Tuning the program transformers from LCC to PDL

Pardo, P.; Sarrión-Morillo, E.; Soler-Toscano, F.; Velázquez-Quesada, F.R.

Tuning the Program Transformers from LCC to PDL

Pere Pardo
Ruhr-Universität Bochum, Germany.
pere.pardoventura@ruhr-uni-bochum.de

Enrique Sarrión-Morillo, Fernando Soler-Toscano
Universidad de Sevilla, Spain.
{esarrion,fsoler}@us.es

Fernando R. Velázquez-Quesada
Universiteit van Amsterdam, The Netherlands.
F.R.VelazquezQuesada@uva.nl

Abstract

This work proposes an alternative definition of the so-called program transformers used to obtain reduction axioms in the Logic of Communication and Change (LCC). Our proposal uses an elegant matrix treatment of Brzozowski's equational method instead of Kleene's translation from finite automata to regular expressions. The two alternatives are shown to be equivalent, with Brzozowski's method having the advantage of generating smaller expressions for models with average connectivity.

Keywords: Logic of communication and change, dynamic epistemic logic, propositional dynamic logic, action model, program transformer, reduction axiom

1 Introduction

Dynamic Epistemic Logic [1, 2] (*DEL*) encompasses several logical frameworks whose main aim is the study of different single- and multi-agent epistemic attitudes and the way they change due to diverse epistemic actions. These frameworks typically have two building blocks: a 'static' component, using some 'epistemic' model to

represent the notion to be studied (e.g., knowledge or belief), and a 'dynamic' component, using model operations to represent actions that affect such notion (e.g., announcements or belief revision).[1]

Among the diverse existing *DEL* frameworks, the Logic of Communication and Change (LCC) of [6] stands as one of the most interesting. It consists of Propositional Dynamic Logic [7] (PDL), interpreted epistemically (its 'static' component), and the action models machinery [8, 9] for representing knowledge about actions (its 'dynamic' component). The LCC framework allows us to model not only diverse epistemic actions (as public, private or secret announcements) but also factual change.

A key feature of this logic is that it characterises the effect of an action model's execution via *reduction axioms*: valid formulas through which it is possible to rewrite a formula with action model (update) modalities as an equivalent one without them, thus reducing LCC to PDL and hence providing a compositional analysis for a wide range of informational events. For example, the reduction axiom for conjunction tells us that $\varphi \wedge \psi$ will be the case true after the pointed action model $(\mathsf{U}, \mathsf{e}_i)$ is executed, $[\mathsf{U}, \mathsf{e}_i](\varphi \wedge \psi)$, if and only if both φ and ψ are true after executing such action, $[\mathsf{U}, \mathsf{e}_i]\varphi \wedge [\mathsf{U}, \mathsf{e}_i]\psi$. For another example, the reduction axiom for atoms p effectively reduces an LCC formula $[\mathsf{U}, \mathsf{e}_i]p$ into a formula about the conditions of the action e_i and its effect on p (see Table 1).

As one might expect, the crucial reduction axiom is the one characterising the effect of an action model over epistemic modalities π (i.e. over PDL programs):

$$[\mathsf{U}, \mathsf{e}_i][\pi]\varphi \;\leftrightarrow\; \bigwedge_{j=0}^{n-1} [T_{ij}^{\mathsf{U}}(\pi)][\mathsf{U}, \mathsf{e}_j]\varphi$$

This axiom, presented in detail in what follows, characterises the epistemic change that the action model U brings about: after the pointed action model $(\mathsf{U}, \mathsf{e}_i)$ is executed, every π-path in the resulting epistemic model leads to a φ-world, $[\mathsf{U}, \mathsf{e}_i][\pi]\varphi$, if and only if, for every action e_j in the action model U, every $T_{ij}^{\mathsf{U}}(\pi)$-path in the original epistemic model ends in a world that, after the execution of $(\mathsf{U}, \mathsf{e}_j)$, will satisfy φ. The axiom is based on the correspondence between action models and finite automata observed in [10]; its main component, the so-called program transformer function T_{ij}^{U}, follows Kleene's translation from finite automata to regular expressions [11].[2]

[1] This form of representing the dynamics is different from other approaches as, e.g., epistemic temporal logic [3, 4] (*ETL*), in which the static model already describes not only the relevant notion but also all the possible ways it can change due to the chosen epistemic action(s). See [5] for a comparison between *DEL* and *ETL*.

[2] See [12] for a deep discussion about the meaning of Kleene's theorem.

The present work proposes an alternative definition of program transformer, using instead a matrix treatment of Brzozowski's equational method for obtaining an expression representing the language accepted by a given finite automaton [13, 14].

Structure of the paper The paper starts in Section 2 by recalling the LCC framework together with its reduction axioms and its definition of program transformers. Section 3 explains how we can obtain, through Brzozowski's equational method, the corresponding expressions for Kleene closure, and then Section 4 introduces this paper's proposal, used to define an alternative translation from LCC to PDL. Section 5 comments on the computational complexity of this approach; the computational costs of the two methods are also compared using Prolog with different test-cases. Section 6 presents a summary and a discussion of further topics for research.

2 Logic of Communication and Change

This section recalls LCC's semantic structure, its language and semantic interpretation, and its axiom system. Throughout this paper, Var will denote a set of atoms (propositional variables), and Ag will denote a finite set of agents.

We start the definition of LCC by introducing the involved structures. First, the structure over which LCC formulas are interpreted.

Definition 1 (Epistemic model)**.** An *epistemic model* M is a triple

$$(W, \langle R_a \rangle_{a \in \mathsf{Ag}}, V)$$

where $W \neq \varnothing$ is a set of worlds, $R_a \subseteq (W \times W)$ is an epistemic relation for each agent $a \in \mathsf{Ag}$ and $V : \mathsf{Var} \to \wp(W)$ is an atomic evaluation.

Note how the epistemic relations R_a are not required to satisfy any particular property. As usual, each possible world can be interpreted as a possible state of affairs (each one of them defined by the atomic valuation), and each relation R_a represents agent a's uncertainty about the situation: at world w, for agent a all worlds u such that wR_au are epistemically possible, i.e. are seen as possible by this agent. Figure 1 shows an example of an epistemic model.

Here is the structure for representing the knowledge about actions in the system.

Definition 2 (Action model)**.** Let $\mathcal{L}$ be a language built upon Var and Ag that can be interpreted over epistemic models. An $\mathcal{L}$ *action model* U is a tuple

$$(\mathsf{E}, \langle \mathsf{R}_a \rangle_{a \in \mathsf{Ag}}, \mathsf{pre}, \mathsf{sub})$$

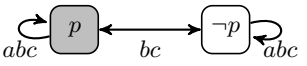


Figure 1: An epistemic model M with an actual p-world (gray) and a possible world with $\neg p$ (white). The arrows represent accessibility relations R_a, R_b and R_c, so only agent a knows that currently p, while b and c ignore whether p.

where $\mathsf{E} = \{\mathsf{e}_0, \ldots, \mathsf{e}_{n-1}\}$ is a *finite* non-empty set of actions, $\mathsf{R}_a \subseteq (\mathsf{E} \times \mathsf{E})$ is a relation for each $a \in \mathsf{Ag}$, $\mathsf{pre} : \mathsf{E} \to \mathcal{L}$ is a precondition map assigning a formula $\mathsf{pre}(\mathsf{e}) \in \mathcal{L}$ to each action $\mathsf{e} \in \mathsf{E}$, and $\mathsf{sub} : (\mathsf{E} \times \mathsf{Var}) \to \mathcal{L}$ is a postcondition map assigning a formula $\mathsf{sub}(\mathsf{e}, p) \in \mathcal{L}$ to each atom $p \in \mathsf{Var}$ at each action $\mathsf{e} \in \mathsf{E}$. The postcondition map should only change a finite number of atoms, so $\mathsf{sub}(\mathsf{e}, p) \neq p$ can hold only for a finite number of $p \in \mathsf{Var}$.[3] We emphasise that, in this definition, the language $\mathcal{L}$ is just a parameter.

Just as each relation R_a describes agent a's uncertainty about the situation, each relation R_a represents a's uncertainty about the executed action: $\mathsf{e}\mathsf{R}_a\mathsf{f}$ indicates a cannot distinguish f from e. Note, again, how the relation is not required to satisfy any particular property.

Example 1 (Announcements)**.** Figure 2 illustrates three action models for announcements in a set of three agents $\mathsf{Ag} = \{a, b, c\}$. Each of the actions, say f, is purely epistemic (i.e., fact-preserving), so $\mathsf{sub}(\mathsf{f}, p) = p$ for any $p \in \mathsf{Var}$. Labeled arrows denote accessibility relations R_a, R_b or R_c; a gray circle denotes the action that is actually being executed, while other actions (wrongly believed by some agents to possibly take place) are represented by white circles. The preconditions are written below the corresponding actions.

As mentioned, action models represent both the actions and the knowledge agents have about these actions. Action models modify epistemic models in the following way.

Definition 3 (Update execution)**.** Let $M = (W, \langle R_a \rangle_{a \in \mathsf{Ag}}, V)$ be an epistemic model and $\mathsf{U} = (\mathsf{E}, \langle \mathsf{R}_a \rangle_{a \in \mathsf{Ag}}, \mathsf{pre}, \mathsf{sub})$ an $\mathcal{L}$ action model, both over Var and Ag. Recall

[3] These 'finiteness' requirements (finite domain and only a finite number of atoms affected by the postcondition function) are needed to allow the pointed action model (U, e) —a pair with U an $\mathcal{L}$ action model and e a distinguished action in it— to be associated to a syntactic object and thus to be used within formulae. For details, the reader is referred to the discussion about action models in Section 6.1 of [1].

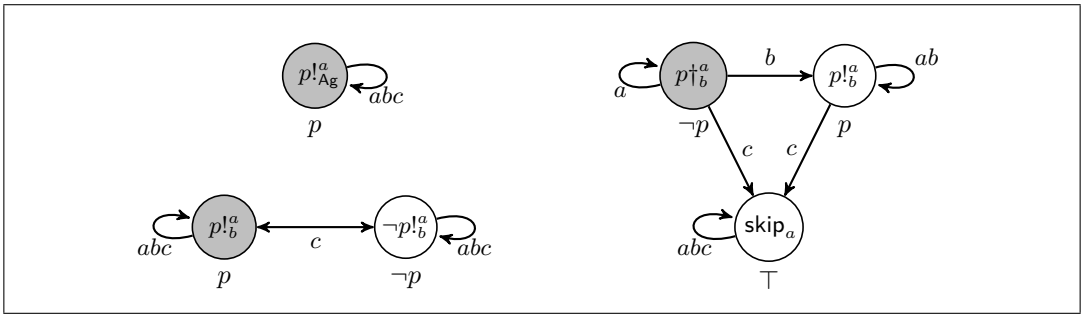


Figure 2: (Top left) A truthful public announcement by a that p, denoted $p!^a_{\mathsf{Ag}}$. (Bottom left) A private announcement by a to b about p, denoted $p!^a_b$; here agent c only knows about the 'topic' of the message. (Right) A secret lie about p made by a to b, denoted $p\dagger^a_b$, is accepted by b as truthful, i.e. as if it was $p!^a_b$; agent c is not aware of any communication between a and b.

that $\mathcal{L}$ is any language built upon Var and Ag that can be interpreted over epistemic models, so we can assume the existence of a function $[\![\cdot]\!]^M$ returning those worlds in M in which each formula of $\mathcal{L}$ holds.

The update execution of U on M produces an epistemic model

$$(M \otimes \mathsf{U}) = (W^{M\otimes\mathsf{U}}, \langle R_a^{M\otimes\mathsf{U}}\rangle_{a\in\mathsf{Ag}}, V^{M\otimes\mathsf{U}})$$

given, for every $a \in \mathsf{Ag}$ and $p \in \mathsf{Var}$, by

$$\begin{aligned} W^{M\otimes\mathsf{U}} &:= \{\, (w,\mathsf{e}) \in W \times \mathsf{E} \mid w \in [\![\mathsf{pre}(\mathsf{e})]\!]^M \,\} \\ R_a^{M\otimes\mathsf{U}} &:= \{\, \langle (w,\mathsf{e}),(v,\mathsf{f})\rangle \in W^{M\otimes\mathsf{U}} \times W^{M\otimes\mathsf{U}} \mid wR_av \text{ and } \mathsf{e}\,\mathsf{R}_a\mathsf{f} \,\} \\ V^{M\otimes\mathsf{U}}(p) &:= \{\, (w,\mathsf{e}) \in W^{M\otimes\mathsf{U}} \mid w \in [\![\mathsf{sub}(\mathsf{e},p)]\!]^M \,\} \end{aligned}$$

Thus, the update execution of U on M produces an epistemic model $M \otimes \mathsf{U}$ whose domain is the restricted cartesian product of the original models' domains.[4] In $M \otimes \mathsf{U}$, a world (w,e) satisfies an atom p if and only if w satisfied the formula $\mathsf{sub}(\mathsf{e},p)$ in M; finally, an agent a sees a world (u,f) as possible from (w,e) if and only if she sees u from w (in M) and sees f from e (in U). If one works with a particular class of epistemic models in which the epistemic relations satisfy specific properties, then the chosen action models should be such that the update execution preserves these properties. This is straightforward in some cases as, e.g., reflexivity, transitivity and symmetry are preserved by update execution when the relations in

[4] If there is no world in M satisfying $\mathsf{pre}(\mathsf{e})$ for some action e in U, then the resulting structure is not an epistemic model, as its domain is empty.

the action models are reflexive, transitive and symmetric, respectively. But this is not always the case: for example, seriality is not preserved, even when the involved action models are serial. See Figure 3 for an illustration of different updates in an epistemic model.

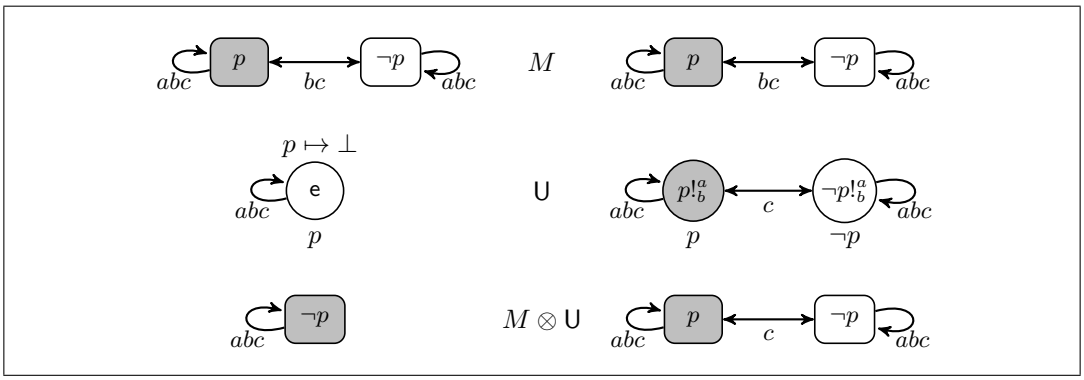


Figure 3: Two illustrations of update execution in the epistemic model M from Fig. 1 (topmost row here), where only agent a knows that p. (Left) Action e represents a public (i.e. publicly observable) change to $\neg p$; note that the postcondition is written on top of the action. After execution, it becomes common knowledge that $\neg p$. (Right) A private announcement by a to b about p results in a new model where it is public that b now knows whether p, and only c remains ignorant about p.

With the semantic structures already defined, it is time now to define the language that will be used to describe them. Note that the formulas (and programs) of the language $\mathcal{L}_{\mathsf{LCC}}$ are defined simultaneously with the notion of an $\mathcal{L}_{\mathsf{LCC}}$ action model (i.e. an action model using $\mathcal{L}_{\mathsf{LCC}}$ for its precondition and postcondition maps).

Definition 4 (Language $\mathcal{L}_{\mathsf{LCC}}$)**.** The formulas φ and programs π of the language $\mathcal{L}_{\mathsf{LCC}}$ are given by, respectively:

$$\varphi \quad ::= \quad \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\pi]\varphi \mid [\mathsf{U},\mathsf{e}]\varphi$$
$$\pi \quad ::= \quad a \mid ?\varphi \mid \pi;\pi \mid \pi \cup \pi \mid \pi^*$$

where $p \in \mathsf{Var}$, $a \in \mathsf{Ag}$ and (U,e) is a pair with an $\mathcal{L}_{\mathsf{LCC}}$ action model U and an action e in this model.[5]

[5]More precisely, the language is defined by a double induction starting from the language $\mathsf{PDL}^0 = \mathsf{PDL}$, then defining $\mathcal{L}^0_{\mathsf{LCC}}$ as PDL^0 plus modalities of the form $[\mathsf{U},\mathsf{e}]$ for U a PDL^0 action model, then defining PDL^1 as $\mathcal{L}^0_{\mathsf{LCC}}$ plus tests $?\varphi$ for $\varphi \in \mathcal{L}^0_{\mathsf{LCC}}$, then defining $\mathcal{L}^1_{\mathsf{LCC}}$ as PDL^1 plus modalities of the form $[\mathsf{U},\mathsf{e}]$ for U a PDL^1 action model, and so on. The full language $\mathcal{L}_{\mathsf{LCC}}$ is then the union of all languages $\mathcal{L}^i_{\mathsf{LCC}}$ with i finite.

As the definition states, the set of LCC formulas contains the atomic propositions and $\top$, and it is closed under negation, conjunction, and modalities $[\pi]$ (for π a program) and $[\mathsf{U}, \mathsf{e}]$ (for U an $\mathcal{L}_{\mathsf{LCC}}$ action model and e an action in it).[6] On the other hand, the set of LCC programs contains basic programs for agents a and 'tests' $?\varphi$ (with φ a formula), and it is closed under sequential composition (;), non-deterministic choice ($\cup$) and Kleene closure (*).

It is only left to define the $[\![\cdot]\!]^M$ function associated to $\mathcal{L}_{\mathsf{LCC}}$ that collects the worlds of a given epistemic model M in which a given $\mathcal{L}_{\mathsf{LCC}}$ formula holds. In the case of LCC, this function also indicates which pairs of worlds are related by a given $\mathcal{L}_{\mathsf{LCC}}$ program.

Definition 5 (Semantics of $\mathcal{L}_{\mathsf{LCC}}$)**.** Let $M = (W, \langle R_a \rangle_{a \in \mathsf{Ag}}, V)$ be an epistemic model and $\mathsf{U} = (\mathsf{E}, \langle \mathsf{R}_a \rangle_{a \in \mathsf{Ag}}, \mathsf{pre}, \mathsf{sub})$ an action model. The function $[\![\cdot]\!]^M$, returning both those worlds in W in which an $\mathcal{L}_{\mathsf{LCC}}$ formula holds and those pairs in $W \times W$ in which an $\mathcal{L}_{\mathsf{LCC}}$ program holds, is given by

$$
\begin{array}{rlrl}
[\![\top]\!]^M &:= W & [\![a]\!]^M &:= R_a \\
[\![p]\!]^M &:= V(p) & [\![?\varphi]\!]^M &:= \mathsf{Id}_{[\![\varphi]\!]^M} \\
[\![\neg\varphi]\!]^M &:= W \setminus [\![\varphi]\!]^M & [\![\pi_1;\pi_2]\!]^M &:= [\![\pi_1]\!]^M \circ [\![\pi_2]\!]^M \\
[\![\varphi_1 \wedge \varphi_2]\!]^M &:= [\![\varphi_1]\!]^M \cap [\![\varphi_2]\!]^M & [\![\pi_1 \cup \pi_2]\!]^M &:= [\![\pi_1]\!]^M \cup [\![\pi_2]\!]^M \\
[\![[\pi]\varphi]\!]^M &:= \{w \in W \mid \forall v((w,v) \in [\![\pi]\!]^M \Rightarrow v \in [\![\varphi]\!]^M)\} & [\![\pi^*]\!]^M &:= ([\![\pi]\!]^M)^* \\
[\![[\mathsf{U},\mathsf{e}]\varphi]\!]^M &:= \{w \in W \mid w \in [\![\mathsf{pre}(\mathsf{e})]\!]^M \Rightarrow (w,\mathsf{e}) \in [\![\varphi]\!]^{M \otimes \mathsf{U}}\}
\end{array}
$$

where $\circ$ and * are the composition and the reflexive transitive closure operator, respectively, and Id_U is the identity relation on $U \subseteq W$. Notice two special cases for *test*: $[\![?\bot]\!]^M = \varnothing$ and $[\![?\top]\!]^M = \mathsf{Id}_W$.

Even though LCC can be seen abstractly as the logic of regular programs (the PDL part) plus action models (modalities of the form $[\mathsf{U}, \mathsf{e}]$), it is also illustrative to discuss its epistemic interpretation, in particular, that of its PDL programs. Basic 'agent' programs $a \in \mathsf{Ag}$ produce formulas of the form $[a]\varphi$, read simply as *"agent a knows/believes φ"* as in standard Epistemic Logic. More complex programs also have epistemic readings. Formulas of the form $[\pi_1;\pi_2]\varphi$, relying on the sequential composition π_1 and then π_2, can be read as *"π_1 knows/believes that π_2 knows/believes φ"*, and thus can be used to express *nested* knowledge/belief; formulas of the form $[\pi_1 \cup \pi_2]\varphi$, relying on the union of the relations for π_1 and π_2, can be read as *"both π_1 and π_2 know/believe φ"*, and thus can be used to express *general* knowledge/belief among a group; finally, formulas of the form $[\pi^*]\varphi$, relying on the reflexive and

[6] From now on, all action models are assumed to be $\mathcal{L}_{\mathsf{LCC}}$ action models.

transitive closure of the relations for π, can be read as *"φ is the case, π knows it, π knows that she knows it, and so on"*, and thus can be used to express *common* knowledge (or, if $\pi^+ := \pi;\pi^*$ is used instead of π^*, *common* belief). The modalities involving action models simply state the action's effects, with formulas of the form $[\mathsf{U},\mathsf{e}]\varphi$ reading *"φ is the case after any execution of the pointed action* (U,e)*"*.

Axiom system The axiom system for LCC, shown in Table 1, combines the known axiom system of its PDL fragment (the left column; [7]) with *recursion* axioms for its action model fragment (the right column). Intuitively, recursion axioms are valid formulae characterising a situation *after* an update execution in terms of a situation *before* such update, and thus indicating how to rewrite a formula with an action model modality as a provably equivalent one without them. Then, while soundness follows from the validity of these new axioms, completeness follows from the completeness of the basic system.[7]

(*taut*) propositional tautologies	(*top*) $[\mathsf{U},\mathsf{e}]\top \leftrightarrow \top$
(*K*) $[\pi](\varphi_1 \to \varphi_2) \to ([\pi]\varphi_1 \to [\pi]\varphi_2)$	(*atm*) $[\mathsf{U},\mathsf{e}]p \leftrightarrow (\mathsf{pre}(\mathsf{e}) \to \mathsf{sub}(\mathsf{e},p))$
(*test*) $[?\varphi_1]\varphi_2 \leftrightarrow (\varphi_1 \to \varphi_2)$	(*neg*) $[\mathsf{U},\mathsf{e}]\neg\varphi \leftrightarrow (\mathsf{pre}(\mathsf{e}) \to \neg[\mathsf{U},\mathsf{e}]\varphi)$
(*seq*) $[\pi_1;\pi_2]\varphi \leftrightarrow [\pi_1][\pi_2]\varphi$	(*conj*) $[\mathsf{U},\mathsf{e}](\varphi_1 \wedge \varphi_2) \leftrightarrow ([\mathsf{U},\mathsf{e}]\varphi_1 \wedge [\mathsf{U},\mathsf{e}]\varphi_2)$
(*choice*) $[\pi_1 \cup \pi_2]\varphi \leftrightarrow [\pi_1]\varphi \wedge [\pi_2]\varphi$	(K_{U}) $[\mathsf{U},\mathsf{e}](\varphi_1 \to \varphi_2) \to ([\mathsf{U},\mathsf{e}]\varphi_1 \to [\mathsf{U},\mathsf{e}]\varphi_2)$
(*mix*) $[\pi^*]\varphi \leftrightarrow \varphi \wedge [\pi][\pi^*]\varphi$	(*prog*) $[\mathsf{U},\mathsf{e}_i][\pi]\varphi \leftrightarrow \bigwedge_{j=0}^{n-1}[T_{ij}^{\mathsf{U}}(\pi)][\mathsf{U},\mathsf{e}_j]\varphi$
(*ind*) $\varphi \wedge [\pi^*](\varphi \to [\pi]\varphi)) \to [\pi^*]\varphi$	(N_{U}) From $\vdash \varphi$ infer $\vdash [\mathsf{U},\mathsf{e}]\varphi$
(*MP*) From $\vdash \varphi_1$ and $\vdash \varphi_1 \to \varphi_2$ infer $\vdash \varphi_2$	
(N_π) From $\vdash \varphi$ infer $\vdash [\pi]\varphi$	

Table 1: LCC calculus in [6] is that of PDL (left column) plus reduction axioms and necessitation rule for $[\mathsf{U},\mathsf{e}]$ (right column).

In our particular case, recursion axioms for atomic propositions and boolean constants/operators are standard for action models with ontic (i.e., valuation) change [16]: while axiom (*atm*) states that an atom p will be the case *after* any update execution with action model U and action e, $[\mathsf{U},\mathsf{e}]p$, if and only if, *before* the update, the formula $\mathsf{sub}(\mathsf{e},p)$ holds whenever $\mathsf{pre}(\mathsf{e})$ holds, $\mathsf{pre}(\mathsf{e}) \to \mathsf{sub}(\mathsf{e},p)$, axioms (*neg*) and (*conj*) state that update execution commutes with negation (modulo its precondition) and distributes over conjunction, respectively.

[7] The reader is referred to Chapter 7 of [1] (see also [15]) for an extensive explanation of this technique.

The most important recursion axiom, (*prog*), characterises the effect of an action model over LCC programs. It states that after any update execution with U on e_i every π-path in the resulting model will lead to a φ-world, $[\mathsf{U},\mathsf{e}_i][\pi]\varphi$, if and only if, before the update, every $T^{\mathsf{U}}_{ij}(\pi)$-path leads to a world that will satisfy φ after any update execution with U on e_j where e_j is any action on U, $\bigwedge_{j=0}^{n-1}[T^{\mathsf{U}}_{ij}(\pi)][\mathsf{U},\mathsf{e}_j]\varphi$. In this axiom, the program transformer T^{U}_{ij} is crucial, taking an LCC program π representing a path on $M \otimes \mathsf{U}$ and returning an LCC program $T^{\mathsf{U}}_{ij}(\pi)$ representing a 'matching' path on M, taking additional care that such path can be also reproduced in the action model U. A program transformer follows Kleene's translation from finite automata to regular expressions [11], and it is formally defined as follows.

Note also that the (valid) formula K_{U} is not listed among the LCC axioms in [6]. It has been added here not only because it cannot be derived from the rest of the system, but also because it allows the derivation of the crucial rule

$$\frac{\chi \leftrightarrow \psi}{[\mathsf{U},\mathsf{e}]\chi \leftrightarrow [\mathsf{U},\mathsf{e}]\psi}\ \mathsf{RE}_{\mathsf{U}}$$

This rule is needed for the *inside-out* translation of nested action model modalities (see footnote 10). The fact that K_{U} is not derivable from the rest of the system is stated in [15] (in particular, its Thm. 29), a paper which examines axiom systems for PAL, the logic of public announcements [φ!]. Their analysis of completeness proofs is based on a reduction from PAL, and thus it applies to LCC as well. (Of course, another alternative is to add RE_{U} directly since, following [15, Prop. 3], K_{U} is derivable from RE_{U} and the original LCC system, Table 1 minus K_{U}.) That the system on Table 1 is indeed sound and (weakly) complete w.r.t. the given semantic interpretation can be shown using the same technique as [15, Corollary 12].

Definition 6 (Program transformer [6])**.** Let $\mathsf{U} = (\mathsf{E}, \langle \mathsf{R}_a\rangle_{a\in\mathsf{Ag}}, \mathsf{pre}, \mathsf{sub})$ be an action model with $\mathsf{E} = \{\mathsf{e}_0,\ldots,\mathsf{e}_{n-1}\}$. The *program transformer* T^{U}_{ij} $(i,j \in \{0,\ldots,n-1\})$ on the set of LCC programs is defined as:

$$T^{\mathsf{U}}_{ij}(a) := \begin{cases} ?\mathsf{pre}(e_i); a & \text{if } \mathsf{e}_i \mathsf{R}_a \mathsf{e}_j \\ ?\bot & \text{otherwise} \end{cases} \qquad T^{\mathsf{U}}_{ij}(?\varphi) := \begin{cases} ?(\mathsf{pre}(\mathsf{e}_i) \wedge [\mathsf{U},\mathsf{e}_i]\varphi) & \text{if } i = j \\ ?\bot & \text{otherwise} \end{cases}$$

$$T^{\mathsf{U}}_{ij}(\pi_1;\pi_2) := \bigcup_{k=0}^{n-1}(T^{\mathsf{U}}_{ik}(\pi_1); T^{\mathsf{U}}_{kj}(\pi_2)) \qquad T^{\mathsf{U}}_{ij}(\pi_1 \cup \pi_2) := T^{\mathsf{U}}_{ij}(\pi_1) \cup T^{\mathsf{U}}_{ij}(\pi_2)$$

$$T^{\mathsf{U}}_{ij}(\pi^*) := K^{\mathsf{U}}_{ijn}(\pi)$$

with K^{U}_{ijn} inductively defined as follows:

$$K^{\mathsf{U}}_{ij0}(\pi) := \begin{cases} ?\top \cup T^{\mathsf{U}}_{ij}(\pi) & \text{if } i = j \\ T^{\mathsf{U}}_{ij}(\pi) & \text{otherwise} \end{cases}$$

$$K^{\mathsf{U}}_{ij(k+1)}(\pi) = \begin{cases} (K^{\mathsf{U}}_{kkk}(\pi))^* & \text{if } i = k = j \\ (K^{\mathsf{U}}_{kkk}(\pi))^*; K^{\mathsf{U}}_{kjk}(\pi) & \text{if } i = k \neq j \\ K^{\mathsf{U}}_{ikk}(\pi); (K^{\mathsf{U}}_{kkk}(\pi))^* & \text{if } i \neq k = j \\ K^{\mathsf{U}}_{ijk}(\pi) \cup (K^{\mathsf{U}}_{ikk}(\pi); (K^{\mathsf{U}}_{kkk}(\pi))^*; K^{\mathsf{U}}_{kjk}(\pi)) & \text{if } i \neq k \neq j \end{cases}$$

Example 2. In the action model of Fig. 3 (left), the axiom for the *public change to* $\neg p$ reduces an epistemic consequence $[\mathsf{U}, \mathsf{e}][a]\neg p$ to a claim before execution, namely $[?\mathsf{pre}(\mathsf{e}); a][\mathsf{U}, \mathsf{e}]\neg p$, which is necessarily true –see the left column below–. Similarly, in the action model of a private lying announcement Fig. 2 (right), enumerate the actions as $p\dagger^a_b = \mathsf{e}_0$ and $p!^a_b = \mathsf{e}_1$ and $\mathsf{skip}_a = \mathsf{e}_2$. Then, the axiom for the lying announcement $p\dagger^a_b$ turns the *believed lie* $[\mathsf{U}, p\dagger^a_b][b]p$ into a claim before the execution, also a tautology –see the right column–.

$$\begin{array}{lll} & [\mathsf{U}, \mathsf{e}][a]\neg p & \\ \equiv & [?\mathsf{pre}(\mathsf{e}); a][\mathsf{U}, \mathsf{e}]\neg p & \\ \equiv & [?p; a]\Big(\mathsf{pre}(\mathsf{e}) \to \neg[\mathsf{U}, \mathsf{e}]p\Big) & \\ \equiv & p \to [a]\Big(p \to \neg(\mathsf{pre}(\mathsf{e}) \to \mathsf{sub}(\mathsf{e}, p))\Big) & \\ \equiv & p \to [a]\Big(p \to \neg(p \to \bot)\Big) & \\ \equiv & p \to [a]\Big(p \to (p \wedge \top)\Big) & \\ \equiv & p \to [a]\top \quad \equiv \quad \top & \end{array} \qquad \begin{array}{lll} & [\mathsf{U}, p\dagger^a_b][b]p & \\ \equiv & [T^{\mathsf{U}}_{01}(b)][\mathsf{U}, p!^a_b]p & \\ \equiv & [?\mathsf{pre}(p\dagger^a_b); b][\mathsf{U}, p!^a_b]p & \\ \equiv & [?\neg p; b]\Big(\mathsf{pre}(p!^a_b) \to \mathsf{sub}(p!^a_b, p)\Big) & \\ \equiv & \neg p \to [b]\Big(p \to p\Big) & \\ \equiv & \neg p \to [b]\top & \\ \equiv & \top & \end{array}$$

3 Program transformation through Brzozowski's equations

This paper proposes an alternative definition of program transformer, denoted $\mu^{\mathsf{U}}(\pi)[i, j]$, that differs from $T^{\mathsf{U}}_{ij}(\pi)$ mainly in the case for the Kleene closure operator. Before presenting the formal definitions in Section 4, we introduce the method in an informal way. In the action models of Figure 4, we tag every edge from e_i to e_j with a label $\pi \mid \mu^{\mathsf{U}}(\pi)[i, j]$ with π a program and and $\mu^{\mathsf{U}}(\pi)[i, j]$ its transformation. For example, in the agents' diagram below, the label from e_0 to e_1

$$a \mid ?\mathsf{pre}(\mathsf{e}_0); a \qquad \text{means} \qquad \mu^{\mathsf{U}}(a)[0, 1] = ?\mathsf{pre}(\mathsf{e}_0); a.$$

i.e. $?\mathsf{pre}(\mathsf{e}_0); a$ is what we should test in (M, w) to ensure that, after executing $(\mathsf{U}, \mathsf{e}_0)$ over (M, w), an a-path from (w, e_0) to some state (w', e_1) will persist in $M \otimes \mathsf{U}$. (If no a-path from e_0 to e_1 exists, the transformation of a is $?\bot$.)

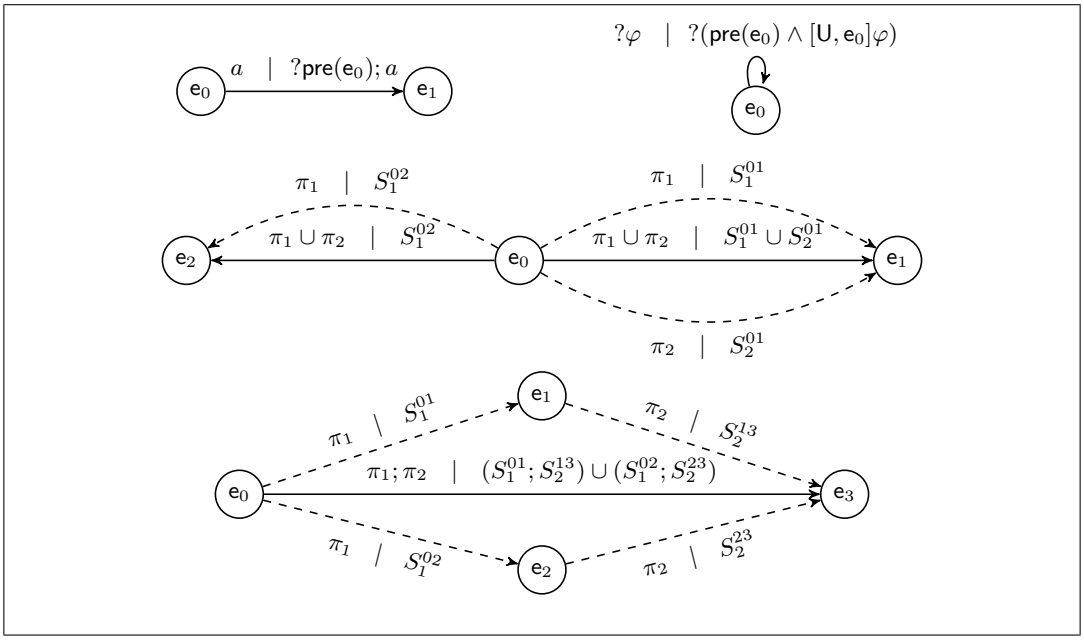
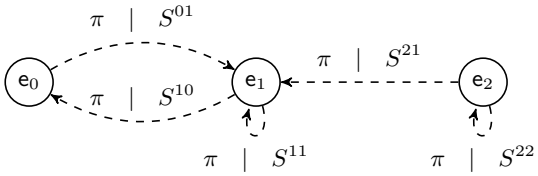


Figure 4: An illustration of action models and their program transformers for the following programs: *agent* (top left), *test* (top right), *choice* (mid) and *composition* (bottom). Dashed and solid lines represent, respectively, the original labels and those obtained after applying choice (mid) or product (bottom).

The construction of the diagrams on Figure 4 proceeds in a very similar way to that of Def. 6, just simplifying some trivial cases like $\pi \cup ?\bot$, which is reduced to π. The main novelty of our transformation is for the Kleene closure. We use a method proposed by Brzozowski [13], presented here in a matrix format (see [17, 18] for more an in-depth analysis about the improvements that we are applying to LCC language).

Kleene closure The following example will be used to illustrate the generation of the transformations of π^* from those of π. (The π^*-paths from e_i to e_j will be denoted by X^{ij}, while the corresponding π-paths are labeled as S^{ij}.)



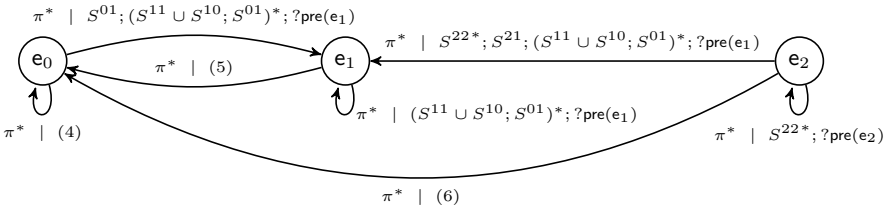
Generate an equation system[8] (Brzozowski [13]). E.g. for paths to e_0:

$$X^{00} = ?\mathsf{pre}(\mathsf{e}_0) \cup (S^{01}; X^{10}) \quad (1)$$
$$X^{10} = (S^{10}; X^{00}) \cup (S^{11}; X^{10}) \quad (2)$$
$$X^{20} = (S^{22}; X^{20}) \cup (S^{21}; X^{10}) \quad (3)$$

Solve the system[9] using: substitution; associativity, distributivity [19], and Arden's Theorem [20] ($X = B \cup (A; X)$ *implies* $X = A^*; B$). E.g.

$$X^{00} = ?\mathsf{pre}(\mathsf{e}_0) \cup (S^{01}; ((S^{10}; S^{01}) \cup S^{11})^*; S^{10}; ?\mathsf{pre}(\mathsf{e}_0)) \quad (4)$$
$$X^{10} = ((S^{10}; S^{01}) \cup S^{11})^*; S^{10}; ?\mathsf{pre}(\mathsf{e}_0) \quad (5)$$
$$X^{20} = (S^{22})^*; S^{21}; ((S^{10}; S^{01}) \cup S^{11})^*; S^{10}; ?\mathsf{pre}(\mathsf{e}_0) \quad (6)$$

Similar processes produce labels for π^*-paths to e_1 and e_2, represented as:



By using a matrix calculus similar to that in Chapter 3 of [14] we calculate all X^{ij} in parallel and thus avoid repeating the process for each destination node. The following section presents the formal definition of the matrix calculus; here we just illustrate the use of the matrix calculus. The equations (1)–(3) used above can be represented in the following matrix:

[8] For equation (2), observe how a π^*-path from e_1 to e_0 might start with S^{10} and then continue with X^{00} (an instance of π^* from e_0 to e_0), but it might also start with S^{11} and then continue with X^{10}. In equation (1), a π^*-path from e_0 to e_0 is to do nothing, but then the transformation should check $?\mathsf{pre}(\mathsf{e}_0)$, i.e. whether e_0 is executable at the target state.

[9] We illustrate first how equation (2) is solved into (5):

$$\begin{aligned} X^{10} &= (S^{10}; (?\mathsf{pre}(\mathsf{e}_0) \cup (S^{01}; X^{10}))) \cup (S^{11}; X^{10}) && \text{(substitute } X^{00} \text{ using (1))} \\ &= (S^{10}; ?\mathsf{pre}(\mathsf{e}_0)) \cup (S^{10}; S^{01}; X^{10}) \cup (S^{11}; X^{10}) && \text{(distributivity)} \\ &= (S^{10}; ?\mathsf{pre}(\mathsf{e}_0)) \cup (((S^{10}; S^{01}) \cup S^{11}); X^{10}) && \text{(associativity)} \\ &= ((S^{10}; S^{01}) \cup S^{11})^*; S^{10}; ?\mathsf{pre}(\mathsf{e}_0) && \text{(Arden's Theorem)} \end{aligned}$$

Next, we use this to substitute X^{10} in (1) to obtain (4). Finally, we substitute X^{10} in (3) and apply Arden's Theorem to obtain (6).

	e_0	e_1	e_2	e_0	e_1	e_2
e_0	$?\bot$	S^{01}	$?\bot$	$?\mathsf{pre}(\mathsf{e}_0)$	$?\bot$	$?\bot$
e_1	S^{10}	S^{11}	$?\bot$	$?\bot$	$?\mathsf{pre}(\mathsf{e}_1)$	$?\bot$
e_2	$?\bot$	S^{21}	S^{22}	$?\bot$	$?\bot$	$?\mathsf{pre}(\mathsf{e}_2)$

The left part contains the π-paths from one node (row) to another one (column). It is an accessibility matrix for the π-graph above. Call $\mu^{\mathsf{U}}(\pi)[i,j]$ the cell corresponding to row e_i and column e_j in this left part and $A^{\mathsf{U}}[i,j]$ the cell with the same position at the right part. Observe that $A^{\mathsf{U}}[i,j] = ?\mathsf{pre}(\mathsf{e}_i)$ if $i = j$ and $?\bot$ otherwise. We may check that the equations for X^{ij} that we created above looking at the π-graph can be created now by:

$$X^{ij} = (\mu^{\mathsf{U}}(\pi)[i,0]; X^{0j}) \cup (\mu^{\mathsf{U}}(\pi)[i,1]; X^{1j}) \cup (\mu^{\mathsf{U}}(\pi)[i,2]; X^{2j}) \cup A^{\mathsf{U}}[i,j] \qquad (7)$$

For example, the equations for X^{10} and X^{00} (equivalent to (2) and (1), resp.) are

$$X^{10} = (S^{10}; X^{00}) \cup (S^{11}; X^{10}) \cup (?\bot; X^{20}) \cup ?\bot \qquad (8)$$
$$X^{00} = (?\bot; X^{00}) \cup (S^{01}; X^{10}) \cup (?\bot; X^{20}) \cup ?\mathsf{pre}(\mathsf{e}_0) \qquad (9)$$

The greatest advantage of working with matrices is that we can perform several operations in parallel by working in a row. Applying Arden's Theorem to the e_1 row of the previous matrix gives:

	e_0	e_1	e_2	e_0	e_1	e_2
e_1	$(S^{11})^*; S^{10}$	$?\bot$	$(S^{11})^*; ?\bot$	$(S^{11})^*; ?\bot$	$(S^{11})^*; ?\mathsf{pre}(\mathsf{e}_1)$	$(S^{11})^*; ?\bot$

We replaced the left cell $[\mathsf{e}_1, \mathsf{e}_1]$ with $?\bot$ and concatenated its previous value $(S^{11})^*$ with the others cells in the row. After simplifying into $?\bot$ cells we get:

	e_0	e_1	e_2	e_0	e_1	e_2
e_1	$(S^{11})^*; S^{10}$	$?\bot$	$?\bot$	$?\bot$	$(S^{11})^*; ?\mathsf{pre}(\mathsf{e}_1)$	$?\bot$

To check that we have applied Arden's Theorem, look at X^{10} (using (7) in the last matrix): $X^{10} = (S^{11})^*; S^{10}; X^{00}$. It is the result of applying Arden's Theorem to (8) (or (2)). Substitution can also be done in parallel:

	e_0	e_1	e_2	e_0	e_1	e_2
e_2	$(S^{21}; (S^{11})^*; S^{10}) \cup ?\bot$	$?\bot$	$(S^{21}; ?\bot) \cup S^{22}$	$(S^{21}; ?\bot) \cup ?\bot$	$(S^{21}; (S^{11})^*; ?\mathsf{pre}(\mathsf{e}_1)) \cup ?\bot$	$(S^{21}; ?\bot) \cup ?\mathsf{pre}(\mathsf{e}_2)$

The above row for e_2 was obtained from the previous row by applying the following substitution into the original matrix: first, the left position $B = [\mathsf{e}_2, \mathsf{e}_1]$ (S^{21} in this case) is replaced with $?\bot$; second, every other (left/right) position $D = [\mathsf{e}_2, \mathsf{e}_i]$ contains now a program with the form $(B; C) \cup D$, where C is the program in the

(resp. left/right) $[\mathsf{e}_1, \mathsf{e}_i]$ position in the previous row for e_1. After simplifying into $?\bot$ cells (where appropriate) we obtain:

	e_0	e_1	e_2	e_0	e_1	e_2
e_2	$(S^{21}; (S^{11})^*; S^{10})$	$?\bot$	S^{22}	$?\bot$	$(S^{21}; (S^{11})^*; ?\mathsf{pre}(\mathsf{e}_1))$	$?\mathsf{pre}(\mathsf{e}_2)$

To illustrate that we have done a substitution, consider the value of X^{21} in the matrix before the substitution (just as in the initial matrix):

$$X^{21} = (S^{21}; X^{11}) \cup (S^{22}; X^{21}) \tag{10}$$

And now consider the value of X^{11} after the application of Arden's Theorem:

$$X^{11} = ((S^{11})^*; S^{10}; X^{01}) \cup ((S^{11})^*; ?\mathsf{pre}(\mathsf{e}_1)) \tag{11}$$

Using (11) to substitute X^{11} in (10) we get:

$$X^{21} = \Big(S^{21}; \Big(((S^{11})^*; S^{10}; X^{01}) \cup ((S^{11})^*; ?\mathsf{pre}(\mathsf{e}_1))\Big)\Big) \cup (S^{22}; X^{21}) \tag{12}$$

that can be rewritten, using the distributive and associative properties, into:

$$X^{21} = (S^{21}; (S^{11})^*; S^{10}; X^{01}) \cup (S^{22}; X^{21}) \cup (S^{21}; (S^{11})^*; ?\mathsf{pre}(\mathsf{e}_1)) \tag{13}$$

which is the equation obtained for X^{21} in the previous matrix, after the substitution.

In the following section we introduce the formal definitions of our matrix calculus to transform LCC programs.

4 A matrix calculus for program transformation

Definition 7 (Program transformation matrix)**.** Let $\mathsf{U} = (\mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{sub})$ be an action model with $\mathsf{E} = \{\mathsf{e}_0, \ldots, \mathsf{e}_{n-1}\}$. The function $\mu^{\mathsf{U}} : \Pi \to \mathcal{M}_{n\times n}$, with Π the set of LCC programs and $\mathcal{M}_{n\times n}$ the class of n-square matrices, takes an LCC program π and returns a n-square matrix $\mu^{\mathsf{U}}(\pi)$ in which each cell $\mu^{\mathsf{U}}(\pi)[i,j]$ is an LCC program representing the transformation of π from e_i to e_j in the sense of the program transformers $T^{\mathsf{U}}_{ij}(\pi)$ of [6]. The recursive definition of $\mu^{\mathsf{U}}(\pi)$ is as follows.

- *Agents*:

$$\mu^{\mathsf{U}}(a)[i,j] := \begin{cases} ?\mathsf{pre}(\mathsf{e}_i); a & \text{if } \mathsf{e}_i \mathsf{R}_a \mathsf{e}_j \\ ?\bot & \text{otherwise} \end{cases} \tag{14}$$

- *Test*:

$$\mu^{\mathsf{U}}(?\varphi)[i,j] := \begin{cases} ?(\mathsf{pre}(\mathsf{e}_i) \wedge [\mathsf{U},\mathsf{e}_i]\varphi) & \text{if } i = j \\ ?\bot & \text{otherwise} \end{cases} \tag{15}$$

- *Non-deterministic choice*:

$$\mu^{\mathsf{U}}(\pi_1 \cup \pi_2)[i,j] := \oplus \left\{ \mu^{\mathsf{U}}(\pi_1)[i,j],\ \mu^{\mathsf{U}}(\pi_2)[i,j] \right\} \tag{16}$$

where $\oplus\Gamma$ is the non-deterministic choice of the programs in Γ set after removing occurrences of $?\bot$, that is,

$$\oplus\Gamma := \begin{cases} \bigcup(\Gamma \setminus \{?\bot\}) & \text{if } \varnothing \neq \Gamma \neq \{?\bot\} \\ ?\bot & \text{otherwise} \end{cases} \tag{17}$$

being $\bigcup$ the generalised non-deterministic choice of a non-empty set of programs.

- *Sequential composition*:

$$\mu^{\mathsf{U}}(\pi_1;\pi_2)[i,j] := \oplus \left\{ \mu^{\mathsf{U}}(\pi_1)[i,k] \odot \mu^{\mathsf{U}}(\pi_2)[k,j] \ \mid\ 0 \leq k \leq n-1 \right\} \tag{18}$$

where $\sigma \odot \rho$ is the sequential composition of σ and ρ after removing superfluous occurrences of $?\bot$ and $?\top$, that is,

$$\sigma \odot \rho := \begin{cases} \sigma;\rho & \text{if } \sigma \neq ?\bot \neq \rho \text{ and } \sigma \neq ?\top \neq \rho \\ \sigma & \text{if } \sigma \neq ?\top \ = \ \rho \\ \rho & \text{if } \sigma \ = ?\top \\ ?\bot & \text{otherwise} \end{cases} \tag{19}$$

- *Kleene closure*:

$$\mu^{\mathsf{U}}(\pi^*) := S_0^{\mathsf{U}} \left(\mu^{\mathsf{U}}(\pi) \mid A^{\mathsf{U}} \right) \tag{20}$$

where $\mu^{\mathsf{U}}(\pi) \mid A^{\mathsf{U}}$ is the $n \times 2n$ matrix obtained by augmenting $\mu^{\mathsf{U}}(\pi)$ with A^{U}, an $n \times n$ matrix defined as

$$A^{\mathsf{U}}[i,j] := \begin{cases} ?\mathsf{pre}(\mathsf{e}_i) & \text{if } i = j \\ ?\bot & \text{otherwise} \end{cases} \tag{21}$$

The function S^{U}_k (with $0 \leq k \leq n$), defined as

$$S^{\mathsf{U}}_k(M \mid A) := \begin{cases} A & \text{if } k = n \\ S^{\mathsf{U}}_{k+1}(Subs_k(Ard_k(M \mid A))) & \text{otherwise} \end{cases} \tag{22}$$

receives an argument $M \mid A$ and performs an iterative process applying Arden's Theorem to row k (via function $Ard_k : \mathcal{M}_{n\times 2n} \to \mathcal{M}_{n\times 2n}$) and substituting rows different from k (via function $Subs_k : \mathcal{M}_{n\times 2n} \to \mathcal{M}_{n\times 2n}$) until a $k = n$, then returning the right part of the augmented matrix. The two auxiliary functions, Ard_k and $Subs_k$, are given by

$$Ard_k(N)[i,j] := \begin{cases} N[i,j] & \text{if } i \neq k \\ ?\bot & \text{if } i = k = j \\ N[i,j] & \text{if } i = k \neq j \text{ and } N[k,k] = ?\bot \\ N[k,k]^* \odot N[i,j] & \text{otherwise} \end{cases} \tag{23}$$

$$Subs_k(N)[i,j] := \begin{cases} N[i,j] & \text{if } i = k \\ ?\bot & \text{if } i \neq k = j \\ \bigoplus\{N[i,k] \odot N[k,j], N[i,j]\} & \text{otherwise} \end{cases} \tag{24}$$

The operators '$\oplus$' and '$\odot$' used in the previous definition are versions of nondeterministic choice and sequential composition that remove unnecessary occurrences of $?\bot$ and $?\top$; thus returning programs that are (potentially) syntactically shorter but nevertheless semantically equivalent to their PDL counterparts '$\cup$' and ';', as the following propositions show.

Proposition 1. *Let M be an epistemic model and Γ a set of* LCC *programs. Then,*

$$[\![\bigoplus \Gamma]\!]^M = [\![\bigcup \Gamma]\!]^M$$

Proof. Take any epistemic model M. Equation (17) states that $\bigoplus\Gamma$ is a nondeterministic choice of the LCC programs in Γ that returns $\bigcup(\Gamma \setminus \{?\bot\})$ when Γ is different from both $\varnothing$ and $\{?\bot\}$, and $?\bot$ otherwise. In the first case, $[\![\bigoplus\Gamma]\!]^M = [\![\bigcup\Gamma]\!]^M$ because $[\![\bigcup\Gamma]\!]^M = [\![\bigcup(\Gamma \setminus \{?\bot\})]\!]^M$; in the second, $[\![\bigoplus\Gamma]\!]^M = [\![\bigcup\Gamma]\!]^M$ because $[\![\bigcup\varnothing]\!]^M = [\![\bigcup\{?\bot\}]\!]^M = [\![?\bot]\!]^M = \varnothing$. □

Proposition 2. *Let M be an epistemic model and σ, ρ two* LCC *programs. Then,*

$$[\![\sigma; \rho]\!]^M = [\![\sigma \odot \rho]\!]^M$$

Proof. Take any epistemic model M. Equation (19) states that $\sigma \odot \rho$ differs from $\sigma;\rho$ only when either σ or else ρ is $?\bot$ or $?\top$. But, in such cases:

- $[\![\sigma;?\bot]\!]^M = [\![?\bot;\sigma]\!]^M = [\![?\bot]\!]^M$; hence, $[\![\sigma;\rho]\!]^M = [\![\sigma \odot \rho]\!]^M$.
- $[\![\sigma;?\top]\!]^M = [\![?\top;\sigma]\!]^M = [\![\sigma]\!]^M$; hence, $[\![\sigma;\rho]\!]^M = [\![\sigma \odot \rho]\!]^M$.

□

The rest of this section is devoted to prove that the function μ^{U} returns an LCC program that is semantically equivalent to the one returned by the program transformer T^{U} of [6].

Lemma 1. *Let* $\mathsf{U} = (\mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{sub})$ *be an action model with* $\mathsf{e}_i, \mathsf{e}_j \in \mathsf{E}$*; let* π *be an* LCC *program. For any epistemic model* M*,*

$$[\![T^{\mathsf{U}}_{ij}(\pi)]\!]^M = [\![\mu^{\mathsf{U}}(\pi)[i,j]]\!]^M$$

Proof. By induction on the complexity of π. Let M be an epistemic model; then

(Base Cases: a and $?\varphi$) Trivial, as the definitions of T^{U}_{ij} and $\mu^{\mathsf{U}}(\pi)[i,j]$ are identical for both a and $?\varphi$.

(Ind. Case $\pi_1 \cup \pi_2$) Suppose (Ind. Hyp.) the claim holds for π_1 and π_2. Then

$$\begin{aligned}
[\![T^{\mathsf{U}}_{ij}(\pi_1 \cup \pi_2)]\!]^M &= [\![T^{\mathsf{U}}_{ij}(\pi_1) \cup T^{\mathsf{U}}_{ij}(\pi_2)]\!]^M && \text{(Def. 6)}\\
&= [\![T^{\mathsf{U}}_{ij}(\pi_1)]\!]^M \cup [\![T^{\mathsf{U}}_{ij}(\pi_2)]\!]^M && \text{(Def. of } [\![\cdot]\!]^M\text{)}\\
&= [\![\mu^{\mathsf{U}}(\pi_1)[i,j]]\!]^M \cup [\![\mu^{\mathsf{U}}(\pi_2)[i,j]]\!]^M && \text{(Ind. Hyp.)}\\
&= [\![\mu^{\mathsf{U}}(\pi_1)[i,j] \cup \mu^{\mathsf{U}}(\pi_2)[i,j]]\!]^M && \text{(Def. of } [\![\cdot]\!]^M\text{)}\\
&= [\![\oplus\{\mu^{\mathsf{U}}(\pi_1)[i,j], \mu^{\mathsf{U}}(\pi_2)[i,j]\}]\!]^M && \text{(Prop. 1)}\\
&= [\![\mu^{\mathsf{U}}(\pi_1 \cup \pi_2)[i,j]]\!]^M && \text{(Def. of } \mu^{\mathsf{U}}(\pi_1 \cup \pi_2) \text{ in (16))}
\end{aligned}$$

(Ind. Case $\pi_1;\pi_2$) Suppose (Ind. Hyp.) the claim holds for π_1 and π_2. Then

$$
\begin{aligned}
[\![T^{\mathsf{U}}_{ij}(\pi_1;\pi_2)]\!]^M &= [\![\textstyle\bigcup_{k=0}^{n-1}(T^{\mathsf{U}}_{ik}(\pi_1);T^{\mathsf{U}}_{kj}(\pi_2))]\!]^M && \text{(Def. 6)}\\
&= \textstyle\bigcup_{k=0}^{n-1}\left([\![T^{\mathsf{U}}_{ik}(\pi_1)]\!]^M \circ [\![T^{\mathsf{U}}_{kj}(\pi_2)]\!]^M\right) && \text{(Def. of } [\![\cdot]\!]^M)\\
&= \textstyle\bigcup_{k=0}^{n-1}\left([\![\mu^{\mathsf{U}}(\pi_1)[i,k]]\!]^M \circ [\![\mu^{\mathsf{U}}(\pi_2)[k,j]]\!]^M\right) && \text{(Ind. Hyp.)}\\
&= [\![\textstyle\bigcup_{k=0}^{n-1}\left(\mu^{\mathsf{U}}(\pi_1)[i,k];\mu^{\mathsf{U}}(\pi_2)[k,j]\right)]\!]^M && \text{(Def. of } [\![\cdot]\!]^M)\\
&= [\![\textstyle\bigcup_{k=0}^{n-1}\left(\mu^{\mathsf{U}}(\pi_1)[i,k]\odot\mu^{\mathsf{U}}(\pi_2)[k,j]\right)]\!]^M && \text{(Prop. 2)}\\
&= [\![\textstyle\bigoplus\{\mu^{\mathsf{U}}(\pi_1)[i,k]\odot\mu^{\mathsf{U}}(\pi_2)[k,j] \mid 0\leq k\leq n-1\}]\!]^M && \text{(Prop. 1)}\\
&= [\![\mu^{\mathsf{U}}(\pi_1;\pi_2)[i,j]]\!]^M && \text{(Def. of } \mu^{\mathsf{U}}(\pi_1;\pi_2) \text{ in (18))}
\end{aligned}
$$

(Ind. Case π^*) Suppose (Ind. Hyp.) the claim holds for π and observe how $[\![\pi^*]\!]^M = [\![?\top \cup (\pi;\pi^*)]\!]^M$. Now,

$$
\begin{aligned}
[\![T^{\mathsf{U}}_{ij}(\pi^*)]\!]^M &= [\![T^{\mathsf{U}}_{ij}(?\top\cup\pi;\pi^*)]\!]^M\\
&= [\![T^{\mathsf{U}}_{ij}(?\top)]\!]^M \cup [\![\textstyle\bigcup_{k=0}^{n-1}(T^{\mathsf{U}}_{ik}(\pi);T^{\mathsf{U}}_{kj}(\pi^*))]\!]^M && \text{(Def. 6)}\\
&= [\![T^{\mathsf{U}}_{ij}(?\top)]\!]^M \cup \textstyle\bigcup_{k=0}^{n-1}\left([\![T^{\mathsf{U}}_{ik}(\pi)]\!]^M \circ [\![T^{\mathsf{U}}_{kj}(\pi^*)]\!]^M\right) && \text{(Def. of } [\![\cdot]\!]^M)\\
&= [\![T^{\mathsf{U}}_{ij}(?\top)]\!]^M \cup \textstyle\bigcup_{k=0}^{n-1}\left([\![\mu^{\mathsf{U}}(\pi)[i,k]]\!]^M \circ [\![T^{\mathsf{U}}_{kj}(\pi^*)]\!]^M\right) && \text{(Ind. Hyp.)}
\end{aligned}
$$

The last equality produces n^2 relational equations. By abbreviating $[\![T^{\mathsf{U}}_{ij}(\pi^*)]\!]^M$ as X^{ij} for every $0\leq i,j\leq n-1$, we get

$$
X^{ij} = [\![T^{\mathsf{U}}_{ij}(?\top)]\!]^M \cup \bigcup_{k=0}^{n-1}\left([\![\mu^{\mathsf{U}}(\pi)[i,k]]\!]^M \circ X^{kj}\right) \tag{25}
$$

Thus, it is enough to prove that $[\![\mu^{\mathsf{U}}(\pi^*)[i,j]]\!]^M$ is a solution for X^{ij}. This is shown in the following three propositions about the functions building $\mu^{\mathsf{U}}(\pi^*)$.

Proposition 3. *Take* $\Omega = (\mu^{\mathsf{U}}(\pi) \mid A^{\mathsf{U}})$ *(see* (20)*). Then,*

$$
X^{ij} = [\![\Omega[i,j+n]]\!]^M \cup \bigcup_{k=0}^{n-1}\left([\![\Omega[i,k]]\!]^M \circ X^{kj}\right) \tag{26}
$$

Proof. It will be shown that the right-hand side (r.h.s.) of (25) and (26) coincide. Their respective rightmost parts are equivalent since, for $0\leq k\leq n-1$, $\Omega[i,k] = \mu^{\mathsf{U}}(\pi)[i,k]$ (recall that Ω is built by adding additional columns at the right of the n first columns of $\mu^{\mathsf{U}}(\pi)$, and the matrix's indexes start from 0). For the leftmost parts,

$$\begin{aligned}
[\![T^{\mathsf{U}}_{ij}(?\top)]\!]^M &= \begin{cases} [\![?(\mathsf{pre}(\mathsf{e}_i) \wedge [\mathsf{U},\mathsf{e}_i]\top)]\!]^M & \text{if } i = j \\ [\![?\bot]\!]^M & \text{otherwise} \end{cases} && \text{(Def. 6)} \\
&= \begin{cases} [\![?\mathsf{pre}(\mathsf{e}_i)]\!]^M & \text{if } i = j \\ [\![?\bot]\!]^M & \text{otherwise} \end{cases} && \text{(as } [\mathsf{U},\mathsf{e}_i]\top \text{ is trivially true)} \\
&= [\![A^{\mathsf{U}}[i,j]]\!]^M = [\![\Omega[i,j+n]]\!]^M && \text{((21) and Def. of } \Omega)
\end{aligned}$$

□

Proposition 4. *For $0 \leq k \leq n-1$, if N is a matrix of size $n \times 2n$ with all cells in columns $0,\ldots,k-1$ equal to $?\bot$, then $Subs_k(Ard_k(N))$ contains all cells in columns $0,\ldots,k$ equal to $?\bot$.*

Proof. Start with $Ard_k(N)$. Observe in (23) that the only modified cells are in the k^{th} row. Cell $Ard_k(N)[k,k]$ in the k^{th} column is converted into $?\bot$. With respect to cells in columns from 0 to $k-1$, if they were $?\bot$, they continue being $?\bot$: those cells $N[i,j]$ do not change, if $N[k,k] = ?\bot$, or otherwise are converted by (23) into $N[k,k]^* \odot N[i,j]$ and, by (19), if $N[i,j] = ?\bot$, then $N[k,k]^* \odot N[i,j] = ?\bot$.

Now, call N' the output of $Ard_k(N)$ and observe $Subs_k(N')$'s definition (24): the only cells that change are in rows different to k. With respect to any such row i, the position in the k^{th} column is made $?\bot$. For cells in previous columns, $j < k$, the last case in the definition returns $\oplus\{N'[i,k] \odot N'[k,j], N'[i,j]\}$. But as N' is the result of $Ard_k(N)$, $N'[k,j]$ is $?\bot$ (because, as argued above, $Ard_k(N)$ works over the k^{th} row and keeps the $?\bot$ in columns before k). Also, $N'[i,j] = ?\bot$, as columns $j < k$ are filled with $?\bot$. So $\oplus\{N'[i,k] \odot N'[k,j], N'[i,j]\}$ becomes $\oplus\{N'[i,k] \odot ?\bot, ?\bot\}$ and, by (17) and (19), it is $?\bot$. □

Proposition 5. *Given an $n \times 2n$ matrix N of* LCC *programs, the equations built using (26), with $\Omega = Subs_k(Ard_k(N))$, $0 \leq k \leq n-1$, are correct transformations of the equations built in the same way with $\Omega = N$.*

Proof. As argued in the proof of Proposition 4, $Ard_k(N)$ works only on the k^{th} row. If $N[k,k] = ?\bot$, nothing is done, so according to (26) the equations for X^{kj} ($0 \leq j \leq n-1$) do not change. Otherwise, the k^{th} row of N changes: all cells $N[k,j]$ with $j \neq k$ become $N[k,k]^* \odot N[k,j]$, except $N[k,k]$ which becomes $?\bot$. Then, for every $0 \leq j \leq n-1$, the equation for X^{kj} becomes (using index t instead of k and removing $[\![?\bot]\!]^M \circ X^{kj}$ from the union):

$$X^{kj} = [\![N[k,k]^* \odot N[k,j+n]]\!]^M \cup \bigcup_{\substack{0 \leq t \leq n-1 \\ t \neq k}} \left([\![N[k,k]^* \odot N[k,t]]\!]^M \circ X^{tj} \right)$$

By Proposition 2 and $[\![\cdot]\!]^M$'s definition, this can be rewritten as

$$\begin{aligned} X^{kj} \;=\; & ([\![N[k,k]]\!]^M)^* \circ [\![N[k,j+n]]\!]^M \cup \\ & \textstyle\bigcup_{\substack{0\leq t\leq n-1 \\ t\neq k}} \Big(([\![N[k,k]]\!]^M)^* \circ [\![N[k,t]]\!]^M \circ X^{tj}\Big) \end{aligned} \tag{27}$$

which is an application of Arden's Theorem [20] to the corresponding equation for the original row in N:

$$X^{kj} = [\![N[k,j+n]]\!]^M \cup \bigcup_{0\leq t\leq n-1} \Big([\![N[k,t]]\!]^M \circ X^{tj}\Big) \tag{28}$$

Arden's Theorem (which works on regular algebras, such as LCC programs) gives $X = A^* \circ B$ as a solution for $X = (A \circ X) \cup B$. In (28), X is X^{kj}, A is $[\![N[k,k]]\!]^M$, and B is the union of all terms in the r.h.s. of (28) except $[\![N[k,k]]\!]^M \circ X^{kj}$. Besides Arden's Theorem, from (28) to (27) we use $\circ$'s distribution over $\cup$, $A \circ (B \cup C) = (A \circ B) \cup (A \circ C)$.

Now denote by N' the output of $Ard_k(N)$. We move to $Subs_k(N')$ to show that the equations obtained from it with (26) are correct transformations of the equations built from N'. The only modified cells in $Subs_k(N')$ are in rows different to k, so it only affects equations for X^{ij} with $i \neq k$. According to (26), if $\Omega = N'$, these equations are (using t instead of k):

$$X^{ij} = [\![N'[i,j+n]]\!]^M \cup \bigcup_{t=0}^{n-1} \Big([\![N'[i,t]]\!]^M \circ X^{tj}\Big) \tag{29}$$

The same equation for $\Omega = Subs_k(N')$ becomes the following (we remove from the union the term $[\![?\bot]\!]^M \circ X^{kj}$, as it is equivalent to $\varnothing$):

$$\begin{aligned} X^{ij} \;=\; & [\![\oplus\{N'[i,k] \odot N'[k,j+n], N'[i,j+n]\}]\!]^M \cup \\ & \bigcup_{\substack{0\leq t\leq n-1 \\ t\neq k}} \Big([\![\oplus\{N'[i,k] \odot N'[k,t], N'[i,t]\}]\!]^M \circ X^{tj}\Big) \end{aligned} \tag{30}$$

By using Propositions 1 and 2 and the properties of $[\![\cdot]\!]^M$, equation (30) becomes

$$\begin{aligned} X^{ij} \;=\; & ([\![N'[i,k]]\!]^M \circ [\![N'[k,j+n]]\!]^M) \cup [\![N'[i,j+n]]\!]^M \cup \\ & \bigcup_{\substack{0\leq t\leq n-1 \\ t\neq k}} \Big((([\![N'[i,k]]\!]^M \circ [\![N'[k,t]]\!]^M) \cup [\![N'[i,t]]\!]^M) \circ X^{tj}\Big) \end{aligned} \tag{31}$$

But note that in the equation for X^{kj}, which is the same at N' and $Subs_k(N')$, the k^{th} row of N' is not changed by $Subs_k(N')$:

$$X^{kj} = [\![N'[k, j+n]]\!]^M \cup \bigcup_{\substack{0 \le t \le n-1 \\ t \neq k}} \left([\![N'[k,t]]\!]^M \circ X^{tj} \right) \tag{32}$$

We have eliminated the term $[\![N'[k,k]]\!]^M \circ X^{kj}$ in (32) because $N' = Ard_k(N)$ and by (23), $N'[k,k] = ?\bot$, which produces $[\![N'[k,k]]\!]^M \circ X^{kj} = \varnothing$.

Observe that (31) can be obtained from (29) by replacing X^{kj} by the r.h.s. of (32) and applying the distribution of $\circ$ over $\cup$. So the modified equation (30) is equivalent to correct transformations of the original one (29). □

The proof of the case π^* in Lemma 1 can be finished now. Take the set of relational equations given by (25). By (20), $\mu^{\mathsf{U}}(\pi^*)$ operates by iterating calls to S_k^{U} (with k from 0 to n) with $\Omega = (\mu^{\mathsf{U}}(\pi) \mid A^{\mathsf{U}})$ as the initial argument. Let M_{-1} be Ω and M_k the output of $S_k^{\mathsf{U}}(M_{k-1})$. By Proposition 3, (26) gives equations equivalent to (25). By Proposition 5, the equations are correct for each successive M_k ($0 \le k \le n-1$). As the calls to S_k^{U} are done iteratively with k from 0 to $n-1$, Proposition 4 guarantees that, in M_{n-1}, all cells in columns for 0 to $n-1$ are equal to $?\bot$. Thus, equations (26) for M_{n-1} are:

$$X^{ij} = [\![M_{n-1}[i, j+n]]\!]^M \tag{33}$$

The rightmost union in (26) has disappeared ($M[i,k] = ?\bot$ for $0 \le k \le n-1$, and $[\![?\bot]\!]^M = \varnothing$). Now, by S_k^{U}'s definition in (22), $M_{n-1}[i, j+n] = M_n[i,j] = \mu^{\mathsf{U}}(\pi^*)[i,j]$, so $X^{ij} = [\![\mu^{\mathsf{U}}(\pi^*)[i,j]]\!]^M$. Then, since X^{ij} represents $[\![T_{ij}^{\mathsf{U}}(\pi^*)]\!]^M$,

$$[\![T_{ij}^{\mathsf{U}}(\pi^*)]\!]^M = [\![\mu^{\mathsf{U}}(\pi^*)[i,j]]\!]^M$$

which completes the proof. □

We can now define new translation functions t', r' as follows. Note that t' and r' are defined as the translation functions t, r for formulas φ and programs π proposed in [6], with the only exception of formulas of the form $[\mathsf{U}, \mathsf{e}_i][\pi]\varphi$.[10] Note also the *inside-out* approach in the case $t([\mathsf{U}, \mathsf{e}][\mathsf{U}', \mathsf{f}]\varphi) = t([\mathsf{U}, \mathsf{e}]t([\mathsf{U}', \mathsf{f}]\varphi))$, which requires rule RE_{U} (with $\chi = [\mathsf{U}', \mathsf{f}]\varphi$ and $\psi = t(\chi)$) in order to prove that the translation is indeed provably equivalent (i.e. $\vdash \phi \leftrightarrow t(\phi)$).

[10] Two minor typos for the cases $[\mathsf{U}, \mathsf{e}]p$ and $[\mathsf{U}, \mathsf{e}_i][\pi]\varphi$ are also corrected here w.r.t. [6] (the first was given by $t(\mathsf{pre}(\mathsf{e})) \to \mathsf{sub}(\mathsf{e}, p)$, and the second by $\bigwedge_{j=0}^{n-1} [T_{ij}^{\mathsf{U}}(r(\pi))]t([\mathsf{U}, \mathsf{e}_j]\varphi)$).

$$
\begin{array}{lll}
t'(\top) & = \top & r'(a) = a \\
t'(p) & = p & r'(B) = B \\
t'(\neg\varphi) & = \neg t'(\varphi) & r'(?\varphi) = ?t'(\varphi) \\
t'(\varphi_1 \wedge \varphi_2) & = t'(\varphi_1) \wedge t'(\varphi_2) & r'(\pi_1;\pi_2) = r'(\pi_1); r'(\pi_2) \\
t'([\pi]\varphi) & = [r'(\pi)]t'(\varphi) & r'(\pi_1 \cup \pi_2) = r'(\pi_1) \cup r'(\pi_2) \\
t'([\mathsf{U},\mathsf{e}]\top) & = \top & r'(\pi^*) = (r'(\pi))^* \\
t'([\mathsf{U},\mathsf{e}]p) & = t'(\mathsf{pre}(\mathsf{e})) \to t'(\mathsf{sub}(\mathsf{e},p)) & \\
t'([\mathsf{U},\mathsf{e}]\neg\varphi) & = t'(\mathsf{pre}(\mathsf{e})) \to \neg t'([\mathsf{U},\mathsf{e}]\varphi) & \\
t'([\mathsf{U},\mathsf{e}](\varphi_1 \wedge \varphi_2)) & = t'([\mathsf{U},\mathsf{e}]\varphi) \wedge t'([\mathsf{U},\mathsf{e}]\varphi_2) & \\
t'([\mathsf{U},\mathsf{e}_i][\pi]\varphi) & = \bigwedge_{\substack{0\le j\le n-1 \\ \mu^{\mathsf{U}}(\pi)[i,j]\neq ?\bot}} [r'(\mu^{\mathsf{U}}(\pi)[i,j])]t'([\mathsf{U},\mathsf{e}_j]\varphi) & \\
t'([\mathsf{U},\mathsf{e}][\mathsf{U}',\mathsf{e}']\varphi) & = t'([\mathsf{U},\mathsf{e}]t'([\mathsf{U}',\mathsf{e}']\varphi)) &
\end{array}
$$

Corollary 1. *The translation functions t', r' reduce the language of* LCC *to that of* PDL*. This translation is correct.*

Proof. The effective reduction from LCC to PDL is immediate by inspection. Its correctness follows from that in [6], with Lemma 1 for the case $[\mathsf{U},\mathsf{e}_i][\pi]\varphi$. □

Definition 8. We define a new axiom system for LCC by replacing the reduction axiom for PDL programs with the following

$$
[\mathsf{U},\mathsf{e}_i][\pi]\varphi \quad \leftrightarrow \bigwedge_{\substack{0\le j\le n-1 \\ \mu^{\mathsf{U}}(\pi)[i,j]\neq ?\bot}} [\mu^{\mathsf{U}}(\pi)[i,j]][\mathsf{U},\mathsf{e}_j]\varphi \qquad \text{(prog)}
$$

Corollary 2. *The axiom system for* LCC *from Def. 8 is sound and complete.*

Proof. The only new axiom, that for PDL-programs, is sound by Lemma 1. For completeness, the proof system for PDL is complete, and every LCC formula is provably equivalent to a PDL formula using Corollary 1. □

5 Complexity of the new transformers

The original program transformers in [6] require exponential time due to the use of Kleene's method [11]. Moreover, the size of the transformed formulas of type π^* is also exponential because of the definition of K^{U}_{ijn} (Def. 6).

In order to study the complexity of our program transformers, we first implemented in Prolog both the original program transformers and our matrix calculus. Figure 5 shows the result for our transformers for two kinds of models, *complete* and *chain* models, from 1 to 20 states. The graph's vertical axis, which is shown in

logarithmic scale, presents the number of PDL operators in the transformed program $\mu^{\mathsf{U}}(\pi^*)[n-1,0]$ for n the number of states in the model.

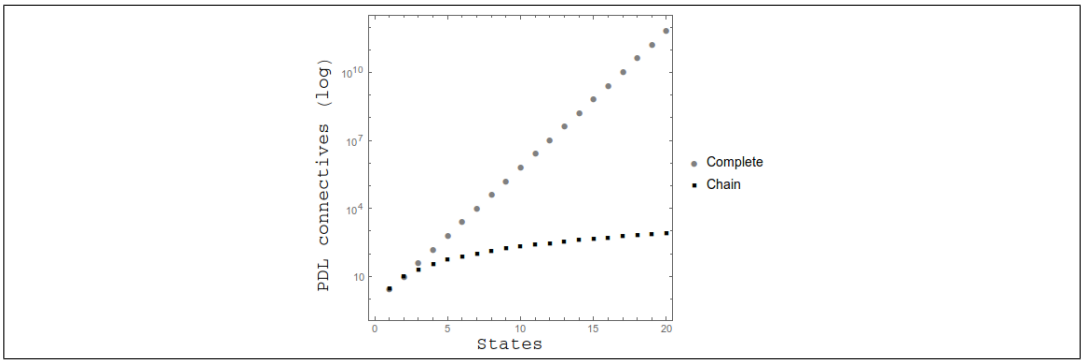


Figure 5: Number of PDL connectives in $\mu^{\mathsf{U}}(\pi^*)$ for different action models

A model is *complete* when it is fully connected, i.e., when each $\mu^{\mathsf{U}}(\pi)[i,j] = s(i,j)$ is an atomic expression (which is not further analysed by the implementations). All values $s(i,j)$ in $\mu^{\mathsf{U}}(\pi)$ are assumed to be different, avoiding simplifications of repeated patterns. The number of operators in $\mu^{\mathsf{U}}(\pi^*)[n-1,0]$ is in the order of 2^{2n}. In the worst case our transformers produce an exponential output, which implies that the required time is also exponential. In a *chain model*, each state is connected with itself, the previous and next one. Thus, $\mu^{\mathsf{U}}(\pi)[i,j]$ is $s(i,j)$ when $i = j$ or $|i-j| = 1$, and $?\bot$ otherwise. Now the number of operators in $\mu^{\mathsf{U}}(\pi^*)[n-1,0]$ is in the order of $2n^2$, so in this case the length of the output is polynomial. We chose models with chain-like structure because it makes it easier to generate models of increasing size with limited connectivity. Similar results can be obtained for other kinds of models with similar average connectivity, as the key is the number of instances of $?\bot$ spread along the matrices.

The results for the original program transformers are not shown in Figure 5 as they are, for both the complete and chain models, as our worst case. (The reason is that they do not benefit from removing superfluous $?\bot$.)

An advantage of our transformers is that they do not require exponential space in cases other than the worst one, in contrast with the original transformers which always perform in the same (exponential) way. An additional advantage can be found in the reusability of the information produced during program transformations. As we argued, working with matrices allows to perform several operations in parallel. Indeed, matrix $\mu^{\mathsf{U}}(\pi)$ contains the transformation of program π within all states in the action model U. As building the matrix with the transformations of a given program involves building the matrices for its subprograms, the information

of each generated matrix is properly stored, and then it can be reused in the same or subsequent transformations within the same action model. This is a practical way to reduce the computation time required for program transformation.

6 Summary and Future Work

In this work, we presented an alternative definition of the program transformers used to obtain reduction axioms in LCC. The proposal uses a matrix treatment of Brzozowski's equational method in order to obtain a regular expression representing the language accepted by a finite automaton. While Brzozowski's method and that used in the original LCC paper [6] are equivalent, the first is computationally more efficient in cases different to the worst one; moreover, the matrix treatment presented here is more synthetic, simple and elegant, thus allowing a simpler implementation.

Towards future work, some definitions used by program transformers (particularly the $\odot$ operation) can be modified to obtain even simpler expressions. For example, $\sigma \odot \rho$ might be defined as σ if $\sigma \neq ?\top = \rho$ and as ρ if $\sigma = ?\top$. Moreover, the algorithm implementing Ard_k and $Subs_k$ functions can be improved by disregarding the $N[i,j]$ elements with $j < k$ or $j > n + k$ (being $N[i,j]$ a $n \times 2n$ matrix), since those are necessarily equal to $?\bot$. These changes, despite not lowering the translation's complexity order, would nevertheless make it more efficient.

Acknowledgements

We would like to thank two anonymous reviewers for their helpful comments. This work is supported by the Spanish Ministry of Economy and Competitiveness, under Research Project FFI2014-56219-P. The research of P. Pardo was supported by a Sofja Kovalevkaja award of the Alexander von Humboldt-Foundation, funded by the German Ministry for Education and Research. This paper is an extended version of a previous conference paper [21] at JELIA 2014.

References

[1] H. van Ditmarsch, W. van der Hoek, B. Kooi, Dynamic Epistemic Logic, Vol. 337 of Synthese Library Series, Springer, 2007.

[2] J. van Benthem, Logical Dynamics of Information and Interaction, Cambridge University Press, 2011.

[3] R. Fagin, J. Y. Halpern, Y. Moses, M. Y. Vardi, Reasoning about knowledge, The MIT Press, Cambridge, Mass., 1995.

[4] R. Parikh, R. Ramanujam, A knowledge based semantics of messages, Journal of Logic, Language and Information 12 (4) (2003) 453–467.

[5] J. van Benthem, J. Gerbrandy, T. Hoshi, E. Pacuit, Merging frameworks for interaction, Journal of Philosophical Logic 38 (5) (2009) 491–526. doi:10.1007/s10992-008-9099-x.

[6] J. van Benthem, J. van Eijck, B. Kooi, Logics of communication and change, Information and Computation 204 (11) (2006) 1620–1662. doi:10.1016/j.ic.2006.04.006.

[7] D. Harel, D. Kozen, J. Tiuryn, Dynamic Logic, MIT Press, Cambridge, MA, 2000.

[8] A. Baltag, L. S. Moss, S. Solecki, The logic of public announcements and common knowledge and private suspicions, in: I. Gilboa (Ed.), TARK, Morgan Kaufmann, San Francisco, CA, USA, 1998, pp. 43–56.

[9] A. Baltag, L. S. Moss, Logics for epistemic programs, Synthese 139 (2) (2004) 165–224.

[10] J. van Benthem, B. Kooi, Reduction axioms for epistemic actions, in: R. Schmidt, I. Pratt-Hartmann, M. Reynolds, H. Wansing (Eds.), Advances in Modal Logic (Number UMCS-04-09-01 in Technical Report Series), Department of Computer Science, University of Manchester, 2004, pp. 197–211.

[11] S. Kleene, Representation of events in nerve nets and finite automata, in: C. E. Shannon, J. McCarthy (Eds.), Automata Studies, Princeton University Press, Princeton, NJ, 1956, pp. 3–41.

[12] J. Sakarovitch, Automata and rational expressions, CoRR abs/1502.03573.
URL http://arxiv.org/abs/1502.03573

[13] J. A. Brzozowski, Derivatives of regular expressions, Journal of the ACM 11 (4) (1964) 481–494.

[14] J. H. Conway, Regular Algebra and Finite Machines, Chapman and Hall, 1971.

[15] Y. Wang, Q. Cao, On axiomatizations of public announcement logic, Synthese 190 (1) (2013) 103–134. doi:10.1007/s11229-012-0233-5.

[16] H. van Ditmarsch, B. Kooi, Semantic results for ontic and epistemic change, in: G. Bonanno, W. van der Hoek, M. Wooldridge (Eds.), Logic and the Foundations of Game and Decision Theory (LOFT7), Vol. 3 of Texts in Logic and Games, Amsterdam University Press, Amsterdam, The Netherlands, 2008, pp. 87–117.

[17] H. Gruber, M. Holzer, Finite automata, digraph connectivity, and regular expression size, in: L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, I. Walukiewicz (Eds.), Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations, Vol. 5126 of Lecture Notes in Computer Science, Springer, 2008, pp. 39–50. doi:10.1007/978-3-540-70583-3_4.

[18] H. Gruber, M. Holzer, Provably shorter regular expressions from finite automata, International Journal of Foundations of Computer Science 24 (8) (2013) 1255–1279. doi:10.1142/S0129054113500330.

[19] D. Kozen, On kleene algebras and closed semirings, in: B. Rovan (Ed.), MFCS, Vol.

452 of Lecture Notes in Computer Science, Springer, 1990, pp. 26–47.

[20] D. N. Arden, Delayed-logic and finite-state machines, in: SWCT (FOCS), IEEE Computer Society, 1961, pp. 133–151.

[21] P. Pardo, E. Sarrión-Morillo, F. Soler-Toscano, F. R. Velázquez-Quesada, Efficient program transformers for translating LCC to PDL, in: E. Fermé, J. Leite (Eds.), Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings, Vol. 8761 of LNCS, Springer, 2014, pp. 253–266.

 Received 22 January 2016