Telling Lies in Process Algebra

Mohammad Reza Mousavi
University of Leicester, UK
Email: mm789@le.ac.uk

Mahsa Varshosaz
Halmstad University, Sweden
Email: mahsa.varshosaz@hh.se

***Abstract*—Epistemic logic is a powerful formalism for reasoning about communication protocols, particularly in the setting with dishonest agents and lies. Operational frameworks such as algebraic process calculi, on the other hand, are powerful formalisms for specifying the narrations of communication protocols. We bridge these two powerful formalisms by presenting a process calculus in which lies can be told. A lie in our framework is a communicated message that is pretended to be a different message (or nothing at all). In our formalism, we focus on what credulous rational agents can infer about a particular run if they know the protocol beforehand. We express the epistemic properties of such specifications in a rich extension of modal μ-calculus with the belief modality and define the semantics of our operational models in the semantic domain of our logic. We formulate and prove criteria that guarantee belief consistency for credulous agents.**

I. Introduction

Motivation. A lie is an intentional announcement of (believed-to-be) incorrect information in order to deceive the audience [8]. A lie is an action, while (untrue) belief is a possible consequence of that action in agents' states. The belief aspect, i.e., epistemics, of lies is often captured by modeling the effect of lies as belief revisions or updates. Modeling lies and their epistemics have attracted substantial attention in the recent literature concerning Dynamic Epistemic Logic (DEL, cf. [7], [8], [11], [13], [19] for some recent examples; also see [18] for applications of DEL in security).

Communication protocols are often described in an operational and narrative style, describing the possible sequences of announcements (message exchanges) among the involved principals. Yet correctness properties of such protocols are often naturally expressed in terms of epistemic properties (e.g., whether a certain fact remains secret after running the protocol). Hence, it is beneficial to check epistemic properties on operational models of protocol specification [14], particularly in the presence of untruthful principals.

In this paper, we present an operational model in which lies can be told, i.e., a message can be communicated but announced to a certain audience as if something else (or nothing at all) has been communicated. In this work we improve the definition of the notion of communication compared to our previous work [4], by allowing for actions to have multiple appearances to different principals. A renaming function specifies for each principal the observed action when a communication happens. This notion is closely related to the notion of function views used in [20] for hiding information and also the notion of appearance in [23]. In our model, we abstract away from the intention of the liar and focus on what credulous rational agents (involved in the communication or external observers) can infer about a particular execution of the protocol, if they know the protocol beforehand. (Credulous agents are those agents that are willing to accept what is being told to them as long as it does not lead to any logical inconsistency with the rest of their belief.) We focus on two types of logical properties: first, what an agent consistently believes and second, whether an agent can detect a particular lie in the course of a protocol execution. The first types of properties are protocols-specific. We express these properties in a rich extension of modal μ-calculus with Dynamic Epistemic Logic constructs and define the semantics of our operational models in the semantic domain of our logic. The second type, the so-called belief properties, are meta-properties of the semantic framework and were used as a yardstick to sanity check our semantic definitions. Although these properties are relatively simple, guaranteeing them in our framework is non-trivial and involved several iteration over laborious proofs (with over 50 case distinctions) and several revisions in the epistemic semantics of our framework.

Related work. This paper integrates process algebra [2] as an operational framework and Temporal Dynamic Epistemic Logic [1] as a logical framework. It builds upon the earlier proposals [4], [5] (co-developed by the first author), in which only knowledge (thus, no lies and untrue beliefs) were incorporated.

The closest frameworks to ours are those based on Dynamic Epistemic Logic such as [7], [8], [13]. In the remainder, we focus on [7] as the point of reference for our comparison, because it is the most recent and the most developed example of such frameworks. In a nutshell, compared to [7], we put an emphasis on ease of modeling and provide a concrete and generic syntax for specifying various types of announcements. Our framework is much simpler than the action models of agent announcements in [7] in that we only specify the operational behavior of the protocol and derive the epistemic models automatically. This simplicity comes at a cost, namely, reduced expressiveness.

The semantics of our process algebra bears close resemblances to that of [7]. In [7, Section 3], a distinction has been made between the belief update for the speaker and the belief update for the addressee. We generalise our process algebraic description of epistemic actions presented in [14]. There, an action could have two appearances: one for its intended audience and another for the rest. However, in the

present paper, we generalise this to a function model, where an action can have several different appearances for different agents. In [5], we gave a translation from a similar process algebraic framework (excluding the aspect of lying) to the interpreted systems model [12], [16]. We expect a similar translation to be possible in our extended settings with lies.

In [20], information hiding is modelled by a principal's lack of knowledge (partial view) of a function, called function view. The system behavior is hence described in terms of functions and then the information hiding properties are formulated as different views of these functions. Our framework can be differentiated from this work by allowing for the possibility of passing lies (i.e., having inconsistent function views). In our framework, we allow for non-deterministic behavior (for uncertainty) and concealment (for information hiding). However, a generalisation of the notion used in [20] allowing for lies can be developed as future work and then compared to our existing framework. In [23], a logic for describing and reasoning about knowledge and the changes made in knowledge resulting from communication in a multi-agent system is presented. Our notion of renaming function in the decorated actions is closely related to the notion of appearance by Sadrzadeh [23], which is used for representing the information of the agents about propositions. In [23], uncertainty is modeled in appearance functions, while in our framework, we allow for nondeterminism in our process algebraic framework. Comparing and consolidating these two aspects is an interesting avenue for future work.

Running example. To illustrate various concepts, we use two variants of the two-player Liar's Dice game, which we call Liar's Coin, adopted from [8].

Example 1: Liar's coin. The plot of the game is as follows: two players, denoted by 1 and 2 play the following scenario:

- Player 1 bets on head and player 2 bets on tail, and stake 10 kronor,
- Player 1 tosses a coin secretly and regardless of the actual outcome, nondeterministically announces head (h) or tail (t),
- If player 1 announces t, she loses the stake, and the game ends.
- If player 1 announces h, player 2 either passes (p), by which she loses the stake, and the game ends, or
- Player 2 challenges (c), by which she adds another 10 kronor to the stake, and
- Player 1 reveals the coin, and the player with the right bet wins the stake.

We specify both the usual model of the game, as well as two variants: in the first variant, player 1 does not lie when the outcome of the coin flip is a head (since there is no incentive to lie in that case). In the second variant, player 2 only passes when she believes that the result of the coin flip has been a head. Also to illustrate our approach, we specify and analyze the following dynamic epistemic properties on both specifications:

- After player 2 passes, she does not form any belief regarding the coin flip.
- After player 2 challenges, she will believe the past result of the coin flip.
- Player 1 can deceive player 2, i.e., player 1 may see a head while player 2 believes she has seen a tail (or vice versa).

Structure of the paper. The rest of this paper is structured as follows. In Section II, we introduce the syntax of our process algebraic formalism and our logical formalism for reasoning about process algebraic specifications. Then, in Section III, we define a semantics for both formalisms in a common semantic domain. In Section IV, we extend our operational framework by allowing for explicit specification of the interplay between epistemic beliefs and operational actions, e.g., choosing among future scenarios based on agents' beliefs. In Section V, we analyze the semantic properties of specifications mainly from the viewpoint of detecting lies. In Section VI, we conclude the paper and present some directions for future research.

II. Syntax

A. Operational framework

Our operational framework is based on Milner's Calculus of Communicating Systems (CCS) [15] and as such is a common process algebra. It generalises our earlier process algebra PAi [4], [5], by allowing for multiple appearances of actions and its semantics by incorporating the concepts of lie and concealment. The grammar for our syntax is given below.

$$\begin{array}{lll} Proc & ::= & 0 \mid D\,;Proc \mid Proc + Proc \mid Proc \parallel Proc \mid X \\ D & ::= & (\alpha, f) \mid (\beta, f)! \mid \beta? \\ f \in \mathcal{F} & & \alpha \in A, \beta \in A \setminus \{\tau\} \end{array}$$

In this grammar $Proc$ represents the syntactic class of processes, ranged over by $p, q, p_0, \ldots$, D represents the syntactic class of decorated actions, ranged over by $d, d_0, \ldots$, and α represents the class A of (atomic) actions, ranged over by $a, a_0, \ldots$. $\mathcal{F}$ denotes the set of all functions such as $f : A \times Id \to A$, where Id is a set of principal identifiers ranged over by natural numbers in this paper. This function denotes the different types of announcements (including lies and concealments) that could be made throughout a protocol. A particular atomic action, designated with τ, denotes internal actions, which are supposed to carry no information (apart from unobservable evolution of the internal state of the system). This particular action is used to model concealing the occurrence of other atomic actions; this becomes clearer in the remainder of our explanation below. In this section, we do not impose further conditions on the function f in the syntax, it can map any action to any other action (including itself); however, as we show in Section V, for a protocol to make intuitive sense, some sanity conditions should be checked on the types of renamings defined by such functions in the protocol.

A process can be the terminated process 0, or an action-prefixed process $d\,;p$, in which first $d \in D$ takes place and subsequently p takes over, or a nondeterministic choice $p + q$ between p and q, in which the first action from p or q resolves

the choice, or a parallel composition $p \,||\, q$, in which p and q can interleave their actions or synchronize on the dual send and receive actions (see below). Moreover, recursive variables can be used to specify infinite behavior; each recursive variable is assumed to have a unique recursive equation of the form $X \doteq Proc$.

A decorated action d is a pair (a, f), which comprises an atomic action a, which is used to model communications and internal actions in a protocol, and a renaming function f, which specifies the action that is perceived by each principal when action a happens (This notion is inspired by the notion of function views used in [20].)

Intuitively, in such decorated action, each principal $i \in Id$ will perceive the message carried in a as $f(a, i)$. Particularly for an action a, and a principal $i \in Id$ such that $f(a, i) = b$ and $b \neq a$ is deceived by observing b instead of a. In this case if $b = \tau$, then i does not notice any action while a has been performed. For a decorated action (a, f), we say that $I \subseteq Id$ is the set of intended audience of the action, where I is the maximal set satisfying $\forall_{i \in I}\ f(a, i) = a$.

Atomic actions can be of the form $(a, f)!$, denoting sending a, $a?$, denoting receiving a, or simply (a, f), denoting the result of synchronization (handshake communication) on a. The set $|D| \subseteq D$ is the set of all actions that are the result of a synchronization, i.e., are not followed by ? or !. As stated before, $\tau \in A$ is designated as the unobservable action, which in our framework models perfect information hiding. In other words, if a decorated action is of the form (a, f) and for $i \in Id$, $f(a, i) = \tau$ then after performing a, it appears to the principal i as if no action has taken place.

Note that we have consciously opted for a simple syntax in which we do not code the intention of principals (e.g., the identifier principal who tells a lie); in keeping with this choice, we do not allow for explicit reasoning about the intentions of principals. We consider the extension to the setting with subject principals and their intentions among our directions for future work.

Example 2: Liar's Coin in process algebra. The following specification models, in our syntax, the normal execution of the Liar's Coin game explained in Example 1. We assume $A = \{h, t, p, c, kr{-}10, kr{-}20, kr10, kr20\}$ is the set of actions and that $Id = \{1, 2\}$ denotes the set of players. (For the sake of brevity, we have skipped the step of selecting the bet and putting up the stake; we assume that player 1 bets on head and player 2 bets on tail.)

$$
\begin{array}{l}
Game1 = \\
(h, f)\,;(((p, f)\,;(kr{-}10, f)) + \\
((c, f)\,;(kr{-}20, f))) + (h, f')\,;(kr10, f) + \\
(t, f)\,;(kr10, f) + (t, f')\,;(((p, f)\,;(kr{-}10, f)) + \\
((c, f)\,;(kr20, f)))
\end{array}
$$

$$
\begin{array}{l}
\forall_{a \in A, i \in Id}\ f(a, i) = a \\
f'(h, 1) = h\ \wedge\ f'(t, 1) = t\ \wedge \\
\forall_{i \in Id}\ i \neq 1 \Rightarrow f'(h, i) = t\ \wedge\ f'(t, i) = h
\end{array}
$$

In this specification, action h or t is first performed by principal 1, but nondeterministically, it is chosen whether the truth or a lie is told to the rest. The Player 1 in all cases sees the true result of the coin flip. Player 2, as well as other principals in Id, observe the message announced by player 1, which may be the truth or a lie, represented by functions f and f', respectively. It is important to note that for the sake of brevity in the above-given specification, the act of flipping a coin and telling the truth or lying have been merged into one atomic action. A more elaborate specification could separate them by firstly denoting the event of flipping a coin and secondly making a decision to tell the truth or a lie. Moreover, we assumed that the decision to pass or challenge and the amount and the person who wins the stake are all announced publicly.

The following specification is a variant of the game, where player 1 does not lie when the result of the coin flip is head. Note that in both cases player 1 does observe the actual result of the coin flip (1 is among the intended audience of h and t actions in all cases), but in some cases, she consciously decides to lie.

$$
\begin{array}{l}
BetterGame1 = \\
(h, f)\,;(((p, f)\,;(kr{-}10, f)) + ((c, f)\,;(kr{-}20, f))) + \\
(t, f)\,;(kr10, f) + (t, f')\,;(((p, f)\,;(kr{-}10, f)) + \\
((c, f)\,;(kr20, f)))
\end{array}
$$

$$
\begin{array}{l}
\forall_{a \in A, i \in Id}\ f(a, i) = a \\
f'(t, 1) = t\ \wedge\ \forall_{i \in Id}\ i \neq 1 \Rightarrow\ f'(t, i) = h
\end{array}
$$

B. Epistemic framework

Our logical framework is a natural combination of modal μ-calculus with past, on the one hand, and epistemic logic (of beliefs), on the other hand. The inclusion of past is crucial in the applicability of our logic, since often belief concerns actions in the past, e.g., after the completion of the protocol an agent believes that a certain action has taken place in the past. (For examples of such phenomena, we refer to the forthcoming formalization of logical properties of Liar's Coin in the remainder of this section.) The grammar of logical formulae is given below.

$$
\phi ::= \top \mid Y \mid \bigwedge_{j \in J} \phi_j \mid \neg\phi \mid \langle\alpha\rangle\phi \mid \langle\overline{\alpha}\rangle\phi \mid B_i\phi \mid \nu Y.\phi(Y) \qquad \textit{(if } Y \textit{ occurs only positively in } \phi\textit{)},
$$

In the above-given grammar, logical connectives have their traditional intuitive meanings; Y stands for the class of recursive variables ranged over by $x, y, x_0, \ldots$; $\langle a\rangle\phi$, means that it is possible to perform an a-transition, after which formula ϕ holds; $\langle\overline{a}\rangle\phi$, means that in the immediate past an a-transition has been made, before which formula ϕ holds; $B_i\phi$ means that principal i believes in ϕ and $\nu Y.\phi(Y)$ denotes the maximal fixed point of the equation $Y = \phi(Y)$.

In order to facilitate the specification of logical properties, we define and use the following acronyms:

$$\begin{array}{lcl} [a]\phi & \doteq & \neg\langle a\rangle\neg\phi \\ \bigvee_{j\in J}\phi_j & \doteq & \neg\bigwedge_{j\in J}\neg\phi_j \\ \mu Y.\phi & \doteq & \neg\nu Y.\neg\phi \text{ with } \neg Y = Y \\ AG\ \phi & \doteq & \nu Y.\phi \wedge (\bigwedge_{a\in A}\langle a\rangle Y) \\ EF\ \phi & \doteq & \neg AG\neg\phi \\ \phi^{\curvearrowleft} & \doteq & \mu Y.\phi \vee (\bigvee_{a\in A}\langle \overline{a}\rangle Y) \end{array}$$

To explain briefly, $[a]\phi$ means that after all a actions ϕ holds, disjunction is known from propositional logic, $\mu Y.\phi$ stands for the minimal fixed point (often used to express that a formula holds in after/before a finite path, see, e.g., the definition of $\phi^{\curvearrowleft}$), $AG\ \phi$ means that ϕ holds everywhere, $EF\ \phi$ means that there is a finite run after which ϕ will eventually hold, and $\phi^{\curvearrowleft}$ means that somewhere in the past ϕ held.

Example 3: Liar's Coin's logical properties. Below we give a formalization of the logical properties stated in Example 1.

- After player 2 passes (p), she does not form any belief regarding the coin flip.

$$AG\ \ [p](\neg B_2\ (\langle\overline{t}\rangle\top)^{\curvearrowleft}) \wedge (\neg B_2\ (\langle\overline{h}\rangle\top)^{\curvearrowleft})$$

- After that player 2 challenges, she does form a belief regarding the past result of the coin flip.

$$AG\ \ [c](B_2\ (\langle\overline{t}\rangle\top)^{\curvearrowleft}) \vee (B_2\ (\langle\overline{h}\rangle\top)^{\curvearrowleft})$$

- Player 1 can deceive player 2, i.e., player 1 may see a head while player 2 believes it has been a tail (or vice versa).

$$\langle t\rangle\ EF\ (B_2\ (\langle\overline{h}\rangle\top)^{\curvearrowleft}) \vee \langle h\rangle\ EF\ (B_2\ (\langle\overline{t}\rangle\top)^{\curvearrowleft})$$

Similar properties could be specified from the view point of the external observer 0, but due to space limitations we dispense with those.

III. Semantics

A. Operational semantics

The operational semantics of our process algebra is a slight variation of the standard Plotkin-style semantics [17] for process algebra [2] by including a sequence of past actions (to build up the epistemic aspect).

Definition 4 (Structural operational semantics): Transition relation $\xrightarrow{d} \subseteq (Proc \times D^*)^2$ is the smallest relation satisfying the following deduction rules (for the sake of brevity, we have omitted the symmetric versions of deduction rules $(\mathbf{n0})$, $(\mathbf{p0})$, and $(\mathbf{p2})$). Note that transition relation $\xrightarrow{d}$ is defined using the auxiliary transition relation $\overset{d}{\Rightarrow}$. The main difference between the two is that $\overset{d}{\Rightarrow}$ allows for individual send and receive actions to take place, while $\xrightarrow{d}$ enforces synchronization by filtering out individual send and receive actions (due to deduction rule $(\mathbf{sync})$).

$$(\mathbf{a})\frac{}{(d\,;p,\pi)\overset{d}{\Rightarrow}(p,\pi\frown d)}\quad (\mathbf{n0})\frac{(x_0,\pi)\overset{d}{\Rightarrow}(y_0,\pi')}{(x_0+x_1,\pi)\overset{d}{\Rightarrow}(y_0,\pi')}$$

$$(\mathbf{p0})\frac{(x_0,\pi)\overset{d}{\Rightarrow}(y_0,\pi')}{(x_0\,||\,x_1,\pi)\overset{d}{\Rightarrow}(y_0\,||\,x_1,\pi')}\quad (\mathbf{sync})\frac{(x,\pi)\overset{(a,f)}{\Rightarrow}(y,\pi')}{(x,\pi)\overset{(a,f)}{\rightarrow}(y,\pi')}$$

$$(\mathbf{r})\frac{(p[p/X],\pi)\overset{d}{\Rightarrow}(y,\pi')}{(X,\pi)\overset{d}{\Rightarrow}(y,\pi')}\ X\doteq p$$

$$(\mathbf{p2})\frac{(x_0,\pi)\overset{a?}{\Rightarrow}(y_0,\pi')\quad (x_1,\pi)\overset{(a,f)!}{\Rightarrow}(y_1,\pi'')}{(x_0\,||\,x_1,\pi)\overset{(a,f)}{\Rightarrow}(y_0\,||\,y_1,\pi\frown(a,f))}a\neq\tau$$

Where $\frown$ is the concatenation operator. Deduction rules $(\mathbf{a})$, $(\mathbf{n0})$, and $(\mathbf{p0})$ are self explanatory; deduction rule $(\mathbf{r})$ specify the behavior of a recursive variable X in terms of the behavior of its definition p. Deduction rule $(\mathbf{p2})$ specifies the synchronization of a send and a receive action. We define that the two synchronizing parties should not only synchronize on the actual message, but also on the possible lie that is to be told to the unintended recipients. In our framework, since each action has a single appearance, defining other semantics for synchronisation with respect to the appearance would not be straightforward. For other semantic frameworks that allow for combining appearance functions we refer to [23]. Deduction rule $(\mathbf{sync})$ specifies synchronization over decorated actions. The operational semantics of a process is the smallest transition relation $\overset{(a,f)}{\rightarrow}$ satisfying the deduction rules given above, with the pair of the process and the empty history as the initial state. It is crucial to note that the transition relation $\overset{(a,f)}{\rightarrow}$ does not allow for individual (not synchronised) send and receive to appear in the semantics of a process.

In order to develop the epistemic aspect of our operational semantics, we start with defining an accessibility relation. This relation specifies for each principal, which runs are suspected to have happened after having seen a run of the protocol.

Definition 5 (Accessibility relation): The accessibility relation $\overset{i}{\cdots\blacktriangleright}\subseteq D^*\times D^*$ is the smallest relation defined by the

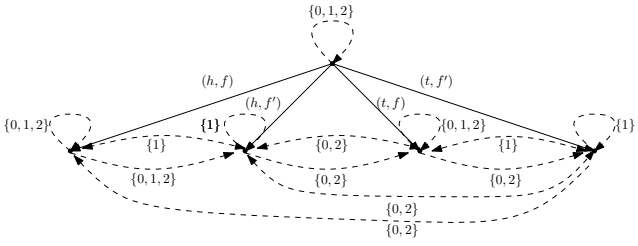

Fig. 1. First steps in the operational semantics of $Game1$.

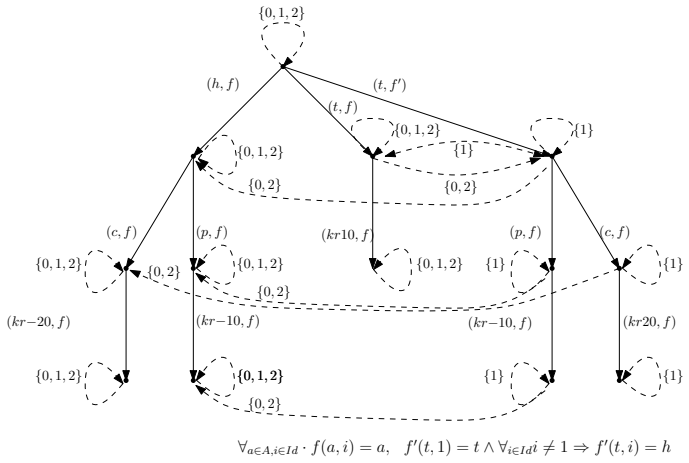

Fig. 2. Operational semantics of $BetterGame1$ in Liar's Coin example.

following deduction rules.

$$(\epsilon)\frac{}{\epsilon \overset{i}{\cdots\blacktriangleright} \epsilon} \qquad (\mathbf{tru})\frac{\pi \overset{i}{\cdots\blacktriangleright} \pi' \quad f(a,i)=a}{\pi \frown (a,f) \overset{i}{\cdots\blacktriangleright} \pi' \frown (a,f')}$$

$$(\mathbf{lie})\frac{\pi \overset{i}{\cdots\blacktriangleright} \pi' \quad f(a,i)=b}{\pi \frown (a,f) \overset{i}{\cdots\blacktriangleright} \pi' \frown (b,f')}$$

$$(\mathbf{hid0})\frac{\pi \overset{i}{\cdots\blacktriangleright} \pi' \quad f(a,i)=\tau}{\pi \frown (a,f) \overset{i}{\cdots\blacktriangleright} \pi'} \qquad (\mathbf{hid1})\frac{\pi \overset{i}{\cdots\blacktriangleright} \pi' \quad f'(a,i)=\tau}{\pi \overset{i}{\cdots\blacktriangleright} \pi' \frown (a,f')}$$

$$(\mathbf{hid2})\frac{\pi \overset{i}{\cdots\blacktriangleright} \pi'}{\pi \frown (\tau,f) \overset{i}{\cdots\blacktriangleright} \pi'} \qquad (\mathbf{hid3})\frac{\pi \overset{i}{\cdots\blacktriangleright} \pi'}{\pi \overset{i}{\cdots\blacktriangleright} \pi' \frown (\tau,f')}$$

Deduction rule (ϵ) is self-explanatory. Deduction rule $(\mathbf{tru})$ specifies that the intended audience of a decorated action always observe it as it is and hence, they can only suspect another run if it ends with the same observable action. There is a built-in asymmetry in deduction rule $(\mathbf{tru})$, namely, we only require $f(a,i)=a$ and not $f'(a,i)=a$. The asymmetry is introduced to model credulous agents that know the actual protocol: the agents believe what is told to them (unless it leads to contradiction) based on what they expect in a truthful execution of the protocol. In other words, if they observe an a, they believe they may be in a run of the protocol where a actually happens regardless whether they may be told the truth in that run or not (i.e., regardless whether $f'(a,i)=a$ or not, respectively). Deduction rule $(\mathbf{lie})$ specifies that an action a may be pretended to have been a b, if the observing agent is not among the intended audience of a and also expects an observable b to happen in another plausible and indistinguishable run. Note that since principals are assumed to know the protocol beforehand, even if they do not observe the other runs of the protocol, e.g., the run in which b happens, they do know of its possibility and hence suspect it to lead to the current state. Deduction rules $(\mathbf{hid0})$ and $(\mathbf{hid1})$ state that an action a may be hidden from an agent that is not among its intended audience, if the perceived appearance of the action is τ, i.e., the unobservable action. Deduction rules $(\mathbf{hid2})$ and $(\mathbf{hid3})$ state that unobservable actions in the protocol are believed to be unobservable by all agents.

Definition 6 (Operational semantics of a process): Given a process $p_0 \in Proc$, the indistinguishability relation $\overset{i}{\text{-}\!\blacktriangleright}\subseteq (Proc\times D^*)^2$ for p_0 is the smallest relation satisfying the following deduction rule:

$$(\mathbf{ind})\frac{(p_0,\epsilon)\rightarrow^*(p,\pi) \quad (p_0,\epsilon)\rightarrow^*(p',\pi') \quad \pi \overset{i}{\cdots\blacktriangleright} \pi'}{(p,\pi) \overset{i}{\text{-}\!\blacktriangleright} (p',\pi')}$$

where $\rightarrow^*$ is the reflexive and transitive closure $\bigcup_{d\in|D|}\overset{d}{\rightarrow}$.

The semantics of a process $p_0 \in Proc$ is defined as a labeled transition relation with (p_0,ϵ) as the initial state, and $(\bigcup_{d\in|D|}\overset{d}{\rightarrow})\cup(\bigcup_{i\in Id}\overset{i}{\text{-}\!\blacktriangleright})$ as its transition relations.

Example 7: Liar's Coin's logical properties. In Figures 1 and 2, respectively, the semantics of $Game1$ before and after the first action of the protocol and the complete semantics of $BetterGame1$ are depicted. In both figures, solid lines denote the decorated actions and dashed lines denote the indistinguishability relation. In order not to clutter the figures, we have left out the configurations of our operational semantics and represented them by black dots.

B. Semantics of Epistemic Logic

The semantics of our epistemic logic follows the common existing semantics and has no peculiarities. Namely, the semantics is verified with respect to the labeled transition system L for processes defined above and a given current state s. The semantic definitions for various constructs of our logic are given below.

L, s	$\models \top$	iff	true
L, s	$\models \bigwedge_{j\in J}\phi_j$	iff	$\mathrm{L}, s \models \phi_j$ for each $j\in J$
L, s	$\models \neg\phi$	iff	$\mathrm{L}, s \models \phi$ is not true
L, s	$\models \langle a\rangle\phi$	iff	there is an $s'\in Proc\times D^*$, $s \overset{(a,f)}{\rightarrow} s'$ and $\mathrm{L}, s' \models \phi$
L, s	$\models \langle\overline{a}\rangle\phi$	iff	there is an $s'\in Proc\times D^*$, $s' \overset{(a,f)}{\rightarrow} s$ and $\mathrm{L}, s' \models \phi$
L, s	$\models B_i\ \phi$	iff	for all $s'\in Proc\times D^*$ such that $s' \overset{i}{\text{-}\!\blacktriangleright} s$ it holds that $\mathrm{L}, s' \models \phi$
L, s	$\models \nu X.\phi(X)$	iff	$s\in\bigcup\{S'\subseteq Proc\times D^* \mid \forall_{s'\in S'}\mathrm{L}, s' \models \phi(X:=S')\}$

The semantics of $\top$, conjunction, negation, and maximal fixed point are standard. The semantics of $\langle a\rangle\phi$ specifies that from the current state s an a-labeled transition (regardless of its perceived action) emanates leading to a state s' satisfying ϕ. Dually, $\langle \overline{a}\rangle\phi$ specifies that from the current state s an incoming a-labeled transition can be rewound arriving in a state s' satisfying ϕ. The semantics of the belief operator $B_i\ \phi$ specifies that for all indistinguishable states s' (that are reachable from the initial state), ϕ should hold.

Example 8: Checking the properties of $BetterGame1$. Consider the epistemic properties specified in Example 3 and the operational semantics of process $Better\ Game1$ depicted in Figure 2. Next, we briefly reason which properties hold for the process.

- Consider the following property:

  $$AG\ \ [p](\neg B_2\ (\langle\overline{t}\rangle\top)^{\urcorner})\wedge(\neg B_2\ (\langle\overline{h}\rangle\top)^{\urcorner})$$

  In states without outgoing p transitions, the property holds trivially. Hence, only the two states with an outgoing p transition are to be checked. In both cases, the target of the p-labeled transitions are only related to the state in which an h has actually happened in the past and hence, principal 2 does believe that an h has happened in the past. Thus, the property does not hold for the initial state of $BetterGame1$, i.e., after passing, principal 2 does form a belief in this case.
- Regarding the second property:

  $$AG\ \ [c](B_2\ (\langle\overline{t}\rangle\top)^{\urcorner})\vee(B_2\ (\langle\overline{h}\rangle\top)^{\urcorner})$$

  following a similar reasoning to the previous case, the only accessible state from the targets of the two c-labeled transitions is the leftmost state which has an h-labeled transition in its past. Hence, after all c transitions, principal 2 does believe that an h has happened. Note that after revealing the coin and exchanging the stakes, if 2 receives $kr20$ (in the bottom rightmost state), then she ceases to hold this belief.
- Finally, concerning the third property:

  $$\langle t\rangle\ EF\ (B_2\ (\langle\overline{h}\rangle\top)^{\urcorner})\vee\langle h\rangle\ EF\ (B_2\ (\langle\overline{t}\rangle\top)^{\urcorner})$$

  we have already seen in the previous two items that at various states, namely the target states of p- and c-labeled transitions in the leftmost branch, as well as the state following the $kr{-}10$ principal 2 believes that an h has happened in the past while actually a t has happened.

IV. Merging the Two Worlds

In this section, we lay the foundation of a framework in which beliefs can integrated into the process language and hence, influence the execution of a protocol. To this end, we extend the syntax of our process algebra with a conditional belief operator as follows.

$$\begin{array}{lll} Proc & ::= & 0 \mid \mathbf{B_i}\psi \Rightarrow Proc \mid D\,;Proc \mid Proc + Proc \mid \\ & & Proc\ ||\ Proc \mid X \\ \psi & ::= & \top \mid X \mid \bigwedge_{j\in J}\psi_j \mid \neg\psi \mid \langle\overline{\alpha}\rangle\psi \mid \nu X.\psi(X) \end{array}$$

The new piece of syntax, designated in bold, is $\mathbf{B_i}\psi \Rightarrow Proc$, which reads "if principal i believes that ψ is the case, then $Proc$ follows (or otherwise the process deadlocks)". Note that we have restricted the syntax of the logical formula ψ to only include past modalities. This is to exclude pathological specifications such as the following: if i believes an a action is disabled, then an a action should be performed. (Such processes can be given an unambiguous semantics thanks to approaches such as [3], [10], but we avoid them in order to avoid the rather complicated semantic machinery required to disambiguate such self-referential specifications.)

The semantics of our new operator is given below. (The rest of the deduction rules for the operational semantics remain intact.)

$$(\mathbf{belief})\frac{\mathrm{L},(x,\pi)\models B_i\ \psi \quad (x,\pi)\stackrel{d}{\Rightarrow}(y,\pi')}{(\mathbf{B_i}\psi\Rightarrow x,\pi)\stackrel{d}{\Rightarrow}(y,\pi')}$$

The above-given rule specifies that if formula $B_i\ \psi$ holds with respect to the current state (and the operational semantics of the initial process), then the behavior of the argument x determines the behavior of the conditional process term $\mathbf{B_i}\psi\Rightarrow x$.

This new piece of syntax allows for compositional specifications in which the components of a parallel process specify each principal's role in the protocol. The following example illustrates this fact.

Example 9: Compositional Specification of Liar's Coin. In the following specification, we first decompose the specification of Liar's Coin into the specification of the scenarios played by the two players. The specification of the game is obtained by the parallel composition of the two player's scenarios.

$$\begin{array}{l} Player1 = (h,f)\,;((p?\,;(kr{-}10,f)!)\ + \\ ((c,f)!\,;kr{-}20?)) + (t,f)\,;kr10?\ + \\ (t,f')\,;(((p,f)!\,;(kr{-}10,f)!) + ((c,f)!\,;(kr20,f)!)) \end{array}$$

$$\begin{array}{l} Player2 = \mathbf{B_2}(\langle\overline{h}\rangle\top)^{\urcorner} \Rightarrow (c?\,;kr20?)+ \\ (p?\,;kr{-}10?) \end{array}$$

$$Game = Player1\,||\,Player2$$

$$\begin{array}{l} \forall_{a\in A,i\in Id}\ f(a,i)=a \\ f'(t,1)=t\ \wedge\ \forall_{i\in Id}\ i\neq 1 \Rightarrow\ f'(t,i)=h \end{array}$$

In the above-given specification, $Player1$ plays the same scenario as specified in $BetterGame1$. However, $Player2$ is only willing to accept the challenge if she believes that the result of coin flip has been a head. Otherwise, $Player2$ is always willing to accept the pass message.

To illustrate the role of our new belief operator, assume that $Player1$ lies. This corresponds to the rightmost transition in Figure 2 and the third summand of the process $Player1$ given above. (Since the behavior of $Player1$ is identical to

$BetterGame1$, we can still refer to Figure 2 to explain the initial steps of this example.) In this case, $Player2$ believes that the result of the coin flip has been a head and hence, accepts the challenge. Based on her earlier belief, she is only willing to receive 20 kronor afterwards. In this scenario $Player1$ has to accept that her lie has not been successful and hence, pays 20 kronor to $Player2$.

V. Detecting Lies

In this section, we formulate two properties of processes, called semantic consistency and syntactic consistency, that guarantee the semantics of a process to satisfy the conditions of a belief model (called $\mathcal{K}$D45, cf. [9]). These two properties characterise the set of protocols in which the agents can consciously form a logically consistent belief after all runs of such protocols. The conditions guaranteeing the properties are sanity conditions of our framework, i.e., if they turn out to be intuitive properties about protocols, then we have confidence that the semantics of our indistinguishability relation is intuitive correct.

The process of coming up with these conditions has been very laborious and involved several iterations over the proofs and the semantics. We first attempted a proof of these properties (involving more than 50 case distinctions) and gathered all the conditions, hereafter called syntactic consistency, that were necessary to make the proof go through. Analysed them against our intuition of a consistent protocol. In case any such condition could not be intuitively motivated, we analysed their causes in terms of the semantics of indistinguisability relation, revised the semantics and re-did the proof. This process led to the existing semantics and the proofs (omitted in the conference publication, due to space limitation). Finally, we gathered all the semantic conditions and summarised them in terms of an easy-to-check and intuitive sufficient condition, called strict syntactic consistency. The latter could be easily built into an automatic type-checking system for protocols to check whether that the agents involved in the protocol can form a logically consistent belief in all its runs.

Semantic consistency is an essential condition which requires that each observed (partial) run of the protocol has a corresponding possible trace in the specification according to each and every principal.

Definition 10 (Semantic consistency): A process p_0 is semantically consistent when for each $(p, \pi) \in Proc \times D^*$ and each $i \in Id$ such that $(p_0, \epsilon) \rightarrow^* (p, \pi)$, there exists a $(p', \pi') \in Proc \times D^*$ such that $(p_0, \epsilon) \rightarrow^* (p', \pi')$ and $(p, \pi) \overset{i}{\rightarrowtail} (p', \pi')$.

For example, neither processes $Game1$, nor $BetterGame1$ is semantically consistent. In both games, if the result of the coin flip for player 1 is tail, but she announces head and player 2 challenges, then she cannot have a consistent belief about the announcements made so far (and hence, detects that an untruthful announcement has been made). However, if the leftmost $(kr-20, f)$ transition is removed from process $Better\ Game1$, it becomes a consistent process. (After having made this change, for any arbitrary principal, each run of the system has at least one corresponding run in the model.)

Syntactic consistency is another important condition to identify those models of our process algebra that satisfy the properties of a belief model. This property consists of a set of syntactic constraints, which are necessary for a process to satisfy conditions of a belief model.

Definition 11 (Syntactic consistency): A process p, with the set of decorated actions D, is syntactically consistent when it satisfies the following syntactic constraints:

(C1): $\forall_{(a,f),\ (a,f'),\ (b,f'')\in D}\ f(a,i) = a \wedge f'(a,i) = b \Rightarrow (a = b) \vee (a = \tau \wedge f''(b,i) = \tau)$

(C2): $\forall_{(a,f),\ (a,f')\in D}\ f(a,i) = a \wedge f'(a,i) = \tau \Rightarrow a = \tau$

(C3): $\forall_{(a,f),\ (b,f'),\ (c,f'')\in D}\ f(a,i) = b \wedge f'(b,i) = c \Rightarrow (b = c) \vee (b = \tau \wedge f''(c,i) = \tau)$

(C4): $\forall_{(a,f),\ (b,f')\in D}\ f(a,i) = b \wedge f'(b,i) = \tau \Rightarrow (a = \tau) \vee (b = \tau)$

(C5): $\forall_{(\tau,f)\in D}\ f(\tau,i) = b \Rightarrow (b = \tau) \vee (f'(b,i) = \tau)$

(C6): $\forall_{(a,f),\ (a,f'),\ (a,f'')\in D}\ f(a,i) = a \Rightarrow (f'(a,i) = a) \vee (a = \tau) \vee (f'(a,i) = \tau \wedge f''(a,i) = \tau)$

(C7): $\forall_{(a,f),\ (b,f'),\ (b,f'')\in D}\ f(a,i) = b \Rightarrow (f'(b,i) = b) \vee (f'(b,i) = \tau \wedge f''(b,i) = \tau) \vee (b = \tau)$

These items are obtained by trying to prove semantic consistency using the deduction rules of the Definition 5. By and large, these items constrain the possibility of telling different lies or telling lies about lies. For example constraint **C1** states that if a principal observes the truth about an action, then she is not lied about it anywhere else in the protocol; the only possibility is that the same actions may be concealed. Constraint **C3** states that if two lies are told about an action to a principal, then either the lies are identical, or one of the lies is just a concealment and any other lie about the same action is a concealment as well. Constraint **C4** states that if a lie is told about an action to a principal and the lie is concealed somewhere in the principal's communications, then either the action is unobservable or the lie has been a concealment. The rest of the constraints can be interpreted similarly.

Although these constraints are intuitively easy to understand, they can be collectively expressed in terms of a stronger sufficient condition for consistency, called strict syntactic consistency, defined below.

Definition 12 (Strict syntactic consistency): A process p, with the set of decorated actions D, is strictly syntactically consistent when it satisfies the following syntactic constraint:

(C0) : $\forall_{(a,f),\ (b,f')\in D}\ f(a,i) = b \Rightarrow f'(b,i) = b \wedge a = \tau \Rightarrow b = \tau$

The following lemma states paves the way for using syntactic consistency instead of semantic consistency in protocol analysis.

Lemma 13: A strictly syntactically consistent process is also syntactically consistent.

(The proof of the lemma is included in the appendix.)

Next, we recall the following basic concept from the theory of epistemic reasoning.

Definition 14 (Belief properties): For a set A, the relation $R \subseteq A \times A$ satisfies the belief properties, also called $\mathcal{K}$D45, when it is:

1) Transitive: for each $a, b, c \in A$, if $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$, and
2) Euclidean: for each $a, b, c \in A$, if $(a, b) \in R$ and $(a, c) \in R$, then $(b, c) \in R$, and
3) Serial: for each a, there exists a b such that $(a, b) \in R$.

The above-mentioned belief properties can be intuitively explained as follows (for a more formal discussion of these properties, we refer to [9, Chapter 3]). Transitivity implies that if a principal believes that ϕ holds, then she is also conscious of this believe (i.e., she believes that she believes in ϕ). Euclidean property can be considered as the dual to transitivity, namely it implies that if principal i does not believe in ϕ, then she is conscious about this lack of belief (i.e., she believes that she does not believe in ϕ). The serial property implies consistency in belief, i.e., one does not believe in falsity.

The following theorem states that semantic consistency and syntactic consistency are sufficient for satisfying belief properties.

Theorem 15: For each semantically consistent and syntactically consistent process p, and each $i \in Id$, where Id denotes the set of principal identifiers, relation $\text{-} \xrightarrow{i} \rightarrow$ induced by the semantics of p satisfies the belief properties.

(The proof of the theorem is included in the appendix)

Given Lemma 13, it follows from the proof of the above theorem that each semantically consistent and strictly syntactically consistent process also satisfies the belief properties.

VI. Conclusions

In this paper, we presented a process algebraic formalism for specifying protocols in which untruthful announcements can be made. We provided the semantics of this process algebra in terms of a combination of labeled transition systems and pointed Kripke structures. We showed how a combination of modal μ-calculus and dynamic epistemic logic can be used to reason about the correctness of such protocols. We also defined conditions, called semantic consistency and syntactic consistency, that make the semantics of a process qualify as a belief model.

As suggested by its name, semantic consistency is a semantic property, while syntactic constancy is a syntactic condition. We are currently investigating sufficient syntactic conditions that imply semantic consistency. Defining a suitable notion of bisimulation and axiomatizing the process algebra are among our other future research directions.

A translation between the public announcement subset of our framework and that of [13], [7] is worth further investigating. This can be achieved by considering an existing translation of labeled transition systems to Kripke structure such as the one proposed in [6]. It then remains to lift the translation into a translation from labeled transition systems to pointed Kripke structures. Making the intentions of principals explicit and reasoning about them is another area of future research. Finally, we would like to compare the expressiveness of our operational framework with and without the belief operator.

References

[1] A. Baltag, L.S. Moss, and S. Solecki. The Logic of Public Announcements, Common Knowledge and Private Suspicions. Proc. of TARK'98, 43–56. Morgan Kaufmann, 1998.

[2] J.C.M. Baeten, T. Basten, and M.A. Reniers. Process Algebra: Equational Theories of Communicating Processes, Cambridge, 2010.

[3] R. Bol and J.F. Groote. The Meaning of Negative Premises in Transition System Specifications, J. ACM 43(5):863–914, 1996.

[4] F. Dechesne, M.R. Mousavi, and S. Orzan, Operational and Epistemic Approaches to Protocol Analysis: Bridging the Gap. Proc. of LPAR'07, 226–241. LNCS 4790, Springer, 2007.

[5] F. Dechesne and M.R. Mousavi. Interpreted Systems Semantics for Process Algebra with Identity Annotations. In Proc. of TbiLLC'11, 182–205. LNCS 7758, Springer, 2013.

[6] R. De Nicola and F.W. Vaandrager. Action versus State based Logics for Transition Systems. Semantics of Systems of Concurrent Processes, 407–419, LNCS 469, Springer, 1990.

[7] H.P. van Ditmarsch. Dynamics of lying, Synthese 191: 745–777, 2014.

[8] H.P. van Ditmarsch, J. van Eijck, F. Sietsma, and Y. Wang, On the logic of lying . In GASS, pages 41-72. LNCS 7010, Springer, 2012.

[9] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. Reasoning About Knowledge. MIT Press, 2004

[10] R.J. van Glabbeek. The Meaning of Negative Premises in Transition System Specifications II. JLAP 60-61:229–258, 2004.

[11] M. Guzman, S. Haar, S. Perchy, C. Rueda, and F, Valencia. Belief, Knowledge, Lies and Other Utterances in an Algebra for Space and Extrusion. JLAMP, 86(1): 107-133, 2017

[12] J.Y. Halpern and Y. Moses. Knowledge and Common Knowledge in a distributed environment. J. ACM, 37(3):549–587, 1990

[13] B. Kooi and B. Renne. Arrow Update Logic. The Review of Symbolic Logic 4(4):536–559, 2011.

[14] S. Knight, C. Palamidessi, P. Panangaden, and F.D. Valencia: Spatial and Epistemic Modalities in Constraint-Based Process Calculi. In Proc. of CONCUR'12, 317–332, LNCS 7454, Springer, 2012.

[15] R. Milner. A Calculus of Communicating Systems. LNCS 92, Springer, 1980.

[16] R. Parikh and R. Ramanujam. Distributed processes and the logic of knowledge. Proc. of CLP'85, volume 193 of LNCS, pages 256–268. Springer, 1985.

[17] G.D. Plotkin. A structural approach to operational semantics. JLAP 60:17–139, 2004.

[18] R. Pucella. Knowledge and Security, CoRR abs/1305.0876, 2013. Available from http://arxiv.org/abs/1305.0876.

[19] C. Sakama. Logical Definitions of Lying. Proc. of TRUST'11, 2011.

[20] D. Hughes and V. Shmatikov. Information hiding, anonymity and privacy: A modular approach. Journal of Computer Security, 12(1):3–36, 2004.

[21] M. Clavel and F. Durán and S. Eker and P. Lincoln and N. Martí-Oliet and J. Meseguer and J. Quesada. Maude: Specification and Programming in Rewriting Logic, SRI International, 285(2):187 - 243, 1999.

[22] A. Lomuscio, H. Qu, F. Raimondi, A. Bouajjani, O. Maler. MCMAS: A Model Checker for the Verification of Multi-Agent Systems, Proc. of CAV'21, 682–688, Springer, 2009.

[23] M. Sadrzadeh. Actions and Resources in Epistemic Logic, PhD Thesis, 2006.