Enhanced Propositional Dynamic Logic for Reasoning about Concurrent Actions (extended abstract)

Giuseppe De Giacomo and Maurizio Lenzerini

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italia
{degiacom,lenzerini}@assi.dis.uniroma1.it

Introduction
This paper presents a work in progress on enhanced Propositional Dynamic Logics for reasoning about actions. Propositional Dynamic Logics (PDL's) are modal logics for describing and reasoning about system dynamics in terms of properties of states and actions modeled as relations between states (see (Kozen & Tiuryn 1990; Harel 1984; Parikh 1981) for surveys on PDL's, see also (Stirling 1992) for a somewhat different account). The language of PDL includes formulae built from the boolean combinations of atomic propositions that are interpreted as simple properties of states, plus the construct $\langle R \rangle \phi$, where $\phi$ is a formula and $R$ is an action, whose meaning is that it is possible to perform $R$ and terminate in a state where $\phi$ is true. The action $R$ can be either an atomic action, or a complex expression denoting sequential composition, nondeterministic choice, iteration, or test.

PDL's have been originally developed in Theoretical Computer Science to reason about program schemas (Fisher & Ladner 1979), and their variants have been adopted to specify and verify properties of reactive processes (e.g., Hennessy Milner Logic (Hennessy & Milner 1985; Milner 1989), modal mu-calculus (Kozen 1983; Larsen 1990; Stirling 1992)). They are also of interest in Philosophical Logic as a formalism to capture "procedural reasoning" (see, for example, (Van Benthem & Bergstra 1993; Van Benthem, Van Eijck, & Steblerstova 1993; de Rijke.M 1992; Van Benthem 1991)).

In Artificial Intelligence, PDL's have been extensively used in establishing decidability and computational complexity results of many formalisms: for example they have been used in investigating Common Knowledge (Halpern 1992), Conditional Logics (Friedman & Halpern 1994), Description Logics (Schild 1991; De Giacomo & Lenzerini 1994a; 1994c), Features Logics (Blackburn & Spaan 1993). However they have been only sparingly adopted for reasoning about actions, main exceptions being (Rosenschein 1991; Kautz 1980) (but also (Cohen & Levesque 1990)).

Propositional Dynamic Logics offer a elegant framework with a well understood semantics and precise computational characterization, that in our opinion makes them a kind of Principled Monotonic Propositional Situation Calculus extended to deal with complex actions.\footnote{In this perspective many recent results on PDL's are relevant, for example (Danecki 1984; STarch & Wolper 1986; Passy & Tinchev 1991; De Giacomo & Lenzerini 1994b).}

In this paper we propose a new Propositional Dynamic Logic that includes boolean expressions of primitive actions denoting sets of primitive actions executed concurrently, and that allows to represent interdependencies between primitive actions as specialization or disjointness. Furthermore the logic includes constructs to impose the determinism of boolean combinations of primitive actions and their inverse. We have established that such logic is decidable and its computational complexity is EXPTIME (tight bound). We show some possible use of this logic in reasoning about actions by means of examples.

The logic DIFR
Formulae in the logic DIFR are of two sorts: action formulae and state formulae.

Action Formulae describe, by means of boolean operators, properties of atomic actions -i.e., actions that cannot be broken into sequences of smaller actions. The abstract syntax of action formulae is as follows:

\[
\rho ::= P \mid \text{any} \mid \rho_1 \land \rho_2 \mid \rho_1 \lor \rho_2 \mid \neg \rho
\]

where $P$ denotes a primitive action, any denotes a special atomic action that can be thought of as "the most general atomic action", and $\rho$ (possibly with subscript) denotes an action formula. Observe that an atomic action denoted by an action formula is composed, in general, by a set of primitive actions intended to be executed in parallel.

State Formulae describe properties of states in terms of propositions and complex actions. The ab-

\[1\]In this work we do not distinguish between actions and events.
expressed as: the inability to perform any atomic action other than those denoted by \( r \). The formula \( (\neg\text{any}) \perp \) expresses that if the atomic action \( r \) is performed, then it deterministically leads to a state where \( \phi \) holds. Note that this does not imply that the action \( r \) can be performed. The formula \( (\neg\text{any}) \perp \) expresses that atomic action \( r \) can be performed and deterministically leads to a state where \( \phi \) holds.

Propositional Dynamic Logics (PDL) are subsets of Second Order Logic, or, more precisely, of First Order Logic plus Fixpoints. Typical properties that are not first order definable are: \( (R^*) \phi \), which expresses the capability for performing \( R \) until \( \phi \) holds, and is equivalent to the least fixpoint of the operator \( \lambda X.(\phi \lor (R)X) \); \( \neg\text{any} \phi \), which expresses that \( \phi \) holds in any state reachable from the current one by performing \( R \) any number of times, and is equivalent to the greatest fixpoint of the operator \( \lambda X.(\phi \land (R)X) \). Interesting special cases of the last formula are: \( \text{any}^* \phi \), which expresses that \( \phi \) holds from now on -i.e., no matter how the world evolves from the current state \( \phi \) will be true; and \( \text{any} \lor \text{any}^- \phi \), which expresses that \( \phi \) holds in the whole connected component containing the current state (the state in which the formula holds).

The formal semantics of \( \text{DLFR} \) is based on the notion of Kripke structure (or interpreted transition system), which is defined as a triple \( M = (S, \{R_R\}, \mathcal{V}) \), where \( S \) denotes a set of states, \( \{R_R\} \) is a family of binary relations over \( S \), such that each action \( R \) is given a meaning through \( R_R \), and \( \mathcal{V} \) is a mapping from \( S \) to atomic propositions such that \( \mathcal{V}(s) \) determines the propositions that are true in the state \( s \). The family \( \{R_R\} \) is systematically defined as follows:

\[
\begin{align*}
\mathcal{R}_\text{any} & \subseteq S \times S, \\
\mathcal{R}_P & \subseteq \mathcal{R}_\text{any}, \\
\mathcal{R}_{R_1 \land R_2} & = \mathcal{R}_{R_1} \cap \mathcal{R}_{R_2}, \\
\mathcal{R}_{R_1 \lor R_2} & = \mathcal{R}_{R_1} \cup \mathcal{R}_{R_2}, \\
\mathcal{R}_\neg & = \mathcal{R}_\text{any} - \mathcal{R}_\text{any}, \\
\mathcal{R}_\neg^* & = \{ (s_1, s_2) \in S \times S \mid (s_2, s_1) \in \mathcal{R}_\neg \}, \\
\mathcal{R}_{R} & = \mathcal{R}_\neg \text{ if } r = \neg, \\
\mathcal{R}_{R} & = \mathcal{R}_{\neg R} \text{ if } r = \neg^- \text{,} \\
\mathcal{R}_{R_1 \lor R_2} & = \mathcal{R}_{R_1} \cup \mathcal{R}_{R_2}, \\
\mathcal{R}_{R_1 ; R_2} & = \mathcal{R}_{R_1} \circ \mathcal{R}_{R_2} \quad \text{(seq. comp. of } \mathcal{R}_{R_1} \text{ and } \mathcal{R}_{R_2} \text{)}, \\
\mathcal{R}_{R}^* & = (\mathcal{R}_{R})^* \text{ (refl. trans. closure of } \mathcal{R}_{R}), \\
\mathcal{R}_{\neg \neg} & = \{ (s_1, s_2) \in S \times S \mid (s_2, s_1) \in \mathcal{R}_{R} \}, \\
\mathcal{R}_{\phi R} & = \{ (s, s) \in S \times S \mid M, s \models \phi \}.
\end{align*}
\]

Note that actions (even primitive actions) are nondeterministic in general.

The conditions for a state formula \( \phi \) to hold at a state \( s \) of a structure \( M \), written \( M, s \models \phi \), are:
\( M, s \models A \text{ iff } s \in V(A) \)
\( M, s \models T \text{ always, } \)
\( M, s \models \bot \text{ never, } \)
\( M, s \models \phi_1 \land \phi_2 \text{ iff } M, s \models \phi_1 \text{ and } M, s \models \phi_2, \)
\( M, s \models \phi_1 \lor \phi_2 \text{ iff } M, s \models \phi_1 \text{ or } M, s \models \phi_2, \)
\( M, s \models \neg \phi \text{ iff } M, s \not\models \phi, \)
\( M, s \models (\forall^\exists) \phi \text{ iff } \exists s'. (s, s') \in R_R \text{ and } M, s' \models \phi, \)
\( M, s \models (\forall^\forall) \phi \text{ iff } \forall s'. (s, s') \in R_R \text{ implies } M, s' \models \phi, \)
\( M, s \models (\forall^\exists^\forall) \phi \text{ iff at most one } s'. (s, s') \in R_R. \)

A structure \( M \) is a model of an action formula \( p \) if \( R_\phi = R_\text{any}. \) A structure \( M \) is a model of a state formula \( \phi \) if for all \( s \) in \( M, M, s \models \phi. \) Let \( \Gamma \) be a finite set of both state and action formulae, a structure is a model of \( \Gamma \) if is a model of every formula in \( \Gamma. \) A set of formulae \( \Gamma \) logically implies a (state or action) formula \( \psi, \) written
\( \Gamma \models \psi \)
if all the models of \( \Gamma \) are models of \( \psi \) as well.

A crucial question to be answered is: Is logical implication decidable in DILFR? And if yes, which is its computational complexity? Note that known results in PDL's do not help directly. We have proven that this problem is indeed decidable and we have precisely characterized its computational complexity, by providing a reduction to the PDL DILF presented in (De Giacomo & Lenzerini 1994a).

**Theorem 1** Logical implication for DILFR is an EXPTIME-complete problem.

Observe that logical implication is already EXPTIME-complete for the basic modal logic \( \mathcal{K} \) (which corresponds to a Propositional Dynamic Logic including just one primitive action, no functional restrictions, and no action constructors at all).

**Using DILFR for reasoning about actions**

Below we show the power of DILFR in modeling a dynamically changing world by means of two examples. We remark that those examples do not aim at providing the definitive DILFR-based formalizations of the scenarios they describe, nor they exhaust the possibility of using DILFR in representing and reasoning about actions. They are intended to give a taste of what can be done with such a logic. In the examples we refer to situation calculus as it is presented in (Reiter 1991; 1992b; 1993; Lin & Reiter 1994).

**Example: lifting both sides of a table**

A vase is on top of a table, and if just one side is lifted then it slides down and falls on the floor. However if both sides are simultaneously lifted this doesn't happen (Grosse 1994). We formalize the scenario as follows. We consider the following primitive propositions (corresponding to "propositional" fluents in situation calculus): \textit{vase.on.table}, \textit{down.left.side}, \textit{down.right.side}; and the following primitive actions (corresponding to actions in situation calculus\(^5\)):
\textit{vase.slides.down, lift.left, lift.right}. The intended meaning of these propositions and actions is the natural one (sometimes we use initials as abbreviations). We do not include actions to put down the table for sake of brevity.

As usual actions have preconditions which are conditions that must be satisfied in order to be able to perform the action\(^6\).

\[
\begin{align*}
(lift.left)T &\equiv down.left.side \\
(lift.right)T &\equiv down.right.side \\
vase_slides_down &\equiv (vol \wedge ((dlsl \wedge \neg drs) \wedge (\neg dls \wedge drs)))
\end{align*}
\]

Actually the if part of the last axioms must be strengthened: If the vase is on the table and one of the side of the table is not on the floor, then it is inevitable (not just possible) that the vase slides towards the floor. This can be enforced by:
\[
(vo\wedge((dls\wedge\neg drs)\wedge(\neg dls\wedge drs))) \Rightarrow (\text{any})T \wedge [\neg vase]_.
\]
We need also to specify when the actions \textit{lift.left} and \textit{lift.right} can be performed simultaneously. With the next axiom we assert that they can be performed simultaneously simply when they both can be performed:
\[
(lift.left \wedge lift.right)T \equiv (lift.left)T \wedge (lift.right)T.
\]

Actions have effects if they can be performed\(^7\):
\[
\begin{align*}
\textit{[lift.left]} &\sim down.left.side \\
\textit{[lift.right]} &\sim down.right.side \\
\textit{[vase.slides.down] } &\sim vase.on.table.
\end{align*}
\]

As usual we need to cope with the frame problem. We do it by adopting a monotonic solution as in (Haas 1987; Schubert 1990; Reiter 1991). We enforce the following frame axioms saying that if the vase in on the table then all atomic actions not including \textit{vase.slides.down} leave the vase on the table; if the vase is not on the table then no atomic action will

---

\(^5\)Note that (contrary to what is usually assumed in situation calculus) actions are not necessarily deterministic in DILFR.

\(^6\)State formulae of the form \((a)T\) have the same role as \textit{Poss(a, s)} in Reiter's situation calculus.

\(^7\)State formulae of the form \([a]\phi\) have the same role as \textit{Poss(a, s) \Rightarrow \phi(do(a, s))} which is a common formula configuration in Reiter's situation calculus (Reiter 1991; 1993).
change its position; etc.:

\[
\begin{align*}
vase_{on\,table} &\Rightarrow \neg [vase_{slides\,down}]vase_{on\,table} \\
down_{left\,side} &\Rightarrow \neg [\text{lift_{left}}]down_{left\,side} \\
down_{right\,side} &\Rightarrow \neg [\text{lift_{right}}]down_{right\,side} \\
\neg vase_{on\,table} &\Rightarrow [\text{any}]vase_{on\,table} \\
\neg down_{left\,side} &\Rightarrow [\text{any}]down_{left\,side} \\
\neg down_{right\,side} &\Rightarrow [\text{any}]down_{right\,side}.
\end{align*}
\]

Let us call \( P \) the set of the axioms above, and let the starting situation be described by

\[
S \equiv vase_{on\,table} \land down_{left\,side} \land down_{right\,side}.
\]

Then we can make the following two inferences. On the one hand:

\[
\Gamma \models S \Rightarrow (ll \land lr)[vase_{slides\,down}]T
\]

that is if the vase is on the table and both the sides of the table are on the floor, then lifting the two sides concurrently does not make the vase falling. On the other hand:

\[
\Gamma \models S \Rightarrow (ll \land \neg lr)[lr]vase_{on\,table}
\]

that is if the vase is on the table and both the sides of the table are on the floor, then lifting first the left side without lifting the right side, and then the right side, has as a result that the vase is fallen. Notice that the above inferences don't say anything about the possibility of performing the actions described, however this possibility is guaranteed by \( \Gamma \models S \Rightarrow (lift_{left} \land lift_{right})T \) and \( \Gamma \models S \Rightarrow ((lift_{left} \land lift_{right}); lift_{right})T \), respectively.

Example: making the heating operative

We want to make our (gas) heating operative. To do so we need to strike a match, to turn its gas handle on and to ignite the security flame spot. To strike a match we need to concurrently press the match against the match box and rub it until it fires.

We make the following intuitive assumption: the past is backward linear that is from any state there is only one accessible (immediately) previous state. This can be easily imposed by means of the following axiom:

\[
\text{(fun any\textsuperscript{\textminus})}.
\]

We assume the following preconditions and effects of actions.

Preconditions:

\[
\begin{align*}
\langle \text{turn}_{on\,gas} \rangle T &\equiv \neg \text{gas}_{open} \\
\langle \text{turn}_{off\,gas} \rangle T &\equiv \text{gas}_{open} \\
\langle \text{ignite}_{flame\,spot} \rangle T &\equiv \text{match}_{lit} \\
\langle \text{press} \rangle T &\Rightarrow \neg \text{match}_{lit} \\
\langle \text{rub} \rangle T &\Rightarrow \neg \text{match}_{lit} \\
\langle \text{while} \neg \text{match}_{lit} \text{ do } (\text{press} \land \text{rub}) \rangle T.
\end{align*}
\]

Effects:

\[
\begin{align*}
\text{match}_{lit} \land \neg \text{gas}_{open} &\Rightarrow \neg \text{match}_{lit} \land \text{gas}_{open} \\
\langle \text{ignite}_{flame\,spot} \rangle \text{heating}_{operative} &\Rightarrow \text{gas}_{open} \\
\langle \text{turn}_{on\,gas} \rangle \text{gas}_{open} &\Rightarrow \text{heating}_{operative} \\
\langle \text{turn}_{off\,gas} \rangle \neg \text{gas}_{open} &\Rightarrow \neg \text{heating}_{operative}.
\end{align*}
\]

In this example we model frame axioms more systematically starting from explanation closure axioms (Schubert 1990) in line with (Reiter 1991; 1993). There are two main difficulty in following this approach in PDL: the first is that, as in any standard modal logic, we can directly refer to just one state, the "current one"; the second is that we cannot quantify on atomic actions. In DLFK we can overcome these difficulties. By assuming (fun any\textsuperscript{\textminus}) from the current state we can univocally refer back to the previous state through the action any\textsuperscript{\textminus}. On the other hand by using the action any we can simulate the universal quantification on atomic actions. Hence we proceed as follows from the current state we make a step forward and then we model the various condition backward. This leads to the following frame axioms:

\[
\begin{align*}
\text{[any]} &\Rightarrow \neg \text{gas}_{open} \Rightarrow \langle \neg \text{turn}_{off\,gas} \rangle \text{gas}_{open} \lor (\langle \text{turn}_{on\,gas} \rangle \text{gas}_{open}) \\
\text{[any]} &\Rightarrow \neg \text{match}_{lit} \Rightarrow \langle \neg \text{match}_{lit} \rangle \\
\text{[any]} &\Rightarrow \neg \text{heating}_{operative} \Rightarrow \langle \neg \text{heating}_{operative} \rangle \\
\text{[any]} &\Rightarrow \text{gas}_{open} \Rightarrow \langle \text{turn}_{on\,gas} \rangle \text{gas}_{open} \lor (\langle \text{turn}_{off\,gas} \rangle \text{gas}_{open}) \\
\text{[any]} &\Rightarrow \text{match}_{lit} \Rightarrow \langle \text{match}_{lit} \rangle \lor (\langle \text{press} \land \text{rub} \rangle \text{match}_{lit}) \\
\text{[any]} &\Rightarrow \langle \text{heating}_{operative} \rangle \Rightarrow \langle \text{heating}_{operative} \rangle \lor (\langle \text{ignite}_{flame\,spot} \rangle \text{gas}_{open}).
\end{align*}
\]

For example the last axiom says: "consider any successor state (such a state has exactly one previous state which is the current state), if the heating is operative in such a state then either it was operative in the previous state or the action \( \text{ignite}_{flame\,spot} \) was just performed and the gas was open in the previous state".\(^8\)

Let us call \( \Gamma \) the set of all these axioms, and let the starting situation be described by

\[
S \equiv \neg \text{open}_{gas} \land \neg \text{match}_{lit} \land \neg \text{heating}_{operative}
\]

\(^8\)The frame axioms can be proved to be equivalent to the following ones (respecting the order):

\[
\begin{align*}
\text{gas}_{open} &\Rightarrow \neg \langle \text{turn}_{off\,gas} \rangle \text{gas}_{open} \\
\text{match}_{lit} &\Rightarrow \langle \text{any} \rangle \text{match}_{lit} \\
\text{heating}_{operative} &\Rightarrow \langle \text{any} \rangle \text{heating}_{operative} \\
\neg \text{gas}_{open} &\Rightarrow \neg \langle \text{turn}_{on\,gas} \rangle \neg \text{gas}_{open} \\
\neg \text{match}_{lit} &\Rightarrow \neg \langle \text{press} \land \text{rub} \rangle \text{match}_{lit} \\
\neg \text{heating}_{operative} &\Rightarrow \langle \neg \text{heating}_{operative} \rangle \\
\langle \neg \text{ignite}_{flame\,spot} \rangle \text{gas}_{open} &\Rightarrow \langle \text{heating}_{operative} \rangle \lor (\langle \neg \text{ignite}_{flame\,spot} \rangle \text{gas}_{open}).
\end{align*}
\]

The last axiom says: "if the heating is not operative then both every performance of an atomic action not including \( \text{ignite}_{flame\,spot} \), and every performance of any action starting from a state in which the gas is not open, leads to a state where the heating is still not operative".
The first inference we are interested in is the following:

\[ \Gamma \models S \Rightarrow (\text{any}^*) \text{heating-operative} \]

i.e. there is a sequence of action (a plan) starting from a situation described by \( S \) resulting in making the heating operative. Assuming all primitive actions to be deterministic, inferences of the form

\[ \Gamma \models S \Rightarrow (\text{any}^*) \]

are the typical starting point in planning synthesis (Green 1969): if the answer is yes then from the proof we can generate a working plan to achieve the goal \( G \) starting from an initial situation described by \( S \). The dual of the above inference

\[ \Gamma \models S \Rightarrow [\text{any}^*] \neg G \]

is of interest as well: it establishes that there are no plan at all achieving a given goal \( G \) starting from a situation described be \( S \).

Next inference says that the complex action “strike a match, turn on the gas, ignite the control flame spot” results in making the heating operative:

\[ \Gamma \models (\text{while}-\neg \text{match\_lit do (press } \land \text{ push); turn\_on\_gas; ignite\_flame\_spot )heating\_operative.} \]

Note that the similar action “turn on the gas, strike a match, ignite the control flame spot” is not guaranteed to make the heating operative:

\[ \Gamma \not\models (\text{turn\_on\_gas}; \text{while}-\neg \text{match\_lit do (press } \land \text{ push); ignite\_flame\_spot )heating\_operative.} \]

The reason why above the complex action may fail is because the gas could be turned off while we are trying to strike the match.

The problem of checking inferences as the two above is known as projection problem (see e.g. (Reiter 1992b)). A typical projection problem as the form: Does \( G \) hold in a state reachable from initial situation, described as \( S \), by executing the (complex) action \( \alpha \)? This corresponds to checking the inference below:

\[ \Gamma \models S \Rightarrow (\alpha)G. \]

We have seen that executing the complex action “turn on the gas, strike a match, ignite the control flame spot” may fail to make the heating operative. If this is the case, the following inference tells us that the gas has been turned off before striking the match succeeded:

\[ \Gamma \models (\text{turn\_on\_gas}; \text{while}-\neg \text{match\_lit do (press } \land \text{ push); ignite\_flame\_spot })(-\text{heating\_operative } \Rightarrow \text{((any\^\_}\_\_;any\^\_\_\_);turn\_off\_gas)T).} \]

Inferences as the one above are answers to “historical queries” (Reiter 1992b; 1992a). i.e., queries of the form: if from the initial state described by \( S \) we execute the complex action \( \alpha \) getting \( \phi \), then does this implies that before the termination of \( \alpha \), \( \phi' \) is true in some state, or does it implies that the action \( \alpha \) as been executed? These questions can be answered by checking the inferences 9:

\[ \Gamma \models S \Rightarrow (\alpha)(\phi \Rightarrow (\text{any}^*)\phi') \]

\[ \Gamma \models S \Rightarrow (\alpha)(\phi \Rightarrow (\text{any}^*; \text{a}^-)T). \]

References


9Observe that \( \phi' (\alpha) \) could be true (executed) before the starting of \( \alpha \) in the formulation above. If we want to avoid this, we may assume that the initial situation does not have a past, which can be done by including in \( S \) the state formula \([\text{any}]_I \). Another solution is to add a new proposition \( \text{starting\_state} \) to \( S \) and impose the axiom

\[ \text{starting\_state } \Rightarrow [\text{any};\text{any}^*] \lor (\text{any}^-; (\text{any}^*^-)\neg \text{starting\_state} \]

saying that in every state reachable from one in which \( \text{starting\_state} \) holds, \( \text{starting\_state} \) does not hold (this does not influence logical implications not involving the proposition \( \text{starting\_state} \), since DTLR satisfies the tree model property). In this way we can test for the starting state in going backward along \( \alpha \).


