Journal of Logic, Language, and Information **5**: 193–208, 1996.

Eliminating "Converse" from Converse PDL

GIUSEPPE DE GIACOMO
Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy
Email: degiacomo@dis.uniroma1.it

Abstract. In this paper we show that it is possible to eliminate the "converse" operator from the propositional dynamic logic *CPDL* (*Converse PDL*), without compromising the soundness and completeness of inference for it. Specifically we present an encoding of *CPDL* formulae into *PDL* that eliminates the converse programs from a *CPDL* formula, but adds enough information so as not to destroy its original meaning with respect to satisfiability, validity, and logical implication. Notably, the resulting *PDL* formula is polynomially related to the original one. This fact allows one to build inference procedures for *CPDL*, by encoding *CPDL* formulae into *PDL*, and then running an inference procedure for *PDL*.

Key words: Propositional dynamic logics, logics of programs, modal logics, decision procedures.

1. Introduction

Propositional dynamic logics are modal logics originally developed for specifying and reasoning on program schemata. Over the years, they have proved to be a valuable theoretical tool in many areas of Computer Science, Logic, Computational Linguistics, and Artificial Intelligence (e.g. (Kozen and Tiuryn, 1990; Stirlink 1992; Van Benthem et al., 1994; Van Benthem and Bergstra, 1995; Blackburn and Spaan, 1993; Halpern and Moses, 1992; Friedman and Halpern, 1994; Schild, 1991)). In particular many inference procedures, decidability results, and complexity results in such areas rely on research done within propositional dynamic logics.

In this paper we consider two well-known propositional dynamic logics, namely *PDL* and *CPDL*. *PDL* is the original propositional dynamic logic defined in (Fisher and Ladner, 1979), whereas *CPDL*, also defined in (Fisher and Ladner, 1979), extends *PDL* by including a special construct to denote the "converse" of a program. Such a construct allows for the expressing of facts about states preceding the current one, i.e. facts about states that can be reached by executing a given program backward.*

* There are uses of propositional dynamic logics where the ability of denoting converse programs is essential. For example, when propositional dynamic logics are applied in the context of knowledge representation formalisms based on classes and links, converse programs are necessary in order to navigate links in both directions (De Giacomo and Lenzerini, 1994; De Giacomo and Lenzerini, 1995; De Giacomo, 1995).

We show that is possible to eliminate the "converse" operator from *CPDL*, without compromising the soundness and completeness of inference for it. Specifically we present an intuitive encoding of *CPDL* formulae into *PDL* that eliminates the converse programs from a *CPDL* formula, but adds enough information so as not to destroy its original meaning with respect to satisfiability, validity, and logical implication. Notably the resulting *PDL* formula is polynomially related to the original one.

This encoding on the one hand helps to better understand the nature of the converse operator. On the other hand it puts the basis to build efficient – in practical cases – inference procedures for *CPDL*. In fact the encoding allows one to build inference procedures for *CPDL*, by translating *CPDL* formulae into *PDL*, and then running an inference procedure for *PDL*. We discuss this issue further, at the end of the paper.

In fact the technique used for deriving the encoding is quite general. The author has used such a technique to prove decidability and to characterize the computational complexity of several variants of propositional dynamic logics (De Giacomo and Lenzerini, 1994; De Giacomo and Lenzerini, 1995; De Giacomo, 1995), which include constructs as "graded modalities" (Fattorosi-Barnaba and De Caro, 1985; Van der Hoek and De Rijke, 1995) and "nominals" (Passy and Tinchev, 1991; Gargov and Goranko, 1993). Intuitively, the technique is based on two main points. Let the "Source Logic" be *SL* and the "Target Logic" be *TL* (in this paper these logics are *CPDL* and *PDL* respectively):

1. Identify a finite set of axiom schemata in the language of *TL* capturing those characteristics that distinguish *SL* from *TL* (in the present case such axiom schemata are of the form $\phi \rightarrow [P]\langle P^c\rangle\phi$, $\phi \rightarrow [P^c]\langle P\rangle\phi$, and force the binary relation interpreting P^c to be the converse of that interpreting P).
2. Devise a function that, given an *SL* formula Φ, returns a finite "closed"* set of *SL* formulae, whose truth-values univocally determine that of Φ, and that will be used to instantiate the axiom schemata in (1) (in the present case such a set is simply the Fisher–Ladner closure).

Indeed, by instantiating the axiom schemata in (1) to the formulae in (2), and by making use of the capability (see Theorem 1) of propositional dynamic logics of internalizing axioms – not axiom schemata – we can derive a *TL* formula (in the present case, the so called *PDL*-counterpart of a *CPDL* formula, see below) which corresponds to the original *SL* formula, in the sense that it preserves satisfiability, validity, and logical implication. If both the cardinality of the sets in (1) and (2) and the size of their elements are polynomially bounded by the original formula, then so is the formula we get. As we shall see, this is the case for the encoding presented here.

* That is, the truth-value of each formula in the set depends only on the truth-value of formulae already in the set.

The encoding in this paper is probably the best illustration of this technique, since every step is highly intuitive, and proofs go through without major difficulties, exhibiting the details of the technique in a very tidy way.

2. Preliminaries

In this section we introduce the relevant background on propositional dynamic logics.* We mainly focus on *CPDL*, but all the notions and results we introduce for *CPDL* can be immediately reformulated for other propositional dynamic logics, including *PDL*.

Propositional dynamic logics represent a computational process in terms of formulae denoting properties of states, and programs denoting state transition relations. Starting from atomic formulae and atomic programs, which are formulae and programs described simply by a name, complex formulae and programs can be built by means of suitable constructs. The formation rules of *CPDL* are specified by the following abstract syntax:

$$\phi ::= \top \mid \bot \mid A \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \rightarrow \phi_2 \mid \neg\phi \mid \langle r\rangle\phi \mid [r]\phi$$
$$r ::= P \mid r_1 \cup r_2 \mid r_1 ; r_2 \mid r^* \mid r^- \mid \phi?$$

where $\top$ denotes *true*, $\bot$ denotes *false*, A denotes a propositional letter, ϕ (possibly with a subscript) denotes a formula, P denotes an atomic program, and r (possibly with a subscript) denotes a program. *PDL* is obtained from *CPDL* by dropping converse programs r^-.

The semantics of propositional dynamic logics is based on Kripke structures,** which are defined as a triple $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$, where $\mathcal{S}$ denotes a non-empty set of states, $\{\mathcal{R}_P\}$ is a family of binary relations over $\mathcal{S}$ such that each atomic program P is given a meaning through $\mathcal{R}_P$, and Π is a mapping from $\mathcal{S}$ to propositional letters such that $\Pi(s)$ determines the letters that are true in the state s.

The basic semantical relation "ϕ holds at state s of structure M", written $M, s \models \phi$, is defined by induction on the formation of ϕ as follows:

$M, s \models A$ iff $A \in \Pi(s)$
$M, s \models \top$ *always*
$M, s \models \bot$ *never*
$M, s \models \phi_1 \wedge \phi_2$ iff $M, s \models \phi_1$ and $M, s \models \phi_2$
$M, s \models \phi_1 \vee \phi_2$ iff $M, s \models \phi_1$ or $M, s \models \phi_2$
$M, s \models \phi_1 \rightarrow \phi_2$ iff $M, s \models \phi_1$ implies $M, s \models \phi_2$
$M, s \models \neg\phi$ iff $M, s \not\models \phi$
$M, s \models \langle r\rangle\phi$ iff $\exists s'.(s, s') \in \mathcal{R}_r$ and $M, s' \models \phi$
$M, s \models [r]\phi$ iff $\forall s'.(s, s') \in \mathcal{R}_r$ implies $M, s' \models \phi$

* For surveys on propositional dynamic logics, see (Harel, 1984; Kozen and Tiuryn, 1990) and also (Stirlink 1992).

** Also called "transition systems".

where, for every program r, the relation $\mathcal{R}_r$ is defined by induction on the formation of r as follows:

$$\begin{aligned}
&\mathcal{R}_P \subseteq \mathcal{S} \times \mathcal{S}\\
&\mathcal{R}_{r_1 \cup r_2} = \mathcal{R}_{r_1} \cup \mathcal{R}_{r_2}\\
&\mathcal{R}_{R_1;R_2} = \mathcal{R}_{r_1} \circ \mathcal{R}_{r_2} \quad \text{(seq. comp. of } \mathcal{R}_{r_1} \text{ and } \mathcal{R}_{r_2})\\
&\mathcal{R}_{r^*} = (\mathcal{R}_r)^* \quad \text{(refl. trans. closure of } \mathcal{R}_r)\\
&\mathcal{R}_{r^-} = \{(s_1, s_2) \in \mathcal{S} \times \mathcal{S} \mid (s_2, s_1) \in \mathcal{R}_r\}\\
&\mathcal{R}_{\phi?} = \{(s, s) \in \mathcal{S} \times \mathcal{S} \mid M, s \models \phi\}.
\end{aligned}$$

A structure $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ is called a model of a formula ϕ if there exists a state $s \in \mathcal{S}$ such that $M, s \models \phi$. A formula ϕ is satisfiable if there exists a model of ϕ, unsatisfiable otherwise. A formula ϕ is valid in a structure M, written $M \models \phi$, if for all $s \in \mathcal{S}$, $M, s \models \phi$.

We call *axioms*, formulae that are assumed to be valid. Formally, a structure M is a model of an axiom ϕ, if $M \models \phi$. A structure M is a model of a finite set of axioms Γ, written $M \models \Gamma$, if for all $\phi \in \Gamma$ we have $M \models \phi$. We say that a finite set Γ of axioms logically implies a formula ϕ, written $\Gamma \models \phi$, if for all M such that $M \models \Gamma$ we have $M \models \phi$.

Observe that satisfiability of a formula ϕ can be reformulated in terms of logical implication simply as $\emptyset \not\models \neg\phi$. In turn a logical implication $\Gamma \models \phi$ can be reformulated in terms of satisfiability, by making use of the following result (Kozen and Tiuryn, 1990).

THEOREM 1. *Let Γ be a finite set of CPDL axioms, and ϕ a CPDL formula. Then $\Gamma \models \phi$ if and only if the CPDL formula*

$$[(P_1 \cup \ldots \cup P_m \cup P_1^- \cup \ldots \cup P_m^-)^*]\Gamma' \wedge \neg\phi$$

is unsatisfiable, where $P_1, \ldots, P_m$ are all atomic programs occurring in $\Gamma \cup \{\phi\}$ and Γ' is the conjunction of all axioms in Γ.

A similar result holds for most propositional dynamic logics, including *PDL*. In particular, in *PDL*, the formula to check for unsatisfiability is $[(P_1 \cup \ldots \cup P_m)^*]\Gamma' \wedge \neg\phi$. Observe that such a result exploits the power of program constructs (union, reflexive transitive closure) and the "connected model property"* of propositional dynamic logics in order to represent axioms (valid formulae).

In the sequel we assume $\vee$, $[\cdot]$ to be expressed by means of $\neg$, $\wedge$, $\langle\cdot\rangle$. We also assume, without loss of generality, that the converse operator is applied to atomic programs only. Indeed it is easy to check that any *CPDL* formula can be transformed in linear time in the size of the formula so that such an assumption is fulfilled, by making use of following equations: $(r_1; r_2)^- = r_2^-; r_1^-$, $(r_1 \cup r_2)^- = r_1^- \cup r_2^-$, $(r_1^*)^- = (r_1^-)^*$, $(\phi?)^- = \phi?$.

* That is, if a formula has a model, it has a model which is connected (see below).

The *Fisher–Ladner closure* (Fisher and Ladner, 1979) of a *CPDL* formula Φ, denoted $CL(\Phi)$, is the least set F such that $\Phi \in F$ and such that:

$$\begin{array}{ll}
\phi_1 \wedge \phi_2 \in F & \Rightarrow \phi_1, \phi_2 \in F \\
\neg\phi \in F & \Rightarrow \phi \in F \\
\phi \in F & \Rightarrow \neg\phi \in F \text{ (if } \phi \text{ is not of the form } \neg\phi') \\
\langle r \rangle \phi \in F & \Rightarrow \phi \in F \\
\langle r_1; r_2 \rangle \phi \in F & \Rightarrow \langle r_1 \rangle \langle r_2 \rangle \phi \in F \\
\langle r_1 \cup r_2 \rangle \phi \in F & \Rightarrow \langle r_1 \rangle \phi, \langle r_2 \rangle \phi \in F \\
\langle r^* \rangle \phi \in F & \Rightarrow \langle r \rangle \langle r^* \rangle \phi \in F \\
\langle \phi'? \rangle \phi \in F & \Rightarrow \phi' \in F.
\end{array}$$

Intuitively the notion of Fisher–Ladner closure of a formula is closely related to the notion of set of subformulae in other modal logics: given a formula Φ, $CL(\Phi)$ includes all the formulae that play some role in establishing the truth-value of Φ. Both the number and the size of the formulae in $CL(\Phi)$ are linearly bounded by the size of Φ (Fisher and Ladner, 1979). Note that, by definition, if $\phi \in CL(\Phi)$, then $CL(\phi) \subseteq CL(\Phi)$.

Let us denote the *empty sequence of programs* by the program ε, and define $\langle \varepsilon \rangle \phi \doteq \phi$ and $[\varepsilon]\phi \doteq \phi$. We call *Post*$(r)$ the set of programs defined by induction on the formation of r as follows ($a = P \mid P^-$):

$$\begin{array}{ll}
Post(a) & = \{\varepsilon, a\} \\
Post(r_1; r_2) & = \{r_1'; r_2 \mid r_1' \in Post(r_1)\} \cup Post(r_2) \\
Post(r_1 \cup r_2) & = Post(r_1) \cup Post(r_2) \\
Post(r_1^*) & = \{r_1'; r_1^* \mid r_1' \in Post(r_1)\} \\
Post(\phi?) & = \{\varepsilon, \phi?\}.
\end{array}$$

Intuitively, the set *Post*(r) is formed by the programs that are (not necessarily proper) "postfix" of the program r. The following proposition holds.

PROPOSITION 2. *Let* $\langle r \rangle \phi$ *be a formula. For all* $r' \in Post(r)$, $\langle r' \rangle \phi \in CL(\langle r \rangle \phi)$.

Proof. By induction on the formation of r.

- $r = a$ or $r = \phi'?$. Then $Post(r) = \{\varepsilon, r\}$. By definition, both $\phi \in CL(\langle r \rangle \phi)$ and $\langle r \rangle \phi \in CL(\langle r \rangle \phi)$.
- $r = r_1; r_2$. Then $Post(r_1; r_2) = \{r_1'; r_2 \mid r_1' \in Post(r_1)\} \cup Post(r_2)$.
  Since r_1 is a subprogram of $r_1; r_2$, by induction hypothesis, for all $r_1' \in Post(r_1)$:

  $$\langle r_1' \rangle (\langle r_2 \rangle \phi) \in CL(\langle r_1 \rangle \langle r_2 \rangle \phi) \subseteq CL(\langle r_1; r_2 \rangle \phi).$$

  On the other hand, since r_2 is a subprogram of $r_1; r_2$, by induction hypothesis, for all $r_2' \in Post(r_2)$:

$$\langle r_2' \rangle \phi \in CL(\langle r_2 \rangle \phi) \subseteq CL(\langle r_1; r_2 \rangle \phi).$$

– $r = r_1 \cup r_2$. Then $Post(r_1 \cup r_2) = Post(r_1) \cup Post(r_2)$. By induction hypothesis, for $i = 1, 2$, for all $r_i' \in Post(r_i)$:

$$\langle r_i' \rangle \phi \in CL(\langle r_i \rangle \phi) \subseteq CL(\langle r_1 \cup r_2 \rangle \phi).$$

– $r = r_1^*$. Then $Post(r_1^*) = \{r_1'; r_1^* \mid r_1' \in Post(r_1)\}$. By induction hypothesis, for all $r_1' \in Post(r_1)$:

$$\langle r_1' \rangle (\langle r_1^* \rangle \phi) \in CL(\langle r_1 \rangle \langle r_1^* \rangle \phi) \subseteq CL(\langle r_1^* \rangle \phi).$$

■

Finally, we introduce the notion of *path*. Intuitively a path describes the sequence of states a given run of a program goes through.* Formally, a *path* in a structure M is a sequence $(s_0, \ldots, s_q)$ of states of M ($q \geq 0$), such that for each $i = 1, \ldots, q$, $(s_{i-1}, s_i) \in \mathcal{R}_a$ for some $a = P \mid P^-$. The length of $(s_0, \ldots, s_q)$ is q. We inductively define the set of paths $Paths_M(r)$ of a program r in a structure M, as follows (the notation r^i stands for i repetitions of r -i.e., $r^1 = r$, and $r^i = r; r^{i-1}$:

$$\begin{array}{lcl}
Paths_M(a) & = & \mathcal{R}_a \ (a = P \mid P^-) \\
Paths_M(r_1 \cup r_2) & = & Paths_M(r_1) \cup Paths_M(r_2) \\
Paths_M(r_1; r_2) & = & \{(s_0, \ldots, s_u, \ldots, s_q) \mid (s_0, \ldots, s_u) \in Paths_M(r_1) \\
 & & \text{and } (s_u, \ldots, s_q) \in Paths_M(r_2)\} \\
Paths_M(r^*) & = & \{(s) \mid s \in \mathcal{S}\} \cup (\bigcup_{i>0} Paths_M(r^i)) \\
Paths_M(\phi'?) & = & \{(s) \mid M, s \models \phi'\}.
\end{array}$$

The next two propositions describe the basic properties of paths. Proposition 3 concerns paths whose length is 0: it says that if a formula $\langle r \rangle \phi$ is satisfied in a state s by means of a path whose length is 0, then there is a formula $\langle \phi_1?; \ldots; \phi_g? \rangle \phi$, where the tests $\phi_1?, \ldots, \phi_g?$ occur in r, that is satisfied in s and implies $\langle r \rangle \phi$.

PROPOSITION 3. *Let M be a structure and $\langle r \rangle \phi$ a formula, such that: $M, s \models \langle r \rangle \phi$, $(s) \in Paths_M(r)$, and $M, s \models \phi$. Then there exists a formula $\langle \phi_1?; \ldots; \phi_g? \rangle \phi$, with $g \geq 0$, such that:*
- *all tests $\phi_i?$ occur in r;*
- *$M, s \models \langle \phi_1?; \ldots; \phi_g? \rangle \phi$;*
- *$\langle \phi_1?; \ldots; \phi_g? \rangle \phi \rightarrow \langle r \rangle \phi$ is valid.*

* The notion of path used here has the same role as the one of *trajectory* used in (Ben-Ari et al., 1982), and that of *execution sequence* in (Streett, 1982). However, the technical details of the various notions differ. In order to make the paper complete and self-contained, we are going to give full-fledged proofs of the basic properties of paths.

Proof. By induction on the formation of r.

(1) $r = \phi'?$.

The thesis holds trivially.

(2) $r = r_1; r_2$.

$M, s \models \langle r_1; r_2\rangle\phi$ and $(s) \in \mathit{Paths}_M(r)$ implies that $M, s \models \langle r_1\rangle\langle r_2\rangle\phi$ and $(s) \in \mathit{Paths}_M(r_1)$ and $(s) \in \mathit{Paths}_M(r_2)$. By induction hypothesis, we can assume that:

- there is a formula $\langle\phi_{1,1}?; \ldots; \phi_{1,g_1}?\rangle\langle r_2\rangle\phi$ such that all tests $\phi_{1,j}?$ occur in r_1, $M, s \models \langle\phi_{1,1}?; \ldots; \phi_{1,g_1}?\rangle\langle r_2\rangle\phi$, and $\langle\phi_{1,1}?; \ldots; \phi_{1,g_1}?\rangle\langle r_2\rangle\phi \rightarrow \langle r_1\rangle\langle r_2\rangle\phi$ is valid;
- there is a formula $\langle\phi_{2,1}?; \ldots; \phi_{2,g_2}?\rangle\phi$ such that all tests $\phi_{2,j}?$ occur in r_2, $M, s \models \langle\phi_{2,1}?; \ldots; \phi_{2,g_1}?\rangle\phi$, and $\langle\phi_{2,1}?; \ldots; \phi_{2,g_2}?\rangle\phi \rightarrow \langle r_2\rangle\phi$ is valid.

Hence, $\langle\phi_{1,1}?; \ldots; \phi_{1,g_1}?; \phi_{2,1}?; \ldots; \phi_{2,g_2}?\rangle\phi$ is such that: (1) all tests $\phi_{i,j}?$ occur in r_1 or r_2 and therefore in r; (2) $M, s \models \langle\phi_{1,1}?; \ldots; \phi_{1,g_1}?; \phi_{2,1}?; \ldots; \phi_{2,g_2}?\rangle\phi$; (3) $\langle\phi_{1,1}?; \ldots; \phi_{1,g_1}?; \phi_{2,1}?; \ldots; \phi_{2,g_2}?\rangle\phi \rightarrow \langle r_1; r_2\rangle\phi$ is valid.

(3) $r = r_1 \cup r_2$.

$M, s \models \langle r_1 \cup r_2\rangle\phi$ implies that, either for $i = 1$ or for $i = 2$, $M, s \models \langle r_i\rangle\phi$ and $(s) \in \mathit{Paths}_M(r_i)$. By induction hypothesis we can assume there is a formula $\langle\phi_{i,1}?; \ldots; \phi_{i,g_i}?\rangle\phi$ such that all tests $\phi_{i,j}?$ occur in r_i, $M, s \models \langle\phi_{i,1}?; \ldots; \phi_{i,g_i}?\rangle\phi$, and $\langle\phi_{i,1}?; \ldots; \phi_{i,g_i}?\rangle\phi \rightarrow \langle r_i\rangle\phi$ is valid. Therefore, considering that $\langle r_i\rangle\phi \rightarrow \langle r_1 \cup r_2\rangle\phi$, we get the thesis.

(4) $r = r_1^*$.

Since $(s) \in \mathit{Paths}_M(r_1^*)$, $\langle r_1^*\rangle\phi$ is equivalent $\phi \vee \langle r_1\rangle\langle r_1^*\rangle\phi$, and $M, s \models \phi$, the thesis holds trivially (with $g = 0$). ■

Proposition 4 concerns paths whose length is greater than 0: it says that if a formula $\langle r\rangle\phi$ is satisfied in a state s by means of a path whose length greater than 0, then there is a formula $\langle\phi_1?; \ldots; \phi_g?; a\rangle\langle r'\rangle\phi$, where the tests $\phi_1?, \ldots, \phi_g?$ occur in r, a is the first transition on the path, and $r' \in \mathit{Post}(r)$, which is satisfied in s and implies $\langle r\rangle\phi$.

PROPOSITION 4. *Let M be a structure, and $\langle r\rangle\phi$ a formula such that: $M, s \models \langle r\rangle\phi$, $(s = s_0, \ldots, s_q) \in \mathit{Paths}_M(r)$ with $q > 0$, and $M, s_q \models \phi$. Then there exists a formula $\langle\phi_1?; \ldots; \phi_g?; a\rangle\langle r'\rangle\phi$, with $g \geq 0$, such that:*

- *all tests $\phi_i?$ occur in r;*
- *$r' \in \mathit{Post}(r)$ (and hence $\langle r'\rangle\phi \in CL(\langle r\rangle\phi)$);*
- *$(s_0, s_1) \in \mathcal{R}_a$;*
- *$M, s_1 \models \langle r'\rangle\phi$;*
- *$(s_1, \ldots, s_q) \in \mathit{Paths}_M(r')$;*
- *$\langle\phi_1?; \ldots; \phi_g?; a\rangle\langle r'\rangle\phi \rightarrow \langle r\rangle\phi$ is valid.*

Proof. By induction on the formation of r.

(1) $r = a$.

The thesis holds trivially.

(2) $r = r_1; r_2$.

Let $(s_0, \ldots, s_i)$ be the segment of $(s_0, \ldots, s_q)$ such that $(s_0, \ldots, s_i) \in Paths_M(r_1)$ and $(s_i, \ldots, s_q) \in Paths_M(r_2)$. We consider two cases:

- $i > 0$. Consider that: (1) $M, s_0 \models \langle r_1 \rangle \phi'$ for $\phi' = \langle r_2 \rangle \phi$; (2) $(s_0, \ldots, s_i) \in Paths_M(r_1)$ with $i > 0$; (3) $M, s_i \models \langle r_2 \rangle \phi$. By induction hypothesis, there is a formula $\langle \phi_1?; \ldots; \phi_g?; a \rangle \langle r_1' \rangle \langle r_2 \rangle \phi$ such that:
  - all tests $\phi_i?$ occur in r_1, and hence in r;
  - $r_1' \in Post(r_1)$, and hence $r_1'; r_2 \in Post(r_1; r_2)$;
  - $(s_0, s_1) \in \mathcal{R}_a$;
  - $M, s_1 \models \langle r_1' \rangle \langle r_2 \rangle \phi$, and hence $M, s_1 \models \langle r_1'; r_2 \rangle \phi$;
  - $(s_1, \ldots, s_i) \in Paths_M(r_1')$ with $i \leq q$, and hence $(s_1, \ldots, s_q) \in Paths_M(\langle r_1'; r_2 \rangle \phi)$;
  - $\langle \phi_1?; \ldots; \phi_g?; a \rangle \langle r_1' \rangle \langle r_2 \rangle \phi \to \langle r_1 \rangle \langle r_2 \rangle \phi$ is valid, and hence also the formula $\langle \phi_1?; \ldots; \phi_g?; a \rangle \langle r_1'; r_2 \rangle \phi \to \langle r_1; r_2 \rangle \phi$ is valid.
- $i = 0$. By Proposition 3, there exists a formula $\langle \phi_{1,1}?; \ldots; \phi_{1,g_1}? \rangle \langle r_2 \rangle \phi$ such that
  - all tests $\phi_{1,j}?$ occur in r_1;
  - $M, s_0 \models \langle \phi_{1,1}?; \ldots; \phi_{1,g_1}? \rangle \langle r_2 \rangle \phi$;
  - $\langle \phi_{1,1}?; \ldots; \phi_{1,g_1}? \rangle \langle r_2 \rangle \phi \to \langle r_1 \rangle \langle r_2 \rangle \phi$ is valid.

  On the other hand, observe that $\langle r_2 \rangle \phi$ is such that: (1) $M, s \models \langle r_2 \rangle \phi$; (2) $(s = s_0, \ldots, s_q) \in Paths_M(r_2)$ with $q > 0$; (3) $M, s_q \models \phi$. Therefore, by induction hypothesis, there is a formula $\langle \phi_{2,1}?; \ldots; \phi_{2,g_2}?; a \rangle \langle r_2' \rangle \phi$ such that
  - all tests $\phi_{2,j}?$ occur in r_2;
  - $r_2' \in Post(r_2)$ ($\subseteq Post(r_1; r_2)$);
  - $(s_0, s_1) \in \mathcal{R}_a$;
  - $M, s_1 \models \langle r_2' \rangle \phi$;
  - $(s_1, \ldots, s_q) \in Paths_M(r_2')$;
  - $\langle \phi_{2,1}?; \ldots; \phi_{2,g_2}?; a \rangle \langle r_2' \rangle \phi \to \langle r_2 \rangle \phi$ is valid.

  Hence the formula $\langle \phi_{1,1}?; \ldots; \phi_{1,g_1}?; \phi_{2,1}?; \ldots; \phi_{2,g_2}?; a \rangle \langle r_2' \rangle \phi$ is such that
  - all tests $\phi_{i,j}?$ occur in either in r_1 or in r_2;
  - $r_2' \in Post(r_1; r_2)$;
  - $(s_0, s_1) \in \mathcal{R}_a$;
  - $M, s_1 \models \langle r_2' \rangle \phi$;
  - $(s_1, \ldots, s_q) \in Paths_M(r_2')$;
  - $\langle \phi_{1,1}?; \ldots; \phi_{1,g_1}? \rangle \langle \phi_{2,1}?; \ldots; \phi_{2,g_2}?; a \rangle \langle r_2' \rangle \phi \to \langle r_1 \rangle \langle r_2 \rangle \phi$ is valid, and hence also $\langle \phi_{1,1}?; \ldots; \phi_{1,g_1}?; \phi_{2,1}?; \ldots; \phi_{2,g_2}?; a \rangle \langle r_2' \rangle \phi \to \langle r_1; r_2 \rangle \phi$ is valid.

(3) $r = r_1 \cup r_2$.

$M, s \models \langle r_1 \cup r_2 \rangle \phi$ with $(s = s_0, \ldots, s_q) \in Paths_M(r_1 \cup r_2)$ implies that either for $i = 1$ or $i = 2$: (1) $M, s \models \langle r_i \rangle \phi$; (2) $(s = s_0, \ldots, s_q) \in Paths_M(r_i)$ with $q > 0$; (3) $M, s_q \models \phi$. Thus, by induction hypothesis, there is a formula $\langle \phi_{i,1}?; \ldots; \phi_{i,g_1}?; a_i \rangle \langle r_i' \rangle \phi$ such that:

- all tests $\phi_{i,j}?$ occur in r_i, and hence in $r_1 \cup r_2$;

- $r'_i \in Post(r_i) \subseteq Post(r_1 \cup r_2)$;
- $(s_0, s_1) \in \mathcal{R}_a$;
- $M, s_1 \models \langle r'_i \rangle \phi$;
- $(s_1, \ldots, s_q) \in Paths_M(r'_i)$;
- $\langle \phi_{i,1}?; \ldots; \phi_{i,g_i}?; a_i \rangle \langle r'_i \rangle \phi \rightarrow \langle r_i \rangle \phi$ is valid, and therefore, considering that, $\langle r_i \rangle \phi \rightarrow \langle r_1 \cup r_2 \rangle \phi$ is valid, we get that $\langle \phi_{i,1}?; \ldots; \phi_{i,g_i}?; a_i \rangle \langle r'_i \rangle \phi \rightarrow \langle r_1 \cup r_2 \rangle \phi$ is valid.

(4) $r = r_1^*$.

Since $q > 0$, we have that $M, s \models \langle r_1^* \rangle \phi$ implies $M, s \models \langle r_1 \rangle \langle r_1^* \rangle \phi$, and furthermore there is a segment $(s_0, \ldots, s_i)$ of $(s_0, \ldots, s_q)$ with $0 < i \leq q$, such that $(s_0, \ldots, s_i) \in Paths_M(r_1)$ and $(s_i, \ldots, s_q) \in Paths_M(r_1^*)$. Thus we have: (1) $M, s_0 \models \langle r_1 \rangle \phi'$ with $\phi' = \langle r_1^* \rangle \phi$; (2) $(s_0, \ldots, s_i) \in Paths_M(r_1)$ with $i > 0$; (3) $M, s_i \models \langle r_1^* \rangle \phi$. By induction hypothesis there exists a formula $\langle \phi_1?; \ldots; \phi_g?; a \rangle \langle r'_1 \rangle \langle r_1^* \rangle \phi$ such that

- all tests $\phi_i?$ occur in r_1, and hence in r_1^*;
- $r'_1 \in Post(r_1)$, and hence $r'_1; r_1^* \in Post(r_1^*)$;
- $(s_0, s_1) \in \mathcal{R}_a$;
- $M, s_1 \models \langle r'_1 \rangle \langle r_1^* \rangle \phi$, and hence $M, s_1 \models \langle r'_1; r_1^* \rangle \phi$;
- $(s_1, \ldots, s_i) \in Paths_M(r'_1)$, and hence $(s_1, \ldots, s_q) \in Paths_M(r'_1; r_1^*)$;
- $\langle \phi_1?; \ldots; \phi_g?; a \rangle \langle r'_1 \rangle \langle r_1^* \rangle \phi \rightarrow \langle r_1 \rangle \langle r_1^* \rangle \phi$ is valid, hence also the formula $\langle \phi_1?; \ldots; \phi_g?; a \rangle \langle r'_1; r_1^* \rangle \phi \rightarrow \langle r_1; r_1^* \rangle \phi$ is valid. Therefore, considering that $\langle r_1; r_1^* \rangle \phi \rightarrow \langle r_1^* \rangle \phi$, we get that $\langle \phi_1?; \ldots; \phi_g?; a \rangle \langle r'_1; r_1^* \rangle \phi \rightarrow \langle r_1^* \rangle \phi$ is valid.

■

3. The Encoding

We now show the encoding of *CPDL* formulae into *PDL*. More precisely, we exhibit a mapping γ from *CPDL* formulae to *PDL* formulae such that, for any *CPDL* formula Φ, Φ is satisfiable if and only if $\gamma(\Phi)$ is satisfiable. The formula $\gamma(\Phi)$, whose size is polynomial with respect to the size of Φ, is said to be the *PDL-counterpart of* Φ. We assume without loss of generality that in Φ the converse operator is applied to atomic programs only.

DEFINITION. Let Φ be a *CPDL* formula with the converse operator applied to atomic programs only. We define the *PDL-counterpart* $\gamma(\Phi)$ of Φ as the conjunction of two formulae, $\gamma(\Phi) = \gamma_1(\Phi) \wedge \gamma_2(\Phi)$, where:

- $\gamma_1(\Phi)$ is obtained from the original formula Φ by replacing each occurrence of P^- with a new atomic program P^c, for all atomic programs P occurring in Φ.
- $\gamma_2(\Phi) = [(P_1 \cup \ldots \cup P_m \cup P_1^c \cup \ldots \cup P_m^c)^*]\gamma_2^1 \wedge \ldots \wedge \gamma_2^g$, where $P_1, \ldots, P_m$ are all atomic programs appearing in Φ, and with a conjunct γ_2^i of the form

$$(\phi \rightarrow [P]\langle P^c\rangle\phi) \wedge (\phi \rightarrow [P^c]\langle P\rangle\phi)$$

for every $\phi \in CL(\gamma_1(\Phi))$ and $P \in \{P_1, \ldots, P_m\}$.

THEOREM 5. *Let Φ be a CPDL formula, and $\gamma(\Phi)$ its PDL-counterpart. Then $\gamma(\Phi)$ is a PDL formula, and its size is polynomially related to the size of Φ.*

Proof. $\gamma(\Phi)$ is obviously a *PDL* formula. Furthermore, since both the number and the size of the formulae in $CL(\gamma_1(\Phi))$ are bounded by the size $|\gamma_1(\Phi)|$ of $\gamma_1(\Phi)$, and $|\gamma_1(\Phi)| = |\Phi|$, it follows that $|\gamma(\Phi)| = O(m \cdot |\Phi| \cdot |\Phi|)$, where m is the number of atomic programs occurring in Φ. ∎

Note that, although the size of $\gamma(\Phi)$ is $O(m \cdot |\Phi| \cdot |\Phi|)$, the special form of $\gamma(\Phi)$ guarantees that $|CL(\gamma(\Phi))| = O(m \cdot |CL(\Phi)|)$, i.e. the size of the Fisher–Ladner closure of $\gamma(\Phi)$ is essentially the same as that of Φ multiplied by the number of atomic programs in Φ. This observation is of significant practical interest since the efficiency of several inference procedures for *PDL* depends, in fact, on the size of the Fisher–Lander closure of the formula, and only indirectly on the size of the formula.

The purpose of $\gamma_1(\Phi)$ is to eliminate the converse of atomic programs (the only converse programs) from Φ and replace them with new atomic programs. Each new atomic program P^c is intended to represent P^- (the converse of the atomic program P) in $\gamma_1(\Phi)$.

The purpose of $\gamma_2(\Phi)$ is to constrain the models M of $\gamma(\Phi)$ so that, for all $\phi \in CL(\gamma_1(\Phi))$, for all states s of M, if ϕ holds in s then all the P-successors of s have a P^c-successor where ϕ holds, and similarly all the P^c-successors of s have a P-successor where ϕ holds. We shall show that, as far as satisfiability (but also validity and logical implication) is concerned, this allows us to faithfully represent the converse of P by means of P^c.

First of all, observe that if instead of $\gamma_2(\Phi)$ we imposed, for each P, the two axiom schemata (ϕ any formula):

$$\phi \rightarrow [P]\langle P^c\rangle\phi$$
$$\phi \rightarrow [P^c]\langle P\rangle\phi$$

then the models of $\gamma_1(\Phi)$ would be isomorphic to the models of Φ. In fact, the above axiom schemata are identical to the ones used in the axiomatization of *CPDL* to force the program P^- to be the converse of P. However the resulting logic would not be *PDL* but trivially *CPDL*.

Instead, $\gamma_2(\Phi)$ can be thought as a finite instantiation of the above two axiom schemata: one instance for each formula in $CL(\Phi)$.* Although imposing the validity of such a finite instantiation does not suffice to guarantee the isomorphism of

* Actually, $\gamma_2(\Phi)$ already takes into account the reduction from logical implication to satisfiability of Theorem 1.

the models of $\gamma_1(\Phi)$ and Φ, we show that it suffices to guarantee that $\gamma_1(\Phi)$ has a model if and only if Φ has a model.

It is a standard result that if a *CPDL* formula Φ has a model, then it has a connected model, where a model $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ of Φ is a *connected model*, if for some $ss \in \mathcal{S}$:

- $M, ss \models \Phi$;
- $\mathcal{S} = \{t \mid (ss, t) \in (\bigcup_P \mathcal{R}_P \cup \mathcal{R}_{P^-})^*\}$.

Let Φ be either a *CPDL* formula or a *PDL* formula. We call a structure $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ a *structure of* Φ, if every atomic program P and every atomic proposition A occurring in Φ is interpreted in M, i.e. $\mathcal{R}_P$ appears in M, and A appears in the co-domain of Π.

In the following we use π as an abstraction for both P and P^c. Moreover, π^c denotes P^c, if $\pi = P$, and it denotes P, if $\pi = P^c$.

Let $M = (\mathcal{S}, \{\mathcal{R}_\pi\}, \Pi)$ be a connected model of $\gamma(\Phi)$. We call the *c-closure* of M, the structure $M' = (\mathcal{S}', \{\mathcal{R}'_\pi\}, \Pi')$ of $\gamma(\Phi)$, defined as follows:

- $\mathcal{S}' = \mathcal{S}$;
- $\mathcal{R}'_\pi = \mathcal{R}_\pi \cup \{(t, s) \mid (s, t) \in \mathcal{R}_{\pi^c}\}$, for each atomic program π in $\gamma(\Phi)$;
- $\Pi' = \Pi$.

Note that in the c-closure M' of a model M, each $\mathcal{R}'_P$ of M' is obtained from $\mathcal{R}_P$ of M by including, for each pair (s, t) in $\mathcal{R}_{P^c}$, the pair (t, s) in $\mathcal{R}'_P$, and similarly each $\mathcal{R}'_{P^c}$ is obtained from R_{P^c} by including, for each pair (s, t) in R_P, the pair (t, s) in $\mathcal{R}'_{P^c}$. As a result in the c-closure of a model each atomic program P^c is interpreted as the converse of P.

The next lemma is the core of the results in this paper. Intuitively it says that the c-closure of a connected model is equivalent to the original model with respect to the formulae in $CL(\gamma_1(\Phi))$.

LEMMA 6. *Let* $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ *be a connected model of* $\gamma(\Phi)$, *and* $M' = (\mathcal{S}', \{\mathcal{R}'_P\}, \Pi')$ *its c-closure. Then, for every* $s \in \mathcal{S}\,(= \mathcal{S}')$, *and every* $\phi \in CL(\gamma_1(\Phi))$:

$$M, s \models \phi \textit{ iff } M', s \models \phi.$$

Proof. We prove the lemma by induction on the formation of ϕ (called formula induction in the following).

- $\phi = A$. $M, s \models A$ iff $A \in \Pi(s)$ iff, by construction of M', $A \in \Pi'(s)$ iff $M', s \models A$.
- $\phi = \neg\phi'$. $M, s \models \neg\phi'$ iff $M, s \not\models \phi'$ iff, by the formula induction hypothesis, $M', s \not\models \phi'$ iff $M', s \models \neg\phi'$.
- $\phi = \phi_1 \wedge \phi_2$. $M, s \models \phi_1 \wedge \phi_2$ iff $M, s \models \phi_1$ and $M, s \models \phi_2$ iff, by the formula induction hypothesis, $M', s \models \phi_1$ and $M', s \models \phi_2$ iff $M', s \models \phi_1 \wedge \phi_2$.

– $\phi = \langle r \rangle \phi'$. $\Rightarrow$. $M, s \models \langle r \rangle \phi'$ iff there is a path $(s = s_0, \ldots, s_q) \in \mathit{Paths}_M(r)$

such that $M, s_q \models \phi'$. We show that $M', s \models \langle r \rangle \phi'$, by induction on the length of the path (called path induction in the following).

$q = 0$. In this case $(s = s_0) \in \mathit{Paths}_M(r)$ and $M, s \models \phi'$. Then, by Proposition 3, there exists a formula $\langle \phi_1?; \ldots; \phi_g? \rangle \phi'$ such that:
- all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $\langle r \rangle \phi'$;
- $M, s \models \langle \phi_1?; \ldots; \phi_g? \rangle \phi'$;
- $\langle \phi_1?; \ldots; \phi_g? \rangle \phi' \rightarrow \langle r \rangle \phi'$ is valid.

By the formula induction hypothesis, for every $\phi_x \in \{\phi_1, \ldots, \phi_g, \phi'\}$, we have that $M, s \models \phi_x$ iff $M', s \models \phi_x$. Hence, since a formula of the form $\langle \phi_1?; \ldots; \phi_g? \rangle \phi'$ is equivalent to $\phi_1 \wedge \ldots \wedge \phi_g \wedge \phi'$, we conclude that $M', s \models \langle r \rangle \phi'$.

$q > 0$. In this case, by Proposition 4, there exists a formula $\langle \phi_1?; \ldots; \phi_g?; \pi \rangle \langle r' \rangle \phi'$ such that:
- all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $\langle r \rangle \phi'$;
- $r' \in \mathit{Post}(r)$, and hence $\langle r' \rangle \phi' \in CL(\langle r \rangle \phi') \subseteq CL(\gamma_1(\Phi))$;
- $(s_0, s_1) \in \mathcal{R}_\pi$;
- $M, s_1 \models \langle r' \rangle \phi'$;
- $(s_1, \ldots, s_q) \in \mathit{Paths}_M(r')$;
- $\langle \phi_1?; \ldots; \phi_g?; \pi \rangle \langle r' \rangle \phi' \rightarrow \langle r \rangle \phi'$ is valid.

By the formula induction hypothesis, for every $\phi_x \in \{\phi_1, \ldots, \phi_g\}$, we have $M, s_0 \models \phi_x$ iff $M', s_0 \models \phi_x$.
By construction of M', $(s_0, s_1) \in \mathcal{R}_\pi$ implies $(s_0, s_1) \in \mathcal{R}'_\pi$.
Considering that $\langle r' \rangle \phi' \in CL(\langle r \rangle \phi') \subseteq CL(\gamma_1(\Phi))$, by the path induction hypothesis, $M, s_1 \models \langle r' \rangle \phi'$ and $(s_1, \ldots, s_q) \in \mathit{Paths}_M(r')$ implies $M', s_1 \models \langle r' \rangle \phi'$.
Hence $M', s_0 \models \langle r \rangle \phi'$.

$\Leftarrow$. $M', s \models \langle r \rangle \phi'$ iff there is a path $(s = s_0, \ldots, s_q) \in \mathit{Paths}_{M'}(r)$ such that $M', s_q \models \phi'$. We prove that $M, s \models \langle r \rangle \phi'$, by induction on the length of the path (called path induction in the following).

$q = 0$. In this case $(s = s_0) \in \mathit{Paths}_{M'}(r)$ and $M', s \models \phi'$. Then, by Proposition 3, there exists a formula $\langle \phi_1?; \ldots; \phi_g? \rangle \phi'$ such that:
- all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $\langle r \rangle \phi'$;
- $M', s \models \langle \phi_1?; \ldots; \phi_g? \rangle \phi'$;
- $\langle \phi_1?; \ldots; \phi_g? \rangle \phi' \rightarrow \langle r \rangle \phi'$ is valid.

By the formula induction hypothesis, for every $\phi_x \in \{\phi_1, \ldots, \phi_g, \phi'\}$, we have that $M', s \models \phi_x$ iff $M, s \models \phi_x$. Hence $M, s \models \langle r \rangle \phi'$.

$q > 0$. In this case, by Proposition 4, there exists a formula $\langle \phi_1?; \ldots; \phi_g?; \pi \rangle \langle r' \rangle \phi'$ such that:

- all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $\langle r\rangle\phi'$;
- $r \in Post(r)$, and hence $\langle r'\rangle\phi' \in CL(\langle r\rangle\phi') \subseteq CL(\gamma_1(\Phi))$;
- $(s_0, s_1) \in \mathcal{R}'_\pi$;
- $M', s_1 \models \langle r'\rangle\phi'$;
- $(s_1, \ldots, s_q) \in Paths_{M'}(r')$;
- $\langle \phi_1?; \ldots; \phi_g?; \pi\rangle\langle r'\rangle\phi' \to \langle r\rangle\phi'$ is valid.

By the formula induction hypothesis, for every $\phi_x \in \{\phi_1, \ldots, \phi_g\}$, we have $M', s_0 \models \phi_x$ iff $M, s_0 \models \phi_x$.

Considering that $\langle r'\rangle\phi' \in CL(\langle r\rangle\phi') \subseteq CL(\gamma_1(\Phi))$, by the path induction hypothesis, $M', s_1 \models \langle r'\rangle\phi'$ and $(s_1, \ldots, s_q) \in Paths_{M'}(r')$ implies $M, s_1 \models \langle r'\rangle\phi'$.

Since $(s_0, s_1) \in \mathcal{R}'_\pi$, by construction of M', we have that either $(s_0, s_1) \in \mathcal{R}_\pi$, or $(s_0, s_1) \notin \mathcal{R}_\pi$ and $(s_1, s_0) \in \mathcal{R}_{\pi^c}$.

- If $(s_0, s_1) \in \mathcal{R}_\pi$, then we can immediately conclude that $M, s_0 \models \langle r\rangle\phi'$.
- If $(s_0, s_1) \notin \mathcal{R}_\pi$ and $(s_1, s_0) \in \mathcal{R}_{\pi^c}$, then considering that $\langle r'\rangle\phi'$ is equivalent to a formula $\psi \in CL(\gamma_1(\Phi))$, by $\gamma_2(\Phi)$ we have that

  $$M, s_1 \models \langle r'\rangle\phi' \to [\pi^c]\langle\pi\rangle\langle r'\rangle\phi'.$$

  Thus there exists a state $s_1' \in \mathcal{S}$ (different from s_1) such that $(s_0, s_1') \in \mathcal{R}_\pi$ and $M, s_1' \models \langle r'\rangle\phi'$. Hence, also in this case, we can conclude that $M, s_0 \models \langle r\rangle\phi'$.

■

The previous lemma has the following consequence.

LEMMA 7. *Let M be a connected model of $\gamma(\Phi)$ and M' its c-closure. Then M' is a model of $\gamma(\Phi)$ as well.*

Proof. Let $M = (\mathcal{S}, \{\mathcal{R}_\pi\}, \Pi)$ and $M' = (\mathcal{S}', \{\mathcal{R}'_\pi\}, \Pi')$. By Lemma 6, for all $s \in \mathcal{S} = \mathcal{S}'$ and all $\phi \in CL(\gamma_1(\Phi))$:

$$M, s \models \phi \text{ iff } M', s \models \phi.$$

Furthermore, by definition of M', $(s, s') \in \mathcal{R}'_\pi$ implies $(s', s) \in \mathcal{R}'_{\pi^c}$. Thus, for all $s \in \mathcal{S}'$ and all $\phi \in CL(\gamma_1(\Phi))$:

$$M', s \models \phi \to [P]\langle P^c\rangle\phi$$
$$M', s \models \phi \to [P^c]\langle P\rangle\phi.$$

Hence we can conclude that the thesis holds. ■

Below we formulate the main result of the present work.

THEOREM 8. *A CPDL formula Φ is satisfiable iff its PDL-counterpart $\gamma(\Phi)$ is satisfiable.*

Proof. $\Rightarrow$. Let $M^{CPDL} = (\mathcal{S}^{CPDL}, \{\mathcal{R}_P^{CPDL}\}, \Pi^{CPDL})$ be a model of Φ. We define a structure $M^{PDL} = (\mathcal{S}^{PDL}, \{\mathcal{R}_\pi^{PDL}\}, \Pi^{PDL})$ of $\gamma(\Phi)$ as follows:

- $\mathcal{S}^{PDL} = \mathcal{S}^{CPDL}$;
- $\mathcal{R}_P^{PDL} = \mathcal{R}_P^{CPDL}$ and $\mathcal{R}_{Pc}^{PDL} = \{(t,s) \mid (s,t) \in \mathcal{R}_P^{CPDL}\}$, for all atomic programs P occurring in Φ;
- $\Pi^{PDL} = \Pi^{CPDL}$.

It is easy to verify that $M^P DL$ is a model of $\gamma(\Phi)$.

$\Leftarrow$. Let $M^{PDL} = (\mathcal{S}^{PDL}, \{\mathcal{R}_\pi^{PDL}\}, \Pi^{PDL})$ be a connected model of $\gamma(\Phi)$ and $M^{PDL'} = (\mathcal{S}^{PDL'}, \{\mathcal{R}_\pi^{PDL'}\}, \Pi^{PDL'})$ its c-closure. By Lemma 7, M' is a model of $\gamma(\Phi)$ as well.

Observe that, by definition, M' is such that, for each atomic program π, $\mathcal{R}_{\pi c}^{PDL'} = (\mathcal{R}_\pi^{PDL'})^-$. We define a structure $M^{CPDL} = (\mathcal{S}^{CPDL}, \{\mathcal{R}_P^{CPDL}\}, \Pi^{CPDL})$ of $\gamma(\Phi)$ as follows:

- $\mathcal{S}^{CPDL} = \mathcal{S}^{PDL'}$;
- $\mathcal{R}_P^{CPDL} = \mathcal{R}_P^{PDL'}$ for all atomic programs P occurring in Φ;
- $\Pi^{CPDL} = \Pi^{PDL'}$.

It is easy to verify that M^{CPDL} is a model of Φ. ■

4. Conclusion

The logics *PDL* and *CPDL* share many characteristics, and many of the results for *PDL* extend to *CPDL* without difficulty. For instance the proofs of finite model property and decidability for *PDL* in (Fisher and Ladner, 1979) are easily extended to *CPDL*, as well as the proof of EXPTIME-completeness of satisfiability in (Pratt, 1979). However, while efficient – in practical cases – inference procedures have been successfully developed for *PDL*, extending them to *CPDL* has proved to be a difficult task, and to the best of our knowledge had been unsuccessful till now.

To be more precise, the inference procedures for *PDL* based on the enumeration of models such as those in (Fisher and Ladner, 1979; Pratt, 1979) can be easily modified to accommodate converse programs. But these procedures are better suited for proving theoretical results than for use in practice, since they are inherently exponential, not only in the worst-case.

In contrast, inference procedures for *PDL* such as those in (Pratt, 1978; Pratt, 1980), based on tableaux methods, which are much more efficient in practical cases, are difficult to modify to cope with converse programs.

The difficulty can be intuitively grasped by observing how these procedures attempt to build a model of a *PDL* formula in order to check its satisfiability. They start by introducing an initial state, and try to make it satisfy the formula. At first, reasoning is carried out locally, i.e. considering subformulae that involve state transitions, simply as atomic propositions. Next, when no more local reasoning is possible, the successor states, introduced by atomic programs, are generated, and the relevant formulae that these states ought to satisfy are propagated. For each successor state the two steps above are recursively repeated until certain termination conditions are met. The key point is that once the successors of a given state have been generated, there will be no more reasoning involving that state carried out. Thus, to check satisfiability of a *PDL* formula, a tableaux based procedure can be organized so as to work only "forward". This feature turns out to be essential in order to ensure efficient termination criteria.

The presence of converse programs does not allow us to extend the above approach in an obvious way. Indeed, reasoning on a state cannot be completely carried out without generating its successors, because, through converse programs, some successors may require further properties to be satisfied by the original state. Therefore, to check satisfiability of a *CPDL* formula, a procedure has to work both "forward" and "backward", thus losing efficiency, since at any point reasoning may involve all of the model built so far.

Is there any way out of this problem? One possible solution is by trying to single out a (hopefully small) set of additional formulae to be checked in every state, that in some sense anticipates the properties successor states may require at a later stage of the computation.

What the encoding of *CPDL* into *PDL* presented in this paper does is single out exactly a set of additional formulae as that mentioned above. Hence it can be the basis to develop better reasoning procedure for *CPDL*, on top of inference procedures for *PDL*. In fact, the encoding allows us to build a satisfiability procedure for *CPDL* by simply translating a *CPDL* formula to a *PDL* formula and then running a *PDL* satisfiability procedure on it. Therefore, considering that the encoding is polynomial, by employing an efficient satisfiability procedure for *PDL* we get an efficient satisfiability procedure for *CPDL*.

References

M. Ben-Ari, J. Y. Halpern, and A. Pnueli. Deterministic propositional dynamic logic: finite models, complexity, and completeness. *Journal of Computer and System Sciences*, 25:402–417, 1982.

P. Blackburn and E. Spaan. A modal perspective on computational complexity of attribute value grammar. *Journal of Logic, Language and Information*, 2:129–169, 1993.

G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1995.

G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the Twelth National Conference on Artificial Intelligence (AAAI-94)*, pages 205–212, 1994.

G. De Giacomo and M. Lenzerini. What's in an aggregate: foundation for description logics with tuples and set. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 801–807, 1995.

M. Fattorosi-Barnaba and F. De Caro. Graded modalities I. *Studia Logica*, 44:197–221, 1985.

N. J. Fisher and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

N. Friedman and J. Halpern. On the complexity of conditional logics. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, page 202–213, 1994.

G. Gargov and V. Goranko. Modal logic with names. *Journal of Philosophical Logic*, 22:607–636, 1993.

J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

D. Harel. Dynamic logic. In *Handbook of Philosophical Logic*, pages 497–603. D. Reidel Publishing Company, Oxford, 1984.

D. Kozen and J. Tiuryn. Logics of programs. In *Handbook of Theoretical Computer Science*, pages 790–840. Elsevier Science Publishers, 1990.

S. Passy and T. Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93:263–332, 1991.

V. R. Pratt. A practical decision method for propositional dynamic logic. In *Proceedings of the 10th Annual Symposium on Theory of Computing*, pages 326–337, 1978.

V. R. Pratt. Models of program logics. In *Proceedings of the 20th IEEE Symposium on the Foundations of Computer Science*, pages 115–122, 1979.

V. R. Pratt. A near-optimal method for reasoning about action. *Journal of Computer and System Sciences*, 20:231–255, 1980.

K. Schild. A correspondence theory for terminological logics: preliminary report. In *Proceedings of the Twelth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471, 1991.

C. Stirling. Modal and temporal logic. In *Handbook of Logic in Computer Science*, pages 477–563. Clarendon Press, Oxford, 1992.

R. S. Streett. Propositional dynamic logic of looping and converse is elementary decidable. *Information and Control*, 54:121–141, 1982.

J. Van Benthem and J. Bergstra. Logic of transition systems. *Journal of Logic, Language and Information*, 3(4):247–283, 1995.

J. Van Benthem, J. Van Eijck, and V. Stebletsova. Modal logic, transition systems and processes. *Journal of Logic and Computation*, 4(5):811–855, 1994.

W. van der Hoek and M. de Rijke. Counting objects. *Journal of Logic and Computation*, 5(3):325–345, 1995.