Giuseppe De Giacomo & Maurizio Lenzerini

Converse, Local Determinism, and Graded Nondeterminism in Propositional Dynamic Logics

Rapporto Tecnico n. 11-94
Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza",
via Salaria 113, I-00198 Roma, Italy
{degiacom,lenzerini}@assi.dis.uniroma1.it

Converse, Local Determinism, and Graded Nondeterminism in Propositional Dynamic Logics

Sommario

Lavoro recente sulla rappresentazione della conoscenza ha portato nuovo interesse nelle Logiche Dinamiche Proposizionali (PDL) evidenziando una stretta corrispondenza tra tali logiche e logiche per rappresentare conoscenza strutturata (Description Logics, Linguaggi Terminologici). Tuttavia, questo lavoro ha anche messo in evidenza la mancanza nelle PDL note di certi costrutti necessari per sfruttare pienamente la corrispondenza. Questi sono costrutti per vincolare localmente (rispetto ai singoli stati) l'esecuzione di un programma atomico o del suo inverso (l'esecuzione del programma atomico all'indietro) ad essere deterministica o ad avere solo un certo ammontare di nondeterminismo. In realtà noi pensiamo che le PDL possono avvantaggiarsi di questo tipo di costrutti per modellare molte proprietà interessanti di computazioni reali. In questo articolo, estendiamo Converse PDL, prima aggiungendo un costrutto per il determinismo locale di programmi semplici (programmi atomici e inversi di programmi atomici), e poi aggiungendo costrutti per il cosiddetto "Graded Nondeterminism" di programmi semplici. Questi ultimi sono costrutti che vincolano localmente il nondeterminismo di programmi semplici, limitando il minimo e il massimo numero di stati soddisfacenti una data proprietà che sono raggiungibili eseguendo un programma semplice partendo da un certo stato. Il risultato principale di questo articolo è che entrambe le logiche sono strettamente più espressive di "Converse Deterministic PDL" (Converse PDL dove i programmi sono interpretati come funzioni parziali) e sono entrambe ancora decidibili in tempo deterministico esponenziale.

Abstract

Recent work on Knowledge Representation has brought new interest to Propositional Dynamic Logics (PDL's) by pointing out a tight correspondence between these logics and logics for representing structured knowledge (Description Logics, Terminological Languages). Nevertheless, this work has also made apparent the lack in the known PDL's of certain constructs needed to fully exploit the correspondence. These are constructs to locally (wrt single states) constrain the running of an atomic program or its converse (the running of the atomic program backward) to be deterministic, or to have a specified amount of nondeterminism only. In fact, we believe that PDL's can take advantage of this kind of constructs to model many interesting properties of actual computations. In this paper, we extend Converse PDL, first by including a construct for local determinism of simple programs (either an atomic program or the converse of an atomic program), and then by including constructs for graded nondeterminism of simple programs. The latter locally constrain the nondeterminism of simple programs by limiting the minimum and the maximum number of states satisfying a specified property that are reachable by the running of a simple program from a certain state. The main result of the paper is that both the resulting logics are strictly more expressive than Converse Deterministic PDL (Converse PDL where all atomic programs are interpreted as partial functions), but still decidable in deterministic exponential time.

1 Introduction

Propositional Dynamic Logic (PDL) was introduced in [6] as a formalism to describe the properties of states reached by programs during their execution, and to model the evolution of the computation process (see [8, 7, 10] for surveys on PDL's, see also [14] for a somewhat different account). The language of PDL includes all formulae of propositional logic over a certain alphabet, plus the construct $< r > \phi$, where ϕ is a formula and r is a program, whose meaning is that it is possible to execute r and terminate in a state where ϕ is true. The program r can be either an atomic program, or a complex expression denoting sequential composition, non deterministic choice, iteration, or test.

Several variants of PDL have been proposed in order to enhance the expressive power of the logic. In this paper we concentrate our attention on an extension of PDL, called Converse PDL, obtained from the basic logic by adding the converse program operator, where the converse of a program P is the program whose running is obtained by running P backwards. Converse PDL is studied in [16], in the case where all atomic programs are assumed to be deterministic - i.e., representing partial functions over a set of states. This special case of the logic is denominated Converse Deterministic PDL. Note that assuming the atomic programs to be deterministic is not a limitation, but rather an improvement of the logic, because nondeterministic programs can be simulated by composing deterministic programs. In [16] a decision procedure for Converse Deterministic PDL is presented running in deterministic exponential time.

In this paper, we propose two extensions of Converse Deterministic PDL which allows for more sophisticated notions of determinism, and investigate their decidability and computational complexity. In particular, we consider the extensions of Converse PDL including the following constructs:

- A construct that allows us to impose the so-called *local determinism* of simple programs (either an atomic program or the converse of an atomic program) - i.e., the running of a simple program is deterministic from a certain state.
- Constructs that allow us to represent the so-called (local) *graded nondeterminism* of simple programs - i.e., to limit the minimum and the maximum number of states satisfying a specified formula that are reachable by the running of a simple program from a certain state.

Observe that by using constructs for local determinism (graded nondeterminism), one can easily impose global determinism (graded nondeterminism) as well. Note also that the construct for graded nondeterminism actually subsumes the one for local determinism: indeed, local determinism can be obtained by imposing the maximum number of states that are reachable by the running of a simple program from a given state, to be 1. Finally it is worth mentioning that our constructs for graded nondeterminism turn out to be strongly related to *graded modalities* in modal logic, which have been studied in [5, 4, 15].

The constructs introduced above can be used to model many interesting properties of actual computations. For example, suppose we want to check/impose some facts about a state s'

preceding the current state s, in a given computation. This may seem possible by "executing backward" the program, associated to computation from s' to s, and test the wanted properties on the resulting state. But "executing backward" this program could also lead to states which are not states of the original computation. In fact, the notion of "actual past" is not captured in Converse PDL. Though, if we can impose that each atomic program is backward deterministic (converse of each atomic programs is deterministic), then the notion of "actual past" become easily modelable. Similarly we may want to constrain the possible states that are reachable from the current state by executing an atomic instruction (running an atomic program) to be no more (no less) than, say, three. This can be easily achieved, having at hand constructs for graded nondeterminism.

Furthermore, the availability of the proposed constructs is crucial to make PDL's exploitable as the basic reasoning paradigm of certain kind of Knowledge Representation Systems. Let us explain this point in some details. Several recent papers (starting from [13]) point out that there is a strong correspondence between Propositional Dynamic Logic and its variants, and a family of Logic-based Knowledge Representation Languages, called Terminological Languages (Description Logics). These languages allow the representation of a real world in terms of objects, classes (unary predicates whose instances are objects) and relations (binary predicates whose instances are pairs of objects), and are characterized by several constructs for establishing the properties of classes and relations. The correspondence is based on a mapping between the models of a knowledge base expressed in a Terminological Language, and the models of a particular PDL formula, where classes correspond to propositional letters, relations correspond to atomic programs, instances of classes correspond to states, and instances of relations correspond to state transitions. Among the various constructs that have been considered in Terminological Languages, the one for denoting the inverse of a relation, and those for expressing functional and number restrictions on the connection between instances of classes and relations have special importance for achieving the desired expressive power. However, despite the relevance of these constructs, no general technique is known for reasoning about knowledge expressed using them. Now, it is easy to see that the inverse of relations corresponds to the converse of a program, the functional restriction on relations corresponds to our notion of local determinism, and number restrictions correspond to graded nondeterminism. It follows that the reasoning techniques developed for the extensions of Converse PDL proposed in this paper, directly provide suitable methods for reasoning in very expressive Terminological Languages. Indeed this was the original motivation that has led us to look into this logics ([2]).

The main result of this paper is that, both by adding local determinism and by adding graded nondeterminism to Converse PDL, we obtain very powerful propositional dynamic logics (they both strictly subsume Converse Deterministic PDL), that are still decidable in deterministic exponential time (as the basic PDL). The method we adopt for proving the result is to show that from any formula expressed in the enhanced logics we can obtain in polynomial time a Converse PDL formula that is satisfiable if and only if the original formula is so, thus proving that the well known decision procedures for Converse PDL (e.g. [11, 12, 9, 16]) can be used as "reasoning engine" for the enhanced logics.

The rest of the paper is organized as follows. In Section 2, we recall the basic notions regarding Converse PDL. In Section 3, we present the results concerning the extension of Converse PDL with local determinism, whereas in Section 4, we deal with the extension of Converse PDL with graded nondeterminism, finally some concluding remarks end the paper.

2 Preliminaries

We base our work on the well-known Converse PDL, called $\mathcal{L}$ in the following, whose basic characteristics are recalled in this section.

The formation rules of $\mathcal{L}$ are specified by the following abstract syntax

$$\begin{array}{lcl} \phi & \longrightarrow & A \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \neg\phi \mid < r > \phi \mid [r]\phi \\ r & \longrightarrow & P \mid r_1 \cup r_2 \mid r_1 ; r_2 \mid r^* \mid \phi ? \mid r^- \end{array}$$

where A denotes a propositional letter, ϕ (possibly with subscript) denotes a formula, P denotes an atomic program and r (possibly with subscript) denotes a program. For notational convenience, we define the additional symbols, $\Rightarrow$ as usual, $\top$ as $A \vee \neg A$, and $\bot$ as $\neg\top$. We use the term *simple program* to refer to either an atomic program or the converse of an atomic program, and we denote it by a (possibly with subscript).

The semantics of $\mathcal{L}$ is based on the notion of structure, which is defined as a triple $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$, where $\mathcal{S}$ denotes a set of states, $\{R_P\}$ is a family of binary relations over $\mathcal{S}$, such that each atomic program P is given a meaning through 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. Given M, the family $\{\mathcal{R}_P\}$ can be extended in the obvious way so as to include, for every program r, the corresponding relation $\mathcal{R}_r$ (for example, $\mathcal{R}_{r_1;r_2}$ is the composition of $\mathcal{R}_{r_1}$ and $\mathcal{R}_{r_2}$). For this reason, we often denote a structure by $(\mathcal{S}, \{\mathcal{R}_r\}, \Pi)$, where $\{\mathcal{R}_r\}$ includes a binary relations for every program (atomic or non-atomic). A structure M is called a model of a formula ϕ if there exists a state s in M such that $M, s \models \phi$. A formula ϕ is satisfiable if there exists a model of ϕ, unsatisfiable otherwise.

The *Fisher-Ladner closure* of a $\mathcal{L}$-formula Φ, denoted $CL(\Phi)$, is the smallest set such that $\Phi \in CL(\Phi)$ and such that (we assume, without loss of generality, $\vee, [\cdot]$ to be expressed by means of $\neg, \wedge, < \cdot >$, and the converse operator to be applied to atomic programs only):

$$\begin{array}{lll} \phi_1 \wedge \phi_2 \in CL(\Phi) & \text{implies} & \phi_1, \phi_2 \in CL(\Phi), \\ \neg\phi \in CL(\Phi) & \text{implies} & \phi \in CL(\Phi), \\ < r > \phi \in CL(\Phi) & \text{implies} & \phi \in CL(\Phi), \\ < r_1 ; r_2 > \phi \in CL(\Phi) & \text{implies} & < r_1 >< r_2 > \phi \in CL(\Phi), \\ < r_1 \cup r_2 > \phi \in CL(\Phi) & \text{implies} & < r_1 > \phi, < r_2 > \phi \in CL(\Phi), \\ < r^* > \phi \in CL(\Phi) & \text{implies} & < r >< r^* > \phi \in CL(\Phi), \\ < \phi' > \phi \in CL(\Phi) & \text{implies} & \phi' \in CL(\Phi). \end{array}$$

A *path* in a structure M is a sequence $(s_0, \ldots, s_q)$ of states of M, such that for each (s_{i-1}, s_i), where $i = 1, \ldots, q$, there exists a simple program $a = P \mid P^-$ in Φ such that $(s_{i-1}, s_i) \in \mathcal{R}_a$. The *length* of $(s_0, \ldots, s_q)$ is q.

We inductively define the set of paths $Paths(r)$ of a program r in a structure M, as follows (again we assume, without loss of generality, that in r all occurrences of the converse operator are moved all way in):

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

We say that a path (s_0) in M *satisfies* a formula ϕ which is not of the form $< r > \phi'$ if $M, s_0 \models \phi$. We say that a path $(s_0, \ldots, s_q)$ in M *satisfies* a formula ϕ of the form $< r_1 > \cdots < r_l > \phi'$, where ϕ' is not of the form $< r' > \phi''$, if $(s_0, \ldots, s_q) \subseteq Paths(r_1; \cdots; r_l)$ and $M, s_q \models \phi'$.

Finally, if a denotes the atomic program P (resp. the converse of an atomic program P^-), then we write a^- to denote P^- (resp. P).

3 Local determinism

The first extension of $\mathcal{L}$, called $\mathcal{L}_{ld}$, is obtained from $\mathcal{L}$ by adding the construct $(\leq\ 1\ a)$, where a is a simple program $(a = P \mid P^-)$. The new construct is interpreted as follows: given a structure $M = (\mathcal{S}, \{\mathcal{R}_r\}, \Pi)$ and a state $s \in \mathcal{S}$,

$$M, s \models (\leq\ 1\ a) \quad \text{iff} \quad \text{there exists } \textit{at most one } t \text{ such that } (s, t) \in \mathcal{R}_a.$$

Observe that the $(\leq\ 1\ a)$ construct allows the notion of local determinism for both atomic programs and the converse of atomic programs to be represented in the logic. With this construct, we can denote states from which the running of an atomic program, or the converse of an atomic program, is deterministic - i.e., it leads to at most one state. It is easy to see that this possibility allows one to impose the so-called global determinism too - i.e., that a given atomic program, or the converse of an atomic program, is (globally) deterministic. Therefore, $\mathcal{L}_{ld}$ subsumes the logic studied in [16], called Converse Deterministic PDL, where atomic programs, *not their converse*, are (globally) deterministic.

The decidability and the complexity of satisfiability in $\mathcal{L}_{ld}$ are to be established yet. We establish them below by showing an encoding of $\mathcal{L}_{ld}$-formulae in $\mathcal{L}$. More precisely we show that, for any $\mathcal{L}_{ld}$-formula Φ, there is a $\mathcal{L}$-formula, denoted $\gamma(\Phi)$, whose size is polynomial with respect to the size of Φ, and such that Φ is satisfiable iff $\gamma(\Phi)$ is satisfiable. Since satisfiability in $\mathcal{L}$ is EXPTIME-complete, this ensures us that satisfiability in $\mathcal{L}_{ld}$ is EXPTIME-complete too. In what follows, we assume, without loss of generality, that Φ is in negation normal form (i.e., negation is pushed inside as much as possible). We define the *$\mathcal{L}$-counterpart* $\gamma(\Phi)$ of a $\mathcal{L}_{ld}$-formula 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 $(\leq 1\ a)$ with a new propositional letter $A_{(\leq 1\ a)}$, and each $\neg(\leq 1\ a)$ with $(< a > H_{(\leq 1\ a)}) \wedge (< a > \neg H_{(\leq 1\ a)})$, where $H_{(\leq 1\ a)}$ is again a new propositional letter.
- $\gamma_2(\Phi) = [(P_1 \cup \cdots \cup P_m \cup P_1^- \cdots \cup P_m^-)^*]\gamma_2^1 \wedge \cdots \wedge \gamma_2^g$, where $P_1, \ldots, P_m$ are all atomic roles appearing in Φ, and with one conjunct γ_2^i of the form

  $$((A_{(\leq 1\ a)} \wedge < a > \phi) \Rightarrow [a]\phi)$$

  for every $A_{(\leq 1\ a)}$ occurring in $\gamma_1(\Phi)$ and every $\phi \in CL(\gamma_1(\Phi))$.

Intuitively $\gamma_2(\Phi)$ constrains the models M of $\gamma(\Phi)$ so that: for every state s of M, if $A_{(\leq 1\ a)}$ holds in s, and there is an a-transition from s to t_1 and an a-transition from s to t_2, then t_1 and t_2 are equivalent wrt the formulae in $CL(\gamma_1(\Phi))$. We show that this allow us to actually collapse t_1 and t_2 into a single state.

To prove that a $\mathcal{L}_{ld}$-formula is satisfiable iff its $\mathcal{L}$-counterpart is, we proceed as follows. Given a model $M = (\mathcal{S}, \{R_r\}, \Pi)$ of $\gamma(\Phi)$, we build a tree-like structure $M^t = (\mathcal{S}^t, \{R_r^t\}, \Pi^t)$ such that $M^t, root \models \gamma(\Phi)$ ($root \in S^t$ is the root of the tree-structure), and the local determinism requirements are satisfied. From such M^t, a model $M^t_{\mathcal{F}}$ of Φ can easily be derived. In order to construct M^t we make use of the following notion: For each state s in M, we call by $ES(s)$ the smallest set of states in M such that

- $s \in ES(s)$, and
- if $s' \in ES(s)$, then for every s'' such that $(s', s'') \in \mathcal{R}_{a;A_{(\leq 1\ a^-)}?;a^-}$, $ES(s'') \subseteq ES(s)$.

The set $ES(s)$ is the set of states of M that are to be collapsed into a single state of M^t. Note that, by $\gamma_2(\Phi)$, all the states in $ES(s)$ satisfy the same formulae in $CL(\gamma_1(\Phi))$. The construction of M^t is done in three stages.

Stage 1. Let $< a_1 > \psi_1, \ldots, < a_h > \psi_h$ be all the formulas of the form $< a > \phi'$ included in $CL(\Phi)$.[1] We consider an infinite h-ary tree $\mathcal{T}$ whose root is $root$ and such that every node x has h children $child_i(x)$, one for each formula $< a_i > \psi_i$. We write $father(x)$ to denote the father of a node x in $\mathcal{T}$. We define two partial mappings m and l: m maps nodes of $\mathcal{T}$ to states of M, and l is used to label the arcs of $\mathcal{T}$ by either atomic programs, converse of atomic programs, or a special symbol 'undefined'. For the definition of m and l, we proceed level by level. Let $s \in \mathcal{S}$ be any state such that $M, s \models \gamma(\Phi)$. We put $m(root) = s$, and for all arcs $(root, child_i(root))$ corresponding to a formula $< a_i > \psi_i$ such that $M, s \models < a_i > \psi_i$ we put $l((root, child_i(root))) = a_i$. Suppose we have defined m and l up to level k, let x be a node at level $k+1$, and let $l((father(x), x)) = a_j$. Then $M, m(father(x)) \models < a_j > \psi_j$, and therefore, there exists a path $(s_o, s_1, \ldots, s_q)$, with $s_o = m(father(x))$ satisfying $< a_j > \psi_j$. Among the

[1]Notice that the formulas ψ_i may be of the form $< r > \phi$, and that $\psi_i \in CL(\Phi)$.

states in $ES(s_1)$ we choose a state t such that there exists a *minimal* path (i.e., a path with minimal length) from t satisfying ψ_j. We put $m(x) = t$ and for every $< a_i > \psi_i \in CL(\Phi)$ such that $M, t \models < a_i > \psi_i$ we put $l((x, child_i(x))) = a_i$.

Stage 2. We change the labeling l, proceeding again level by level. If $M, m(root) \models A_{(\leq 1\ a)}$, then for each arc $(root, child_i(root))$ labeled a, except for one randomly chosen, we put $l((root, child_i(root)) =$ 'undefined'. Assume we have modified l up to level k, and let x be a node at level $k+1$. Suppose $M, m(x) \models A_{(\leq 1\ a)}$. Then if $l((father(x), x)) = a^-$, for each arc $(x, child_i(x))$ labeled a, we put $l((x, child_i(x)) =$ 'undefined', otherwise (i.e. $l((father(x), x)) \neq a^-$) we put $l((x, child_i(x)) =$ 'undefined' for every arc $(x, child_i(x))$ labeled a, except for one randomly chosen.

Stage 3. For each P, let $\mathcal{R}'_P = \{(x, y) \in \mathcal{T} \mid l((x, y)) = P$ or $l((y, x)) = P^-\}$. We define the structure $M^t = (\mathcal{S}^t, \{\mathcal{R}^t_r\}, \Pi^t)$ as follows: $\mathcal{S}^t = \{x \in \mathcal{T} \mid (root, x) \in (\bigcup_P (\mathcal{R}'_P \cup \mathcal{R}'^-_P))^*\}$, $\mathcal{R}^t_P = \mathcal{R}'_P \cap (\mathcal{S}^t \times S^t)$, and $\Pi^t(x) = \Pi(m(x))$ $(\forall x. x \in \mathcal{S}^t)$. From $\{\mathcal{R}^t_P\}$ we get all $\{\mathcal{R}^t_r\}$ as usual.

The basic property of M^t is stated in the following lemma.

Lemma 1 *Let Φ be a $\mathcal{L}_{ld}$-formula, and let M be a model of $\gamma(\Phi)$. Then, for every formula $\phi \in CL(\gamma_1(\Phi))$ and every $x \in \mathcal{S}^t$, $M^t, x \models \phi$ iff $M, m(x) \models \phi$.*

Proof We prove the lemma by induction on the formation of ϕ. We assume, without loss of generality, $\vee, [\cdot]$ to be expressed by means of $\neg, \wedge, < \cdot >$.

1. $\phi = A$. $M, m(x) \models A$ iff $A \in \Pi(m(x))$ iff $A \in \Pi^t(x)$ (by construction of M^t) iff $M^t, x \models A$.

2. $\phi = \phi_1 \wedge \phi_2 \mid \neg\phi_1$. $M, m(x) \models \phi_1 \wedge \phi_2$ iff $M, m(x) \models \phi_1 \wedge M, m(x) \models \phi_2$ iff $M^t, x \models \phi_1 \wedge M^t, x \models \phi_2$ (by induction on the structure of the formula) iff $M^t, x \models \phi_1 \wedge \phi_2$. Similarly, $M, m(x) \models \neg\phi_1$ iff $M, m(x) \not\models \phi_1$ iff $M^t, x \not\models \phi_1$ (by induction on the structure of the formula) iff $M^t, x \models \neg\phi_1$.

3. $< r_1; r_2 > \phi_1, < r_1 \cup r_2 > \phi_1, < r^* > \phi_1, < \phi_2? > \phi_1$, and converse of non-atomic programs.

   Recall that the following equivalences hold:

$$\begin{array}{lll} < r_1; r_2 > \phi_1 & \text{iff} & < r_1 >< r_2 > \phi_1 \\ < r_1 \cup r_2 > \phi_1 & \text{iff} & < r_1 > \phi_1 \vee < r_2 > \phi_1 \\ < r^* > \phi_1 & \text{iff} & \phi_1 \vee < r >< r^* > \phi_1 \\ < \phi_2? > \phi_1 & \text{iff} & \phi_1 \wedge \phi_2. \end{array}$$

   Moreover we may assume the converse operator applied only to atomic programs since we have:

$$\begin{array}{lll} (r_1; r_2)^- & = & r_1^-; r_2^- \\ (r_1 \cup r_2)^- & = & r_1^- \cup r_2^- \\ (r_1^*)^- & = & (r_1^-)^* \\ (\phi_2?)^- & = & (\phi_2?). \end{array}$$

   Hence all these cases are reducible to 1,2,4.

4. The only case that remains to be considered is $\phi =< a_i > \psi_i$, $i = 1, \ldots, h$, $a_i = P \mid P^-$.

   - $\Rightarrow$ Assume $M, m(x) \models< a_i > \psi_i$. Then there exists a path, $(s_0, s_1, \ldots, s_q)$ with $s_0 = m(x)$, satisfying $< a_i > \psi_i$ such that $m(child_i(x)) \in ES(s_1)$ (by construction of M^t). We prove $M^t, x \models< a_i > \psi_i$, by induction on the length of this path. Notice that the following facts hold: $(s_0, s_1) \in \mathcal{R}_{a_i}$; $(s_1, \ldots s_q)$ is a path satisfying ψ_i; and $(x, child_i(x)) \in \mathcal{R}^t_{a_i}$.

     Base case: $q = 1$, i.e. either ψ_i is not of the form $< r > \phi'$, or ψ_i is of the form $< r_1 > \ldots < r_l > \phi'$, where all r_k are made up just of tests, and ϕ' is not of the form $< r > \phi''$. Then $M, s_1 \models \psi_i$ and $m(child_i(x)) \in ES(s_1)$ imply $M^t, child_i(x) \models \psi_i$ (by induction on the structure of the formula), and so $M^t, x \models< a_i > \psi_i$.

     Inductive case: $q > 1$, i.e. $\psi_i =< r > \phi'$.[2] Then from $m(child_i(x))$ there exists a *minimal* path $(s'_1, \ldots, s'_{q'})$ satisfying ψ_i, which is shorter or of the same length as $(s_1, \ldots, s_q)$. Now, $\psi_i =< r > \phi'$ implies that there exists a formula of the form $< \phi_1?; \phi_2?; \cdots; \phi_m?; a_j > \psi_j \in CL(\gamma_1(\Phi))$ such that $< \phi_1?; \phi_2?; \cdots \phi_m?; a_j > \psi_j \Rightarrow< r > \phi'$ (see 3) and $(s'_1, \ldots, s'_{q'})$ satisfies $< a_j > \psi_j$. By induction on q it holds that $M^t, child_i(x) \models< a_j > \psi_j$, while by induction on the structure of the formulae, $M^t, child_i(x) \models \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_m$.[3] Therefore, $M^t, child_i(x) \models< \phi_1?; \phi_2?; \cdots; \phi_m?; a_j > \psi_i$, which implies $M^t, child_i(x) \models \psi_i$, and thus $M^t, x \models< a_i > \psi_i$.

   - $\Leftarrow$ Assume that $M^t, x \models< a_i > \psi_i$ and let $(x_0, x_1, \ldots, x_q)$, with $x_0 = x$ and $x_1 = child_i(x)$, be a path satisfying $< a_i > \psi_i$. Then $M^t, x_1 \models \psi_i$ and by construction of M^t there exists a state s such that $s \in ES(m(x_1))$ and $(m(x), s) \in \mathcal{R}_{a_i}$. Again we prove that $M, m(x) \models< a_i > \psi_i$ by induction on q.

     Base case: $q = 1$, i.e. either ψ_i is not of the form $< r > \phi'$, or ψ_i is of the form $< r_1 > \ldots < r_l > \phi'$, where all r_k are made up just of tests, and ϕ' is not of the form $< r > \phi''$. Then by induction on the structure of the formula, $M, s \models \psi_i$ and so $M, m(x) \models< a_i > \psi_i$.

     Inductive case: $q > 1$, i.e. $\psi_i =< r > \phi'$. Then $M^t, x_1 \models \psi_i$ and $(x_1, \ldots, x_q)$, which satisfies ψ_i, is obviously shorter than $(x_0, x_1, \ldots, x_q)$. Now, $\psi_i =< r > \phi'$ implies that there exists a formula of the form $< \phi_1?; \phi_2?; \cdots; \phi_m?; a_j > \psi_j \in CL(\gamma_1(\Phi))$ such that $< \phi_1?; \phi_2?; \cdots \phi_m?; a_j > \psi_j \Rightarrow< r > \phi'$ (see 3), and $(x_1, \ldots, x_q)$ satisfies $< a_j > \psi_j$. By induction on q it holds that $M, m(x_1) \models< a_j > \psi_j$, while by induction on the structure of the formulae, $M, m(x_1) \models \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_m$. Therefore, $M, m(x_1) \models< \phi_1?; \phi_2?; \cdots; \phi_m?; a_j > \psi_i$, which implies $M, m(x_1) \models \psi_i$. Hence, since $s \in ES(m(x_1))$, we have $M, s \models \psi_i$, and so $M, m(x) \models< a_i > \psi_i$.

□

[2]Note that ϕ' may be of the form $< r' > \phi''$ itself.

[3]Note that $\phi_1, \ldots \phi_m \in CL(\gamma_1(\Phi))$.

Note that M^t is a model of $\gamma(\Phi)$, since by Lemma 1 $M^t, root \models \gamma_1(\Phi)$, and on the other hand M^t trivially satisfies $\gamma_2(\Phi)$, because whenever $M^t, x \models A_{(\leq 1a)}$, there exists just one $child_i(x)$ such that $(x, child_i(x)) \in \mathcal{R}^t_a$.

Once we have obtained M^t from a model M of $\gamma(\Phi)$, we define $M^t_{\mathcal{F}} = (\mathcal{S}^t, \{\mathcal{R}^t_r\}, \Pi^t_{\mathcal{F}})$, where for each $x \in \mathcal{S}$, $\Pi^t_{\mathcal{F}}(x) = \Pi^t(x) - \{A_{(\leq 1\ a)}, H_{(\leq 1\ a)} \mid A_{(\leq 1\ a)}, H_{(\leq 1\ a)} \in \Pi^t(x)\}$. The new structure $M^t_{\mathcal{F}}$ has the following property.

Lemma 2 *Let Φ be a $\mathcal{L}_{ld}$-formula, and let $M^t, M^t_{\mathcal{F}}$ be obtained from a model M of $\gamma(\Phi)$ as specified above. Then $M^t, root \models \gamma_1(\Phi)$ implies $M^t_{\mathcal{F}}, root \models \Phi$.*

Proof Notice that, if $M^t, s \models A_{(\leq 1\ a)}$ then, by construction of M^t, there exists at most one w such that $(s, w) \in \mathcal{R}^t_a$, implying that $M^t_{\mathcal{F}}, s \models (\leq 1\ a)$. On the other hand, if $M^t, s \models (< a > H_{(\leq 1\ a)}) \wedge (< a > \neg H_{(\leq 1\ a)})$, then there are at least two states t_1, t_2 such that $(s, t_1) \in \mathcal{R}^t_a$ and $(s, t_2) \in \mathcal{R}^t_a$, implying that $M^t_{\mathcal{F}}, s \models \neg(\leq 1\ a)$ The proof is easily completed by induction on the structure of Φ. $\Box$

The main theorem of this section can now be stated as follows.

Theorem 3 *A $\mathcal{L}_{ld}$-formula Φ is satisfiable iff its $\mathcal{L}$-counterpart $\gamma(\Phi)$ is satisfiable.*

Proof $\Rightarrow$ Notice that every model $M = (\mathcal{S}, \{\mathcal{R}_r\}, \Pi)$ of Φ can be extended to a structure $M' = (\mathcal{S}, \{\mathcal{R}_r\}, \Pi')$, where, for each $s \in \mathcal{S}$ such that $M, s \models (\leq\ 1\ a)$, we put $\Pi'(s) = \Pi(s) \cup \{A_{(\leq 1\ a)}\}$, and, for each $s \in \mathcal{S}$ such that $M, s \models \neg(\leq\ 1\ a)$, we choose one t such that $(s, t) \in \mathcal{R}_a$ and we put $\Pi'(t) = \Pi(t) \cup \{H_{(\leq 1\ a)}\}$. M' satisfies $\gamma_1(\Phi)$ and moreover it (trivially) satisfies $\gamma_2(\Phi)$. Therefore $M, s \models \Phi$ implies $M', s \models \gamma(\Phi)$.

$\Leftarrow$ Suppose that there exists a model M of $\gamma(\Phi)$. Then, by applying the above described construction we can build a new structure M^t satisfying the local determinism requirements such that, by the Lemma 1, $M^t, root \models \gamma_1(\Phi)$, and thus by Lemma 2, $M^t_{\mathcal{F}}, root \models \Phi$. $\Box$

Corollary 4 *Satisfiability in $\mathcal{L}_{ld}$ is an EXPTIME-complete problem.*

Proof The satisfiability problem for $\mathcal{L}$ is EXPTIME-complete, and the size of the $\mathcal{L}$-counterpart $\gamma(\Phi)$ of a $\mathcal{L}_{ld}$-formula Φ is polynomially related to the size of Φ. $\Box$

The fact that $\mathcal{L}_{ld}$-formulae can be encoded in $\mathcal{L}$, calls for some comments. Notice that $\mathcal{L}$-formulae have always a finite model M (finite model property) while $\mathcal{L}_{ld}$-formalae don't - e.g. the $\mathcal{L}_{ld}$ formula $A \wedge [(P^-)^*]((\leq 1\ P^-) \wedge < P^- > \neg A$ does not have any finite model[4]. Indeed, M^t, and thus $M^t_{\mathcal{F}}$, build from a finite model M are not finite in general.

It is also interesting to observe that, since $\mathcal{L}_{ld}$ subsumes Converse Deterministic PDL, also formulae of that logic can be encoded in $\mathcal{L}$. This fact gives us procedures to decide satisfiability

[4]This formula is a variant of the Converse Deterministic PDL formula $A \wedge [(P^-)^*] < P^- > \neg A$ (see for example [16]).

of Converse Deterministic PDL formulae that do not rely on techniques based on automata on infinite structures as those in [16].

Finally, the construction above can be easily modified/restricted to encode Deterministic PDL formulae in PDL. In fact, the construction used in [1] to study satisfiability of Deterministic PDL, is similar in the spirit, though not in the development, to such a restricted version of the our construction.

4 Graded nondeterminism

The second extension of $\mathcal{L}$, called $\mathcal{L}_{gn}$, is obtained from $\mathcal{L}$ by adding constructs for *graded nondeterminism* of the form $(\leq n\ a.\phi)$, $(\geq n\ a.\phi)$ where $n \geq 1$ and, as usual, a is a simple program ($a = P \mid P^-$). The semantics of these constructs is as follows: given a structure M and a state $s \in \mathcal{S}$,

$M, s \models (\leq n\ a.\phi)$ iff there are *at most* n states t such that $(s,t) \in \mathcal{R}_a$ and $M, t \models \phi$

$M, s \models (\geq n\ a.\phi)$ iff there are *at least* n states t such that $(s,t) \in \mathcal{R}_a$ and $M, t \models \phi$.

Note that $(\geq n\ a.\phi)$ is equivalent to $\neg(\leq n-1\ \ a.\phi)$.

Intuitively, if s is a state in which $(\leq n\ a.\phi)$ holds, then there may be at most n a-successors of s in which ϕ holds. In particular $(\leq 1\ a.\top)$ can be used for representing local determinism as defined in the previous section.

Below we show that $\mathcal{L}_{gn}$-formulae can be encoded in $\mathcal{L}_{ld}$. Though, to gain some intuition on this result for $\mathcal{L}_{gn}$, we first discuss the issues involved, in the context of the basic PDL.

On this simpler logic, we can work with *deterministic* structures - i.e., all atomic programs are (globally) deterministic - instead of non-deterministic ones. In fact it is well-known (see [10]) that if we replace each atomic program P in a formula Φ by $F_P;(F'_P)^*$ where F_P and F'_P are new atomic programs that are (globally) deterministic, then, calling the resulting formula Φ', we have that Φ is satisfiable iff Φ' is so.[5] We briefly sketch the reasoning behind the proof of this statement. Let M be a model of Φ. We may "unfold" M so to get a new model M^T having a the form of a *tree*. Now there is a one-to-one transformation form tree models M^T of Φ to a *binary-tree* models M^B of Φ'. Indeed, given a state x of M^T having as P-successor $z_1, \ldots, z_l$, we put $(x, z_1) \in \mathcal{R}^B_{F_P}$, and $(z_i, z_{i+1}) \in \mathcal{R}^B_{F'_P}$, for $i = 1, \ldots, l-1$. In this way we have $(x, z_i) \in \mathcal{R}^T_P$ iff $(x, z_i) \in \mathcal{R}^B_{F_P;(F'_P)^*}$. We remark that the fact that M^T is a tree is required because in the deterministic structure M^B we need to be able to recover the "original" P-predecessor x of a state z_i (that is we need $(F_P;(F'_P)^*)^-$ to be *deterministic*), otherwise the mapping from the nondeterministic structure M^T and the deterministic structure M^B is not one-to-one anymore.

In deterministic structures it is easy to express graded nondeterminism by means of constraints on the chain of $F_P;(F'_P)^*$-successors of a state. For example,

[5]Note that while it is necessary to introduce one F_P for each P, we could introduce just one F_U for all P_i, instead of all F'_{P_i}. Here we have preferred to be slightly redundant, for sake of clarity.

- $(\leq 3\ P.\phi)$ can be expressed by $[F_P; (F'_P)^*; \phi?; (F'_P)^*; \phi?; (F'_P)^*; \phi?; (F'_P)^*]\neg\phi$ that is equivalent to $[F_P; (F'_P)^*](\phi \Rightarrow [(F'_P)^*](\phi \Rightarrow [(F'_P)^*](\phi \Rightarrow [(F'_P)^*]\neg\phi)))$, that can be read as "everywhere along the chain $F_P; (F'_P)^*$ there are at most three states in which ϕ holds", that corresponds exactly the intended meaning.

- $(\geq 3\ P.\phi)$ can be expressed by $< F_P; (F'_P)^*; \phi?; (F'_P)^*; \phi?; (F'_P)^* > \phi$ that is equivalent to $< F_P; (F'_P)^* > (\phi \wedge < (F'_P)^* > (\phi \wedge < (F'_P)^* > \phi))$, that can be read as "somewhere along the chain $F_P; (F'_P)^*$ there are at least three states in which ϕ holds", that again corresponds exactly the intended meaning.

Getting back to $\mathcal{L}_{gn}$, the presence of converse programs makes its structures no longer reducible to tree structures as above[6], making the technique sketched above inapplicable. Nonetheless we are able to obtain essentially the same results, by developing a more involved reduction.

Indeed, we are going to prove that for any $\mathcal{L}_{gn}$ formula Φ there exists a $\mathcal{L}_{ld}$ formula Φ', whose size is polynomial wrt the size of Φ, that is satisfiable iff Φ' is so. Since we have proved in the last section that satisfiability in $\mathcal{L}_{ld}$ is EXPTIME-complete, this ensures us that satisfiability in $\mathcal{L}_{gn}$ is EXPTIME-complete too. The reduction is performed in two phases.

Phase 1

Let Φ be a $\mathcal{L}_{gn}$ formula, we define $\mu_1(\Phi)$ as follows:

1. In Φ, we replace, every atomic program P_i, $i = 1 \ldots m$, by the complex program $f_1^-; A_{P_i}?; f_2$, where f_1, f_2 are new atomic programs (the only one present after the transformation) and A_{P_i} is a new atomic proposition. Let us call the resulting formula $\mu_0(\Phi)$.

2. We put in conjunction with $\mu_0(\Phi)$, the formula $\Theta_1 = [(f_1 \cup f_2 \cup f_1^- \cup f_2^-)^*]((\leq\ 1\ f_1) \wedge (\leq\ 1\ f_2))$.

   This imposes the global determinism of both f_1 and f_2 - i.e., in each model $M = (\mathcal{S}, \{\mathcal{R}_{f_1}, \mathcal{R}_{f_2}\}, \Pi)$, $\mathcal{R}_{f_1}$ and $\mathcal{R}_{f_2}$ are partial functions.

Note that formulae of the form $(\cdot\ n\ P.\varphi)/(\cdot\ n\ P^-.\varphi)$ become of the form $(\cdot\ n\ f_1^-; A_P?; f_2.\phi)/$ $(\cdot\ n\ f_2^-; A_P?; f_1.\phi)$, thus specifying graded nondeterminism of complex programs. Though, observe that by Θ_1, the programs f_1/f_2 are deterministic. In fact the following equivalences hold:

$$\begin{array}{lcl}
(\leq n\ f_1^-; A_P?; f_2.\phi) & \equiv & (\leq n\ f_1^-.(< A_P?; f_2 > \phi)), \\
(\geq n\ f_1^-; A_P?; f_2.\phi) & \equiv & (\geq n\ f_1^-.(< A_P?; f_2 > \phi)), \\
(\leq n\ (f_1^-; A_P?; f_2)^-.\phi) & \equiv & (\leq n\ f_2^-.(< A_P?; f_1 > \phi)), \\
(\geq n\ (f_1^-; A_P?; f_2)^-.\phi) & \equiv & (\geq n\ f_2^-.(< A_P?; f_1 > \phi)).
\end{array}$$

Lemma 5 *Φ is satisfiable iff $\mu_1(\Phi)$ is satisfiable.*

Proof $\Rightarrow$ Let $M = \{\mathcal{S}, \{\mathcal{R}_P\}, \Pi\}$ be a model of Φ. We can define a model of $M' = \{\mathcal{S}', \{\mathcal{R}'_{f_1}, \mathcal{R}'_{f_2}\}, \Pi'\}$ of $\mu_1(\Phi)$ as follows:

[6] Indeed the presence of converse programs makes the structures reducible to "two-ways" tree structures, as opposite to "one-way" tree structures as needed here.

- $\mathcal{S}' = \mathcal{S} \cup \{z_{xy} \mid (x,y) \in \mathcal{R}_{P_i} \text{ for some } P_i\}$,
- $\mathcal{R}'_{f_1} = \{(z_{xy}, x) \mid (x,y) \in \mathcal{R}_{P_i}\}$, $\mathcal{R}'_{f_2} = \{(z_{xy}, y) \mid (x,y) \in \mathcal{R}_{P_i}\}$,
- $\Pi'(x) = \begin{cases} \Pi(t) & t \in \mathcal{S} \\ \{A_{P_i}\} & t = z_{xy} \text{ and } (x,y) \in \mathcal{R}_{P_i}. \end{cases}$

The construction above implies $(x,y) \in \mathcal{R}_{P_i}$ iff $(x,y) \in \mathcal{R}'_{f_1^-;A_{P_i}?;f_2}$.

Since $\mathcal{R}'_{f_1}, \mathcal{R}'_{f_2}$ are partial functions, it follows that Θ_1 is satisfied all over M'. Finally, it is easy to verify by induction on Φ that $M, s \models \Phi$ iff $M', s \models \mu_0(\Phi)$.

$\Leftarrow$ Let $M' = \{\mathcal{S}', \{\mathcal{R}'_{f_1}, \mathcal{R}'_{f_2}\}, \Pi'\}$ be a model of $\mu_1(\Phi)$. We can define a model $M = \{\mathcal{S}, \{\mathcal{R}_P\}, \Pi\}$ of Φ as follows. First we define $\overline{\mathcal{R}}_{P_i} = \mathcal{R}'_{f_1^-;A_{P_i}?;f_2}$ Then, let $s \in \mathcal{S}'$ be a state such that $M', s \models \mu_1(\Phi)$, we define $\mathcal{S} = \{t \mid (s,t) \in (\bigcup_i(\overline{\mathcal{R}}_{P_i} \cup \overline{\mathcal{R}}_{P_i}^-))^*\}$, $\mathcal{R}_{P_i} = \overline{\mathcal{R}}_{P_i} \cap (\mathcal{S} \times \mathcal{S})$, $\Pi(t) = \Pi'(t) - \{A_{P_i} \text{ for any } P_i\}$ for all $t \in \mathcal{S}$. Observe however that because of the constructs for graded nondeterminism, we need to make sure that for each $(x,y) \in \mathcal{R}'_{f_1^-;A_{P_i}?;f_2}$ there is exactly one z_{xy} such that $(z_{xy}, x) \in \mathcal{R}'_{f_1}$ and $(z_{xy}, y) \in \mathcal{R}'_{f_2}$. That is the following constraint must hold:

$$\begin{aligned} &(A_{P_i} \in \Pi'(z_1) \wedge A_{P_i} \in \Pi'(z_2)) \Rightarrow \\ &\quad \neg((z_1, x) \in \mathcal{R}'_{f_1} \wedge (z_2, x) \in \mathcal{R}'_{f_1} \wedge (z_1, y) \in \mathcal{R}'_{f_2} \wedge (z_2, y) \in \mathcal{R}'_{f_2}). \end{aligned}$$

We claim that without loss of generality we can assume the constraint above to be satisfied by M'. Indeed, suppose that this was not the case - i.e., suppose that there exist z_1, z_2 such that $A_{P_i} \in \Pi'(z_1) \wedge A_{P_i} \in \Pi'(z_2)$ and $(z_1, x) \in \mathcal{R}'_{f_1} \wedge (z_2, x) \in \mathcal{R}'_{f_1} \wedge (z_1, y) \in \mathcal{R}'_{f_2} \wedge (z_2, y) \in \mathcal{R}'_{f_2}$. Then, consider the model M'' made up by the model M' and an exact copy of M'. Let z'_1, z'_2, x', y' be the states corresponding to z_1, z_2, x, y, respectively, in the copy. By definition of M'', the states y and y' satisfy exactly the same formulae. So we can modify M'' by removing (z_2, y) and (z'_2, y') from $\mathcal{R}''_{f_2}$ replacing them with (z_2, y') and (z'_2, y). The thus modified structure is still model of $\mu_1(\Phi)$.[7] Proceeding in this way for all the states violating the above constraint, we get a model that satisfies it.

Finally, assuming that M' does satisfy the above constraint, it is easy to verify that by induction of $\mu_0(\Phi)$ that $M', s \models \mu_0(\Phi)$ iff $M, s \models \Phi$. $\Box$

Phase 2

We define $\mu_2(\Phi)$ as follows:

1. In $\mu_o(\Phi)$, we recursively replace
   - every occurrence of program $f_1^-; A_{P_i}; f_2$ by

     $$(F_{i,1}; A_{P_i}?; (F'_{i,1}; A_{P_i}?)^*; (F_{i,2}; A_{P_i}?; (F'_{i,2}; A_{P_i}?)^*)^-,$$

     except for those in a construct for graded nondeterminism, where $F_{i,j}, F'_{i,j}$ ($i = 1, \ldots, m$ and $j = 1, 2$) are new atomic programs;

[7] Obviously the same thing can be done starting from x and x'.

- every $(\leq n\ f_1^-; A_{P_i}?; f_2.\phi)$ by

$$[(F_{i,1}; A_{P_i}?; (F'_{i,1}; A_{P_i}?)^*; (\phi'?; (F'_{i,1}; A_{P_i}?)^*)^n]\neg\phi',$$

  and every $(\geq n\ f_1^-; A_{P_i}; f_2.\phi)$ by

$$< (F_{i,1}; A_{P_i}?; (F'_{i,1}; A_{P_i}?)^*; (\phi'?; (F'_{i,1}; A_{P_i}?)^*)^{n-1} > \phi',$$

  where $\phi' =< ((F_{i,2}; A_{P_i}?; (F'_{i,2}; A_{P_i}?)^*)^- > \phi$;[8]

- every $(\leq n\ (f_1^-; A_{P_i}?; f_2)^-.\phi)$ by

$$[(F_{i,2}; A_{P_i}?; (F'_{i,2}; A_{P_i}?)^*; (\phi''?; (F'_{i,2}; A_{P_i}?)^*)^n]\neg\phi'',$$

  and every $(\geq n\ (f_1^-; A_{P_i}; f_2)^-.\phi)$ by

$$< (F_{i,2}; A_{P_i}?; (F'_{i,2}; A_{P_i}?)^*; (\phi''?; (F'_{i,2}; A_{P_i}?)^*)^{n-1} > \phi'',$$

  where $\phi'' =< ((F_{i,1}; A_{P_i}?; (F'_{i,1}; A_{P_i}?)^*)^- > \phi$.

Let us denote the resulting formula by $\mu'_0(\Phi)$.

2. We put in conjunction with $\mu'_0(\Phi)$, the formula $\Theta_2 = [(\bigcup_{i=1,m}\bigcup_{j=1,2}((F_{i,j} \cup F'_{i,j} \cup F^-_{i,j} \cup F'^-_{i,j}))^*]\theta_{1,1} \wedge \theta_{1,2} \wedge \ldots \wedge \theta_{m,1} \wedge \theta_{m,2}$, where each $\theta_{i,j}$ is of the form:

$$(\leq 1\ F_{i,j}) \wedge (\leq 1\ F'_{i,j}) \wedge (\leq 1\ F^-_{i,j}) \wedge (\leq 1\ F'^-_{i,j}) \wedge \neg(< F_{i,j} > \top\wedge < F'_{i,j} > \top).$$

This formula constraints the models of $\mu_2(\Phi)$ so that $\mathcal{R}_{F_{i,j}}, \mathcal{R}_{F^-_{i,j}}, \mathcal{R}_{F'_{i,j}}, \mathcal{R}_{F'^-_{i,j}}$ are (partial) functions, and each state cannot be linked to other states by both $\mathcal{R}_{F_{i,j}}$ and $\mathcal{R}_{F'_{i,j}}$. Together these facts imply that $\mathcal{R}_{(F_{i,j};A_{P_i}?;(F_{i,j};A_{P_i}?)^*)^-}$ is a (partial) function.

Lemma 6 *$\mu_1(\Phi)$ is satisfiable iff $\mu_2(\Phi)$ is satisfiable.*

Proof $\Rightarrow$ Let $M = \{\mathcal{S}, \{\mathcal{R}_{f_1}, \mathcal{R}_{f_2}\}, \Pi\}$ be a model of $\mu_1(\Phi)$. Then we can build a model $M' = \{\mathcal{S}', \{\mathcal{R}'_F\}, \Pi'\}$ of $\mu_2(\Phi)$ as follows. First, we define $\{\overline{\mathcal{R}}'_F\}$. Let $x \in \mathcal{S}$ be any state such that $M, x \models< f_1^-; A_{P_i}?; f_2 > \top$, and let $z_1, \ldots, z_l$ be all the states such that $(x, z_k) \in \mathcal{R}_{f_1^-}$ and $M, z_k \models< A_{P_i}?; f_2 > \top$. We put $(x, z_1) \in \overline{\mathcal{R}}'_{F_{i,1}}$ and for all $k = 1, \ldots, l-1$ we put $(z_k, z_{k+1}) \in \overline{\mathcal{R}}'_{F'_{i,1}}$. We proceed similarly for $x \in S$ such that $M, x \models< f_2^-; A_{P_i}?; f_1 > \top$. Then, let $s \in \mathcal{S}$ be such that $M, s \models \mu_1(\Phi)$, we define $\mathcal{S}' = \{t \mid (s,t) \in (\bigcup_{i=1,m}\bigcup_{j=1,2}(\overline{\mathcal{R}}'_{F_{i,j}} \cup \overline{\mathcal{R}}'_{F'_{i,j}} \cup \overline{\mathcal{R}}'^-_{F_{i,j}} \cup \overline{\mathcal{R}}'^-_{F'_{i,j}}))^*\}$, $\mathcal{R}_F = \overline{\mathcal{R}}'_F \cap (\mathcal{S}' \times \mathcal{S}')$, and $\Pi'(t) = \Pi(t)$ for all $t \in \mathcal{S}'$. Note that since $\mathcal{R}_{f_i}$ is a partial function, $\mathcal{R}'_{(F_{i,j};A_{P_i}?;(F'_{i,j};A_{P_i}?)^*)^-}$ is a partial function as well. By this construction we have that

$$(x,y) \in \mathcal{R}_{f_1^-;A_{P_i};f_2} \quad \text{iff} \quad (x,y) \in \mathcal{R}'_{F_{i,1};A_{P_i}?;(F'_{i,1};A_{P_i}?)^*;(F_{i,2};A_{P_i}?;(F'_{i,2};A_{P_i}?)^*)^-}.$$

Moreover, Θ_2 is satisfied all over M'.

[8] $(\phi'?; (F'_{i,j}; A_{P_i}?)^*)^n$ stands for n repetitions of $\phi'?; (F'_{i,j}; A_{P_i}?)^*)$.

Finally, considering that $\mathcal{R}'_{(F_{i,j};A_{P_i}?;(F'_{i,j};A_{P_i}?)^*)^-}$ is a partial function, and that $[(F_{i,j}; A_{P_i}?; (F'_{i,j}; A_{P_i}?)^*; (\phi?; (F'_{i,j}; A_{P_i}?)^*)^n]\neg\phi$ $(< (F_{i,j}; A_{P_i}?; (F'_{i,j}; A_{P_i}?)^*; (\phi?; (F'_{i,j}; A_{P_i}?)^*)^{n-1} > \phi)$ specifies that there are at most (at least) n states satisfying ϕ, along the chain $(F_{i,j}; A_{P_i}?; (F'_{i,j}; A_{P_i}?)^*$, it is easy to verify by induction on $\mu_0(\Phi)$ that $M, s \models \mu_0(\Phi)$ iff $M', s \models \mu'_0(\Phi)$.

$\Leftarrow$ Let $M' = \{\mathcal{S}', \{\mathcal{R}'_F\}, \Pi'\}$ be a model of $\mu_2(\Phi)$. We can define a model $M = \{\mathcal{S}, \{\mathcal{R}_{f_1}, \mathcal{R}_{f_2}\}, \Pi\}$ of $\mu_1(\Phi)$ as follows. First we define $\overline{\mathcal{R}}_{f_j} = \mathcal{R}'_{(F_{i,j};A_{P_i}?;(F'_{i,j};A_{P_i}?)^*)^-}$ $(j = 1, 2)$. Then let $s \in S'$ be such that $M', s \models \mu_2(\Phi)$, we define $\mathcal{S} = \{t \mid (s,t) \in (\overline{\mathcal{R}}_{f_1} \cup \overline{\mathcal{R}}_{f_2} \cup \overline{\mathcal{R}}^-_{f_1} \cup \overline{\mathcal{R}}^-_{f_2})^*\}$, $\mathcal{R}_{f_j} = \overline{\mathcal{R}}_{f_j} \cap (\mathcal{S} \times \mathcal{S})$, and $\Pi(t) = \Pi'(t)$ for all $t \in S$.

Note that, by Θ_2, $\mathcal{R}'_{(F_{i,j};A_{P_i}?;(F'_{i,j};A_{P_i}?)^*)^-}$ is a partial function, and hence $\mathcal{R}_{f_j}$ is a partial function as well, thus Θ_1 is satisfied all over M.

Finally, considering again the meaning of $[(F_{i,j}; A_{P_i}?; (F'_{i,j}; A_{P_i}?)^*; (\phi?; (F'_{i,j}; A_{P_i}?)^*)^n]\neg\phi$ $(< (F_{i,j}; A_{P_i}?; (F'_{i,j}; A_{P_i}?)^*; (\phi?; (F'_{i,j}; A_{P_i}?)^*)^{n-1} > \phi)$, it is easy to verify by induction on $\mu'_0(\Phi)$ that $M', s \models \mu'_0(\Phi)$ iff $M, s \models \mu_0(\Phi)$. $\Box$

Observing that $\mu_2(\Phi)$ is a formula of $\mathcal{L}_{ld}$, we get the main result of this section.

Theorem 7 *A formula Φ of $\mathcal{L}_{gn}$ is satisfiable iff the formula $\mu_2(\Phi)$ of $\mathcal{L}_{ld}$ is satisfiable.*

Considering that $\mu_2(\Phi)$ is at most polynomially longer than Φ we can state what is the complexity of reasoning in $\mathcal{L}_{gn}$.

Corollary 8 *Satisfiability in $\mathcal{L}_{gn}$ is an EXPTIME-complete problem.*

5 Conclusions

We have discussed two extensions of Converse PDL, which include constructs for local determinism and graded nondeterminism respectively, showing that satisfiability in the resulting logics is polynomially reducible to satisfiability in Converse PDL.

Among the various features of the logics presented in this paper, it is worth mentioning that they allow us to model states that are in relationship with n other states through n-ary relations (vs. binary relations as usual) - e.g., $(s_1, \ldots, s_n) \in \mathcal{R}_R$ can be expressed by $(s_R, s_1) \in \mathcal{R}_{r_1}, \ldots, (s_R, s_n) \in \mathcal{R}_{r_n}$ where s_R is a new state and each r_i is an atomic program which is deterministic wrt the state s_R - and assign to such relations functionality/cardinality constraints - e.g., there are at most, say, three tuples whose first component is the same (see[3]). Although the full impact of these possibility on reasoning about programs needs still to be investigated, we believe that being able to specify n-ary relations is important at least for the use of these logics in Knowledge Representation.

Acknowledgments

This work was partly funded be the ESPRIT BRA Compulog II, and the italian CNR under Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, LDR Ibridi.

References

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

[2] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94). To appear, 1994.

[3] G. De Giacomo and M. Lenzerini. Description logics with inverse roles, functional restrictions, and n-ary relations. In Proceedings of the 4th European Workshop on Logics in AI (JELIA-94). To appear, 1994.

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

[5] K. Fine. In so many possible worlds. *Notre Dame Journal of Formal Logic*, 13(4):516–520, 1972.

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

[7] D. Harel. Dynamic logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, pages 497–603. D. Reidel Publishing Company, Oxford, 1984.

[8] D. Kozen and J. Tiuryn. Logics of programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 790–840. Elsevier Science Publishers, 1990.

[9] R. Parikh. The completeness of propositional dynamic logic. In *Proceedings of the 7th Symposium on the Mathematical Foundations of Computer Science*, number 64 in Lecture Notes in Computer Science, pages 403–415. Springer-Verlag, 1978.

[10] R. Parikh. Propositional dynamic logic of programs: A survey. In *Proceedings of the 1st Workshop on Logic of Programs*, number 125 in Lecture Notes in Computer Science, pages 102–144. Springer-Verlag, 1981.

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

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

[13] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471, 1991.

[14] C. Stirling. Modal and temporal logic. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, pages 477–563. Clarendon Press, Oxford, 1992.

[15] W. van der Hoek. On the semantics of graded modalities. *Journal of Applied Non-Classical Logics*, 2(1):81–123, 1992.

[16] M. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.