Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 269 (2011) 95–107
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Propositional Dynamic Logic with Storing, Recovering and Parallel Composition

Mario R. F. Benevides[1,2]

Computer Science Department and Systems and Computer Engineering Program
Federal University of Rio de Janeiro
Brazil

Renata de Freitas[1,3] Petrucio Viana[1,4]

Institute of Mathematics
Fluminense Federal University
Brazil

Abstract

This work extends Propositional Dynamic Logic (PDL) with parallel composition operator and four atomic programs which formalize the storing and recovering of elements in data structures. A generalization of Kripke semantics is proposed that instead of using set of possible states it uses structured sets of possible states. This new semantics allows for representing data structures and using the five new operator one is capable of reasoning about the manipulation of these data structures. The use of the new language (PRSPDL) is illustrated with some examples. We present sound and complete set of axiom schemata and inference rules to prove all the valid formulas for a restricted fragment called *RSPDLo*.

Keywords: Propositional Dynamic Logic, Parallel Composition, Modal Logic for Program Specification, Concurrency

1 Introduction and Motivation

Propositional Dynamic Logic (PDL) [7,14] plays an important role in formal specification and reasoning about sequential programs and systems. It has been used to describe and verify properties as correctness, termination, fairness, liveness and equivalence of programs.

PDL is a multi-modal logic with one modality $\langle \pi \rangle$ for each program π. The logic has a set of basic programs and a set of operators (sequential composition,

[1] The author was partially supported by grants from CNPq, Capes and FAPERJ

[2] Email: mario@cos.ufrj.br

[3] Email: naborges@cos.ufrj.br

[4] Email: petrucio@cos.ufrj.br

1571-0661 © 2011 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2011.03.008

nondeterministic choice, test and iteration) that are used to inductively build the set of non-basic programs. A Kripke semantics can be provided, with a frame $\mathcal{F} = (W, R_\pi)$, where W is a non-empty set of possible program states and, for each program π, R_π is a binary relation on W such that $(s, t) \in R_\pi$ if and only if there is a computation of π starting in s and terminating in t.

In modal logics in general and, consequently, in PDL an state has no internal structure, in the sense that its possible constituents play no role in the process of evaluating a formula in that state. In the last decade, many logical formalisms have been proposed to cope with mutable data structures and updates. Separation Logic [17,21] was proposed to reasoning about imperatives programs with shared mutable data structures, i.e structures with fields that can be updated and referenced in different points of its execution. An interesting extension of this logic was proposed in [15] to deal with concurrency. Moreover, in the field of Epistemic Logic, many formalism have been proposed to deal with the dynamics of knowledge in situations like, agent based systems, games and social networks. Logics like Dynamic Epistemic Logic DEL [11] and Public Announcement Logic PAL [20] are examples. These logics must deal with updates and changes of knowledge as actions are performed by the agents or by the environment.

Another weakness PDL suffers is the lack of operators for the treatment of parallelism and concurrence of programs. There are many extensions of PDL to deal with this kind of operators [19,18,2,3,12,16,1,6]. The aim of all these logics is to reasoning about parallel or concurrent programs.

In this work — although it is not our main concern to reasoning about parallel execution of programs — we stay close to the above traditions, proposing an extension of the PDL regular language with a parallel operator and four operators: two to store data and two to recover data. Our language, which we call PRSPDL, is endowed with a semantics based on structured sets, in which the parallel operator together with the projections can be used to represent and manipulate data structures. This semantics is a generalization of Kripke semantics which instead of using a set of possible states uses a structured set of states. The idea of providing structure to a set was inspired in fork algebras [13,9] and has been used in many formalisms [8,22,10].

We start the study of PRSPDL by presenting the system, exemplify its expressive power, and show a completeness result for a fragment of it.

The paper is structured as follows. Section 2 presents the basics on syntax and semantics of the PRSPDL language. Section 3, presents some programs written in PRSPDL language and provide some motivations. Section 4 presents axioms and rules as well as soundness for the restricted system, RSPDL^0, obtained from PRSPDL by excluding the iteration, parallel composition and the test operators. Section 5 presents a completeness result for RSPDL^0. Finally, section 6 contains some discussion about our contribution and some future works.

2 Syntax and semantics

The language is the usual PDL language with composition, choice, iteration, and test operators added with four atomic programs s_1, s_2, r_1, r_2 and a binary operator of parallel composition. Intuitively, the semantics of these new operators is as follows. It is important to notice that ours states are "ordered pairs". The intended meaning of the nondeterministc program s_1, called *store first*, is to store the current state as the first component of the resulting state, i.e., when the program s_1 runs at state s it finishes its running producing a new state (s, t) whose first coordinate is s. Analogously, s_2, called *store second*, stores the current state s as a second coordinate of the resulting state (t, s). The intended meaning of the deterministc program r_1, called *recover first*, is to recover a data (state) that is stored at the first component of the current state. When the program r_1 runs at state (s, t) it finishes its running at (a recovered) state s, Analogously, the program r_2, called *recover second*, recovers the second component of the current state (s, t) and finishes at state t. Finally, when program $\pi_1 \parallel \pi_2$ runs at state (s_1, t_1), its effect is to run π_1 and π_2 in parallel at state s_1 and t_1 respectively, yielding a new state (s_2, t_2).

Formally, we have the following definitions:

Definition 2.1 Let $\mathcal{A}ct = \{a, b, c, \ldots\}$ be the set of basic programs, typically denoted by α. The PRSPDL *programs*, typically denoted π, are defined as follows:

$$\pi ::= \alpha \mid \pi_1; \pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \mid \pi_1 \parallel \pi_2 \mid ?\phi \mid \mathsf{s}_1 \mid \mathsf{s}_2 \mid \mathsf{r}_1 \mid \mathsf{r}_2.$$

Definition 2.2 The *dynamic modal language with parallel composition, storing and recovering* (PRSPDL) is a multi-modal language consisting of a set Φ of countably many propositional symbols, typically denoted by $p, q, r, \ldots$, the boolean connectives $\neg$ and $\wedge$, and a family of modal operators $\{\langle\pi\rangle : \pi \text{ is a PRSPDL program}\}$. The PRSPDL *formulas*, typically denoted ϕ, are defined as follows:

$$\phi ::= \bot \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle\pi\rangle\phi.$$

Definition 2.3 A *frame* is a pair $\mathcal{F} = (W, \{R_\pi : \pi \text{ is a program}\})$, where:

– W is a non-empty set,

– $R_\pi \subseteq W \times W$, for each program π.

Definition 2.4 A *model* is a pair $\mathcal{M} = (\mathcal{F}, V)$, where $\mathcal{F}$ is a frame and $V : \Phi \to 2^W$ is a valuation function mapping proposition symbols into subsets of W.

Definition 2.5 A model is *standard* when it satisfies the following conditions:

– $R_{\pi_1;\pi_2} = R_{\pi_1}; R_{\pi_2}$,

– $R_{\pi_1 \cup \pi_2} = R_{\pi_1} \cup R_{\pi_2}$,

– $R_{\pi^*} = (R_\pi)^*$,

– $R_{?\phi} = \{(w, w) \in W^2 : \mathcal{M}, w \models \phi\}$.

The main semantical difference between PDL and PRSPDL is that in the later formulas are interpreted on sets of structured states [9].

Definition 2.6 A set of structured states is a triple $(S, E, \star)$ where S is a non-empty set, E is an equivalence relation on S, and $\star : S^2 \to S$ is injective, i.e., a binary operation satisfying

$$s_1 \star s_2 = t_1 \star t_2 \text{ iff } s_1 = t_1 \text{ and } s_2 = t_2,$$

for every $(s_1, s_2), (t_1, t_2) \in E$.

Definition 2.7 A *structured frame* is a pair

$$\mathcal{F} = ((S, E, \star), \{R_\pi : \pi \text{ is a program}\}),$$

where:

- $(S, E, \star)$ is a non-empty set of structured states,
- $R_\pi \subseteq E$, for each program π,
- $(S, \{R_\pi : \pi \text{ is a program}\})$ is a frame.

A *structured model* is model based on a structured frame.

Definition 2.8 A structured frame $\mathcal{F}$ is *proper* when it satisfies the following conditions:

- $R_{\mathsf{s}_1} = \{(s, s \star t) : s, t \in S\}$,
- $R_{\mathsf{s}_2} = \{(t, s \star t) : s, t \in S\}$,
- $R_{\mathsf{r}_1} = \{(s \star t, s) : s, t \in S\}$,
- $R_{\mathsf{r}_2} = \{(s \star t, t) : s, t \in S\}$,
- $R_{\pi_1 \| \pi_2} = \{(s_1 \star t_1, s_2 \star t_2) : s_1, t_1, s_2, t_2 \in S \textit{ and } (s_1, s_2) \in R_{\pi_1} \textit{ and } (t_1, t_2) \in R_{\pi_2}\}$.

A structured model is *proper* when it is based on a proper structured frame.

Definition 2.9 An PRSPDL *model* is a proper standard model.

Observe that, in proper standard frames, the relations R_{s_1} and R_{r_1} are converse of each other. A similar remark applies to the relations R_{s_2} and R_{r_2}. Besides, in proper standard frames, the following properties are true, where $I_S = \{(s, s) : s \in S\}$ is the identity relation on S:

$$R_{\mathsf{s}_1}; R_{\mathsf{r}_1} = I_S \tag{1}$$

$$R_{\mathsf{s}_2}; R_{\mathsf{r}_2} = I_S \tag{2}$$

$$R_{\mathsf{s}_1}; R_{\mathsf{r}_2} = E \tag{3}$$

$$(R_{\mathsf{r}_1}; R_{\mathsf{s}_1}) \cap (R_{\mathsf{r}_2}; R_{\mathsf{s}_2}) \subseteq I_S \tag{4}$$

$$R_{\mathsf{r}_1}; E = R_{\mathsf{r}_2}; E \tag{5}$$

It is important to notice that there are other properties that hold in proper standard frames, but the ones listed above are used in the proofs in the rest of the paper.

Definition 2.10 Let $\mathcal{M}$ be a model. The notion of *satisfaction* of a formula ϕ in a model $\mathcal{M}$ at a state s, notation $\mathcal{M}, s \models \phi$ is inductively defined as follows:

(i) $\mathcal{M}, s \not\models \bot$,

(ii) $\mathcal{M}, s \models p$ iff $s \in V(p)$,

(iii) $\mathcal{M}, s \models \neg\phi$ iff $\mathcal{M}, s \not\models \phi$,

(iv) $\mathcal{M}, s \models \phi \wedge \psi$ iff $\mathcal{M}, s \models \phi$ and $\mathcal{M}, s \models \psi$,

(v) $\mathcal{M}, s \models \langle\pi\rangle\phi$ iff there exists $t \in S$, $sR_\pi t$ and $\mathcal{M}, t \models \phi$.

The interpretations of the constant atomic programs $\mathsf{s}_1, \mathsf{s}_2, \mathsf{r}_1$ and r_2 and parallel composition on proper standard models are as expected:

- $\mathcal{M}, s \models \langle\mathsf{s}_1\rangle\varphi$ iff there is $t \in S$ such that $sR_{\mathsf{s}_1}t$ and $\mathcal{M}, t \models \varphi$ iff there are $t, s_2 \in S$ such that $t = s \star s_2$, and $\mathcal{M}, t \models \varphi$ iff there is $s_2 \in S$ such that $\mathcal{M}, s \star s_2 \models \varphi$. In another words, $\mathcal{M}, s \models \langle\mathsf{s}_1\rangle\varphi$ iff s is the first coordinate of an element standing for an ordered pair in which φ is true.
- $\mathcal{M}, s \models \langle\mathsf{s}_2\rangle\varphi$ iff there is $t \in S$ such that $sR_{\mathsf{s}_2}t$ and $\mathcal{M}, t \models \varphi$ iff there are $t, s_1 \in S$ such that $t = s_1 \star s$, and $\mathcal{M}, t \models \varphi$ iff there is $s_1 \in S$ such that $\mathcal{M}, s_1 \star s \models \varphi$. In another words, $\mathcal{M}, s \models \langle\mathsf{s}_2\rangle\varphi$ iff s is the second coordinate of an element standing for an ordered pair in which φ is true.
- $\mathcal{M}, s \models \langle\mathsf{r}_1\rangle\varphi$ iff there is $t \in S$ such that $sR_{\mathsf{r}_1}t$ and $\mathcal{M}, t \models \varphi$ iff there are $t, s_1, s_2 \in S$ such that $s = s_1 \star s_2$, $t = s_1$ and $\mathcal{M}, s_1 \models \varphi$ iff there are $s_1, s_2 \in S$ such that $s = s_1 \star s_2$ and $\mathcal{M}, s_1 \models \varphi$. In another words, $\mathcal{M}, s \models \langle\mathsf{r}_1\rangle\varphi$ iff s stands for an ordered pair in whose first coordinate φ is true.
- $\mathcal{M}, s \models \langle\mathsf{r}_2\rangle\varphi$ iff there is $t \in S$ such that $sR_{\mathsf{r}_2}t$ and $\mathcal{M}, t \models \varphi$ iff there are $t, s_1, s_2 \in S$ such that $s = s_1 \star s_2$, $t = s_2$ and $\mathcal{M}, s_2 \models \varphi$ iff there are $s_1, s_2 \in S$ such that $s = s_1 \star s_2$ and $\mathcal{M}, s_2 \models \varphi$. In another words, $\mathcal{M}, s \models \langle\mathsf{r}_2\rangle\varphi$ iff s stands for an ordered pair in whose second coordinate φ is true.
- $\mathcal{M}, s \models \langle\pi_1 \parallel \pi_2\rangle\varphi$ iff there is $t \in S$ such that $sR_{\pi_1\parallel\pi_2}t$ and $\mathcal{M}, t \models \varphi$ iff there are $s_1, s_2, t_1, t_2 \in S$ such that $s = (s = s_1 \star s_2)$ and $t = (t_1 \star t_2)$ and $s_1R_{\pi_1}s_2$ and $t_1R_{\pi_2}t_2$, and $\mathcal{M}, t \models \varphi$. In another words, $\mathcal{M}, s \models \langle\pi_1 \parallel \pi_2\rangle\varphi$ iff programs π_1 and π_2 are executed in parallel in s and reach a state t where φ holds.

By applying the PRSPDL operators to these basic programs we can define some new useful operators. For instance, we can define the operators in [4,5] which are used in diagrammatic reasoning based on allegories. We also leave this matter for further investigation.

If $\mathcal{M}, w \Vdash \varphi$ for every state w, we say that φ is *globally satisfied* in the model $\mathcal{M}$, notation $\mathcal{M} \Vdash \varphi$. If φ is globally satisfied in all models $\mathcal{M}$ of a frame $\mathcal{F}$, we say that φ is *valid* in $\mathcal{F}$, notation $\mathcal{F} \Vdash \varphi$. Finally, if φ is valid in all frames, we say that φ is valid, notation $\Vdash \varphi$. Two formulas φ and ψ are *semantically equivalent* if $\Vdash \varphi \leftrightarrow \psi$.

3 Examples

In order to illustrate the usage of the PRSPDL language we present four examples. In all of them, we take advantage of the operations of storing and recovering, to store some data and then recover this data during the computation. One powerful mechanism is the combination of these operations of store/recover with test, it allows for reasoning about properties that holds at previous states in the computation and use this information at the current state. In what follows, we abbreviate $?\neg\bot$ as 1.

3.1 Example 1

In this example, we present a program π_1 which when start to run on input u, stores the initial state u at the second coordinate of an ordered pair, then executes actions α and β over the first coordinate of the pair, successively and, after that, returns to the initial state by restoring the second coordinate. This sequence of actions is displayed at the following diagram:

$$\begin{tikzcd}
u \arrow[r, "\mathsf{s}_2"] & (v_0, u) \arrow[r, "\alpha \| \mathsf{1}"] & (v_1, u) \arrow[r, "\beta \| \mathsf{1}"] & (v_2, u) \arrow[lll, bend left, "\mathsf{r}_2"]
\end{tikzcd}$$

When written as a PRSPDL program, π_1 can be specified as:

$$\pi_1 \equiv \mathsf{s}_2; (\alpha \parallel \mathsf{1}); (\beta \parallel \mathsf{1}); \mathsf{r}_2.$$

3.2 Example 2

In this example, we present a program π_2 which when start to run on input u, stores the initial state u at the second coordinate of an ordered pair, then executes action α on the first coordinate of the current pair, until property ϕ is true, then after that, returns to the initial state by restoring the second coordinate of the current pair.

$$\begin{tikzcd}
u \arrow[r, "\mathsf{s}_2"] & (v_0, u) \arrow[r, "(\neg\phi?;\alpha \| \mathsf{1})^\star"] & (v_1, u) \arrow[r, "\phi? \| \mathsf{1}"] & (w, u) \arrow[lll, bend left, "\mathsf{r}_2"]
\end{tikzcd}$$

When written as a PRSPDL program, π_2 can be specified as:

$$\pi_2 \equiv \mathsf{s}_2; (\neg\phi?; \alpha \parallel \mathsf{1})^\star; (\phi? \parallel \mathsf{1}); \mathsf{r}_2.$$

3.3 Example 3

In this example, we present a program π_3 which when start to run on input u, stores the initial state u at second coordinate of an ordered pair, then executes action α over the first coordinate of the pair and, after that, if property ϕ is true at the initial state then it performs action β over the first coordinate of the current pair, else it performs action γ over it. It is important to notice that formula ϕ is tested at the initial state and not at current state.

$$\begin{tikzcd}
u \arrow[r, "\mathsf{s}_2"] & (v_0, u) \arrow[r, "\alpha \| \mathsf{1}"] & (v_1, u) \arrow[r, "(\langle \mathsf{r}_2;\phi?\rangle\top)?"] \arrow[d, "(\langle \mathsf{r}_2;\neg\phi?\rangle\top)?"] & (v_1, u) \arrow[r, "\beta \| \mathsf{1}"] & (v_2, u) \\
 & & (v_1, u) \arrow[r, "\gamma \| \mathsf{1}"] & (v_3, u) &
\end{tikzcd}$$

When written as a PRSPDL program, π_3 can be specified as:

$$\pi_3 \equiv \mathsf{s}_2; (\alpha \parallel 1); (((\langle \mathsf{r}_2; \phi? \rangle \top)?; \beta \parallel 1) \cup ((\langle \mathsf{r}_2; \neg\phi? \rangle \top)?; (\gamma \parallel 1).$$

3.4 Example 4

In this example, we present a program π_4 which when start to run on input u, stores the initial state u at second coordinate of an ordered pair, then executes action α over the first coordinate of the pair and, after that, it either stores the current state as the second coordinate of an ordered pair, executes action α over the new first coordinate, and returns to the second pair obtained in the computation; or executes action β over the first coordinate of the pair and returns to the initial state.

$$\begin{tikzcd}
u \arrow[r, "\mathsf{s}_2"] & (v_0, u) \arrow[r, "\alpha\|1"] & (v_1, u) \arrow[r, "\mathsf{s}_2"] \arrow[d, "\beta\|1"] & (v_2, (v_1, u)) \arrow[r, "\alpha\|1"] & (v_3, (v_1, u)) \arrow[ll, bend left, "\mathsf{r}_2"] \\
& & (v_4, u) \arrow[ull, "\mathsf{r}_2"] & &
\end{tikzcd}$$

When written as a PRSPDL program, π_4 can be specified as:

$$\pi_4 \equiv \mathsf{s}_2; (\alpha \parallel 1); ((\mathsf{s}_2; (\alpha \parallel 1); \mathsf{r}_2) \cup (\beta \parallel 1)); \mathsf{r}_2.$$

4 Axiomatic System for RSPDL^0

In this section, we restrict the language presented in Section 2 to a fragment called RSPDL^0. In this fragment, we do not allow the use of the operators of test (?), iteration ($\star$) and parallel composition ($\|$). We intend to use the work reported in this fragment as a basis for the investigation of the whole language.

Our objective is to present a set of axioms for RSPDL^0 and prove soundness and completeness for it with respect to the semantics on structured sets. We use the standard Boolean abbreviations $\top$, $\vee$, $\rightarrow$ and $\leftrightarrow$, and the modal abbreviations $[\pi]\phi := \neg\langle\pi\rangle\neg\phi$, for every program π.

Let RSPDL^0 be the modal logic defined by the schemata and rules in Table 1.

Axiom 2 is the standard K axiom of distributivity. Axioms 3 and 4 are the standard PDL axioms for composition and non-deterministic choice, respectively. Axiom 5 expresses that the relations R_{r_1} and R_{r_2}, interpreting r_1 and r_2, respectively, are functional. Axiom 6 is the standard temporal axiom expressing that the relations R_{s_1} and R_{r_1} (also R_{s_2} and R_{r_2}), interpreting s_1 and r_1 (also s_2 and r_2), respectively, are the converse of each other. Axiom 7 expresses that the relations R_{r_1} and R_{r_2}, interpreting r_1 and r_2, respectively, have the same domain. Axiom 8 warrants unicity of ordered pairs. Axioms 9, 10 and 11, in conjunct, express that the relation $R_{\mathsf{s}_1;\mathsf{r}_2}$, interpreting the composite program $\mathsf{s}_1; \mathsf{r}_2$, is an equivalence relation on its field.

Theorem 4.1 (Soundness) *If* $\vdash \varphi$, *then* φ *is valid in all* RSPDL^0 *frames.*

Proof. The proof of the soundness of the first four axioms are usual.

Axioms

1. *All tautologies*

2. $[\pi](\varphi \to \psi) \to ([\pi]\varphi \to [\pi]\psi)$

3. $[\pi_1;\pi_2]\varphi \leftrightarrow [\pi_1][\pi_2]\varphi$

4. $[\pi_1 \cup \pi_2]\varphi \leftrightarrow [\pi_1]\varphi \wedge [\pi_2]\varphi$

5. $\langle \mathsf{r}_1 \rangle\varphi \to [\mathsf{r}_1]\varphi$

$\langle \mathsf{r}_2 \rangle\varphi \to [\mathsf{r}_2]\varphi$

6. $\varphi \to [\mathsf{s}_1]\langle \mathsf{r}_1 \rangle\varphi$

$\varphi \to [\mathsf{r}_1]\langle \mathsf{s}_1 \rangle\varphi$

$\varphi \to [\mathsf{s}_2]\langle \mathsf{r}_2 \rangle\varphi$

$\varphi \to [\mathsf{r}_2]\langle \mathsf{s}_2 \rangle\varphi$

7. $\langle \mathsf{r}_1 \rangle\top \leftrightarrow \langle \mathsf{r}_2 \rangle\top$

$\langle \mathsf{s}_1 \rangle\top \leftrightarrow \langle \mathsf{s}_2 \rangle\top$

8. $\langle \mathsf{s}_1;\mathsf{r}_1 \rangle\varphi \to [\mathsf{s}_1;\mathsf{r}_1]\varphi$

$\langle \mathsf{s}_2;\mathsf{r}_2 \rangle\varphi \to [\mathsf{s}_2;\mathsf{r}_2]\varphi$

9. $[\mathsf{s}_1;\mathsf{r}_2]\varphi \to \varphi$

10. $\varphi \to [\mathsf{s}_1;\mathsf{r}_2]\langle \mathsf{s}_1;\mathsf{r}_2 \rangle\varphi$

11. $[\mathsf{s}_1;\mathsf{r}_2]\varphi \to [\mathsf{s}_1;\mathsf{r}_2][\mathsf{s}_1;\mathsf{r}_2]\varphi$

Inference Rules

$$\mathsf{MP)}\quad \frac{\phi \qquad \phi \to \psi}{\psi}$$

$$\mathsf{Nec}_\pi\mathsf{)}\quad \frac{\phi}{[\pi]\phi}$$

Table 1
RSPDL^0 axiomatics.

Axiom 5. We treat just the instances relative to r_1. The instances relative to r_2 can be treated in a similar way.

Suppose $\mathcal{M}, s \models \langle \mathsf{r}_1 \rangle \varphi$. So, there are s_1 and s_2 such that $s = s_1 \star s_2$ and $\mathcal{F}, V, s_1 \models \varphi$. Let $u \in S$ be any state for which $sR_{\mathsf{r}_1}u$. By definition, there are s_1' and s_2' such that $s = s_1' \star s_2'$ and $u = s_1'$. From $s = s_1 \star s_2 = s_1' \star s_2'$ and the injectivity of $\star$, we have $s_1 = s_1'$. Since, $\mathcal{M}, s_1 \models \varphi$ and $u = s_1' = s_1$, we have $\mathcal{M}, u \models \varphi$. So, we can conclude $\mathcal{M}, u \models \varphi$, for every $u \in S$ such that $sR_{\mathsf{r}_1}u$, i.e., $\mathcal{M}, s \models [\mathsf{s}_1]\varphi$.

Axiom 6. We treat just the instances relative to s_1 and r_1. The instances relative to s_2 and r_2 can be treated in a similar way.

Suppose $\mathcal{M}, s \models \varphi$. Let t be such that $sR_{\mathsf{s}_1}t$. Hence, there is a $s_2 \in S$ such that $t = s \star s_2$. So, there are $s_1', s_2' \in S$ such that $t = s_1' \star s_2'$ and $\mathcal{M}, s_1' \models \varphi$, which is the same as $\mathcal{M}, t \models \langle \mathsf{r}_1 \rangle \varphi$. So, we can conclude $\mathcal{M}, t \models \langle \mathsf{r}_1 \rangle \varphi$, for every $t \in S$ such that $sR_{\mathsf{s}_1}t$, i,e., $\mathcal{M}, s \models [\mathsf{s}_1]\langle \mathsf{r}_1 \rangle \varphi$.

Now, suppose again that $\mathcal{M}, s \models \varphi$ and let t be such that $sR_{\mathsf{r}_1}t$. Hence, there are $s_1, s_2 \in S$ such that $s = s_1 \star s_2$ and $t = s_1$. By definition, $s_1 R_{\mathsf{s}_1} s_1 \star s_2$. From $\mathcal{M}, s \models \varphi$ and $s = s_1 \star s_2$, we have $\mathcal{M}, s_1 \star s_2 \models \varphi$. So, there is a $u \in S$ such that $tR_{\mathsf{s}_1}u$ and $\mathcal{M}, u \models \varphi$, which is the same as $\mathcal{M}, t \models \langle \mathsf{s}_1 \rangle \varphi$. So, we can conclude $\mathcal{M}, t \models \langle \mathsf{s}_1 \rangle \varphi$, for every $t \in S$ such that $sR_{\mathsf{r}_1}t$, i,e., $\mathcal{M}, s \models [\mathsf{r}_1]\langle \mathsf{s}_1 \rangle \varphi$.

Axiom 7. We treat just the instance relative to r_1 and r_2. The instance relative to s_1 and s_2 can be treated in a similar way.

We have that $\mathcal{M}, s \models \langle \mathsf{r}_1 \rangle \top$ iff there is some $t \in S$ such that $sR_{\mathsf{r}_1}t$ and $\mathcal{M}, t \models \top$, iff there is some $t \in S$ such that $sR_{\mathsf{r}_1}t$, iff there are $t, t' \in S$ such that $s = t \star t'$, iff there is some $t' \in S$ such that $sR_{\mathsf{r}_2}t'$, iff there is some $t' \in S$ such that $sR_{\mathsf{r}_2}t'$ and $\mathcal{M}, t' \models \top$, iff $\mathcal{M}, s \models \langle \mathsf{r}_2 \rangle \top$.

Axiom 8. We treat just the instance relative to s_1 and r_1. The instance relative to s_2 and r_2 can be treated in a similar way.

Suppose $\mathcal{M}, s \models \langle \mathsf{s}_1; \mathsf{r}_1 \rangle \varphi$. Hence, there are $u, v \in S$ such that $sR_{\mathsf{s}_1}uR_{\mathsf{r}_1}v$ and $\mathcal{M}, v \models \varphi$. Hence, there are $u, v, v', s' \in S$ such that $u = s \star s'$, $u = v \star v'$, and $\mathcal{M}, v \models \varphi$. By the injectivity of $\star$, we have $s = v$. Hence, $\mathcal{M}, s \models \varphi$. Now, let $t \in S$ be such that $sR_{\mathsf{s}_1;\mathsf{r}_1}t$. Hence, there are $u, t', s' \in S$ such that $u = s \star s'$ and $u = t \star t'$. By the injectivity of $\star$, we have $s = t$. Hence, $\mathcal{M}, t \models \varphi$.

Axiom 9. Suppose $\mathcal{M}, s \models [\mathsf{s}_1; \mathsf{r}_2]\varphi$. Hence, for every $t \in S$ if $sR_{\mathsf{s}_1;\mathsf{r}_2}t$, then $\mathcal{M}, t \models \varphi$. Now, since we have $sR_{\mathsf{s}_1}s \star s$ and $s \star sR_{\mathsf{r}_2}s$, we have $sR_{\mathsf{s}_1;\mathsf{r}_2}s$. So, we conclude $\mathcal{M}, s \models \varphi$.

Axiom 10. Suppose $\mathcal{M}, s \models \varphi$. Let $t \in S$ be such that $sR_{\mathsf{s}_1;\mathsf{r}_2}t$. Now, since we have $tR_{\mathsf{s}_1}t \star s$ and $t \star sR_{\mathsf{r}_2}s$, we have $tR_{\mathsf{s}_1;\mathsf{r}_2}s$. This, together with $\mathcal{M}, s \models \varphi$, gives us $\mathcal{M}, t \models \langle \mathsf{s}_1; \mathsf{r}_2 \rangle \varphi$. So, we can conclude that $\mathcal{M}, t \models \langle \mathsf{s}_1; \mathsf{r}_2 \rangle \varphi$, for every $t \in S$ such that $sR_{\mathsf{s}_1;\mathsf{r}_2}t$, i,e. $\mathcal{M}, s \models [\mathsf{s}_1; \mathsf{r}_2]\langle \mathsf{s}_1; \mathsf{r}_2 \rangle \varphi$.

Axiom 11. Suppose $\mathcal{M}, s \models [\mathsf{s}_1; \mathsf{r}_2]\varphi$. Hence, for every $t \in S$ if $sR_{\mathsf{s}_1;\mathsf{r}_2}t$, then $\mathcal{M}, t \models \varphi$. Let $u, v \in S$ be such that $sR_{\mathsf{s}_1;\mathsf{r}_2}u$ and $uR_{\mathsf{s}_1;\mathsf{r}_2}v$. Since we have $sR_{\mathsf{s}_1}s \star v$ and $s \star vR_{\mathsf{r}_2}v$, we have $sR_{\mathsf{s}_1;\mathsf{r}_2}v$, which gives us $\mathcal{M}, v \models \varphi$. Hence, we have $\mathcal{M}, v \models \varphi$ for every $v \in S$ such that $uR_{\mathsf{s}_1;\mathsf{r}_2}v$, i.e., $\mathcal{M}, u \models [\mathsf{s}_1; \mathsf{r}_2]\varphi$. Moreover, we have $\mathcal{M}, u \models [\mathsf{s}_1; \mathsf{r}_2]\varphi$ for every $u \in S$ such that $sR_{\mathsf{s}_1;\mathsf{r}_2}u$, i.e. $\mathcal{M}, s \models [\mathsf{s}_1; \mathsf{r}_2][\mathsf{s}_1; \mathsf{r}_2]\varphi$, as required. □

5 Completeness of RSPDL^0

System RSPDL^0 is the restriction of PRSPDL obtained by the exclusion of the iteration operator $*$, the test operator and the parallel composition operator. A proof system for RSPDL^0 was presented in Section 4. In this section, we prove the completeness of this proof system for RSPDL^0.

Theorem 5.1 *If $\nvdash_{\mathrm{RSPDL}^0} \varphi$, then there is a model in which φ is not valid.*

Proof. The *canonical model* is the structure

$$\mathcal{M}^c = (W^c, \{R^c_\pi : \pi \text{ is a program}\}, V^c),$$

defined as usual:

- W^c is the set of all maximal consistent sets of formulas,
- $sR^c_\pi t$ iff formula φ is in t for every formula $[\pi]\varphi$ in s,
- $V^c p$ is the set of all maximal consistent sets of formulas containing p.

The *canonical frame* is the structure $\mathcal{F}^c = (W^c, \{R^c_\pi : \pi \text{ is a program}\})$.

Observe that we do not have neither that the canonical frame nor the canonical model is a proper frame or model. This is because we do not have that W^c is a structured set. Anyway, by a standard modal logic reasoning, we have:

Lemma 5.2 *If a formula φ is such that $\nvdash \varphi$, then there is some state w in the canonical model $\mathcal{M}^c$ such that $\mathcal{M}^c, w \not\models \varphi$.*

Axioms 3 and 4 warrant that the canonical model is standard.

Lemma 5.3 *$\mathcal{M}^c$ is standard.*

Proof. First, we shall prove $R^c_{\pi_1;\pi_2} = R^c_{\pi_1}; R^c_{\pi_2}$.

To prove the inclusion from left to right, suppose Σ and Σ' are MCS satisfying $\Sigma R^c_{\pi_1;\pi_2}\Sigma'$. First, we prove that the set of formulas $\Gamma = \{\varphi : [\pi_1]\varphi \in \Sigma\} \cup \{\neg[\pi_2]\psi : \psi \notin \Sigma'\}$ is consistent. In fact, if $\Gamma \vdash \bot$, then there are $\varphi_1, \ldots, \varphi_n$ such that $[\pi_1]\varphi_1, \ldots, [\pi_1]\varphi_n \in \Sigma$ and there are $\psi_1, \ldots, \psi_m \notin \Sigma'$ such that $\vdash \varphi_1 \wedge \cdots \wedge \varphi_n \wedge \neg[\pi_2]\psi_1 \wedge \cdots \wedge \neg[\pi_2]\psi_m \to \bot$. By normality, we obtain $\vdash \varphi_1 \wedge \cdots \wedge \varphi_n \to ([\pi_2]\psi_1 \vee \cdots \vee [\pi_2]\psi_m)$. Hence, $\vdash [\pi_1]\varphi_1 \wedge \cdots \wedge [\pi_1]\varphi_n \to [\pi_1]([\pi_2]\psi_1 \vee \cdots \vee [\pi_2]\psi_m)$. So, $\Sigma \vdash [\pi_1][\pi_2]\psi_1 \vee \cdots \vee [\pi_1][\pi_2]\psi_m$. Now, by applying Axiom 3, we have $\Sigma \vdash [\pi_1;\pi_2]\psi_1 \vee \cdots \vee [\pi_1;\pi_2]\psi_m$, which, since $\Sigma R^c_{\pi_1;\pi_2}\Sigma'$, implies $\psi_1 \in \Sigma'$ or $\ldots$ or $\psi_m \in \Sigma'$, a contradiction.

Now, let Σ'' be a maximal consistent set such that $\Gamma \subseteq \Sigma''$. For each φ such that $[\pi_1]\varphi \in \Sigma$, we have $\varphi \in \Gamma \subseteq \Sigma''$. Hence, $\Sigma R^c_{\pi_1}\Sigma''$. Besides, let $\psi \in \Sigma'$, that is, $\neg\psi \notin \Sigma'$. We have $\neg[\pi_2]\neg\psi \in \Gamma \subseteq \Sigma''$, which is the same as $\langle\pi_2\rangle\psi \in \Sigma''$. Hence, $\Sigma'' R^c_{\pi_2}\Sigma'$. From these, we obtain $\Sigma R^c_{\pi_1}; R^c_{\pi_2}\Sigma'$.

To prove the other inclusion, let Σ and Σ' be MCS satisfying $\Sigma R^c_{\pi_1}; R^c_{\pi_2}\Sigma'$. Let Σ'' be a MCS such that $\Sigma R^c_{\pi_1}\Sigma''$ and $\Sigma'' R^c_{\pi_2}\Sigma'$. Let $\varphi \in \Sigma'$. Hence, $\langle\pi_2\rangle\varphi \in \Sigma''$, and so $\langle\pi_1\rangle\langle\pi_2\rangle\varphi \in \Sigma$. Now, by Axiom 3, we have $\langle\pi_1;\pi_2\rangle\varphi \in \Sigma$ which proves $\Sigma R^c_{\pi_1;\pi_2}\Sigma'$.

The proof that $R^c_{\pi_1 \cup \pi_2} = R^c_{\pi_1} \cup R^c_{\pi_2}$ is trivial, using Axiom 4. □

Axioms 5–11 warrant that relations have the required properties to make W^c a structured set.

Lemma 5.4 (i) *Relations $R^c_{\mathsf{r}_1}$ and $R^c_{\mathsf{r}_2}$ are functional.*

(ii) *Relations $(R^c_{\mathsf{r}_1})^{-1}; R^c_{\mathsf{r}_1}$ and $(R^c_{\mathsf{r}_2})^{-1}; R^c_{\mathsf{r}_2}$ are injective.*

(iii) *Relations $R^c_{\mathsf{r}_1}$ and $R^c_{\mathsf{r}_2}$ have the same domain.*

(iv) *Relations $R^{s_\varphi}_{\mathsf{s}_1}$ and $R^{s_\varphi}_{\mathsf{r}_1}$ are the converse of each other. Also, relations $R^{s_\varphi}_{\mathsf{s}_2}$ and $R^{s_\varphi}_{\mathsf{r}_2}$ are the converse of each other.*

Proof. To prove $R^c_{\mathsf{r}_1}$ is functional, we proceed as follows. Suppose Σ, Σ_1, and Σ_2 are MCSs satisfying $\Sigma R^c_{\mathsf{r}_1} \Sigma_1$ and $\Sigma R^c_{\mathsf{r}_1} \Sigma_2$. Let $\phi \in \Sigma_1$. Since $\Sigma R^c_{\mathsf{r}_1} \Sigma_1$, we have $\langle \mathsf{r}_1 \rangle \phi \in \Sigma$. Now, by applying Axiom 5, we obtain $[\mathsf{r}_1]\phi \in \Sigma$, and since $\Sigma R^c_{\mathsf{r}_1} \Sigma_2$, we have $\phi \in \Sigma_2$. Hence, $\Sigma_1 \subseteq \Sigma_2$. The inclusion $\Sigma_2 \subseteq \Sigma_1$ can be proved analogously.

The proof that $R^c_{\mathsf{r}_2}$ is functional is entirely similar.

To prove $(R^c_{\mathsf{r}_1})^{-1}; R^c_{\mathsf{r}_1}$ is injective, we proceed as follows. Suppose Σ, Σ_1, and Σ_2 are MCSs satisfying $\Sigma_1 (R^c_{\mathsf{r}_1})^{-1}; R^c_{\mathsf{r}_1} \Sigma$ and $\Sigma_2 (R^c_{\mathsf{r}_1})^{-1}; R^c_{\mathsf{r}_1} \Sigma$. Let $\phi \in \Sigma_1$. Since $\Sigma_1 (R^c_{\mathsf{r}_1})^{-1}; R^c_{\mathsf{r}_1} \Sigma$, there is some MCS Σ' such that $\Sigma' R^c_{\mathsf{r}_1} \Sigma_1$ and $\Sigma' R^c_{\mathsf{r}_1} \Sigma$. Since $\phi \in \Sigma_1$ and $\Sigma' R^c_{\mathsf{r}_1} \Sigma_1$, we have $\langle \mathsf{r}_1 \rangle \phi \in \Sigma'$. Now, by applying Axiom 5, we obtain $[\mathsf{r}_1]\phi \in \Sigma'$, and since $\Sigma' R^c_{\mathsf{r}_1} \Sigma$, we have $\phi \in \Sigma$. Besides, since $\Sigma_2 (R^c_{\mathsf{r}_1})^{-1}; R^c_{\mathsf{r}_1} \Sigma$, there is some MCS Σ'' such that $\Sigma'' R^c_{\mathsf{r}_1} \Sigma_1$ and $\Sigma'' R^c_{\mathsf{r}_1} \Sigma$. Since $\Sigma'' R^c_{\mathsf{r}_1} \Sigma$ and $\phi \in \Sigma$, we have $\langle \mathsf{r}_1 \rangle \phi \in \Sigma''$. Again, by applying Axiom 5, we obtain $[\mathsf{r}_1]\phi \in \Sigma''$, and since $\Sigma'' R^c_{\mathsf{r}_1} \Sigma_2$, we have $\phi \in \Sigma_2$. Hence, $\Sigma_1 \subseteq \Sigma_2$. The inclusion $\Sigma_2 \subseteq \Sigma_1$ can be proved analogously.

The proof that $(R^c_{\mathsf{r}_2})^{-1}; R^c_{\mathsf{r}_2}$ is injective is entirely similar.

To prove that $R^c_{\mathsf{r}_1}$ and $R^c_{\mathsf{r}_2}$ have the same domain, we proceed as follows. Suppose Σ and Σ' are MCSs satisfying $\Sigma R^c_{\mathsf{r}_1} \Sigma'$. Since $\top \in \Sigma'$, we have $\langle \mathsf{r}_1 \rangle \top \in \Sigma$. Now, by applying Axiom 7, we obtain $[\mathsf{r}_2]\top \in \Sigma$ and, then there is some MCS Σ'' such that $\Sigma R^c_{\mathsf{r}_2} \Sigma''$. Hence, the domain of $R^c_{\mathsf{r}_1}$ is included in the domain of $R^c_{\mathsf{r}_2}$.

The proof of the other inclusion is entirely similar.

The proofs that $R^c_{\mathsf{s}_1}$ and $R^c_{\mathsf{r}_1}$ are converses, as well the proof of the same property for $R^c_{\mathsf{s}_2}$ and $R^c_{\mathsf{r}_2}$, are as usual in temporal logic, using Axiom 6. □

Given $\Sigma \in W^c$, let $\mathcal{M}^\Sigma$ be the sub-model of $\mathcal{M}^c$, generated by Σ and $(R^c_{\mathsf{r}_1})^{-1}; R^c_{\mathsf{r}_2}$. By the Generated Sub-model Lemma, we have:

Lemma 5.5 (i) *For any $\Sigma \in W^c$, $\mathcal{M}^\Sigma \models \mathrm{RSPDL}^0$.*

(ii) *If $\nvdash_{\mathrm{RSPDL}^0} \varphi$, then there is a $\Sigma \in W^c$ such that $\mathcal{M}^\Sigma, \Sigma \not\models \varphi$.*

(iii) *Relation $(R^\Sigma_{\mathsf{r}_1})^{-1}; R^\Sigma_{\mathsf{r}_2}$ is total.*

Now we see prove that, given $\Sigma \in W^c$, the model $\mathcal{M}^\Sigma$ has enough nice properties to be the counter-model we need. In fact, by applying a reasoning similar to that employed in [9], we can prove that $\mathcal{M}^\Sigma$ is indeed a model of RSPDL^0, i.e. that W^Σ is a structured set. More specifically, since $R_{\mathsf{s}_1}, R_{\mathsf{r}_1}, R_{\mathsf{s}_2}$ and R_{r_2} are programs of RSPDL^0 and since they satisfy Lemmas 5.4 and 5.5, we have that $R_{\mathsf{s}_1}, R_{\mathsf{r}_1}, R_{\mathsf{s}_2}$ and R_{r_2} are functional relations sharing the same domain, covering $W^\Sigma \times W^\Sigma$, and warranting unicity of ordered pairs. These conditions suffice to define an injective function $\star : W^{\Sigma} \times W^{\Sigma} \to W^{\Sigma}$ for which $R_{\mathsf{s}_1} = \{(w, w \star v) : w, v \in W^{\Sigma}\}$, $R_{\mathsf{s}_2} = \{(v, w \star v) : w, v \in W^{\Sigma}\}$, $R_{\mathsf{r}_1} = \{(w \star v, w) : w, v \in W^{\Sigma}\}$, and $R_{\mathsf{r}_2} = \{(w \star v, v) : w, v \in W^{\Sigma}\}$, as follows.

Define $f \subseteq (W^{\Sigma} \times W^{\Sigma}) \times W^{\Sigma}$ in the following way, for all $w, v, u \in W^{\Sigma}$:

$$((w, v), u) \in f \text{ iff } (u, w) \in R_{\mathsf{r}_1} \text{ and } (u, v) \in R_{\mathsf{r}_2}.$$

We have that f is an injective functional relation. All these facts together allow us to define $\star : W^{\Sigma} \times W^{\Sigma} \to W^{\Sigma}$ by setting:

$$w \star v = f(w, v),$$

for any pair $(w, v) \in W^{\Sigma} \times W^{\Sigma}$. From this definition it is obvious that

$$w \star v = u \text{ iff } (u, w) \in R_{\mathsf{r}_1} \text{ and } (u, v) \in R_{\mathsf{r}_2},$$

for any $(w, v) \in W^{\Sigma} \times W^{\Sigma}$ and $u \in W^{\Sigma}$, and this gives us $R_{\mathsf{s}_1} = \{(w, w \star v) : w, v \in W^{\Sigma}\}$, $R_{\mathsf{s}_2} = \{(v, w \star v) : w, v \in W^{\Sigma}\}$, $R_{\mathsf{r}_1} = \{(w \star v, w) : w, v \in W^{\Sigma}\}$, and $R_{\mathsf{r}_2} = \{(w \star v, v) : w, v \in W^{\Sigma}\}$.

To conclude the proof just observe that f is functional and injective, and $\mathsf{Dom} f = W^{\Sigma} \times W^{\Sigma}$. This is what we need to show that $\mathcal{M}^{\Sigma}$ is a proper model. □

6 Final Remarks

This paper starts the study of PRSPDL, an extension of the PDL regular language with a parallel operator and four operators: two to store data and two to recover data. More specifically, we exemplify the expressive power of PRSPDL and present an axiomatization for a restrict fragment, RSPDL^0, without parallel composition, iteration and test and provide a proof of completeness for this fragment.

The semantics of PRSPDL is a generalization of Kripke semantics with a notion of structured set of possible states instead of sets of states. Structured sets allows one to represent structured data in a very natural way, as we show in some specific examples.

There are many possibilities for future work, we just list the most prominent. First, we would like to establish decidability and complexity issues for the fragment RSPDL^0. Second, we would like to provide an axiomatization for PRSPDL with parallel composition, iteration and test, provide a proof of completeness, and investigate decidability and complexity questions for it. Finally, it would be interesting to have some application of the PRSPDL language in specification of properties of programs with mutable data structures and updates.

References

[1] Abrahamson, K., "Decidability and expressiveness of logics of processes," Ph.D. thesis, Ph.D. Dissertation, Department of Computer Science, University of Washington (1980).

[2] Benevides, M. and L. Schechter, *A propositional dynamic logic for CCS programs*, in: *Proceedings of the XV Workshop on Logic, Language, Information and Computation*, LNAI **5110** (2008), pp. 83–97.

[3] Benevides, M. and L. Schechter, *A propositional dynamic logic for concurrent programs based on the π-calculus*, in: *Proceedings of Methods for Modalities*, Electronic Notes in Theoretical Computer Science ENTCS **262** (2010), pp. 49–64.

[4] Brown, C. and G. Hutton, *Categories, allegories and circuit design.*, in: *Proc. Ninth Annual IEEE Symp. on Logic in Computer Science* (1994), pp. 372–381.

[5] Brown, C. and A. Jeffrey, *Allegories of circuits.*, in: *Proc. Logical Foundations of Computer Science* (1994), pp. 56–68.

[6] dos Santos, V., "Concorrência e Sincronização para Lógica Dinâmica de Processos," Ph.D. thesis, Federal University of Rio de Janeiro (2005).

[7] Fischer, M. and R. Ladner, *Propositional dynamic logic of regular programs*, Journal of Computer and System Sciences **18** (1979), pp. 194–211.

[8] Freitas, R., P. Viana, M. Benevides, S. Veloso and P. Veloso, *Squares in fork arrow logic*, Journal of Philosophical Logic **32-4** (2003).

[9] Frias, M., "Fork Algebras in Algebra, Logic and Computer Science," Advances in Logic, Vol. 2, World Scientific, 2002.

[10] Frias, M., P. Veloso and G. Baum, *Fork algebras: past, present and future*, Journal of Relational Methods in Computer Science **1** (2004), pp. 181–216.

[11] Gerbrand, J. and W. Groeneveld, *Reasoning about information change*, Journal of Logic, Language and Information **6** (1997), pp. 147–169.

[12] Goldblatt, R., *Parallel action: Concurrent dynamic logic with independent modalities*, Studia logica **51** (1992).

[13] Haeberer, A., M. Frias, G. Baum and P. Veloso, *Fork algebras*, in: W. K. C. Brink and G. Schmidt, editors, *Relational Methods in Computer Science*, Springer, Viena, 1997 .

[14] Harel, D., D. K. D. and Tiuryn, "Dynamic Logics," MIT Press, 2000.

[15] Hoare, T. and P. P. O'Hearn, *Separation logic semantics for communicating processes*, Electron. Notes Theor. Comput. Sci. **212** (2008), pp. 3–25.

[16] Mayer, A. and L. Stockmeyer, *The complexity of pdl with interleaving*, Theoretical Computer Science **161-1-2** (1996).

[17] O'Hearn, P., J. Reynolds and H. Yang, *Local reasoning about programs that alter data structures*, in: *Lecture Notes in Computer Science*, Computer Science Logica **2142** (2001), pp. 1–19.

[18] Peleg, D., *Communication in concurrent dynamic logic*, Journal of Computer and System Sciences **35** (1987), pp. 23–58.

[19] Peleg, D., *Concurrent dynamic logic*, Journal of the Association for Computing Machinery **34** (1987), pp. 450–479.

[20] Plaza, J., *Logics of public communications*, in: *Proceedings of 4^{th} Intrenational Symposium on Methodologies for Intelligent Systems*, 1989, pp. 201–216.

[21] Reynolds, J., *Separation logic: A logic for shared mutable data structures*, in: *Proceedings of 7^{th} Annual IEEE Symposium on Logic in Computer Science* (2002), pp. 55–74.

[22] Veloso, P., R. Freitas, P. Viana, M. Benevides and S. Veloso, *On fork arrow logic and its expressive power*, Journal of Philosophical Logic **36-5** (2007).