Resource separation in dynamic logic of propositional assignments

Joseph Boudou, Andreas Herzig, Nicolas Troquard

Resource Separation in Dynamic Logic of Propositional Assignments

Joseph Boudou[1], Andreas Herzig[1], Nicolas Troquard[2]

[1] IRIT, CNRS, France
[2] Free University of Bozen-Bolzano, Italy

Abstract. We extend dynamic logic of propositional assignments by adding an operator of parallel composition that is inspired by separation logics. We provide an axiomatisation via reduction axioms, thereby establishing decidability. We also prove that the complexity of both the model checking and the satisfiability problem stay in PSPACE.

Keywords: Dynamic logic, separation logic, propositional assignments, parallel composition

1 Introduction

It is notoriously delicate to extend Propositional Dynamic Logic PDL with an operator of parallel composition of programs. Several attempts were made in the literature: Abrahamson as well as Mayer and Stockmeyer studied a semantics in terms of interleaving [MS96]; Peleg and Goldblatt modified the interpretation of programs from a relation between possible worlds to a relation between possible worlds and sets thereof [Pel87,Gol92]; Balbiani and Vakarelov studied the interpretation of parallel composition of programs π_1 and π_2 as the intersection of the accessibility relations interpreting π_1 and π_2 [BV03]. However, it seems fair to say that there is still no consensus which of these extensions is the 'right' one.

Dynamic Logic of Propositional Assignments DL-PA [BHT13,BHST14] is a version of Propositional Dynamic Logic PDL whose atomic programs are assignments of propositional variables p to true or false, respectively written $+p$ and $-p$. We and coauthors have shown that many knowledge representation concepts and formalisms can be captured in DL-PA, such as update and revision operations [Her14], database base fusion and repair operations [FHR19], planning [HMNDBW14,HMV19], lightweight dynamic epistemic logics [CS15,CHM$^+$16,CS17], and judgment aggregation [NGH18]. The mathematical properties of DL-PA are simpler than those of PDL, in particular, the Kleene star can be eliminated [BHT13] and satisfiability and model checking are both PSPACE complete [BHST14].

In this paper we investigate how dynamic logic can be extended with a program operator of parallel composition $\pi_1 \| \pi_2$ of two programs π_1 and π_2 that is inspired by separation logic. The latter was proposed in the literature as an account of concurrency, e.g. by Brookes and by O'Hearn [O'H04,Bro04,Bro07,BO16]. Their Concurrent Separation Logic is characterised by two main principles:

1. When two programs are executed in parallel then the state of the system is partitioned ('separated') between the two programs: the perception of the state and its modification is viewed as being local to each of the two parallel programs. Each of them therefore has a partial view of the global state. This in particular entails that parallelism in itself does not modify the state of the system: the parallel execution of two programs that do nothing does not change the state. This means that the formula $\varphi \to [\top?\|\top?]\varphi$ should be valid, where "$?$" is the test operator. These tests $\varphi?$ differ from standard PDL tests; this will be explained when we discuss what system states should look like.
2. The execution of a parallel program $\pi_1\|\pi_2$ should be insensitive to the way the components of π_1 and π_2 are interleaved. So "race conditions" [BO16] must be avoided: the execution should not depend on the order of execution of atomic actions in π_1 and π_2, where we consider tests to be atomic, too. Here we interpret this in a rather radical way: when there is a race condition between two programs then they cannot be executed in parallel. For example, the parallel program $+p\|-p$ where $+p$ makes p true and $-p$ makes p false is inexecutable because there is a conflict: the two possible interleavings $+p;-p$ and $-p;+p$ are not equivalent. We even consider that $+p\|+p$ and $p?\|+p$ are inexecutable, which some may consider a bit over-constrained.

Together, the above two principles entail that the dynamic logic formula

$$[(\pi_1;\varphi_1?)\|(\pi_2;\varphi_2?)](\varphi_1 \wedge \varphi_2)$$

should be valid. If we replace φ_1 by p and φ_2 by $\neg p$ then the above tells us that $(\pi_1; p?)\|(\pi_2; \neg p?)$ is inexecutable. So the program $\pi_1;(p?\|\pi_2);\neg p?$ that is obtained from it by interleaving should be inexecutable, too.

We have not yet said what one should understand by a DL-PA system state. A previous approach of ours only considered the separation of valuations, i.e., of truth values of propositional variables [Her13]. Two separating conjunctions in the style of separation logic were defined on such models. This however did not allow us to define an adjoint implication as usually done in the separation logic literature, which was somewhat unsatisfactory. The paper [HMV19] has richer models where valuations are supplemented by information about writability of variables, supposing that a variable can only be assigned by a program when it is writable. Splitting and merging of such models can be defined in a natural way, thus providing a meaningful interpretation of parallel composition. We here push this program further and consider models having moreover information about readability of variables. We suppose that writability implies readability[3] and that a variable can only be tested if it is readable. So our tests $\varphi?$ differ from standard PDL tests and also from DL-PA tests in that their executability depends on whether the relevant variables are readable. In particular, while $\langle p?\rangle\top \to p$, remains valid, its converse $p \to \langle p?\rangle\top$ becomes invalid in our logic.

[3] As suggested by one of the reviewers, it may be relaxed and one may suppose that a program can only modify a variable without being able to read its value. This would simplify the presentation of the logic; however, we believe that our inclusion constraint is natural in most applications.

The paper is organised as follows. In Section 2 we define models and the two ternary relations 'split' and 'merge' on models. In Section 3 we define the language of our logic and in Section 4 we give the interpretation of formulas and programs. In Section 5 we axiomatise the valid formulas by means of reduction axioms and in Section 6 we establish that the satisfiability problem is PSPACE complete. Section 7 concludes.

2 Models and Their Splitting and Merging

Let $\mathbb{P}$ be a countable set of propositional variables. A model is a triple $\mathsf{m} = \langle \mathsf{Rd}, \mathsf{Wr}, \mathsf{V} \rangle$ where Rd, Wr, and V are subsets of $\mathbb{P}$ such that $\mathsf{Wr} \subseteq \mathsf{Rd}$. The idea is that Rd is the set of readable variables, Wr is the set of writable variables, and V is a valuation: its elements are true, while those of its complement $\mathbb{P} \setminus \mathsf{V}$ are false. The constraint that $\mathsf{Wr} \subseteq \mathsf{Rd}$ means that writability implies readability.

Two models $\mathsf{m}_1 = \langle \mathsf{Rd}_1, \mathsf{Wr}_1, \mathsf{V}_1 \rangle$ and $\mathsf{m}_2 = \langle \mathsf{Rd}_2, \mathsf{Wr}_2, \mathsf{V}_2 \rangle$ are *RW-compatible* if and only if writable variables of one model and the readable variables of the other do not interfere, i.e., if and only if $\mathsf{Wr}_1 \cap \mathsf{Rd}_2 = \mathsf{Wr}_2 \cap \mathsf{Rd}_1 = \emptyset$. For example, $\mathsf{m}_1 = \langle \{p\}, \{p\}, \emptyset \rangle$ and $\mathsf{m}_2 = \langle \{p\}, \emptyset, \emptyset \rangle$ are not RW-compatible: in m_1, some program π_1 modifying the value of p may be executable, while for programs executed in m_2, the value of p may differ depending on whether it is read before or after the modification by π_1 took place.

As writability implies readability, RW-compatibility of m_1 and m_2 implies $\mathsf{Wr}_1 \cap \mathsf{Wr}_2 = \emptyset$.

We define ternary relations $\triangleleft$ ('split') and $\triangleright$ ('merge') on models as follows:

$$\mathsf{m} \triangleleft \begin{smallmatrix} \mathsf{m}_1 \\ \mathsf{m}_2 \end{smallmatrix} \text{ iff } \mathsf{m}_1 \text{ and } \mathsf{m}_2 \text{ are RW-compatible, } \mathsf{Rd} = \mathsf{Rd}_1 \cup \mathsf{Rd}_2, \mathsf{Wr} = \mathsf{Wr}_1 \cup \mathsf{Wr}_2, \text{ and } \mathsf{V} = \mathsf{V}_1 = \mathsf{V}_2$$

$$\begin{smallmatrix} \mathsf{m}_1 \\ \mathsf{m}_2 \end{smallmatrix} \triangleright \mathsf{m} \text{ iff } \mathsf{m}_1 \text{ and } \mathsf{m}_2 \text{ are RW-compatible, } \mathsf{Rd} = \mathsf{Rd}_1 \cup \mathsf{Rd}_2, \mathsf{Wr} = \mathsf{Wr}_1 \cup \mathsf{Wr}_2, \mathsf{V}_1 \setminus \mathsf{Wr} = \mathsf{V}_2 \setminus \mathsf{Wr}, \text{ and } \mathsf{V} = (\mathsf{V}_1 \cap \mathsf{Wr}_1) \cup (\mathsf{V}_2 \cap \mathsf{Wr}_2) \cup (\mathsf{V}_1 \cap \mathsf{V}_2)$$

For example, for $\mathsf{m} = \langle \mathsf{Rd}, \mathsf{Wr}, \mathsf{V} \rangle$ we have $\mathsf{m} \triangleleft \begin{smallmatrix} \mathsf{m} \\ \mathsf{m}_2 \end{smallmatrix}$ for every $\mathsf{m}_2 = \langle \mathsf{Rd}_2, \emptyset, \mathsf{V} \rangle$ such that $\mathsf{Rd}_2 \subseteq \mathbb{P} \setminus \mathsf{Wr}$. Observe that, contrarily to splitting, merging does not keep the valuation constant: it only keeps constant the non-modifiable part $\mathsf{V} \setminus \mathsf{Wr}$ of the valuation V and puts the results of the allowed modifications of Wr together. These modifications cannot conflict because m_1 and m_2 are RW-compatible. Figure 1 illustrates each of these two operations by an examples. The checks that are performed in the merge operation are reminiscent of the self composition technique in the analysis of secure information flows ([DHS05,SG16].[4]

The set Rd of readable variables of a model m induces an indistinguishability relation between models:

$$\mathsf{m} \sim \mathsf{m}' \text{ iff } \mathsf{Rd} = \mathsf{Rd}', \mathsf{Wr} = \mathsf{Wr}', \mathsf{V} \cap \mathsf{Rd} = \mathsf{V}' \cap \mathsf{Rd}'$$

So m and m' are indistinguishable if (1) they have the same readable and writable variables and (2) the valuations are identical as far as their readable parts are concerned. This relation will serve to interpret tests: the test $\varphi?$ of a formula φ is conditioned by its truth in all read-indistinguishable models, i.e., in all models where the readable variables have the same truth value.

[4] We are grateful to Rainer Hähnle for pointing this out to us.

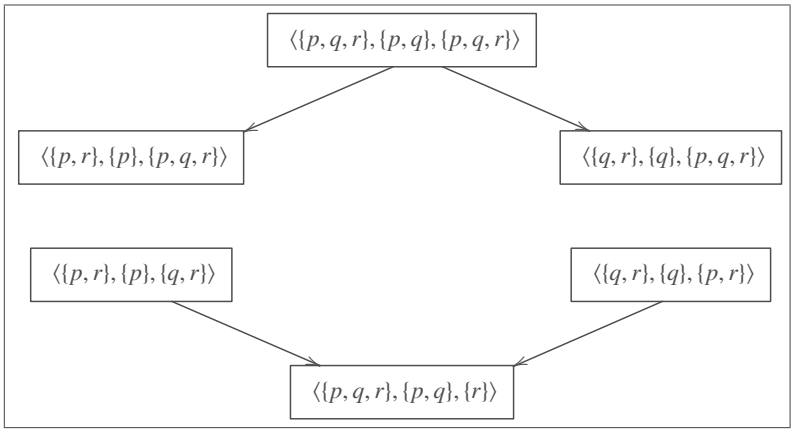

Fig. 1. Examples of split and merge operations: the top half illustrates the split of the model $\langle\{p,q,r\},\{p,q\},\{p,q,r\}\rangle$ into $\langle\{p,r\},\{p\},\{p,q,r\}\rangle$ and $\langle\{q,r\},\{q\},\{p,q,r\}\rangle$; the bottom half illustrates the merge of the models $\langle\{p,r\},\{p\},\{p,q,r\}\rangle$ and $\langle\{q,r\},\{q\},\{p,q,r\}\rangle$ into $\langle\{p,q,r\},\{p,q\},\{r\}\rangle$.

3 Language

We use $p, q, \ldots$ for variables in the set of propositional variables $\mathbb{P}$. Formulas and programs are defined by the following grammar, where p ranges over $\mathbb{P}$:

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle\varphi$$
$$\pi ::= +p \mid -p \mid \mathtt{r}{+}p \mid \mathtt{r}{-}p \mid \mathtt{w}{+}p \mid \mathtt{w}{-}p \mid \varphi? \mid \varphi\textbf{?} \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid \pi\|\pi$$

The program $+p$ makes p true and $-p$ makes p false. The executability of these two programs is conditioned by the writability of p. The program $\mathtt{r}{+}p$ makes p readable and $\mathtt{r}{-}p$ makes p unreadable; similarly, $\mathtt{w}{+}p$ makes p writable and $\mathtt{w}{-}p$ makes p non-writable. We suppose that these four programs are always executable. The program $\varphi?$ is the PDL test that φ, that we call *exogeneous*; $\varphi\textbf{?}$ is the *endogeneous* test that φ: it is conditioned by the readability of the relevant variables of φ.

The formula $[\pi]\varphi$ abbreviates $\neg\langle\pi\rangle\neg\varphi$. Given an integer $n \geq 0$, the program π^n is defined inductively by $\pi^0 = \top?$ and $\pi^{n+1} = \pi;\pi^n$. Similarly, $\pi^{\leq n}$ is defined by $\pi^{\leq 0} = \top?$ and $\pi^{\leq n+1} = \top? \cup (\pi;\pi^{\leq n})$. The program **if** φ **then** π abbreviates $(\varphi?;\pi) \cup \neg\varphi?$. For a finite set of variables $P = \{p_1, \ldots, p_n\}$ and associated programs $\{\pi_1(p_1), \ldots, \pi_n(p_n)\}$, we are going to use the notation $\fatsemi_{p\in P}\, \pi(p)$ to denote the sequence $\pi_1(p_1); \cdots ; \pi_n(p_n)$, in some order. We will make use of this notation with care to guarantee that the ordering of the elements of P does not matter.

The set of propositional variables occurring in a formula φ is noted $\mathbb{P}(\varphi)$ and the set of those occurring in a program π is noted $\mathbb{P}(\pi)$. For example, $\mathbb{P}(p \vee \langle{+}q\rangle\neg r) = \{p, q, r\}$.

4 Semantics

Let $\mathsf{m} = \langle \mathsf{Rd}, \mathsf{Wr}, \mathsf{V} \rangle$ be a model. Formulas are interpreted as sets of models:

$$
\begin{array}{ll}
\mathsf{m} \models \top & \\
\mathsf{m} \models p & \text{iff } p \in \mathsf{V}, \text{ for } p \in \mathbb{P} \\
\mathsf{m} \models \neg\varphi & \text{iff } \mathsf{m} \not\models \varphi \\
\mathsf{m} \models \varphi \vee \psi & \text{iff } \mathsf{m} \models \varphi \text{ or } \mathsf{m} \models \psi \\
\mathsf{m} \models \langle\pi\rangle\varphi & \text{iff there is a model } \mathsf{m}' \text{ such that } \mathsf{m}[\![\pi]\!]\mathsf{m}' \text{ and } \mathsf{m}' \models \varphi
\end{array}
$$

Programs are interpreted as relations on the set of models:

$$
\begin{array}{ll}
\mathsf{m}[\![+p]\!]\mathsf{m}' & \text{iff } \mathsf{Rd}' = \mathsf{Rd}, \mathsf{Wr}' = \mathsf{Wr}, \mathsf{V}' = \mathsf{V} \cup \{p\}, \text{ and } p \in \mathsf{Wr} \\
\mathsf{m}[\![-p]\!]\mathsf{m}' & \text{iff } \mathsf{Rd}' = \mathsf{Rd}, \mathsf{Wr}' = \mathsf{Wr}, \mathsf{V}' = \mathsf{V} \setminus \{p\}, \text{ and } p \in \mathsf{Wr} \\
\mathsf{m}[\![\mathtt{r}{+}p]\!]\mathsf{m}' & \text{iff } \mathsf{Rd}' = \mathsf{Rd} \cup \{p\}, \mathsf{Wr}' = \mathsf{Wr}, \text{ and } \mathsf{V}' = \mathsf{V} \\
\mathsf{m}[\![\mathtt{r}{-}p]\!]\mathsf{m}' & \text{iff } \mathsf{Rd}' = \mathsf{Rd} \setminus \{p\}, \mathsf{Wr}' = \mathsf{Wr} \setminus \{p\}, \text{ and } \mathsf{V}' = \mathsf{V} \\
\mathsf{m}[\![\mathtt{w}{+}p]\!]\mathsf{m}' & \text{iff } \mathsf{Rd}' = \mathsf{Rd} \cup \{p\}, \mathsf{Wr}' = \mathsf{Wr} \cup \{p\}, \text{ and } \mathsf{V}' = \mathsf{V} \\
\mathsf{m}[\![\mathtt{w}{-}p]\!]\mathsf{m}' & \text{iff } \mathsf{Rd}' = \mathsf{Rd}, \mathsf{Wr}' = \mathsf{Wr} \setminus \{p\}, \text{ and } \mathsf{V}' = \mathsf{V} \\
\mathsf{m}[\![\varphi?]\!]\mathsf{m}' & \text{iff } \mathsf{m} = \mathsf{m}' \text{ and } \mathsf{m} \models \varphi \\
\mathsf{m}[\![\varphi\mathbin{?\!?}]\!]\mathsf{m}' & \text{iff } \mathsf{m} = \mathsf{m}' \text{ and } \mathsf{m}'' \models \varphi \text{ for every } \mathsf{m}'' \text{ such that } \mathsf{m}'' \sim \mathsf{m} \\
\mathsf{m}[\![\pi_1;\pi_2]\!]\mathsf{m}' & \text{iff there is an } \mathsf{m}'' \text{ such that } \mathsf{m}[\![\pi_1]\!]\mathsf{m}'' \text{ and } \mathsf{m}''[\![\pi_2]\!]\mathsf{m}' \\
\mathsf{m}[\![\pi_1 \cup \pi_2]\!]\mathsf{m}' & \text{iff } \mathsf{m}[\![\pi_1]\!]\mathsf{m}' \text{ or } \mathsf{m}[\![\pi_2]\!]\mathsf{m}' \\
\mathsf{m}[\![\pi^*]\!]\mathsf{m}' & \text{iff there is an } n \geq 0 \text{ such that } \mathsf{m}[\![\pi]\!]^n\mathsf{m}' \\
\mathsf{m}[\![\pi_1 \| \pi_2]\!]\mathsf{m}' & \text{iff there are } \mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_1', \mathsf{m}_2' \text{ such that } \mathsf{m} \triangleleft \begin{smallmatrix} \mathsf{m}_1 & \mathsf{m}_1' \\ \mathsf{m}_2 & \mathsf{m}_2' \end{smallmatrix} \triangleright \mathsf{m}', \\
 & \quad \mathsf{m}_1[\![\pi_1]\!]\mathsf{m}_1', \mathsf{Rd}_1 = \mathsf{Rd}_1', \mathsf{Wr}_1 = \mathsf{Wr}_1', \mathsf{V}_1 \setminus \mathsf{Wr}_1 = \mathsf{V}_1' \setminus \mathsf{Wr}_1', \\
 & \quad \mathsf{m}_2[\![\pi_2]\!]\mathsf{m}_2', \mathsf{Rd}_2 = \mathsf{Rd}_2', \mathsf{Wr}_2 = \mathsf{Wr}_2', \mathsf{V}_2 \setminus \mathsf{Wr}_2 = \mathsf{V}_2' \setminus \mathsf{Wr}_2'
\end{array}
$$

In the interpretation of assignments of atomic formulas we require propositional variables to be modifiable, while readability and writability can be modified unconditionally. When a variable is made writable then it is made readable, too, in order to guarantee the inclusion constraint on models; similarly when a variable is made unreadable.

The interpretation of parallel composition $\pi_1 \| \pi_2$ is such that both π_1 and π_2 only modify 'their' variables: parallel composition $\pi_1 \| \pi_2$ of two programs π_1 and π_2 relates two models m and m' when the following conditions are satisfied: (1) m can be split into m_1 and m_2; (2) the execution of π_1 on m_1 may lead to m_1' and the execution of π_2 on m_2 may lead to m_2'; (3) m_1' and m_2' can be merged into m'. Moreover, (4) the modifications are legal: π_1 and π_2 neither change readability nor writability, and each of them only modifies variables that were allocated to it by the split.

Figure 2 illustrates the interpretation of the parallel program $-p \| -q$. Some more examples follow.

Example 1. Suppose $\mathsf{m} = \langle \mathsf{Rd}, \mathsf{Wr}, \mathsf{V} \rangle$ with $\mathsf{Wr} = \mathsf{Rd} = \mathsf{V} = \{p, q, r\}$. Then $\mathsf{m}' = \langle \mathsf{Rd}, \mathsf{Wr}, \mathsf{V}' \rangle$ with $\mathsf{V}' = \{p, r\}$ is the only model such that $\mathsf{m}[\![+p \| -q]\!]\mathsf{m}'$.

The next example illustrates the last condition (4) in the interpretation of parallel composition.

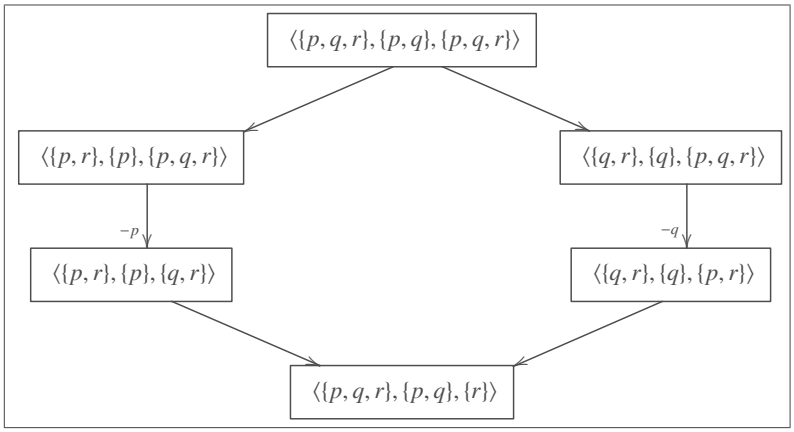

Fig. 2. Illustration of an execution of $-p\|-q$ at the model $\langle\{p,q,r\},\{p,q\},\{p,q,r\}\rangle$.

Example 2. The program $+p\|(\mathtt{w}{+}p;-p;\mathtt{r}{-}p)$ cannot be executed on the model $\mathsf{m} = \langle\{p\},\{p\},\{p\}\rangle$. Indeed, suppose there are m_1 and m_2 such that $\mathsf{m} \triangleleft \frac{\mathsf{m}_1}{\mathsf{m}_2}$ and suppose $+p$ is executed on m_1 and $\mathtt{w}{+}p;-p;\mathtt{r}{-}p$ on m_2. For $+p$ to be executable we must have $p \in \mathsf{Wr}_1$, and therefore $p \notin \mathsf{Wr}_2$ (and a fortiori $p \notin \mathsf{Rd}_2$). So $\mathsf{m}_1 = \langle\{p\},\{p\},\{p\}\rangle$ and $\mathsf{m}_2 = \langle\emptyset,\emptyset,\{p\}\rangle$. Then $\mathsf{m}'_1 = \mathsf{m}_1$ is the only model such that $\mathsf{m}_1[\![+p]\!]\mathsf{m}'_1$; and $\mathsf{m}'_2 = \langle\emptyset,\emptyset,\emptyset\rangle$ is the only model such that $\mathsf{m}_2[\![\mathtt{w}{+}p;-p;\mathtt{r}{-}p]\!]\mathsf{m}'_2$. These two models cannot be merged: $\mathsf{V}'_2 \setminus \mathsf{Wr}'_2 = \emptyset$ fails to be equal to $\mathsf{V}_2 \setminus \mathsf{Wr}_2 = \{p\}$.

Let us finally explain the different semantics of the two test operators of our logic.

Example 3. Suppose m is such that $p \notin \mathsf{Rd}$ and $p \in \mathsf{V}$. Then the program $p?$ is executable at m because $p \in \mathsf{V}$. In contrast, there is no m' such that $\mathsf{m}[\![p\overline{?}]\!]\mathsf{m}'$, the reason being that there is always an m'' such that $\mathsf{m} \sim \mathsf{m}''$ and $p \notin \mathsf{V}''$. So $p\overline{?}$ fails at m.

In practice, parallel programs should only contain endogeneous tests in order to avoid that a subprogram accesses the truth value of a variable that is not among its readable variables. Actually we have kept PDL tests for technical reasons only: we could not formulate some of the reduction axioms without them.

Satisfiability and validity of formulas are defined in the expected way.

Example 4. The formulas $\varphi \to [\top\overline{?}\|\top\overline{?}]\varphi$, $[+p\|-p]\bot$, $[+p\|+p]\bot$ and $[p\overline{?}\|+p]\bot$ whose programs were mentioned in the introduction are all valid.

The formulas $\langle+p\rangle\top$ and $\langle-p\rangle\top$ both express that p is writable. Similarly, $\langle p\overline{?}\rangle\top$ expresses that p is true and readable, and $\langle\neg p\overline{?}\rangle\top$ expresses that p is false and readable; therefore $\langle p\overline{?}\rangle\top \vee \langle\neg p\overline{?}\rangle\top$ expresses that p is readable. This will be instrumental in our axiomatisation.

5 Axiomatisation via Reduction Axioms

We axiomatise the validities of our logic by means of reduction axioms, as customary in dynamic epistemic logics [vDvdHK07]. These axioms transform every formula into a boolean combination of propositional variables and formulas of the form $\langle +p\rangle\top$ and $\langle p?\rangle\top \vee \langle \neg p?\rangle\top$. The former expresses that p is writable: we abbreviate it by $\mathtt{w}_p$. The latter expresses that p is readable: we abbreviate it by $\mathtt{r}_p$.

We start by eliminating all the program operators from formulas, where the elimination of parallel composition is done by sequentialising it while keeping track of the values of the atoms. After that step, the only remaining program operators either occur in formulas of the form $\mathtt{r}_p$ or $\mathtt{w}_p$, or are assignments of the form $\langle +p\rangle$, $\langle -p\rangle$, $\langle \mathtt{r}{+}p\rangle$, $\langle \mathtt{r}{-}p\rangle$, $\langle \mathtt{w}{+}p\rangle$, or $\langle \mathtt{w}{-}p\rangle$. The assignments can be distributed over the boolean operators, taking advantage of the fact that all of them are deterministic modal operators (validating the Alt_1 axiom $\langle\pi\rangle\varphi \to [\pi]\varphi$). Finally, sequences of such modalities facing a propositional variable can be transformed into boolean combinations of readability and writability statements $\mathtt{r}_p$ and $\mathtt{w}_p$. The only logical link between these statements is that writability of p implies readability of p. This is captured by the axiom schema $\mathtt{w}_p \to \mathtt{r}_p$.

The sequentialisation of parallel composition uses copies of variables, so we start by introducing that notion. We then define some programs and formulas that will allow us to formulate the reduction axioms more concisely.

5.1 Copies of Atomic Propositions

We are going to need fresh copies of each propositional variable, one per occurrence of the parallel composition operator. In order to keep things readable we neglect that the index of the copies should be attached to programs and denote the copies of the variable p by $p^{\mathbf{k}}$, where $\mathbf{k}$ is some integer. In principle we should introduce a bijection between the indexes $\mathbf{k}$ and the subprogram they are attached to; we however do not do so to avoid overly complicated notations.

Given a set of propositional variables $P \subseteq \mathbb{P}$ and an integer $\mathbf{k} \in \{1, 2\}$, we define the set of copies $P^{\mathbf{k}} = \{p^{\mathbf{k}} \ : \ p \in P\}$.

5.2 Useful Programs

Let $P \subseteq \mathbb{P}$ be some finite set of propositional variables. Figure 3 lists four programs that will be useful to concisely formulate the reduction axioms. We comment on them in the sequel.

The $copyV^{\mathbf{k}}(P)$ program assigns the truth value of each variable $p \in P$ to its copy $p^{\mathbf{k}}$.

The $splitRW(P)$ program simulates the split operation by nondeterministically assigning two copies of each read and write variable in a way such that a counterpart of the RW-compatibility constraint $\mathsf{Wr}_1 \cap \mathsf{Rd}_2 = \mathsf{Wr}_2 \cap \mathsf{Rd}_1 = \emptyset$ is guaranteed. Note that the assignments $\mathtt{w}{+}p^{\mathbf{k}}$ also make $p^{\mathbf{k}}$ readable.

The $copybackRW^{\mathbf{k}}(P)$ program will be executed after the split and before the subprogram $\pi_{\mathbf{k}}$ in order to make the readable and writable variables be all those p that $splitRW(P)$ has assigned to $\pi_{\mathbf{k}}$.

$$
\begin{aligned}
\mathit{copyV}^{\mathbf{k}}(P) &= \mathbin{;}_{p\in P} ((p?; {+}p^{\mathbf{k}}) \cup (\neg p?; {-}p^{\mathbf{k}})) \\
\mathit{splitRW}(P) &= \mathbin{;}_{p\in P} (\mathtt{r}{-}p^{\mathbf{1}}; \mathtt{r}{-}p^{\mathbf{2}}; \\
&\qquad\qquad \textbf{if } \mathtt{w}_p \textbf{ then } (\mathtt{w}{+}p^{\mathbf{1}} \cup \mathtt{w}{+}p^{\mathbf{2}}); \\
&\qquad\qquad \textbf{if } \mathtt{r}_p \wedge \neg \mathtt{w}_p \textbf{ then } (\mathtt{r}{+}p^{\mathbf{1}} \cup \mathtt{r}{+}p^{\mathbf{2}} \cup (\mathtt{r}{+}p^{\mathbf{1}}; \mathtt{r}{+}p^{\mathbf{2}}))) \\
\mathit{copybackRW}^{\mathbf{k}}(P) &= \mathbin{;}_{p\in P} ((\neg \mathtt{r}_{p^{\mathbf{k}}}?; \mathtt{r}{-}p) \cup \\
&\qquad\qquad (\mathtt{r}_{p^{\mathbf{k}}} \wedge \neg \mathtt{w}_{p^{\mathbf{k}}}?; \mathtt{r}{+}p; \mathtt{w}{-}p) \cup \\
&\qquad\qquad (\mathtt{w}_{p^{\mathbf{k}}}?; \mathtt{w}{+}p)) \\
\mathit{mergeRW}(P) &= \mathbin{;}_{p\in P} (\textbf{if } \mathtt{r}_{p^{\mathbf{1}}} \textbf{ then } \mathtt{r}{+}p \ ; \ \textbf{if } \mathtt{w}_{p^{\mathbf{1}}} \textbf{ then } \mathtt{w}{+}p)
\end{aligned}
$$

Fig. 3. Useful programs, for $P \subseteq \mathbb{P}$ and $\mathbf{k} \in \{1, 2\}$

The *mergeRW*(P) program simulates the merge operation by reinstating all those read- and write-atoms that had been allocated to the first subprogram in the sequentialisation.

Observe that in the sequential compositions $\mathbin{;}_{p\in P}(\ldots)$ occurring in the above programs, the order of the variables does not matter. Observe also that the only endogeneous tests on the right hand side occur in readability statements $\mathtt{r}_p$ (which abbreviate $\langle p? \rangle \top \vee \langle \neg p? \rangle \top$).

Lemma 1. $\mathsf{m}[\![\mathit{copyV}^{\mathbf{k}}(P)]\!]\mathsf{m}'$ *if and only if* $\mathsf{Rd}' = \mathsf{Rd}$, $\mathsf{Wr}' = \mathsf{Wr}$, *and* $\mathsf{V}' = (\mathsf{V} \setminus P^{\mathbf{k}}) \cup (\mathsf{V} \cap P)^{\mathbf{k}}$.

Lemma 2. *For all models* m *and* m' *and all sets of propositional variables P, the following three are equivalent:*

1. $\mathsf{m}[\![\mathit{splitRW}(P)]\!]\mathsf{m}'$*;*
2. *for* $\mathbf{k} \in \{1, 2\}$, $\mathsf{Rd} \setminus P^{\mathbf{k}} = \mathsf{Rd}' \setminus P^{\mathbf{k}}$, $\mathsf{Wr} \setminus P^{\mathbf{k}} = \mathsf{Wr}' \setminus P^{\mathbf{k}}$, $\mathsf{V} = \mathsf{V}'$, *and for all* $p \in P$,
   - $p \in \mathsf{Rd}$ *iff* $(p^{\mathbf{1}} \in \mathsf{Rd}'$ *or* $p^{\mathbf{2}} \in \mathsf{Rd}')$,
   - $p \in \mathsf{Wr}$ *iff* $(p^{\mathbf{1}} \in \mathsf{Wr}'$ *or* $p^{\mathbf{2}} \in \mathsf{Wr}')$,
   - *if* $p^{\mathbf{1}} \in \mathsf{Wr}'$ *then* $p^{\mathbf{2}} \notin \mathsf{Rd}'$, *and if* $p^{\mathbf{2}} \in \mathsf{Wr}'$ *then* $p^{\mathbf{1}} \notin \mathsf{Rd}'$*;*
3. *there are* $\mathsf{m}_1, \mathsf{m}_2$ *such that* $\mathsf{m} \triangleleft \begin{smallmatrix}\mathsf{m}_1\\ \mathsf{m}_2\end{smallmatrix}$ *and for* $\mathbf{k} \in \{1, 2\}$*:*
   - $\mathsf{Rd}_{\mathbf{k}} = (\mathsf{Rd} \setminus P) \cup \{p \in P : p^{\mathbf{k}} \in \mathsf{Rd}'\}$,
   - $\mathsf{Wr}_{\mathbf{k}} = (\mathsf{Wr} \setminus P) \cup \{p \in P : p^{\mathbf{k}} \in \mathsf{Wr}'\}$,
   - $\mathsf{V}_{\mathbf{k}} = \mathsf{V} = \mathsf{V}'$.

Lemma 3. $\mathsf{m}[\![\mathit{copybackRW}^{\mathbf{k}}(P)]\!]\mathsf{m}'$ *if and only if:*

- $\mathsf{Rd}' = (\mathsf{Rd} \setminus P) \cup \{p \in P : p^{\mathbf{k}} \in \mathsf{Rd}\}$,
- $\mathsf{Wr}' = (\mathsf{Wr} \setminus P) \cup \{p \in P : p^{\mathbf{k}} \in \mathsf{Wr}\}$, *and*
- $\mathsf{V}' = \mathsf{V}$.

Lemma 4. $\mathsf{m}[\![\mathit{mergeRW}(P)]\!]\mathsf{m}'$ *if and only if:*

- $\mathsf{Rd}' = \mathsf{Rd} \cup \{p \in P : p^{\mathbf{1}} \in \mathsf{Rd}\}$,
- $\mathsf{Wr}' = \mathsf{Wr} \cup \{p \in P : p^{\mathbf{1}} \in \mathsf{Wr}\}$, *and*
- $\mathsf{V}' = \mathsf{V}$.

$$
\begin{aligned}
\langle \varphi ? \rangle \psi &\leftrightarrow \psi \wedge \varphi \\
\langle \varphi \text{?}? \rangle \psi &\leftrightarrow \psi \wedge [\ ;_{p \in \mathbb{P}(\varphi)} (\textbf{if } \neg \mathtt{r}_p \textbf{ then } (+p \cup -p))] \varphi \\
\langle \pi_1 ; \pi_2 \rangle \varphi &\leftrightarrow \langle \pi_1 \rangle \langle \pi_2 \rangle \varphi \\
\langle \pi_1 \cup \pi_2 \rangle \varphi &\leftrightarrow \langle \pi_1 \rangle \varphi \vee \langle \pi_2 \rangle \varphi \\
\langle \pi^* \rangle \varphi &\leftrightarrow \langle \pi^{\leq 2^{|\mathbb{P}(\varphi)|}} \rangle \varphi \\
\langle \pi_1 \| \pi_2 \rangle \varphi &\leftrightarrow \langle \mathit{splitRW}(\mathbb{P}(\pi_1) \cup \mathbb{P}(\pi_2)); \\
&\qquad \mathit{copybackRW}^1(\mathbb{P}(\pi_1)); \mathit{copyV}^1(\mathbb{P}(\pi_1)); \pi_1; \\
&\qquad\qquad \mathit{NochangeRW}^1(\mathbb{P}(\pi_1))?; \mathit{OkChangeV}^1(\mathbb{P}(\pi_1))?; \\
&\qquad \mathit{copybackRW}^2(\mathbb{P}(\pi_2)); \mathit{copyV}^2(\mathbb{P}(\pi_2)); \pi_2; \\
&\qquad\qquad \mathit{NochangeRW}^2(\mathbb{P}(\pi_2))?; \mathit{OkChangeV}^2(\mathbb{P}(\pi_2))?; \\
&\qquad \mathit{mergeRW}(\mathbb{P}(\pi_1))\rangle\, \varphi
\end{aligned}
$$

Fig. 4. Reduction axioms for program operators

5.3 Useful Formulas

The following formulas will be of use in reduction axioms:

$$
\mathit{NochangeRW}^{\mathbf{k}}(P) = \bigwedge_{p \in P} ((\mathtt{r}_p \leftrightarrow \mathtt{r}_{p^{\mathbf{k}}}) \wedge (\mathtt{w}_p \leftrightarrow \mathtt{w}_{p^{\mathbf{k}}}))
$$

$$
\mathit{OkChangeV}^{\mathbf{k}}(P) = \bigwedge_{p \in P} (\neg \mathtt{w}_p \rightarrow (p \leftrightarrow p^{\mathbf{k}}))
$$

The first is true if and only if readability and writability statements have the same value for every p in P and its copy $p^{\mathbf{k}}$.

5.4 Reduction Axioms for Program Operators

The reduction axioms for program operators are in Figure 4. Those for sequential and nondeterministic composition and for exogeneous tests (PDL tests) are as in PDL. That for endogeneous tests φ?? varies the truth values of the non-readable variables of φ. That for the Kleene star is familiar from DL-PA. That for parallel composition $\pi_1 \| \pi_2$ executes π_1 and π_2 in sequence: it starts by splitting up readability and writability between the two programs, then executes π_1, checks whether π_1 didn't change the readability and writability variables and whether all truth value changes it brought about are legal, and finally executes π_2 followed by the same checks for π_2.

Observe that the validity of the reduction axiom for $\|$ relies on the fact that the copies $p^{\mathbf{1}}$ and $p^{\mathbf{2}}$ that are introduced by the program $\mathit{splitRW}(\mathbb{P}(\pi_1) \cup \mathbb{P}(\pi_2))$ are fresh. One may also note that the length of the right hand side can be shortened by restricting $\mathbb{P}(\pi_{\mathbf{k}})$ to the propositional variables that are assigned by $\pi_{\mathbf{k}}$, i.e., to elements $p \in \mathbb{P}$ such that $+p$ or $-p$ occurs in $\pi_{\mathbf{k}}$.

$$\begin{array}{rcl}
\langle +p\rangle\top & \leftrightarrow & \mathtt{w}_p \\
\langle \mathtt{r}{+}p\rangle\top & \leftrightarrow & \top \\
\langle \mathtt{w}{+}p\rangle\top & \leftrightarrow & \top \\
\langle +p\rangle\neg\varphi & \leftrightarrow & \mathtt{w}_p \wedge \neg\langle +p\rangle\varphi \\
\langle \mathtt{r}{+}p\rangle\neg\varphi & \leftrightarrow & \neg\langle \mathtt{r}{+}p\rangle\varphi \\
\langle \mathtt{w}{+}p\rangle\neg\varphi & \leftrightarrow & \neg\langle \mathtt{w}{+}p\rangle\varphi \\
\langle +p\rangle(\varphi \vee \psi) & \leftrightarrow & \langle +p\rangle\varphi \vee \langle +p\rangle\psi \\
\langle \mathtt{r}{+}p\rangle(\varphi \vee \psi) & \leftrightarrow & \langle \mathtt{r}{+}p\rangle\varphi \vee \langle \mathtt{r}{+}p\rangle\psi \\
\langle \mathtt{w}{+}p\rangle(\varphi \vee \psi) & \leftrightarrow & \langle \mathtt{w}{+}p\rangle\varphi \vee \langle \mathtt{w}{+}p\rangle\psi
\end{array}
\qquad
\begin{array}{rcl}
\langle -p\rangle\top & \leftrightarrow & \mathtt{w}_p \\
\langle \mathtt{r}{-}p\rangle\top & \leftrightarrow & \top \\
\langle \mathtt{w}{-}p\rangle\top & \leftrightarrow & \top \\
\langle -p\rangle\neg\varphi & \leftrightarrow & \mathtt{w}_p \wedge \neg\langle -p\rangle\varphi \\
\langle \mathtt{r}{-}p\rangle\neg\varphi & \leftrightarrow & \neg\langle \mathtt{r}{-}p\rangle\varphi \\
\langle \mathtt{w}{-}p\rangle\neg\varphi & \leftrightarrow & \neg\langle \mathtt{w}{-}p\rangle\varphi \\
\langle -p\rangle(\varphi \vee \psi) & \leftrightarrow & \langle -p\rangle\varphi \vee \langle -p\rangle\psi \\
\langle \mathtt{r}{-}p\rangle(\varphi \vee \psi) & \leftrightarrow & \langle \mathtt{r}{-}p\rangle\varphi \vee \langle \mathtt{r}{-}p\rangle\psi \\
\langle \mathtt{w}{-}p\rangle(\varphi \vee \psi) & \leftrightarrow & \langle \mathtt{w}{-}p\rangle\varphi \vee \langle \mathtt{w}{-}p\rangle\psi
\end{array}$$

Fig. 5. Reduction axioms for boolean operators

The exhaustive application of the equivalences of Figure 4 from the left to the right results in formulas whose program operators are either endogeneous tests occurring in a readability statement $\mathtt{r}_p = \langle p?\rangle\top \vee \langle \neg p?\rangle\top$, or assignments of the form $\mathtt{r}{+}p$, $\mathtt{r}{-}p$, $\mathtt{w}{+}p$, $\mathtt{w}{-}p$, $+p$, or $-p$. (We recall that the last but one, $+p$, may also be written as $\mathtt{w}_p$.)

5.5 Reduction Axioms for Boolean Operators

We now turn to modal operators $\langle\pi\rangle$ where π is of the form $\mathtt{r}{+}p$, $\mathtt{r}{-}p$, $\mathtt{w}{+}p$, $\mathtt{w}{-}p$, $+p$, or $-p$. They are deterministic and can therefore be distributed over the boolean operators. The corresponding reduction axioms are in Figure 5.

We note that in the first equivalence $\langle +p\rangle\top \leftrightarrow \mathtt{w}_p$, the right hand side actually abbreviates the left hand side. We nevertheless state it in order to highlight that the exhaustive application of these reduction axioms results in sequences of atomic assignments facing either propositional variables or $\mathtt{w}_p$ or $\mathtt{r}_q$. These sequences are going to be reduced in the next step.

5.6 Reduction Axioms for Assignments

When atomic programs face propositional variables or readability and writability statements then the modal operator can be eliminated (sometimes introducing a writability statement $\mathtt{w}_p$). The reduction axioms doing that are in Figure 6.

As announced, the exhaustive application of the above equivalences results in boolean combinations of propositional variables and readability and writability statements.

5.7 Soundness, Completeness, and Decidability

Let us call DL-PA$^{\|}$ our extension of DL-PA with parallel composition. Its axiomatisation is made up of

– an axiomatisation of propositional logic;

$$\langle +p\rangle q \leftrightarrow \begin{cases} \mathtt{w}_p & \text{if } q = p \\ \mathtt{w}_p \wedge q & \text{otherwise} \end{cases} \qquad \langle -p\rangle q \leftrightarrow \begin{cases} \bot & \text{if } q = p \\ \mathtt{w}_p \wedge q & \text{otherwise} \end{cases}$$

$$\langle \mathtt{r}{+}p\rangle q \leftrightarrow q \qquad \langle \mathtt{r}{-}p\rangle q \leftrightarrow q$$

$$\langle \mathtt{w}{+}p\rangle q \leftrightarrow q \qquad \langle \mathtt{w}{-}p\rangle q \leftrightarrow q$$

$$\langle +p\rangle \mathtt{r}_q \leftrightarrow \mathtt{w}_p \wedge \mathtt{r}_q \qquad \langle -p\rangle \mathtt{r}_q \leftrightarrow \mathtt{w}_p \wedge \mathtt{r}_q$$

$$\langle \mathtt{r}{+}p\rangle \mathtt{r}_q \leftrightarrow \begin{cases} \top & \text{if } q = p \\ \mathtt{r}_q & \text{otherwise} \end{cases} \qquad \langle \mathtt{r}{-}p\rangle \top \leftrightarrow \begin{cases} \bot & \text{if } q = p \\ \mathtt{r}_q & \text{otherwise} \end{cases}$$

$$\langle \mathtt{w}{+}p\rangle \mathtt{r}_q \leftrightarrow \mathtt{r}_q \qquad \langle \mathtt{w}{-}p\rangle \mathtt{r}_q \leftrightarrow \mathtt{r}_q$$

$$\langle +p\rangle \mathtt{w}_q \leftrightarrow \mathtt{w}_p \wedge \mathtt{w}_q \qquad \langle -p\rangle \mathtt{w}_q \leftrightarrow \mathtt{w}_p \wedge \mathtt{w}_q$$

$$\langle \mathtt{r}{+}p\rangle \mathtt{w}_q \leftrightarrow \mathtt{w}_q \qquad \langle \mathtt{r}{-}p\rangle \mathtt{w}_q \leftrightarrow \mathtt{w}_q$$

$$\langle \mathtt{w}{+}p\rangle \mathtt{w}_q \leftrightarrow \begin{cases} \top & \text{if } q = p \\ \mathtt{w}_q & \text{otherwise} \end{cases} \qquad \langle \mathtt{w}{-}p\rangle \mathtt{w}_q \leftrightarrow \begin{cases} \bot & \text{if } q = p \\ \mathtt{w}_q & \text{otherwise} \end{cases}$$

Fig. 6. Reduction axioms for assignments

- the equivalences of Sections 5.4, 5.5, and 5.6;
- the inclusion axiom schema $\mathtt{w}_p \rightarrow \mathtt{r}_p$, which is an abbreviation of the formula $\langle +p\rangle\top \rightarrow (\langle p?\rangle\top \vee \langle \neg p?\rangle\top)$;
- the rule of equivalence $RE(\langle\pi\rangle)$ "from $\varphi \leftrightarrow \psi$ infer $\langle\pi\rangle\varphi \leftrightarrow \langle\pi\rangle\psi$".

The inference rules preserve validity and the axioms are valid:

Theorem 1. *The axiomatisation of* DL-PA$^{\|}$ *is sound: if φ is provable with the axiomatics of* DL-PA$^{\|}$ *then it is* DL-PA$^{\|}$ *valid.*

As to completeness, the reduction axioms of Sections 5.4, 5.5, and 5.6 allow us to transform any formula into an equivalent boolean combination of propositional variables and readability and writability statements. (Their application requires the rule of replacement of equivalents, which is derivable because we have rules of equivalence for all the connectives of the language, in particular the above $RE(\langle\pi\rangle)$.) Let φ be the resulting formula. Then φ has a DL-PA$^{\|}$ model if and only if

$$\varphi \wedge \bigwedge_{p\in\mathbb{P}} (\mathtt{w}_p \rightarrow \mathtt{r}_p)$$

has a model in propositional logic, where in propositional logic, $\mathtt{r}_p$ and $\mathtt{w}_p$ are considered to be arbitrary propositional variables; so there is a priori no connection between them nor with the propositional variable p.

Theorem 2. *The axiomatisation of* DL-PA$^{\|}$ *is complete: if φ is* DL-PA$^{\|}$ *valid then it is provable in the axiomatics of* DL-PA$^{\|}$.

Based on the reduction of DL-PA$^{\|}$ formulas to boolean formulas (and the transformation of $\mathtt{r}_p$ and $\mathtt{w}_p$ from abbreviations into propositional variables), we may check

the satisfiability of DL-PA$^{\|}$ formulas by means of propositional logic SAT solvers. This is however suboptimal because reduction may result in a formula that is super-exponentially longer than the original formula. In the next section we explore another route.

6 Complexity via Translation into DL-PA

We establish PSPACE complexity of DL-PA$^{\|}$ satisfiability and model checking by translating formulas and programs to Dynamic Logic of Propositional Assignments DL-PA. The language of the latter is the fragment of that of DL-PA$^{\|}$: it has neither endogeneous tests, nor readability and writability assignments, nor parallel composition. So the language of DL-PA is built by the following grammar:

$$\varphi ::= p \mid \top \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \langle\pi\rangle\varphi$$
$$\pi ::= +p \mid -p \mid \varphi? \mid (\pi;\pi) \mid (\pi \cup \pi) \mid \pi^*$$

None of the operators of the language refers to the Rd-component or the Wr-component of models. The interpretation of DL-PA formulas and programs therefore only requires a valuation V.

Our translation from DL-PA$^{\|}$ to DL-PA eliminates endogeneous tests and parallel composition. This is done in a way that is similar to their reduction axioms of Figure 4. It moreover transforms readability and writability statements into special propositional variables $\mathtt{r}_p$ and $\mathtt{w}_p$, similar to the reduction axioms of Figure 6.

To make this formal, let the set of *atomic formulas* be

$$\mathbb{X} = \mathbb{P} \cup \{\mathtt{w}_p \ : \ p \in \mathbb{P}\} \cup \{\mathtt{r}_p \ : \ p \in \mathbb{P}\}.$$

Given a set of propositional variables $P \subseteq \mathbb{P}$, $\mathbb{R}_P = \{\mathtt{r}_p \ : \ p \in P\}$ is the associated set of read-variables and $\mathbb{W}_P = \{\mathtt{w}_p \ : \ p \in P\}$ is the associated set of write-variables. So $\mathbb{X} = \mathbb{P} \cup \mathbb{R}_{\mathbb{P}} \cup \mathbb{W}_{\mathbb{P}}$. As before, the set of propositional variables occurring in a formula φ is noted $\mathbb{P}(\varphi)$ and the set of those occurring in a program π is noted $\mathbb{P}(\pi)$. This now includes the p's in $\mathtt{r}_p$ and $\mathtt{w}_p$. For example, $\mathbb{P}(p \wedge \langle +w_q\rangle\neg\mathtt{r}_p) = \{p, q\}$.

We translate the DL-PA$^{\|}$ programs $\mathtt{r}{+}p$, $\mathtt{r}{-}p$, $\mathtt{w}{+}p$, and $\mathtt{w}{-}p$ into the DL-PA programs $+\mathtt{r}_p$, $-\mathtt{r}_p$, $+\mathtt{w}_p$ and $-\mathtt{w}_p$. Moreover, we have to 'spell out' that $-\mathtt{w}_p$ has side effect $-\mathtt{r}_p$ and that $+\mathtt{r}_p$ has side effect $+\mathtt{w}_p$. So the programs of Figure 3 become the following DL-PA programs:

$$\begin{aligned}
\mathit{copyV}^{\mathbf{k}}(P) &= ;_{p\in P} ((p?;+p^{\mathbf{k}}) \cup (\neg p?;-p^{\mathbf{k}}))\\
\mathit{splitRW}(P) &= ;_{p\in P} ((-\mathtt{w}_{p^1};-\mathtt{r}_{p^1});(-\mathtt{w}_{p^2};-\mathtt{r}_{p^2});\\
&\qquad \textbf{if } \mathtt{w}_p \textbf{ then } ((+\mathtt{w}_{p^1};+\mathtt{r}_{p^1}) \cup (+\mathtt{w}_{p^2};+\mathtt{r}_{p^2}));\\
&\qquad \textbf{if } \mathtt{r}_p\wedge\neg\mathtt{w}_p \textbf{ then } (+\mathtt{r}_{p^1} \cup +\mathtt{r}_{p^2} \cup (+\mathtt{r}_{p^1};+\mathtt{r}_{p^2})))\\
\mathit{copybackRW}^{\mathbf{k}}(P) &= ;_{p\in P} ((\neg\mathtt{r}_{p^{\mathbf{k}}}?;-\mathtt{r}_p;-\mathtt{w}_p) \cup\\
&\qquad (\mathtt{r}_{p^{\mathbf{k}}} \wedge \neg\mathtt{w}_{p^{\mathbf{k}}}?;+\mathtt{r}_p;-\mathtt{w}_p) \cup\\
&\qquad (\mathtt{w}_{p^{\mathbf{k}}}?;+\mathtt{w}_p;+\mathtt{r}_p))\\
\mathit{mergeRW}(P) &= ;_{p\in P} (\textbf{if } \mathtt{r}_{p^1} \textbf{ then } +\mathtt{r}_p \ ; \ \textbf{if } \mathtt{w}_{p^1} \textbf{ then } +\mathtt{w}_p)
\end{aligned}$$

The useful formulas of Section 5.3 remain unchanged, except that readability and writability statements are no longer DL-PA$^{\|}$ abbreviations, but are now DL-PA propositional variables.

Given a DL-PA$^{\|}$ program or formula, its translation into DL-PA basically follows the reduction axiom for endogeneous tests $?$ and parallel composition $\|$ of Figure 4. We replace:

1. all occurrences of $\varphi?$ by $[\mathbf{;}_{p\in\mathbb{P}(\varphi)}\ (\mathbf{if}\ \neg \mathtt{r}_p\ \mathbf{then}\ (+p \cup -p))]\varphi?$
2. all occurrences of $\pi_1\|\pi_2$ by

   $$\begin{array}{l} \mathit{splitRW}(\mathbb{P}(\pi_1) \cup \mathbb{P}(\pi_2)); \\ \mathit{copybackRW}^1(\mathbb{P}(\pi_1)); \mathit{copyV}^1(\mathbb{P}(\pi_1)); \pi_1; \mathit{NochangeRW}^1(\mathbb{P}(\pi_1))?; \mathit{OkChangeV}^1(\mathbb{P}(\pi_1))?; \\ \mathit{copybackRW}^2(\mathbb{P}(\pi_2)); \mathit{copyV}^2(\mathbb{P}(\pi_2)); \pi_2; \mathit{NochangeRW}^2(\mathbb{P}(\pi_2))?; \mathit{OkChangeV}^2(\mathbb{P}(\pi_2))?; \\ \mathit{mergeRW}(\mathbb{P}(\pi_1)) \end{array}$$

3. all occurrences of $\mathtt{r}{+}p$, $\mathtt{r}{-}p$, $\mathtt{w}{+}p$, and $\mathtt{w}{-}p$ by the DL-PA programs $+\mathtt{r}_p$, $-\mathtt{r}_p$, $+\mathtt{w}_p$ and $-\mathtt{w}_p$, for $p \in \mathbb{P}$
4. all occurrences of $-\mathtt{w}_p$ by $-\mathtt{w}_p; -\mathtt{r}_p$ and all occurrences of $+\mathtt{r}_p$ by $+\mathtt{r}_p; +\mathtt{w}_p$, for $p \in \mathbb{P}$

Let $t(\varphi)$ be the translation of the DL-PA$^{\|}$ formula φ. Remember that in $t(\varphi)$, the variables $\mathtt{r}_p$ and $\mathtt{w}_p$ are considered to be arbitrary propositional variables.

Theorem 3. *A* DL-PA$^{\|}$ *formula* φ *is* DL-PA$^{\|}$*-satisfiable if and only if the* DL-PA *formula* $t(\varphi) \wedge \bigwedge_{p\in\mathbb{P}(\varphi)}(\mathtt{w}_p \to \mathtt{r}_p)$ *is* DL-PA *satisfiable.*

As the length of $t(\varphi)$ is polynomial in that of φ, it follows that DL-PA$^{\|}$ model and satisfiability checking are both PSPACE complete.

7 Conclusion

We have added to Dynamic Logic of Propositional Assignments DL-PA a parallel composition operator in the spirit of separation logics. To that end we have augmented DL-PA valuations by readability and writability information. We have adopted a stricter stance on race conditions than in the approach of [HMV19] where e.g. the program $+p\|+p$ is executable. We have provided a complete axiomatisation in terms of reduction axioms, ensuring at the same time decidability. We have also proved PSPACE complexity via a translation to DL-PA.

The mathematical properties of DL-PA$^{\|}$ compare favourably with the high complexity or even undecidability of the separation logics in the literature. They also compare favourably with those of other extensions of dynamic logic by a separating parallel composition operator that were proposed in the literature [BdFV11,VVB14,BB18]. Just as ours, the latter line of work is in the spirit of separation logic, having splitting and merging operations that are defined on system states. The axiomatisation that was introduced and studied in [BB18] is restricted to the star-free fragment and the authors had to add propositional quantifiers in order to make parallel composition definable. This contrasts

with the simplicity of our axiomatisation of DL-PA$^{\|}$ that we obtained by adding reduction axioms to the axiomatisation of DL-PA. Just as DL-PA can be viewed as an instance of PDL—the interpretation of atomic programs moves from PDL's abstract relation between states to concrete updates of valuations—, DL-PA$^{\|}$ can be viewed as an instance of the logic of [BdFV11] where the interpretation of parallel composition no longer resorts to an abstract relation $\star$ associating three states, but instead has concrete functions that split and merge valuations and that are constrained by readability and writability information.

8 Acknowledgements

The paper benefitted from comments and remarks from the reviewers as well as from the attendees of DaLí 2019, in particular Alexandru Baltag, Raul Fervari, Rainer Hähnle and Dexter Kozen. We did our best take their comments into account.

References

[BB18] Philippe Balbiani and Joseph Boudou. Iteration-free PDL with storing, recovering and parallel composition: a complete axiomatization. *J. Log. Comput.*, 28(4):705–731, 2018.

[BdFV11] Mario R. F. Benevides, Renata P. de Freitas, and Jorge Petrucio Viana. Propositional dynamic logic with storing, recovering and parallel composition. *Electr. Notes Theor. Comput. Sci.*, 269:95–107, 2011.

[BHST14] Philippe Balbiani, Andreas Herzig, François Schwarzentruber, and Nicolas Troquard. DL-PA and DCL-PC: model checking and satisfiability problem are indeed in PSPACE. *CoRR*, abs/1411.7825, 2014.

[BHT13] Philippe Balbiani, Andreas Herzig, and Nicolas Troquard. Dynamic logic of propositional assignments: a well-behaved variant of PDL. In O. Kupferman, editor, *Logic in Computer Science (LICS)*. IEEE, 2013.

[BO16] Stephen Brookes and Peter W. O'Hearn. Concurrent separation logic. *SIGLOG News*, 3(3):47–65, 2016.

[Bro04] Stephen D. Brookes. A semantics for concurrent separation logic. In Gardner and Yoshida [GY04], pages 16–34.

[Bro07] Stephen Brookes. A semantics for concurrent separation logic. *Theor. Comput. Sci.*, 375(1-3):227–270, 2007.

[BV03] Philippe Balbiani and Dimiter Vakarelov. PDL with intersection of programs: A complete axiomatization. *J. of Applied Non-Classical Logics*, 13(3-4):231–276, 2003.

[CHM+16] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, and Pierre Régnier. A simple account of multi-agent epistemic planning. In G.A. Kaminka, M. Fox, P. Bouquet, E. Hüllermeier, V. Dignum, F. Dignum, and F. van Harmelen, editors, *ECAI 2016 - 22nd Eur. Conf. on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 193–201. IOS Press, 2016.

[CS15] Tristan Charrier and François Schwarzentruber. Arbitrary public announcement logic with mental programs. In G. Weiss, P. Yolum, R.H. Bordini, and E. Elkind, editors, *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 1471–1479. ACM, 2015.

[CS17] Tristan Charrier and François Schwarzentruber. A succinct language for dynamic epistemic logic. In K. Larson, M. Winikoff, S. Das, and E.H. Durfee, editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 123–131. ACM, 2017.

[DHS05] Ádám Darvas, Reiner Hähnle, and David Sands. A theorem proving approach to analysis of secure information flow. In D. Hutter and M. Ullmann, editors, *Security in Pervasive Computing*, pages 193–209, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[FHR19] Guillaume Feuillade, Andreas Herzig, and Christos Rantsoudis. A Dynamic Logic Account of Active Integrity Constraints. *Fundamenta Informaticae*, 169(3):179–210, 2019.

[Gol92] Robert Goldblatt. Parallel action: Concurrent dynamic logic with independent modalities. *Studia Logica*, 51(3/4):551–578, 1992.

[GY04] Philippa Gardner and Nobuko Yoshida, editors. *CONCUR 2004 - Concurrency Theory, 15th Int. Conf., London, UK, August 31 - September 3, 2004, Proceedings*, volume 3170 of *Lecture Notes in Computer Science*. Springer, 2004.

[Her13] Andreas Herzig. A simple separation logic. In L. Libkin, U. Kohlenbach, and R.J.G.B. de Queiroz, editors, *WoLLIC*, volume 8071 of *Lecture Notes in Computer Science*, pages 168–178. Springer, 2013.

[Her14] Andreas Herzig. Belief change operations: a short history of nearly everything, told in dynamic logic of propositional assignments. In Chitta Baral and Giuseppe De Giacomo, editors, *Proc. KR 2014*. AAAI Press, 2014.

[HMNDBW14] Andreas Herzig, Viviane Menezes, Leliane Nunes De Barros, and Renata Wassermann. On the revision of planning tasks. In T. Schaub, editor, *European Conference on Artificial Intelligence (ECAI)*, August 2014.

[HMV19] Andreas Herzig, Frédéric Maris, and Julien Vianey. Dynamic logic of parallel propositional assignments and its applications to planning. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5576–5582. ijcai.org, 2019.

[MS96] Alain J. Mayer and Larry J. Stockmeyer. The complexity of PDL with interleaving. *Theor. Comput. Sci.*, 161(1&2):109–122, 1996.

[NGH18] Arianna Novaro, Umberto Grandi, and Andreas Herzig. Judgment aggregation in dynamic logic of propositional assignments. *J. Log. Comput.*, 28(7):1471–1498, 2018.

[O'H04] Peter W. O'Hearn. Resources, concurrency and local reasoning. In Gardner and Yoshida [GY04], pages 49–67.

[Pel87] David Peleg. Concurrent dynamic logic. *J. of the ACM*, 34(2):450–479, 1987.

[SG16] Christoph Scheben and Simon Greiner. Information flow analysis. In W. Ahrendt, B. Beckert, R. Bubel, R. Hähnle, P. H. Schmitt, and M. Ulbrich, editors, *Deductive Software Verification - The KeY Book - From Theory to Practice*, volume 10001 of *Lecture Notes in Computer Science*, pages 453–471. Springer, 2016.

[vDvdHK07] Hans P. van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*. Kluwer Academic Publishers, 2007.

[VVB14] Paulo A. S. Veloso, Sheila R. M. Veloso, and Mario R. F. Benevides. PDL for structured data: a graph-calculus approach. *Logic J. of the IGPL*, 22(5):737–757, 2014.