EPiC Series in Computing

Volume 45, 2017, Pages 29–13

SCSS 2017. The 8th International Symposium on Symbolic Computation in Software Science 2017

$\mathcal{C}2\mathcal{PDLS}$: A Combination of Combinatory and Converse PDL with Substitutions

Jon Haël Brenas[1] Rachid Echahed[1] Martin Strecker[2]

[1] CNRS and University Grenoble Alpes
[2] Université de Toulouse / IRIT

Abstract

We introduce a logic called $\mathcal{C}2\mathcal{PDLS}$, motivated by some reasoning about graph rewriting systems. $\mathcal{C}2\mathcal{PDLS}$ is an extension of both combinatory propositional dynamic logic, usually written $\mathcal{CPDL}$, and converse propositional dynamic logic, usually written $\mathcal{CPDL}$ too. In addition to the existing features of both $\mathcal{CPDL}$s, the introduced logic offers the possibility to use the notion of substitutions à la Hoare within its formulae. Such substitutions reflect the effect of some actions on graph structures such as addition or deletion of edges or nodes. These last features led us to introduce restricted universal roles over subsets of the universe. We propose a sound and complete deductive system for $\mathcal{C}2\mathcal{PDLS}$ and show that its validity problem is decidable.

1 Introduction

Graph structures play an important role when modeling complex systems. They are used in different areas going from computer programs to modeling tools in natural science. These graphs could be either static or dynamic. In this paper we are rather concerned with the latter case where graphs can evolve thanks to some actions aimed at performing some graph transformations.

There are several approaches to write programs which handle graph transformations going from classical imperative languages to dedicated rule-based approaches [20]. Reasoning on these transformations is still not mature enough. Some logics tailored to graph transformations have already been proposed (see, *e.g.* [3, 5]) but suffer from the lack of decision procedures.

Recently, an approach to reasoning on graph transformation based on Hoare like calculi [16] has been proposed by several authors (see, *e.g.*, [19, 10, 1]). Roughly speaking, within such an approach, one may prove that the resulting graph G', obtained after performing an action α over G, fulfills a property P, written $(G' \models P)$, whenever a given graph G satisfies the precondition $P[\alpha]$, $(G \models P[\alpha])$. The notation $[\alpha]$ is known as a *substitution* induced from action α. Different logics could be used to specify formulas such as P or $P[\alpha]$ depending on the properties one may wish to prove. The main features of such logics include facilities to handle substitutions on one hand and the fact to be endowed by some decision procedures on the other hand.

In this paper, we investigate a dynamic logic where substitutions reflecting several elementary actions over graphs are part of its syntax. This logic, called $\mathcal{C}2\mathcal{PDLS}$, is an extension of both converse [14] and combinatory [17] propositional dynamic logics, both classically designated by $\mathcal{CPDL}$. This allows us to give quite expressive characterizations of graphs: for instance, the formula $[\nu](\langle(\alpha^- \cup \beta^-)^*\rangle C \Rightarrow D)$ describes the graphs such that all nodes that can reach a node labeled C using only edges labeled α or β are labeled D. One of the main contributions of combinatory $\mathcal{PDL}$ is the introduction of the nominals of Hybrid Logics. These can be combined with several modals for instance in temporal logics [2].

The use of dynamic logics [15], an extension of modal logics [7, 8], is among the most widespread ways to reason about complex and evolving data. In particular, modal logics are particularly well-suited to describe the relations between diverse states of the data using the programs that allow to go from one state to another. They are also a very efficient way to represent graph structured data. For instance, the formula $\langle\alpha\rangle\phi$ can be used to signify either that from the current state of execution, it is possible to reach a state where ϕ is true by performing α or that the current node is linked, via an edge labeled with α, to a node labeled with ϕ.

On the other hand, Description Logics [4] form one of the most important families of logics used to describe graphs and some of them have thus been extended so that they are closed under substitutions [1, 12]. The main issue with DLs is that, even though the more expressive logics allow to define a role as transitive, they lack the reachability assertions that can be expressed using $\mathcal{CPDL}$. That is why we decided to propose and investigate $\mathcal{C}2\mathcal{PDLS}$.

The paper is organized as follows. The logic $\mathcal{C}2\mathcal{PDLS}$ is defined in Sect. 2. Then, in Sect. 3, we prove that the presence of substitutions, while convenient to express the correctness of graph transformations, does not increase the expressive power of $\mathcal{C}2\mathcal{PDLS}$. In Sect. 4, the proof that $\mathcal{C}2\mathcal{PDLS}$ is decidable is sketched. Related work and concluding remarks are given respectively in Sect. 5 and Sect. 6. The missing proofs and more details can be found in [11].

2 Syntax and models of $\mathcal{C}2\mathcal{PDLS}$

In this section, we introduce $\mathcal{C}2\mathcal{PDLS}$, a combination of both Combinatory and Converse Propositional Dynamic Logic, both named $\mathcal{CPDL}$ [14, 17] augmented with a notion of substitutions. These kind of substitutions are very common in Hoare-like program verification procedures [16]. In order to take into account the interpretations of formulae with substitutions we assume the universe (of names), Σ, split into two subsets Σ_1 and Σ_2. Intuitively, elements whose names are in Σ_1 are the building blocks of the formulae before substitutions are taken into account. Elements whose names are in Σ_2, on the other hand, are introduced when a substitution creates a new node. It also stores the names of elements that have been deleted. The need to access nodes outside of those that form the initial model leads us to the modification of the universal program ν. It is now used with an index indicating on which set of names it operates.

Definition 2.1 (Syntax of $\mathcal{C}2\mathcal{PDLS}$)**.** *Given three countably infinite and pairwise disjoint alphabets* Σ*, the set of* names*,* Φ_0*, the set of* atomic propositions*, and* Π_0*, the set of* atomic programs*, the language of* $\mathcal{C}2\mathcal{PDLS}$ *is composed of* formulae*,* programs *and* substitutions*. We partition the set of names* Σ *into two countably infinite sets* Σ_1 *and* Σ_2 *such that* $\Sigma_1 \cup \Sigma_2 = \Sigma$ *and* $\Sigma_1 \cap \Sigma_2 = \emptyset$*. Formulae* ϕ*, programs* α *and substitutions* θ *are defined as follows:*

$$\begin{array}{lcl} \phi & := & i \mid \phi_0 \mid \neg\phi \mid \phi \vee \phi \mid \langle\alpha\rangle\phi \mid \phi[\theta] \\ \alpha & := & \alpha_0 \mid \nu_S \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \alpha^- \mid \phi? \\ \theta & := & add(i,\phi_0) \mid del(i,\phi_0) \mid add(i,j,\alpha_0) \mid del(i,j,\alpha_0) \mid i \gg j \mid add(i) \mid del(i) \end{array}$$

where $i, j \in \Sigma$, $\phi_0 \in \Phi_0$, $\alpha_0 \in \Pi_0$ *and* $S \subseteq \Sigma$.

We denote by Π *the set of programs,* Φ *the set of formulae and* Θ *the set of substitutions. As usual,* $\phi \wedge \psi$ *stands for* $\neg(\neg\phi \vee \neg\psi)$, $\phi \Rightarrow \psi$ *stands for* $\neg\phi \vee \psi$, $\phi \Leftrightarrow \psi$ *stands for* $(\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)$ *and* $[\alpha]\phi$ *stands for* $\neg(\langle\alpha\rangle\neg\phi)$.

It is noteworthy that programs define paths between nodes and not actual modifications of the graphs that are handled by the substitutions. One can also see that not all formulae use names in a way that makes sense. Let $\phi \equiv (\langle\nu_{\Sigma_1}\rangle\phi_0)[add(i, \phi_0)]$. Intuitively, ϕ means that there exists a node labeled ϕ_0 ($\langle\nu_{\Sigma_1}\rangle\phi_0$) after labeling i with ϕ_0. On the other hand, let $\phi' \equiv (\langle\nu_{\Sigma_1}\rangle\phi_0)[add(i)][add(i, \phi_0)]$. As substitutions alter graphs from right to left, ϕ' says that there is a node labeled with ϕ_0 after labeling i with ϕ_0 and then creating i. It should not be possible to modify a node that has yet to be created and thus ϕ' should not be possible. Henceforth, we only consider *well-formed* formulae.

Definition 2.2 (Well-formed formula)**.** *A formula is said to be well-formed if it is possible to find a set, called the* annotation*,* $E \subseteq \Sigma$ *such that the following inference rules are respected:*

$$\frac{\phi^{E\cup\{i\}} \quad i \notin E}{\phi[add(i)]^E} \qquad \frac{\phi^{E-\{i\}} \quad i \in E}{\phi[del(i)]^E} \qquad \frac{\phi^E \quad \theta \neq add(i), \theta \neq del(i) \quad i \in E}{\phi[\theta]^E}$$

$$\frac{}{i^{\Sigma_1}} \qquad \frac{}{\phi_0^{\Sigma_1}} \qquad \frac{\phi^E}{(\neg\phi)^E} \qquad \frac{\phi_1^E \quad \phi_2^E}{(\phi_1 \vee \phi_2)^E}$$

Informally, well-formed formulae are formulae that can be annotated with a coherent set of "existing" nodes, *i.e.* nodes whose labels can be modified. The labeling for substitution-free formulae is Σ_1 as given in the signature and only $add(i)$ and $del(i)$ actions can modify the annotation by respectively requiring that i did not or did belong to the annotation.

We define below how (well-formed) formulae are interpreted using *models*.

Definition 2.3 (Model)**.** *A* model *is a tuple* $\mathcal{M} = (M, R, \chi, V, \mathcal{E})$ *where* M *is a set called the* universe*,* $\chi : \Sigma \to M$ *is a surjective mapping such that* $\chi(\Sigma_1) \cap \chi(\Sigma_2) = \emptyset$, $R : \Pi \to \mathcal{P}(M^2)$ *is a mapping,* $V : \Phi \to \mathcal{P}(M)$ *is a mapping and* $\mathcal{E} : \Phi \to \mathcal{P}^2(\Sigma)$ *keeps track of the annotation. They are defined such that:*

- *For each* $i \in \Sigma$, $V(i) = \{\chi(i)\}$ *and* $\mathcal{E}(i) = \{\Sigma_1\}$
- *For each* $\phi_0 \in \Phi_0$, $V(\phi_0) \in \mathcal{P}(\chi(\Sigma_1))$ *and* $\mathcal{E}(\phi_0) = \{\Sigma_1\}$
- $V(\neg\phi) = \overline{V(\phi)}$ *and* $\mathcal{E}(\neg\phi) = \mathcal{E}(\phi)$
- $V(\phi \vee \psi) = V(\phi) \cup V(\psi)$ *and* $\mathcal{E}(\phi \vee \psi) = \mathcal{E}(\phi) \cup \mathcal{E}(\psi)$
- $V(\langle\alpha\rangle\phi) = \{s \in M | \exists t \in M.((s, t) \in R(\alpha) \wedge t \in V(\phi))\}$ *and* $\mathcal{E}(\langle\alpha\rangle\phi) = \mathcal{E}(\phi)$
- *For each* $\alpha_0 \in \Pi_0$, $R(\alpha_0) \in \mathcal{P}(\chi(\Sigma_1)^2)$
- $R(\nu_S) = \chi(S)^2$ *for* $S \subseteq \Sigma$
- $R(\alpha \cup \beta) = R(\alpha) \cup R(\beta)$

- $R(\alpha;\beta) = \{(s,t) \in M^2 | \exists v.((s,v) \in R(\alpha) \wedge (v,t) \in R(\beta))\}$
- $R(\alpha^-) = \{(s,t) \in M^2 | (t,s) \in R(\alpha)\}$
- $R(\alpha^*) = \bigcup_{k<\omega} R(\alpha^k)$ *where* α^k *stands for the sequence* $\alpha;\ldots;\alpha$ *of length* k
- $R(A?) = \{(s,s) \in M^2 | s \in V(A)\}$
- $V(\phi[add(i,\phi_0)]) = V'(\phi)$ *where* $\mathcal{M}' = (M,R,\chi,V',\mathcal{E})$ *is a model such that* $\forall \psi_0 \in \Phi_0, \psi_0 \neq \phi_0.V'(\psi_0) = V(\psi_0)$ *and* $V'(\phi_0) = V(\phi_0) \cup \{\chi(i)\}$
- $V(\phi[del(i,\phi_0)]) = V'(\phi)$ *where* $\mathcal{M}' = (M,R,\chi,V',\mathcal{E})$ *is a model such that* $\forall \psi_0 \in \Phi_0, \psi_0 \neq \phi_0.V'(\psi_0) = V(\psi_0)$ *and* $V'(\phi_0) = V(\phi_0) \cap \overline{\{\chi(i)\}}$
- $V(\phi[add(i,j,\pi_0)]) = V(\phi)$ *where* $\mathcal{M}' = (M,R',\chi,V,\mathcal{E})$ *is a model such that* $\forall \alpha_0 \in \Pi_0, \pi_0 \neq \alpha_0.R'(\alpha_0) = R(\alpha_0)$ *and* $R'(\pi_0) = R(\pi_0) \cup \{(\chi(i),\chi(j))\}$
- $V(\phi[del(i,j,\pi_0)]) = V'(\phi)$ *where* $\mathcal{M}' = (M,R',\chi,V,\mathcal{E})$ *is a model such that* $\forall \alpha_0 \in \Pi_0, \pi_0 \neq \alpha_0.R'(\alpha_0) = R(\alpha_0)$ *and* $R'(\pi_0) = R(\pi_0) \cap \overline{\{(\chi(i),\chi(j))\}}$
- $V(\phi[i \gg j]) = V'(\phi)$ *where* $\mathcal{M}' = (M,R',\chi,V,\mathcal{E})$ *is a model such that* $\forall \pi_0 \in \Pi_0.R'(\pi_0) = R(\pi_0) \cap \overline{\{(a,i) \in R(\pi_0)\}} \cup \{(a,j) | (a,i) \in R(\pi_0)\}$
- $V(\phi[add(i)]) = V'(\phi)$ *where* $\mathcal{M}' = (M,R,\chi,V,\mathcal{E}')$ *is a model such that* $\forall \psi \neq \phi[add(i)].\mathcal{E}'(\psi) = \mathcal{E}(\psi)$ *and* $(\mathcal{E}'(\phi[add(i)]) = \{S \cap \overline{\{i\}} | S \in \mathcal{E}(\phi)\}$
- $V(\phi[del(i)]) = V'(\phi)$ *where* $\mathcal{M}' = (M,R',\chi,V',\mathcal{E}')$ *is a model such that* $\forall \pi_0, R'(\pi_0) = R(\pi_0) \cap \overline{\{(k,l) | k = i \vee l = i\}}$, $\forall \phi_0.V'(\phi_0) = V(\phi_0) \cap \overline{\{i\}}$, $\forall \psi \neq \phi[add(i)].\mathcal{E}'(\psi) = \mathcal{E}(\psi)$ *and* $\mathcal{E}'(\phi[del(i)]) = \{S \cup \{i\} | S \in \mathcal{E}(\phi)\}$

where $\mathcal{P}(E)$ *(resp.* $\mathcal{P}^2(E)$*) stands for the powerset of* E *(resp. the powerset of the powerset of* E*),* $\overline{E}$ *is the complement of* E *w.r.t. the superset used in the context. In the following, we write* $m \models A$ *whenever* $m \in V(A)$ *for* m *being an element of the universe* M *and* A *is a formula. As usual, a formula* A *is said to be satisfiable if there exist a model* $\mathcal{M} = (M,R,\chi,V)$ *and an element* m *of* M *such that* $m \models A$*. When it is not the case,* A *is said to be unsatisfiable.* A *is said to be valid, written* $\models A$*, if* $\neg A$ *is unsatisfiable, that is for every model* $\mathcal{M}$*, for all elements* m *of* $\mathcal{M}$*,* $m \models A$*, and invalid otherwise. We say that a model* $\mathcal{M}$ *satisfies a formula* A *and write* $\mathcal{M} \models A$ *if there exists an element* m *in* M *such that* $m \models A$*.*

Intuitively, the function χ, which is surjective and such that $\chi(\Sigma_1) \cap \chi(\Sigma_2) = \emptyset$, splits the universe into elements whose names are in Σ_1 and those whose names are in Σ_2. This splitting of the universe M is motivated by the evolution of models (graphs) that we consider in the forthcoming sections. M is split into nodes that are initially part of the graph $\chi(\Sigma_1)$ and nodes that may be used in the future or may have been deleted in the past $\chi(\Sigma_2)$. This is a way to provide the logic with evolving sets that handle nodes that are either used or potentially usable. So, for all $\phi_0 \in \Phi_0$, $V(\phi_0) \in \mathcal{P}(\chi(\Sigma_1))$, thus for all $n_2 \in \chi(\Sigma_2)$, $n_2 \notin V(\phi_0)$ which means that no atomic proposition is satisfied by any unused node. For almost the same reason, for all $\pi \in \Pi_0$, there is no $m \in M$ such that $(n_2,m) \in R(\pi)$ or $(m,n_2) \in R(\pi)$. Thus, all nodes of $\chi(\Sigma_2)$ are such that they satisfy no atomic proposition and they have no incoming or outgoing edge. Moreover, as χ is surjective, ν_Σ is the usual universal role, that is $R(\nu_\Sigma) = M \times M$.

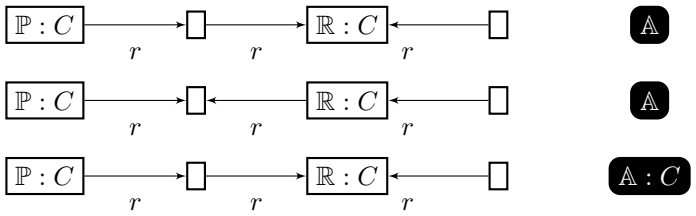

Figure 1: Models and counter-model. White nodes correspond to names in Σ_1, black ones correspond to those in Σ_2. The first example is a model and every node $\models cs$. The second one is a model but $\mathbb{P} \not\models cs$ hence cs is not valid. The third example is not a model since $\chi(\mathbb{A}) \in V(C)$ which violates Definition 2.3.

Example 2.1. *Let us consider a specification of the notions of cities and roads. We define* Σ, Φ_0 *and* Π_0 *such that* $\{\mathbb{R}, \mathbb{P}\} \subset \Sigma_1$, $\mathbb{A} \in \Sigma_2$, $C \in \Phi_0$ *and* $r \in \Pi_0$ *where* $\mathbb{R}$ *is the name associated with the city Rome,* $\mathbb{P}$ *is the name associated with the city of Paris,* $\mathbb{A}$ *is the name associated with the city of Atlantis,* C *is the atomic proposition associated with cities and* r *is the atomic program associated with roads.*

The common saying that all roads lead to Rome becomes $cs \equiv [r]\langle r^* \rangle \mathbb{R}$*, that is for all roads (*$[r]$*) it eventually leads(*$\langle r^* \rangle$*) to Rome (*$\mathbb{R}$*). Fig. 1 shows a model and a counter-model of* cs*.*

Consider now the formula $ac \equiv \langle \nu_{\Sigma_1 \cup \{\mathbb{A}\}} \rangle (C \vee \mathbb{A})$*, it says that there exists an initial node or Atlantis (*$\langle \nu_{\Sigma_1 \cup \{\mathbb{A}\}} \rangle$*) such that it is a city or Atlantis (*$C \vee \mathbb{A}$*). It is an obvious tautology as Atlantis is Atlantis.*

As a last example of a formula, we consider $ac' \equiv (\langle \nu_{\Sigma_1} \rangle C)[add(\mathbb{A}, C)][add(\mathbb{A})]$ *which says that there exists a city (*$\langle \nu_{\Sigma_1} \rangle C$*) after adding Atlantis (*$[add(\mathbb{A})]$*) and making it a city (*$[add(\mathbb{A}, C)]$*). It is noteworthy that* $\mathbb{A} \in \Sigma_2$ *initially but as* $[add(\mathbb{A})]$ *occurs first (that is on the right), it is in* Σ_1 *when* $[add(\mathbb{A}, C)]$ *occurs and thus* ac' *is a (well-formed) formula of* $\mathcal{C}2\mathcal{PDLS}$*.*

Let us consider the formula $ac_1 \equiv (\langle \nu_{\Sigma_1} \rangle C)[add(\mathbb{A}, C)]$*.* ac_1 *is a* $\mathcal{C}2\mathcal{PDLS}$ *formula only if we assume that* $\mathbb{A}$ *is in* Σ_1*.*

Finally, let us consider $ac_2 \equiv (\langle \nu_{\Sigma_1} \rangle C)[add(\mathbb{A})][add(\mathbb{A}, C)]$*.* ac_2 *is not a well-formed* $\mathcal{C}2\mathcal{PDLS}$ *formula since it is impossible to find a* E *such that* $\mathbb{A} \in E$*, required for* $add(\mathbb{A}, C)$*, and* $\mathbb{A} \notin E$*, required for* $add(\mathbb{A})$*.*

3 $\mathcal{C}2\mathcal{PDLS}$ vs $\mathcal{C}2\mathcal{PDL}$

In this section we investigate a relation between the logic $\mathcal{C}2\mathcal{PDLS}$ and its substitution free counterpart named $\mathcal{C}2\mathcal{PDL}$.

Definition 3.1. *We define the logic* $\mathcal{C}2\mathcal{PDL}$ *as the restriction of* $\mathcal{C}2\mathcal{PDLS}$ *to formulae without substitutions.*

Definition 3.2. *Two formulae* A *and* A' *are said to be equivalent, written* $A \equiv A'$ *if, given any model* $\mathcal{M} = (M, R, \chi, V, \mathcal{E})$*,* $V(A) = V(A')$*. Similarly, two programs* α *and* α' *are said to be equivalent, written* $\alpha \equiv \alpha'$*, if* $R(\alpha) = R(\alpha')$*.*

In the following we state that the logic $\mathcal{C}2\mathcal{PDL}$ is as expressive as $\mathcal{C}2\mathcal{PDLS}$. That is to say, that for every formula Φ of $\mathcal{C}2\mathcal{PDLS}$ there exist a corresponding one Φ' in $\mathcal{C}2\mathcal{PDL}$ such that ϕ and ϕ' are equivalent.

Theorem 3.1. *$\mathcal{C}2\mathcal{PDLS}$ is as expressive as $\mathcal{C}2\mathcal{PDL}$.*

To prove the theorem, we introduce a rewriting system $\mathcal{RS}$. Its goal is to transform any formula where substitutions occur into a substitution-free formula. It is not always possible to do that in one step. The rewriting system thus contains rules that remove substitutions completely and other rules that move the substitution inward. Each rule is such that the left-hand side and the right-hand side are equivalent.

Let $\sigma, \sigma', \sigma'' \in \Theta$, $\sigma' \in \{[add(i,j,\alpha_0)], [del(i,j,\alpha_0)], [add(i)], [i \gg j]\}$, ϕ_0 and $\phi_1 \in \Phi_0$, $\phi_0 \neq \phi_1$, ϕ and $\psi \in \Phi$, $i \in \Sigma$ and $\alpha \in \Pi$ then $\mathcal{RS}_\phi$ is:

Rule $\phi_1 : \top\sigma \rightsquigarrow \top$	*Rule* $\phi_2 : i\sigma \rightsquigarrow i$
Rule $\phi_3 : \phi_0\sigma' \rightsquigarrow \phi_0$	*Rule* $\phi_4 : \phi_0[add(i,\phi_1)] \rightsquigarrow \phi_0$
Rule $\phi_5 : \phi_0[add(i,\phi_0)] \rightsquigarrow \phi_0 \vee i$	*Rule* $\phi_6 : \phi_0[del(i,\phi_1)] \rightsquigarrow \phi_0$
Rule $\phi_7 : \phi_0[del(i,\phi_0)] \rightsquigarrow \phi_0 \wedge \neg i$	*Rule* $\phi_8 : \phi_0[del(i)] \rightsquigarrow \phi_0 \wedge \neg i$
Rule $\phi_9 : (\neg\phi)\sigma \rightsquigarrow \neg(\phi\sigma)$	*Rule* $\phi_{10} : (\phi \vee \psi)\sigma \rightsquigarrow (\phi\sigma) \vee (\psi\sigma)$
Rule $\phi_{11} : (\langle\alpha\rangle\phi)\sigma \rightsquigarrow \langle\alpha\sigma\rangle(\phi\sigma)$	

Similarly, rewriting rules are used to get rid of substitutions affecting roles and name sets. They are not reported here for lack of space but they can be found in [11].

Proving that these rules are correct, that is that the valuations of the left- and right-hand sides are equal, is not difficult. In order to save space, only few of these proofs are reported here.

Proof.

Rule ϕ_2 : As nodes are never renamed, $V(i\sigma) = V(i)$

Rule ϕ_5 : As $V(\phi[add(i_1,\phi)]) = V'(\phi) = \{\chi(i_1)\} \cup V(\phi)$, $V(\phi[add(i_1,\phi)]) = V(\phi \vee i_1)$.

□

Example 3.1. *Applying the rules given in the proof allows one to prove that the formula ac' rewrites to ac.*

4 Deductive system for $\mathcal{C}2\mathcal{PDL}$

We now introduce a deductive system $\mathcal{DS}$ for $\mathcal{C}2\mathcal{PDL}$. It is composed of 17 axioms (from (*Bool*) to ($\Sigma_2 2$)) and 5 deductive rules (from (Ax) to (Nec)). It is noteworthy that formulae (resp. programs) of $\mathcal{C}2\mathcal{PDL}$ are also formulae (resp. programs) of $\mathcal{DS}$ and the other way round. The deductive system as well as the proofs of the following theorems are inspired by the ones in [17] and for PDL.

4.1 Deductive system $\mathcal{DS}$

Let A and $B \in \Phi$, α and $\beta \in \Pi$, c and $d \in \Sigma$, $S \subseteq \Sigma$,

	PDL Axioms:		Names:
(Bool)	All boolean tautologies	**(Σ1)**	$\langle \nu_\Sigma \rangle c$
($\Box$)	$[\alpha](A \Rightarrow B) \Rightarrow ([\alpha]A \Rightarrow [\alpha]B)$	**(Σ2)**	$\langle \nu_\Sigma \rangle (c \wedge A) \Leftrightarrow [\nu_\Sigma](c \Rightarrow A)$
(;)	$\langle \alpha;\beta \rangle A \Leftrightarrow \langle \alpha \rangle \langle \beta \rangle A$		Universal programs:
($\cup$)	$\langle \alpha \cup \beta \rangle A \Leftrightarrow \langle \alpha \rangle A \vee \langle \beta \rangle A$	**(ν_S1)**	$\forall c', d' \in S.c' \Rightarrow \langle \nu_S \rangle d'$
(?)	$\langle A? \rangle B \Leftrightarrow A \wedge B$	**(ν_S2)**	$\forall \{c'', d''\} \not\subseteq S.c'' \Rightarrow [\nu_S] \neg d''$
(*)	$\langle \alpha^* \rangle A \Leftrightarrow A \vee \langle \alpha \rangle \langle \alpha^* \rangle A$	**(ν_S3)**	$\langle \nu_S \rangle \langle \nu_S \rangle A \Rightarrow \langle \nu_S \rangle A$
(-)	$A \Rightarrow [\alpha] \langle \alpha^- \rangle A$	**(ν_S4)**	$A \Rightarrow [\nu_S] \langle \nu_S \rangle A$
	Rules:	**(ν_Σ1)**	$A \Rightarrow \langle \nu_\Sigma \rangle A$
(Ax)	If A is an axiom, $\vdash A$	**(ν_Σ2)**	$\langle \alpha \rangle A \Rightarrow \langle \nu_\Sigma \rangle A$
(MP)	If $\vdash A$ and $\vdash A \Rightarrow B$, then $\vdash B$		Names $\in \Sigma_2$:
(Ind)	If $\vdash [\gamma][\alpha^k]A$, for all $k < \omega$, then $\vdash [\gamma][\alpha^*]A$	**($\Sigma_2$1)**	$\forall c \in \Sigma_2, \forall \phi \in \Phi_0.c \Rightarrow \neg\phi$
(Cov)	If $\vdash [\gamma]\neg c$, for all $c \in \Sigma$, then $\vdash [\gamma]\bot$	**($\Sigma_2$2)**	$\forall c \in \Sigma_2, \forall \alpha \in \Pi_0.c \Rightarrow$ $[\alpha]\bot \wedge [\alpha^-]\bot$
(Nec)	If $\vdash A$, then $\vdash [\nu_\Sigma]A$		

Definition 4.1. *We write $\vdash A$ if A is an axiom of $\mathcal{DS}$ or A can be inferred from the axioms using the deductive rules of $\mathcal{DS}$. We call $\mathcal{LDS}$ the set of $\mathcal{C}2\mathcal{PDL}$-formulae $\{A | \vdash A\}$.*

4.2 Soundness

Theorem 4.1 (Soundness)**.** *Let A be a $\mathcal{C}2\mathcal{PDL}$ formula, if $\vdash A$ then $\vDash A$.*

We discuss below the case of three axioms, namely the axioms $(-)$, which is not part of Combinatory PDL, $(\Sigma 2)$, which is not part of Converse PDL, and $(\Sigma_2 1)$, which is introduced due to the splitting of the universe. The idea of the proof consists, for a formula A such that $\vdash A$, to show that one can pick any model $\mathcal{M} = (M, R, V, \chi, \mathcal{E})$, any element m of M and prove that $m \in V(A)$. The other cases are treated in [11].

(-) Let m be an element of a model $\mathcal{M}$ then:

- Either $m \in V(A)$ and then $\forall m', ((m, m') \notin R(\alpha)$ or $\exists m'' = m.$ such that $(m', m'') \in R(\alpha^-))$. Thus $m \in V([\alpha]\langle \alpha^- \rangle A)$,
- or $m \notin V(A)$ and thus $m \in V(\neg A)$.

In all cases, $m \in V([\alpha]\langle \alpha^- \rangle A \vee \neg A)$ thus $m \in V(A \Rightarrow [\alpha]\langle \alpha^- \rangle A)$.

($\Sigma 2$) Let m be an element of a model $\mathcal{M}$ then:

- either $\chi(c) \notin V(A)$ and thus $m \in V(\langle \nu_\Sigma \rangle (c \wedge \neg A))$ but then $\forall m''.\ m'' \notin V(c) = \{\chi(c)\}$ or $m'' \notin V(A)$ thus $m \in V([\nu_\Sigma](\neg c \vee \neg A)$. Thus $m \in V([\nu_\Sigma](\neg c \vee \neg A) \wedge \langle \nu_\Sigma \rangle (c \wedge \neg A))$,
- or $\chi(c) \in V(A)$ and thus $\forall m', m' \notin V(c)$ or $m' \in V(A)$ thus $m \in V([\nu_\Sigma](\neg c \vee A)$. But then $\exists m'' = \chi(c)$ such that $m'' \in V(c \wedge A)$ and thus $m \in V(\langle \nu_\Sigma \rangle (c \wedge A))$ thus $m \in V([\nu_\Sigma](\neg c \vee A) \wedge \langle \nu_\Sigma \rangle (c \wedge A))$

In all possible cases, $m \in V(([\nu_\Sigma](\neg c \vee \neg A) \wedge \langle \nu_\Sigma \rangle (c \wedge \neg A)) \wedge ([\nu_\Sigma](\neg c \vee A) \wedge \langle \nu_\Sigma \rangle (c \wedge A)))$ that is $m \in V(\langle \nu_\Sigma \rangle (c \wedge A) \Leftrightarrow [\nu_\Sigma](c \Rightarrow A))$

($\Sigma_2 1$) Let m be a element of a model $\mathcal{M}$, $\phi \in \Phi_0$ then:

- Either $m \in V(c) = \{\chi(c)\}$ and then as $V(\phi) \subseteq \chi(\Sigma_1)$ and $\chi(\Sigma_1) \cap \chi(\Sigma_2) = \emptyset$, $m \notin V(\phi)$ and thus $m \in V(\neg\phi)$,

- or $m \notin V(c)$ and thus $m \in V(\neg c)$.

In all possible cases, $m \in V(\neg c \vee \neg\phi)$ that is $m \in V(c \Rightarrow \neg\phi)$

4.3 Completeness

Theorem 4.2 (Completeness)**.** *Let A be a $\mathcal{C}2\mathcal{PDL}$ formula, if $\vDash A$ then $\vdash A$.*

The completeness proof is much more involved than the soundness proof. The idea is to prove that if $\nvdash A$ then $\nvDash A$, which is obviously equivalent to Theorem 4.2.

The main argument of the proof makes use of the notion of extension of the logic $\mathcal{C}2\mathcal{PDL}$:

Definition 4.2. *A logic (over $\mathcal{DS}$) is any set of $\mathcal{C}2\mathcal{PDL}$ formulae L such that:*

- *L contains all axioms of $\mathcal{DS}$*
- *L is closed under (MP), (Ind), (Cov) and (Nec).*

To establish the completeness, the notion of consistent logics is used.

Definition 4.3. *A logic L is consistent if $\bot \notin L$.*

We can now state the following theorem.

Theorem 4.3. *If L is a consistent logic, then L has a model.*

Let $log(\Gamma, A)$ denote the least logic containing the set of formulae Γ and the formula A. Then we can show that Theorem 4.2 is a consequence of Theorem 4.3. Indeed, assume $\nvdash A$. Then, $log(\mathcal{LDS}, \neg A)$ is consistent. Thus, from Theorem 4.3, $log(\mathcal{LDS}, \neg A)$ has a model. That is $\nvDash A$.

In order to prove Theorem 4.3, one may use the notion of maximal logics and the Lindenbaum lemma.

Definition 4.4. *A logic L is said to be maximal if for all $\mathcal{C}2\mathcal{PDL}$-formulae A, either $A \notin L$ or $A \in L$.*

Lemma 1 (Lindenbaum lemma)**.** *If L is a consistent logic then there exists a maximal consistent logic L^* such that $L \subseteq L^*$.*

The proof of Lemma 1 is quite straightforward. The set of $\mathcal{C}2\mathcal{PDL}$-formulae being recursively enumerable, it is possible to make a list of them, say $\{\phi_1, \phi_2, \dots\}$. Then, starting from $L_0 = L$, we try adding the n-th formula ϕ_n to L_{n-1}. If $log(L_{n-1}, \phi_n)$ is consistent, we define it at L_n, if not $log(L_{n-1}, \neg\phi_n)$ is consistent and it is defined as L_n. The final step consists in proving that the union of all the L_ns is a maximal consistent logic. Details can be found in [11].

In order to build the model for the proof of Theorem 4.3, the main remaining obstacle is that names can occur several times, at different nodes (elements of the universe). Remember that each name can only name one element. Then, to solve this issue, we introduce an equivalence relation over names: $c \sim d = \langle\nu_\Sigma\rangle(c \wedge d) \in L^*$. $[c]_\sim$ is defined as the equivalence class of c. Intuitively, $c \sim d$ if both c and d name the same node. We define $\mathcal{M}_\sim = (M_\sim, R_\sim, \chi_\sim, V_\sim)$, where $M_\sim = \{[c]_\sim | c \in \Sigma\}$, for all $\alpha \in \Pi$, $R_\sim(\alpha) = \{([c]_\sim, [d]_\sim) | \langle\nu_\Sigma\rangle(c \wedge \langle\alpha\rangle d) \in L^*\}$, for all $c \in \Sigma$, $\chi_\sim(c) = [c]_\sim$ and for all $A \in \Phi$, $V_\sim(A) = \{[c]_\sim | \langle\nu_\Sigma\rangle(c \wedge A) \in L^*\}$. One now has to prove that $\mathcal{M}_\sim$ is a model. It is straightforward thanks to the Lemma 1 that allows us to use a maximal logic.

4.4 Decidability

Theorem 4.4 (Decidability)**.** *The validity problem of $\mathcal{C}2\mathcal{PDL}$ is decidable.*

The first step towards proving Theorem 4.4 is to find two semi-decision procedures: one that stops when the formula given as argument is valid and another one that stops if the formula is not valid. As a formula has to be one of the two, the decision procedure will stop.

We start with the validity semi-decision procedure. The deductive system is a good starting point but there are problems with the rules (Ind) and (Cov) as they both quantify on infinite sets (the integers and the names respectively). We thus drop them to form a new logic that generates the same set of valid formulae but whose validity problem is decidable:

Definition 4.5. *Let $\mathcal{FDS}$ be the deductive system obtained from $\mathcal{DS}$ by dropping the rules (Ind) and (Cov) and adding the axiom (ind): $(A \wedge [\alpha^*](A \Rightarrow [\alpha]A)) \Rightarrow [\alpha^*]A$. Let $\vdash_F$ denote provability in $\mathcal{FDS}$. We call $\mathcal{LFDS}$ the set of $\mathcal{C}2\mathcal{PDL}$-formulae $\{A| \vdash_F A\}$.*

As $\mathcal{FDS}$ is ω-rule-free, it is obvious that $\mathcal{LFDS}$ is a recursively enumerable set. We now have to prove that every formula of $\mathcal{LFDS}$ is also a valid formula of $\mathcal{C}2\mathcal{PDL}$.

Lemma 2. *Let A be a formula of $\mathcal{C}2\mathcal{PDL}$, if $\vdash_F A$ then $\vdash A$*

The proof of Lemma 2 simply amounts to proving that $\vdash ind$. It can be found in [11].
The semi-decision procedure that decides whether a formula is valid is based on Lemma 2.
To find an invalidity semi-decision procedure, we use the following theorem.

Theorem 4.5. *Let A be a formula of $\mathcal{C}2\mathcal{PDL}$, if $\nvdash_F A$ then, for some finite model $\mathcal{M}$, $\mathcal{M} \nvDash A$.*

To prove Theorem 4.5 we build a canonical quasi-model.

Definition 4.6. *We name canonical quasi-model the model $\mathcal{M}_c = (M_c, R_c, V_c)$ where:*

- *M_c is the set of all maximal consistent sets of formulae*
- *for every program α and for all $u, v \in M_c$, $u\ R_c(\alpha)\ v$ iff, for every formula A, if $[\alpha]A \in u$ then $A \in v$*
- *for every atomic proposition ϕ, $V_c(\phi) = \{u \in M_c | \phi \in u\}$*
- *for every name i, $V_c(i) = \{u \in M_c | i \in u\}$*

$\mathcal{M}_c$ is still not a model but it is possible to obtain a finite model from it by doing a filtration [22] as defined below:

Definition 4.7. *Let $\mathcal{M} = (M, R, \chi, V)$ be a model and let Γ be any set of formulae closed under sub-formulae. We define the equivalence relation $\sim_\Gamma$ on M by:*
$\forall s, t \in M. s \sim_\Gamma t$ iff $\forall \phi \in \Gamma$, $(s \vDash \phi$ iff $t \vDash \phi)$.

We note $[s]_\Gamma$ the equivalence class of s with respect to $\sim_\Gamma$. The structure $\mathcal{M}_\Gamma = (M_\Gamma, R_\Gamma, \chi_\Gamma, V_\Gamma)$ is called filtration of $\mathcal{M}_c$ with respect to Γ if:

- *$M_\Gamma := \{[s]_\Gamma | s \in M_c\}$*
- *for every program $\alpha \in \Gamma$, if $sR_c(\alpha)t$, then $[s]_\Gamma R_\Gamma(\alpha)[t]_\Gamma$*

- *for every program* $\alpha \in \Gamma$, *if* $[s]_\Gamma R_\Gamma(\alpha)[t]_\Gamma$, *then for all formulae* A, $[\alpha]A \in s \cap \Gamma$ *only if* $A \in t$
- *for every name in* $o \in \Gamma$, *if* $o \in s$, $[s]_\Gamma \in \chi_\Gamma(o)$
- *for every atomic proposition* $\phi_0 \in \Gamma$, $V_\Gamma(\phi_0) = \{[s]_\Gamma | s \in V_c(\phi_0)\}$

The proof that $\mathcal{M}_\Gamma$ is a model is not very involved, as it is a simple check of all the conditions, but it will not be reported here for lack of space. The proof that $\mathcal{M}_\Gamma \not\Vdash A$ is less obvious and it rests mainly on the fact that if $\not\vdash_F A$, then there exists a maximal set of formulae u not containing A. Thus $[u]_\Gamma \not\Vdash A$. The exact definition of u as well as more details can be found in [11].

We can now prove that we obtain that way a finite model.

Lemma 3. *If* Γ *is such that* $|\Gamma| = n$, *where* $|\Gamma|$ *is the cardinality of* Γ *that is the number of formulae it contains, then* $|\mathcal{M}_\Gamma| \leq 2^n$ *where* $|\mathcal{M}_\Gamma|$ *is the number of nodes in* $\mathcal{M}_\Gamma$.

The proof of Lemma 3 is straightforward as there are at most 2^n equivalence classes for n formulae.

It is thus possible to exhibit a finite model $\mathcal{M}$ such that if $\not\vdash_F A$ then $\mathcal{M} \not\Vdash A$ which proves Theorem 4.5.

Theorem 4.5 is used to prove that A is invalid. The procedure tries all finite models and stops when it finds one such that $\not\Vdash A$. On the other hand, Lemma 2 gives us the assurance that, if A is valid, it will be generated eventually by $\mathcal{FDS}$. We can thus decide whether or not a $\mathcal{C}2\mathcal{PDL}$-formula A is valid. As usual, we can also prove that a formula A is satisfiable by proving that $\neg A$ is invalid.

Another possible approach would be to use the Hybrid μ-calculus [21]. This logic has almost the same constructors as $\mathcal{C}2\mathcal{PDL}$ but replaces the closure with the μ and ν constructors of μ calculus. It is known to be decidable. The translation of the closure in μ-calculus is simple and well-known. The only difficulty is the use of the sets in ν_S. This can be tackled by introducing new atomic propositions that label the nodes in S and only them. One also has to had in the formula that nodes named with elements of Σ_2 are such that none of the atomic propositions label them and they have neither incoming nor outgoing edges.

5 Related work

We recall that our goal in defining $\mathcal{C}2\mathcal{PDLS}$ was to introduce a logic that would be both decidable and expressive enough to characterize basic substitutions over graphs. Furthermore, we wanted the logic to be able to speak about named nodes, and thus to contain nominals, and to be able to express reachability, and thus contain the Kleene star.

Some expressive logics have been introduced that are able to deal with actions over graphs. In [3], the authors introduced two ways to extend 'static' modal logics with actions that are similar to ours. The first one allows to modify node and edge labelling globally while the second one modifies them locally. The first one is proven to have the same status w.r.t. completeness and decidability as the original 'static' modal logic. The second one yields undecidable logics. Among the crucial differences between their work and ours is that they allow actions, that they introduce, and programs, that describe the models, to interact while we separate them. Furthermore, their models are rooted in that the exact position at which one is in the graph is key. We do not care on the other hand. The second logic being undecidable does not meet

our requirements and the first one, by allowing only global modifications, prevents us from explicitly stating where the modifications occur.

In [5], the authors introduced a quite expressive logic that extends both logics of [3] by allowing global and local modifications of the labelings. This logic is obviously very expressive but its validity problem is undecidable.

Some approaches introduce actions that are relevant for the subject they deal with but cannot really be used when one tries, as we do, to define graph transformations. Public Announcement Logic [18], for instance, deals with multi-agent epistemic logic and adds operators for public communications of message α that removes from the models all nodes that do not satisfy α. This approach is generalized by van Ditmarsch et al. [23]. In a different direction, Fernandez-Duque et al. [13] introduced an epistemic logic allowing to forget information. These logics target some particular classes of graphs which limit their use for graph transformations in general.

In [6], Balbiani et al. introduced the Dynamic Logic of Propositional Assignment. This logic allows to dynamically assign propositional values. The expressive power of this logic and $\mathcal{C}2\mathcal{PDLS}$ are not comparable in the sense that there are formulae that can be expressed in one logic but not in the other one and the other way round. Furthermore, with $\mathcal{C}2\mathcal{PDLS}$ one may reason about evolving graph structures which is not that obvious in [6].

In [12], we have studied the same substitutions as in the present paper, minus the creation and deletion of nodes and the global redirection of edges, and looked at whether or not some Description Logics are closed under them. Once again, the main goal is to identify logics which are decidable and can also express substitutions.

In [1], Ahmetaj et al. studied the logic $\mathcal{ALCHOIQ}br$. It is an unusual Description Logic that has been extended so that substitutions can be written as part of the formulae. It is then used to solve planning problems. $\mathcal{ALCHOIQ}br$ is finitely decidable but lacks the Kleene star.

One could hope to merge the gap between dynamic logics and description logics to obtain a logic containing both regular role expressions and counting quantifiers. This is particularly interesting as some important graph-like structures are defined using both. The logic that is reached, though, will not be decidable as it has been proven, in [9], that logics with regular role expressions and counting quantifiers are undecidable.

6 Conclusion

We introduced a new extension of dynamic logics, we called $\mathcal{C}2\mathcal{PDLS}$, to define properties of evolving graphs. Our main goal in doing so was to provide a logic able to express two different ideas that are usually associated with the dynamic part of dynamic logics. First, the use of classical constructors of dynamic logics (such as union, intersection, composition, ... of programs) to express complex conditions on the way the various nodes of graphs can be connected. Meanwhile, other constructs borrowed from Hoare logics and called substitutions are used to express conditions on future states of graphs after performing some actions. These differ from the usual programs of dynamic logic in that the effect of each substitution in term of model is well-defined and not left to be chosen when the model is built. In order to avoid confusion, these two dynamic aspects were completely separated out with different notations.

In addition to the usual constructs of dynamic logic, we split the nodes of the considered graphs into two sets: one contains the nodes that are actually part of the graph at the inception of the formulae, the other contains all nodes that do not belong to the graph. This second set of nodes is intended to store nodes that will be created by future transformations.

Once the syntax of $\mathcal{C}2\mathcal{PDLS}$ and its models have been properly defined, we looked at the properties of $\mathcal{C}2\mathcal{PDLS}$. We proved that the substitutions do not increase the expressive power of the logic and provided a rewriting system allowing to translate a formula with substitutions (that is in $\mathcal{C}2\mathcal{PDLS}$) to a substitution-free formula (that is in $\mathcal{C}2\mathcal{PDL}$). We provided a reasoning system that we proved to be both sound and complete. We also proved that the validity (and thus also the satisfiability) problem in $\mathcal{C}2\mathcal{PDL}$ (and consequently in $\mathcal{C}2\mathcal{PDLS}$) is decidable.

An interesting way to continue this line of work would be to extend the substitutions toward a more dynamic structure in which substitutions would not only allow one action to be performed but full programs as is usual in dynamic logic and then to see whether or not it would be possible to make logical programs (those used to model the arcs of the graph) and the graph transformations interact.

References

[1] Shqiponja Ahmetaj, Diego Calvanese, Magdalena Ortiz, and Mantas Simkus. Managing change in graph-structured data using description logics. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 966–973, 2014.

[2] C. Areces, P. Blackburn, and M. Marx. The computational complexity of hybrid temporal logics. *Logic Journal of IGPL*, 8(5):653–679, 2000.

[3] Guillaume Aucher, Philippe Balbiani, Luis Fariñas del Cerro, and Andreas Herzig. Global and local graph modifiers. *Electronic Notes in Theoretical Computer Science*, 231:293 – 307, 2009. Proceedings of the 5th Workshop on Methods for Modalities (M4M5 2007).

[4] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[5] Philippe Balbiani, Rachid Echahed, and Andreas Herzig. A dynamic logic for termgraph rewriting. In *Procs. of the 5th International Conference on Graph Transformations (ICGT)*, volume 6372 of *Lecture Notes in Computer Science*, pages 59–74. Springer, 2010.

[6] Philippe Balbiani, Andreas Herzig, and Nicolas Troquard. Dynamic logic of propositional assignments: A well-behaved variant of PDL. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 143–152, 2013.

[7] Patrick Blackburn, Johan F. A. K. van Benthem, and Frank Wolter. *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning)*. Elsevier Science Inc., New York, NY, USA, 2006.

[8] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2001. Cambridge Books Online.

[9] Piero A. Bonatti. On the undecidability of description and dynamic logics with recursion and counting. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 331–336, 2003.

[10] Jon Haël Brenas, Rachid Echahed, and Martin Strecker. A hoare-like calculus using the SROIQ σ logic on transformations of graphs. In *Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*, pages 164–178, 2014.

[11] Jon Hael Brenas, Rachid Echahed, and Martin Strecker. A combination of combinatory and converse PDL with substitutions. 2016. http://lig-membres.imag.fr/echahed/c2pdls.pdf.

[12] Jon Haël Brenas, Rachid Echahed, and Martin Strecker. On the closure of description logics under substitutions. In *Proceedings of the 29th International Workshop on Description Logics, Cape Town, South Africa, April 22-25, 2016.*, 2016.

[13] David Fernández Duque, Ángel Nepomuceno-Fernández, Enrique Sarrión-Morillo, Fernando Soler-Toscano, and Fernando R. Velázquez-Quesada. Forgetting complex propositions. *CoRR*, abs/1507.01111, 2015.

[14] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.

[15] David Harel, Jerzy Tiuryn, and Dexter Kozen. *Dynamic Logic*. MIT Press, Cambridge, MA, USA, 2000.

[16] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.

[17] Solomon Passy and Tinko Tinchev. An essay in combinatory dynamic logic. *Inf. Comput.*, 93(2):263–332, 1991.

[18] Jan Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007.

[19] Christopher M. Poskitt and Detlef Plump. Verifying monadic second-order properties of graph programs. In *Graph Transformation - 7th International Conference, ICGT 2014, Held as Part of STAF 2014, York, UK, July 22-24, 2014. Proceedings*, pages 33–48, 2014.

[20] Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific, 1997.

[21] Ulrike Sattler and Moshe Y. Vardi. The hybrid μ-calculus. In *Proceedings of the First International Joint Conference on Automated Reasoning*, IJCAR '01, pages 76–91, London, UK, UK, 2001. Springer-Verlag.

[22] K. Segerberg. A completeness theorem in the modal logic of programs. In *Universal algebra and applications*, pages 31–46. PWN-Polish Scientific Publishers, 1982.

[23] Hans P. van Ditmarsch, Wiebe van der Hoek, and Barteld P. Kooi. Dynamic epistemic logic and knowledge puzzles. In *Conceptual Structures: Knowledge Architectures for Smart Applications, 15th International Conference on Conceptual Structures, ICCS 2007, Sheffield, UK, July 22-27, 2007, Proceedings*, pages 45–58, 2007.