

# Operational and Epistemic Approaches to Protocol Analysis: Bridging the Gap

Francien Dechesne<sup>1</sup>, MohammadReza Mousavi<sup>1,2</sup>, and Simona Orzan<sup>1</sup>

<sup>1</sup> Department of Computer Science, Eindhoven University of Technology,  
P.O. Box 513, NL-5600MB, Eindhoven, The Netherlands

<sup>2</sup> Department of Computer Science, Reykjavík University,  
Kringlan 1, IS-103, Reykjavík, Iceland.

**Abstract.** Operational models of (security) protocols, on one hand, are readable and conveniently match their implementation (at a certain abstraction level). Epistemic models, on the other hand, are appropriate for specifying knowledge-related properties such as anonymity or secrecy. These two approaches to specification and verification have so far developed in parallel and one has either to define ad hoc correctness criteria for the operational model or use complicated epistemic models to specify the operational behavior. We work towards bridging this gap by proposing a combined framework which allows for modeling the behavior of a protocol in a process language with an operational semantics and supports reasoning about properties expressed in a rich logic which combines temporal and epistemic operators.

## 1 Introduction

Knowledge-related aspects are currently being recognized as very relevant when expressing and analyzing correctness requirements of complex distributed algorithms and communication protocols, from the fundamental ones aimed to reach consensus in a network, to applications like information flow control and security protocols (secrecy, anonymity, fair exchange). Therefore, many approaches based on epistemic logics have been developed for the analysis of such protocols; some examples are the BAN logic [BAN96], the theory of function views [HS04], interpreted systems [FHMV95,HO05,RL06].

They allow for natural and effective representations of subtle effects of communication acts such as classified information leaking to attackers or, on the positive side, participants gaining the common knowledge that the protocol they were running meets its goal. But on the other hand, modeling protocols using epistemic-logic-based approaches requires a high degree of expertise and verification of functional properties is often very complex. The information updates generating the transitions between epistemic states are especially tedious to specify, because logics are geared to expressing properties rather than operational steps of a protocol.

The operational behaviour of protocols are, however, easily and conveniently specified in languages such as process algebras [BHR84,Mil80,AG99] and message sequence charts [CVB06]. Functional requirements such as liveness and safety are then easily verified by model checking applied on the underlying transition systems. Unfortunately, these standard and successful verification schemes use temporal logics that are not well-suited for expressing knowledge-related properties, therefore complex specialized solutions need to be sought in order to make process algebras suitable for the analysis of epistemic-flavoured properties like anonymity [SS96,COPT07]. See [HS04,EO06] for a more detailed comparison of epistemic-based vs. process-based protocol verification.

In this paper, we propose a framework that allows one to benefit the best of the two worlds, i.e., one can specify the behavior of a protocol in a process language and verify properties expressed in a logic with both temporal and epistemic operators. To achieve this, the key idea is to introduce explicit identities in our process language  $PAi$ . Then every action is annotated with a visibility range — i.e., a set of identities that are allowed to observe it and a “public appearance” — i.e., an alternative action that is observed by the identities outside the visibility range. This language is supported by an operational semantics generating annotated labeled transition systems (ALTSs), which are LTSs with, for every identity, an extra indistinguishability relation on states. These

relations model the uncertainties of the identities (typically principals in a protocol) about the current state, similar to the way uncertainties are represented in standard possible-world semantics for epistemic logics [FHMV95]. Thanks to the combination of transitions and indistinguishability relations, ALTSs naturally support verification of logic formulae containing both temporal and epistemic operators. We introduce a rich logic,  $E\overline{\mu}$  (epistemic  $\mu$ -calculus with past) and give it an interpretation on ALTSs.

Due to the explicit use of identities, our process algebra  $PAi$  allows a precise specification of the information hiding behaviour within protocols, and it is therefore more expressive and flexible than traditional process algebras. It is also more intuitive and more formal than epistemic logics, when it comes to modelling behaviour. Further on,  $E\overline{\mu}$  is more expressive than the usual temporal logics used in traditional protocol verification. The resulting model checking framework  $PAi+E\overline{\mu}$  soundly extends the traditional process-based and epistemic model checking settings.

**Related work** The fact that the two verification approaches, process algebraic and epistemic, are complementary and that they should ideally be combined has already been recognized in [HS04], where the aim is, just as here, to provide a framework in which both protocol specification and correctness criteria can be specified succinctly and intuitively (and the authors indeed put the two approaches in sharp contrast). However, [HS04] takes a totally different path towards its goals and develops a domain-theoretic notion of function views to define security protocols and their properties.

BAN-logic [BAN96] was designed for the analysis of authentication in security protocols and became very popular, but it is a known problem that a clear semantics, linking the high-level BAN-specification to runs of the protocol, is still missing. Also in other interesting recent work concerning Dynamic Epistemic Logic [GG97,Bal01,HMV04] with an operational flavor, it turns out that protocol specification is rather subtle and ad hoc. The same lack of a proper semantics can be seen even in tool-supported temporal epistemic approaches [RL06,HW02], where existing temporal specification languages are used, but the embedding of the epistemic aspects remains (for a large part) informal. We start from the other side - a process specification language with a formal semantics, and work towards properly integrating epistemic aspects.

Interpreted Systems [FHMV95,RL06,HO05] are close to the operational semantics of our process language. In fact, it is possible to translate ALTSs defined by our SOS rules to interpreted systems. Our key improvement is the introduction of a process specification language with a formal semantics, which enables the modelling of systems at a reasonable abstraction level.

The concept of indistinguishability used here bears resemblances to the data independence technique in [Bro01]. We consider runs of a protocol indistinguishable if they appear equal to a principal (as defined by the visibility range of actions and their public appearance). It is worthwhile to investigate an extension of our framework along the lines of [Bro01], by allowing the visibility range of actions to be dynamically updated.

Finally, concurrently with our work, efforts have been made towards the development of a rich language  $C^3$  [BKN06] and a powerful logic CPL [Kra06] for analyzing cryptographic protocols. That framework is comparable to ours, although specifically geared towards cryptography. The aim there is integrating a wide range of features, from deontic and spatial operators to probabilities, in one unified setting.  $C^3+CPL$  is therefore very expressive, but very complex and seems difficult to implement, while our simple language with an easy to grasp operational semantics can immediately serve as basis for a practical verification toolset. In fact, a prototype implementation already exists [PAi].

In [HO05], the notion of interpreted systems is used to model and verify anonymity. There, a system consists of a number of agents and the semantics of a system is given in terms of sequences of tuples comprising local states of individual agents. Afterwards, different notions of (probabilistic) anonymity are defined using their epistemic logic formulation. Our approach is related to and complements that one, by providing a way of verifying, on process-based specifications, anonymity notions as defined by [HO05].

**Overview** Section 2 introduces our generic process language for specifying protocols and a transition-system semantics for it. Section 3 defines our temporal epistemic logic  $E\overline{\mu}$  and defines how the formulas should be interpreted on the transition systems. Then we show that this construction does indeed bridge the gap between process-based and epistemic-logic-based approaches to protocol analysis, by proving that its projections on the two worlds are consistent with established definitions in the two worlds separately (Section 4). Section 5 shows an example and Section 7 concludes the paper and presents directions for future research.

## 2 $PAi$ : Syntax and Operational Semantics

In this section, we present the syntax and the operational semantics of a simple modeling language which we call process algebra with identities ( $PAi$ ).  $PAi$  has generic features, that can be adapted to match constructs of any classical operational modeling language (such as CCS [Mil80], CSP [BHR84] or Spi-Calculus [AG99]). It resembles Milner’s CCS, but we deviate from CCS in a few ways. Apart from adding identities, we use sequential composition instead of action prefixing (and thus, we also introduce a termination predicate), since this is very handy in writing protocol specifications. Also, we do not hide the result of a communication automatically and leave this, if at all desired, to the renaming function since the communicated message can be of relevance in the correctness specification of the protocol.

**$PAi$  : syntax** Let  $Act$  be a finite set of *action names* which will be ranged over by  $a, b, a_0, ?a, !a, \dots$ , and let  $Id$  be a finite set of *identities* typically denoted by  $i, j, \dots i_1, i_2, \dots$ . We designate an action  $\tau \in Act$  to denote the internal (silent) action; in addition to its common process-algebraic meaning, an internal action here represents a message that offers no new information to the observer principal. Question mark and exclamation mark (preceding actions) represent the receiving and the sending parts of a communication, respectively, and an action without such marks is the outcome of the communication.

$$\begin{aligned} Proc &::= 0 \mid D \mid Proc; Proc \mid Proc + Proc \mid Proc \parallel Proc \\ D &::= (J)\alpha \end{aligned}$$

0 denotes inaction (the process that has terminated).  $d = (J)\alpha \in D$  denotes a decorated action and has the following intuitive meaning: action  $\alpha \in Act$  is taken and is visible to principals  $i \in J \subseteq Id$ , while principals  $j \notin J$  observe  $\rho(a)$  being taken, where  $\rho : Act \rightarrow Act$  is a global renaming function, which assigns to every action its “public appearance”. The renaming function  $\rho$  should be defined by the specifier of a protocol but we assume that  $\rho(\tau)$  is always defined to be  $\tau$ . For any other action  $a$ , if  $\rho(a) = \tau$ , then  $(J)a$  becomes unobservable to the principals not in  $J$ . The combination of identity annotations on actions and the action renaming provides different views on the behavior of the system, according to different principals. Modeling passive observation of a system by hiding parts of it to specific principals is already done in the literature [SS96], but we will generate the views for all principals simultaneously. This enables talking about properties such as “ $i$  knows that  $j$  knows that  $k$  has communicated message  $a$ ”.  $Proc; Proc$  denotes sequential composition,  $Proc + Proc$  denotes nondeterministic choice, and  $Proc \parallel Proc$  denotes parallel composition.

*Example 1.* As a small example of a  $PAi$  process, take  $P = (1)a; (1, 2)d + (1)b + (1)c$ , with the renaming function  $\rho(a) = \rho(b) = \rho(c) = dummy$  where *dummy* is a dummy basic action and over the identity set  $Id = \{1, 2\}$ .  $P$  denotes the process that executes one of the actions  $a, b, c$ , but only principal 1 is aware of the exact action taking place. 1 could be the principal making a choice between actions  $a, b$  and  $c$ , and 2 could be an observer who only notices that a choice has been made, but not what the outcome was. This would be a process-style formalization of the *private communication* from epistemic modeling, where a party learns something while other parties watch and learn that the party learned something, but not precisely what. After the first step, the process terminates or, if the first step was  $a$ , continues with the execution of  $d$ . Since principal 2 is allowed to observe the execution of  $d$ , she may now conclude that the first step

$$\begin{array}{c}
\text{(0)} \frac{}{(0, \pi)\checkmark} \quad \text{(a)} \frac{}{(d, \pi) \xrightarrow{d} (0, \pi \frown d)} \\
\text{(s0)} \frac{(x_0, \pi) \xrightarrow{d} (y_0, \pi')}{(x_0; x_1, \pi) \xrightarrow{d} (y_0; x_1, \pi')} \quad \text{(s1)} \frac{(x_0, \pi)\checkmark \quad (x_1, \pi) \xrightarrow{d} (y_1, \pi')}{(x_0; x_1, \pi) \xrightarrow{d} (y_1, \pi')} \quad \text{(s2)} \frac{(x_0, \pi)\checkmark \quad (x_1, \pi')\checkmark}{(x_0; x_1, \pi'')\checkmark} \\
\text{(n0)} \frac{(x_0, \pi) \xrightarrow{d} (y_0, \pi')}{(x_0 + x_1, \pi) \xrightarrow{d} (y_0, \pi')} \quad \text{(n2)} \frac{(x_0, \pi)\checkmark}{(x_0 + x_1, \pi')\checkmark} \quad \text{(p0)} \frac{(x_0, \pi) \xrightarrow{d} (y_0, \pi')}{(x_0 \parallel x_1, \pi) \xrightarrow{d} (y_0 \parallel x_1, \pi')} \\
\text{(p2)} \frac{(x_0, \pi)\checkmark \quad (x_1, \pi')\checkmark}{(x_0 \parallel x_1, \pi'')\checkmark} \quad \text{(p3)} \frac{(x_0, \pi) \xrightarrow{(J)?a} (y_0, \pi') \quad (x_1, \pi) \xrightarrow{(J)!a} (y_1, \pi'')}{(x_0 \parallel x_1, \pi) \xrightarrow{(J \cup J')a} (y_0 \parallel y_1, \pi \frown (J \cup J')a)} \\
\text{---} \\
\text{(= refl)} \frac{}{\pi \stackrel{i}{=} \pi} \quad \text{(= } \rho 0) \frac{\pi \stackrel{i}{=} \pi' \quad a = b \quad i \in J \cap J'}{\pi \frown (J)a \stackrel{i}{=} \pi' \frown (J')b} \\
\text{(= } \rho 1) \frac{\pi \stackrel{i}{=} \pi' \quad \rho(a) = \rho(b) \quad i \notin J' \cup J}{\pi \frown (J)a \stackrel{i}{=} \pi' \frown (J')b} \quad \text{(= } \rho 2) \frac{\pi \stackrel{i}{=} \pi' \quad a = \rho(b) \quad i \in J \setminus J'}{\pi \frown (J)a \stackrel{i}{=} \pi' \frown (J')b} \\
\text{(= } \tau 0) \frac{\pi \stackrel{i}{=} \pi' \quad i \notin J \quad \rho(a) = \tau}{\pi \frown (J)a \stackrel{i}{=} \pi'} \quad \text{(= } \tau 2) \frac{\pi \stackrel{i}{=} \pi'}{\pi \frown (J)\tau \stackrel{i}{=} \pi'} \\
\text{---} \\
\text{(strip)} \frac{(x, \pi) \xrightarrow{(J)a} (y, \pi')}{(x, \pi) \xrightarrow{a} (y, \pi')} \quad \text{(I)} \frac{\pi_0 \stackrel{i}{=} \pi_1}{(x_0, \pi_0) \cdot \cdot \cdot (x_1, \pi_1)}
\end{array}$$

Fig. 1. SOS of  $PAi$

must have been  $a$ , although 2 was not actually allowed to observe the  $a$ . This is exactly the type of information leaks that we aim at capturing with our verification framework. The rightmost model in Figure 3 is the state space generated from this process specification, using the semantic rules introduced in the next section. The fact that 2 is not aware which first step has been taken is represented by 2 not being able to distinguish between the three possible destination states. By tuning the  $\rho$  function, we can get various levels of action visibility for the observing party 2. Namely, if we define  $\rho(a) = \rho(b) = \rho(c) = \tau$ , then  $P$  describes the process where principal 1 is the target of a *secret communication*, in the sense that 1 executes the choice  $a + b + c$ , while 2 sees all four states — the one before the choice and the three after the choice — as one state. To the other extreme, if  $\rho$  is the identity function  $\rho(a) = a, \rho(b) = b, \rho(c) = c$ , then 1 and 2 are both fully aware of the actual step taken in the system (*public announcement*).

**$PAi$  : operational semantics** We introduce the notion of *Annotated Labeled Transition Systems* (ALTS) as labeled transition systems extended with annotations that denote when two states are deemed indistinguishable from the viewpoint of a principal, based on the actions taken so far. This is determined by the information that a principal receives in the course of protocol execution. For instance, if during a particular communication transition a principal does not receive any new information (since the principal's identity is not in the visibility range of the action and the action is renamed to  $\tau$ ), the principal would consider the source and the target states of such transitions indistinguishable.

**Definition 1 (ALTS).** *Given the set  $Act$ , an ALTS is a 5-tuple  $\langle St, \rightarrow, \checkmark, I, s_0 \rangle$ , where  $St$  is the set of operational states,  $\rightarrow \subseteq St \times Act \times St$  is the transition relation,  $\checkmark \subseteq St$  is the termination predicate,  $I \subseteq St \times Id \times St$  is the indistinguishability relation and  $s_0$  is the initial state.*

For readability, we denote statements  $(s, l, s') \in \rightarrow$ ,  $s \in \checkmark$  and  $(s, i, s') \in I$  by  $s \xrightarrow{l} s'$ ,  $s\checkmark$  and  $s \overset{i}{\cdots} s'$ , respectively, for each  $s, s' \in St$ ,  $l \in Act$  and  $i \in Id$ .

In the above definition, the transition relation  $\rightarrow$  has exactly the same role and meaning as in the standard notion of LTS. Formula  $s\checkmark$  means that in state  $s$  it is possible to terminate. Expression  $s_0 \overset{i}{\cdots} s_1$  denotes that the principal with identity  $i$  cannot distinguish  $s_0$  from  $s_1$  since both  $s_0$  and  $s_1$  are reachable through paths that look identical as far as as principal  $i$  can observe and distinguish. It is desirable for  $\overset{i}{\cdots}$  to be an equivalence relation for each  $i \in Id$  since this leads to a natural representation of knowledge (i.e., S5 Kripke models in modal logic, see [FHMV95]).

In Figure 1, we associate ALTS's to  $PAi$  processes by means of a semantics in the SOS style of [Plo04]. The operational state of  $PAi$  is a pair  $(p, \pi)$  where  $p \in Proc$  is a  $PAi$  process and  $\pi$  is a finite sequence of decorated actions recording the perception of the process gathered so far. First we define auxiliary relations  $\overset{d}{\Rightarrow} \subseteq St \times St$  and  $\overset{i}{\equiv} \subseteq D^* \times D^*$  for each decorated action  $d$  and identity  $i$ . Transition relation  $\overset{d}{\Rightarrow}$  defines transitions among operational states labeled with decorated action  $d$  and  $\overset{i}{\equiv}$  defines when two traces are deemed indistinguishable by principal  $i$ . In the deduction rule (**strip**), we strip off the extra information on the labels (concerning the visibility range) and apply encapsulation (leaving out individual send and receive actions) and obtain the transition relation  $\rightarrow$ .<sup>3</sup> Deduction rule (**I**) lifts the concept of indistinguishability from traces to operational states. For brevity, we omitted symmetric rules (**n1**), (**n3**), (**p1**), (**p4**), (**= $\rho$ 3**), (**= $\tau$ 1**), and (**= $\tau$ 3**). Termination of a process is orthogonal to its past history, so we use different meta-variables for the traces in the premises and the conclusion of rules (**s2**), (**n2**), and (**p2**). The transition relation  $\Rightarrow$  and indistinguishability relation  $\cdots$  are the sets of all closed statements provable using the deduction rules (plus their symmetric versions) from Figure 1. The semantics of a process  $p$  is defined by the ALTS with pairs of processes and decorated traces as states,  $\rightarrow$  as transition relation,  $\checkmark$  as termination relation,  $\cdots$  as indistinguishability relation, and  $(p, [])$  as the initial state, where  $[]$  denotes the empty sequence of decorated actions. The following lemma states that  $\overset{i}{\cdots}$  is an equivalence relation.<sup>4</sup>

**Lemma 1.** *Relation  $\overset{i}{\cdots}$  is an equivalence relation.*

*Proof.* Once we prove that  $\overset{i}{\equiv}$  is an equivalence, the lemma follows immediately. Thus, we proceed with the proof of equivalence for  $\overset{i}{\equiv}$ .

Reflexivity follows vacuously from (**= refl**).

For symmetry, the proof goes by an induction on the sum of the lengths of sequences  $\pi_0$  and  $\pi_1$  in  $\pi_0 \overset{i}{\equiv} \pi_1$ . The base case, where  $\pi_0 = \pi_1 = []$  follows vacuously from (**refl**). For the induction step, we distinguish different cases based on the last deduction rule in the proof. The rest of the proof is straightforward but is given for the sake of completeness.

(**= refl**) Vacuous.

(**= $\rho$ 0**)  $\pi_0 = \pi'_0 \frown (J_0)a_0$  and  $\pi_1 = \pi'_1 \frown (J_1)a_1$  for some  $\pi'_0, \pi'_1, (J_0)a_0$ , and  $(J_1)a_1$  such that  $\pi'_0 \overset{i}{\equiv} \pi'_1$ ,  $i \in J_0 \cap J_1$ , and  $a_0 = a_1$ . It follows from the induction hypothesis that  $\pi'_1 \overset{i}{\equiv} \pi'_0$  and by applying the deduction rule (**= $\rho$ 0**), we obtain  $\pi_1 \overset{i}{\equiv} \pi_0$ .

(**= $\rho$ 1**) Similar to above.

(**= $\rho$ 2**)  $\pi_0 = \pi'_0 \frown (J_0)a_0$  and  $\pi_1 = \pi'_1 \frown (J_1)a_1$  for some  $\pi'_0, \pi'_1, (J_0)a_0$ , and  $(J_1)a_1$  such that  $\pi'_0 \overset{i}{\equiv} \pi'_1$ ,  $i \in J_0 \setminus J_1$ , and  $a_0 = \rho(a_1)$ . Then, it follows from the induction hypothesis that  $\pi'_1 \overset{i}{\equiv} \pi'_0$  and by applying (**= $\rho$ 3**), we have  $\pi_1 \overset{i}{\equiv} \pi_0$ .

(**= $\rho$ 3**) Symmetric to the above case.

<sup>3</sup> We could have used an explicit encapsulation (restriction) operator but decided not to do so to keep the presentation simple.

<sup>4</sup> We intentionally did not add deduction rules to enforce symmetry and transitivity of  $\overset{i}{\equiv}$  explicitly in order to preserve the inductive structure of our SOS specification.

- (=  $\tau\mathbf{0}$ )  $\pi_0 = \pi'_0 \frown (J_0)a_0$  for some  $\pi'_0$  and  $(J_0)a_0$  such that  $\pi'_0 \stackrel{i}{=} \pi_1$ ,  $i \notin J_0$ , and  $\rho(a_0) = \tau$ . Then, it follows from the induction hypothesis that  $\pi_1 \stackrel{i}{=} \pi'_0$  and by applying (=  $\tau\mathbf{1}$ ), we have  $\pi_1 \stackrel{i}{=} \pi_0$ .
- (=  $\tau\mathbf{1}$ ) Symmetric to the above case.
- (=  $\tau\mathbf{2}$ )  $\pi_0 = \pi'_0 \frown (J_0)\tau$  for some  $\pi'_0$  and  $J_0$  such that  $\pi'_0 \stackrel{i}{=} \pi_1$ . Then, it follows from the induction hypothesis that  $\pi_1 \stackrel{i}{=} \pi'_0$  and by applying (=  $\tau\mathbf{3}$ ), we have  $\pi_1 \stackrel{i}{=} \pi_0$ .
- (=  $\tau\mathbf{3}$ ) Symmetric to the above case.

For transitivity, we use an induction on the sum of the sizes of  $\pi_0$ ,  $\pi_1$  and  $\pi_2$  in  $\pi_0 \stackrel{i}{=} \pi_1$  and  $\pi_1 \stackrel{i}{=} \pi_2$ . The induction basis holds vacuously. We distinguish cases based on the last deduction rule used to prove each of  $\pi_0 \stackrel{i}{=} \pi_1$  and  $\pi_1 \stackrel{i}{=} \pi_2$  (due to symmetry, it suffices to consider the cases in one particular order and the other case can be obtained from the one considered by symmetry). Most of the cases are straightforward but all cases are checked for the sake of completeness.

- (= **refl**), (**x**) The case is vacuous if one of the deduction rules is (**refl**).
- (=  $\rho\mathbf{0}$ ), (=  $\rho\mathbf{0}$ ) Then,  $\pi_0 = \pi'_0 \frown (J_0)a_0$ ,  $\pi_1 = \pi'_1 \frown (J_1)a_1$ ,  $\pi_2 = \pi'_2 \frown (J_2)a_2$  for some  $\pi'_0, \pi'_1, \pi'_2, J_0, J_1, J_2, a_0, a_1$  and  $a_2$  such that  $\pi'_0 \stackrel{i}{=} \pi'_1, \pi'_1 \stackrel{i}{=} \pi'_2, i \in J_0 \cap J_1, i \in J_1 \cap J_2, a_0 = a_1$  and  $a_1 = a_2$ . It follows from the induction hypothesis that  $\pi'_0 \stackrel{i}{=} \pi'_2$  and from  $i \in J_0 \cap J_1$  and  $i \in J_1 \cap J_2$  that  $i \in J_0 \cap J_2$  and from  $a_0 = a_1$  and  $a_1 = a_2$  that  $a_0 = a_2$ . By applying (=  $\rho\mathbf{0}$ ) on these hypothesis, we have  $\pi_0 \stackrel{i}{=} \pi_2$  which was to be shown.
- (=  $\rho\mathbf{0}$ ), (=  $\rho\mathbf{1}$ ) Then,  $\pi_0 = \pi'_0 \frown (J_0)a_0$ ,  $\pi_1 = \pi'_1 \frown (J_1)a_1$ ,  $\pi_2 = \pi'_2 \frown (J_2)a_2$  for some  $\pi'_0, \pi'_1, \pi'_2, J_0, J_1, J_2, a_0, a_1$  and  $a_2$  such that  $\pi'_0 \stackrel{i}{=} \pi'_1, \pi'_1 \stackrel{i}{=} \pi'_2, i \in J_0 \cap J_1, i \notin J_1 \cup J_2, a_0 = a_1$  and  $\rho(a_1) = \rho(a_2)$ . Impossible since from  $i \in J_0 \cap J_1$  it follows that  $i \in J_1$  and this contradicts  $i \notin J_1 \cup J_2$ . From this point on, we do not repeat the structure of  $\pi_0, \pi_1$ , and  $\pi_2$ ; when needed and unless we state otherwise, we assume the decomposition mentioned in the above items.
- (=  $\rho\mathbf{0}$ ), (=  $\rho\mathbf{2}$ ) Then,  $\pi'_0 \stackrel{i}{=} \pi'_1, \pi'_1 \stackrel{i}{=} \pi'_2, i \in J_0 \cap J_1, i \in J_1 \setminus J_2, a_0 = a_1$  and  $a_1 = \rho(a_2)$ . It follows from the induction hypothesis that  $\pi'_0 \stackrel{i}{=} \pi'_2$  and from  $i \in J_0 \cap J_1$  and  $i \in J_1 \setminus J_2$  that  $i \in J_0 \setminus J_2$  and from  $a_0 = a_1$  and  $a_1 = \rho(a_2)$  that  $a_0 = \rho(a_2)$ . By applying (=  $\rho\mathbf{0}$ ) on these hypothesis, we have  $\pi_0 \stackrel{i}{=} \pi_2$  which was to be shown.
- (=  $\rho\mathbf{0}$ ), (=  $\rho\mathbf{3}$ ) Then,  $\pi'_0 \stackrel{i}{=} \pi'_1, \pi'_1 \stackrel{i}{=} \pi'_2, i \in J_0 \cap J_1, i \notin J_2 \setminus J_1, a_0 = a_1$  and  $\rho(a_1) = a_2$ . Impossible since it follows from  $i \in J_0 \cap J_1$  that  $i \in J_1$  and from  $i \in J_2 \setminus J_1$  that  $i \notin J_1$ .
- (=  $\rho\mathbf{0}$ ), (=  $\tau\mathbf{0}$ ) Then,  $\pi'_0 \stackrel{i}{=} \pi'_1, \pi'_1 \stackrel{i}{=} \pi_2, i \in J_0 \cap J_1, i \notin J_1, a_0 = a_1$  and  $\rho(a_1) = \tau$ . Impossible since it follows from  $i \in J_0 \cap J_1$  that  $i \in J_1$  and this contradicts  $i \notin J_1$ .
- (**x**), (=  $\tau\mathbf{1}$ ) (Assume that the last deduction rule used to derive  $\pi_1 \stackrel{i}{=} \pi_2$  is (=  $\tau\mathbf{1}$ ); the last deduction rule used to derive  $\pi_0 \stackrel{i}{=} \pi_1$  is immaterial.) Then, Then,  $\pi_0 \stackrel{i}{=} \pi_1, \pi_1 \stackrel{i}{=} \pi'_2, i \notin J_2$ , and  $\rho(a_2) = \tau$ . It follows from the induction hypothesis that  $\pi_0 \stackrel{i}{=} \pi'_2$ , we already have  $i \notin J_2$  and  $\rho(a_2) = \tau$ . By applying (=  $\tau\mathbf{1}$ ) on these hypothesis, we have  $\pi_0 \stackrel{i}{=} \pi_2$  which was to be shown.
- (=  $\rho\mathbf{0}$ ), (=  $\tau\mathbf{2}$ ) Then,  $\pi_1 = \pi'_1 \frown (J_1)\tau$ , for some  $\pi'_1$  and  $J_0$  such that  $\pi'_0 \stackrel{i}{=} \pi'_1, \pi'_1 \stackrel{i}{=} \pi_2, i \in J_0 \cap J_1, a_0 = \tau$ . It follows from the induction hypothesis that  $\pi'_0 \stackrel{i}{=} \pi_2$  and since  $a_0 = \tau$ , by applying (=  $\tau\mathbf{2}$ ) on these hypotheses, we have  $\pi_0 \stackrel{i}{=} \pi_2$  which was to be shown.
- (**x**), (=  $\tau\mathbf{3}$ ) Then  $\pi_2 = \pi'_2 \frown (J_2)\tau$  and  $\pi_0 \stackrel{i}{=} \pi_1$  and  $\pi_1 \stackrel{i}{=} \pi'_2$ . It follows from the induction hypothesis that  $\pi_0 \stackrel{i}{=} \pi'_2$  and from by applying (=  $\tau\mathbf{3}$ ), we have  $\pi_0 \stackrel{i}{=} \pi_2$  which was to be shown. Note that we do not need to consider the symmetric cases, e.g., (=  $\rho\mathbf{1}$ ), (=  $\rho\mathbf{0}$ ) since they follow from symmetry and the other cases due to symmetric rules, e.g., (=  $\rho\mathbf{0}$ ) and (=  $\rho\mathbf{2}$ ).
- (=  $\rho\mathbf{1}$ ), (=  $\rho\mathbf{1}$ ) Then,  $\pi'_0 \stackrel{i}{=} \pi'_1, \pi'_1 \stackrel{i}{=} \pi'_2, i \notin J_0 \cup J_1, i \notin J_1 \cup J_2, \rho(a_0) = \rho(a_1)$  and  $\rho(a_1) = \rho(a_2)$ . Then, by the induction hypothesis, we have  $\pi'_0 \stackrel{i}{=} \pi'_2$  and by  $i \notin J_0 \cup J_1$  and  $i \notin J_1 \cup J_2$ , we have  $i \notin J_0 \cup J_2$  and by  $\rho(a_0) = \rho(a_1)$  and  $\rho(a_1) = \rho(a_2)$ , we have  $\rho(a_0) = \rho(a_2)$ . Thus, it follows from applying (=  $\rho\mathbf{1}$ ) on these hypothesis that  $\pi_0 \stackrel{i}{=} \pi_2$ .
- (=  $\rho\mathbf{1}$ ), (=  $\rho\mathbf{2}$ ) Then,  $\pi'_0 \stackrel{i}{=} \pi'_1, \pi'_1 \stackrel{i}{=} \pi'_2, i \notin J_0 \cup J_1, i \in J_1 \setminus J_2, \rho(a_0) = \rho(a_1)$  and  $a_1 = \rho(a_2)$ . Impossible since it follows from  $i \notin J_0 \cup J_1$  that  $i \notin J_1$  and from  $i \in J_1 \setminus J_2$  that  $i \in J_1$ .

- (=  $\rho\mathbf{1}$ ), (=  $\rho\mathbf{3}$ ) Then,  $\pi'_0 \stackrel{i}{=} \pi'_1$ ,  $\pi'_1 \stackrel{i}{=} \pi'_2$ ,  $i \notin J_0 \cup J_1$ ,  $i \in J_2 \setminus J_1$ ,  $\rho(a_0) = \rho(a_1)$  and  $a_2 = \rho(a_1)$ . Then, it follows from the induction hypothesis that  $\pi'_0 \stackrel{i}{=} \pi'_1$ ; it also follows from  $i \notin J_0 \cup J_1$  and  $i \in J_2 \setminus J_1$  that  $i \in J_2 \setminus J_0$ ; moreover, from  $\rho(a_0) = \rho(a_1)$  and  $a_2 = \rho(a_1)$  that  $\rho(a_0) = a_2$ . Thus, by applying (=  $\rho\mathbf{3}$ ), we have  $\pi_0 \stackrel{i}{=} \pi_2$ .
- (=  $\rho\mathbf{1}$ ), (=  $\tau\mathbf{0}$ ) It follows from the induction hypothesis that  $\pi'_0 \stackrel{i}{=} \pi_2$ ; we also have that  $\rho(a_0) = \rho(a_1) = \tau$  and  $i \notin J_0 \cup J_1$ , thus  $i \notin J_1$  and by applying ( $\tau\mathbf{0}$ ), we derive  $\pi_0 \stackrel{i}{=} \pi_2$ .
- (=  $\rho\mathbf{1}$ ), (=  $\tau\mathbf{2}$ ) Then,  $a = b = \tau$  and it follows from the induction hypothesis that  $\pi'_0 \stackrel{i}{=} \pi_2$ . Thus, by applying (=  $\tau\mathbf{2}$ ), we have that  $\pi_0 \stackrel{i}{=} \pi_2$ .
- (=  $\rho\mathbf{2}$ ), (=  $\rho\mathbf{2}$ ) Impossible since  $i \in J_0 \setminus J_1$  and  $i \in J_1 \setminus J_2$  cannot hold both.
- (=  $\rho\mathbf{2}$ ), (=  $\rho\mathbf{3}$ )  $i \in J_0 \setminus J_1$ ,  $i \in J_2 \setminus J_1$ ,  $a_0 = \rho(a_1)$  and  $a_2 = \rho(a_1)$ . Hence, (by the induction hypothesis) we have that  $\pi'_0 \stackrel{i}{=} \pi'_2$  and (from the hypotheses of the rules) that  $i \in J_0 \cap J_1$  and  $a = b$ . Thus, by applying (=  $\rho\mathbf{0}$ ), we have that  $\pi_0 \stackrel{i}{=} \pi_2$ .
- (=  $\rho\mathbf{2}$ ), (=  $\tau\mathbf{0}$ ,  $\mathbf{2}$ ) Then,  $a_0 = \rho(a_1) = \rho(\tau) = \tau$  and it follows from the induction hypothesis that  $\pi'_0 \stackrel{i}{=} \pi_2$ . Thus, by applying (=  $\tau\mathbf{2}$ ), we have that  $\pi_0 \stackrel{i}{=} \pi_2$ .
- (=  $\rho\mathbf{3}$ ), (=  $\rho\mathbf{3}$ ) Impossible since  $i \in J_1 \setminus J_0$  and  $i \in J_2 \setminus J_1$  cannot hold both.
- (=  $\rho\mathbf{3}$ ), (=  $\tau\mathbf{0}$ ) Impossible since  $i \in J_1 \setminus J_0$  and  $i \notin J_1$  cannot hold both.
- (=  $\rho\mathbf{3}$ ), (=  $\tau\mathbf{2}$ ) Then  $\rho(a_0) = a_1 = \tau$  and  $i \in J_1 \setminus J_0$  and hence,  $i \notin J_0$ . It follows from the induction hypothesis that  $\pi'_0 \stackrel{i}{=} \pi_2$  and by applying (=  $\tau\mathbf{0}$ ), we have that  $\pi_0 \stackrel{i}{=} \pi_2$ .
- (=  $\tau\mathbf{0}$ ), ( $\mathbf{x}$ ) It follows from the induction hypothesis that  $\pi'_0 \stackrel{i}{=} \pi_2$ . Since,  $i \notin J_0$ , by applying (=  $\tau\mathbf{0}$ ), we have that  $\pi_0 \stackrel{i}{=} \pi_2$ .
- (=  $\tau\mathbf{1}$ ), (=  $\tau\mathbf{2}$ ) It follows from  $\pi_0 \stackrel{i}{=} \pi'_1$ ,  $\pi'_1 \stackrel{i}{=} \pi_2$  and the induction hypothesis that  $\pi_0 \stackrel{i}{=} \pi_2$ .
- (=  $\tau\mathbf{2}$ ), (=  $\tau\mathbf{2}$ ) It follows from  $\pi'_0 \stackrel{i}{=} \pi_1$ ,  $\pi_1 \stackrel{i}{=} \pi_2$  and the induction hypothesis that  $\pi'_0 \stackrel{i}{=} \pi_2$ . By applying deduction rule (=  $\tau\mathbf{2}$ ), we have that  $\pi_0 \stackrel{i}{=} \pi_2$ .
- (=  $\tau\mathbf{3}$ ), (=  $\tau\mathbf{3}$ ) Symmetric to the above case.

□

### 3 An epistemic mu-calculus

We introduce an epistemic mu-calculus with past ( $E\bar{\mu}$ ) which combines temporal, epistemic, and fixed point constructs. We give our logic an interpretation on the operational model introduced in Section 2.

**Syntax** The syntax of  $E\bar{\mu}$  is given by the following grammar:

$$\phi ::= \top \mid X \mid \phi \wedge \phi \mid \neg\phi \mid \langle a \rangle \phi \mid \langle \bar{a} \rangle \phi \mid K_i \phi \mid \nu X. \phi(X)$$

(if  $X$  occurs only positively in  $\phi$ ),

where  $a$  ranges over the set of actions ( $a \in \mathcal{Act}$ ). Then  $\langle a \rangle \phi$  stands for “after some execution of  $a$ ,  $\phi$  holds”;  $\langle \bar{a} \rangle \phi$  has the same intuition as  $\langle a \rangle \phi$ , except that it refers to the *past*, i.e., there is a state in which  $\phi$  holds and from which it is possible to take an  $a$ -step to the current state.  $K_i \phi$  should be read as “principal  $i$  knows that  $\phi$  holds”. The greatest fixed point operator  $\nu X. \phi(X)$  is used to define recursive concepts. It intuitively means that the current state is in the largest set  $X$  of states that satisfy  $\phi(X)$ . (Here  $X$  is a variable ranging over propositional formulas, which can be identified by the sets of states in which such a formula is true. This is made formal by introducing valuations, but we leave this correspondence informal here.) For convenience, we define and use the following abbreviations for commonly used logical formulae:

---

$[a]\phi$	i.e., $\neg\langle a\rangle\neg\phi$ and intuitively means that after <i>all</i> $a$ -transitions, $\phi$ holds.
$\mu X.\phi(X)$	(with $X$ occurring positively in $\phi$ ) is the least fixed point operator, which is defined by $\neg\nu X.\neg\phi(\neg X)$ ( $X$ also occurs positively in $\neg\phi$ ). The current state is in the smallest set of states satisfying $\phi(X)$ .
$\langle\cdot\rangle\phi$	(similarly, $\langle\bar{\cdot}\rangle\phi$ ) stands for $\bigvee_{a\in Act}\langle a\rangle\phi$ ( $\bigvee_{a\in Act}\langle\bar{a}\rangle\phi$ ), which is by itself an abbreviation for a finite number of disjunctions. Intuitively, it means that after (before) <i>some</i> transition $\phi$ holds.
$\overset{r}{\langle} a$	(similarly, $a^{\uparrow}$ ) is an abbreviation for $\mu X.\langle a\rangle\top \vee \langle x\rangle.X$ (or $\mu X.\langle\bar{a}\rangle\top \vee \langle\bar{x}\rangle.X$ ). So, it is possible to reach a state in the future where an $a$ -transition is possible (go back to a state in the past that results from an $a$ -transition).
$[\cdot^*]\phi$	(similarly, $[\bar{\cdot}^*]\phi$ ) is an abbreviation for $\mu X.\phi \vee [\cdot]X$ (or $\mu X.\phi \vee [\bar{\cdot}]X$ ). The intuition behind this abbreviation is that all future paths will (paths in the past) lead to a state, in which there is a state satisfying $\phi$ . ( $\langle\cdot^*\rangle\phi$ and $\langle\bar{\cdot}^*\rangle\phi$ are defined accordingly.)
$C_J\phi$	stands for $\nu X.(\bigwedge_{i\in J} K_i(X \wedge \phi))$ [FHMV95], meaning: “it is common knowledge among the principals in the set $J$ that $\phi$ holds”.

---

Common knowledge is a very powerful construction, expressing that agents in  $J$  not only know that  $\phi$  holds, but also that all agents in  $J$  know that  $\phi$  holds, and that all agents in  $J$  know that all agents in  $J$  know that  $\phi$  holds, and so on. This property has so far not been amenable to specification and verification with standard operational techniques, while it is in fact very interesting, particularly for protocols where trust is an issue. Common knowledge can express, for instance, that participants in a multiparty fair exchange protocol trust each other and the protocol they are running.

Let  $E\bar{\mu}$ -forms denote the set of  $E\bar{\mu}$  formulas.

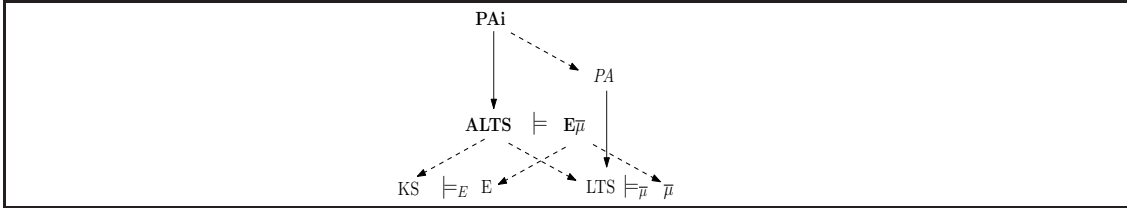
**Interpreting  $E\bar{\mu}$  formulas on ALTSs** We now define what it means for a formula  $\phi \in E\bar{\mu}$ -forms to be satisfied in the ALTS  $A$ .

**Definition 2 (satisfaction).** *Let  $A = \langle S, \rightarrow, \surd, I, s_0 \rangle$  be an ALTS. The satisfaction relation  $\models$  for formulas  $\phi \in E\bar{\mu}$ -forms is defined inductively as follows:*

$A, s \models \top$	<i>iff</i>	<i>true</i>
$A, s \models \phi_1 \wedge \phi_2$	<i>iff</i>	$A, s \models \phi_1$ and $A, s \models \phi_2$
$A, s \models \neg\phi$	<i>iff</i>	$A, s \models \phi$ is not true
$A, s \models \langle a\rangle\phi$	<i>iff</i>	there is an $s' \in S$ s.t. $s \xrightarrow{a} s'$ and $A, s' \models \phi$
$A, s \models \langle\bar{a}\rangle\phi$	<i>iff</i>	there is an $s' \in S$ s.t. $s' \xrightarrow{\bar{a}} s$ and $A, s' \models \phi$
$A, s \models K_i\phi$	<i>iff</i>	for all reachable $s' \in S$ s.t. $s \cdot^i \cdot s' : A, s' \models \phi$
$A, s \models \nu X.\phi(X)$	<i>iff</i>	$s \in \bigcup\{S' \subseteq S \mid \forall s' \in S'. A, s' \models \phi(X := S')\}$

$A$  satisfies a formula  $\phi$ , denoted  $A \models \phi$ , if  $s_0 \models \phi$ .

The most noticeable of the rules above is the one for  $K_i\phi$ . It expresses the fact that  $i$  knows  $\phi$  if  $\phi$  holds in all states considered possible by  $i$  when residing in  $s$ , that is in all states belonging to the  $\cdot^i \cdot$  equivalence class of  $s$ . The semantic rules in the previous section constructed this relation based on what  $i$  was allowed to observe from the run of the protocol. The intention behind the formula  $K_i\phi$  is not to check what  $i$  learned in terms of explicit information the principal received (e.g., as contents of some message), but what  $i$  learned through observation. Observation (partial observation) of what actually happens, can reduce a principal’s uncertainties and thereby ‘leak’ information. Particularly, if principles are familiar with the protocol, they may derive from certain actions taking place, that the previous action must have been a particular one, even if they did not



**Fig. 2.** Projecting into process-theoretic domain and epistemic domain. A dashed arrow  $x \dashrightarrow y$  means that  $x$  is an extension of  $y$ . The arrow  $x \rightarrow y$  means  $y$  is the semantic model of  $x$ . The links between ALTS, LTS, KS,  $E\bar{\mu}$ ,  $\bar{\mu}$ , E are discussed in this paper. The connection with the process languages  $PAi$  and  $PA$  (a pure process theoretic formalism) is explained in the appendix.

know it before. This is the case in the example depicted in Figure 3, where principal 2 learns from observation of action  $d$ , that the choice made before must have been  $a$ . More exactly, sequences of actions which are not properly protected by the visibility restrictions  $\rho$  may lead to a refinement of the  $\cdot^i$  class which is sufficient for  $i$  to distinguish between a state where agent’s  $j$  secret key is 100 and a state where agent  $j$ ’s secret key is 200, even if  $i$  never participated in a direct communication over  $j$ ’s key. This process of learning by the refinement of the indistinguishability relations along the traces is captured in the definition of  $A, s \models K_i\phi$ . Our logic satisfies the standard axioms for a logic of knowledge:

**Theorem 1.** *The so-called S5 axioms (cf. [FHMV95, p.59]) hold in  $E\bar{\mu}$ :*

$$\begin{array}{ll} \mathbf{K} : K_i\phi \wedge K_i(\phi \rightarrow \psi) \rightarrow K_i\psi & \mathbf{4} : K_i\phi \rightarrow K_iK_i\phi \quad (\text{positive introspection}) \\ \mathbf{T} : K_i\phi \rightarrow \phi \quad (\text{reflexivity}) & \mathbf{5} : \neg K_i\phi \rightarrow K_i\neg K_i\phi \quad (\text{negative introspection}) \end{array}$$

*Proof.* S5 is a sound and complete axiomatization with respect to Kripke models where the indistinguishability relations are equivalence relations (cf. e.g. [FHMV95, Theorem 3.1.5(c)]). The relations  $\cdot^i$  are equivalence relations (Lemma 1), so in particular, the S5 axioms hold.  $\square$

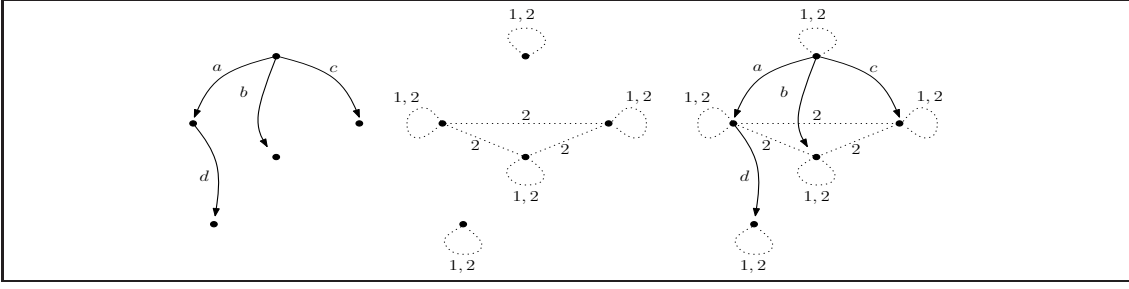
The definition of satisfaction provides a model checking algorithm, that will be decidable on the finite trees generated by the semantics of our  $PAi$ . Since the  $E\bar{\mu}$  satisfaction relation on ALTSs rests on classically accepted definitions for similar but less expressive models, we expect that it should be possible to reuse and extend existing efficient model checking tools.

An interesting and non-trivial question is to find a behavioral equivalence that is characterized by  $E\bar{\mu}$ . We expect the answer to be some notion of bisimilarity that considers both  $\xrightarrow{a}$  and  $\cdot^i$  as transition relations. Due to the presence of past temporal operators, we may have to resort to some notion of bisimilarity that takes backward steps also into account (a notion of forward-backward or history-preserving bisimilarity).

## 4 Bridging the gap: relation to existing theories

In this section we show that the framework introduced in this paper is a conservative extension of the traditional process theoretic modelling on the one hand, and epistemic modelling on the other hand. To this end, we prove that the satisfaction relation defined in Section 3 preserves the standard satisfaction relations of  $\bar{\mu}$  ( $\mu$ -calculus with past) formulae on labeled transition systems and of E (epistemic logic) formulae on Kripke structures. Figure 3 illustrates the three semantic models discussed in this section: the existing LTS and KS, and the newly introduced ALTS. Figure 2 provides an overview of the connections between the various notions.

**Projecting into the process-theoretic domain** A Labeled Transition System (LTS) is a standard semantic domain for process-theoretic formalisms. Formally, an LTS over a set of labels  $L$  is a tuple  $\langle St, \rightarrow, \checkmark, s_0 \rangle$ , where  $St$  is the set of operational states,  $\rightarrow \subseteq St \times L \times St$  is the



**Fig. 3.** An ALTS  $A$  (rightmost), together with its projections: 'the temporal part'  $lts(A)$  (leftmost) and 'the epistemic part'  $em(A)$  (center). In  $lts(A)$ , the points are *states*, the arrows are *transitions*. In  $em(A)$ , points are *possible worlds* and lines are *indistinguishability* relations labeled with identities of agents. In  $(A)$ , the points are states and possible worlds simultaneously. Both temporal and epistemic relations are present. The epistemic valuation in a state is given by the actions executed from the initial state to that state. In the initial state, combined temporal epistemic formulae hold like  $\langle a \rangle (K_1 a^\uparrow \wedge \neg K_2 a^\uparrow)$  — expressing that after an  $a$ -action, it is known to principal 1 that action  $a$  has been executed, but 2 doesn't know that. However, 2 knows that one of the actions  $a, b, c$  has been executed ( $\langle a \rangle (K_2 (a^\uparrow \vee b^\uparrow \vee c^\uparrow))$ ). More interestingly, after step  $d$  is executed, 2 has learned that  $a$  must have been the first step:  $\langle a \rangle \langle d \rangle K_2 a^\uparrow$ . Modeling this phenomenon of agents learning facts that were never explicitly told to them is exactly the power of epistemic logic approaches, that we took over in the combined framework.

transition relation,  $\checkmark \subseteq St$  is the termination predicate and  $s_0$  is the initial state. It typically represents the behavior of a reactive system in terms of states and transitions. Then requirements formulated in a temporal logic are matched against this behavior in the process of model checking.

A very general logical language to reason about processes is the  $\mu$ -calculus with past ( $\bar{\mu}$ ) [Nie98], which is obtained by leaving out the knowledge construct  $K_i \phi$  from the syntax of our logic presented in Section 3. That a state  $s$  in the LTS  $T = \langle S, \rightarrow, \checkmark, s_0 \rangle$  satisfies a  $\bar{\mu}$  formula  $\phi$  (denoted  $T, s \models_{\bar{\mu}} \phi$ ) is defined inductively as follows:

$$\begin{aligned}
T, s \models_{\bar{\mu}} \top & \quad \text{iff true} \\
T, s \models_{\bar{\mu}} \neg \phi & \quad \text{iff } T, s \not\models_{\bar{\mu}} \phi \\
T, s \models_{\bar{\mu}} \phi_1 \wedge \phi_2 & \quad \text{iff } T, s \models_{\bar{\mu}} \phi_1 \text{ and } s \models_{\bar{\mu}} \phi_2 \\
T, s \models_{\bar{\mu}} \langle a \rangle \phi & \quad \text{iff exists } s' \in S, \text{ s.t. } s \xrightarrow{a} s' \text{ and } T, s' \models_{\bar{\mu}} \phi \\
T, s \models_{\bar{\mu}} \langle \bar{a} \rangle \phi & \quad \text{iff exists } s' \in S, \text{ s.t. } s' \xrightarrow{a} s \text{ and } T, s' \models_{\bar{\mu}} \phi \\
T, s \models_{\bar{\mu}} \nu X. \phi(X) & \quad \text{iff } s \in \bigcup \{ S' \subseteq S \mid \forall s' \in S'. T, s' \models_{\bar{\mu}} \phi(X := S') \}
\end{aligned}$$

We prove that the ALTS +  $E\bar{\mu}$  model checking framework properly extends the LTS +  $\bar{\mu}$  model checking framework, in the sense that whatever was possible in the latter, is still possible and has the same meaning in the former. This is witnessed by the fact that LTS +  $\bar{\mu}$  can be immediately obtained by simply stripping the ALTS from the  $I$  relations and the  $E\bar{\mu}$  logic from the epistemic operator  $K_i$ . The following theorem formalizes this.

**Theorem 2.** Consider a PAi process  $p$  and the ALTS  $A = \langle St, \rightarrow, \checkmark, I, s_0 \rangle$  obtained as semantics of  $(p, \square)$  by following the SOS rules in Figure 1. Let  $(q, \pi)$  be a state in  $A$ , reachable from  $(p, \square)$  (i.e. in the transitive closure of  $\rightarrow$  from  $s_0 = (p, \square)$ ). Let us define  $lts(A) = (St, \rightarrow, \checkmark, s_0)$ . Then, for each  $\bar{\mu}$  formula  $\phi$ ,  $A, (q, \pi) \models \phi$  iff  $lts(A), q \models_{\bar{\mu}} \phi$ .

*Proof.* Straightforward, since the definitions of  $\models$  and  $\models_{\bar{\mu}}$  are identical for the  $\bar{\mu}$  fragment of  $E\bar{\mu}$ .  $\square$

This means that for purely temporal aspects of correctness, one can safely ignore the epistemic aspects of our semantics and our logic.

**Projecting into the epistemic domain** Epistemic logics are mainly concerned with expressing subtle properties of communication acts, related to the knowledge, beliefs and intentions of

communicating parties. In standard epistemic logic (following [Hin62]), epistemic properties are validated in static rich snapshots of communications (*epistemic models*), that don't express the temporal evolution of the system. The language of epistemic logic with common knowledge is defined by:

$$\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid K_i\phi \mid C_J\phi$$

Here the  $p$  comes from a given set of propositional variables  $\mathcal{Prop}$ . These propositions represent the atomic facts the agents may know about. The subscript  $i$  ranges over a given set of agents  $\mathcal{I}$ , and  $J$  over subsets of  $\mathcal{I}$ . The standard reading of the epistemic modalities  $K_i$  and  $C_J$  is the same as ours in the previous section: “ $i$  knows that...” and “it is common knowledge among the agents in  $J$  that...”, respectively.

An *epistemic (S5-)model* is a Kripke structure  $\langle W, \{R_i \mid i \in \mathcal{I}\}, V \rangle$ , where  $W$  is a nonempty set of possible worlds,  $R_i$  is an equivalence relation on  $W$  for each  $i \in \mathcal{I}$ , and  $V : \mathcal{Prop} \rightarrow \mathcal{P}(W)$  is a valuation function assigning to each propositional variable the set of worlds in which it holds. Given an epistemic model  $M$  and world  $s \in W$ , satisfaction ( $\models_{\mathbf{E}}$ ) is defined recursively as follows:

$$\begin{aligned} M, s \models_{\mathbf{E}} p & \quad \text{iff } s \in V(p) \\ M, s \models_{\mathbf{E}} \neg\phi & \quad \text{iff it is not true that } M, s \models_{\mathbf{E}} \phi \\ M, s \models_{\mathbf{E}} \phi_1 \wedge \phi_2 & \quad \text{iff } M, s \models_{\mathbf{E}} \phi_1 \text{ and } M, s \models_{\mathbf{E}} \phi_2 \\ M, s \models_{\mathbf{E}} K_i\phi & \quad \text{iff for all } M, s' \in S, \text{ if } sR_i s' \text{ then } M, s' \models_{\mathbf{E}} \phi \\ M, s \models_{\mathbf{E}} C_J\phi & \quad \text{iff for all } M, s' \in S, \text{ if } s(\cup_{i \in J} R_i)^* s' \text{ then } M, s' \models_{\mathbf{E}} \phi \end{aligned}$$

To isolate ‘the epistemic part’ of our framework, we make suitable choices for the set of propositions, and the set of agents. In the context of our  $PAi$ -processes we associate with every action  $a \in \mathcal{Act}$  a proposition  $\underline{a}$  (which can be read as “ $a$  has been executed sometime before”), and we let  $\mathcal{Prop} := \{\underline{a} \mid a \in \mathcal{Act}\} \cup \{\top\}$ . Furthermore, we let  $\mathcal{I}$  be our set of identities  $\mathcal{Id}$ . We call the resulting logic  $\mathbf{E}$ .

We can then say that our modelling and verification framework is also conservative when it comes to purely epistemic aspects. Namely, if we restrict the ALTS associated with a  $PAi$  process to the  $I$  relations, we obtain an epistemic model where purely epistemic formulas hold exactly when they hold in the original ALTS, according to the  $E\bar{\mu}$  satisfaction relation. Let us define an embedding  $\mathcal{E} : \mathbf{E}\text{-forms} \rightarrow E\bar{\mu}\text{-forms}$  of formulas into  $E\bar{\mu}$  formulas, by taking  $\mathcal{E}(\underline{a}) = a^\dagger$  and extending from there:

$$\begin{aligned} \mathcal{E}(\top) &= \top & \mathcal{E}(\phi_1 \wedge \phi_2) &= \mathcal{E}(\phi_1) \wedge \mathcal{E}(\phi_2) \\ \mathcal{E}(\underline{a}) &= a^\dagger & \mathcal{E}(K_i\phi) &= K_i\mathcal{E}(\phi) \\ \mathcal{E}(\neg\phi) &= \neg\mathcal{E}(\phi) & \mathcal{E}(C_J\phi) &= \nu X. (\bigwedge_{i \in J} K_i(X \wedge \phi)). \end{aligned}$$

The following theorem formally expresses the conservativeness of  $E\bar{\mu}$  w.r.t.  $\mathbf{E}$ .

**Theorem 3.** *Consider a  $PAi$  process  $p$  over the set of actions  $\mathcal{Act}$ . Let  $\mathbf{A} = \langle St, \rightarrow, \surd, I, s_0 \rangle$  be the ALTS obtained as semantics of  $(p, \square)$  by following the SOS rules in Figure 1. Let us define its associated epistemic model as  $em(\mathbf{A}) = \langle St, \{\cdot^i \mid i \in \mathcal{Id}\}, V \rangle$ , with propositions from  $\mathcal{Prop}$ ,  $V(\underline{a}) = \{s \in S \mid \mathbf{A}, s \models \mathcal{E}(\underline{a})\}$  and  $V(\top) = St$ . Then for any  $\mathbf{E}$  formula  $\phi$  and any possible world  $s \in St$ ,  $\mathbf{A}, s \models \mathcal{E}(\phi)$  iff  $em(\mathbf{A}), s \models_{\mathbf{E}} \phi$ .*

*Proof.* Let  $\phi$  be an arbitrary EL formula. We prove both implications simultaneously by induction on the structure of  $\phi$ :

- $\phi = \top$ . Immediate, by definition of  $\models$  and  $\models_{\mathbf{E}}$ :  $\mathbf{A}, s \models \mathcal{E}(\top)$  iff (definition of  $\mathcal{E}$ )  $\mathbf{A}, s \models \top$  iff (definition of  $\models$ ) true iff (definition of  $V(\top)$ )  $em(\mathbf{A}), s \models_{\mathbf{E}} \top$ .
- $\phi = \underline{a}$ .  $\mathbf{A}, s \models \mathcal{E}(\underline{a})$  iff (definition of  $V$ )  $em(\mathbf{A}), s \models_{\mathbf{E}} \underline{a}$
- $\phi = \neg\psi$ .  $\mathbf{A}, s \models \mathcal{E}(\neg\psi)$  iff (definition of  $\mathcal{E}$ )  $\mathbf{A}, s \models \neg\mathcal{E}(\psi)$  iff (definition of  $\models$ )  $\mathbf{A}, s \not\models \mathcal{E}(\psi)$  iff (induction hypothesis)  $em(\mathbf{A}), s \not\models_{\mathbf{E}} \psi$  iff (definition of  $\models_{\mathbf{E}}$ )  $em(\mathbf{A}), s \models_{\mathbf{E}} \neg\psi$ .
- $\phi = \phi_1 \wedge \phi_2$ .  $\mathbf{A}, s \models \mathcal{E}(\phi_1 \wedge \phi_2)$  iff (definition of  $\mathcal{E}$ )  $\mathbf{A}, s \models \mathcal{E}(\phi_1)$  and  $\mathbf{A}, s \models \mathcal{E}(\phi_2)$  iff (induction hypothesis)  $em(\mathbf{A}), s \models_{\mathbf{E}} \phi_1$  and  $em(\mathbf{A}), s \models_{\mathbf{E}} \phi_2$  iff (definition of  $\models_{\mathbf{E}}$ )  $em(\mathbf{A}), s \models_{\mathbf{E}} \phi_1 \wedge \phi_2$ .

- $\phi = K_i\psi$ .  $A, s \models \mathcal{E}(K_i\psi)$  iff (definition of  $\mathcal{E}$ )  $A, s \models K_i\mathcal{E}(\psi)$  iff (definition of  $\models$ ) for all  $s' \in S$  s.t.  $s \cdot^i \cdot s' : A, s' \models \mathcal{E}(\psi)$ . Let  $s'$  be an arbitrary state in  $S$  s.t.  $s \cdot^i \cdot s'$ , and therefore  $A, s' \models \mathcal{E}(\psi)$ . From the induction hypothesis, this happens iff  $em(A), s' \models_{\mathbb{E}} \psi$ . It follows that for all  $s' \in S$ , if  $s \cdot^i \cdot s'$  then  $em(A), s' \models_{\mathbb{E}} \psi$ . This holds iff (definition of  $\models_{\mathbb{E}}$ )  $em(A), s \models_{\mathbb{E}} K_i\psi$ .
- $\phi = C_J\psi$ .  $A, s \models \mathcal{E}(C_J\psi)$  iff (definition of  $\mathcal{E}$ )  $A, s \models \nu X. (\bigwedge_{i \in J} K_i(X \wedge \psi))$  iff (equivalent characterization from [FHMV95]) for all  $s' \in S$ , if  $s(\bigcup_{i \in J} \cdot^i \cdot)^* s'$  then  $A, s' \models \psi$ . Let  $s'$  be an arbitrary state in  $S$  s.t.  $s(\bigcup_{i \in J} \cdot^i \cdot)^* s'$ , and therefore  $A, s' \models \mathcal{E}(\psi)$ . From the induction hypothesis, this happens iff  $em(A), s' \models_{\mathbb{E}} \psi$ . It follows that for all  $s' \in S$ , if  $s(\bigcup_{i \in J} \cdot^i \cdot)^* s'$  then  $em(A), s' \models_{\mathbb{E}} \psi$ . This holds iff (definition of  $\models_{\mathbb{E}}$ )  $em(A), s \models_{\mathbb{E}} C_J\psi$ .

□

Besides the conservation of standard epistemic model checking, our framework also meets another relevant characteristics of epistemic modelling. Namely, we can show that the behavior part of an ALTS preserves the property of *perfect recall* ('no forgetting' in [HV88]) for the epistemic part, in the sense that knowledge accumulated is not lost. To formalize this, we need to refine the transition relation  $\rightarrow$ , for a fixed identity  $i$ , into update transitions and invisible transitions. Therefore, let  $\overset{i}{\hookrightarrow} = \bigcup_{a \in Act} \overset{a}{\rightarrow} \cap \cdot^i \cdot$  be the relation that gives the invisible transitions for  $i$ , and let  $\overset{i}{\rightsquigarrow}$  be the ' $i$ -update' relation  $\{(s, t) \mid \exists a \in Act, s' \in St \text{ s.t. } s \overset{a}{\rightarrow} s', s \cdot^i \cdot s' \text{ and } s' \overset{i}{\hookrightarrow} *t\}$ .

**Theorem 4 (perfect recall).** *Let  $A$  be a fixed ALTS generated by the SOS rules, as above, and let  $i \in \mathcal{I}d$  be any identity. For any states  $s, t, s', t' \in St$  and any  $n \in Nat$ , if  $s \overset{i}{\rightsquigarrow}^n s', t \overset{i}{\rightsquigarrow}^n t'$  and  $s \cdot^i \cdot t$ , then  $s' \cdot^i \cdot t'$ , (where  $\overset{i}{\rightsquigarrow}^n$  denotes the application of  $\overset{i}{\rightsquigarrow}$   $n$  times).*

We prove the theorem by a sequence of lemmas:

**Lemma 2.** *For any  $a \in Act$  and  $p, q, \pi, \pi'$ , if  $(p, \pi) \overset{a}{\Rightarrow} (q, \pi')$  then  $\pi' = \pi \frown d$ .*

*Proof.* It follows immediately from rule **(a)**, which is the only rule that affects the construction of histories. □

**Lemma 3 (one-step).** *Let  $A$  be an ALTS produced by the SOS. If  $s, s', t, t'$  are reachable states in  $St$  s.t.  $s \overset{a}{\rightarrow} t, s' \overset{b}{\rightarrow} t'$  and  $t \cdot^i \cdot t'$ , then either  $s \cdot^i \cdot s'$  or  $s \cdot^i \cdot t'$  or  $s' \cdot^i \cdot t$ .*

*Proof.*  $s \overset{a}{\rightarrow} t$  and  $s' \overset{b}{\rightarrow} t'$  have been generated by rule **(strip)** and  $t \cdot^i \cdot t'$  has been generated by rule **(I)**, so there exist  $x_0, \pi_0, y_0, \pi'_0, x_1, \pi_1, y_1, \pi'_1, J, J'$  s.t.  $s = (x_0, \pi_0), s' = (y_0, \pi'_0), t = (x_1, \pi_1), t' = (y_1, \pi'_1), s \overset{(J)a}{\Rightarrow} t, s' \overset{(J')b}{\Rightarrow} t'$ , and  $\pi_1 \stackrel{i}{=} \pi'_1$ . We need to prove that  $\pi_0 \stackrel{i}{=} \pi'_0$ .

From Lemma 2,  $\pi_1 = \pi_0 \frown (J)a$  and  $\pi'_1 = \pi'_0 \frown (J')b$ . Further,  $\pi_1 \stackrel{i}{=} \pi'_1$  has been generated by one of the following rules:

- (= **refl**) So,  $\pi_1 = \pi'_1 = \pi_0 \frown (J)a = \pi'_0 \frown (J')b$ .  $\frown$  is string concatenation, so it must hold that  $(J)a = (J')b$  and  $\pi_0 = \pi'_0$ . By applying (= **refl**), we get indeed  $\pi_0 \stackrel{i}{=} \pi'_0$ .
- ( $\rho\mathbf{0}$ ) Since  $\pi_1 = \pi_0 \frown (J)a$  and  $\pi'_1 = \pi'_0 \frown (J')b$ , the first premise of this rule translates to  $\pi_0 \stackrel{i}{=} \pi'_0$ .
- ( $\rho\mathbf{1}$ ), ( $\rho\mathbf{2}$ ) The argument is similar to the one for ( $\rho\mathbf{0}$ ).
- ( $\tau\mathbf{0}$ )  $\pi_1 = \pi_0 \frown (J)a \stackrel{i}{=} \pi'_1$ . Then, according to the premise of ( $\tau\mathbf{0}$ ),  $\pi_0 \cdot^i \cdot \pi'_1$ . Then we can apply **(I)** and obtain  $(x_0, \pi_0) \cdot^i \cdot (y_1, \pi'_1)$ , that is  $s \cdot^i \cdot t'$ .
- ( $\tau\mathbf{1}$ ) The argument is symmetric to the one used for ( $\tau\mathbf{0}$ ). We obtain  $s' \cdot^i \cdot t$ .
- ( $\tau\mathbf{2}$ ) Then  $a = \tau$  and the premise translates to  $\pi_0 \stackrel{i}{=} \pi'_1$ , that is, by **(I)**,  $s \cdot^i \cdot t'$ .

□

**Lemma 4 (update).** *Let  $A$  be an ALTS produced by the SOS and let  $i \in \mathcal{I}d$  be an identity. If  $s, s', t, t'$  are reachable states in  $St$  s.t.  $s \overset{i}{\rightsquigarrow} t, s' \overset{i}{\rightsquigarrow} t'$  and  $t \cdot^i \cdot t'$ , then  $s \cdot^i \cdot s'$ .*

*Proof.* From the definition of  $\overset{i}{\rightsquigarrow}$ , there exist action labels  $a, b \in \mathcal{Act}$  and states  $u$  and  $u'$  s.t.  $s \xrightarrow{a} u$  and  $u \xrightarrow{i} t$ , respectively  $s' \xrightarrow{b} u'$  and  $u' \xrightarrow{i} t'$ . With the extra condition that  $s \overset{j}{\cdot} u$  and  $s' \overset{j}{\cdot} u'$ . From the definition of  $\overset{i}{\rightsquigarrow}$  and the fact that  $\overset{i}{\cdot}$  is an equivalence relation (Lemma 1), it follows that  $u \overset{i}{\cdot} t$  and  $u' \overset{i}{\cdot} t'$ . Therefore also  $u \overset{i}{\cdot} u'$ . By applying Lemma 3 to the states  $s, s', u, u'$ , we obtain that one of the following holds:  $s \overset{i}{\cdot} u', s' \overset{i}{\cdot} u$ , or  $s \overset{i}{\cdot} s'$ . The first two options are excluded, because they imply, by  $\overset{i}{\cdot}$  being an equivalence, that  $s \overset{i}{\cdot} u$  or  $s' \overset{i}{\cdot} u'$ , which are in contradiction with the definition of  $\overset{i}{\rightsquigarrow}$ . Therefore,  $s \overset{i}{\cdot} s'$ .  $\square$

*Proof (Theorem 4).* By induction on  $n$ . If  $n = 0$ , then  $s'$  is identified with  $s$  and  $t'$  with  $t$ . So, trivially,  $s \overset{j}{\cdot} t$  implies  $s' \overset{j}{\cdot} t'$ . In the general case, suppose  $s' \overset{i}{\cdot} t'$  and let  $s'' \xrightarrow{a} s'$  be the last step of the update trace  $s \overset{i}{\rightsquigarrow}^n s''$  and  $t'' \xrightarrow{i} t'$  the last step of the trace  $t \overset{i}{\rightsquigarrow}^n t''$ . By Lemma 4, it follows that  $s'' \overset{i}{\cdot} t''$  and we can rely on the induction hypothesis to obtain  $s \overset{i}{\cdot} t$ , which contradicts theorem's assumption  $s \overset{j}{\cdot} t$ .  $\square$

## 5 An example protocol: Dining Cryptographers

In order to illustrate the relative advantages of the combined framework compared to using exclusively the operational approach or the epistemic one, we discuss the Dining Cryptographers protocol [Cha88], which has already been independently and extensively analyzed using both operational [SS96,BP05] and epistemic approaches [HS04,HO05,RL04]. This is a well-known example of a protocol in which anonymity is the main requirement. The story, a metaphor for anonymous broadcast, is about three cryptographers having dinner together. At the end, they learn that the bill has been paid anonymously by one of them, or by the NSA (the National Security Agency). They respect each other's right to anonymity, but they wish to find out whether the payer was NSA or not. To this end, they come up with the following protocol: each neighboring pair of cryptographers generates a shared bit, by flipping a coin; then each cryptographer computes the exclusive or (XOR) of the two bits she sees, then announces the result — or the flipped result, if she was herself the payer. The XOR of the three publicly announced results indicates whether the payer was an insider or NSA.

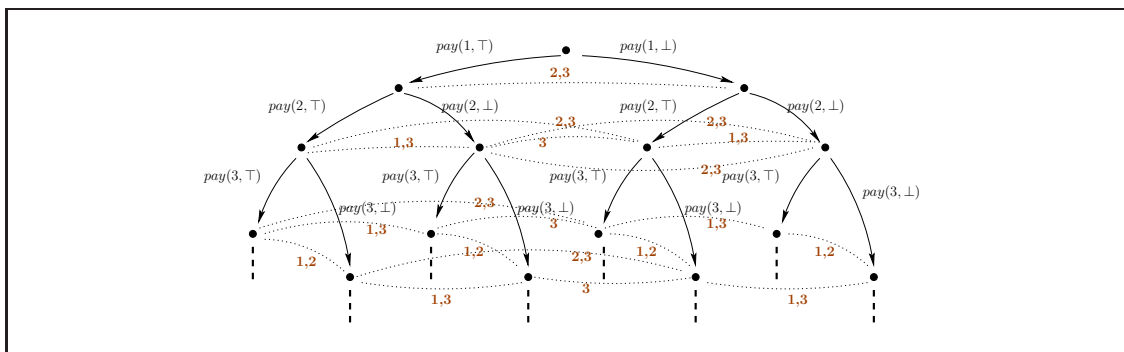
**Model** A model of this protocol in our process language is shown in Figure 4. Inspired by the input construction in the algebraic specification language  $\mu CRL$ , we use  $\sum_{x:\{x_1 \dots x_n\}} P(x)$  as an abbreviation for  $P(x_1) + \dots + P(x_n)$ , where  $\{x_1 \dots x_n\}$  is a finite set and  $P(x_i)$  denotes the process expression  $P(x)$  in which  $x_i$  has been substituted for  $x$ . The model is rather close to the CSP description presented in [SS96], the only significant difference being that the actions are annotated with identities from the set  $\mathcal{Id} = \{1, 2, 3, M\}$ . Note that the parameters used in the basic actions and process definitions are just generic names for the concrete instances resulting from instantiating them. For example,  $?pay(i, b)$  is not defined in our process language but rather it stands for a number of instances such as  $?pay(1, \top)$ ,  $?pay(i, \perp)$  each of which are basic actions (obtained by globally replacing  $i$  and  $b$  with a member of  $\mathcal{Id}$  and  $\{\perp, \top\}$  in the process definition each time). The behavior of the  $i$ th cryptographer is specified by the process  $Crypt(i)$  and the behavior of the whole DC system as a parallel composition of  $Crypt(i)$ 's and the *Master* process:

$$DC_3 = Crypt(1) \parallel Crypt(2) \parallel Crypt(3) \parallel Master$$

A cryptographer process executes a series of actions corresponding to the three big steps of the protocol: decide whether to pay or not, flip the coins together with the neighbors, and announce the result of XOR-ing the two coins and her own paying bit. The first step is modeled as a statement  $pay(i, b)$ , which is in fact a communication step with the *Master*. The second step is modeled by the processes  $CryptFlip(i)$  and  $CryptShare(i)$ . In other existing models [SS96,BP05], the shared

$$\begin{aligned}
Crypt(i) &= \sum_{b:Bool} ( (i)?pay(i, b); CryptFlip(i, b) ) \\
CryptFlip(i, b) &= \sum_{c:Bool} ( (i)flip(i, c); CryptShare(i, b, c) ) \\
CryptShare(i, b, c) &= \sum_{d:Bool} ( ((i)!share(i \bmod 3 + 1, c) \parallel (i)?share(i, d)); \\
&\quad CryptBcast(i, b, c, d) ) \\
CryptBcast(i, b, c, d) &= ((i)!bcast(i, b \oplus c \oplus d); (i)!bcast(i, b \oplus c \oplus d)) \\
&\quad \parallel \sum_{x,y:Bool} (((i)?bcast(i + 1 \bmod 3 + 1, x) \\
&\quad \parallel (i)?bcast(i \bmod 3 + 1, y)); \\
&\quad nsa(i, \neg(b \oplus c \oplus d \oplus x \oplus y))) \\
Master &= (M)!pay(1, \top); (M)!pay(2, \perp); (M)!pay(3, \perp) \\
&+ (M)!pay(1, \perp); (M)!pay(2, \top); (M)!pay(3, \perp) \\
&+ (M)!pay(1, \perp); (M)!pay(2, \perp); (M)!pay(3, \top) \\
&+ (M)!pay(1, \perp); (M)!pay(2, \perp); (M)!pay(3, \perp)
\end{aligned}$$

**Fig. 4.** A *PAi* model of The Dining Cryptographers protocol.  $\oplus$  denotes exclusive or.



**Fig. 5.** A small fragment from the ALTS generated for the DC specification. For readability, we omitted some  $\cdot \cdot \cdot$  relations that can be generated by the reflexive and transitive closure rules.

coins are represented by separate processes, but in order to keep the specification simple, we merge the behavior of the  $i$ th coin with the behavior of the  $i$ th cryptographer. Therefore, process  $Crypt(i)$  will execute a  $flip$  action and then share the result with the right-hand neighbor, by executing an action  $!share$  which will synchronize with the  $?share$  from the next cryptographer in the ring.  $CryptBcast$  models the last phase, announcing the result of one's computation ( $!bcast$ ), receiving the results from all the others ( $?bcast$ ) and concluding for itself that  $NSA$  paid or not ( $nsa(i, \top)$ ,  $nsa(i, \perp)$ ).

The renaming function  $\rho$  specifies how much of a cryptographers' actions is visible for observing parties. For any  $i \in \{1, 2, 3\}$  and  $b \in \{\top, \perp\}$ , we define

$$\begin{aligned}
\rho(pay(i, b)) &= pay(i) & \rho(bcast(i, b)) &= bcast(i, b) & \rho(share(i, c)) &= share(i) \\
\rho(flip(i, b)) &= flip(i) & \rho(nsa(i, b)) &= nsa(i, b)
\end{aligned}$$

where  $pay(1)$ ,  $bcast(1, \top)$ ,  $\dots$  are basic actions.

**Analysis** Figure 5 shows the top part of the ALTS generated by the rules in Figure 1 from the process specification in Figure 4. We check relevant functional and epistemic properties of this protocol by matching  $E\bar{\mu}$  formulas against this ALTS, as dictated by the satisfaction relation  $\models$  (Definition 2).

First of all, we can check functional correctness, by asking for instance that in all executions where one of the cryptographers paid, the action  $nsa(1, \top)$  is eventually observable, meaning that the first cryptographer draws the right conclusion that the payer was an insider. This requirement is a purely temporal formula, for each  $i \in \{1, 2, 3\}$ :

$$[pay(i, \top)] \bigwedge_{j \in \{1, 2, 3\}} [.*]nsa(j, \perp).$$

Better yet, we can check the following temporal epistemic statement, for each  $i \in I$ .

$$[pay(i, \top)][.*]C_{\{1, 2, 3\}}(\bigwedge_{j \in \{1, 2, 3\}} nsa(j, \perp)^\dagger)$$

which expresses the fact that “everybody knows that the payer is an insider” eventually becomes common knowledge among the three cryptographers.

Anonymity, the main goal of the protocol, is not expressible as a purely temporal property, but it is conveniently expressible as a temporal epistemic property. The anonymity of cryptographer  $i$  (holding in the initial state of our model) is specified by the following formula:

$$[pay(i, \top)] \bigwedge_{j \in \{1, 2, 3\} \setminus \{i\}} \neg \langle .* \rangle K_j (pay(i, \top)^\dagger).$$

All these properties are satisfied by our  $PAi$  model, according to the satisfaction relation  $\models$  defined in Section 3.

**Comparison to other DC models** Our process language allows a simple and operational modeling, just as intuitive as any other process language, see also for instance a CSP model [SS96] and a pi-calculus model [BP05] of the Dining Cryptographers. All these models are definitely closer to the protocol description than logic models [HW02, RL06] and moreover, they are supported by a semantics which formally links the description of a protocol to its actual behavior model.

On the other hand, epistemic logic models allow expressing and checking anonymity as epistemic formulae, which is much more natural than the equivalence checking method employed in the process theoretic approach. More precisely, operational approach to verification of anonymity requires writing down new descriptions for each anonymity property that has to be checked, because these properties are dependent on the point of view of the observer. In the ALTS that our specification generates, all points of view are simultaneously present, thus a direct and natural (epistemic) verification is possible.

## 6 Implementation

We have implemented the semantics of  $PAi$  and a subset of our  $E\bar{\mu}$  logic in the rewriting logic of Maude. This subsets includes the basic temporal and epistemic operators; however, instead of defining the general fixpoint construction, we restricted our attention to a few useful acronyms (involving fixpoints) defined in Section 3. Our implementation enables one to investigate the behavior of protocols specified in  $PAi$  by generating their traces (simulating them), and verifying  $E\bar{\mu}$  formulae on them. The current implementation can be enhanced considerably in terms of efficiency and is only meant to be a working prototype for our theory and small examples therein (such as the case study of Section 5). Next, we present a brief overview of the implementation.

### 6.1 $PAi$ in Maude

The syntax of  $PAi$  is modeled in Maude as follows.

```

sort Act      .
sort DAct     .
sort Proc     .

op tau  : -> Act [ ctor ] .

op rcv ( _ ) _ : IdSet Act -> DAct [ctor]      .
op snd ( _ ) _ : IdSet Act -> DAct [ctor]      .
op sync ( _ ) _ : IdSet Act -> BDAct [ctor]     .

```

In the above specification `Act`, `DAct` and `Proc` are sorts for basic actions, decorated actions and processes, respectively. `ctor` designates constructors of each sort. In this case, `tau` is defined to be an action. `rcv`, `snd` and `sync` define decorated actions for send, receive and synchronization (or just non-communicating actions, denoted in the original syntax by `?`, and without any preceding mark, respectively).

```

subsort DAct < Proc .

op NIL   : -> Proc      [ctor]      .
op _ ; _ : Proc Proc -> Proc [ctor prec 22] .
op _ + _ : Proc Proc -> Proc [ctor prec 23] .
op _ || _ : Proc Proc -> Proc [ctor prec 24] .

```

In the above specification, the `subsort` expression defines that each decorated action is process, other definitions define `NIL` (denoted in the original syntax by `0`), `;`, `+` and `||` as constructors of the sort `Proc`. Expression `prec i` defines their precedence (binding power); operators with a lower `i` bind stronger.

The implementation of our operational semantics (given in Figure 1) is a straightforward translation of the deduction rules. In our Maude formalization, termination is modeled by a set of equations and transitions are modeled by a set of rewrite rules For example deduction rules **(0)**, **(a)** and **(s1)** are implemented as follows.

```

eq ( tick NIL ) = true .

rl $* ( d , pi ) => ( {d} R ( NIL , pi ^ d ) ) .

crl $* ( p0 ; p1 , pi ) => ( {d} R ( pp1 , pip ) )
  if ( tick p0 ) /\
    $* ( p1 , pi ) => ( {d} R ( pp1 , pip ) ) .

```

In the above specification `eq` stands for equality and `rl` and `crl` stand for rewrite and conditional rewrite rules, respectively. We use the auxiliary symbol `$*` at the left-hand side of the rewrite rules to designate this particular type of transition, denoted in our semantics with  $\Rightarrow$ .

In the semantics, we have another transition relation, denoted by  $\rightarrow$ . Instead of defining  $\rightarrow$  in terms of  $\Rightarrow$ , as in **(strip)**, for the sake of efficiency, we implemented  $\rightarrow$  directly by stripping down the decorated action in each semantic rule. For example, for the three deduction rules for  $\Rightarrow$  specified above, we get the following corresponding rules for  $\rightarrow$  (in the following rules `$` designates the transition relation  $\rightarrow$ ).

```

rl $ ( ( sync (I) a ) , pi )
  => ( {a} r ( NIL , pi ^ sync (I) a ) ) .

crl $ ( p0 ; p1 , pi ) => ( {a} r ( pp1 , pip ) )
  if ( tick p0 ) /\
    $ ( p1 , pi ) => ( {a} r ( pp1 , pip ) ) .

```

Finally, the concept of indistinguishability is formalized in Maude. The following two equalities, represent deduction rules ( $= \mathbf{refl}$ ) and ( $= \rho\mathbf{0}$ ) in Figure 1.

```

eq ( pi obseq ( i ) pi ) = true .

ceq ( ( pi ^ ( sync (I)a ) )
      obseq ( i ) ( pip ^ ( sync (J)a ) ) ) = true
      if ( ( pi obseq ( i ) pip ) and
            ( i in ( I cap J ) ) ) .

```

## 6.2 $E\bar{\mu}$ in Maude

The syntax of a subset of  $E\bar{\mu}$  is formalized in Maude as follows.

```

sort Emuform .

op top : -> Emuform [ctor] .
op bot : -> Emuform [ctor] .
op emnot _ : Emuform -> Emuform .
op _ emand _ : Emuform Emuform -> Emuform [ctor] .
op _ emor _ : Emuform Emuform -> Emuform [ctor] .
op <_>_ : Act Emuform -> Emuform [ctor] .
op [_]_ : Act Emuform -> Emuform [ctor] .
op K{ }_ : IdSet Emuform -> Emuform [ctor] .
op C{ }_ : IdSet Emuform -> Emuform [ctor] .
op _ <-| : Act -> Emuform [ctor] .
op [.*]_ : Emuform -> Emuform [ctor] .

```

Basic logical connectives “not”, “and”, and “or” are denoted by `emnot`, `emand`, and `emor` to avoid confusion with their boolean counterparts. Temporal operators  $\langle a \rangle$  and  $[a]$  are denoted by `<a>` and `[a]`, respectively. Knowledge and common knowledge operators are denoted by `K{ }` and `C{ }`, respectively. The two acronyms  $\_^\dagger$  and  $[.*]$  are respectively written as `<-|` and `[.*]` in Maude.

The formalization of the semantics of logical connectors is straightforward. We write  $s_0 \models \phi$  for the semantics of formula  $\phi$  at state  $s$  in an LTS with the initial state  $s_0$  (with the semantics given before). The following conditional rewrite rule, for example, specifies the semantics of `emand`.

```

crl ( s0 (p, pi) |= ( phi emand psi ) ) => ( b0 and b1 )
    if ( s0 (p, pi) |= phi ) => b0 /\
        ( s0 (p, pi) |= psi ) => b1 .

```

The semantics of temporal and epistemic constructs is a bit more involved and makes use of the reflective semantics of Maude. For example, consider the following semantics of the operator `<a>`.

```

crl [atrue] : ( s0 (p, pi) |= ( < a > phi ) ) => true
    if trans {a} (p, pi) => target (pp, pip) /\
        ( s0 (pp, pip) |= phi ) => true .

crl ( s0 s |= ( < a > phi ) ) => false
    if ( not ( isaTrue ( upTerm ( s0 s |= ( < a > phi ) ) ) ) ) .

```

If at state  $(p, \text{pi})$  a transition labeled with  $a$  is enabled, it is easy to establish that  $\langle a \rangle \text{phi}$  is true; it suffices to find a transition labeled with  $a$  at the target of which formula  $\text{phi}$  holds. This is naturally captured by the first conditional rewrite rule, labeled `atru`, given above. However, to establish that  $\langle a \rangle \text{phi}$  is false, either all transition labeled with  $a$  lead to a state in which  $\text{phi}$  does not hold, or there is no transition labeled with  $a$  enabled at  $(p, \text{pi})$ . Both these statements involve checking the impossibility of a certain transition which can be achieved through the reflective semantics of Maude. In other words, propositions like `isaTrue` and `rMove` involve using the meta-level function `metaXapply` which gives us information about possibility of applying a certain rewrite rule on a term. The following is the implementation of function `rMove`.

```
ceq isaTrue ( t ) =
  isfalse? :: Result4Tuple
  if isfalse? :=
    metaXapply(['EmuBasic], t , 'atru , none , 0 , unbounded, 0 ) .
```

The above function returns whether it is possible to apply rewrite rule `atru` (given before) on term  $t$ . A similar approach has been taken in the implementation of the semantics of other constructs such as `[a]_` or `K{-}_`.

## 7 Conclusion

Motivated by protocols and properties where much importance is given to the participating entities and not only to the actual evolution of the system — like certain security protocols, information flow — we presented a simple process language where the concept of *identity* is explicitly present. We gave it an operational semantics in terms of an extended form of labeled transition systems and defined a satisfaction relation for properties expressed in a rich logic combining temporal and epistemic operators. The result is a specification and verification framework that combines the best parts of two complementary approaches to protocol analysis: process algebras and epistemic logics.

Our framework is particularly suitable for modeling and verification of protocols on top of authenticated secret channels, ensured for instance by a Public Key Infrastructure. In these protocols, the security threats typically do not come from an external intruder controlling the communication channels, but from the participants themselves. Examples are protocols for fair exchange, voting, auctions, anonymity. In security protocols with cryptography or active attackers, some behavioral choices are determined by the current knowledge of the principals. In particular, a principal can distinguish more traces by gaining access to keys. To properly accommodate this, our framework should be extended, possibly by allowing dynamic update of indistinguishability relation in the course of protocol execution. Note however that the current framework is just as powerful in modeling cryptography aspects as any other (traditional) process algebra. So, for these cases, more research is needed in order to find the best way of integrating the elegance of representing knowledge by indistinguishability relations with the ease of specifying the protocol operationally.

**Future work** First of all, we will build tool support for model checking  $E\bar{\mu}$  properties on ALTSS. Ideally, this can be achieved by embedding the new framework in an existing verification tool-set. The starting point will be our already existing Maude prototype [PAi].

Then we wish to experiment with applying this technique to protocols from the categories mentioned above. On a more theoretical direction, a question is whether it is possible to extend the sequent-based compositional proof system developed for the SOS + Hennessy-Milner Logic [Sim04] in order to cope with  $E\bar{\mu}$  formulas, as well. Finally, this framework can support a direct comparison of the operational and epistemic definitions of various properties. For instance, anonymity is defined operationally as (trace) equivalence between certain processes, while epistemically it is simply a negative knowledge formula. The issue of which of these definitions is stronger, if any, is not clear yet and deserves further investigation.

*Acknowledgments* We thank Luca Aceto, Dave Clarke, Jan van Eijck and Michel Reniers for comments on earlier versions of this work. Michael Huth pointed out the link between our work and data independence [Bro01].

## References

- [AG99] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.
- [Bal01] A. Baltag. Logics for insecure communication. In *Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge (TARK '01)*, pages 111–121, 2001.
- [BAN96] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Practical Cryptography for Data Internetworks*. IEEE Computer Society Press, 1996. Reprinted from the Proceedings of the Royal Society, volume 426, number 1871, 1989.
- [BHR84] D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984.
- [BKN06] J. Borgström, S. Kramer, and U. Nestmann. Calculus of cryptographic communication. In *Proceedings of FCS-ARSPA*, 2006.
- [BP05] M. Bhargava and C. Palamidessi. Probabilistic anonymity. In *Proc. 16th International Conference on Concurrency Theory (CONCUR'05)*, volume 3653 of LNCS, pages 171–185. Springer, 2005.
- [Bro01] P.J. Broadfoot. *Data Independence in the Model Checking of Security Protocols*. PhD thesis, Oxford University, 2001.
- [Cha88] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.
- [COPT07] T. Chothia, S.M. Orzan, J. Pang, and M. Torabi Dashti. A framework for automatically checking anonymity with mCRL. In *Proceedings TGC'06*, LNCS, 2007. To appear.
- [CVB06] C. Caleiroa, L. Viganò, and D. Basin. On the semantics of Alice & Bob specifications of security protocols. *Theoretical Computer Science*, 367(1-2):88–122, 2006.
- [EO06] J. van Eijck and S.M. Orzan. Epistemic verification of anonymity. In *Proceedings VODCA'06*, volume 168 of ENTCS, 2006.
- [FHMV95] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.
- [GG97] Jelle Gerbrandy and Willem Groeneveld. Reasoning about information change. *Journal of Logic Language and Information*, 6:147–169, 1997.
- [Hin62] Jaakko Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.
- [HMV04] Arjen Hommersom, John-Jules Ch. Meyer, and Erik P. de Vink. Update semantics of security protocols. *Synthese*, 142:229–267, 2004.
- [HO05] J.Y. Halpern and K.R. O’Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, pages 483–514, 2005.
- [HS04] D. Hughes and V. Shmatikov. Information hiding, anonymity and privacy: A modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.
- [HV88] Joseph Halpern and Mosche Y. Vardi. Reasoning about knowledge and time in asynchronous systems. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC'88)*, pages 53–65, New York, NY, USA, 1988. ACM Press.
- [HW02] Wiebe van der Hoek and Michael Wooldridge. Model checking knowledge and time. In Dragan Bosnacki and Stefan Leue, editors, *Proceedings of the 9th International Workshop on Model Checking of Software (SPIN'02)*, volume 2318 of *Lecture Notes in Computer Science*, pages 95–111. Springer, 2002.
- [Kra06] Simon Kramer. Logical concepts in cryptography. Cryptology ePrint Archive, Report 2006/262, 2006. <http://eprint.iacr.org/>.
- [Mil80] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.
- [Nie98] M. Nielsen. Reasoning about the past. In *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science (MFCS '98)*, pages 117–128, London, UK, 1998. Springer-Verlag.
- [PAi] A Maude implementation of PAi. <http://www.win.tue.nl/~mousavi/pai.htm>.
- [Plo04] Gordon D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60:17–139, 2004.

- [RL04] Franco Raimondi and Alessio Lomuscio. A tool for specification and verification of epistemic properties in interpreted systems. *Electronic Notes in Theoretical Computer Science*, 85(4), 2004.
- [RL06] Franco Raimondi and Alessio Lomuscio. Automatic verification of deontic interpreted systems by model checking via OBDD's. *Journal of Applied Logic*, 2006. In Press.
- [Sim04] Alex K. Simpson. Sequent calculi for process verification: Hennessy-Milner logic for an arbitrary GSOS. *Journal of Logic and Algebraic Programming*, 60–61:287–322, 2004.
- [SS96] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In Elisa Bertino, Helmut Kurth, Giancarlo Martella, and Emilio Montolivo, editors, *Proceedings of 4th European Symposium on Research in Computer Security (ESORICS'96)*, volume 1146 of *LNCS*, pages 198–218, 1996.