A Complete Proof System for a Dynamic Epistemic Logic Based upon Finite π-Calculus Processes

Eric Ufferman [1]

Department of Mathematics, School of Science, National University of Mexico (UNAM)
Circuito exterior, Ciudad Universitaria. C.P. 04510
México D.F., México

Pedro Arturo Góngora [1]

Postgraduate Program in Computer Science, National University of Mexico (UNAM)
Circuito exterior, Ciudad Universitaria. C.P. 04510
México D.F., México

Francisco Hernández-Quiroz [1]

Department of Mathematics, School of Science, National University of Mexico (UNAM)
Circuito exterior, Ciudad Universitaria. C.P. 04510
México D.F., México

Abstract

The pi-calculus process algebra describes the interaction of concurrent and communicating processes. In this paper we present the syntax and semantics of a dynamic epistemic logic for multi-agent systems, where the epistemic actions are finite processes in the pi-calculus. We then extend the language to include actions from a specified set of action structures. We define a proof system for the extended language, and prove the completeness of the proof system. Thus any valid formula in the original language without action structures can be proved in the proof system for the extended language.

Keywords: Dynamic Epistemic Logic, pi-Calculus, Completeness

[1] The authors received support for this paper from the grant PAPIIT IN109010.

1 Background

1.1 Epistemic Logic

Epistemic logic is a branch of modal logic concerned with reasoning about the knowledge of agents. Given a countable set of proposition symbols $\mathcal{P}$, and a finite set of agents $\mathcal{A}$, formulas of the basic epistemic $\mathcal{L}_E$ logic are formed using the usual connectives and formation rules for formulas of propositional logic, with the additional proviso that for any formula φ, and agent $a \in \mathcal{A}$, $K_a\varphi$ is also a formula. The formula can be read either as "Agent a *knows* φ", or "Agent a *believes* φ", depending on the interpretation.

Dynamic epistemic logics model situations in which external events or actions can modify the knowledge or beliefs of agents. Examples of dynamic epistemic logics include the Logic of Public Announcements introduced in [6], the Action Model Logic introduced in [9], and the logic $\mathcal{L}_{E\pi}$ introduced in [2]. The Logic of Public Announcements is capable of modeling the knowledge of agents after a public announcement that a formula in the language holds. The Action Model Logic is a generalization of the first that is capable of modeling a broad variety of events involving communication, including public and private messages. The logic $\mathcal{L}_{E\pi}$ is capable of modeling knowledge of agents after the execution of a process in a modified version of the π-calculus, in which agents are allowed to send private messages – which are formulas in the logic – to each other.

Here, we will present a dynamic epistemic logic $\mathcal{L}^*_{E\pi}$, which is similar to $\mathcal{L}_{E\pi}$, but involves only the use of finite processes in the modified π-calculus. Due to the private nature of the communication modeled, agents may have misconceptions about epistemic states of other agents. Therefore, the interpretation of $K_a\varphi$ as "Agent a believes φ" is more appropriate for our logic.

1.2 The π-calculus

The π-calculus [5] is a process algebra used to model concurrent computations that may exchange names during computation. The names may represent information or communication links. The ability to exchange communication links makes the π-calculus useful for modeling mobile processes.

In this paper we will work with a basic subset of the π-calculus defined below.

Definition 1.1 [π-Calculus Syntax] Let $\mathcal{N} = \{x, y, \ldots\}$ be a denumerable set of *name* symbols. The *set of all processes of π-calculus* is the least set generated by the grammar:

$$\begin{aligned} P &::= \mathbf{0} \bigm| \pi.P \bigm| P \mid P \bigm| (\boldsymbol{\nu} x)\ P \bigm| !\pi.P \\ \pi &::= \overline{x}y \bigm| x(z) \bigm| \tau \end{aligned}$$

where $x, y \in \mathcal{N}$ and $\tau, \boldsymbol{\nu} \notin \mathcal{N}$.

Processes are represented by P and prefixes by π. An important aspect of a process is the set of actions it is capable of performing. The process $\mathbf{0}$ is the null process, which is incapable of action. Processes of the form $\pi.P$ are called *guarded*, which means that they must realize the action represented by the prefix π before proceeding. A process $\overline{x}y.P$

is capable of sending message y along channel x and then proceeding as P. The process $x(y).P$ is capable of receiving a name z along channel x and proceeding as $P\{z/y\}$ – P with z substituted for every free occurrence (see below) of y. A process $\tau.P$ can proceed as P after some internal action without any interaction with its environment. The process $P_1|P_2$ represents processes P_1 and P_2 acting in parallel. The notation $(\boldsymbol{\nu}x)\ P$ signifies that the name x is restricted, and cannot be used in communication between P and its environment, but may be used for communication among subprocesses of P. Finally, a process $!P$ is capable of self-replication, and may proceed as $!P|P$.

We often omit null processes from process expressions and may write $\overline{x}y$ in place of $\overline{x}y.\mathbf{0}$, for example.

The variable y is bound if it is the scope of a receiving prefix $x(y)$, or operator $(\boldsymbol{\nu}y)$. All other occurrences of variables in prefixes are considered to be free.

The *transition semantics* of processes describes how processes can be reduced. We write $P \to Q$ if the process P can be reduced to Q in a single step. The basic reductions are $\tau.P \to P$ and $\overline{x}y|x(z).P \to P\{y/z\}$. The operation of parallel composition is considered to be associative and commutative, so parentheses may be omitted, and when multiple parallel components appear, multiple reductions may be possible.

Processes P and Q that can mimic any of each other's transitions are said to be *bisimilar*, written $P \sim Q$. We refrain from giving the full definition of bisimilarity here, but note that if a process P has no possible reductions, then $P \sim \mathbf{0}$.

We give some basic examples of reduction below. For more detail about the π-calculus, we refer the reader to [8].

1.3 Examples of Reductions of Processes

Example 1.2 Let

$$P \overset{\text{def}}{=} \overline{x}y|\overline{x}w|\tau.x(z).Q$$

Then P has only one immediate transition, as the rightmost process is "guarded" by the τ prefix. We have:

$$P \to P' \overset{\text{def}}{=} \overline{x}y|\overline{x}w|x(z).Q$$

Now there are two components capable of sending a message along channel x, but only process capable of receiving the message. So P' may proceed as either of two distinct processes in a nondeterministic manner.

$$P' \to \overline{x}w|Q\{y/z\}$$

and

$$P' \to \overline{x}y|Q\{w/z\}$$

Example 1.3 Now let

$$P \overset{\text{def}}{=} \overline{x}y|(\boldsymbol{\nu}x)\ (\overline{x}w|x(z).Q)$$

Here, the appearance of the operator $(\boldsymbol{\nu}x)$ means that the components within its scope may use the name x only to communicate with each other. So despite the fact that the symbol x also appears in the first component, it does not represent the same channel that is referred to by x in the second two components. Therefore, there is actually only one transition of P:

$$P \rightarrow \overline{x}y|(\boldsymbol{\nu}x)\ Q\{w/z\}$$

Example 1.4 Our final example shows the phenomenon of scope extrusion of a ν-operator. Given the process

$$P \overset{\text{def}}{=} ((\boldsymbol{\nu}y)\ \overline{x}(y).Q)|x(z).Q'$$

then we have the transition

$$P \rightarrow (\boldsymbol{\nu}y)\ (Q|Q'\{y/z\})$$

provided y does not appear free in Q'. Before the transition, the scope of the operator $(\boldsymbol{\nu}y)$ is the process Q, but after it is expanded to include the entire parallel composition. The idea is that by sending the restricted name y, process Q has established a private communication link with process Q'. This privacy persists after such a transition, even if the process P is put in parallel composition with other processes that use the name y.

2 Syntax of the logic $\mathcal{L}^*_{E\pi}$

In this section we define the syntax of the logic $\mathcal{L}^*_{E\pi}$, which is a dynamic epistemic logic for modeling knowledge updates using *finite* processes of the π-calculus. The logic $\mathcal{L}_{E\pi}$ defined in [2] - which allowed the use of both finite and infinite processes of the π-calculus - is an extension of $\mathcal{L}^*_{E\pi}$.

Definition 2.1 [Dynamic Epistemic Logic with Finite Processes Syntax] Let $\mathcal{P}$ be a denumerable set of atomic proposition symbols, $\mathcal{A}$ a finite set of agents, and $\mathcal{N}$ a denumerable set of name symbols. The *set* $\mathcal{L}^*_{E\pi}$ *of all formulas of Dynamic Epistemic Logic with Finite Processes* is the least set generated by the grammar:

$$\begin{aligned}
\varphi &::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_a\varphi \mid [P]\varphi \\
\psi &::= p \mid \neg\psi \mid (\psi \wedge \psi) \mid K_a\psi \\
P &::= \mathbf{0} \mid \pi.P \mid P \mid P \mid (\boldsymbol{\nu}x)\ P \\
\pi &::= \overline{x}_a y \mid \overline{x}_a\psi \mid x_a(z) \mid \tau
\end{aligned}$$

where $p \in \mathcal{P}$, $a \in \mathcal{A}$, $x, y \in \mathcal{N}$ and $\tau, \boldsymbol{\nu} \notin \mathcal{N}$.

We call the processes represented by P in the above grammar $\mathcal{L}^*_{E\pi}$ *processes.* These differ from standard π-calculus process in two fundamental ways. First, every sending and receiving action is associated with an agent a – the agent responsible for sending or receiving the name or message. Second, not only names but also purely epistemic formulas are allowed to be sent and received by processes.

We note that the processes are the same as those used in $\mathcal{L}_{E\pi}$ with the exception that the replication operation is not allowed in their formation, and that only purely epistemic formulas are allowed to be sent by agents.

Note: A small technical issue arises here when discussing transitions of $\mathcal{L}^*_{E\pi}$ processes. When we write $P \to Q$ we assume both processes are $\mathcal{L}^*_{E\pi}$ processes. So, for example, the following

$$\overline{x}\varphi|x(y).\overline{y}z \to \overline{\varphi}z$$

is *not* an acceptable reduction, as the process on the right has a formula where a name should be, and therefore is not well-formed.

Otherwise, reductions for $\mathcal{L}^*_{E\pi}$ processes are defined using the same rules as for processes in the regular π-calculus.

3 Semantics of Epistemic Logics

Here we give the semantics of the static portion of $\mathcal{L}^*_{E\pi}$, which coincides with the standard definition of semantics for basic epistemic logic $\mathcal{L}_E$. We will later extend the definition to the dynamic part of $\mathcal{L}^*_{E\pi}$. Formulas are interpreted in Kripke models.

Definition 3.1 A Kripke model $\mathfrak{M}$ is a tuple $\mathfrak{M} = \langle \mathcal{W}, \{R_i\}_{i\in\mathcal{A}}, V\rangle$, where:

(i) $\mathcal{W} = \{w_1, w_2, \ldots\}$ is a countable set of possible worlds (also called states).

(ii) $R_i \subseteq \mathcal{W} \times \mathcal{W}$ is an accessibility relation between worlds for each agent $i \in \mathcal{A}$.

(iii) $V : \mathcal{W} \times \mathcal{P} \to \{T, F\}$ is a valuation function that assigns a truth value to each propositional symbol in each possible world.

Given a model $\mathfrak{M}$ and a world $w \in \mathcal{W}$, we call the pair $(\mathfrak{M}, w)$ a pointed Kripke model.

Any sentence in $\mathcal{L}^*_{E\pi}$ can be evaluated in any pointed Kripke model. For purely epistemic formulas, the evaluation is just given by the standard Kripke semantics.

Definition 3.2 [Satisfaction in $\mathcal{L}^*_{E\pi}$ - static part.] We define the satisfaction relation $\models$ pointed Kripke models $\mathfrak{M}, w$ and (purely epistemic) formulas in $\mathcal{L}^*_{E\pi}$ as follows

(i) $\mathfrak{M}, w \models p$ iff $V(w, p) = T$

(ii) $\mathfrak{M}, w \models \neg\varphi$ iff $\mathfrak{M}, w \not\models \varphi$

(iii) $\mathfrak{M}, w \models (\varphi \wedge \psi)$ iff $\mathfrak{M}, w \models \varphi$ and $\mathfrak{M}, w \models \psi$

(iv) $\mathfrak{M}, w \models K_a\varphi$ iff for all $w' \in \mathcal{W}$ if $(w, w') \in R_a$ then $\mathfrak{M}, w' \models \varphi$

We defer the "dynamic" portion of the definition, (ie. the case $\mathfrak{M}, w \models [P]\varphi$) to Section 5.1.

4 Action Model Logic

To introduce our proof system for the logic $\mathcal{L}^*_{E\pi}$, we will use ideas from the Action Model Logic introduced in [1].

Definition 4.1 Let $\mathcal{L}$ be any epistemic language over a set of agents $\mathcal{A}$ and propositional symbols $\mathcal{P}$. An action model $\mathfrak{A}$ is a tuple $\langle E, \{\to_a\}_{a\in\mathcal{A}}, PRE\rangle$, where E is a set of events,

$\rightarrow_a$ is an accessibility relation on $E \times E$ for each $a \in \mathcal{A}$, and $PRE : E \rightarrow \mathcal{L}$ is a function assigning to each action point a precondition, which is a formula in the language $\mathcal{L}$.

We may write $(e, e') \in \rightarrow_a$ as $e \rightarrow_a e'$.

An action α is a pointed action model $(\mathfrak{A}, e)$ where $e \in E$. We may blur the distinction between actions and the associated point in the action model, and write $PRE(\alpha)$ for $PRE(e)$. If $\alpha = (\mathfrak{A}, e)$ and $\alpha' = (\mathfrak{A}, e')$ are actions, and $e \rightarrow_a e'$, we write $\alpha \rightarrow_a \alpha'$. Note that the actions must be pointed models for the same action structure for this relation to hold.

The idea is that in a Kripke model an action may occur, which requires updating the model. The resulting model is the composition of the model and the action, defined below.

Definition 4.2 Given a model and an action structure:

$$\mathfrak{M} = \langle \mathcal{W}, \{R_i\}_{i \in \mathcal{A}}, V \rangle$$

$$\mathfrak{A} = \langle E, \{\rightarrow_a\}_{a \in \mathcal{A}}, PRE \rangle$$

and distinguished elements $w_0 \in \mathcal{W}$, $e_0 \in E$, the product $(\mathfrak{M}, w_0) \otimes (\mathfrak{A}, e_0)$ of pointed models is defined iff $\mathfrak{M}, w_0 \models PRE(e_0)$. When defined,

$$(\mathfrak{M}, w_0) \otimes (\mathfrak{A}, e_0) = (\mathfrak{M}', (w_0, e_0))$$

Where $\mathfrak{M}' = \langle \mathcal{W}', \{R'_a\}_{a \in \mathcal{A}}, V' \rangle$ such that:

(i) $\mathcal{W}' = \{(w, e) \mid w \in \mathcal{W}, e \in E \text{ and } \mathfrak{M}, w \models PRE(e)\}$

(ii) $((w, e), (w', e')) \in R'_a$ iff $(w, w') \in R_a$ and $(e, e') \in \rightarrow_a$

(iii) $V'((w, e), p) = V(w, p)$ for all $(w, e) \in \mathcal{W}'$ and $p \in \mathcal{P}$

So an action is possible in a given world if that world meets the action's precondition. If multiple actions are possible in a given world w, then the world "splits" in the updated model. That is, we get a world of the form (w, e) for each action e that is possible in world w. If an agent considers w' to be possible given that the actual world is w, and also considers e' to be possible given action e, then he considers the pair (w', e') possible in the updated model, given actual world (w, e). Finally, the actions do not change the actual facts of a world w, so w and (w, e) will share a truth valuation, regardless of e.

4.1 Syntax and Semantics of Action Model Logic

We now introduce syntax of the action model logic $\mathcal{L}_{Act}$ as follows:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_a\varphi \mid [\alpha]\varphi$$

We let α range over the set of all actions (up to isomorphism), with preconditions that are $\mathcal{L}_{Act}$ formulas already constructed at a previous stage of the inductive hierarchy. The semantics can then be defined as in Definition 3.2. We need only show how to handle the induction in the case of formulas like $[\alpha]\varphi$, where $\alpha = (\mathfrak{A}, e)$:

$$\mathfrak{M}, w \models [\alpha]\varphi \text{ iff } \mathfrak{M}, w \models PRE(e) \text{ implies } (\mathfrak{M}, w) \otimes (\mathfrak{A}, e) \models \varphi$$

We abbreviate $\neg[\alpha]\neg\varphi$ as $\langle\alpha\rangle\varphi$.

4.2 Composition of Action Models

We may also define an operation of composition on pairs of action structures.

Definition 4.3 Let $\mathfrak{A} = \langle E, \{\rightarrow_a\}, PRE\rangle$ and $\mathfrak{A}' = \langle E', \{\rightarrow'_a\}, PRE'\rangle$ be actions, and let $e_0 \in E$, $e'_0 \in E'$. We define $(\mathfrak{A}, e_0) \circ (\mathfrak{A}', e'_0) = (\mathfrak{A}'', (e_0, e'_0))$, where $\mathfrak{A}'' = \langle E'', \{\rightarrow''_a$ $\}, PRE''\rangle$ such that:

(i) $E'' = E \times E'$

(ii) $((d, d'), (e, e')) \in \rightarrow''_a$ iff $(d, e) \in \rightarrow_a$ and $(d', e') \in \rightarrow'_a$

(iii) $PRE((e, e')) = \langle\mathfrak{A}, e\rangle PRE'(e')$

The following indicates that the effect of executing a composition of actions is the same as that of executing the actions in succession:

Proposition 4.4 *[1] Let $\mathfrak{M}, w$ be a pointed Kripke model, α and β be actions and φ a formula of $\mathcal{L}_{Act}$. Then*

$$\mathfrak{M}, w \models [\alpha][\beta]\varphi \Leftrightarrow \mathfrak{M}, w \models [\alpha \circ \beta]\varphi$$

5 The extended language $\mathcal{L}^+_{E\pi}$

We define an extension $\mathcal{L}^+_{E\pi}$ of the language $\mathcal{L}^*_{E\pi}$, by adding actions to the language. We will define a complete proof system for $\mathcal{L}^+_{E\pi}$. Because $\mathcal{L}^+_{E\pi}$ is an extension of $\mathcal{L}^*_{E\pi}$, every valid formula of $\mathcal{L}^*_{E\pi}$ will be a theorem in the proof system.

We first specify the set of actions that we will add to the language. The *basic actions* consist of the following two types:

(i) The trivial action $\tau = (\mathfrak{A}, t)$, where $\mathfrak{A} = \langle\{t\}, \{\rightarrow_a\}, PRE\rangle$, such that $\rightarrow_a = \{(t, t)\}$ for all $a \in \mathcal{A}$ and $PRE(t) = \top$.

(ii) The communication actions $\alpha^{\varphi}_{i,j} = (\mathfrak{A}^{\varphi}_{i,j}, e)$, representing a message φ sent on some channel x by agent i and received by agent j. Here

$$\mathfrak{A}^{\varphi}_{i,j} = \langle\{e, t\}, \{\rightarrow_a\}, PRE\rangle,$$

such that $\rightarrow_j = \rightarrow_i = \{(e, e), (t, t)\}$, $\rightarrow_a = \{(e, t), (t, t)\}$ for all $a \neq i, j$, $PRE(e) = K_i\varphi$ and $PRE(t) = \top$.

The motivation for including these types of actions is that they mimic the sorts of epistemic updates that correspond to transitions of processes in our modified π-calculus. The communication actions represent a message φ being sent from i to j, unbeknownst to the other agents. The precondition $K_i\varphi$ indicates that agent i may only send a message she believes to be true, ie., agents may not attempt to lie or deceive. We need trivial actions because processes may have transitions in which no information whatsoever is exchanged. Updating with trivial actions has no semantic effect.

The set *Act* of $\mathcal{L}^+_{E\pi}$ *actions* is the closure of the set of basic actions under composition. We are now ready to formally define the language $\mathcal{L}^+_{E\pi}$.

Definition 5.1 [Dynamic Epistemic Logic with Finite Processes and Actions Syntax] Let $\mathcal{P}$ be a denumerable set of atomic proposition symbols, $\mathcal{A}$ a finite set of agents, and $\mathcal{N}$ a denumerable set of name symbols. The *set* $\mathcal{L}^+_{E\pi}$ *of all formulas of Dynamic Epistemic Logic with Finite Processes and Actions* is the least set generated by the grammar:

$$\begin{aligned}
\varphi &::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_a\varphi \mid [P]\varphi \mid [\alpha]\varphi \\
\psi &::= p \mid \neg\psi \mid (\psi \wedge \psi) \mid K_a\psi \\
P &::= \mathbf{0} \mid \pi.P \mid P \,|\, P \mid (\boldsymbol{\nu}x)\, P \\
\pi &::= \overline{x}_a y \mid \overline{x}_a\psi \mid x_a(z) \mid \tau
\end{aligned}$$

where $p \in \mathcal{P}$, $a \in \mathcal{A}$, $x, y \in \mathcal{N}$, $\alpha \in Act$ and $\tau, \boldsymbol{\nu} \notin \mathcal{N}$.

Essentially, we have taken the language $\mathcal{L}^*_{E\pi}$ and actions from the set *Act* to the syntax. The semantics for the "new" formulas of the type $[\alpha]\varphi$ is defined in the same way as formulas of that type in $\mathcal{L}_{Act}$ (see Section 4.1).

5.1 A translation function for $\mathcal{L}^+_{E\pi}$

We wish to define a translation function $t : \mathcal{L}^+_{E\pi} \to \mathcal{L}_E$. We need some preliminary definitions. The first allows us to distinguish reductions of processes in which some information is acquired by an agent from those in which no information (but possibly a name) is acquired by any agent.

Definition 5.2 [Reduction-action Types] To every reduction of a process, we associate a basic action as follows:

(i) If a process $P \to P'$, such that the reduction used is either a τ-reduction, or the transmission of a *name* from one component to another, then we write $P \rightsquigarrow_\tau P'$.

(ii) If $P \to P'$, such that the reduction is the reception by agent j of a *formula* φ sent by agent i, then we write $P \rightsquigarrow_{\alpha^\varphi_{i,j}} P'$.

Next we define a one-step translation for formulas of shape $[P]\varphi$.

Definition 5.3 Let $\psi = [P]\varphi$. We define $s(\psi)$ as follows:

(i) $s([P]\varphi) = \varphi$ if $P \sim \mathbf{0}$

(ii) Otherwise, suppose that $P \rightsquigarrow_{\alpha_i} P_i$, where $1 \leq i \leq k$, and each α_i is a basic action. We assume that if $P \to Q$ then $Q = P_i$ for some $1 \leq i \leq k$, and that there are no repetitions among the P_i. Then we write:

$$s([P]\varphi) = \bigwedge_{1 \leq i \leq k} [\alpha_i][P_i]\varphi$$

We may now define the full translation function t:

Definition 5.4 The translation function $t : \mathcal{L}_{E\pi}^{+} \to \mathcal{L}_E$ is defined inductively according the following rules:

(i) $t(p) = p$

(ii) $t(\neg\varphi) = \neg t(\varphi)$

(iii) $t(\varphi \wedge \psi) = t(\varphi) \wedge t(\psi)$

(iv) $t(K_a\varphi) = K_a t(\varphi)$

(v) $t([\alpha]p) = PRE(\alpha) \to p$

(vi) $t([\alpha]\neg\varphi) = PRE(\alpha) \to \neg[\alpha]\varphi$

(vii) $t([\alpha](\varphi \wedge \psi)) = t([\alpha]\varphi) \wedge t([\alpha]\psi)$

(viii) $t([\alpha]K_a\varphi) = t(PRE(\alpha) \to \bigwedge_{\{\alpha' \mid \alpha \to_a \alpha'\}} K_a[\alpha']\varphi)$

(ix) $t([\alpha][\beta]\varphi) = t([\alpha \circ \beta]\varphi)$

(x) $t([P]\varphi) = t(s([P]\varphi))$

(xi) $t([\alpha][P]\varphi) = t([\alpha]s([P]\varphi))$

It is not obvious that $t(\varphi)$ is defined for all $\varphi \in \mathcal{L}_{E\pi}^{+}$, because in some cases $t(\varphi)$ is defined in terms of formulas that are not subformulas of φ. We will later define a well-ordered complexity measure on formulas of $\mathcal{L}_{E\pi}^{+}$, and show in Lemma 5.9 that $t(\varphi)$ is always defined in terms of t applied to formulas of strictly smaller complexity. As a consequence, $t(\varphi)$ is defined for every $\varphi \in \mathcal{L}_{E\pi}^{+}$.

We may now also give the missing case of Definition 3.2:

Definition 5.5 [Satisfaction in $\mathcal{L}_{E\pi}^{*}$ - dynamic part.]
If φ is an $\mathcal{L}_{E\pi}^{*}$ formula, and P is an $\mathcal{L}_{E\pi}^{*}$ process, then

$$\mathfrak{M}, w \models [P]\varphi \text{ iff } \mathfrak{M}, w \models s([P]\varphi).$$

Although $s([P]\varphi)$ is not a subformula of $[P]\varphi$ in general, the induction is still well-founded, as a consequence of Lemma 5.9.

5.2 The Proof System

We are now ready to define a proof system for the extended language $\mathcal{L}_{E\pi}^{+}$. The system is an extension of the proof system for Action Model Logic given in [1], which is itself an extension of the standard complete proof system **K** for modal logic (see, for example, [7]), with axioms added corresponding to each case in our translation function. Its axioms and inference rules are given in the following table:

All instantiations of proposition tautologies	
K-normality	$K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$
atomic permanence	$[\alpha]p \leftrightarrow (PRE(\alpha) \to p)$
action and negation	$[\alpha]\neg\varphi \leftrightarrow (PRE(\alpha) \to \neg[\alpha]\varphi)$
action and conjunction	$[\alpha](\varphi \wedge \psi) \leftrightarrow ([\alpha]\varphi \wedge [\alpha]\psi)$
action and knowledge	$[\alpha]K_a\varphi \leftrightarrow (PRE(\alpha) \to \bigwedge_{\{\alpha' \mid \alpha \to_a \alpha'\}} K_i[\alpha']\varphi)$
composition of actions	$[\alpha][\beta]\varphi \leftrightarrow [\alpha \circ \beta]\varphi$
processes	$[P]\varphi \leftrightarrow s([P]\varphi)$
	$[\alpha][P]\varphi \leftrightarrow [\alpha]s([P]\varphi)$
modus ponens	From φ and $\varphi \to \psi$ infer ψ
necessitation of K_a	From φ infer $K_a\varphi$

The soundness of all axioms not involving processes follows from the soundness of the proof system given in [1]. The soundness of the axioms involving processes is immediate from our alternate definition of semantics of formulas of type $[P]\varphi$. Hence:

Proposition 5.6 *The proof system is sound.*

The following lemma will be the key to our completeness proof:

Lemma 5.7 *For every $\varphi \in \mathcal{L}^+_{E\pi}$, $t(\varphi)$ is defined, and moreover $\vdash \varphi \leftrightarrow t(\varphi)$*

In order to prove the lemma, we first define a *complexity function* on formulas of $\mathcal{L}^+_{E\pi}$.

Definition 5.8 For a finite process P, let $l(P)$ be the length of the longest possible reduction of P. The complexity function $c : \mathcal{L}^+_{E\pi} \cup Act \to \mathbb{N}^2$, is defined recursively as follows, where if $c(\varphi) = (x, y)$, we write $c_1(\varphi) = x$, and $c_2(\varphi) = y$.

(i) $c(p) = (0, 1)$

(ii) $c(\neg\varphi) = (c_1(\varphi), c_2(\varphi) + 1)$

(iii) $c(K_a\varphi) = (c_1(\varphi), c_2(\varphi) + 1)$

(iv) $c(\varphi \wedge \psi) = (\max\{c_1(\varphi), c_1(\psi\}, \max\{c_2(\varphi), c_2(\psi\} + 1)$

(v) $c([\alpha]\varphi) = (c_1(\varphi), (4 + c_2(\alpha))c_2(\varphi))$

(vi) $c([P]\varphi) = (c_1(\varphi) + l(P) + 1, c_2(\varphi))$

(vii) $c(\alpha) = (0, \max\{c_2(PRE(d)) \mid d \in \mathfrak{A}\})$ where $\alpha = (\mathfrak{A}, e)$

The following lemma will allow us to see that the semantics of $\mathcal{L}^+_{E\pi}$ is well-founded for formulas of type $[P]\varphi$, and also to prove the key Lemma 5.7.

Lemma 5.9 *Let $<_L$ represent the lexicographic order on $\mathbb{N}^2$. Then the following hold:*

(i) $c(\psi) \leq_L c(\varphi)$ *if ψ is a subformula of φ.*

(ii) $c(PRE(\alpha) \to p) <_L c([\alpha]p)$

(iii) $c(PRE(\alpha) \to \neg[\alpha]\varphi) <_L c([\alpha]\neg\varphi)$

(iv) $c([\alpha](\varphi \wedge \psi)) <_L c([\alpha]\varphi \wedge [\alpha]\psi)$

(v) $c(PRE(\alpha) \to K_a[\beta]\varphi) <_L c([\alpha]K_a\varphi)$ *for all β such that $\alpha \to_a \beta$*

(vi) $c([\alpha \circ \beta]\varphi) <_L c([\alpha][\beta]\varphi)$

(vii) $c(s([P]\varphi)) <_L c([P]\varphi)$

(viii) $c([\alpha]s([P]\varphi)) <_L c([\alpha][P]\varphi)$

Proof. Item (i) is clear from the definition of c. Items (ii)-(vi) are proved in a similar manner. In (iii), for example, we have that $c_1(PRE(\alpha) \to \neg[\alpha]\varphi) = c_1([\alpha]\neg\varphi)$, but that $c_2(PRE(\alpha) \to \neg[\alpha]\varphi) < c_2([\alpha]\neg\varphi)$. For details on the latter claim, see [9]. Analogous remarks hold for the others.

For (vii), we claim $c_1(s([P]\varphi)) < c_1([P]\varphi)$. In the case where $P \sim \mathbf{0}$, $[P]\varphi = \varphi$, so the claim follows immediately from (i). If $P \nsim \mathbf{0}$, then $s([P]\varphi) = \bigwedge_{1 \leq i \leq k} [\alpha_i][P_i]\varphi$, where $P \rightsquigarrow P_i$ for each $1 \leq i \leq k$. But if $P \rightsquigarrow P'$, then $l(P') < l(P)$. It follows that $c_1([\alpha_i][P_i]\varphi) < c_1[P]\varphi$ for all i. But c_1 evaluated on a conjunction is the maximum of c_1 of any of the conjuncts, so the claim holds, and (vii) follows immediately. An analogous statement holds for (viii) . □

Proof of Lemma 5.7. By induction on the complexity of formulas. If $c(\varphi) = (0, 1)$, then φ is some propositional symbol p, and therefore $t(p) = p$. By propositional tautology, $\vdash p \leftrightarrow t(p)$. Now suppose that $\vdash \psi \leftrightarrow t(\psi)$ for all ψ such that $c(\psi) <_L c(\varphi)$.

The rest of the proof is handled in various cases. For example, suppose $\varphi = \neg\psi$. Then by Lemma 5.9 and the induction hypothesis, we have $\vdash \psi \leftrightarrow t(\psi)$. By tautology, we have $\vdash \neg\psi \leftrightarrow \neg t(\psi)$, which by definition of t, is the same as $\vdash \varphi \leftrightarrow t(\varphi)$. The other cases are handled similarly.

Lemma 5.10 *For all $\varphi \in \mathcal{L}^+_{E\pi}$, $t(\varphi) \in \mathcal{L}_E$.*

The proof is an easy induction over complexity of formulas. We may now give a short proof of completeness.

Theorem 5.11 *The proof system is complete. Ie., if $\models \varphi$ then $\vdash \varphi$.*

Proof. Suppose $\models \varphi$. Then, by soundness and the fact that $\vdash \varphi \leftrightarrow t(\varphi)$, we have $\models t(\varphi)$. By Lemma 5.10, $t(\varphi)$ is a formula of $\mathcal{L}_E$ (ie., contains no processes or actions). Since our proof system is an extension of the standard complete proof system **K** for $\mathcal{L}_E$, we have that $\vdash t(\varphi)$. This, together with $\vdash \varphi \leftrightarrow t(\varphi)$, yields the desired conclusion $\vdash \varphi$. □

6 Conclusions and Future Work

Following the ideas of [2], we have presented an attempt to integrate the approach of the π-calculus – which allows us to reason about communicating systems with mobility – with that of dynamic epistemic logic – which concerns updating the knowledge or beliefs of agents that are able to communicate. Having defined a dynamic epistemic language whose actions are finite processes in a modified version of the π-calculus, and extending that language with action structures as in [1] representing private messages between pairs of agents, we were able to define a proof system for the extended language and prove its completeness.

Although the above shows that our language is subsumed by the Action Model Logic of [1], there are reasons that the logic $\mathcal{L}^*_{E\pi}$ may be preferable in many applications. In cases where many sequences of actions are possible and can be captured by a single process, $\mathcal{L}^*_{E\pi}$ will gave a much more natural and succinct may of representing the sequences. Furthermore, the possible actions are inherent in the language, and do not need to be defined whenever they appear in a formula. The language is also highly adaptable; there is no reason it has to be restricted to two-party honest communication (as was done here for simplicity's sake). For example, if we wanted to model public or group communication rather than private communication, we need only subscript prefixes with sets of agents, and adjust the actions we use accordingly. Also, the semantics of $\mathcal{L}^*_{E\pi}$ is readily extendible to handle infinite processes (see [2]), which is not the case for Action Model Logic.

There are some obvious directions for future work. We aim to develop a proof system for a logic where the restriction to finite processes is dropped, and also where the formulas that can be sent are not limited to purely epistemic formulas. We do not expect that the present techniques will be applicable to the language with infinite processes, as our system makes essential use of the fact that any sequence of reductions of a process terminates. The introduction of fixed-point operators may be useful in attacking this problem. We also aim to develop a type system for the language so that we need not restrict the allowable transitions of processes. In [2], a slightly different version of the logic $\mathcal{L}^*_{E\pi}$ is used to proof the correctness of preservation of anonymity in the Dining Cryptographer's Algorithm. We believe that additional applications of the logic may be found in the areas of cryptography and security.

References

[1] Baltag, A., L. Moss and S. Solecki, *The logic of public annnouncements, common knowledge and private suspicions*, Technical report, CWI (1999).

[2] Góngora, P., E. Ufferman and F. Hernández-Quiroz, *Formal semantics of a dynamic epistemic logic for describing knowledge properties of π-calculus processes*, in: *Proceedings of Computational Logic in Multi-Agent Systems XI (to appear)*, LNCS (2010).

[3] Mardare, R., *Decidable extensions of hennessy-milner logic*, in: *Proceedings of International Conference on Formal Methods for Networked and Distributed Systems (FORTE 2006)*, LNCS 4229 (2006).

[4] Mardare, R., *Observing distributed computation. a dynamic-epistemic approach*, in: *Proceedings of the second Conference on Algebra and Coalgebra in Computer Science (CALCO2007)*, LNCS 4624 (2007).

[5] Milner, R., J. Parrow and J. Walker, *A calculus of mobile processes i and ii*, Information and Computation **100** (1992), pp. 1–77.

[6] Plaza, J., *Logics of public communications*, in: *Proceedings of 4th International Symposium on Methodologies for Intelligent Systems*, 1989, pp. 201–216.

[7] Popkorn, S., "First Steps in Modal Logic," Cambridge University Press, 2008.

[8] Sangiorgi, D. and D. Walker, "The Pi-Calculus A Theory of Mobile Processes," Cambridge University Press, 2003.

[9] van Ditmarsch, H., W. van der Hoek and B. Kooi, "Dynamic Epistemic Logic," Springer, 2007.