline{S} \times S) \cap \sqsupset_P)$$

Using these orderings we define the three corresponding attacks relations as:

- Conservative Preferential Trust: $Attacks_C = Attacks \cap \sqsupset_C$.
- Paranoid Preferential Trust: $Attacks_P = Attacks \cap \sqsupset_P$.
- Asymmetric Trust: $Attacks_A^S = Attacks \cap \sqsupset_A^S$.

Finally we demonstrate that the asymmetric trust systems have a pleasing simplification of the acceptability condition in that $Attacks_A$ factors into $Attacks_P$ and $Attacks_C$ on either side of the acceptability condition.

Proposition 5. $Attacks_A^S = ((S \times \overline{S}) \cap Attacks_C) \cup ((\overline{S} \times S) \cap Attacks_P)$

Proof Sketch. by boolean algebra

Proposition 6. *An set, S, is acceptable in the asymmetric trust system iff* $(\overset{\smile}{Attacks_P})^{\exists}S \subseteq (Attacks_C)^{\exists}S$

Proof Sketch.

$$(\overset{\smile}{Attacks_P^S})^{\exists} S \subseteq {Attacks_C^S}^{\exists} S$$

by proposition 2

$$(((S \times \overline{S}) \cap Attacks_C) \cup \overset{\smile}{((\overline{S} \times S) \cap Attacks_P)}^{\exists}) S \subseteq$$
$$(((S \times \overline{S}) \cap Attacks_C) \cup ((\overline{S} \times S) \cap Attacks_P)^{\exists}) S$$

by proposition 5

$$\overset{\smile}{((S \times \overline{S}) \cap Attacks_C)}^{\exists} S \cup \overset{\smile}{((\overline{S} \times S) \cap Attacks_P)}^{\exists} S \subseteq$$
$$((S \times \overline{S}) \cap Attacks_C)^{\exists} S \cup ((\overline{S} \times S) \cap Attacks_P)^{\exists} S$$

by distribute $(\_)^{\exists}$ over union

$$(\overset{\smile}{Attacks_P})^{\exists} S \subseteq (Attacks_C)^{\exists}$$

by domain restrictions

8 Conclusions

For us at least, the idea of using argumentation to reason about networks of trust, and distrust, is in its infancy. The work presented here raises more questions than it answers, some of which we raise below (and there are many more than raised here).

Trust systems as outlined above offer a logically well founded approach to reasoning about trust based on minimal information gathered from individuals i.e. the individuals relative assessment of the efficacy of the judgements of others and a map of immediate trust and distrust relations between individuals. The natural next step is to investigate this in practice in an actual social network application.

The asymmetric preferential trust systems above rely on the fact that conservative attacks are a subset of paranoid attacks. Clearly it is possible to generalize further and define relevant attacks and acceptable rebuttals to relevant attacks. Given a set of attacks, we classify some attacks as relevant, some as acceptable rebuttals of relevant attacks, and some as neither. S is built as the closure of attacks on a trust path p as above and we define S^+ and S^- by:

– $p \in S^+$,
– S^+ is *consistent* in that no path in S^+ attacks any other path in S^+,
– S^+ is a *rebuttal set* in that every path in S^+ is either p or a rebuttal attack,
– S^+ *defends against relevant attacks* from S^- in that every relevant attack in S^- that attacks a path in S^+ is itself attacked by a path in S^+.

This generalization opens up the possibility of considering richer asymmetries between trust and distrust arguments. For example, if we drop the use of the extended order relation and consider using a labeling of the individuals in paths. Consider, as illustration, a sensor network based on three kinds of individual sensor: electronic sensing and people that perform either casual or detailed inspections. We may trust an individual because we have a mixed trust path to it but relevant attacks may be limited to chains that exclude electronic sensors and rebuttals may be limited to chains of people who perform detailed inspection[7]. This approach will be the subject of further investigation.

The relation to Dungian argumentation outlined in section 7 uses only the most basic semantic notion of admissibility. This raises the question whether or not the other possible semantics have a useful meaning for trust (and distrust) relations. The question is why we would *want* a richer set of arguments than that required to support the sounds of a particular trust path? Perhaps there is a useful notion of sets of individuals you can consistently trust corresponding to the other possible semantics. It certainly is worth investigating.

During the revision of this paper the authors encountered the work of Cayrol and Lagasquie-Schiex on Bipolar Argumentation [5] systems, and of Kaci and Torre , and Amgoud, Dimopoulos and Moraitis Preference Based Argumentation (se e.g. [11] and [1] respectively). Both seem to overlap on the intent pursued here and offer interesting directions for future investigation.

9 Acknowledgements

The authors would like to thank the reviewers for their constructive criticism, knowledgeable comments and insightful questions. In particular we would like to thank two of the reviews for their detailed comments and references to the work of other authors. This latter has been particularly useful to the ongoing work, even though not adequately reflected in this paper. In addition to minor corrections of the text the comments have been addressed with additional footnotes and the addition of Section 7 (in response to the reviewer plea for more mathematics).

[7] Admittedly, this example can be done using order relations, but it is seems conceptional simpler as a predicate on the acceptable sets of attacks and counter attacks.

Bibliography

[1] Leila Amgoud, Yannis Dimopoulos, and Pavlos Moraitis. Making decisions through preference-based argumentation. In Gerhard Brewka and Jérôme Lang, editors, *KR*, pages 113–123. AAAI Press, 2008.

[2] Philippe Besnard and Sylvie Doutre. Characterization of semantics for argument systems. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *KR*, pages 183–193. AAAI Press, 2004.

[3] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. MIT Press, 2008.

[4] A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93:63–101, 1997.

[5] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In Lluis Godo, editor, *ECSQARU*, volume 3571 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2005.

[6] James Coleman. *Foundations of Social theory*. Belknap Press, 1990.

[7] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.

[8] Russell Hardin, editor. *Trust & Distrust*. Russell Sage Foundation, 2004.

[9] Russell Hardin. *Trust*. Polity Press, 2006.

[10] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[11] Souhila Kaci and Leendert van der Torre. Preference-based argumentation: Arguments supporting multiple values. *Int. J. Approx. Reasoning*, 48(3):730–751, 2008.

[12] Stephen Paul Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Dept. of Computing and Mathematics, University of Stirling, 1994.

[13] Phil Zimmermann. *PGP User Guide*. MIT Press, 1994.

Dynamic languages of propositional control for protocol specification

Andreas Herzig[1] and Nicolas Troquard[2]

1. Université de Toulouse, CNRS, IRIT, France
2. Department of Computer Science, University of Liverpool, UK

Abstract. We propose a family of dynamic logics of propositional control. They extend classical propositional logic by a variety of modal operators of assignment, and modal operators of transfer of control over a propositional variable. We also present an extension with operators of knowledge. We essentially focus on their formal properties, stating their complexity and their proof theory.

1 Introduction

The logic of propositional control CL-PC was introduced in [10] as a reconstruction of coalition logic. What agents can achieve is explained there in terms of their control over propositional variables. The central construction is a modal operator $\langle \overset{J}{\leftarrow} \rangle$ of contingent ability, where J is a set of agents.[1] The formula $\langle \overset{J}{\leftarrow} \rangle\varphi$ reads "the coalition of agents J can assign truth values to the variables under its control in such a way as to make φ true". For example, $\langle \overset{\{i\}}{\leftarrow} \rangle q \land \langle \overset{\{i\}}{\leftarrow} \rangle \neg q$ expresses that agent i is able both to make q true and to make q false; which means that agent i controls q.

The logic was further studied and extended in [7, 15, 14, 13]. A particularly interesting such extension is that by *delegation* CL-PC, coined δCL-PC; see [8] for a recent presentation.[2] It introduces into CL-PC new modalities of control transfer (baptized 'delegation' in the original papers). In that work, atomic delegation programs take the form $i \overset{q}{\hookrightarrow} j$, whose intended meaning is that i who currently controls q turns over its control to j. Moreover, complex control transfer programs δ are constructed by means of the standard PDL constructs ? (test), ; (sequential composition), $\cup$ (nondeterministic composition) and * (iteration). The modal formula $\langle \delta \rangle\varphi$ then reads "there exists a computation of the delegation program δ, starting from the current situation, such that after δ has terminated φ holds". Such constructs allow to talk about interaction protocols.

The semantics of CL-PC and δCL-PC are originally in terms of couples (ξ, θ) where θ maps every propositional variable q to a truth value $\theta(q)$ in $\{\mathtt{tt}, \mathtt{ff}\}$, and ξ maps every propositional variable q to one agent. We will here consider a more general setting (to be introduced in Section 2), where ξ maps every propositional variable q to a —possibly

[1] The original notation is $\Diamond_J\varphi$ instead of $\langle \overset{J}{\leftarrow} \rangle\varphi$.

[2] It was originally abbreviated DCL-PC, but we changed it in order to avoid confusion with the logics we introduce here. In these logics (and more generally in dynamic epistemic logics [5]) the letter 'D' stands for 'dynamic'.

empty— *set* of agents $\xi(q)$. We call the mapping ξ a *control allocation* and the mapping θ a *valuation*. ξ is about control, while θ is about truth. Given (ξ, θ), call an *update of J's part of* θ any model (ξ', θ') such that $\xi' = \xi$ and $\theta'(q) = \theta(q)$ for every q such that $\xi(q)$ is not in J.

In addition to generalising the models of CL-PC, our main objective is to elaborate on the basic bricks[3] of the language. Indeed, in much the same spirit as the program $i\overset{q}{\hookrightarrow}j$ updates the allocation function ξ, one may view the operator $\langle\overset{J}{\leftarrow}\rangle$ of CL-PC as the application of a program $\overset{J}{\leftarrow}$ that updates the valuation function θ by changing at most the value of the propositions under the control of the agents in J.

This observation made, one of the main contributions of this paper is to show that we can redefine the logics of CL-PC and δCL-PC from a dynamic logic with very simple programs. Our logic $\mathsf{DL}^{\mathsf{PC}}$ (that we introduce in Section 3) makes use of four types of atomic programs:

$q{\leftarrow}\top$: q is given the value `tt`
$q{\leftarrow}\bot$: q is given the value `ff`
$i\overset{q}{\hookrightarrow}$: i loses the control of q
$\overset{q}{\hookrightarrow}i$: i receives the control of q

We keep track of who owns a propositional variable by the use of a theory of propositions $c_{i,q}$, for every agent i and proposition q, that reads that i controls q. For instance, the δCL-PC program $i\overset{q}{\hookrightarrow}j$ will then be simulated by a test of $c_{i,q}$ followed by i's loss and j's gain of control over q.

In Section 4 we introduce two new primitive programs to obtain a logic that we call Q-$\mathsf{DL}^{\mathsf{PC}}$, for *quantified* $\mathsf{DL}^{\mathsf{PC}}$. The program $\overset{q}{\hookrightarrow}J$ gives the control of q to one of the agents in J, and the program $q\overset{J}{\leftarrow}$ sets the value of q that is controlled by some agent in J to `tt` or `ff`. These programs may look a bit abstract. They can in fact be defined in terms of the four programs listed above as we will see, but to the price of losing a bit of succinctness. Moreover, they will provide the good level of abstraction to relate our languages with those of CL-PC and Coalition Logic.

Here is an example illustrating the concepts that we have introduced so far.

Example 1. Consider the reviewing process of some conference. Let the set of agents be $\mathbb{A} = \{Chair\} \cup RV$, where $Chair$ is the PC chair and $RV = \{R_1, \ldots, R_M\}$ is the set of reviewers. Let there be N papers $PP = \{1, \ldots, N\}$ to be reviewed. We suppose that the decision of acceptance of each paper $n \in PP$ is based on three opinions. Each opinion is modelled by the truth value of a propositional variable. Let their set be $\mathbb{P} = \bigcup_{1 \leq n \leq N}\{p_n^1, p_n^2, p_n^3\}$. Initially it is the PC chair who controls all the papers, i.e. we have an initial model where $\xi(p_n^k) = \{Chair\}$ for all $n \in PP$ and $k \in \{1, 2, 3\}$. The assignment of papers to reviewers corresponds to the execution of the sequence $\overset{p_1^1}{\hookrightarrow}RV, \cdots, \overset{p_N^3}{\hookrightarrow}RV$. After that, every p_n^k is controlled by both the PC chair and some reviewer. Finally, the

[3] We will generalise the atomic programs, but we will keep a complete investigation of the PDL constructs for later work.

reviewers' decisions are modelled as the assignment of truth values to papers under their control, i.e. the programs $p_n^k \overset{R_m}{\leftarrow} \top$ and $p_n^k \overset{R_m}{\leftarrow} \bot$. One may then check properties such as $\langle \overset{PP}{\hookrightarrow} RV\rangle\langle PP \overset{RV}{\leftarrow}\rangle\varphi$ for some property φ expressing for instance that the acceptance rate is 25%. A more complex and more realistic check where the final decision is up to the PC chair can be expressed by $\langle Chair \overset{PP}{\hookrightarrow} RV\rangle\langle PP \overset{RV}{\leftarrow}\rangle\langle RV \overset{PP}{\hookrightarrow} Chair\rangle\langle PP \overset{RV}{\leftarrow}\rangle\langle PP \overset{Chair}{\leftarrow}\rangle\varphi$.

We are going to take this example up in the end of Section 4.1.

Later on in this paper (in Section 5) we will investigate an adequate mix of the dynamic logic of propositional control with knowledge. Though related, up to now the logic of assignments was studied independently of CL-PC in the framework of extensions of public announcement logic [5, 11]. The latter aim at modelling how agents' knowledge changes when some propositional variable is publicly assigned to true or to false. There, assignments take the form $q \leftarrow \psi$, and the formula $\langle q \leftarrow \psi\rangle\varphi$ reads "φ is true after the assignment of ψ to q". So these assignments differ in two respects from ours: first, q may be assigned any formula ψ, and second, no agent performing an assignment is mentioned. As to the first point, assigning only $\top$ and $\bot$ is going to simplify our technicalities; as to the second point, in the perspective of extending a logic of agency such as CL-PC it is appealing to consider that assignments are performed by agents. We will see that there are interesting and intricate complications that arise when the agents learn that the value of a proposition has changed or when an agent publicly transfers her control over a proposition to another agent.

We believe that logics of propositional control offer a concrete and general tool for specifying interaction protocols of intelligent agents. The investigation of dynamic logics of propositional control appears bottomless. We present a few of these possible variants in Section 6.

2 PC models: models of propositional control

Throughout the paper, $\mathbb{A}$ denotes a (fixed) finite set of agents and $\mathbb{P}$ denotes a (fixed) countable set of propositional variables. A *coalition* is a subset of agents $J \subseteq \mathbb{A}$. The set $\overline{J} = \mathbb{A} \setminus J$ is the *complement* of J.

Definition 1. *A model of propositional control (PC model) is a couple* (ξ, θ) *where:*

- $\xi : \mathbb{P} \longrightarrow 2^{\mathbb{A}}$*, called an allocation;*
- $\theta : \mathbb{P} \longrightarrow \{\mathtt{tt}, \mathtt{ff}\}$*, called a valuation.*

An allocation maps propositional variables to agents. The set $\{q : i \in \xi(q)\}$ is i's part of ξ: the set of propositions under the control of i. The function ξ determines the initial allocation of propositional variables to agents, and θ determines the initial truth value of the propositional variables.

In the terminology of Gerbrandy [7], the models of CL-PC and δCL-PC are PC models where the control of every variable is both exclusive (allocated to at most one agent) and actual (allocated to at least one agent).[4]

[4] To match van der Hoek and Wooldridge's models, Gerbrandy actually needs to strengthen his abstract models with a property of full control. It says that if an agent i controls a set At_i of

Definition 2. *We say a PC model* (ξ, θ) *has* exclusive and actual control *if* $\xi(q)$ *is a singleton for every variable* $q \in \mathbb{P}$.

We are going to present several languages to talk about these structures of propositional control. Apart from the epistemic extension that is presented in Section 5, all languages will be interpreted on these models. The epistemic extension is going to be interpreted on a generalisation of PC models.

For each of the languages that we are going to introduce, we define $\mathbb{A}_\varphi$ to be the set of agents from $\mathbb{A}$ occurring in φ, and we define $\mathbb{P}_\varphi$ to be the set of variables from $\mathbb{P}$ occurring in φ.

3 $\mathsf{DL}^{\mathsf{PC}}$: dynamic logic of propositional control

In this section we define syntax and semantics of the dynamic logic of propositional control $\mathsf{DL}^{\mathsf{PC}}$, whose modal operators are $\langle q{\leftarrow}\top\rangle$ (setting q to true), $\langle q{\leftarrow}\bot\rangle$ (setting q to false), $\langle i\overset{q}{\hookrightarrow}\rangle$ (i loosing control of q), and $\langle \overset{q}{\hookrightarrow} i\rangle$ (i obtaining control of q). We prove the NP-completeness of $\mathsf{DL}^{\mathsf{PC}}$ satisfiability.

3.1 Language and semantics

Beyond the modal operators that we have introduced informally in the introduction, we also need constants $c_{i,q}$ that are read "agent i controls variable q". They are going to be useful to state the reduction axioms for our basic logic.

The language of $\mathsf{DL}^{\mathsf{PC}}$ is defined by the following BNF:

$$\varphi ::= q \mid \top \mid \bot \mid c_{i,q} \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle q{\leftarrow}\top\rangle\varphi \mid \langle q{\leftarrow}\bot\rangle\varphi \mid \langle i\overset{q}{\hookrightarrow}\rangle\varphi \mid \langle \overset{q}{\hookrightarrow} i\rangle\varphi$$

where i ranges over $\mathbb{A}$ and q ranges over $\mathbb{P}$.

We use the logical connectives $\wedge$, $\rightarrow$ and $\leftrightarrow$ with the usual meaning. We use $q{\leftarrow}\tau$ in order to talk about $q{\leftarrow}\top$ and $q{\leftarrow}\bot$ in an economic way, where τ is a placeholder for either $\top$ or $\bot$.

The *length* of a formula φ, noted $|\varphi|$, is the number of symbols used to write down φ (without $\langle$, $\rangle$, parentheses and commas). For example $|c_{i,q}| = 3$ and $|\langle q{\leftarrow}\top\rangle(q \wedge r)| = 3 + 3 = 6$.

Given a PC model (ξ, θ), we are going to update θ in order to interpret the assignment operators, and we are going to update ξ in order to interpret allocation programs.

Given a valuation θ and an allocation ξ, we define the updates $\xi^{i\overset{q}{\hookrightarrow}}$, $\xi^{\overset{q}{\hookrightarrow}i}$, $\theta^{q\leftarrow\top}$ and $\theta^{q\leftarrow\bot}$ as follows:

$$\xi^{i\overset{q}{\hookrightarrow}}(p) = \begin{cases} \xi(p) \setminus \{i\} & \text{when } p = q \\ \xi(p) & \text{otherwise.} \end{cases} \qquad \xi^{\overset{q}{\hookrightarrow}i}(p) = \begin{cases} \xi(p) \cup \{i\} & \text{when } p = q \\ \xi(p) & \text{otherwise.} \end{cases}$$

$$\theta^{q\leftarrow\top}(p) = \begin{cases} \mathtt{tt} & \text{when } p = q \\ \theta(p) & \text{otherwise.} \end{cases} \qquad \theta^{q\leftarrow\bot}(p) = \begin{cases} \mathtt{ff} & \text{when } p = q \\ \theta(p) & \text{otherwise.} \end{cases}$$

propositions, then she has a strategy for every valuation of the propositions in At_i. However, this property is commonplace here.

The truth conditions are the usual ones for $\top$, $\bot$, negation and disjunction, plus:

$$\begin{array}{ll}
(\xi,\theta) \models q & \text{iff } \theta(q) = \mathtt{tt} \\
(\xi,\theta) \models c_{i,q} & \text{iff } i \in \xi(q) \\
(\xi,\theta) \models \langle q{\leftarrow}\tau\rangle\varphi & \text{iff } (\xi,\theta^{q\leftarrow\tau}) \models \varphi \\
(\xi,\theta) \models \langle i\overset{q}{\hookrightarrow}\rangle\varphi & \text{iff } (\xi^{i\overset{q}{\hookrightarrow}},\theta) \models \varphi \\
(\xi,\theta) \models \langle \overset{q}{\hookrightarrow}i\rangle\varphi & \text{iff } (\xi^{\overset{q}{\hookrightarrow}i},\theta) \models \varphi
\end{array}$$

$\mathsf{DL}^{\mathsf{PC}}$ validity and $\mathsf{DL}^{\mathsf{PC}}$ satisfiability are defined as usual.

Since the updates $\delta \in \{i\overset{q}{\hookrightarrow}, \overset{q}{\hookrightarrow}j, q{\leftarrow}\tau\}$ are always successful and deterministic, the formulas $\langle\delta\rangle\top$ and $\langle\delta\rangle\varphi \leftrightarrow \neg\langle\delta\rangle\neg\varphi$ are valid. We have as well that $\langle\overset{q}{\hookrightarrow}j\rangle c_{j,q}$ and $\langle i\overset{q}{\hookrightarrow}\rangle\neg c_{i,q}$ are valid.

Remark 1. Note that our truth conditions for the modal operators are slightly simpler than the original ones in terms of partitions of $\mathbb{P}$ [9, 15, 8]. In particular, they naturally account for (trivial) 'auto-delegations' $i\overset{p}{\hookrightarrow}i$, while the original semantics has to explicitly distinguish this case.

3.2 $\mathsf{DL}^{\mathsf{PC}}$: complexity and completeness

Proposition 1. *The following equivalences are $\mathsf{DL}^{\mathsf{PC}}$ valid.*

$$\begin{array}{ll}
\langle q{\leftarrow}\tau\rangle p & \leftrightarrow \begin{cases} p & \textit{if } q \neq p \\ \tau & \textit{if } q = p \end{cases} \\
\langle q{\leftarrow}\tau\rangle\top & \leftrightarrow \top \\
\langle q{\leftarrow}\tau\rangle\bot & \leftrightarrow \bot \\
\langle q{\leftarrow}\tau\rangle c_{i,p} & \leftrightarrow c_{i,p} \\
\langle q{\leftarrow}\tau\rangle\neg\varphi & \leftrightarrow \neg\langle q{\leftarrow}\tau\rangle\varphi \\
\langle q{\leftarrow}\tau\rangle(\varphi_1 \vee \varphi_2) & \leftrightarrow \langle q{\leftarrow}\tau\rangle\varphi_1 \vee \langle q{\leftarrow}\tau\rangle\varphi_2
\end{array}$$

$$\begin{array}{ll}
\langle j\overset{q}{\hookrightarrow}\rangle p & \leftrightarrow p \\
\langle j\overset{q}{\hookrightarrow}\rangle\top & \leftrightarrow \top \\
\langle j\overset{q}{\hookrightarrow}\rangle\bot & \leftrightarrow \bot \\
\langle j\overset{q}{\hookrightarrow}\rangle c_{i,p} & \leftrightarrow \begin{cases} c_{i,p} & \textit{if } q \neq p \textit{ or } j \neq i \\ \bot & \textit{if } q = p \textit{ and } j = i \end{cases} \\
\langle j\overset{q}{\hookrightarrow}\rangle\neg\varphi & \leftrightarrow \neg\langle j\overset{q}{\hookrightarrow}\rangle\varphi \\
\langle j\overset{q}{\hookrightarrow}\rangle(\varphi_1 \vee \varphi_2) & \leftrightarrow \langle j\overset{q}{\hookrightarrow}\rangle\varphi_1 \vee \langle j\overset{q}{\hookrightarrow}\rangle\varphi_2
\end{array}$$

$$\begin{array}{ll}
\langle \overset{q}{\hookrightarrow} j\rangle p & \leftrightarrow p \\
\langle \overset{q}{\hookrightarrow} j\rangle \top & \leftrightarrow \top \\
\langle \overset{q}{\hookrightarrow} j\rangle \bot & \leftrightarrow \bot \\
\langle \overset{q}{\hookrightarrow} j\rangle c_{i,p} & \leftrightarrow \begin{cases} c_{i,p} & \textit{if } q \neq p \textit{ or } j \neq i \\ \top & \textit{if } q = p \textit{ and } j = i \end{cases} \\
\langle \overset{q}{\hookrightarrow} j\rangle \neg\varphi & \leftrightarrow \neg\langle \overset{q}{\hookrightarrow} j\rangle \varphi \\
\langle \overset{q}{\hookrightarrow} j\rangle (\varphi_1 \vee \varphi_2) & \leftrightarrow \langle \overset{q}{\hookrightarrow} j\rangle \varphi_1 \vee \langle \overset{q}{\hookrightarrow} j\rangle \varphi_2
\end{array}$$

These equivalences provide a complete set of reduction axioms for $\langle q \leftarrow \tau\rangle$, $\langle i \overset{q}{\hookrightarrow}\rangle$ and $\langle \overset{q}{\hookrightarrow} j\rangle$. Call *red* the mapping which iteratively applies the above equivalences from the left to the right, starting from one of the innermost modal operators. *red* pushes the dynamic operators inside the formula, and finally eliminates them when facing an atomic formula. Each step increases the length of the formula by at most 3 (when distributing dynamic operators over disjunctions). The length of the reduced formula is therefore linear in the length of the original formula.

Note that although no dynamic operator occurs in $red(\varphi)$, it is not a formula of classical propositional logic because of the control atoms $c_{i,q}$. The next proposition shows how they can be dealt with.

Proposition 2. *Let φ be a formula in the language of $\mathsf{DL}^{\mathsf{PC}}$. Then*

1. *$red(\varphi)$ has no modal operators*
2. *$|red(\varphi)| \leq 3 \times |\varphi|$*
3. *$red(\varphi) \leftrightarrow \varphi$ is $\mathsf{DL}^{\mathsf{PC}}$ valid*
4. *$red(\varphi)$ is $\mathsf{DL}^{\mathsf{PC}}$ valid iff $red(\varphi)$ is valid in classical propositional logic, where the $c_{i,q}$ in $red(\varphi)$ are understood as propositional variables.*

Theorem 1. *Satisfiability in $\mathsf{DL}^{\mathsf{PC}}$ is NP-complete.*

PROOF. Hardness is the case because $\mathsf{DL}^{\mathsf{PC}}$ is a conservative extension of classical propositional logic: for every formula φ in the language of classical propositional logic, φ is classically valid if and only if φ is $\mathsf{DL}^{\mathsf{PC}}$ valid (where it is supposed that control atoms $c_{i,q}$ are not in the language of classical propositional logic).

As to membership, items 3 and 4 of Proposition 2 guarantee that φ is $\mathsf{DL}^{\mathsf{PC}}$ satisfiable iff $red(\varphi)$ is satisfiable in classical propositional logic. Moreover, $red(\varphi)$ is a polynomial reduction from CL-PC to classical propositional logic. ■

Theorem 2. *The validities of $\mathsf{DL}^{\mathsf{PC}}$ are completely axiomatized by*

– *some axiomatization of classical propositional logic*
– *the reduction axioms of Proposition 1*
– *the rule of equivalence*

$$\textit{from } \ \varphi \leftrightarrow \varphi' \ \ \textit{infer } \ \langle\delta\rangle\varphi \leftrightarrow \langle\delta\rangle\varphi'$$

where δ is $\langle q \leftarrow \top\rangle$, $\langle q \leftarrow \bot\rangle$, $\langle i \overset{q}{\hookrightarrow}\rangle$ or $\langle \overset{q}{\hookrightarrow} j\rangle$.

PROOF. Soundness is guaranteed by Proposition 1, plus the fact that the inference rules preserve validity.

The completeness proof proceeds as follows. Suppose φ is $\mathsf{DL}^{\mathsf{PC}}$ valid. Then $red(\varphi)$ is classically valid due to Proposition 2. By the completeness of classical propositional logic, $red(\varphi)$ is also provable there. $\mathsf{DL}^{\mathsf{PC}}$ being a conservative extension of classical propositional logic, $red(\varphi)$ is provable in $\mathsf{DL}^{\mathsf{PC}}$, too. Then the formula φ must be provable in $\mathsf{DL}^{\mathsf{PC}}$, because the reduction axioms are part of our axiomatics and because the rule of substitution of equivalents is derivable.[5] ■

It is appealing to consider that assignments are performed by agents. Indeed, as for the moment, assignment are mere events. To reason about protocols of interacting agents, it is important to raise these events to the status of action. Authored assignments and control transfers are expressed in $\mathsf{DL}^{\mathsf{PC}}$ by the following abbreviations:

$$\langle q \overset{i}{\leftarrow} \tau \rangle \varphi \overset{\text{def}}{=} c_{i,q} \wedge \langle q \leftarrow \tau \rangle \varphi$$
$$\langle i \overset{q}{\hookrightarrow} j \rangle \varphi \overset{\text{def}}{=} c_{i,q} \wedge \langle i \overset{q}{\hookrightarrow} \rangle \langle \overset{q}{\hookrightarrow} j \rangle \varphi$$

'Spelling out' these abbreviations only polynomially increases the size of formulas. The modal operators that we are going to introduce in the next section are also going to be reducible to $\mathsf{DL}^{\mathsf{PC}}$, but not polynomially so (or rather, we don't know a polynomial reduction).

In the next section, we fully integrate these notions of agency into the logic.

4 Q-$\mathsf{DL}^{\mathsf{PC}}$: Quantified $\mathsf{DL}^{\mathsf{PC}}$

In this section we extend $\mathsf{DL}^{\mathsf{PC}}$ with two new constructs and modal operators. The modal operator $\langle \overset{q}{\hookrightarrow} J \rangle$ quantifies over control transfer targets in the set of agents J and $\langle q \overset{J}{\leftarrow} \rangle$ quantifies over both assignment authors in the set J and over truth values. We could as well introduce operators $\langle I \overset{q}{\hookrightarrow} \rangle$: the presentation would be symmetrical to that of $\langle \overset{q}{\hookrightarrow} J \rangle$.[6]

4.1 Language and semantics

The language of Q-$\mathsf{DL}^{\mathsf{PC}}$ is defined by adding formulas of the form $\langle q \overset{J}{\leftarrow} \rangle \varphi$ and $\langle \overset{q}{\hookrightarrow} J \rangle \varphi$ to the language of $\mathsf{DL}^{\mathsf{PC}}$, where q is a propositional variable in $\mathbb{P}$ and $J \subseteq \mathbb{A}$. $\langle q \overset{J}{\leftarrow} \rangle \varphi$ reads "the agents in J can ensure that φ by possibly changing the truth value of q", and $\langle \overset{q}{\hookrightarrow} J \rangle \varphi$ reads "φ holds after the transfer of q to some agent in J".

The truth conditions of the operators $\langle q \overset{J}{\leftarrow} \rangle$ and $\langle \overset{q}{\hookrightarrow} J \rangle$ are:

[5] The rule of substitution of equivalents is necessary in order to apply the reduction axioms 'deeply' inside formulas. It can be derived from the rules of equivalence for the classical connectives (that are derivable with classical propositional logic) and the rule of equivalence for δ of the axiomatics of $\mathsf{DL}^{\mathsf{PC}}$.

[6] We note that the strategy is similar to Borgo's in [4], who also proposes a reconstruction of coalition logic starting from a dynamic logic.

$$\begin{aligned}(\xi,\theta) \models \langle q\overset{J}{\leftarrow}\rangle\varphi \text{ iff } (\xi,\theta) \models \varphi \vee (\langle q{\leftarrow}\top\rangle\varphi \wedge \textstyle\bigvee_{i\in J} c_{i,q})\vee\\ (\langle q{\leftarrow}\bot\rangle\varphi \wedge \textstyle\bigvee_{i\in J} c_{i,q})\\ (\xi,\theta) \models \langle \overset{q}{\hookrightarrow} J\rangle\varphi \text{ iff } (\xi,\theta) \models \textstyle\bigvee_{i\in J} \langle \overset{q}{\hookrightarrow} i\rangle\varphi\end{aligned}$$

Examples of valid equivalences are: $\langle \overset{q}{\hookrightarrow}\emptyset\rangle\varphi \leftrightarrow \bot$, $\langle q\overset{\emptyset}{\leftarrow}\rangle\varphi \leftrightarrow \varphi$, $\langle q\overset{\mathbb{A}}{\leftarrow}\rangle\varphi \leftrightarrow \langle q{\leftarrow}\top\rangle\varphi\vee \langle q{\leftarrow}\bot\rangle\varphi$, $\langle q{\leftarrow}\top\rangle\varphi \leftrightarrow \langle q\overset{\mathbb{A}}{\leftarrow}\rangle(q \wedge \varphi)$, and $\langle q{\leftarrow}\bot\rangle\varphi \leftrightarrow \langle q\overset{\mathbb{A}}{\leftarrow}\rangle(\neg q \wedge \varphi)$.

We observe that the program $q{\leftarrow}$ that we discussed in the introduction can be defined as $q\overset{\mathbb{A}}{\leftarrow}$. We also observe that $\langle q\overset{\{i\}}{\leftarrow}\rangle\top$ is logically equivalent to $\top$, while both $\langle q\overset{i}{\leftarrow}\top\rangle\top$ and $\langle q\overset{i}{\leftarrow}\bot\rangle\top$ are logically equivalent to $c_{i,q}$. We also observe that $c_{i,p} \leftrightarrow (\langle\overset{\{i\}}{\leftarrow}\rangle p \wedge \langle\overset{\{i\}}{\leftarrow}\rangle\neg p)$.

The next proposition is going to be useful to prove several results.

Proposition 3. *Let φ be a Q-DL^{PC} formula. Let (ξ_1,θ_1) and (ξ_2,θ_2) be PC models agreeing on every variable outside φ, i.e. such that for every $q \notin \mathbb{P}_\varphi$, $\xi_1(q) = \xi_2(q)$ and $\theta_1(q) = \theta_2(q)$. Then $(\xi_1,\theta_1) \models \varphi$ iff $(\xi_2,\theta_2) \models \varphi$.*

4.2 Two abbreviations in Q-DL$^{\text{PC}}$

Let $P = \{q_1, \cdots, q_n\}$ be a finite set of propositional variables. We define:

$$\begin{aligned}\langle P\overset{J}{\leftarrow}\rangle\varphi &\overset{\text{def}}{=} \langle q_1\overset{J}{\leftarrow}\rangle\cdots\langle q_n\overset{J}{\leftarrow}\rangle\varphi\\ \langle\overset{P}{\hookrightarrow}J\rangle\varphi &\overset{\text{def}}{=} \langle\overset{q_1}{\hookrightarrow}J\rangle\cdots\langle\overset{q_n}{\hookrightarrow}J\rangle\varphi\end{aligned}$$

For the case $n = 0$ we suppose that $\langle\emptyset\overset{J}{\leftarrow}\rangle\varphi$ and $\langle\overset{\emptyset}{\hookrightarrow}J\rangle\varphi$ are both equal to φ. Just as for assignments with authors $q\overset{i}{\leftarrow}\top$ and $q\overset{i}{\leftarrow}\bot$, expanding these abbreviations only polynomially increases the size of the formula (precisely, the size of the rewritten formula is quadratic in the size of the original formula).

Example 2. Let us take up our running example. Consider the following formulas.

$$\begin{aligned}\mathsf{\le 4} &= \textstyle\bigwedge_{m\in RV} \bigvee_{X\subseteq PP, |X|\ge|PP|-4} \bigwedge_{n\in X, q\in\{p_n^1,p_n^2,p_n^3\}} \neg c_{m,q}\\ \mathsf{no2} &= \textstyle\bigwedge_{m\in RV} \bigwedge_{n\in PP} \bigvee_{\{q,r\}\subset\{p_n^1,p_n^2,p_n^3\}}(\neg c_{m,q} \wedge \neg c_{m,r})\\ \mathsf{gets3} &= \textstyle\bigwedge_{q\in\mathbb{P}} \bigvee_{m\in RV} c_{m,q}\\ \mathsf{noExtr} &= \textstyle\bigwedge_{m\in RV} \bigvee_{\{q,r\}\subseteq\mathbb{P}}(c_{m,q} \wedge c_{m,r} \wedge q \wedge \neg r)\\ \mathsf{acc}_n &= \textstyle\bigvee_{\{q,r\}\subseteq\{p_n^1,p_n^2,p_n^3\}}(q \wedge r)\\ \mathsf{25\%} &= \textstyle\bigwedge_{X\subset PP, |X|\ge|PP|/4} \bigvee_{n\in X} \neg\mathsf{acc}_n\end{aligned}$$

They express that: each reviewer gets at most four papers ($\le$4); no reviewer gets a paper twice (no2); each paper gets three reviewers (gets3); no reviewer can have only positive or only negative opinions (noExtr); paper n is accepted if at least two opinions are positive (acc$_n$); at most 25% of the papers can be accepted (25%).

We can verify that the program chair can distribute the papers according to the constraints and such that the acceptance rate can be obtained, by checking Q-DL$^{\text{PC}}$ validity of the following formula:

$$\bigwedge_{q\in\mathbb{P}}\left(c_{Chair,q} \wedge \bigwedge_{R\in RV} c_{R,q}\right) \longrightarrow \langle\overset{\mathbb{P}}{\hookrightarrow}RV\rangle(\mathsf{\le 4} \wedge \mathsf{no2} \wedge \mathsf{gets3} \wedge \langle\mathbb{P}\overset{RV}{\leftarrow}\rangle(\mathsf{noExtr} \wedge \mathsf{25\%}))$$

4.3 $\mathsf{Q\text{-}DL}^{\mathsf{PC}}$: complexity and completeness

Theorem 3. *The validities of $\mathsf{Q\text{-}DL}^{\mathsf{PC}}$ are completely axiomatized by*

- *the axiomatization of $\mathsf{DL}^{\mathsf{PC}}$ of Theorem 2*
- *the following axiom schemas:*

$$\langle q \overset{J}{\leftarrow} \rangle\varphi \leftrightarrow \varphi \vee (\langle q{\leftarrow}\top\rangle\varphi \wedge \textstyle\bigvee_{i\in J} c_{i,q}) \vee (\langle q{\leftarrow}\bot\rangle\varphi \wedge \textstyle\bigvee_{i\in J} c_{i,q})$$
$$\langle \overset{q}{\hookrightarrow} J\rangle\varphi \leftrightarrow \textstyle\bigvee_{i\in J} \langle \overset{q}{\hookrightarrow} i\rangle\varphi$$

Proof. The proof follows the lines of that of Theorem 2, for the appropriately defined reduction mapping *red*. ■

The axiom schemas of Theorem 3 allow to rewrite every $\mathsf{Q\text{-}DL}^{\mathsf{PC}}$ formula to a $\mathsf{DL}^{\mathsf{PC}}$ formula. Combining this with a decision procedure for $\mathsf{DL}^{\mathsf{PC}}$ we obtain a decision procedure for $\mathsf{Q\text{-}DL}^{\mathsf{PC}}$. However the rewriting step increases the length of the formula exponentially, and the decision procedure runs in exponential space. One can do better:

Theorem 4. *The model checking problem for $\mathsf{Q\text{-}DL}^{\mathsf{PC}}$ is PSPACE-complete.*

Proof. Hardness can be proved like for $\mathsf{CL\text{-}PC}$ by reducing QBF satisfiability [10].

It is easy to adapt the algorithm for model checking $\delta\mathsf{CL\text{-}PC}$ of [15] in order to deal with our operators. The procedure still does not require more that $|\varphi|$ recursive calls, where φ is the input formula, and every call requires to store only one model at a time. Hence, the algorithm runs in polynomial space. ■

Theorem 5. *Satisfiability in $\mathsf{Q\text{-}DL}^{\mathsf{PC}}$ is PSPACE-complete.*

Proof. Hardness can again be proved like for $\mathsf{CL\text{-}PC}$ [10].

As for easiness, first observe that like $\mathsf{CL\text{-}PC}$ and $\delta\mathsf{CL\text{-}PC}$, $\mathsf{Q\text{-}DL}^{\mathsf{PC}}$ has the small model property. Hence, just as in [15], given a formula φ we guess a model (ξ,θ) and check whether $(\xi,\theta) \models \varphi$. According to Theorem 4 it takes space polynomial in $|\varphi|$ to check whether $(\xi,\theta) \models \varphi$. As we had guessed (ξ,θ), satisfiability checking is in NPSPACE = PSPACE. ■

The proofs of Theorem 4 and Theorem 5 are analogous to those of the complexity results for $\mathsf{CL\text{-}PC}$ in [10]. The only difference is that in $\mathsf{CL\text{-}PC}$ a model for a formula φ can be encoded in space linear in the length of φ, while in $\mathsf{Q\text{-}DL}^{\mathsf{PC}}$, a model for a formula φ can be encoded in space quadratic in the length of φ. (This is because a propositional variable is not controlled by a single agent but by a set of agents.)

4.4 Defining van der Hoek and Wooldridge's coalition modality in $\mathsf{Q\text{-}DL}^{\mathsf{PC}}$

We now consider the coalition operators of $\mathsf{CL\text{-}PC}$. These are normal modal operators that we here write $\langle \overset{J}{\leftarrow} \rangle$.[7] They intend to grasp a local (or contingent) ability. When the set of propositional variables $\mathbb{P}$ is finite then the semantics is the same as that of the above $\langle \mathbb{P} \overset{J}{\leftarrow} \rangle$. As we allow for $\mathbb{P}$ to be infinite we have to consider $\langle \overset{J}{\leftarrow} \rangle$ to be primitive. The accessibility relations are defined as follows:

[7] The original notation is $\Diamond_J$.

$$(\xi,\theta)R_{\overset{J}{\leftarrow}}(\xi',\theta') \text{ iff } \xi' = \xi, \text{ and if } \xi(p)\cap J = \emptyset \text{ then } \theta'(p) = \theta(p)$$

and the truth condition is:

$$(\xi,\theta) \models \langle\overset{J}{\leftarrow}\rangle\varphi \text{ iff there is } (\xi',\theta') \text{ such that } (\xi,\theta)R_{\overset{J}{\leftarrow}}(\xi',\theta') \text{ and } (\xi',\theta') \models \varphi$$

The next result shows that actually there was no need to add the primitive $\langle\overset{J}{\leftarrow}\rangle$: we may restrict the variables that are assigned by J, to the set $\mathbb{P}_\varphi$ of propositional variables occurring in φ.

Proposition 4. *The schema* $\langle\overset{J}{\leftarrow}\rangle\varphi \leftrightarrow \langle\mathbb{P}_\varphi\overset{J}{\leftarrow}\rangle\varphi$ *is* Q-DL$^{\mathsf{PC}}$ *valid.*

PROOF. The right-to-left direction is straightforward.

For the other direction, suppose $(\xi,\theta) \models \langle\overset{J}{\leftarrow}\rangle\varphi$. Hence there is a model (ξ',θ') such that $(\xi,\theta)R_{\overset{J}{\leftarrow}}(\xi',\theta')$ and $(\xi',\theta') \models \varphi$. Observe that $\xi' = \xi$. Let (ξ'',θ'') be such that $\xi'' = \xi'$ and

$$\theta''(q) = \begin{cases} \theta'(q) & \text{if } q \in \mathbb{P}_\varphi \\ \theta(q) & \text{if } q \notin \mathbb{P}_\varphi \end{cases}$$

We have $(\xi,\theta)R_{\overset{J}{\leftarrow}}(\xi'',\theta'')$, and by Proposition 3 we have $(\xi'',\theta'') \models \varphi$ iff $(\xi',\theta') \models \varphi$; therefore $(\xi,\theta) \models \langle\mathbb{P}_\varphi\overset{J}{\leftarrow}\rangle\varphi$. ■

4.5 Defining Pauly's coalition modality in Q-DL$^{\mathsf{PC}}$

As we said in the introduction, the original motivation of the inventors of CL-PC was to reconstruct Pauly's Coalition Logic CL. There, the CL formula $\langle[J]\rangle\varphi$ reads "the coalition J can ensure that φ holds next, *whatever the other agents choose to do*".[8] Van der Hoek and Wooldridge proposed to identify the CL formula $\langle[J]\rangle\varphi$ with the CL-PC formula $\langle\overset{J}{\leftarrow}\rangle[\overset{\overline{J}}{\leftarrow}]\varphi$ (where as usual in modal logic $[\overset{J}{\leftarrow}]\varphi$ abbreviates $\neg\langle\overset{J}{\leftarrow}\rangle\neg\varphi$). It represents the so-called $\exists\forall$-*ability* of J for φ, and generally called α-*ability* in social choice theory [1]. As van der Hoek and Wooldridge point out, the δCL-PC formula $[\overset{\overline{J}}{\leftarrow}]\langle\overset{J}{\leftarrow}\rangle\varphi$ expresses $\forall\exists$-*ability*, alias β-*ability*.

It follows from the above Proposition 4 that $\langle\overset{J}{\leftarrow}\rangle[\overset{\overline{J}}{\leftarrow}]\varphi \leftrightarrow \langle\mathbb{P}_\varphi\overset{J}{\leftarrow}\rangle[\mathbb{P}_\varphi\overset{\overline{J}}{\leftarrow}]\varphi$ is Q-DL$^{\mathsf{PC}}$ valid. We are therefore entitled to consider from now on that $\langle[J]\rangle\varphi$ is an abbreviation of $\langle\mathbb{P}_\varphi\overset{J}{\leftarrow}\rangle[\mathbb{P}_\varphi\overset{\overline{J}}{\leftarrow}]\varphi$.

5 DEL$^{\mathsf{PC}}$: dynamic epistemic logic of propositional control

We mentioned before that the assignment operator was introduced in the context of dynamic epistemic logics, which are extensions of epistemic logic by dynamic operators

[8] The original notation is $[J]\varphi$; van der Hoek and Wooldridge use $\langle\langle J\rangle\rangle\varphi$.

such as assignments and announcements. In the same spirit we now extend our framework by modal operators of knowledge and call the logic *dynamic epistemic logic of propositional control*, abbreviated $\mathsf{DEL}^{\mathsf{PC}}$.

We will assume that assignments and control transfers are *public* events and are therefore fully observable by the agents.

We are going to give an axiomatization and a decision procedure for our extension.

5.1 Language and semantics

We consider the extension of the language of $\mathsf{DL}^{\mathsf{PC}}$ by modal operators of knowledge K_i, one per agent $i \in \mathbb{A}$, and by modal operators $\langle \varphi ! \rangle$ of truthful public announcement of φ, where φ is any formula. $K_i\varphi$ is read "i knows that φ", and $\langle \varphi ! \rangle \psi$ is read "the announcement of φ is possible, and ψ holds afterwards".

To interpret the epistemic operators we move from PC models (ξ, θ) to *epistemic PC models* of the form $M = (W, \sim, \Xi, \Theta)$, where

- W is a nonempty set of possible worlds
- $\sim : \mathbb{A} \longrightarrow (W \times W)$ associates an equivalence relation $\sim_i$ to every agent i
- $\Xi : W \longrightarrow (\mathbb{P} \longrightarrow 2^{\mathbb{A}})$ associates allocations to possible worlds
- $\Theta : W \longrightarrow (\mathbb{P} \longrightarrow \{\mathtt{tt}, \mathtt{ff}\})$ associates valuations to possible worlds

It is convenient to write $\Xi_w(p)$ and $\Theta_w(p)$ instead of $\Xi(w)(p)$ and $\Theta(w)(p)$. Every couple (Ξ_w, Θ_w) is a PC model.

We now define the updates on an epistemic model. For conciseness we introduce two notations. For $\delta \in \{i \overset{q}{\hookrightarrow}, \overset{q}{\hookrightarrow} i\}$, we note Ξ^{δ} the function that maps every state v to the updated allocation Ξ^{δ}_v. Also, $\Theta^{q \leftarrow \tau}$ is the function that maps every v to the valuation $\Theta^{q \leftarrow \tau}_v$.

Let $M = (W, \sim, \Xi, \Theta)$ be a pointed model, and let $w \in W$. Its updates are defined as follows.

$$
\begin{aligned}
&M^{i \overset{q}{\hookrightarrow}} = (W, \sim, \Xi^{i \overset{q}{\hookrightarrow}}, \Theta)\\
&M^{\overset{q}{\hookrightarrow} i} = (W, \sim, \Xi^{\overset{q}{\hookrightarrow} i}, \Theta)\\
&M^{q \leftarrow \tau} = (W, \sim, \Xi, \Theta^{q \leftarrow \tau})\\
&M^{\psi !} = (W', \sim', \Xi', \Theta') \text{ such that } \begin{cases} W' = \{v \in W \ : \ M, w \models \psi\} \\ \sim' = \sim \cap \, (W' \times W') \\ \Xi' = \Xi|_{W'} \\ \Theta' = \Theta|_{W'} \end{cases}
\end{aligned}
$$

According to our semantics, assignments and control transfers are public: when one of these events occurs then every agent updates his epistemic possibilities accordingly.

The truth conditions also have to be adapted and extended accordingly; in particular:

$$
\begin{aligned}
&M, w \models q \quad \text{iff } \Theta_w(q) = \mathtt{tt}, \text{ for } q \in \mathbb{P}\\
&M, w \models c_{i,q} \ \text{iff } i \in \Xi_w(q)\\
&M, w \models K_i\varphi \ \text{iff } M, v \models \varphi \text{ for every } v \text{ such that } w \sim_i v
\end{aligned}
$$

Moreover, for every program $\delta \in \{i\overset{q}{\hookrightarrow}, \overset{q}{\hookrightarrow}i, q{\leftarrow}\tau\}$ we define:

$$\begin{aligned} M, w \models \langle\delta\rangle\varphi \quad &\text{iff } M^{\delta}, w \models \varphi \\ M, w \models \langle\psi!\rangle\varphi \ &\text{iff } M, w \models \psi \text{ and } M^{\psi!}, w \models \varphi \end{aligned}$$

Let us call the resulting logic $\mathsf{DEL}^{\mathsf{PC}}$. Examples of $\mathsf{DEL}^{\mathsf{PC}}$ validities are $\langle q{\leftarrow}\bot\rangle K_i\neg q$, $\langle i\overset{q}{\hookrightarrow}\rangle K_i\neg c_{i,q}$, and $\langle\overset{q}{\hookrightarrow}j\rangle K_i c_{j,q}$, highlighting that assignments and control transfer are public events.

Remark 2. According to our semantics, agent i does not necessarily know whether p is allocated to j or not, and so even if $i = j$. In formulas, $c_{i,q} \wedge \neg K_i c_{i,q}$ is satisfiable. One could however easily guarantee that agents are aware of what is or is not allocated to them, by imposing the following constraint on models: if $w \sim_i w'$ then for every $q \in \mathbb{P}$, $i \in \Xi_w(q)$ iff $i \in \Xi_{w'}(q)$. Such models validate $c_{i,q} \to K_i c_{i,q}$.

5.2 Completeness and complexity

We are now able to formulate reduction axioms for our logic.

Proposition 5. *The reduction axioms of Proposition 1 are $\mathsf{DEL}^{\mathsf{PC}}$ valid, as well as the following equivalences:*

$$\begin{aligned} \langle q{\leftarrow}\tau\rangle K_i\varphi &\leftrightarrow K_i\langle q{\leftarrow}\tau\rangle\varphi \\ \langle j\overset{q}{\hookrightarrow}\rangle K_i\varphi &\leftrightarrow K_i\langle j\overset{q}{\hookrightarrow}\rangle\varphi \\ \langle\overset{q}{\hookrightarrow}j\rangle K_i\varphi &\leftrightarrow K_i\langle\overset{q}{\hookrightarrow}j\rangle\varphi \\ \langle\psi!\rangle\varphi &\leftrightarrow \psi \wedge \varphi \qquad\qquad \textit{if } \varphi \textit{ is of the form } q,\ \top,\ \bot, \textit{ or } c_{i,q} \\ \langle\psi!\rangle\neg\varphi &\leftrightarrow \psi \wedge \neg\langle\psi!\rangle\varphi \\ \langle\psi!\rangle(\varphi_1 \vee \varphi_2) &\leftrightarrow \langle\psi!\rangle\varphi_1 \vee \langle\psi!\rangle\varphi_2 \\ \langle\psi!\rangle K_i\varphi &\leftrightarrow \psi \wedge K_i\neg\langle\psi!\rangle\neg\varphi \end{aligned}$$

In the 4th equivalence, φ may more generally be any formula without modal operators.

The axiom schemas of Proposition 5 together with the reduction axioms of $\mathsf{DL}^{\mathsf{PC}}$ of Theorem 3 provide a complete set of axioms for the reduction of $\mathsf{DEL}^{\mathsf{PC}}$ to standard epistemic logic $\mathsf{S5}_n$. Call $red(\varphi)$ the resulting formula.

Theorem 6. *Let φ be a formula in the language of $\mathsf{DEL}^{\mathsf{PC}}$. Then*

1. *$red(\varphi)$ has no modal operators other than epistemic operators*
2. *$red(\varphi) \leftrightarrow \varphi$ is $\mathsf{DEL}^{\mathsf{PC}}$ valid*
3. *$red(\varphi)$ is $\mathsf{DEL}^{\mathsf{PC}}$ valid iff $red(\varphi)$ is $\mathsf{S5}_n$ valid, where the $c_{i,q}$ in $red(\varphi)$ are understood as propositional variables.*

As before, the reduction axioms allow to show completeness.

Theorem 7. *The validities of $\mathsf{DEL}^{\mathsf{PC}}$ are axiomatized by*

– *the axioms and inference rules of $\mathsf{DL}^{\mathsf{PC}}$ of Theorem 1*
– *the axioms and inference rules of $\mathsf{S5}_n$, for every modal operator K_i*
– *the axiom schemas of Proposition 5*

– *the rules of equivalence for $\langle\psi!\rangle$:*

$$\begin{array}{llll} \textit{from} & \varphi \leftrightarrow \varphi' & \textit{infer} & \langle\psi!\rangle\varphi \leftrightarrow \langle\psi!\rangle\varphi' \\ \textit{from} & \psi \leftrightarrow \psi' & \textit{infer} & \langle\psi!\rangle\varphi \leftrightarrow \langle\psi'!\rangle\varphi \end{array}$$

PROOF. The proof uses the reduction axioms and then follows the lines of that for Theorem 2. ■

While for $\mathsf{DL}^{\mathsf{PC}}$ we were able to show that the length of the reduced formula is polynomial in the length of the original formula (Proposition 2), this is no longer the case for $\mathsf{DEL}^{\mathsf{PC}}$. This is due to the form of the reduction axioms for announcements of Proposition 5 where φ occurs twice on right hand sides (cf. [12]). A reduction-based decision procedure is therefore suboptimal. However, reduction allows to establish completeness and decidability.

Theorem 8. *Satisfiability in $\mathsf{DEL}^{\mathsf{PC}}$ is PSPACE-complete if there are at least two agents, and NP-complete if there is only one agent.*

PROOF. Hardness is the case because $\mathsf{DL}^{\mathsf{PC}}$ is a conservative extension of epistemic logic (whose satisfiability problem is NP-hard for one agent and PSPACE-hard for more than one agent).

Membership can be proved by applying the abbreviation technique of [12]. ■

5.3 Authored assignment

In the end of Section 3 we had proposed the following definition $\langle i\overset{q}{\hookrightarrow}j\rangle\varphi \overset{\mathtt{def}}{=} c_{i,q} \wedge \langle i\overset{q}{\hookrightarrow}\rangle\langle\overset{q}{\hookrightarrow}j\rangle\varphi$. It identified the control transfer programs $i\overset{q}{\hookrightarrow}j$ of $\delta\mathsf{CL}\text{-}\mathsf{PC}$, with a mere *test* of $c_{i,q}$ followed by $i\overset{q}{\hookrightarrow}$ and $\overset{q}{\hookrightarrow}j$. Such an abbreviation is no longer intuitive in $\mathsf{DEL}^{\mathsf{PC}}$ because the events are public by assumption. The occurrence of the control transfer being public, every agent can eliminate her epistemic possibilities where *i did not* control *q* right before the transfer occurred. Hence, $i\overset{q}{\hookrightarrow}j$ should amount to the *public announcement* of $c_{i,q}$ followed by $i\overset{q}{\hookrightarrow}$ and $\overset{q}{\hookrightarrow}j$. This leads to the following (re)definition that suits better.

$$\langle i\overset{q}{\hookrightarrow}j\rangle\varphi \overset{\mathtt{def}}{=} \langle c_{i,q}!\rangle\langle i\overset{q}{\hookrightarrow}\rangle\langle\overset{q}{\hookrightarrow}j\rangle\varphi$$

Similarly, we had identified *i*'s assignment of *q* to τ, $q\overset{i}{\leftarrow}\tau$, with a test of $c_{i,q}$ followed be the execution of the program $q\leftarrow\tau$. Again, the occurrence of an action $q\overset{i}{\leftarrow}\bot$ being public, every agent eliminates his possibility where *i* does not control *q*. In formulas, we expect $\langle q\overset{i}{\leftarrow}\bot\rangle\neg K_j c_{i,j}$ to be unsatisfiable. However, with the abbreviation of Section 3 this formula would be satisfiable. Since we assumed that events are public, $q\overset{i}{\leftarrow}\tau$ should actually amount to the public announcement of $c_{i,q}$ followed by the public assignment $q\leftarrow\tau$. This leads to the following (re)definition that suits better.

$$\langle q\overset{i}{\leftarrow}\tau\rangle_e\varphi \overset{\mathtt{def}}{=} \langle c_{i,q}!\rangle\langle q\leftarrow\tau\rangle\varphi$$

The reader may check that $\langle q\overset{i}{\leftarrow}\bot\rangle\neg K_j c_{i,j}$ is unsatisfiable in $\mathsf{DEL}^{\mathsf{PC}}$.

6 Conclusion and perspectives

We have relaxed the original assumption of exclusive and actual control of van der Hoek and Wooldridge's coalition logic of propositional control $\mathsf{CL\text{-}PC}$ and have shown that their logic can be embedded in ours. We have moreover extended the existing logics of propositional control by several concepts stemming from dynamic epistemic logics: knowledge, assignments, and announcements. We have shown how the resulting logics relate to $\mathsf{CL\text{-}PC}$ and its extension $\delta\mathsf{CL\text{-}PC}$. We have also established their axiomatization and their complexity for satisfiability and model checking.

The remaining of the section is devoted to the sketch of some possible extensions of our logic.

The most obvious variant is to integrate PDL-style constructs to $\mathsf{DL}^{\mathsf{PC}}$. In a nutshell, it allows the following direct definitions of some programs we have considered in this paper.

$$\begin{aligned}
q\overset{J}{\leftarrow} &\overset{\text{def}}{=} \top? + \Big((\textstyle\bigvee_{i\in J} c_{i,q})?; (q\leftarrow\top + q\leftarrow\bot)\Big)\\
\overset{q}{\hookrightarrow}\{j_1 \ldots j_n\} &\overset{\text{def}}{=} \overset{q}{\hookrightarrow}j_1 + \ldots + \overset{q}{\hookrightarrow}j_n\\
\{q_i \ldots q_n\}\overset{J}{\leftarrow} &\overset{\text{def}}{=} q_1\overset{J}{\leftarrow}; \ldots ; q_n\overset{J}{\leftarrow}\\
\overset{\{q_1\ldots q_n\}}{\hookrightarrow} J &\overset{\text{def}}{=} \overset{q_1}{\hookrightarrow}J; \ldots ; \overset{q_n}{\hookrightarrow}J\\
i\overset{q}{\hookrightarrow}j &\overset{\text{def}}{=} c_{i,q}?; i\overset{q}{\hookrightarrow}; \overset{q}{\hookrightarrow}j
\end{aligned}$$

Being a dynamic logic, the logic can capture the usual instructions of structured programming. For instance, for every complex program δ_1 and δ_2:

$$\begin{aligned}
\texttt{if } \varphi \texttt{ then } \delta_1 \texttt{ else } \delta_2 &\overset{\text{def}}{=} (\varphi?; \delta_1) \cup (\neg\varphi?; \delta_2)\\
\texttt{while } \varphi \texttt{ do } \delta_1 &\overset{\text{def}}{=} (\varphi?; \delta_1)^*; \neg\varphi
\end{aligned}$$

We conjecture that the resulting logic is PSPACE-complete, too. That is, it would be no more complex than van der Hoek et al.'s $\delta\mathsf{CL\text{-}PC}$, despite a greater expressivity and the use of more general models.

Another straightforward generalization of PC models would be to *distinguish between making a variable q false and making it true*. This is related to Gerbrandy's notion of positive and negative control. In order to take that into account the function ξ has to be split up into ξ^+ and ξ^-: $\xi^+(q)$ is the set of those agents which may make q true, and $\xi^-(q)$ is the set of those agents which may make q false. In the language one may then have control atoms ${c_{i,q}}^+$ and ${c_{i,q}}^-$ which are interpreted as expected. Moreover, one may have modal operators $i\overset{+q}{\hookrightarrow}$, $i\overset{-q}{\hookrightarrow}$, $\overset{+q}{\hookrightarrow}i$, and $\overset{-q}{\hookrightarrow}i$ of loosing or obtaining positive or negative control. This then may be taken into account by defining e.g. authored assignment as $\langle q\overset{i}{\leftarrow}\top\rangle\varphi \overset{\text{def}}{=} {c_{i,q}}^+ \land \langle q\leftarrow\top\rangle\varphi$.

Concerning the epistemic extension we observe that while the extension by public announcements is technically straightforward it does not account for *announcements that are made by agents*. In a first approach one might identify the announcement of φ by i with the public announcement of $K_i\varphi$; however, a full analysis requires more work.

In a similar spirit, one might extend our logic by *event models* [3, 2, 6], which account for incomplete (and even erroneous) perception of events by agents. This should be possible without difficulties. As a teaser, it would allow to adequately handle protocols with intricate epistemic aspects such as for instance a variant of Example 1 with double blind reviewing, or a more realistic setting where the protocol is specified in a way such that a reviewer neither know what the allocations of the other members of the committee are, nor which opinions were already expressed about a paper.

Acknowledgements

Thanks for several suggestions are due to Guillaume Aucher, Pierre Marquis, Leon van der Torre and Emil Weydert. We are also grateful to the reviewers of LIS@ESSLLI for their comments that helped to improve the paper. Andreas Herzig's research on this paper is partly funded by the ANR project *Social trust analysis and formalization* (ForTrust). Nicolas Troquard's is supported by the EPSRC grant EP/E061397/1.

References

1. Joseph Abdou and Hans Keiding. *Effectivity functions in social choice*. Kluwer Academic, 1991.
2. Alexandru Baltag and Lawrence S. Moss. Logics for epistemic programs. *Synthese*, 139(2):165–224, 2004.
3. Alexandru Baltag, Lawrence S. Moss, and Slawomir Solecki. The logic of public announcements, common knowledge, and private suspicions. In *Proc. TARK'98*, pages 43–56, 1998.
4. Stefano Borgo. Coalitions in action logic. In *Proc. IJCAI'07*, pages 1822–1827, 2007.
5. Hans P. van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. Dynamic epistemic logic with assignment. In *Proc. AAMAS'05*, pages 141–148, 2005.
6. Hans P. van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*. Kluwer Academic Publishers, 2007.
7. Jelle Gerbrandy. Logics of propositional control. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pages 193–200. ACM, 2006.
8. Wiebe van der Hoek, Dirk Walther, and Michael Wooldridge. On the logic of cooperation and the transfer of control. *J. of AI Research (JAIR)*, 37:437–477, 2010.
9. Wiebe van der Hoek and Michael Wooldridge. On the dynamics of delegation, cooperation and control: a logical account. In *Proc. AAMAS'05*, 2005.
10. Wiebe van der Hoek and Michael Wooldridge. On the logic of cooperation and propositional control. *Artif. Intell.*, 164(1-2):81–119, 2005.
11. Barteld Kooi. Expressivity and completeness for public update logic via reduction axioms. *Journal of Applied Non-Classical Logics*, 17(2):231–253, 2007.
12. Carsten Lutz. Complexity and succintness of public announcement logic. In *Proc. AAMAS'06*, pages 137–144, 2006.
13. Nicolas Troquard, Wiebe van der Hoek, and Michael Wooldridge. A logic of propositional control for truthful implementations. In *Proc. TARK'09*, pages 237–246, 2009.
14. Nicolas Troquard, Wiebe van der Hoek, and Michael Wooldridge. A logic of games and propositional control. In *Proc. AAMAS'09*, pages 961–968, 2009.
15. Dirk Walther. *Strategic Logics: Complexity, Completeness and Expressivity*. PhD thesis, University of Liverpool, 2007.

Formal Definitions and Complexity Results for Trust Relations and Trust Domains⋆

Simon Kramer⋆⋆, Rajeev Goré, and Eiji Okamoto

[1] University of Tsukuba, Japan
simon.kramer@a3.epfl.ch
[2] Australian National University
rajeev.gore@anu.edu.au
[3] University of Tsukuba, Japan
okamoto@risk.tsukuba.ac.jp

Abstract. We propose computational, declarative definitions of the concepts of weak and strong *trust relations* between interacting agents, and *trust domains* of trust-related agents in distributed systems. Our definitions yield computational complexity results for deciding potential and actual trust relationships and membership in trust domains, as well as a positive (negative) compositionality result for strong (weak) trust domains. Our defining principle for weak and strong trust is (common) belief in and knowledge of agent correctness, respectively.

Keywords computability, compositionality, and scalability of trust and trustworthiness; computer-aided decision making (CADM); dependable distributed or multi-agent systems; modal logics of belief and knowledge.

1 Introduction

1.1 Motivation

Dependable distributed systems guarantee their functionality both in spite of and thanks to the technologies that implement these systems, and the (man and/or machine) agents[4] that partake in them. The functionality guarantee is conditioned on *naming* and *number*, i.e., on agent identity for the information security aspect [2, Chapter 6] (anonymity, pseudonymity), and on a minimal number of correct (or dually, a maximal number of faulty/corrupt) agents for the aspects of fault tolerance [26] (classical distributed computation) and corruption tolerance [37] (secure multiparty computation). The notion of *agent correctness* (e.g., as induced by a *policy*) in turn depends on the system itself. For example, agent correctness can include: absence of crash (liveness); absence of cryptographic-key compromise; algorithmic, legal, and policy compliance; due

⋆ A technical-report version of this paper including the appendix appears in [22].

⋆⋆ This (corresponding) author's contribution was funded with Grant P 08742 from the Japan Society for the Promotion of Science.

[4] network nodes, parties, processes, processors, real or virtual machines, users, etc.

debt repayment in credit banking; fairness in scheduling; etc. In any case, agent correctness captures the *predictability* of each correct partaking agent that guarantees the functionality of the system to all, correct or faulty, partaking agents. We stress that this predictability is one of agent behaviour rather than mental attitude. All that matters is the behavioural *effect* of an attitude, not the attitude itself. (An attitude may have no or no relevant effect.) In sum, system functionality depends on agent correctness, and agents depend on each other via each other's correctness. Whence the social need, called *trust*, to know whether or not someone behaves correctly. At first sight, such knowledge may seem difficult to attain. Yet in our standard understanding of knowledge defined in terms of indistinguishability of system states, an agent a simply attains knowledge about a fact ϕ in a given state s as soon as ϕ also holds in all the states that are indistinguishable from s to a (cf. Section 2). In particular, a need not be in control of the system.

The concept of trust has at least three different aspects, namely trust *relations* and *domains,* and trust *management.* Our intuitions of them are as follows.

Trust relations An agent a trusts an agent b when a believes or even knows that b behaves correctly, independently of b's mental attitude. Hence, defining trust based on the correct behaviour of agents is more general than defining trust based on their mental attitude. The reader interested in the latter is referred to [11], which is a substantial study of various kinds of trust relations based on belief in and knowledge of mental attitudes.

Trust domains A trust domain is a community of mutually trusting agents with the common belief in or even knowledge of the trust relationships in the community. Informally, a statement ϕ is common belief or knowledge in a community $\mathcal{C}$ when all agents in $\mathcal{C}$ believe or know that ϕ is true (call this new statement ϕ'), all believe or know that ϕ' is true (call this new statement ϕ''), all believe or know that ϕ'' is true (call this new statement ϕ'''), etc. Notice the *mutual awareness* among agents w.r.t. the commonality of their knowledge or belief. This awareness is computationally costly (cf. Section 3). More intuitions on common knowledge in distributed systems can be found in [15].

Trust management Trust management is (cf. [30] for a recent survey):

1. the organisation of trust relations into trust domains (compartments), i.e., the sociology
2. the coordination of trust-building actions, i.e., the flow of trust (partial dehierarchisation and decompartmentation, e.g., by building reputation [31]).

The organisation of trust relations into trust domains requires the ability to decide whether or not a given relation is a trust relation, and a given domain is a trust domain. Ideally, this ability appeals to formal definitions and decidability results for trust relations and trust domains, in order to support the human brain with *computer-aided decision making (CADM)*, as motivated by the following example.

Example 1 (Group size and the human brain). According to [32], “150 is the cognitive limit to the number of people a human brain can maintain a coherent social relationship with”. “More generally, there are several layers of natural human group size that increase with a ratio of approximately three: 5, 15, 50, 150, 500, and 1,500”. And “[a]s group sizes grow across these boundaries, they have more externally imposed infrastructure—and more formalized security systems.”

The motivation for formal definitions now follows from the assumption that trust is a fundamental element of any coherent social relationship; and the motivation for CADM (requiring decidability results) additionally from the desire to extend the cognitive limit of the human brain.

It turns out that deciding trust relationships can be tractable with the aid of modern computers. However, deciding membership in trust domains, though computationally possible in theory, is computationally intractable in practice, even with the aid of clusters or clouds of (super-)computers. What is worse: not only can we not make use of the promised power of cloud computing[5] [1] for deciding membership in trust domains in general, but also in particular when the candidate domains are the computing clouds themselves!

Example 2 (Trust and Cloud Computing). According to [21], in cloud computing, “users are universally required to accept the underlying premise of trust. In fact, some have conjectured that trust is the biggest concern facing cloud computing. Nowhere is the element of trust more apparent than in security, and many believe trust and security to be synonymous.” Also according to [28]: “The growing importance of cloud computing makes it increasingly imperative that we grapple with the meaning of trust in the cloud and how the customer, provider, and society in general establish that trust.”

Indeed, the automatic validation of the underlying premise of cloud computing, i.e., trust, is an intractably big concern for the clouds themselves, as indicated above. However, the validation of trust can of course still be a tractably big concern for the relations between the cloud members. Anyway, what remains formally elusive is the *declarative meaning of trust.* As a matter of fact, the vast majority of the research literature on trust focuses on *how* to (operationally) establish and maintain trust of some form (e.g., with protocols, recommendation/reputation systems, reference monitors, trusted computing bases [14], etc.), but without actually defining *what* trust of that form (declaratively) means. And the very few works that do attempt declarative definitions of forms of trust do not provide insights in the decidability or complexity of trust domains, or applications to security such as trust in cryptographic-key management (cf. Section 4)

The bottom line is that declaratively defining the meaning of trust and obtaining estimates of the decidability or complexity of trust can be difficult. Yet formally defining trust in terms of (declarative) belief or knowledge of behavioural correctness turns out to be natural, since humans often naturally refer

[5] Cloud computing is automated outsourcing of IT (data storage, calculation routines) into evolving opaque clouds of anonymous computing units.

to these or similar notions when informally explaining what they mean by trust. In distributed or multi-agent systems, attaining an *individual* consciousness of trust in terms of belief (weak trust) or knowledge (strong trust) from agent to agent can be computationally *tractable*. Whereas attaining a *collective* consciousness (mutual awareness) of trust in terms of *common* belief or knowledge in a greater domain (e.g., a cluster, cloud, or other collectives) of agents is computationally *in*tractable. Computationally speaking, *collective trust does not scale*. Trust domains should be family-sized, so to speak.

1.2 Goal

Our goal is four-fold, namely:

1. to provide *computational, declarative definitions for trust relations* between interacting agents, *and trust domains* of trust-related agents in distributed systems
2. to obtain *computational complexity results* for deciding trust relationships and membership in trust domains
3. to instantiate our concepts of trust relations and trust domains in four major applications of trust, namely: Trusted Third Parties (TTPs), the Web of Trust, Public-Key Infrastructures (PKIs), and Identity-Based Cryptography (ID-Based Cryptography).
4. to point out limited computational means for *building trust*, and by that, building up trust relations and trust domains in computer-communication networks.

Contribution To the best of our knowledge, general formal definitions for trust domains, general complexity results for trust relations as well as trust domains, a positive (negative) compositionality result for strong (weak) trust domains, and a generic formalisation of trust in TTPs, the Web of Trust, PKIs, and ID-Based Cryptography are all novel. The resulting (in)tractability insights are of great practical importance for cryptographic-key management, and could may well be similarly important for computing clouds, which we believe should be conceived as trust domains, and for which trust is the underlying premise [21].

1.3 Methodology

Our methodology is to develop our formal definitions for trust relations and trust domains in a framework that is a semantically defined, standard modal logic of belief and knowledge (cf. Section 2). In that, we are interested in the *descriptive* (as opposed to deductive) use of an off-the-shelf, general-purpose (as opposed to special-purpose, e.g., the famous BAN-logic, which uses but does not define trust) logic that is as simple as possible and as complex as necessary—both syntactically and semantically as well as computationally. Our *defining principle* for weak and strong trust is belief in and knowledge of agent correctness, respectively. We then derive our complexity results for deciding trust relationships and membership in trust domains by reduction to known results for the complexity of belief and knowledge (cf. Section 3). In spite of the practical significance of our results, their derivation is quite simple (which increases their value), thanks to our modal logical framework. The difficulty was to find what we believe to be an interesting formal point of view on trust, which happens to be modal. Other points of view have, to the best of our knowledge, not resulted in general (in)tractability insights into trust, nor a positive (negative) compositionality result for strong (weak) trust domains, nor a generic formalisation of TTPs as well as trust in the Web of Trust, PKIs, and ID-Based Cryptography (cf. Section 4).

2 Formal definitions

We develop our formal definitions of trust relations and trust domains in a framework that is a semantically defined, standard modal logic of common belief and knowledge. The logic is *parametric* in the notion of agent correctness, to be instantiated for each considered distributed system (cf. Appendices C.2–C.4 for our three examples).

Let S designate the considered distributed system (e.g., of sub-systems).

Definition 1 (Framework). *Let*

- $\mathcal{A}$ *designate an arbitrary finite set of unique* agent names[6] *a, b, c, etc.*
- $\mathcal{C} \subseteq \mathcal{A}$ *denote (finite and not necessarily disjoint) communities of agents (referred to by their name)*
- $\mathcal{P} := \{\ \mathsf{correct}(a) \mid a \in \mathcal{A}\ \}$ *designate our (finite) set of* atomic propositions *P for referring to agent correctness*
- $\mathcal{L} \ni \phi\ ::=\ P \bigm| \neg\phi \bigm| \phi \wedge \phi \bigm| \mathsf{CB}_{\mathcal{C}}(\phi) \bigm| \mathsf{CK}_{\mathcal{C}}(\phi)$ *designate our modal language of formulae ϕ, with $\mathsf{CB}_{\mathcal{C}}(\phi)$ for "it is common belief in the community $\mathcal{C}$ that ϕ", and $\mathsf{CK}_{\mathcal{C}}(\phi)$ for "it is common knowledge in the community $\mathcal{C}$ that ϕ".*

Then given the set $\mathcal{S}$ (the state space) of system states *s induced by S (e.g., via a reachability or, in modal jargon, temporal accessibility relation)*[7]*, we define the* satisfaction relation $\models$ *of our framework in Table 1. There,*

- *":iff" abbreviates "by definition, if and only if"*
- $(\mathfrak{S}, \mathcal{V})$ *designates the (modal)* model *of our framework*
- $\mathfrak{S} := (\mathcal{S}, \{\mathrm{D}_a\}_{a\in\mathcal{A}}, \{\mathrm{E}_a\ \}_{a\in\mathcal{A}})$ *designates the (modal)* frame *with appropriate (for the system S)*
  - *serial*[8]*, transitive, and Euclidean*[9] *relations $\mathrm{D}_a \subseteq \mathcal{S} \times \mathcal{S}$ of* doxastic accessibility *(used for defining* belief*)*

[6] i.e., agent names injectively map to agents (here, names are identifiers)

[7] For example, suppose that there is a set $\mathcal{S}_i$ of initial states for every system S, T designates the system's reachability or, synonymously, temporal accessibility relation, and T^* designates the reflexive transitive closure of T. Then, $\mathcal{S}$ is induced by S in that sense that $\mathcal{S} := \{\ s \mid \text{there is } s_i \in \mathcal{S}_i \text{ such that } s_i\ \mathrm{T}^*\ s\ \}$.

[8] for all $s \in \mathcal{S}$, there is $s' \in \mathcal{S}$ s.t. $s\ \mathrm{D}_a\ s'$

[9] for all $s, s', s'' \in \mathcal{S}$, if $s\ \mathrm{D}_a\ s'$ and $s\ \mathrm{D}_a\ s''$ then $s'\ \mathrm{D}_a\ s''$

Table 1. Satisfaction relation

$(\mathfrak{S},\mathcal{V}),s\models P$	:iff $s\in\mathcal{V}(P)$
$(\mathfrak{S},\mathcal{V}),s\models\neg\phi$	:iff not $(\mathfrak{S},\mathcal{V}),s\models\phi$
$(\mathfrak{S},\mathcal{V}),s\models\phi\land\phi'$	:iff $(\mathfrak{S},\mathcal{V}),s\models\phi$ and $(\mathfrak{S},\mathcal{V}),s\models\phi'$
$(\mathfrak{S},\mathcal{V}),s\models\mathsf{CB}_{\mathcal{C}}(\phi)$	:iff for all $s'\in\mathcal{S}$, if $s\ \mathrm{D}^+_{\mathcal{C}}\ s'$ then $(\mathfrak{S},\mathcal{V}),s'\models\phi$
$(\mathfrak{S},\mathcal{V}),s\models\mathsf{CK}_{\mathcal{C}}(\phi)$	:iff for all $s'\in\mathcal{S}$, if $s\ \mathrm{E}^*_{\mathcal{C}}\ s'$ then $(\mathfrak{S},\mathcal{V}),s'\models\phi$

- *equivalence relations* $\mathrm{E}_a\subseteq\mathcal{S}\times\mathcal{S}$ *of* epistemic accessibility *(e.g., state* indistinguishability*, used for defining* knowledge*)*

such that $\mathrm{D}_a\subseteq\mathrm{E}_a$ *for any* $a\in\mathcal{A}$

– $\mathcal{V}:\mathcal{P}\to 2^{\mathcal{S}}$ *designates the* valuation function *(returning for every* $P\in\mathcal{P}$ *the set of states where* P *is true) to be defined according to the appropriate notion of agent correctness for the system* S *(e.g., see Appendix C)*

– $\mathrm{D}^+_{\mathcal{C}}$ *designates the transitive closure of* $\bigcup_{a\in\mathcal{C}}\mathrm{D}_a$

– $\mathrm{E}^*_{\mathcal{C}}$ *designates the reflexive transitive closure of* $\bigcup_{a\in\mathcal{C}}\mathrm{E}_a$.

Note that defining (common) belief and knowledge abstractly with a serial, transitive, and Euclidean relation, and an equivalence relation, respectively, has emerged as a common practice that gives greater generality over more concrete approaches [27, Section 7.1]: the concrete definitions of the accessibility relations can be freely determined for a given distributed system provided they comply with the prescribed, characteristic properties. Typically, these definitions involve the projection of global states onto agents' local views [13]. For example, let $a\in\mathcal{A}$, and let π_a designate such a projection (function) for a. Then, epistemic accessibility in the sense of state indistinguishability can be defined such that for all $s,s'\in\mathcal{S}$,

$$s\ \mathrm{E}_a\ s' \text{ :iff } \pi_a(s)=\pi_a(s'),$$

which guarantees that E_a is an equivalence relation. Doxastic accessibility $\mathrm{D}_a\subseteq\mathrm{E}_a$ can be defined from E_a by weakening the reflexivity of E_a to seriality as appropriate for the considered application.

Further note the following macro-definitions: $\phi\lor\phi':=\neg(\neg\phi\land\neg\phi')$, $\top:=\mathsf{correct}(a)\lor\neg\mathsf{correct}(a)$, $\bot:=\neg\top$, $\phi\to\phi':=\neg\phi\lor\phi'$, $\phi\leftrightarrow\phi':=(\phi\to\phi')\land(\phi'\to\phi)$, $\mathsf{B}_a(\phi):=\mathsf{CB}_{\{a\}}(\phi)$ (for "a believes that ϕ"), and $\mathsf{K}_a(\phi):=\mathsf{CK}_{\{a\}}(\phi)$ (for "a knows that ϕ"). Likewise we now obtain our declarative definitions of weak and strong trust relations and domains as mere macro-definitions, i.e., as simple syntactic constructions of semantically defined building blocks (cf. Table 2, conjunction over $\mathcal{C}=\emptyset$ being $\top$). Note that we could dually define weak and strong *dis*trust relations, i.e., as belief in and knowledge of agent *in*correctness, respectively, which use *verb-phrase* negation. The reader is invited not to confuse distrust with *absence of trust*, e.g., with $\neg\mathsf{B}_a(\mathsf{correct}(b))$ or $\neg\mathsf{K}_a(\mathsf{correct}(b))$, which use *sentence* negation. Further note that we could define *weak-strong* (dis)trust domains and *strong-weak* (dis)trust domains, i.e., as the common knowledge of weak (dis)trust relations, and the common belief of strong (dis)trust relations,

Table 2. Weak and strong trust relations and domains

$a\ \mathsf{wTrusts}\ b := \mathsf{B}_a(\mathsf{correct}(b))$	a weakly trusts b
$\mathsf{wTD}(\mathcal{C}) := \mathsf{CB}_{\mathcal{C}}(\bigwedge_{a,b\in\mathcal{C}} a\ \mathsf{wTrusts}\ b)$	$\mathcal{C}$ is a weak trust domain.
$a\ \mathsf{sTrusts}\ b := \mathsf{K}_a(\mathsf{correct}(b))$	a strongly trusts b
$\mathsf{sTD}(\mathcal{C}) := \mathsf{CK}_{\mathcal{C}}(\bigwedge_{a,b\in\mathcal{C}} a\ \mathsf{sTrusts}\ b)$	$\mathcal{C}$ is a strong trust domain.

respectively. The difference between weak and strong trust is induced by the difference between belief and knowledge, respectively: weak trust possibly is wrong (i.e., mistaken belief), whereas strong trust necessarily is right (i.e., truthful belief).

Social networking systems furnish evidence for the adequacy of defining trust domains in terms of common knowledge or at least belief. As a matter of fact, the enumeration of "friends" on a member page in such systems constitutes a public announcement to the readers of that page who are logged in, who see all logged-in readers, etc. And it is common knowledge in the community of (dynamic) epistemic logicians that public announcements of (verifiable) elementary facts induce common knowledge within the addressed public (cf. [34] for a public announcement). So, suppose that you are a member of Facebook, and $\mathcal{C}$ designates the set consisting of you and those of your "friends" that are enumerated on your member page. Further, fix the current moment in time, and call it s. (We may talk about time here; see Footnote 7.) Then the formula $\bigwedge_{a,b\in\mathcal{C}} a\ \mathsf{sTrusts}\ b \leftrightarrow \mathsf{sTD}(\mathcal{C})$ is true at s (with no outermost $\mathsf{CK}_{\mathcal{C}}$ operator on the left since the aforementioned public announcement implies it). The formula is a (bi-)conditional because we have not fixed the notion of agent correctness for Facebook (*they* should). Note that the trust relations between you and your "friends" are symmetric, because each "friend" had to give their consent for having the privilege of being enumerated as such on your member page in Facebook.

Definition 2 (Truth & Validity). *The formula ϕ is* true *(or* satisfied*) in the model $(\mathfrak{S},\mathcal{V})$ at the state $s\in\mathcal{S}$:iff $(\mathfrak{S},\mathcal{V}),s\models\phi$. The formula ϕ is* satisfiable *in the model $(\mathfrak{S},\mathcal{V})$:iff there is $s\in\mathcal{S}$ such that $(\mathfrak{S},\mathcal{V}),s\models\phi$. The formula ϕ is* globally true *(or* globally satisfied*) in the model $(\mathfrak{S},\mathcal{V})$, written $(\mathfrak{S},\mathcal{V})\models\phi$, :iff for all $s\in\mathcal{S}$, $(\mathfrak{S},\mathcal{V}),s\models\phi$. The formula ϕ is* satisfiable *:iff there is a model $(\mathfrak{S},\mathcal{V})$ and a state $s\in\mathcal{S}$ such that $(\mathfrak{S},\mathcal{V}),s\models\phi$. The formula ϕ is* valid, *written $\models\phi$, :iff for all models $(\mathfrak{S},\mathcal{V})$, $(\mathfrak{S},\mathcal{V})\models\phi$. (cf. [5])*

Fact 1 (Common belief) *Being defined in terms of a serial, transitive, and Euclidean relation, $\mathsf{CB}_{\{a\}}$ is KD45 for any $a\in\mathcal{C}\subseteq\mathcal{A}$, i.e.:*

K: $\models \mathsf{CB}_{\mathcal{C}}(\phi\to\phi')\to(\mathsf{CB}_{\mathcal{C}}(\phi)\to\mathsf{CB}_{\mathcal{C}}(\phi'))$ *(Kripke's law)*
D: $\models \mathsf{CB}_{\{a\}}(\phi)\to\neg\mathsf{CB}_{\{a\}}(\neg\phi)$ *(consistency of beliefs, seriality)*
4: $\models \mathsf{CB}_{\mathcal{C}}(\phi)\to\mathsf{CB}_{\mathcal{C}}(\mathsf{CB}_{\mathcal{C}}(\phi))$ *(positive introspection, transitivity)*
5: $\models \neg\mathsf{CB}_{\mathcal{C}}(\phi)\to\mathsf{CB}_{\mathcal{C}}(\neg\mathsf{CB}_{\mathcal{C}}(\phi))$ *(negative introspection, Euclideanness)*
N: *if* $\models\phi$ *then* $\models\mathsf{CB}_{\mathcal{C}}(\phi)$ *(necessitation).*

Further, let $\mathsf{EB}_{\mathcal{C}}(\phi) := \bigwedge_{a\in\mathcal{C}} \mathsf{B}_a(\phi)$ *("everybody in* $\mathcal{C}$ *believes that* ϕ*"). Then:*

- $\models \mathsf{CB}_{\mathcal{C}}(\phi) \rightarrow \mathsf{EB}_{\mathcal{C}}(\phi)$
- $\models \mathsf{CB}_{\mathcal{C}}(\phi) \rightarrow \mathsf{EB}_{\mathcal{C}}(\mathsf{CB}_{\mathcal{C}}(\phi))$
- $\models \mathsf{CB}_{\mathcal{C}}(\phi \rightarrow \mathsf{EB}_{\mathcal{C}}(\phi)) \rightarrow (\mathsf{EB}_{\mathcal{C}}(\phi) \rightarrow \mathsf{CB}_{\mathcal{C}}(\phi))$.

For details see [27, Section 7.1].

The difference between belief and knowledge is that belief possibly is wrong (cf. the **D** law), whereas knowledge necessarily is right (cf. the following **T** law).

Fact 2 (Common knowledge) *Being defined in terms of an equivalence relation,* $\mathsf{CK}_{\mathcal{C}}$ *is S5 for any* $\mathcal{C} \subseteq \mathcal{A}$*, i.e.:*

K: $\models \mathsf{CK}_{\mathcal{C}}(\phi \rightarrow \phi') \rightarrow (\mathsf{CK}_{\mathcal{C}}(\phi) \rightarrow \mathsf{CK}_{\mathcal{C}}(\phi'))$ *(Kripke's law)*
T: $\models \mathsf{CK}_{\mathcal{C}}(\phi) \rightarrow \phi$ *(truth law, reflexivity)*
4: $\models \mathsf{CK}_{\mathcal{C}}(\phi) \rightarrow \mathsf{CK}_{\mathcal{C}}(\mathsf{CK}_{\mathcal{C}}(\phi))$ *(positive introspection)*
5: $\models \neg\mathsf{CK}_{\mathcal{C}}(\phi) \rightarrow \mathsf{CK}_{\mathcal{C}}(\neg\mathsf{CK}_{\mathcal{C}}(\phi))$ *(negative introspection)*
N: *if* $\models \phi$ *then* $\models \mathsf{CK}_{\mathcal{C}}(\phi)$ *(necessitation).*

Further, let $\mathsf{EK}_{\mathcal{C}}(\phi) := \bigwedge_{a\in\mathcal{C}} \mathsf{K}_a(\phi)$ *("everybody in* $\mathcal{C}$ *knows that* ϕ*"). Then:*

- $\models \mathsf{CK}_{\mathcal{C}}(\phi) \rightarrow \mathsf{EK}_{\mathcal{C}}(\mathsf{CK}_{\mathcal{C}}(\phi))$
- $\models \mathsf{CK}_{\mathcal{C}}(\phi \rightarrow \mathsf{EK}_{\mathcal{C}}(\phi)) \rightarrow (\phi \rightarrow \mathsf{CK}_{\mathcal{C}}(\phi))$.

For details see [27, Section 7.1].

Note that depending on the properties of the employed communication lines, common knowledge may have to be pre-established, i.e., off those lines [15].

Fact 3 (Knowledge versus belief) *For all* $\mathcal{C} \subseteq \mathcal{A}$*,* $\models \mathsf{CK}_{\mathcal{C}}(\phi) \rightarrow \mathsf{CB}_{\mathcal{C}}(\phi)$*. In particular when* $\mathcal{C} = \{a\}$*,* $\models \mathsf{K}_a(\phi) \rightarrow \mathsf{B}_a(\phi)$*.*

Proof. By the fact that for all $a \in \mathcal{A}$, $\mathrm{D}_a \subseteq \mathrm{E}_a$ (cf. Definition 1).

The following corollary is immediate.

Corollary 1 (Strong versus weak trust).

1. *For all* $a, b \in \mathcal{A}$*,* $\models a \mathrel{\mathsf{sTrusts}} b \rightarrow a \mathrel{\mathsf{wTrusts}} b$*.*
2. *For all* $\mathcal{C} \subseteq \mathcal{A}$*,* $\models \mathsf{sTD}(\mathcal{C}) \rightarrow \mathsf{wTD}(\mathcal{C})$*.*

Trust relations and trust domains can be related as follows.

Proposition 1 (Trust relations and domains). *In trust domains, trust relations are* universal *(i.e., correspond to the Cartesian product on those domains). That is, for all* $a, b \in \mathcal{C}$*,* $\models \mathsf{wTD}(\mathcal{C}) \rightarrow a \mathrel{\mathsf{wTrusts}} b$ *and* $\models \mathsf{sTD}(\mathcal{C}) \rightarrow a \mathrel{\mathsf{sTrusts}} b$*.*

Proof. Almost by definition of weak and strong trust relations and domains.

Hence in trust domains, trust relations are equivalence relations. (The universal relation contains all other relations.)

Corollary 2 (Trust relations and domains). *In trust domains, trust relations are ($a, b, c \in \mathcal{C} \subseteq \mathcal{A}$):* reflexive *(i.e.,* $\models \mathsf{wTD}(\mathcal{C}) \to a \ \mathsf{wTrusts}\ a$ *and* $\models \mathsf{sTD}(\mathcal{C}) \to a \ \mathsf{sTrusts}\ a$*),* symmetric *(i.e.,* $\models \mathsf{wTD}(\mathcal{C}) \to (a \ \mathsf{wTrusts}\ b \to b \ \mathsf{wTrusts}\ a)$ *and* $\models \mathsf{sTD}(\mathcal{C}) \to (a \ \mathsf{sTrusts}\ b \to b \ \mathsf{sTrusts}\ a)$*), and* transitive *(i.e.,* $\models \mathsf{wTD}(\mathcal{C}) \to ((a \ \mathsf{wTrusts}\ b \wedge b \ \mathsf{wTrusts}\ c) \to a \ \mathsf{wTrusts}\ c)$ *and* $\models \mathsf{sTD}(\mathcal{C}) \to ((a \ \mathsf{sTrusts}\ b \wedge b \ \mathsf{sTrusts}\ c) \to a \ \mathsf{sTrusts}\ c)$*).*

A more interesting condition for the transitivity of trust relations than their universality is knowledge (which is implied in strong trust domains by common knowledge) in the following sense.

Lemma 1 (Transitivity Lemma).

1. $\models \mathsf{B}_a(b \ \mathsf{sTrusts}\ c) \to a \ \mathsf{wTrusts}\ c$
2. $\models \mathsf{K}_a(b \ \mathsf{sTrusts}\ c) \to a \ \mathsf{sTrusts}\ c.$

Proof. The first validity follows from $\mathbf{T}(\mathsf{K}_b)$ and $\models (\mathsf{B}_a(\phi) \wedge (\phi \to \phi')) \to \mathsf{B}_a(\phi')$, and the second from $\mathbf{T}(\mathsf{K}_b)$ and $\models (\mathsf{K}_a(\phi) \wedge (\phi \to \phi')) \to \mathsf{K}_a(\phi')$.

Note that b acts as a *reference* of c's trustworthiness to a. This is an important concept for applications. An example is the construction of transitive *trust paths,* e.g., of length 3: $\models \mathsf{B}_a(b \ \mathsf{sTrusts}\ c) \to ((a \ \mathsf{wTrusts}\ b \wedge b \ \mathsf{wTrusts}\ c) \to a \ \mathsf{wTrusts}\ c)$ and $\models \mathsf{K}_a(b \ \mathsf{sTrusts}\ c) \to ((a \ \mathsf{sTrusts}\ b \wedge b \ \mathsf{sTrusts}\ c) \to a \ \mathsf{sTrusts}\ c)$, respectively.

Fact 4 *1.* $\not\models \mathsf{B}_a(b \ \mathsf{wTrusts}\ c) \to a \ \mathsf{wTrusts}\ c$
2. $\not\models \mathsf{K}_a(b \ \mathsf{wTrusts}\ c) \to a \ \mathsf{wTrusts}\ c.$

Proof. By the absence of the $\mathbf{T}$-law for B_b.

Here are some simple facts about trust domains.

Proposition 2 (Trust domains).

0. $\models \mathsf{wTD}(\emptyset)$ *and* $\models \mathsf{sTD}(\emptyset)$
1. *Separating a trust domain:*
   $\models \mathsf{wTD}(\mathcal{C} \cup \mathcal{C}') \to (\mathsf{wTD}(\mathcal{C}) \wedge \mathsf{wTD}(\mathcal{C}'))$
   $\models \mathsf{sTD}(\mathcal{C} \cup \mathcal{C}') \to (\mathsf{sTD}(\mathcal{C}) \wedge \mathsf{sTD}(\mathcal{C}'))$
2. $\models (\mathsf{wTD}(\mathcal{C}) \wedge \mathsf{wTD}(\mathcal{C}')) \to \mathsf{wTD}(\mathcal{C} \cap \mathcal{C}')$
   $\models (\mathsf{sTD}(\mathcal{C}) \wedge \mathsf{sTD}(\mathcal{C}')) \to \mathsf{sTD}(\mathcal{C} \cap \mathcal{C}')$
3. *if* $\mathcal{C} \subseteq \mathcal{C}'$ *then* $\models \mathsf{wTD}(\mathcal{C}') \to \mathsf{wTD}(\mathcal{C})$ *and* $\models \mathsf{sTD}(\mathcal{C}') \to \mathsf{sTD}(\mathcal{C})$.

Proof. Straightforward from definitions.

Now consider a more complex fact about (actually an insight into) trust domains.

Theorem 1 (Merging strong trust domains). *Merging two strong trust domains is compositional in the sense that a necessary and sufficient condition for merging two strong trust domains is that it be common knowledge in the union of both domains that each domain is a strong trust domain. Formally, for all* $\mathcal{C}, \mathcal{C}' \subseteq \mathcal{A}$,

$$\models \mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}(\mathsf{sTD}(\mathcal{C}) \wedge \mathsf{sTD}(\mathcal{C}')) \leftrightarrow \mathsf{sTD}(\mathcal{C} \cup \mathcal{C}').$$

Proof. See Appendix A.

Note that there is no analogous result for weak trust domains due to Fact 4. That is, merging *weak* trust domains is *non*-compositional in the sense that the result of merging two weak trust domains is not necessarily again a weak trust domain — not even when there is common knowledge (instead of only common belief) in the union of both domains that each domain is a weak trust domain (cf. Fact 4.2). The theorem yields a simple, though computationally intrinsically costly *design pattern* for recursively building up strong trust domains (cf. Appendix B). Again, due to the absence of an analogous theorem for weak trust domains, there is no analogous design pattern either. Hence, there really is a strong practical interest in strong trust domains. As a matter of fact, this practical interest is even stronger because checking membership in strong trust domains is computationally no more complex (up to a constant) than checking membership in weak trust domains (cf. Section 3).

We continue to define *potential* trust between two agents a, called *potential truster*, and b, called *potential trustee*, and within communities $\mathcal{C}$. The idea is to define *potentiality* as satis*fiability*.

Definition 3 (Potential trust).

- *There is a potential weak (strong) trust relationship between a and b in the system S :iff a $\mathsf{wTrusts}$ b (a $\mathsf{sTrusts}$ b) is satisfiable in the model $(\mathfrak{S}, \mathcal{V})$ induced by S.*
- *The community $\mathcal{C}$ is a potential weak (strong) trust domain in the system S :iff $\mathsf{wTD}(\mathcal{C})$ ($\mathsf{sTD}(\mathcal{C})$) is satisfiable in the model $(\mathfrak{S}, \mathcal{V})$ induced by S.*

Similarly, we define *actual* trust between two agents a, called *truster*, and b, called *trustee*, and within communities $\mathcal{C}$. The idea is to define (two degrees of) *actuality* as (two degrees of) satis*faction*.

Definition 4 (Actual trust).

- *There is a weak (strong) trust relationship between a and b in the model $(\mathfrak{S}, \mathcal{V})$ at the state $s \in \mathcal{S}$:iff a $\mathsf{wTrusts}$ b (a $\mathsf{sTrusts}$ b) is satisfied in $(\mathfrak{S}, \mathcal{V})$ at s.*
- *There is a weak (strong) trust relationship between a and b in the model $(\mathfrak{S}, \mathcal{V})$:iff a $\mathsf{wTrusts}$ b (a $\mathsf{sTrusts}$ b) is globally satisfied in $(\mathfrak{S}, \mathcal{V})$.*
- *The community $\mathcal{C}$ is a weak (strong) trust domain in the model $(\mathfrak{S}, \mathcal{V})$ at the state $s \in \mathcal{S}$:iff $\mathsf{wTD}(\mathcal{C})$ ($\mathsf{sTD}(\mathcal{C})$) is satisfied in $(\mathfrak{S}, \mathcal{V})$ at s.*
- *The community $\mathcal{C}$ is a weak (strong) trust domain in the model $(\mathfrak{S}, \mathcal{V})$:iff $\mathsf{wTD}(\mathcal{C})$ ($\mathsf{sTD}(\mathcal{C})$) is globally satisfied in $(\mathfrak{S}, \mathcal{V})$.*

Since satisfaction implies satisfiability, but not vice versa, actual trust implies potential trust, but not vice versa. For example, if two agents do not even know each other then they can not be in an actual trust relationship. However, they may be in a potential trust relationship: maybe in another system state, their trust potential can become actualised. On the other hand, in a given system two agents may well know each other but not be in a potential trust relationship: the system may be designed so that trust between them is impossible — in any system state.

Table 3. Computational time complexities

		Trust relations		Trust domains	
	degree	*weak* a wTrusts b	*strong* a sTrusts b	*weak* wTD$(\mathcal{C})$	*strong* sTD$(\mathcal{C})$
actual	local satisfaction in models $(\mathfrak{S}, \mathcal{V})$ and states s	$\mathcal{O}(f_S(\lvert s \rvert))$		$\mathcal{O}(2^{\lvert\mathcal{C}\rvert \cdot f_S(\lvert s \rvert)})$	
	global satisfaction in models $(\mathfrak{S}, \mathcal{V})$				
potential	satisfiability in models $(\mathfrak{S}, \mathcal{V})$				

Recall that there is a universal quantification over states s in the definition of global satisfaction, and an existential quantification over states s in the definitions of satisfiability in models.

3 Complexity results

We obtain our complexity results for deciding trust relationships and membership in trust domains by reduction to known results for the complexity of common belief and knowledge (cf. [17] and [16]). As usual for logics, the valuation function (here $\mathcal{V}$) acts as an oracle, which is assumed to decide in a single step whether or not an atomic proposition is true at the current state in the model induced by the considered distributed system S. However, deciding agent correctness is not trivial and depends on S. For example, in our applications (cf. Appendix C), deciding agent correctness can be at least polynomial in the size of the current state, which depending on the system modelling, may contain the history of system events. Another example is the case study in [24], where deterministically deciding agent correctness is quadratic in the size of the current state (containing the history of system events). The notion of state size really is system-specific. For example, when states contain the history of system events, state size can be defined as history length. Hence, we may have to account for the complexity of deciding agent correctness in the complexity of deciding trust relationships and membership in trust domains.

So, suppose that the truth of each atomic proposition can be deterministically decided in a polynomial number $f_S(|s|)$ of steps in the size $|s|$ of the state s of the model $(\mathfrak{S}, \mathcal{V})$ induced by S. We recall that since potential trust is defined in terms of satisfiability, and actual trust in terms of satisfaction, and since decidability of satisfiability implies decidability of satisfaction, decidability of potential trust implies decidability of actual trust, and complexities of potential trust are upper bounds for complexities of actual trust. Furthermore, satisfiability and validity are inter-solvable (ϕ is valid iff $\neg\phi$ is not satisfiable), and satisfiability complexities yield satisfaction (model-checking) complexities.

Theorem 2. *The computational time complexities of deterministically deciding trust relations is $\mathcal{O}(f_S(|s|))$ for potential and actual, weak and strong trust (cf. Table 3).*

Proof. Notice that our definitions of trust relations refer to a finite number (i.e., $|\mathcal{P}|$) of atomic propositions P, and that each definition uses exactly one atomic proposition (e.g., $\mathsf{correct}(b)$) and exactly one modal operator (e.g., B_a or K_a). Now according to [17], the complexity of the satisfiability of formulae $\mathsf{B}_a(\phi)$ and $\mathsf{K}_a(\phi)$ in a language with a finite number of atomic propositions and a bounded nesting depth of modal operators B_a and K_a is in (oracle) linear time in the length (here constantly 1) of the formula. Hence, the complexity of the satisfiability of formulae expressing weak and strong trust relations is even in (oracle) constant time, and thus $\mathcal{O}(f_S(|s|))$ without oracle. Yet $\mathcal{O}(f_S(|s|))$ is an absolute lower bound and thus the complexity of *all* trust relationships.

We can learn at least two lessons from these results. The first lesson is that we do have to account for the complexity of deciding agent correctness in the complexity of deciding trust relationships. The second lesson is that, surprisingly, deciding agent correctness is, up to a constant, equionerous to deciding potential and actual as well as weak and strong trust relationships.

Theorem 3. *The computational time complexities of deterministically deciding trust domains is* $\mathcal{O}(2^{|\mathcal{C}|\cdot f_S(|s|)})$ *for potential and actual, weak and strong trust (cf. Table 3).*

Proof. According to [16], the complexity of the satisfiability of formulae $\mathsf{CB}_{\mathcal{C}}(\phi)$ and $\mathsf{CK}_{\mathcal{C}}(\phi)$ is in (oracle) deterministic single exponential time in the length of the sub-formula ϕ. The intuition is that formulae $\mathsf{CB}_{\mathcal{C}}(\phi)$ and $\mathsf{CK}_{\mathcal{C}}(\phi)$ correspond to formulae of infinitely deeply nested operators B_a and K_a with $a \in \mathcal{C}$, respectively, and that in that case, a finite number of atomic propositions does not help. In our case, the length of the conjunctive sub-formula in $\mathsf{wTD}(\mathcal{C})$ and $\mathsf{sTD}(\mathcal{C})$ is polynomial in the size $|\mathcal{C}|$ of the community $\mathcal{C}$. Further, the complexity of each conjunct is $\mathcal{O}(f_S(|s|))$, which we assumed to be polynomial. Yet a single exponential of a polynomial cost is still "only" a single exponential cost. Hence, the complexity of the satisfiability of formulae $\mathsf{wTD}(\mathcal{C})$ and $\mathsf{sTD}(\mathcal{C})$ is deterministic single exponential time in the size of $\mathcal{C}$ times $f_S(|s|)$. That is, the complexity of membership in potential weak and strong trust domains is $\mathcal{O}(2^{|\mathcal{C}|\cdot f_S(|s|)})$. Finally according to [16], the operators $\mathsf{CB}_{\mathcal{C}}$ and $\mathsf{CK}_{\mathcal{C}}$ force satisfying models of a size that is exponential in $|\mathcal{C}|$. Hence, $\mathcal{O}(2^{|\mathcal{C}|\cdot f_S(|s|)})$ is the complexity of *all*, potential or actual, memberships problems.

4 Related work

There is a huge literature on notions of trust that are not formal in the sense of formal languages and semantics, and also on trust management, which however is not the subject matter of this paper. We are aware of the following formal work related to ours.

4.1 Trust relations

As explained in the introduction, [11] presents various kinds of trust relations based on belief in and knowledge of mental attitudes, which is less general than belief in or knowledge of agent behaviour.

In [39], trust relationships are defined as four-tuples of a set R of trusters, a set E of trustees, a set C of conditions, and a set P of properties (the actions or attributes of the trustees). Thereby, conditions and properties are fully abstract, i.e., without pre-determined form. According to the authors, "Trust relationship T means that under the condition set C, truster set R trust that trustee set E have the properties in set P.", where the meaning of "trust that" is formally primitive and thus left to interpretation. Given that trust can possibly be wrong, a plausible interpretation of "trust that" could be "believe that" (rather than "know that"). The authors then define operations on trust relationships in terms of set-theoretic operations on the projections of their tuples. We attempt to relate the authors' notion of trust relationship to our notion of weak trust relation (based on belief rather than knowledge) by coercing their definition of T in the following macro-definition of our logic, assuming their P is included in our $\mathcal{P}$:

$$T(C, R, E, P) := (\bigwedge_{c \in C} c) \to \bigwedge_{r \in R} \mathsf{B}_r(\bigwedge_{e \in E} \bigwedge_{p \in P} p(e)).$$

If the authors agree with this coercion, we have the following definability result:

$$\models T(\{\top\}, \{a\}, \{b\}, \{\mathsf{correct}\}) \leftrightarrow a \ \mathsf{wTrusts} \ b,$$

where $\mathsf{correct}$ is an attribute in the authors' sense.

In [18], so-called *trust in belief* and *trust in performance* are formalised in the situation calculus, such that "axioms and theorems can be used to infer whether a thing can be believed". Whereas we define (weak) trust *in agents*, and moreover *in terms of belief* taken off-the-shelf as a standard primitive. In [19], the ideas of trust in belief and trust in performance are taken up again similarly.

In [36], "trust is a state at which the host believes, expects, or accepts that the effects from the cleint are the positive", although belief is not formal in the sense of modal logic. The authors' idea is that "the host's trust on a client is obtained based on the trust evaluation of the client by the host. When the host trusts a client, the host will believe, expect or accept that the client will do no harm to the host in the given context". Hence, this notion of trust is agent-centric in the sense of being defined in terms of (local) effects at an agent's location. This is a less general notion of trust than ours, which is *systemic* in the sense of being defined in terms of correct agent behaviour within a certain system. Also, we recall again that agent correctness is a standard primitive in the distributed-systems community [26].

In [9], a domain-theoretic model of trust relations between agents is presented. In that model, a given directed trust relation from a truster to a trustee is abstracted as a value reflecting the truster's degree of trust in the trustee. Thus, [9]'s notion of trust is, as opposed to ours, quantitative, but, as opposed to ours, lacks a behavioural (e.g., in terms of agent correctness) and doxastic/epistemic explication (in terms of belief/knowledge). The purpose of the model is the definition of a unique global trust state arising from the trust relations between the agents via trust policies. Complexities for computing portions of that global state are given in [25].

4.2 Trust domains

To the best of our knowledge, the only formal piece of work on trust domains is [38], based on description logic. However, the authors' definition is a conceptual modelling of trust domains limited to PKIs.

5 Conclusion

Assessment We have provided simple, smooth definitions and complexity results for multi-agent trust in a single, standard framework. More precisely, we have delivered definitions for weak and strong trust relations and trust domains, as well as potential and actual trust relationships and membership in trust domains. All our definitions have the advantage of being declarative *and* computational, as well as being parametric in the notion of agent correctness. Thanks to being declarative, our definitions are independent of the manifold manifestations of trust establishment (e.g., via recommendations and/or reputation). They are meaningful in any concrete distributed system with a notion of agent correctness and state space. We recall that agent correctness is a primitive in the distributed-systems community [26], and that state space is forced in a world of digital computers. A surprising insight gained from our computational analysis of trust is that given weak trust, strong trust is for free (up to a constant) from the point of view of complexity theory. Finally, we have shown that our trust domains are fit as such for TTPs and the Web of Trust, and that with a minor modification in the form of a constraint, they can be made to fit PKIs, ID-Based Cryptography, and others.

Future work We could unify our notions of weak and strong trust relation (trust domain) in a notion of *graded trust relation* (*graded trust domain*) defined in terms of graded (common) belief instead of plain (common) belief and plain (common) knowledge, respectively [23]. Informally, knowledge is belief with 100% certitude. With the additional introduction of *temporal* modalities, it then becomes possible to study the evolution of the quality and quantity of trust in a given distributed system, by observing the evolution of the grade of each trust relation and trust domain in the system. Finally, we could build actual trust-management systems for trust relations and trust domains in our present sense of building trust *from absence of trust* and in a future sense of *re*building trust from *dis*trust.

Acknowledgements The first author thanks Jean-Luc Beuchat, Akira Kanaoka, Kanta Matsuura and the members of his laboratory, and Andrey Rybalchenko for stimulating discussions.

References

1. Cloud computing. `http://csrc.nist.gov/groups/SNS/cloud-computing/index.html`.

2. R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, second edition, 2008.
3. R. Anderson, B. Crispo, J.-H. Lee, Ch. Manifavas, V. Matyas, and F. Petitcolas. *The Global Internet Trust Register*. The MIT Press, 1999.
4. C. Areces and B. ten Cate. *Handbook of Modal Logic*, chapter Hybrid Logics. Volume 3 of Blackburn et al. [6], 2007.
5. P. Blackburn and J. van Benthem. *Handbook of Modal Logic*, chapter Modal Logic: A Semantic Perspective. Volume 3 of Blackburn et al. [6], 2007.
6. P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier, 2007.
7. C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.
8. J. Bradfield and C. Stirling. *Handbook of Modal Logic*, chapter Modal Mu-Calculi. Volume 3 of Blackburn et al. [6], 2007.
9. M. Carbone, M. Nielsen, and V. Sassone. A formal model for trust in dynamic networks. In *Proceedings of the IEEE Conference on Software Engineering and Formal Methods*, 2003.
10. B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990 (2002).
11. R. Demolombe. Reasoning about trust: A formal logical framework. In *Proceedings of the Conference on Trust Management*, volume 2995 of *LNCS*. Springer, 2004.
12. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(12), 1983.
13. R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.
14. E. Gallery and Ch.J. Mitchell. Trusted computing: Security and applications. *Cryptologia*, 33(3), 2009.
15. J.Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3), 1990.
16. J.Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3), 1992.
17. J.Y. Halpern and Y. Moses. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75(2), 1995.
18. J. Huang and M.S. Fox. An ontology of trust — formal semantics and transitivity. In *Proceedings of the ACM Conference on Electronic Commerce*, 2006.
19. J. Huang and D. Nicol. A calculus of trust and its application to PKI and identity management. In *Proceedings of the ACM Symposium on Identity and Trust on the Internet*, 2009.
20. M. Joye and G. Neven. *Identity-Based Cryptography*. IOS Press, 2009.
21. L.M. Kaufman. Data security in the world of cloud computing. *IEEE Security & Privacy*, 7(4), 2009.
22. S. Kramer, R. Goré, and E. Okamoto. Formal definitions and complexity results for trust relations and trust domains fit for TTPs, the Web of Trust, PKIs, and ID-Based Cryptography. Logic Column of ACM SIGACT News, March 2010.
23. S. Kramer, C. Palamidessi, R. Segala, A. Turrini, and Ch. Braun. A quantitative doxastic logic for probabilistic processes and applications to information-hiding. *Journal of Applied Non-Classical Logic*, 19(4), 2009.
24. S. Kramer and A. Rybalchenko. A multi-modal framework for achieving accountability in multi-agent systems. In *Proceedings of the ESSLLI-affiliated Workshop on Logics in Security*, 2010.

25. K. Krukowa and A. Twigg. The complexity of fixed point models of trust in distributed networks. *Theoretical Computer Science*, 389(3), 2007.
26. N.A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, 1996.
27. J.-J. Meyer and F. Veltnam. *Handbook of Modal Logic*, chapter Intelligent Agents and Common Sense Reasoning. Volume 3 of Blackburn et al. [6], 2007.
28. B. Michael. In clouds shall we trust? *IEEE Security & Privacy*, 7(5), 2009.
29. L.C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1), 1998.
30. S. Ruohomaa and L. Kutvonen. Trust management survey. In *Proceedings of the Conference on Trust Management*, volume 3477 of *LNCS*. Springer, 2005.
31. A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2), 2007.
32. B. Schneier. Security, group size, and the human brain. *IEEE Security & Privacy*, 7(4), 2009.
33. A. Tiu and R. Goré. A proof theoretic analysis of intruder theories. volume 5595 of *Lecture Notes in Computer Science*. Springer, 2009.
34. H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2007.
35. H.C.A. van Tilborg, editor. *Encyclopedia of Cryptography and Security*, pages 398–400. Springer, 2005.
36. D. Xiu and Z. Liu. A formal definition for trust in distributed systems. In *Proceedings of the Information Security Conference*, volume 3650 of *LNCS*. Springer, 2005.
37. A. Yao. Protocols for secure computations. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 1982.
38. H. Yu, Ch. Jin, and H. Che. A description logic for PKI trust domain modeling. In *Proceedings of the IEEE Conference on Information Technology and Applications*, 2005.
39. W. Zhao, V. Varadharajan, and G. Bryan. Analysis and modelling of trust in distributed information systems. In *Proceedings of the Conference on Information Systems Security*, volume 3803 of *LNCS*. Springer, 2005.
40. Ph. Zimmermann. `http://www.philzimmermann.com/`.

A A formal proof

We recall the laws $\models (\mathsf{CK}_{\mathcal{C}}(\phi) \land (\phi \to \phi')) \to \mathsf{CK}_{\mathcal{C}}(\phi')$, $\models (\mathsf{CK}_{\mathcal{C}}(\phi) \land \mathsf{CK}_{\mathcal{C}}(\phi')) \leftrightarrow \mathsf{CK}_{\mathcal{C}}(\phi \land \phi')$, and $\models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\phi) \to (\mathsf{CK}_{\mathcal{C}}(\phi) \land \mathsf{CK}_{\mathcal{C}'}(\phi))$; and call them $\mathbf{L1}(\mathsf{CK}_{\mathcal{C}})$, $\mathbf{L2}(\mathsf{CK}_{\mathcal{C}})$, and $\mathbf{L3}(\mathsf{CK}_{\mathcal{C}})$, respectively. Now, let $\mathfrak{S}$ designate an arbitrary frame for our logic, and $\mathcal{V}$ an arbitrary valuation function for agent correctness. Then:

1	$s \in \mathcal{S}$	hyp.
2	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{sTD}(\mathcal{C}) \land \mathsf{sTD}(\mathcal{C}'))$	hyp.
3	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{sTD}(\mathcal{C})) \land \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{sTD}(\mathcal{C}'))$	2, $\mathbf{L2}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$
4	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}}(\mathsf{sTD}(\mathcal{C}))$ and $(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{sTD}(\mathcal{C}'))$	3, def.
5	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{sTD}(\mathcal{C}) \to \bigwedge_{a,b\in\mathcal{C}} a \ \mathsf{sTrusts}\ b$	$\mathbf{T}(\mathsf{CK}_{\mathcal{C}})$
6	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{sTD}(\mathcal{C}') \to \bigwedge_{a,b\in\mathcal{C}'} a \ \mathsf{sTrusts}\ b$	$\mathbf{T}(\mathsf{CK}_{\mathcal{C}'})$
7	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}} a \ \mathsf{sTrusts}\ b)$	4, 5 $\mathbf{L1}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$
8	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'} a \ \mathsf{sTrusts}\ b)$	4, 6 $\mathbf{L1}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$
9	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}} a \ \mathsf{sTrusts}\ b)$ and $(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'} a \ \mathsf{sTrusts}\ b)$	7, 8
10	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}} a \ \mathsf{sTrusts}\ b) \land \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'} a \ \mathsf{sTrusts}\ b)$	9, def.
11	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C}} a \ \mathsf{sTrusts}\ b \land \\ \bigwedge_{a,b\in\mathcal{C}'} a \ \mathsf{sTrusts}\ b\end{pmatrix}$	10, $\mathbf{L2}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$
12	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}\cup\mathcal{C}'} a \ \mathsf{sTrusts}\ b)$	11, Lemma 1.2
13	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{sTD}(\mathcal{C} \cup \mathcal{C}')$	12, def.
14	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{sTD}(\mathcal{C} \cup \mathcal{C}')$	hyp.
15	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}\cup\mathcal{C}'} a \ \mathsf{sTrusts}\ b)$	14, def.
16	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C}} a \ \mathsf{sTrusts}\ b \land \\ \bigwedge_{a,b\in\mathcal{C}'} a \ \mathsf{sTrusts}\ b\end{pmatrix}$	15
17	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C}} a \ \mathsf{sTrusts}\ b \land \\ \bigwedge_{a,b\in\mathcal{C}'} a \ \mathsf{sTrusts}\ b\end{pmatrix})$	16, $\mathbf{4}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$
18	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}} a \ \mathsf{sTrusts}\ b) \land \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'} a \ \mathsf{sTrusts}\ b)$	16, $\mathbf{L2}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$
19	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}}(\bigwedge_{a,b\in\mathcal{C}} a \ \mathsf{sTrusts}\ b) \land \mathsf{CK}_{\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'} a \ \mathsf{sTrusts}\ b)$	18, $\mathbf{L3}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$
20	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{sTD}(\mathcal{C}) \land \mathsf{sTD}(\mathcal{C}')$	19, def.
21	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C}} a \ \mathsf{sTrusts}\ b \land \\ \bigwedge_{a,b\in\mathcal{C}'} a \ \mathsf{sTrusts}\ b\end{pmatrix} \to (\mathsf{sTD}(\mathcal{C}) \land \mathsf{sTD}(\mathcal{C}'))$	20
22	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{sTD}(\mathcal{C}) \land \mathsf{sTD}(\mathcal{C}'))$	17, 21, $\mathbf{L1}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$
23	$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{sTD}(\mathcal{C} \cup \mathcal{C}')$ if and only if $(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{sTD}(\mathcal{C}) \land \mathsf{sTD}(\mathcal{C}'))$	2–13, 14–22

24 for all $s \in \mathcal{S}$, $(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{sTD}(\mathcal{C}) \land \mathsf{sTD}(\mathcal{C}')) \leftrightarrow \mathsf{sTD}(\mathcal{C} \cup \mathcal{C}')$ 1–23

25 $(\mathfrak{S}, \mathcal{V}) \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{sTD}(\mathcal{C}) \land \mathsf{sTD}(\mathcal{C}')) \leftrightarrow \mathsf{sTD}(\mathcal{C} \cup \mathcal{C}')$ 24

B Building trust

Building trust in the sense of building *from absence of trust* (as opposed to *re*building from *dis*trust, cf. 2nd paragraph after Definition 1) is possible if and only if there is at least *potential* trust in the sense of Definition 3. That is, given $a, b \in \mathcal{A}$ and $\mathcal{C} \subseteq \mathcal{A}$, the formulae a $\mathsf{wTrusts}$ b or a $\mathsf{sTrusts}$ b, and $\mathsf{wTD}(\mathcal{C})$ or $\mathsf{sTD}(\mathcal{C})$ must at least be *satisfiable in the model induced by the considered system*, in order for a weak or strong trust relationship from a to b to possibly exist, and in order for $\mathcal{C}$ to possibly be a weak or strong trust domain, respectively. Yet thanks to the computability of our notions of trust, a computer can aid us in our decision of whether or not building trust from a current absence of trust in a given system, and between a given pair of agents, or within a given (small) group of agents is actually possible.

When it is indeed possible to actualise a certain potential trust, the next question is how to actually *build up* the trust. A potential trustee has at least two non-mutally-exclusive possibilities of earning the trust of a potential truster:

1. by *behaving correctly* according to the notion of agent correctness (cf. $\mathsf{correct}$) of the system where the computer has found the considered kind of trust to be actually possible. (Behaving correctly can include doing nothing, when the considered notion of agent correctness does not exclude such inactivity.)
2. by *producing evidence* (e.g., a recommendation) for *or proof* (e.g., a signed log file) of behavioural correctness, whenever requested to do so by the potential truster (here in the role of an auditor).

The potential truster may then decide to trust the potential trustee based on the observation of the potential trustee's behaviour, and/or based on the knowledge of certain data, i.e., evidence or even proof produced by the potential trustee. Depending on the value of the information obtained, a relationship of weak or strong trust is established from the potential truster to the potential trustee, who have now become *de facto* an actual truster and trustee, respectively. This process of building up oriented trust relations can be extended to building up symmetric trust relationships, and trust domains of agents in such relationships. As a matter of fact, Theorem 1 yields the following *design pattern*, called RD, for building up via recursive descent strong trust domains from a possible absence of trust in a given system. RD relies on the communication abstraction of *public announcement* (cf. 3rd paragraph after Definition 1), which we use as a black-box plug-in. We recall that the effect of a public announcement of a fact is to induce the common knowledge of that fact with the addressed public by minimally changing the epistemic state of each agent in that public (cf. [34] for details). Depending on the communication context, the mechanism ranges from trivial (e.g., in a public assembly) to difficult (e.g., with communication asynchrony), possibly requiring implementation as a full-fledged communication protocol. So, although ‘announcement’ may suggest triviality rather than difficulty, public announcements may well be non-trivial to implement in a given context, which is why we call RD *design pattern* rather than *algorithm*:

```
1. Input: a model $(\mathfrak{S}, \mathcal{V})$, a state $s$, and $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{A}$;
2. Divide: for $i \in \{1, 2\}$ do {
   when $(\mathfrak{S}, \mathcal{V}), s \models \neg\mathsf{sTD}(\mathcal{C}_i)$:
   (a) divide $\mathcal{C}_i$ freely into $\mathcal{C}_{i.1}$ and $\mathcal{C}_{i.2}$;
   (b) $s := \mathrm{RD}((\mathfrak{S}, \mathcal{V}), s, \mathcal{C}_{i.1}, \mathcal{C}_{i.2})$; };
3. Conquer: announce to the community $\mathcal{C}_1 \cup \mathcal{C}_2$ that $\mathsf{sTD}(\mathcal{C}_1) \wedge \mathsf{sTD}(\mathcal{C}_2)$ is true
   (choose appropriate communication channels and an appropriate protocol),
   which takes the system from $s$ to some $s' \in \mathcal{S}$ such that $s'$ is reachable (cf.
   Footnote 7) from $s$ and

       $(\mathfrak{S}, \mathcal{V}), s' \models \mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}(\mathsf{sTD}(\mathcal{C}) \wedge \mathsf{sTD}(\mathcal{C}'))$;

4. Output: $s' \in \mathcal{S}$;
5. Effect: $(\mathfrak{S}, \mathcal{V}), s' \models \mathsf{sTD}(\mathcal{C}_1 \cup \mathcal{C}_2)$.
```

Building up trust domains from possible absence of trust generally is computationally costly, in particular when made by means of the recursive-descent design pattern RD.

Theorem 4. *The complexity of building up trust domains is exponential in the number of potential members.*

Proof. By the fact that membership in trust domains has to be checked for each potential new member anew, which is exponential in the size of the trust domain being checked in a given model and at a given state (cf. Table 3).

In contrast, it is common knowledge (among humans) that for *destroying* actual trust relationships, and thus also trust domains, a single (side) step is sufficient — metaphorically speaking. And *re*building trust from *dis*trust is difficult.

C Application to cryptographic-key management

We instantiate our concepts of trust relations and trust domains in four major applications of trust, namely Trusted Third Parties (TTPs), the Web of Trust, Public-Key Infrastructures (PKIs), and ID-Based Cryptography. For the latter three, we will have to define the valuation function $\mathcal{V}$ on the atomic propositions $\mathsf{correct}(a)$ about agent correctness (cf. Definition 1 and Table 2). (We will also have to refine the definition of trust domains in Table 2 for the latter two.) That is, ***each notion of agent correctness is specific***[10] ***to each system rather than general to all systems.*** (Thus, ***trust is system-specific to some extent.***) However, we can define agent correctness *generically* for the Web of Trust and PKIs, by means of the following, common auxiliary logic,

[10] like policies, which induce notions of agent correctness, as mentioned in Section 1.1

called AuxLog. The logic is a modal fixpoint logic [8] operating on points that are agents $a \in \mathcal{A}$ rather than states $s \in \mathcal{S}$. AuxLog is *parametric* in a binary relation $R \subseteq \mathcal{A} \times \mathcal{A}$ to be fixed separately for the Web of Trust and PKIs, but with the commonality of depending on a fixed state s ($R \in \{\mathrm{DTI}_s, \mathrm{CERT}_s\}$).

Definition 5 (Auxiliary Logic). *Let $\mathcal{X}$ designate a countable set of propositional variables C, and let*

$$\mathcal{L}' \ni \alpha ::= \mathsf{OK} \mid C \mid \neg\alpha \mid \alpha \wedge \alpha \mid \overline{\Box}\alpha \mid \boldsymbol{\nu} C(\alpha)$$

designate the language $\mathcal{L}'$ of AuxLog where all free occurrences of C in α of $\boldsymbol{\nu} C(\alpha)$ are assumed to occur within an even number of occurrences of $\neg$ to guarantee the existence of (greatest) fixpoints (expressed by $\boldsymbol{\nu} C(\alpha)$) [8]. Then, given a relation $R \subseteq \mathcal{A} \times \mathcal{A}$ decidable in deterministic constant time but structurally arbitrary, and an auxiliary interpretation $[\![\cdot]\!] : \mathcal{X} \cup \{\mathsf{OK}\} \to 2^{\mathcal{A}}$ partially pre-defined as

$$[\![\mathsf{OK}]\!] := \{\ a \in \mathcal{A} \mid \textit{at most } a \textit{ can access } a\textit{'s private key}\ \}^{11},$$

the interpretation $\|\cdot\|_{[\![\cdot]\!]} : \mathcal{L}' \to 2^{\mathcal{A}}$ of AuxLog-propositions is as follows:

$$\begin{aligned}
\|C\|_{[\![\cdot]\!]} &:= [\![C]\!] \\
\|\mathsf{OK}\|_{[\![\cdot]\!]} &:= [\![\mathsf{OK}]\!] \\
\|\neg\alpha\|_{[\![\cdot]\!]} &:= \mathcal{A} \setminus \|\alpha\|_{[\![\cdot]\!]} \\
\|\alpha \wedge \alpha'\|_{[\![\cdot]\!]} &:= \|\alpha\|_{[\![\cdot]\!]} \cap \|\alpha'\|_{[\![\cdot]\!]} \\
\|\overline{\Box}\alpha\|_{[\![\cdot]\!]} &:= \{\ a \in \mathcal{A} \mid \textit{for all } b \in \mathcal{A}, \textit{ if } b\ R\ a \textit{ then } b \in \|\alpha\|_{[\![\cdot]\!]}\ \} \\
\|\boldsymbol{\nu} C(\alpha)\|_{[\![\cdot]\!]} &:= \bigcup\{\ A \subseteq \mathcal{A} \mid A \subseteq \|\alpha\|_{[\![\cdot]\!]_{[C \mapsto A]}}\ \},
\end{aligned}$$

where $[\![\cdot]\!]_{[C \mapsto A]}$ maps C to A and otherwise agrees with $[\![\cdot]\!]$.

Further, $\alpha \vee \alpha' := \neg(\neg\alpha \wedge \neg\alpha')$, $\top := \alpha \vee \neg\alpha$, $\bot := \neg\top$, $\alpha \to \alpha' := \neg\alpha \vee \alpha'$, $\alpha \leftrightarrow \alpha' := (\alpha \to \alpha') \wedge (\alpha' \to \alpha)$, $\overline{\Diamond}\alpha := \neg\overline{\Box}(\neg\alpha)$, and, notably, $\boldsymbol{\mu} C(\alpha(C)) := \neg\boldsymbol{\nu} C(\neg\alpha(\neg C))$.

Finally, for all $a \in \mathcal{A}$ and $\alpha \in \mathcal{L}'$,

$$\langle(\mathcal{A}, R), [\![\cdot]\!]\rangle, a \vDash \alpha \quad \textit{:iff} \quad a \in \|\alpha\|_{[\![\cdot]\!]},$$

and for all $\alpha \in \mathcal{L}'$,

$$\vDash \alpha \quad \textit{:iff} \quad \textit{for all } [\![\cdot]\!] \textit{ and } a \in \mathcal{A},\ \langle(\mathcal{A}, R), [\![\cdot]\!]\rangle, a \vDash \alpha.$$

[11] This phrasing can be made formal provided that a notion of data space $\mathcal{D}$ (including keys) and data derivation for agents is fixed, e.g., à la Dolev-Yao [12]. Then data derivation can be formalised as a relation $\vdash\ \subseteq 2^{\mathcal{D}} \times \mathcal{D}$ (the first formalisation was in terms of closure operators [29]), and the phrase "at most a can access a's private key" as "for all $b \in \mathcal{A}$, if k is the private key of a and $\mathrm{m}_b(s) \vdash k$ then $b = a$", where $\mathrm{m}_b(s)$ returns the set of data that b generated or received as such in s. Note however that depending on the structure of $\mathcal{D}$, the computability of $\vdash$ may range from polynomial time to undecidability [33].

Table 4. Trustworthy Trusted Third Parties

$\mathsf{wtTTP}(c,a,b) := \mathsf{CB}_{\{a,b,c\}}(\mathsf{wTD}(\{c,a\}) \land \mathsf{wTD}(\{c,b\}))$	c is a weakly trustworthy TTP (wtTTP) of a and b
$\mathsf{stTTP}(c,a,b) := \mathsf{CK}_{\{a,b,c\}}(\mathsf{sTD}(\{c,a\}) \land \mathsf{sTD}(\{c,b\}))$	c is a strongly trustworthy TTP (stTTP) of a and b

The reader is invited not to confuse the auxiliary satisfaction relation $\vDash$ of AuxLog with $\models$, the main one from Definition 1. Further note that AuxLog is a member of the family of μ-calculi over the modal system **K**, which is characterised by the laws of propositional logic and the modal laws $\vDash \overline{\Box}(\alpha \to \alpha') \to (\overline{\Box}\alpha \to \overline{\Box}\alpha')$ and "if $\vDash \alpha$ then $\vDash \overline{\Box}\alpha$". The reason is that, as mentioned, $R \subseteq \mathcal{A} \times \mathcal{A}$ is structurally arbitrary. Hence, no more structural properties than those of the modal system **K**, i.e., none, can generally be assumed to hold for $\overline{\Box}$. As a corollary, the model-checking problem, i.e., "Given $a \in \mathcal{A}$ and $\alpha \in \mathcal{L}'$, is it the case that $\langle(\mathcal{A}, R), [\![\cdot]\!]\rangle, a \vDash \alpha$?" is decidable in deterministic polynomial time in the size of α. See [8] for details.

C.1 Trusted Third Parties

The concept of a Trusted Third Party (TTP) is a folklore concept for much of information security, e.g., for many protocols for authentication and key establishment [7], as well as for secure multiparty computation [37]. Recall that "a secure multiparty computation for function f can be viewed as an implementation of a trusted third party T, which, upon receipt of the input values $x_1, \ldots, x_n$ from parties $P_1, \ldots, P_n$, respectively, produces the output value $y = f(x_1, \ldots, x_n)$. Party T is trusted for (i) providing the correct value for y and (ii) [n]ot revealing any further information to parties $P_1, \ldots, P_n$" [35]. In our terminology of agent correctness, the conjunction of Condition (i) and (ii) informally stipulates what it means for T to be correct. Notice that the above definition of secure multiparty computation merely defines the *object* of trust (i.e., agent correctness), but not trust itself (which, in our definition, is belief or even knowledge *of* agent correctness). To the best of our knowledge, the concept of a TTP has never been formally defined, i.e., mathematically analysed into its conceptual constituents. Here, we are able to define the TTP-concept in terms of our concept of trust domains (cf. Table 4), and instantiate TTPs in the Web of Trust (cf. Section C.2) and Public-Key Infrastructures (PKIs) (cf. Section C.3). More precisely, we define the concepts of a weakly and a strongly *trustworthy* TTP, i.e., TTPs that may or even must *deserve* the trust of their trusters—and vice versa. Note that the trustworthiness of TTPs (e.g., the certification authorities in a PKI) is absolutely crucial, without which whole security architectures (e.g., a PKI) can break down. Observe that thanks to Theorem 1, the two sides (e.g., $\{c, a\}$ and $\{c, b\}$) in a strongly (but not in a weakly) trustworthy TTP constitute a (strong) trust domain as a whole (i.e., as $\{c, a\} \cup \{c, b\}$).

C.2 The Web of Trust

In the (decentralised) Web of Trust, as defined by Philip Zimmermann [40] in 1992, any agent can independently establish its own domain of trusted correspondents by publicly designating (e.g., on their homepage) so-called *trusted introducers*, who by this very act become commonly known as such. In PGP, the designation of a trusted introducer is implemented as the (publicly exportable) signing of the designated trusted introducer's public key with the designator's private key. Additional assurance can be provided by The Global Internet Trust Register [3]. The role of an agent a's trusted introducer b is to act as a *guarantor* for the trustworthiness of a, and by that, to catalyse the building up of trust relationships between a and those agents c who are only potential (not yet actual) trustees of a but who are (already) actual trustees of b. Notice the importance of distinguishing between potential and actual trust (cf. Definition 3 and 4). Thus, the more guarantors (actual trustees) an agent (as an actual truster) has, the more potential trustees the agent (as a potential truster) has. In the Web of Trust, agents are (socially speaking) trustworthy, or (technically speaking) ***correct if and only if all their designated trusted introducers are, and at most they (the correct agents) can access their (own) private key.*** (Agents with untrustworthy introducers or a corrupt private key are untrustworthy.) Notice the possible *mutuality* in this social notion of agent correctness.

We model the designated-trusted-introducer relationships between agents in system states $s \in \mathcal{S}$ with a family of relations (a kind of data base) $\mathrm{DTI}_s \subseteq \mathcal{A} \times \mathcal{A}$ such that

$$b \; \mathrm{DTI}_s \; a \text{ :iff } b \text{ is a designated trusted introducer of } a \text{ in } s.$$

The valuation function $\mathcal{V}$ on the propositions $\mathsf{correct}(a)$ can then be formally defined with the aid of AuxLog as follows:

$$\boxed{\mathcal{V}(\mathsf{correct}(a)) := \{\ s \mid \langle (\mathcal{A}, \mathrm{DTI}_s), \emptyset \rangle, a \vDash \boldsymbol{\nu} C(\mathsf{OK} \wedge \overline{\Box} C)\ \},}$$

where $\emptyset$ designates the empty auxiliary interpretation (C is bound!). The greatest-fixpoint assertion $\langle (\mathcal{A}, \mathrm{DTI}_s), \emptyset \rangle, a \vDash \boldsymbol{\nu} C(\mathsf{OK} \wedge \overline{\Box} C)$ says that a is in the greatest fixpoint of the interpretation of *the property* C *such that:*

> if a satisfies C (i.e., a is in the interpretation of C) then a satisfies $\mathsf{OK} \wedge \overline{\Box} C$, which in turn says that at most a can access a's private key and for all $b \in \mathcal{A}$, if b is a designated trusted introducer of a in the state s then b satisfies C.

Observe that all (=1) free occurrences of C in $\mathsf{OK} \wedge \overline{\Box} C$ of $\boldsymbol{\nu} C(\mathsf{OK} \wedge \overline{\Box} C)$ occur within an even (=0) number of occurrences of $\neg$. Hence, our definition is formally well-defined. Further observe that the possible mutuality in our notion of agent correctness corresponds to the co-inductiveness of the greatest fixpoint, which allows direct (*self-designation*) and indirect (*mutual designation*) loops in the designated-trusted-introducer relationships. The interpretation of the corresponding (inductive) least-fixpoint formula would wrongly not allow such loops.

Of course, the language of AuxLog allows for other, more complex definitions of agent correctness, e.g., ones disallowing self-designation[12], and/or ones with a more complex notion of being OK. Our present definition is just an inceptive example proposal. The *co*-inductive definition has the following iterative paraphrase from *above* (iterated *de*construction).

> Everybody is correct (the Web of Trust is born in the plenum, so to say); except for the following agents (*ex*clude those which are clearly not OK):
> 0. agents with a corrupt private key (Type 0 agents)
> 1. agents with a designated trusted introducer of Type 0 (Type 1 agents)
> 2. agents with a designated trusted introducer of Type 1 (Type 2 agents)
> 3. etc.

Clearly, weak or strong trust relations in the Web of Trust must be universal within an agent's domain of correspondents in the sense of Proposition 1: designated trusted introducers are trusted; and they would not act as such, if they did not trust their designator and their designator's other designated trusted introducers, etc. Hence, our trust relations and trust domains from Table 2 as well as our weakly and strongly trustworthy TTPs from Table 4 are fit for the Web of Trust without further adaptation.

C.3 Public-Key Infrastructures

In Public-Key Infrastructures (PKIs), centralised *certificate authorities* (CAs) act as guarantors for the trustworthiness of the public key of their clients by issuing certificates that bind the public key of each client (the legitimate key owner) to the client's (unique) name. In PKIs, agents are (socially speaking) trustworthy, or (technically speaking) ***correct if and only if all their certified agents are, and at most they (the correct agents) can access their (own) private key.*** (Agents who certify incorrect agents or agents with a corrupt private key are incorrect.) Notice the absence of mutuality in this notion of agent correctness; it is intrinsically unilateral. However, the notion of PKI trust to be built from this notion of agent correctness will be again bilateral (i.e., mutual, and thus symmetric): the certifying correct agent trusts the certified correct agent, and vice versa.

We model the relationships from certifying agents to certified agents (which may themselves be certifying agents to agents certified by them, etc.) in system states $s \in \mathcal{S}$ with a family of relations (a kind of data base) $\mathrm{CRT}_s \subseteq \mathcal{A} \times \mathcal{A}$ such that

$$b \; \mathrm{CRT}_s \; a \text{ :iff } b \text{ is certified by } a \text{ in } s,$$

[12] Yet whether or not you declare yourself as trustworthy may well be legally critical (cf. for example such hand-written self-declarations in certain immigration procedures).
Anyway, the disallowance of self-designation could be implemented by introducing a binding operator $\downarrow$ à la hybrid logic [4] into the language of AuxLog, such that $\langle(\mathcal{A}, R), [\![\cdot]\!]\rangle, a \vDash {\downarrow}C(\alpha)$:iff $\langle(\mathcal{A}, R), [\![\cdot]\!]_{[C \mapsto \{a\}]}\rangle, a \vDash \alpha$, and stipulating that $\mathcal{V}(\mathsf{correct}(a)) := \{\ s \mid \langle(\mathcal{A}, \mathrm{DTI}_s), \emptyset\rangle, a \vDash \boldsymbol{\mu}C(\mathsf{OK} \land \neg{\downarrow}C'(\overline{\Diamond}C') \land \overline{\Box}C)\ \}$.

where "b is certified by a in s" means "a has issued a valid certificate for b in s", i.e., a certificate that is non-revoked in s and signed by a with the private key of a. The valuation function $\mathcal{V}$ on the propositions $\mathsf{correct}(a)$ can then be formally defined with the aid of AuxLog as follows:

$$\boxed{\mathcal{V}(\mathsf{correct}(a)) := \{\ s \mid \langle(\mathcal{A}, \mathrm{CRT}_s), \emptyset\rangle, a \vDash \boldsymbol{\mu} C(\mathsf{OK} \land \overline{\Box} C)\ \}.}$$

The least-fixpoint assertion $\langle(\mathcal{A}, \mathrm{CRT}_s), \emptyset\rangle, a \vDash \boldsymbol{\mu} C(\mathsf{OK} \land \overline{\Box} C)$ says that a is in the least fixpoint of the interpretation of *the property C such that:*

> if a satisfies $\mathsf{OK} \land \overline{\Box} C$ (i.e., a is in the interpretation of $\mathsf{OK} \land \overline{\Box} C$)—which in turn says that at most a can access a's private key and for all $b \in \mathcal{A}$, if b is certified by a in the state s then b satisfies C—then a satisfies C.

Observe that certification is unilateral (i.e., non-mutual, and thus not symmetric) in the sense that certification relationships must not be directly (*self-certification*) nor indirectly (*mutual certification*) looping, which forces a least-fixpoint formulation. The PKI-approach to trustworthiness is thus diametrically opposed to the approach of the Web of Trust. This opposition is reflected first, in the least/greatest fixpoint "duality" of the two paradigms; and second, in the fact that PKIs are based on (ultimately national) *authority* (hierarchical CAs), whereas the Web of Trust is based on (borderless) *peership* (peer-guarantors). (At the international level, it makes sense to organise the totality of national CAs as a Web of Trust.) Yet again of course, as with the Web of Trust, the language of AuxLog allows for other, more complex definitions of agent correctness, e.g., ones with self-certification for the root-CA[13] (the *trust anchor*), and/or ones with a more complex notion of being OK (e.g., including the possibility of *key escrow*). Again, our present definition is just an inceptive example proposal. The *in*ductive definition has the following iterative paraphrase from *below* (iterated *con*struction).

> Nobody is correct (PKIs are born *ex nihilo*, so to say); except for the following agents (*in*clude those which are clearly OK): agents without a corrupt private key (Type 0 agents), whose certified agents are also of Type 0 (Type 1 agents), whose certified agents are again also of Type 0 (Type 2 agents), etc. (In other words, being of Type 0 is an invariant in the transitive closure of certification relationships.)

Notice the structural difference between this paraphrase for agent correctness in PKIs and the previous one for agent correctness in the Web of Trust. The "duality" is not pure.

As suggested, CAs are commonly organised in a hierarchy, which induces *structured trust domains* in the form of finite trees. Recall that a finite tree is a partially-ordered set (here say $\langle\mathcal{C}, \leq\rangle$) with a bottom (top) element such that

[13] Self-certification for the root-CA could be implemented by introducing an atomic proposition root true at and only at the root-CA agent, and stipulating that $\mathcal{V}(\mathsf{correct}(a)) := \{\ s \mid \langle(\mathcal{A}, \mathrm{CRT}_s), \emptyset\rangle, a \vDash (\mathsf{root} \to \overline{\Diamond}\mathsf{root}) \land \boldsymbol{\mu} C(\mathsf{OK} \land \overline{\Box} C)\ \}$.

Table 5. Public-Key Infrastructure trust domains

$\mathsf{wTD}_{\mathsf{PKI}}(\mathcal{C}) := \mathsf{CB}_{\mathcal{C}}(\bigwedge_{a,\,b\,\in\,\mathcal{C}\ \text{and}\ (a \leq b \text{ or } b \leq a)} a \mathrel{\mathsf{wTrusts}} b)$	$\mathcal{C}$ is a weak PKI trust domain
$\mathsf{sTD}_{\mathsf{PKI}}(\mathcal{C}) := \mathsf{CK}_{\mathcal{C}}(\bigwedge_{a,\,b\,\in\,\mathcal{C}\ \text{and}\ (a \leq b \text{ or } b \leq a)} a \mathrel{\mathsf{sTrusts}} b)$	$\mathcal{C}$ is a strong PKI trust domain.

for each element in the set, the down-set (up-set) of the element is a finite chain [10]. In PKI trust domains, trust relations are symmetric (up- and downwards the tree branches) and transitive (along the tree branches) but not universal (after all, a tree is a tree and not felt fabric), and the root CA corresponds to the tree root, the intermediate CA's correspond to the intermediate tree nodes, and the clients to the tree leafs. Hence we can fit our weak and strong trust domains to PKI trust domains $\langle\mathcal{C}, \leq\rangle$ by simply stipulating that the conjunction in the respective definition respect the finite-tree structure $\leq$ of $\mathcal{C}$, and reflect the symmetry and transitivity of the trust relations. And that is all: see Table 5. We can now instantiate our weakly and strongly trustworthy TTPs from Table 4 for PKIs as $\mathsf{wtTTP}_{\mathsf{PKI}}(c, a, b) := \mathsf{CB}_{\{a,b,c\}}(\mathsf{wTD}_{\mathsf{PKI}}(\{c, a\}) \wedge \mathsf{wTD}_{\mathsf{PKI}}(\{c, b\}))$ and $\mathsf{stTTP}_{\mathsf{PKI}}(c, a, b) := \mathsf{CK}_{\{a,b,c\}}(\mathsf{sTD}_{\mathsf{PKI}}(\{c, a\}) \wedge \mathsf{sTD}_{\mathsf{PKI}}(\{c, b\}))$, respectively, with $\leq := \{(c, a), (c, b)\}$ as (tree) domain structure. Fits for trust domains with other structures (e.g., buses, chains, rings, stars, etc.) can be made by similarly simple stipulations (e.g., for computing clouds, which have not a fixed but a dynamic, an evolving structure).

In sum, a weak or strong PKI trust domain $\mathcal{C}$ is built from a *certification hierarchy* $\leq$ of certifying (CAs) and certifiable agents $a \in \mathcal{C}$ such that for all states $s \in \mathcal{S}$ there is a (possibly empty) *certification record* $\{\, b \in \mathcal{C} \mid b \mathrel{\mathrm{CRT}_s} a \,\}$. Thereby, the certification hierarchy acts as a constraining skeleton (there is no such skeleton in the Web of Trust; it is unconstrained) for the potential and the actual trust relationships in $\mathcal{C}$, and, by that, the respective memberships in the trust domain $\mathcal{C}$ itself. And the certification records act as evidential support for the actuality of the trust relationships and the memberships in the trust domain.

C.4 Identity-Based Cryptography

Identity-Based Cryptography is a variation of Public-Key Cryptography in which the intending sender of a message derives the (public) encryption key from the public identity (e.g., a telephone number, an email address, etc., or a combination thereof) of the intended recipient [20]. In our setting, we abstractly model an agent a's public identity with the symbol 'a'. For the sake of the security of ID-based encryption, an ID-based private key must not be derivable from its corresponding public counterpart without an additional trap-door information. This trap-door information is owned by a central CA (cCA $\in \mathcal{A}$), which therefore can derive the private keys of all its certified agents. (Thus ID-based domains have a star structure: $\leq$ is an n-ary tree of depth 1 with as root cCA and as leaves the n agents certified by cCA.) Hence for ID-Based Cryptography, the definition of an agent being OK (cf. Section C) must be weakened, e.g.,

$$[\![\mathsf{OK}]\!] := \{\ a \in \mathcal{A} \mid \text{at most } a \text{ and cCA can access } a\text{'s private key}\ \}.$$

Note that as a consequence, the notion of trust (and thus the value of the trustworthiness of cCA) is weakened! Of course, a restrengthening is possible, e.g., by stipulating that cCA has not *used* the private keys of its certified agents (except possibly for *key escrow*). In sum, the flexibility of ID-Based Cryptography for its users (the certified agents) is paid with a devaluation of the trustworthiness of its provider (cCA).

A Multi-Modal Framework for Achieving Accountability in Multi-Agent Systems

Simon Kramer* and Andrey Rybalchenko

[1] University of Tsukuba, Japan
simon.kramer@a3.epfl.ch
[2] Technical University Munich, Germany
rybal@in.tum.de

Abstract. We present a multi-modal, model-theoretic framework for achieving accountability in multi-agent systems through formal proof. Our framework provides modalities for knowledge, provability, and time. With these modalities, we formalise the two main aspects of accountability, which are: *soundness (accountability proper),* i.e., for correct agents, the provability of their correctness by themselves; and *completeness (auditability),* i.e., for faulty agents, the eventual provability of their faultiness by others. In our framework, the accountability proof of a particular system is reduced to the proof of a few key lemmata, which the system designer needs to establish for a considered system.

Keywords
accountability (including auditability, liability, and non-repudiation); modal logics of knowledge, provability, and time; dependable distributed or multi-agent systems; Popper's critical rationalism; Russel's paradox.

1 Introduction

The subject matter of this paper is *accountability* in multi-agent or distributed systems [23], i.e., the possibility of enforcing responsibility for illegitimate or even illegal (in)action (in)effectuated by faulty agents in those systems. In plainer words, accountability allows to *place blame* [26] with all faulty agents (completeness aspect), and only with those agents (soundness aspect). Note that when faultiness implies illegality, accountability implies liability. In the present section, we introduce the motivation for our subject matter, the goal to be achieved in the matter, and the methodology that we employ to meet our goal.

Motivation In [28], accountability is promoted as a first-class design principle for dependable distributed systems. According to these authors' manifesto,

* This (corresponding) author's contribution was initiated in the Comète group at Ecole Polytechnique and INRIA (France), and completed under Grant P 08742 from the Japan Society for the Promotion of Science in the Laboratory of Cryptography and Information Security (U Tsukuba).

"conventional techniques for dependable systems design are insufficient to defend against an adversary that manipulates the system covertly in order to lie, cheat, or steal". According to [28], conventional techniques are insufficient due to the increasing integration of system functionality across different *trust domains* (cf. [20, 19] for a formal definition of trust domains). Within and across such domains, abuse of trust must be not only detectable (i.e., knowable), but also *provable*, in order to protect agents who behave correctly (the correct agents) from agents who do not (the faulty agents). Provability protects correct agents from faulty agents because provability is a necessary and sufficient condition for the *non-repudiation* of faulty behaviour: if you have a proof that I (in)effectuated an illegitimate (in)action then I cannot repudiate having (in)effectuated that (in)action, and vice versa. Note that the provability of a state of affairs is strictly stronger than the knowledge that that state is the case. For example, an agent may know that a certain state of affairs is the case *from observation*, yet not be able to prove her knowledge to the non-observers (e.g., a judge) for lack of *sufficient evidence* (i.e., proof). Conversely, correct agents should be able to prove their correct behaviour to the other agents (e.g., in order to protect themselves from corrupt auditors). Note that the creation of an accountable Internet has been proposed as a national goal for cyberspace [22].

Goal The authors of [28] argue for the need of "new ways of thinking about dependability and new methodologies to build and evaluate dependable systems". Thereby, "[t]he key challenge is to develop general and practical methodologies and techniques that can approach the ideal of full accountability for networked systems". Our ambition is to take up and meet this challenge. In the present paper, our goal is to distil the declarative essence of accountability, and to deliver a formal framework for achieving accountability in distributed or multi-agent systems through the ideal of formal proof.

Contribution This paper applies the *formal methodology* of multi-agent systems [25] to a real-world distributed system supposed to guarantee accountability. Being application-driven, we focus on the *application* of the methodology rather than the meta-logical study of our (mostly standard) framework within which we illustrate our methodology. Further, when applying the methodology, we will focus on semantic proof, in order to make explicit the *intuitive reasoning* content rather than the automatisable, computational content of accountability. The formal treatment of accountability is a recent research topic; it is therefore prudent to clarify our intuitions of and reasoning about accountability before we try to automatise our reasoning about it with computers.

More precisely, our contribution is five-fold (cf. Section 3.2 for related work):

1. a general, declarative *definition* of accountability that achieves the ideal of a formal transcription of the original, natural-language formulation from the distributed systems community (cf. Sections 2.2–2.2)

2. the isolation and formalisation of three logically sufficient and modelling-wise necessary *conditions* under which distributed systems (as modelled in our framework) are indeed accountable (cf. Section 2.2)
3. a flexible formal *framework* for achieving accountability in distributed systems (cf. Section 2.1) that provides:
   (a) powerful descriptive idioms in the form of three primitives for *logs* (introduced here) and multiple modal operators (standard or introduced in [18])
   (b) an intuitive semantic setting for developing succinct formal proofs
4. a generic *pre-establishment* of accountability (cf. Section 2.2) that allows the proof for any candidate system to be:
   (a) factored into the contingent (i.e., application-specific) and the logical (i.e., conceptual) content
   (b) reduced to the proof of a few key lemmata
5. a principled *case study* as an illustrative example of how to apply our framework to a real-world system (cf. Section B).

Methodology In order to meet our goal, we formalise accountability—and the assumptions based on which it can be established—in a multi-modal language of a model-theoretic framework. The key concept for our formalisations is a modal notion of agent-based *provability.* This methodology will yield the general and declarative definition of accountability and the flexibility of the corresponding framework that we seek.

2 Accountability

In this section, we present our multi-modal framework for achieving accountability in distributed systems through formal proof. The framework provides modalities for knowledge, provability, and time. Knowledge will be used in the definition of provability; and provability, like also time, will be used in the formalisation of the two main aspects of accountability. According to the distributed-systems community [28], these aspects are: ***accountability soundness (accountability proper),*** i.e., for correct agents, the provability of their correctness by themselves. ***Accountability completeness (auditability),*** i.e., for faulty agents, the eventual provability of their faultiness by others. We recall that agent correctness (actually dually, faultiness) is a fundamental, primitive notion for distributed systems, whose guarantees are *conditioned* on this notion [23]. (A typical condition is that there be a minimal number of correct agents.)

2.1 Framework

We define our multi-modal framework model-theoretically for the sake of greater *flexibility* of modelling and proving in the framework as well as extending it. (Recall that in logic, 'model-theoretic' means 'set-theoretic'.) More precisely:

1. when modelling in our framework, we may exploit the *definitional power* of our set-theoretic semantic setting, e.g., for the application-specific definition of agent correctness (cf. Appendix B.3 for a functional example)
2. for proving in our framework and when extending it, we need not worry about:
   (a) difficult or even impossible completeness proofs of axiomatisations because our framework is set up in the already axiomatised set-theoretic setting
   (b) constraining decidability and complexity issues, which, when respected, could limit the expressiveness and thus applicability of our framework
3. when proving (semantically) in our framework, we may fully exploit the:
   (a) *descriptive power* of our modal operators, which are logical formalisations of frequently needed, intuitive natural-language idioms
   (b) *deductive power* of the deduction theorem[3], which we could not when proving syntactically because axiomatisations of knowledge and time (which are concepts to which we need appeal) do not enjoy the (global) deduction theorem [10].

With our framework, we get the advantages of the deductive world of (semantic) proofs and the descriptive world of modal operators, *without* the disadvantages of either world, namely the need for completeness proofs and the absence of a (global) deduction theorem for most modal proof systems.

Communication model For the sake of the faithful modelling of distributed systems, we choose *message passing* (as opposed to shared-variable communication) as the underlying communication model of our framework. For that, we define the following generic *message language*.

Definition 1 (Message language and derivation). *Let $\mathcal{A}$ designate an arbitrary finite set of unique* agent names[4] *a, b, c etc. Then,*

$$\mathcal{M} \ni M ::= B \mid a \mid \lceil M \rceil \mid [M]_a \mid (M, M) \mid \text{“}S\text{”}$$

defines the set $\mathcal{M}$ of message terms *M with application-specific base data B, agent names*[5] *a, message hashes $\lceil M \rceil$, signed messages $[M]_a$, message pairs (M, M), and quoted pieces of syntax S such as message-carrying events (see below for their definition and Section B for an example of their quoted use in log messages).*

Further,

$$\mathcal{L} \subseteq \mathcal{M}$$

[3] The property of a proof system that implications can be proven from the conclusion of their consequent under the hypothesis of their antecedent.

[4] i.e., agent names injectively map to agents (here, names are identifiers)

[5] Agents are referred to by their (unique) name, which are transmittable data, i.e., messages. Agents can be network nodes, parties, processes, processors, real or virtual machines, users, etc.

designates a set of application-specific logs L*, to be determined by the considered application (cf. Section B for an example).*

Furthermore, $\vdash_a \subseteq 2^{\mathcal{M}} \times \mathcal{M}$ *designates a relation of* message derivation *à la Dolev-Yao [8] for agent* $a \in \mathcal{A}$ *such that for all* $\mathcal{D} \subseteq \mathcal{M}$*:*

- $\mathcal{D} \cup \{M\} \vdash_a M$ *(the trivial derivation)*
- *for all* $b \in \mathcal{A}$*,* $\mathcal{D} \vdash_a b$ *(agent names are guessable)*
- *if* $\mathcal{D} \vdash_a M$ *then* $\mathcal{D} \vdash_a \lceil M \rceil$ *(hashing)*
- *if* $\mathcal{D} \vdash_a M$ *then* $\mathcal{D} \vdash_a [M]_a$ *(personal signature synthesis)*
- *for all* $b \in \mathcal{A}$*, if* $\mathcal{D} \vdash_a [M]_b$ *then* $\mathcal{D} \vdash_a M$ *(universal signature "analysis" and message recovery)*
- *(*$\mathcal{D} \vdash_a M$ *and* $\mathcal{D} \vdash_a M'$*) iff* $\mathcal{D} \vdash_a (M, M')$ *([un]pairing).*

Notice that $\mathcal{M}$ is *generic* in the application-specific base data B and the quoted pieces of syntax "S", which can be freely instantiated in the considered application. Further, observe that we assume the existence of an unforgeable mechanism for signing messages, which we model with the above signature synthesis and "analysis" rules. In *trusted* distributed systems, such a mechanism is trivially given by the inclusion of the sender's name in the sent message. In *dis*trusted distributed systems, such a mechanism can be implemented with classical certificate-based or directly with identity-based [16] public-key cryptography. We also assume the existence of a mechanism for hashing messages. According to [28], "[s]ecure hashes and digital signatures are fundamental building blocks for accountable systems".

Setup We subsequently define our (mostly standard) framework, which consists of a logical language with a model-theoretic semantics. The logical language provides the following, particularly notable (epistemic) constructions:

- A relational symbol k "knows" for *individual knowledge*, which is an instance of knowledge in the sense of the *transitive* use of the verb 'to know', here 'to know *a message à la Dolev-Yao*' [8]. Individual knowledge will be modelled with the relation $\vdash_a$ of *message derivation* (cf. Definition 1).
- A standard constructor K_a "a knows that" for *propositional* knowledge, which is an instance of knowledge in the sense of the use of the verb 'to know' *with a clause*, (i.e., to know *that a statement is true*). Propositional knowledge will be modelled with an *indistinguishability relation* between system states. For a monograph on propositional knowledge, see [9].
- A constructor K_a^M "if a knew M then" for a non-trivial combination of the two previous kinds of knowledge that expresses propositional knowledge *conditioned on* the individual knowledge of M. This conditioning will be modelled as a hypothetical message reception event (e.g., from an oracle) that is inserted at the current moment in time (and thus added to the agent's individual knowledge). For a detailed exposition, see [18].
- A standard constructor CK "it is common knowledge that" for a *collective* kind of propositional knowledge. Informally, a statement ϕ is common knowledge when all agents know that ϕ is true (call this new statement ϕ'), all

agents know that ϕ' is true (call this new statement ϕ''), all agents know that ϕ'' is true (call this new statement ϕ'''), etc. Note that depending on the properties of the employed communication lines, common knowledge may have to be pre-established off those lines along other lines. For a detailed treatment of common knowledge in distributed systems, see [14].

The formal definitions for all this knowledge follow now.

Definition 2 (Framework). *Let Φ designate our logical language of* closed *formulae ϕ as defined in Table 1. There, φ denotes corresponding unary* open *formulae with a single free term variable m. Notice that the formulae above the dashed line are atomic (for elementary facts), and those below compound (formed with operators). Also, note that all operators except K_a^M, which was introduced in essentially the same form in [18], are standard [25]. Then, given the set*

$$\begin{array}{rll} \mathcal{E} \ni \varepsilon ::= & \mathtt{S}(a, M, b) & \text{“}a \text{ sends } M \text{ to } b\text{”} \\ \mid & \mathtt{R}(a, M) & \text{“}a \text{ receives } M\text{”} \end{array}$$

of system events *ε (also noted as $\varepsilon(a)$) for message sending and receiving, and the set $\mathcal{E}^*$ ($\mathcal{E}^\omega$) of* (in)finite traces *over $\mathcal{E}$, we define the* satisfaction relation *$\models \subseteq (\mathcal{E}^\omega \times \mathbb{N}) \times \Phi$ of our framework in Table 2. There,*

- *“:iff” abbreviates “by definition, if and only if”*
- *$E@i$ designates the system event (say ε) of the infinite trace $E \in \mathcal{E}^\omega$ inspected at the position $i \in \mathbb{N}$*
- $\mathrm{msg}(\varepsilon) := \{M, \text{“}\varepsilon\text{”}\}$ *where $\varepsilon \in \{\mathtt{S}(a, M, b), \mathtt{R}(a, M)\}$ designates a function for* message extraction*, which we tacitly lift from events $\varepsilon \in \mathcal{E}$ to finite traces thereof (Notice the use of quoted syntax, cf. Definition 1!)*
- *$E\lfloor i$ (resp. $E\rfloor i$) designates the finite prefix trace up to (resp. infinite suffix trace from) and including the event at the position $i \in \mathbb{N}$ of the infinite trace $E \in \mathcal{E}^\omega$*
- *$\downarrow : (\mathcal{E}^* \times \mathcal{A}) \to \mathcal{E}^*$ designates a* projection *function projecting a finite trace onto an agent's (local) view such that for all $E \in \mathcal{E}^*$ and $a \in \mathcal{A}$,*

$$\begin{aligned} \epsilon{\downarrow}a &:= \epsilon \quad \textit{(the empty trace)} \\ (\varepsilon \cdot E){\downarrow}a &:= \begin{cases} \varepsilon \cdot (E{\downarrow}a) & \textit{if } \varepsilon \in \{\mathtt{S}(a, M, b), \mathtt{R}(a, M)\} \\ E{\downarrow}a & \textit{otherwise} \end{cases} \end{aligned}$$

- logs : $\mathcal{A} \to 2^{\mathcal{L}}$ *designates an application-specific* log selection *function selecting the set of well-formed logs for a given agent, to be defined by the considered application (cf. Section B for an example)*
- $[\![\cdot]\!] : \mathcal{L} \to \mathcal{E}^*$ *designates a* log-to-trace transcription *(or* de-quotation*) function (again to be defined by the considered application)*
- $\uparrow : (\mathcal{E}^* \times \mathcal{A}) \to \mathcal{E}^\omega$ *designates a* completion *function completing finite traces with an infinite suffix of arbitrary events beyond the agent's view (for well-definedness only, i.e., conceptually inessential)*

Table 1. Logical language

$\Phi \ni \phi ::=$	$\mathsf{correct}(a)$	"a is correct"
$\mid$	$\mathsf{sends}(a, M, b)$	"a sends M to b"
$\mid$	a $\mathsf{receives}$ M	"a receives M"
$\mid$	a k M	"a knows M"
$\mid$	M wfLog a	"M is a well-formed log for a"
$\mid$	M $\mathsf{soundLog}$ a	"M is a sound log for a"
$\mid$	M $\mathsf{completeLog}$ a	"M is a complete log for a"
$\mid$	$\neg\phi$	"not ϕ"
$\mid$	$\phi \wedge \phi$	"ϕ and ϕ"
$\mid$	$\forall m(\varphi)$	"for all m, φ"
$\mid$	$\Box\phi$	"henceforth ϕ"
$\mid$	$\overline{\Box}\phi$	"so far ϕ"
$\mid$	$\mathsf{K}_a(\phi)$	"a knows that ϕ"
$\mid$	$\mathsf{K}_a^M(\phi)$	"if a knew M then a would know that ϕ"
$\mid$	$\mathsf{CK}(\phi)$	"it is commonly known that ϕ"

Table 2. Satisfaction relation

$(E, i) \models \mathsf{sends}(a, M, b)$	:iff	$E@i = \mathtt{S}(a, M, b)$
$(E, i) \models a$ $\mathsf{receives}$ M	:iff	$E@i = \mathtt{R}(a, M)$
$(E, i) \models a$ k M	:iff	$\mathrm{msg}((E{\downharpoonleft}i){\downarrow}a) \vdash_a M$
$(E, i) \models M$ wfLog a	:iff	$M \in \mathrm{logs}(a)$
$(E, i) \models M$ $\mathsf{soundLog}$ a	:iff	$(E, i) \models M$ wfLog a and $((\llbracket M \rrbracket{\uparrow}a), i) \lesssim_a (E, i)$
$(E, i) \models M$ $\mathsf{completeLog}$ a	:iff	$(E, i) \models M$ wfLog a and $(E, i) \lesssim_a ((\llbracket M \rrbracket{\uparrow}a), i)$
$(E, i) \models \neg\phi$	:iff	not $(E, i) \models \phi$
$(E, i) \models \phi \wedge \phi'$	:iff	$(E, i) \models \phi$ and $(E, i) \models \phi'$
$(E, i) \models \forall m(\varphi)$	:iff	for all $M \in \mathcal{M}$, $(E, i) \models \varphi[M/m]$
$(E, i) \models \Box\phi$	:iff	for all $j \geq i$, $(E, j) \models \phi$
$(E, i) \models \overline{\Box}\phi$	:iff	for all $j \leq i$, $(E, j) \models \phi$

$(E, i) \models \mathsf{K}_a(\phi)$	:iff	for all $(E', i') \in \mathcal{E}^\omega \times \mathbb{N}$,
		if $(E, i) \approx_a (E', i')$ then $(E', i') \models \phi$

$(E, i) \models \mathsf{K}_a^M(\phi)$	:iff	for all $M' \in \mathcal{M}$,
		if $\{M'\} \vdash_a M$
		then $(E{\downharpoonleft}i \cdot \mathtt{R}(a, M') \cdot E{\downharpoonleft}i + 1, i + 1) \models \mathsf{K}_a(\phi)$

$(E, i) \models \mathsf{CK}(\phi)$	:iff	for all $(E', i') \in \mathcal{E}^\omega \times \mathbb{N}$,
		if $(E, i) \approx_{\cup}^{*} (E', i')$ then $(E', i') \models \phi$

- $\lesssim_a \subseteq (\mathcal{E}^\omega \times \mathbb{N})^2$ *designates a pre-order expressing non-empty pre-fixing up to the currently inspected positions and modulo the agent's* $a \in \mathcal{A}$ *local view, defined such that for all* $(E, i), (E', i') \in \mathcal{E}^\omega \times \mathbb{N}$,

$$(E, i) \lesssim_a (E', i') \quad :\!\textit{iff} \quad \textit{there is } E'' \in \mathcal{E}^* \textit{ such that } ((E{\downharpoonleft}i){\downarrow}a) \cdot E'' = (E'{\downharpoonleft}i'){\downarrow}a \neq E''$$

- $\varphi[M/m]$ *designates the substitution of the (closed) term* M *for all free occurrences of the term variable* m *in the formula* φ
- $\leq$ *designates the standard total order on* $\mathbb{N}$*, and* $\geq$ *the converse of* $\leq$
- $\approx_a \subseteq (\mathcal{E}^\omega \times \mathbb{N})^2$ *designates a relation of* epistemic accessibility *expressing indistinguishability of traces up to the currently inspected positions and modulo the agent's* $a \in \mathcal{A}$ *local view, defined such that for all* $(E, i), (E', i') \in \mathcal{E}^\omega \times \mathbb{N}$,

$$(E, i) \approx_a (E', i') \quad :\!\textit{iff} \quad (E, i) \lesssim_a (E', i') \textit{ and } (E', i') \lesssim_a (E, i)$$

- $\approx^*_\cup := (\bigcup_{a \in \mathcal{A}} \approx_a)^*$, *where* * *designates the Kleene (i.e., the reflexive transitive) closure operation*

Note that the semantics of the predicate correct is application-specific, and thus left to be defined for the considered system (cf. Section B for an example). Further notice that the relation $\lesssim_a$ uniformly serves as the basis for the semantics of the predicates soundLog and completeLog as well as for the definition of the epistemic accessibility relation. Finally notice that formulae $\phi \in \Phi$ have a Herbrand-style semantics, i.e., logical constants (agent names) and functional symbols (hashing, signing, and pairing) are self-interpreted rather than interpreted in terms of (other, semantic) constants and functions. This simplifying design choice spares our framework from term-variable assignments [7]. The choice is admissible because our individuals (messages) are finite. Hence, substituting (syntactic) messages for message variables into (finite) formulae preserves the well-formedness of formulae (cf. the semantics of universal quantification).

Now, we can macro-define the following standard concepts from first-order and temporal logic: $\top := a \mathsf{k}\, a$, $\bot := \neg\top$, $\phi \vee \phi' := \neg(\neg\phi \wedge \neg\phi')$, $\phi \rightarrow \phi' := \neg\phi \vee \phi'$, $\exists m(\varphi) := \neg\forall m(\neg\varphi)$, $\Diamond\phi := \neg\Box\neg\phi$ ("eventually ϕ"), $\overline{\Diamond}\phi := \neg\overline{\Box}\neg\phi$ ("once ϕ"). Moreover, and more interestingly, we can macro-define the concepts in Table 3, which are important for our formal definition of accountability and the assumptions based on which accountability can be established. In Table 3, observe our definition of agent-based provability, which captures the idea of *proofs as sufficient evidence.* The definition stipulates that it be common knowledge among the agents that to the designated verifier a,

1. the *actual* (cf. *material* implication) and
2. the *hypothetical* (cf. *conditional* implication)

knowledge of M be individually necessary and jointly (vacuity!) sufficient for the knowledge of ϕ. Note that a material conditional alone would not do here because any message unknown to a would vacuously qualify as a proof. See [18] for a detailed exposition of our provability modality.

Table 3. Important macro-definitions

a sTrusts $b :=$	$\mathsf{K}_a(\mathsf{correct}(b))$	"a strongly trusts b"
$\mathsf{faulty}(a) :=$	$\neg\mathsf{correct}(a)$	
$M :_a \phi :=$	$\mathsf{CK}\left((a \mathrel{\mathsf{k}} M \to \mathsf{K}_a(\phi)) \land \mathsf{K}_a^M(\phi)\right)$	"M is a proof of ϕ for a"
M $\mathsf{decides}_a$ $\phi :=$	$(M :_a \phi) \lor (M :_a \neg\phi)$	"M decides ϕ for a"
$M : \phi :=$	$\forall a(M :_a \phi)$	"M is a proof of ϕ"
M decides $\phi :=$	$\forall a(M\ \mathsf{decides}_a\ \phi)$	
$\mathsf{P}_{(a,b)}(\phi) :=$	$\exists m((m :_b \phi) \land a \mathrel{\mathsf{k}} m)$	"a can prove to b that ϕ"
$\mathsf{P}_a(\phi) :=$	$\exists m((m : \phi) \land a \mathrel{\mathsf{k}} m)$	"a can prove that ϕ"
a canExpose $b :=$	$\mathsf{P}_a(\mathsf{faulty}(b))$	
M adequateLog $a :=$	M soundLog $a \land M$ completeLog a	
M decisiveLog $a :=$	M wfLog $a \land M$ decides $\mathsf{correct}(a)$	
a logs $M :=$	M wfLog $a \land \exists b(\mathsf{sends}(a, M, b))$	
M filedLog $a :=$	$\overline{\Diamond}(a \text{ logs } M)$	

Finally, following [4], we define the concepts of validity, logical consequence, and logical equivalence.

Definition 3. *A formula $\phi \in \Phi$ is* valid*, written $\models \phi$, :iff for all $(E, i) \in \mathcal{E}^\omega \times \mathbb{N}$, $(E, i) \models \phi$. A formula $\phi' \in \Phi$ is* a logical consequence of *a formula $\phi \in \Phi$, written $\phi \Rightarrow \phi'$, :iff for all $(E, i) \in \mathcal{E}^\omega \times \mathbb{N}$, if $(E, i) \models \phi$ then $(E, i) \models \phi'$. A formula $\phi' \in \Phi$ is* logically equivalent to *a formula $\phi \in \Phi$, written $\phi \Leftrightarrow \phi'$, :iff $\phi \Rightarrow \phi'$ and $\phi' \Rightarrow \phi$.*

Properties

Fact 1 $\models \phi \to \phi'$ *iff* $\phi \Rightarrow \phi'$

Proof. By expansion of definitions.

The following proposition provides further useful intuitions about our framework. Technically, the proposition proves that our framework provides standard epistemic operators, and an agent-based provability operator enjoying standard properties, and that provability implies knowledge.

Proposition 1.

1. *The auxiliary relation $\lesssim_a$ is pre- but not partially ordering.*
2. *The epistemic accessibility relation $\approx_a$ is an equivalence relation. Or, equivalently, the epistemic modality K_a is S5, i.e., captures the standard notion of knowledge (cf. [9] and [25, Section 7.1]):*
   K $\models \mathsf{K}_a(\phi \to \phi') \to (\mathsf{K}_a(\phi) \to \mathsf{K}_a(\phi'))$ *(Kripke's law / closure under implication)*
   T $\models \mathsf{K}_a(\phi) \to \phi$ *(truth law / reflexivity)*

4 $\models \mathsf{K}_a(\phi) \to \mathsf{K}_a(\mathsf{K}_a(\phi))$ *(positive introspection / transitivity)*
5 $\models \neg\mathsf{K}_a(\phi) \to \mathsf{K}_a(\neg\mathsf{K}_a(\phi))$ *(negative introspection / Euclideanness)*
N *if* $\models \phi$ *then* $\models \mathsf{K}_a(\phi)$ *(epistemic necessitation / all agents know all validities)*

3. *Being defined in terms of an equivalence relation,* CK *is S5 too. In particular, if* $\models \phi$ *then* $\models \mathsf{CK}(\phi)$*, i.e., validities are common knowledge. Further,* $\models \mathsf{CK}(\phi) \to \forall a(\mathsf{K}_a(\phi))$ *and* $\models \mathsf{CK}(\phi \to \forall a(\mathsf{K}_a(\phi))) \to (\phi \to \mathsf{CK}(\phi))$ *[25, Section 7.1].*
4. P_a *is S4, i.e., captures a standard (though even interactive) notion of Gödel-style provability [2]:*
K $\models \mathsf{P}_a(\phi \to \phi') \to (\mathsf{P}_a(\phi) \to \mathsf{P}_a(\phi'))$
T $\models \mathsf{P}_a(\phi) \to \phi$
4 $\models \mathsf{P}_a(\phi) \to \mathsf{P}_a(\mathsf{P}_a(\phi))$
N *if* $\models \phi$ *then* $\models \mathsf{P}_a(\phi)$
5. $\models \mathsf{P}_a(\phi) \to \mathsf{K}_a(\phi)$.

Proof. For the failure of the anti-symmetry of the relation $\lesssim_a$ consider the following counter-example with $a, b \in \mathcal{A}$ such that $a \neq b$, $M \in \mathcal{M}$, and $E \in \mathcal{E}^\omega$: $(\mathtt{S}(a, M, b) \cdot E, 1) \approx_a (\mathtt{S}(a, M, b) \cdot \mathtt{R}(b, M) \cdot E, 2)$, but $(\mathtt{S}(a, M, b) \cdot E, 1) \neq (\mathtt{S}(a, M, b) \cdot \mathtt{R}(b, M) \cdot E, 2)$. Thus the culprits for the failure are trace positions and the (necessary) projection of the trace onto agent (local) views.

The equivalence property of $\approx_a$ follows from the pre-order property of $\lesssim_a$.

For the proof of the properties of the provability modality and the law connecting provability to knowledge, see [18].

Lemma 1. *Individual knowledge is never forgotten. Formally,*

$$\models \forall a \forall m \Box(a \mathrel{\mathsf{k}} m \to \Box(a \mathrel{\mathsf{k}} m)).$$

Proof. Straightforward from definitions.

This lemma obviously depends on *persistent storage* for the essential parts of an agent's individual knowledge (i.e., those parts that cannot be reconstructed from other parts).

The following nice-to-know proposition gives guarded *Barcan laws* (cf. first-order epistemic logic [7]), i.e., guarded quantifiers (w.r.t. to individual knowledge of say a) can be freely extruded from and intruded into the scope of epistemic modalities (capturing propositional knowledge w.r.t. a).

Proposition 2 (Guarded Barcan laws).

1. $\models \mathsf{K}_a(\exists m(a \mathrel{\mathsf{k}} m \land \phi)) \leftrightarrow \exists m(a \mathrel{\mathsf{k}} m \land \mathsf{K}_a(\phi))$
2. $\models \mathsf{K}_a(\forall m(a \mathrel{\mathsf{k}} m \to \phi)) \leftrightarrow \forall m(a \mathrel{\mathsf{k}} m \to \mathsf{K}_a(\phi))$

Proof. Straightforward from definitions.

2.2 Pre-establishment

An important particularity of our framework is that it is parametric in the proof of a few key lemmata, whose contents we state as assumptions, and to which we reduce the soundness and the completeness proof of potential applications *a priori*, i.e., before their design. That is, our framework pre-establishes the soundness and the completeness aspect of accountability by factoring the proof of each aspect into the contingent (i.e., application-specific) and the logical (i.e., conceptual) content. As a result, the accountability proof of a particular system is reduced to the proof of the key lemmata (i.e., to the discharge of the key assumptions), which the system designer needs to establish for a considered system *a posteriori*, i.e., after its design.

More precisely, the factoring confines the contingent content to a finite conjunction $\phi' \in \Phi$ of system assumptions that logically implies the considered system goal $\phi \in \Phi$ (i.e., soundness or completeness), which contains the logical content, i.e., $\phi' \Rightarrow \phi$. Then, given a definition of $\mathsf{correct}$ for the considered system and a finite conjunction ϕ'' of system axioms[6] that logically implies the system assumptions ϕ', i.e., $\phi'' \Rightarrow \phi'$, the system axioms logically imply the system goal, i.e., $\phi'' \Rightarrow \phi$. Note that a definition of $\mathsf{correct}$ may entail a system assumption or axiom to become a system validity, and thus common knowledge, as already indicated in Proposition 1.3. This is a useful fact for formal proofs, which we will develop in Fitch-style natural deduction. (Recall that Fitch-style proofs are read *outside-in*.)

Key assumptions We make three succinct key assumptions that are sufficient for achieving the soundness and the completeness aspect of accountability in distributed systems. Our assumptions are also necessary for faithful modelling, i.e., not making one of these assumptions would imply not modelling faithfully these systems. In particular, faithful modelling requires the use of *logs* (i.e., accounting entries) of some form (cf. Section B for an example).

Two of our assumptions cannot be discharged by proof but only by psychology and physics because their discharge depends on the system user and/or the communication medium. The other assumption can be discharged by proof because its discharge only depends on the mathematics that models the cryptographic mechanisms (e.g., hashing and signing) required for the system implementations. However, observe that our assumptions are fully abstract w.r.t. these mechanisms in the sense that the assumptions do not mention functional symbols, which represent these mechanisms at the term-language level.

Assumption 1 *Decisive logs are necessarily* known *(though not necessarily* filed*, e.g., by corrupt agents)*[7]*. Formally,*

$$\mathsf{A1} := \forall a \Box \exists m (a \;\mathsf{k}\; m \land m \;\mathsf{decisiveLog}\; a).$$

[6] Here, a system axiom is a sentence stipulating a characteristic property of the considered system, and not necessarily an axiom in a proof system.

[7] Recall from the last three lines of Table 3 that, as opposed to being a *filed* log, being a mere log is a mere *well-formedness* criterion.

This assumption can be discharged by proof, on the condition that $\mathsf{correct}$ (being part of $\mathsf{decisiveLog}$, cf. Table 3) be defined for the considered system.

Assumption 2 *Faultiness is persistently provable by filed logs. Formally,*

$$\begin{aligned}\mathsf{A2} := \forall a \Box(&\mathsf{faulty}(a) \rightarrow \\ &\exists m(m \mathrel{\mathsf{filedLog}} a \land \Box(m : \mathsf{faulty}(a)))).\end{aligned}$$

This assumption cannot be discharged by proof due to its dependence on system users (e.g., the willingness of a to file a log m). Observe that this assumption implies the assumption (made also by the PeerReview and other systems [13, Section 4.4]) that log files cannot be tampered with.

Assumption 3 *Logs are eventually known. Formally,*

$$\mathsf{A3} := \forall a \forall m \Box(a \mathrel{\mathsf{logs}} m \rightarrow \Diamond \forall b(b \mathrel{\mathsf{k}} m)).$$

This assumption (made also by the PeerReview and other systems [13, Section 4.3]) cannot be discharged by proof either, due to its dependence on, again, system users (e.g., the willingness of a to commit logs m) and also, the communication medium (e.g., the reliability of the communication channels).

Soundness theorem As mentioned before, soundness in accountability means that correct agents can prove their correctness to the other agents.

We can transcribe this natural-language formulation into our formal language with the following macro-definition:

$$\boxed{\mathsf{Soundness} := \forall a \Box(\mathsf{correct}(a) \rightarrow \mathsf{P}_a(\mathsf{correct}(a))).}$$

Observe that soundness builds *trust* in the sense of Table 3, Line 1, i.e., the proof of an agent a's correctness to another agent, say b, induces the knowledge with b that a is correct (cf. [20] for a detailed exposition of this sense of trust, with example applications in cryptographic-key management). Also notice that in $\mathsf{Soundness}$, the converse implication is for free due to the truth law of P_a for arbitrary $a \in \mathcal{A}$ (cf. Proposition 1.4.T).

Theorem 1 (Accountability soundness).

1. *Assumption* $\mathsf{A1}$ *is a* sufficient *condition for* $\mathsf{Soundness}$*. Formally,*

$$\boxed{\mathsf{A1} \Rightarrow \mathsf{Soundness}.}$$

2. *Whenever accountability is* log-based *(which is almost always the case in practice), Assumption* $\mathsf{A1}$ *also is a* necessary *condition for* $\mathsf{Soundness}$.

Proof. For (1), see Table 4. For (2), consider that if there is an agent such that eventually all her known logs are non-decisive (this is the negation of $\mathsf{A1}$) then she has no (log-based) means of proving her correctness — even when she does happen to be correct.

Completeness theorem As mentioned before, completeness in accountability means that all agents can eventually always prove the faultiness of faulty agents.

We can transcribe this natural-language formulation into our formal language with the following macro-definition:

$$\boxed{\begin{aligned}\mathsf{Completeness} := \forall a\Box(\mathsf{faulty}(a) \to \\ \forall b\Diamond\Box(b\ \mathsf{canExpose}\ a)).\end{aligned}}$$

Theorem 2 (Accountability completeness). *Assumptions* A2 *and* A3 *jointly are a* sufficient *condition for* Completeness. *Formally,*

$$\boxed{\mathsf{A2} \wedge \mathsf{A3} \Rightarrow \mathsf{Completeness.}}$$

Proof. See Table 5.

Accountability theorem As mentioned before, accountability is the conjunction of accountability soundness and accountability completeness.

We macro-define,

$$\boxed{\mathsf{Accountability} := \mathsf{Soundness} \wedge \mathsf{Completeness}\,.}$$

Theorem 3 (Accountability).

1. *Assumptions* A1*,* A2*, and* A3 *jointly are a* sufficient *condition for* Completeness. *Formally,*

$$\boxed{\mathsf{A1} \wedge \mathsf{A2} \wedge \mathsf{A3} \Rightarrow \mathsf{Accountability}}$$

2. *Whenever accountability is* log-based*, Assumption* A1 *also is a* necessary *condition for* Accountability.

Proof. By Theorem 1 and Theorem 2.

3 Conclusion

3.1 Assessment

Our goal has been to take up and meet [28]'s challenge of developing a general and practical methodology and technique that can approach the ideal of full accountability for networked systems.

We have delivered this methodology and technique within a multi-modal, model-theoretic framework that is parametric in an application-specific

1. set $\mathcal{L}$ of well-formed logs (cf. Definition 1)
2. predicate correct for agent correctness (cf. Table 1).

The methodology consists in proving that the characteristic properties of a considered system (i.e., the system axioms) logically imply three logically sufficient and modelling-wise necessary conditions for accountability.

The technique then consists in:

1. instantiating $\mathcal{L}$ and $\mathsf{correct}$ for the considered system
2. formalising the system axioms with the powerful descriptive abstractions built-in in our framework, i.e., the modalities, most notably Gödel-style provability
3. proving the implication by exploiting the inferential content of these modalities.

Finally, we have illustrated the applicability of our framework on the case study of the Distributed Book-Keeping (DBK) design pattern, which we introduced as an analogue for distributed systems of Pacioli's famous double-entry book-keeping principle (cf. Section B).

We have defined DBK in terms of a distributed *accounting* daemon, which:

1. pinpoints an instance of Russel's paradox for distributed systems
2. provides *falsifiability* of agent correctness in the sense of Popper
3. effects a timeline entanglement in the sense of [24].

On top of the accounting daemon, a distributed *auditing* daemon can provide also *verifiability* of agent correctness by encouraging agents to produce proof of their behavioural status (correct or faulty). Such proofs could be encouraged by creating the appropriate *deterrents* and *incentives* for the agents in the considered system.

3.2 Related work

The following three formal approaches to accountability with a general aim exist.

To the best of our knowledge, [17] is the first to have published the idea of formalising accountability with a notion of provability. However, the author seems to have been unaware of standard modal provability, as corresponding work is not mentioned in his paper. Rather, the author construes a supposed notion of provability from two postulates: if an agent can prove two statements then the agent can prove the conjunction of them, and if an agent can prove a statement and that statement implies another statement then the agent can prove the implied statement. Hence, the author's notion of provability need not even respect the truth law, i.e., that the provability of a statement imply the truth of that statement.

To the best of our knowledge, [3] are the first to have published the idea of *proof as sufficient evidence*. However, the authors do not seem to have been aware of standard modal provability either, nor do they have a general account of provability in the sense that they "can only formalize [protocol, not agent] correctness with respect to a specific protocol", which is a proof (in our sense) that their logical formalisation of provability is not satisfactory yet.

The same comment applies for the same reason to [21], where accountability-related properties for electronic contract-signing are studied in terms of Alternating Temporal Logic (ATL).

Then, in [15], a process-algebraic programming model for distributed systems for accountability and audit is presented in the particular context of authorisation. In that approach, finitary systems are compiled to turn-based games, and ATL is used to specify system properties. The authors actually call their property formulations "logical encodings". In contrast, we define the properties in our logic directly as mere transcriptions of their natural-language formulations. Hence again, a similar comment as for the previous approach can be made.

Finally, in [11], a type-based definition of log-based auditability is given. Thereby, the authors informally refer to concepts like provability (w.r.t. identity, authenticity), agreement (w.r.t. public-key infrastructures, properties of judges), knowledge (w.r.t. properties of judges), and trust (w.r.t. agents, functions, function libraries). All these concepts can be captured formally in the logical language of our framework: provability with the provability modality, agreement with the common-knowledge modality, (property) knowledge with the knowledge modality, and trust as the knowledge of (the property of) agent correctness.

Future work In future work, we would like to employ our framework in a provability-based study of the Accountable Internet Protocol [1]. Given that our notion of provability is defined in terms of knowledge, it could be possible to reduce such a provability-based study to the knowledge-based study of the Internet Protocol without accountability [27].

References

1. D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable internet protocol (AIP). In *Proceedings of ACM SIGCOMM*, 2008.
2. S. Artemov. *Handbook of Modal Logic*, chapter Modal Logic in Mathematics. Volume 3 of Blackburn et al. [5], 2007.
3. G. Bella and L. C. Paulson. Accountability protocols: formalized and verified. *ACM Transactions on Information and System Security*, 9, 2006.
4. P. Blackburn and J. van Benthem. *Handbook of Modal Logic*, chapter Modal Logic: A Semantic Perspective. Volume 3 of Blackburn et al. [5], 2007.
5. P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier, 2007.
6. A. Blass and Y. Gurevich. The logic of choice. *The Journal of Symbolic Logic*, 65(3), 2000.
7. T. Braüner and S. Ghilardi. *Handbook of Modal Logic*, chapter First-Order Modal Logic. Volume 3 of Blackburn et al. [5], 2007.
8. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(12), 1983.
9. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

10. M. Fitting. *Handbook of Modal Logic*, chapter Modal Proof Theory. Volume 3 of Blackburn et al. [5], 2007.
11. N. Guts, C. Fournet, and F.Z. Nardelli. Reliable evidence: Auditability by typing. In *Proceedings of ESORICS*, volume 5789 of *LNCS*, 2009.
12. A. Haeberlen, P. Kuznetsov, and P. Druschel. The case for Byzantine fault detection. In *Proceedings of the IEEE Workshop on Hot Topics in System Dependability*, 2006.
13. A. Haeberlen, P. Kuznetsov, and P. Druschel. PeerReview: Practical accountability for distributed systems. In *Proceedings of the ACM Symposium on Operating Systems Principles*, 2007.
14. J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3), 1990.
15. R. Jagadeesan, A. Jeffrey, C. Pitcher, and J. Riely. Towards a theory of accountability and audit. In *Proceedings of ESORICS*, volume 5789 of *LNCS*, 2009.
16. M. Joye and G. Neven. *Identity-Based Cryptography*. IOS Press, 2009.
17. R. Kailar. Accountability in electronic commerce protocols. *IEEE Transactions on Software Engineering*, 22, 1996.
18. S. Kramer. Reducing provability to knowledge in multi-agent systems. In *Proceedings of the LiCS-affiliated Intuitionistic Modal Logics and Applications Workshop*, 2008. `ftp://ftp.research.microsoft.com/pub/tr/TR-2008-90.pdf`.
19. S. Kramer, R. Goré, and E. Okamoto. Formal definitions and complexity results for trust relations and trust domains. In *Proceedings of the ESSLLI-affiliated Workshop on Logics in Security*, 2010.
20. S. Kramer, R. Goré, and E. Okamoto. Formal definitions and complexity results for trust relations and trust domains fit for TTPs, the Web of Trust, PKIs, and ID-Based Cryptography. Logic Column of ACM SIGACT News, March 2010.
21. S. Kremer and J.-F. Raskin. Game analysis of abuse-free contract signing. In *Proceedings of the IEEE Computer Security Foundations Workshop*, 2002.
22. C. E. Landwehr. A national goal for cyberspace: Create an open, accountable internet. *IEEE Security & Privacy*, 7(3), 2009.
23. N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.
24. P. Maniatis and M. Baker. Secure history preservation through timeline entanglement. In *Proceedings of the USENIX Security Symposium*, 2002. `http://www.usenix.org/events/sec02/full_papers/maniatis/maniatis.pdf`.
25. J.-J. Meyer and F. Veltnam. *Handbook of Modal Logic*, chapter Intelligent Agents and Common Sense Reasoning. Volume 3 of Blackburn et al. [5], 2007.
26. F. B. Schneider. Accountability for perfection. *IEEE Security & Privacy*, 7(2), 2009.
27. F. Stulp and R. Verbrugge. A knowledge-based algorithm for the internet transmission protocol (TCP). *Bulletin of Economic Research*, 54(1), 2002.
28. A. R. Yumerefendi and J. S. Chase. The role of accountability in dependable distributed systems. In *Proceedings of the IEEE Workshop on Hot Topics in System Dependability*, 2005.

A Formal proofs

Table 4. Accountability soundness

1.	$(E,i) \in \mathcal{E}^{\omega} \times \mathbb{N}$	hyp.
2.	$(E,i) \models \mathsf{A1}$	hyp.
3.	$a \in \mathcal{A}$	hyp.
4.	$j \geq i$	hyp.
5.	$(E,j) \models \mathsf{correct}(a)$	hyp.
6.	there is $M \in \mathcal{M}$ s.t. $(E,j) \models a\ \mathsf{k}\ M \wedge M\ \mathsf{decisiveLog}\ a$	2, 3, 4
7.	$M \in \mathcal{M}$ and $(E,j) \models a\ \mathsf{k}\ M \wedge M\ \mathsf{decisiveLog}\ a$	hyp.
8.	$(E,j) \models M : \mathsf{correct}(a) \vee M : \neg\mathsf{correct}(a)$	7
9.	$(E,j) \models \neg(M : \neg\mathsf{correct}(a))$	5
10.	$(E,j) \models M : \mathsf{correct}(a)$	8, 9
11.	$(E,j) \models a\ \mathsf{k}\ M$	7
12.	$(E,j) \models \mathsf{P}_a(\mathsf{correct}(a))$	10, 11
13.	$(E,j) \models \mathsf{P}_a(\mathsf{correct}(a))$	6, 7–12
14.	$(E,j) \models \mathsf{correct}(a) \rightarrow \mathsf{P}_a(\mathsf{correct}(a))$	5–13
15.	$(E,i) \models \Box(\mathsf{correct}(a) \rightarrow \mathsf{P}_a(\mathsf{correct}(a)))$	4–14
16.	$(E,i) \models \forall a \Box(\mathsf{correct}(a) \rightarrow \mathsf{P}_a(\mathsf{correct}(a)))$	3–15
17.	$(E,i) \models \mathsf{Soundness}$	16
18.	$(E,i) \models \mathsf{A1} \rightarrow \mathsf{Soundness}$	2–17
19.	$\mathsf{A1} \Rightarrow \mathsf{Soundness}$	1–18

Table 5. Accountability completeness

1.	$(E, i) \in \mathcal{E}^{\omega} \times \mathbb{N}$	hyp.
2.	$(E, i) \models \mathsf{A2} \wedge \mathsf{A3}$	hyp.
3.	$a \in \mathcal{A}$	hyp.
4.	$j \geq i$	hyp.
5.	$(E, j) \models \mathsf{faulty}(a)$	hyp.
6.	$b \in \mathcal{A}$	hyp.
7.	there is $M \in \mathcal{M}$ s.t. $(E, j) \models M\ \mathsf{filedLog}\ a \wedge \Box(M : \mathsf{faulty}(a))$	2[A2], 3, 4, 5
8.	$M \in \mathcal{M}$ and $(E, j) \models M\ \mathsf{filedLog}\ a \wedge \Box(M : \mathsf{faulty}(a))$	hyp.
9.	$(E, j) \models M\ \mathsf{filedLog}\ a$	8
10.	$(E, j) \models \Diamond(b\ \mathsf{k}\ M)$	2[A3], 6, 9
11.	$(E, j) \models \Diamond\Box(b\ \mathsf{k}\ M)$	6, 10, Lemma 1
12.	$(E, j) \models \Box(M : \mathsf{faulty}(a))$	8
13.	$(E, j) \models \Diamond\Box(M : \mathsf{faulty}(a))$	12
14.	$(E, j) \models \Diamond\Box((M : \mathsf{faulty}(a)) \wedge b\ \mathsf{k}\ M)$	11, 13
15.	$(E, j) \models \Diamond\Box\exists m((m : \mathsf{faulty}(a)) \wedge b\ \mathsf{k}\ m)$	14
16.	$(E, j) \models \Diamond\Box\mathsf{P}_b(\mathsf{faulty}(a))$	15
17.	$(E, j) \models \Diamond\Box(b\ \mathsf{canExpose}\ a)$	16
18.	$(E, j) \models \Diamond\Box(b\ \mathsf{canExpose}\ a)$	7, 8–17
19.	$(E, j) \models \forall b \Diamond\Box(b\ \mathsf{canExpose}\ a)$	6–18
20.	$(E, j) \models \mathsf{faulty}(a) \rightarrow \forall b \Diamond\Box(b\ \mathsf{canExpose}\ a)$	5–19
21.	$(E, i) \models \Box(\mathsf{faulty}(a) \rightarrow \forall b \Diamond\Box(b\ \mathsf{canExpose}\ a))$	4–20
22.	$(E, i) \models \forall a \Box(\mathsf{faulty}(a) \rightarrow \forall b \Diamond\Box(b\ \mathsf{canExpose}\ a))$	3–21
23.	$(E, i) \models \mathsf{Completeness}$	22
24.	$(E, i) \models (\mathsf{A2} \wedge \mathsf{A3}) \rightarrow \mathsf{Completeness}$	2–23
25.	$\mathsf{A2} \wedge \mathsf{A3} \Rightarrow \mathsf{Completeness}$	1–24

B Case study

In this section, we illustrate the applicability of our framework on the principled, real-world case study of the Distributed Book-Keeping (DBK) design pattern. Our case study is real-world in the sense that the intuition of the design pattern is already implicit in the design of Byzantine Fault Detection [12] and in its implementation, the PeerReview system [13]. The case study is meant to exemplify how to distil the declarative essence of an accountable distributed system in our multi-modal, model-theoretic framework.

B.1 The Distributed Book-Keeping design pattern DBK

Distributed Book-Keeping (DBK) is a design pattern for accountable distributed systems, which we define hereafter in terms of a *distributed accounting daemon*.

DBK is the analogue for distributed systems of Fra Luca Bartolomeo de Pacioli's (1446/7–1517) codification of the Venetian double-entry book-keeping principle for financial transactions. In DBK, the analogue of a financial transaction is a message-passing communication, and the analogue of a crediting or debiting account entry is a sending- or receiving-event log entry, respectively. More precisely in DBK, the sending or receiving of a *data* message by an agent must match the immediate sending of an adequate *log* message of the corresponding *quoted* data-message *event* to the other agents. (Bear in mind that we cannot transmit events *as such*, i.e., without quotation. After all, a sending or reception event symbolises the very transmission of a message. Thus we do need to marshal/serialise events for their own transmission, which is what their quotation symbolises.) DBK is a design *pattern* in the sense that DBK delegates the specification of the following control and data refinements to designs:

- resolution of non-determinism (e.g., scheduling of broadcasts)
- utilisation of storage (e.g., choice of data structures)
- computational efficiency (e.g., hashing).

The designs in turn delegate the realisation of these specifications to implementations.

However, what DBK does specify is the definition of the predicate $\mathsf{correct}$, and it does so by means of the distributed accounting daemon shown in Table 6, as follows:

$$(E, i) \models \mathsf{correct}(a) \text{ :iff } (E\lfloor i)\downarrow a \text{ is compatible with } \texttt{Accounting-Daemon(}a\texttt{)},$$

where to be compatible with agent a's accounting daemon means for the finite trace $(E\lfloor i)\downarrow a$ to exhibit the *event* $\mathtt{S}(a, L', c)$ (generated by the *action* `send(L', c)`) whenever a `WHEN`-guard applies at its currently inspected position — for all positions (cf. Appendix B.3 for a formal definition in terms of a quadratic-time functional program).

`Accounting-Daemon` employs the following notable programming constructs:

Table 6. Distributed accounting daemon

```
PROGRAM Accounting-Daemon (a : agent) {
  LOOP SUCH-THAT (b : agent) AND (D : data) {
    WHEN sends(a, sign(D, a), b) {
      FOR EVERY agent c EXCEPT {a,b} {
        CHOOSE log L SUCH-THAT adequateLog(L', a) {
          send(L', c)
        } WHERE L' = sign((quote(S(a, sign(D, a), b)), L), a)
      }
    }
    WHEN receives(a, sign(D, b)) {
      FOR EVERY agent c EXCEPT a {
        CHOOSE log L SUCH-THAT adequateLog(L', a) {
          send(L', c)
        } WHERE L' = sign((quote(R(a, sign(D, b))), L), a)
      }
    }
  }
}
```

– `sends(a, sign(D, a), b)` and `receives(a, sign(D, b))`, in logical notation $\mathsf{sends}(a, [D]_a, b)$ and a $\mathsf{receives}$ $[D]_b$, respectively
– the declarative programming abstraction `CHOOSE`—`SUCH-THAT` for storage utilisation, which we use as an analogue in our programming setting of Hilbert's choice operator in logical settings (cf. [6], where this operator is also mentioned as a specification construct in Abstract-State-Machine programs), e.g., "`CHOOSE` x `SUCH-THAT` $P(x)$" chooses an x such that x has the property P
– `adequateLog(L', a)`, in logical notation L' $\mathsf{adequateLog}$ a, whose truth condition relies on the two parameters of our framework, namely:
  1. the set (and function) $\mathcal{L}$ of (an agent's) well-formed logs, which for all $\varepsilon \in \mathcal{E}$ and $a \in \mathcal{A}$ we fix as

$$\mathcal{L} \ni L ::= [\text{“}\varepsilon\text{”}]_a \mid [(\text{“}\varepsilon\text{”}, L)]_a$$

  2. the obvious function $[\![\cdot]\!] : \mathcal{L} \to \mathcal{E}^*$ transcribing log messages to event traces.

Observe that *inside* the accounting daemon, the sending of a log that is complete *a priori* (i.e., before sending) immediately entails its incompleteness *a posteriori* (i.e., after sending). In other words, sending a log that is also complete *a posteriori* is impossible due to Russel's paradox — the log would have to contain itself. (This state of affairs will motivate the introduction of *semi-decisive logs* in the next section.) Hence without further ado, we have the following two validities, and thus pieces of common knowledge among agents (cf. Proposition 1.3).

Fact 2 (Intrinsic incompleteness of committed logs) *For all agents $a \neq b$ and messages M:*

1. $\models a \mathsf{\ logs\ } M \rightarrow \neg(M \mathsf{\ completeLog\ } a)$
2. $\models (b \mathsf{\ k\ } M \wedge M \mathsf{\ wfLog\ } a) \rightarrow \neg(M \mathsf{\ completeLog\ } a)$

Whence the following two corollaries with practical impacts on the programming of accountable distributed systems and the philosophy thereof.

Corollary 1 (The purpose of audit). *An agent's correctness can only be proven by that very agent itself* outside *the accounting daemon, e.g., within the protocol of a meta- or* auditing *daemon, which is why the addition of an auditing daemon is desirable.*

Corollary 2 (Falsifiability of agent correctness). *The other agents must content themselves with the possibility of proving an agent's faultiness* **if so***, i.e., the accounting daemon exactly provides falsifiability in the sense of Popper's critical rationalism, namely Popper's dictum that a hypothesis (here, agent correctness) should be falsifiable in the sense that if the hypothesis is false then its falsehood should be cognisable (here, by another agent).*

Further observe that `Accounting-Daemon` only logs sent and received *data* but not *log* messages because logging log messages would entail an explosion in network traffic, and thus immediately saturate the network bandwidth. However, it is conceivable to refine `Accounting-Daemon` with *bounded* logging of log messages, which would induce logging degrees, which in turn would induce degrees of mutuality of knowledge between agents. Recall that mutuality of knowledge between agents can be expressed with nested knowledge modalities (e.g., $\mathsf{K}_a(\mathsf{K}_b(\mathsf{K}_a(\phi)))$), but that the evaluation of the truth of the resulting formulae requires the communication of at least one acknowledgement per nesting between the agents. Ultimately, logging degrees induce degrees of accountability since accountability is defined in terms of provability, which in turn is defined in terms of knowledge (cf. Table 2).

Finally note that `Accounting-Daemon` actually effects a *timeline entanglement* in the sense of [24], such that the effectuated entanglement is:

- all-to-all, i.e., *all* agents are to entangle their timelines with *all* other agents
- data event-driven, i.e., only *data* sending and reception *events* are to be entangled
- all-inclusive, i.e., *all* other agents' logs are to be *included* in an agent's own logs.

For details on timeline entanglement, see [24].

B.2 Results

DBK yields the discharge of Assumption $\mathsf{A1}$ as Corollary 4 by way of Corollary 3 and Lemma 2.

Lemma 2. $\models \forall a \Box \exists m(a \mathsf{\ k\ } m \land m \mathsf{\ adequateLog\ } a)$

Proof. By inspection of the definition of k and $\mathsf{adequateLog}$.

Corollary 3. $\models \forall m \forall a(m \mathsf{\ adequateLog\ } a \to m \mathsf{\ decisiveLog\ } a)$

Proof sketch 1 *By induction over the structure of logs $L \in \mathcal{L}$ employing disjunction of the contrary cases for $a \in \mathcal{A}$ to be or not to be correct.*

The base case is

$$\models [\text{“}\varepsilon(a)\text{”}]_a \mathsf{\ adequateLog\ } a \to [\text{“}\varepsilon(a)\text{”}]_a \mathsf{\ decisiveLog\ } a\ ,$$

and the inductive case is

$$\models \begin{pmatrix} [(\text{“}\varepsilon(a)\text{”}, L)]_a \mathsf{\ adequateLog\ } a \\ \land\ L \mathsf{\ decisiveLog\ } a \end{pmatrix} \to [(\text{“}\varepsilon(a)\text{”}, L)]_a \mathsf{\ decisiveLog\ } a\ .$$

Corollary 4. $\models \mathsf{A1}$

Proof. By Lemma 2 and Corollary 3.

Hence,

$$\boxed{\models \mathsf{Soundness.}}$$

That is, our distributed *accounting* daemon *ensures soundness.* (Dually, a distributed *auditing* daemon *encourages completeness.*)

Finally, DBK yields the conditional discharge of Assumption A2 as Corollary 6 by way of Lemma 4 and Corollary 5 from Lemma 3. (Recall from Section 2.2 that the assumptions for accountability completeness cannot be fully discharged by proof.)

Lemma 3. $\models \forall a(\mathsf{correct}(a) \to \forall m(m \mathsf{\ filedLog\ } a \to m \mathsf{\ soundLog\ } a))$

Proof. By inspection of the definition of $\mathsf{correct}$.

Corollary 5.

$$\models \forall m \forall a \Box(m \mathsf{\ filedLog\ } a \to m \mathsf{\ semiDecisiveLog\ } a)$$

where $m \mathsf{\ semiDecisiveLog\ } a := \neg(m \mathsf{\ soundLog\ } a) \to m : \mathsf{faulty}(a)$.

Proof. See Table 7, where: (1) Line 8 follows from Lemma 3 and the necessitation law (**N**) for K_b (cf. Proposition 1). (2) Line 12 follows from Line 5 and 6, and the facts that (2.1) logs (e.g., M) bear the signature of their generator (here a, thus $(E, j) \models \mathsf{K}_b(M \mathsf{\ filedLog\ } a)$), and (2.2) unsound logs (here M) in a given system state (here (E, j)) of the system execution tree remain unsound in any other (in particular $\approx_b$-accessible) state of the same tree (thus $(E, j) \models \mathsf{K}_b(\neg(M \mathsf{\ soundLog\ } a)))$. (3) Line 13 and 14 follow both from Line 11 and 12 by Kripke's law (**K**) for K_b (cf. Proposition 1). (4) Common knowledge in Line 15 follows from the fact that the form and effect of logs is (pre-established) common knowledge.

Table 7. The accounting daemon forces semi-decisive logs

1.	$(E,i) \in \mathcal{E}^\omega \times \mathbb{N}$	hyp.
2.	$M \in \mathcal{M}$	hyp.
3.	$a \in \mathcal{A}$	hyp.
4.	$j \geq i$	hyp.
5.	$(E,j) \models M \ \mathsf{filedLog}\ a$	hyp.
6.	not $(E,j) \models M \ \mathsf{soundLog}\ a$	hyp.
7.	$b \in \mathcal{A}$	hyp.
8.	$\models \mathsf{K}_b(\forall a(\mathsf{correct}(a) \to \forall m(m \ \mathsf{filedLog}\ a \to m \ \mathsf{soundLog}\ a)))$	cf.
9.	$\models \mathsf{K}_b(\forall a \forall m(\mathsf{correct}(a) \to (m \ \mathsf{filedLog}\ a \to m \ \mathsf{soundLog}\ a)))$	8
10.	$\models \mathsf{K}_b(\mathsf{correct}(a) \to (M \ \mathsf{filedLog}\ a \to M \ \mathsf{soundLog}\ a))$	2, 3, 9
11.	$\models \mathsf{K}_b((M \ \mathsf{filedLog}\ a \wedge \neg(M \ \mathsf{soundLog}\ a)) \to \mathsf{faulty}(a))$	10
12.	$(E,j) \models \mathsf{K}_b(M \ \mathsf{filedLog}\ a \wedge \neg(M \ \mathsf{soundLog}\ a))$	cf.
13.	$(E,j) \models b \ \mathsf{k} \ M \to \mathsf{K}_b(\mathsf{faulty}(a))$	11, 12
14.	$(E,j) \models \mathsf{K}_b^M(\mathsf{faulty}(a))$	11, 12
15.	$(E,j) \models \mathsf{CK}\left((b \ \mathsf{k} \ M \to \mathsf{K}_b(\mathsf{faulty}(a))) \wedge \mathsf{K}_b^M(\mathsf{faulty}(a))\right)$	13, 14
16.	$(E,j) \models M :_b \mathsf{faulty}(a)$	15
17.	$(E,j) \models M : \mathsf{faulty}(a)$	7–16
18.	$(E,j) \models M \ \mathsf{semiDecisiveLog}\ a$	6–17
19.	$(E,j) \models M \ \mathsf{filedLog}\ a \to M \ \mathsf{semiDecisiveLog}\ a$	5–18
20.	$(E,j) \models \Box(M \ \mathsf{filedLog}\ a \to M \ \mathsf{semiDecisiveLog}\ a)$	4–19
21.	$(E,j) \models \forall a \Box(M \ \mathsf{filedLog}\ a \to m \ \mathsf{semiDecisiveLog}\ a)$	3–20
22.	$(E,j) \models \forall m \forall a \Box(m \ \mathsf{filedLog}\ a \to m \ \mathsf{semiDecisiveLog}\ a)$	2–21
23.	$\models \forall m \forall a \Box(m \ \mathsf{filedLog}\ a \to m \ \mathsf{semiDecisiveLog}\ a)$	1–22

Lemma 4. $\models \forall a \Box(\mathsf{faulty}(a) \to \Box(\mathsf{faulty}(a)))$

Proof. By inspection of the definition of correct.

Corollary 6. *If* $\models \forall a \Box(\mathsf{faulty}(a) \to \exists m(m \ \mathsf{filedLog}\ a))$ *then* $\models \mathsf{A2}$.

Proof. By Corollary 5 and Lemma 4.

Hence,

$$\boxed{\begin{array}{l}\text{If} \models \forall a \Box(\mathsf{faulty}(a) \to \exists m(m \ \mathsf{filedLog}\ a)) \\ \text{then } \mathsf{A3} \Rightarrow \mathsf{Completeness}.\end{array}}$$

Notice the three different implications.

B.3 Formal definition of agent correctness in DBK

We present a definition of correct in Table 8. For concreteness, our presenta-

Table 8. Functional program for the correctness predicate.

```
let rec prefix xs ys =
  match xs, ys with
    | x :: xs’, y :: ys’ when x = y ->
        prefix xs’ ys’
    | [], _ -> true
    | _ -> false
let msgOf e =
  match e with S (_, m, _) | R (_, m) -> m
let sender e =
  match e with S (a, _, _) | R (a, _) -> a
let rec containsQuoted m =
  match m with
    | BaseData _ | Agent _ -> false
    | Hash m’ | Signed (m’, _) ->
        containsQuoted m’
    | Pair (m’, m”) ->
        containsQuoted m’ || containsQuoted m”
    | Quoted _ -> true
let proj es a =
  List.filter (fun e -> sender e = a) es
let rec wfLog l =
  match l with
    | Signed (Quoted(e), _) ->
        not(containsQuoted (msgOf e))
    | Signed (Pair(Quoted(e), l’), _) ->
        not(containsQuoted (msgOf e))
        && wfLog l’
    | _ -> false
let rec logToTrace l =
  match l with
    | Signed (Quoted(e), a) -> [e]
    | Signed (Pair(Quoted(e), l’), a) ->
        (logToTrace l’) @ [e]
    | _ -> raise LogNotFound
let soundLog a l es =
  prefix (proj l a) (proj es a)
let completeLog a l es =
  prefix (proj es a) (proj l a)
let adequateLog a l es =
  soundLog a l es && completeLog a l es
let rec findLogs a agents es =
  if is_empty agents then [], es
  else
    match es with
      | R (c, (Signed (_, a’) as l)) :: es’ when
          mem c agents && a’ = a && wfLog l ->
          let ls, es” =
            findLogs a (remove c agents) es’ in
            l :: ls, es”
      | _ -> raise LogNotFound
let rec correctIter a agents sofar rest =
  match rest with
    | e :: rest’ ->
        if sender e = a then
          let sofar’ = sofar @ [e] in
          let ls, rest” =
            findLogs a agents rest’ in
            List.for_all
              (fun l ->
                 adequateLog
                   a (logToTrace l) sofar’
              ) ls
            &&
              correctIter a agents sofar’ rest”
        else
          correctIter a agents sofar rest’
    | [] -> true
let correct a agents es =
  try correctIter a agents [] es
  with LogNotFound -> false
```

Table 9. Data types for the correctness predicate.

```
type basedata = string
type agent = string
type message =
  | BaseData of basedata
  | Agent of agent
  | Hash of message
  | Signed of message * agent
  | Pair of message * message
  | Quoted of event
and event =
  | S of agent * message * agent
  | R of agent * message
exception LogNotFound
```

tion uses a functional style and is given in the functional programming language OCaml. The function defining correct is shown on Line 68. We describe its components below.

First, Table 9 declares OCaml data types that represent messages and events according to Definition 2. We assume that base-data items and agent names are strings. Messages and events are terms of respective algebraic data types, which are mutually recursive. We represent (finite) traces by lists of events, where the head of the list is the first event in the trace. We will use exceptions when checking logs, hence a corresponding exception declaration. Next, we define auxiliary functions. The function `msgOf` extracts the message from a given event using pattern-matching, `sender` returns the sender of a sending event, `containsQuoted` tests if a message contains a quoted event `Qouted(...)` by recursively traversing the message structure, `proj` projects a trace onto an agent's view using a standard library function that filters out non-local events, cf. $\downarrow$ on Page 7. We check if a message represents a well-formed log using `wfLog`, as defined on Page 20. Such logs can be transcribed into events by applying the function `logToTrace`, while any attempt to transcribe a message that is not a well-formed log raises an exception.

The compatibility condition requires that each send and receive event is adequately logged, and the log distribution is required to take place immediately after the event occurs. The functions `soundLog`, `completeLog`, and their composition `adequateLog` are used to compare a logged sequence of events with a given trace, cf. soundLog, completeLog, and adequateLog in Table 3. They use a standard function `prefix` that tests the prefix relation on pairs of lists, which is applied on projected traces. We find logging events using the function `findLogs`. It succeeds if a trace `es` contains a log event for each agent that is participating in the system and is required to receive a log according to the accounting daemon. We assume that agent names are kept in a set `agents`, which we manipulate using standard functions `is_empty`, `mem`, and `remove` that check for emptiness, element membership, and remove an element, respectively. To find a log event for each participating agent, `findLogs` scans the trace until it finds a log event for each participating agent. As a result, `findLogs` either returns a list of log

Table 10. Functional program for the accounting daemon.

```
let adeqLog : message ref = ref (BaseData "")
let logSend a m b =
  adeqLog := Signed (Pair (Quoted (S (a, m, b)),
                          !adeqLog), a)
let logRecv a m =
  adeqLog := Signed (Pair (Quoted (R (a, m)),
                          !adeqLog), a)
let send m b = ()
let mkLogMsg m = Pair (m, BaseData "log")
let isLogMsg m =
  match m with
    | Pair (_, BaseData "log") -> true
    | _ -> false
let onSend a m b agents =
  match m with
    | Signed (m', a') when
        a = a' && not(isLogMsg m') ->
        logSend a m b;
        List.iter
          (fun c ->
             if c <> a && c <> b then
               send
                 (mkLogMsg !adeqLog) c
          ) agents
    | _ -> ()
let onReceive a m agents =
  match m with
    | Signed (m', b) when not(isLogMsg m') ->
        logRecv a m;
        List.iter
          (fun c ->
             if c <> a then
               send (mkLogMsg !adeqLog) c
          ) agents
    | _ -> ()
```

events and the rest of es without the log events, or raises an exception if some log event is missing.

We check if an agent is compatible with the DBK pattern using the function correct. It takes an agent name a, a set of participating agents agents including a and a trace es as inputs. correct iterates over the trace using the function correctIter, which checks if each event originated by a is treated as the accounting daemon prescribes by searching for the corresponding log events, transcribing them and checking for adequacy.

The upper bound on the time complexity of checking agent correctness is quadratic in the length of the trace. The function correct calls correctIter for each event occurring in the trace, and each call to correctIter iterates over such events while checking adequacy of logs.

B.4 A functional implementation of the accounting daemon for DBK

We present a functional implementation of the accounting daemon in Table 10. This definition refines the declarative formulation of the distributed account-

Pos	Agent a	Agent b	Agent c
1	$\mathtt{S}(a, [M]_a, b)$		
2	$\mathtt{S}(a, [\text{“}\mathtt{S}(a, [M]_a, b)\text{”}, L_a]_a, c)$		
3		$\mathtt{R}(b, [M]_a)$	
4			$\mathtt{R}(c, [\text{“}\mathtt{S}(a, [M]_a, b)\text{”}, L_a]_a)$
5		$\mathtt{S}(b, [\text{“}\mathtt{R}(b, [M]_a)\text{”}, L_b]_b, a)$	
6		$\mathtt{S}(b, [\text{“}\mathtt{R}(b, [M]_a)\text{”}, L_b]_b, c)$	
7	$\mathtt{R}(a, [\text{“}\mathtt{R}(b, [M]_a)\text{”}, L_b]_b)$		
8			$\mathtt{S}(c, [\text{“}\mathtt{R}(b, [M]_a)\text{”}, L_b]_b)$

Fig. 1. Example trace for accounting daemon. We assume that initially agents a and b locally store logs L_a and L_b, respectively.

ing daemon shown in Table 6, in particular by implementing the declarative construct `SELECT-SUCH-THAT` using a mutable store and its update.

We assume that each agent has a local mutable store `adeqLog` that is used to keep the log. An agent can use the functions `logSend` and `logRecv` to update the stored log in order to take into account event sending and receiving, respectively. We assume that agents use the function `send m a` for sending a message `m` to an agent `a`. In order to avoid logging of logs, as discussed before, we use messages of a special form for the communication of logs, as facilitated by the function `mkLogMsg`.

An agent can implement accountability by executing `onSend` and `onReceive` upon each occurrence of send and receive event respectively.

Figure 1 illustrates the distributed accounting daemon. It presents a trace in which agents respect the accounting daemon by sending appropriate log messages. At point 1, the agent a sends a data message to b and logs it by issuing a logging message at point 2, which is sent to the agent c. Upon reception of the data message, the agent b issues corresponding log messages, see points 5 and 6. Note that log messages do not trigger further logging, hence, for example, the agent c does not send any messages.