A Dynamic Logic of Normative Systems

Andreas Herzig
IRIT-CNRS, France
herzig@irit.fr

Emiliano Lorini
IRIT-CNRS, France
lorini@irit.fr

Frédéric Moisan
IRIT, France
moisan@irit.fr

Nicolas Troquard
University of Essex, UK
ntroqu@essex.ac.uk

Abstract

We propose a logical framework to represent and reason about agent interactions in normative systems. Our starting point is a dynamic logic of propositional assignments whose satisfiability problem is PSPACE-complete. We show that it embeds Coalition Logic of Propositional Control CL-PC and that various notions of ability and capability can be captured in it. We illustrate it on a water resource management case study. Finally, we show how the logic can be easily extended in order to represent constitutive rules which are also an essential component of the modelling of social reality.

1 Introduction

We show that dynamic logic of propositional assignment PDL-PA provides a logical framework for normative systems, with concrete models to design rich social situations and an intuitive language to talk about institutional facts. The basic idea is that agents have a repertoire of actions to set propositional variables p to true ($+p$) or to false ($-p$). These assignments are combined by means of the standard PDL program operators of test, sequential and nondeterministic composition and iteration. We will show that PDL-PA embeds Coalition Logic of Propositional Control CL-PC [Hoek and Wooldridge, 2005].

We implement in PDL-PA an agentive and a normative dimension by supposing that the logic has special propositional variables $\mathrm{A}_i(\alpha)$ and $\mathrm{P}_i(\alpha)$, where i is an agent and α is an assignment. $\mathrm{A}_i(\alpha)$ read as "i is able to perform α" and $\mathrm{P}_i(\alpha)$ as "i is allowed to perform α". This allows to distinguish between practical possibilities (*alias* abilities) and deontic possibilities (*alias* permissions, authorizations): an agent may be able to make a propositional variable q true, but not allowed to, and the other way round. This distinction also shows up when it comes to dynamics: we have *change of abilities* and *change of permissions*. An example of the former is when the president of a university gives some amount of resources to a department. An example of the latter is when he authorizes the department to spend that money for a particular purpose. Abilities and permissions may evolve, too; We capture this in our language by assignments such as $+\mathrm{A}_i(+p)$, $+\mathrm{P}_i(-p)$, etc.

In the logic PDL-PA we may moreover reason about *higher-order* abilities and permissions: agents may be able or allowed to change abilities or permissions. For example the president of the university i has the permission to authorize the vice-president j to sign a document in the name of the institution. In our language this can be written $\mathrm{P}_i(+\mathrm{P}_j(+s))$, where s denotes that the document is signed. Assignments may again modify these abilities and permissions. Beyond the (ontic and deontic) possibility to perform an action the logic PDL-PA allows to talk about the (ontic and deontic) possibility to attain a certain state of affairs. We call this the *capability* of an agent or a coalition of agents. (Our distinction between 'ability' and 'capability' is therefore: ability is about agents' actions, whereas capability is about outcomes that an agent can ensure.)

We introduce the basic logic PDL-PA in Section 2 and prove that satisfiability checking is PSPACE-complete. In Section 3 we embed CL-PC into PDL-PA by using a basic notion of capability. In Section 4 we provide a general analysis of different kinds of ontic and deontic abilities and capabilities (possibly of higher order). In Section 5 we provide a case study of water resource management. Finally, in Section 6 we extend PDL-PA in order to represent constitutive rules [Searle, 1969; 1995], that are essential when modelling social reality. To that end we add a binary connective $\rightsquigarrow$ to the language whose arguments are assignments and where $\alpha_1 \rightsquigarrow \alpha_2$ reads "α_1 *counts as* α_2". We prove that satisfiability checking within that extension of PDL-PA has the same complexity as in PDL-PA. We conclude in Section 7 and discuss research avenues for future work.

2 Dynamic logic of propositional assignment

This section introduces the syntax and the semantics of the logic PDL-PA. It is basically an instantiation of propositional dynamic logic PDL [Harel *et al.*, 2000] with concrete programs to assign propositional variables to true or false. We will show how it allows to represent and reason about abilities, permissions, and changes of abilities and permissions in multiagent systems (MASs).

2.1 Language

Assignments are expressions of the form $+p$ or $-p$ and modify the truth value of the propositional variable p: $+p$ sets p to true, and $-p$ sets p to false.

Throughout the paper, $\mathbb{P} = \{p, q, \dots\}$ is a countable set of propositional variables. The language of PDL-PA is made up of events π and formulas φ and is defined by the following BNF:

$$\begin{array}{lll} \pi & ::= & +p \mid -p \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid \varphi? \\ \varphi & ::= & p \mid \top \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle\varphi \end{array}$$

where p ranges over $\mathbb{P}$.

Just as in PDL, $\pi_1;\pi_2$ is sequential composition, $\pi_1 \cup \pi_2$ is nondeterministic composition, π^* is iteration, and $\varphi?$ is the test of φ. The formula $\langle\pi\rangle\varphi$ reads "there is an execution of π such that φ is true afterwards".

The set of all assignments is $\mathcal{A} = \{+p : p \in \mathbb{P}\} \cup \{-p : p \in \mathbb{P}\}$. $\alpha, \alpha', \dots$ range over $\mathcal{A}$. The set of all formulas is noted $\mathcal{F}$, and the set of all events is noted $\mathcal{E}$. We define $\mathbb{P}_\varphi$ and $\mathbb{P}_\pi$ to be the set of variables from $\mathbb{P}$ occurring in φ (resp. π).

The *length* of a formula φ, noted $|\varphi|$, is the number of symbols used to write down φ (without $\langle$, $\rangle$, and parentheses), where the length of a propositional variable is 1. For example $|\langle +p\rangle(p \vee q)| = 2 + 3 = 5$.

The logical connectives $\wedge$, $\rightarrow$ and $\leftrightarrow$ have the usual meaning. The formula $[\pi]\varphi$ abbreviates $\neg\langle\pi\rangle\neg\varphi$. The event `skip` abbreviates $\top?$ ("nothing happens").

2.2 Semantics

Definition 1 *A valuation V is a subset of $2^{\mathbb{P}}$.*

The truth conditions are the customary ones for $\top$, $\bot$, negation and disjunction. The dynamic operators are interpreted by means of an update relation on valuations.

$$\begin{array}{lll} V \models p & \text{iff} & p \in V \\ V \models \langle\pi\rangle\varphi & \text{iff} & \text{there is a } V' \text{ such that } VR_\pi V' \text{ and } V' \models \varphi \end{array}$$

R_π is a binary relation on valuations that allows to interpret events and that is defined together with $\models$ by mutual recursion:

$$\begin{array}{lll} R_{+p} & = & \{(V, V \cup \{p\}) : V \text{ is a valuation }\} \\ R_{-p} & = & \{(V, V \setminus \{p\}) : V \text{ is a valuation }\} \\ R_{\pi_1;\pi_2} & = & R_{\pi_1} \circ R_{\pi_2} \\ R_{\pi_1 \cup \pi_2} & = & R_{\pi_1} \cup R_{\pi_2} \\ R_{\pi^*} & = & (R_\pi)^* \\ R_{\varphi?} & = & \{(V, V) : V \models \varphi\} \end{array}$$

A formula φ is satisfiable if and only if $V \models \varphi$ for some valuation V, and φ is valid if and only if $\neg\varphi$ is not satisfiable.

2.3 Complexity

The following can be proved by adapting the proofs of [Hoek and Wooldridge, 2005] and [Hoek *et al.*, 2010].

Theorem 1 *The problem of PDL-PA model checking is PSPACE-complete.*

Proof (sketch). Let us first establish that model checking is PSPACE-hard. We reduce QSAT into the problem of model checking PDL-PA. Consider a quantified Boolean formula

$$\Phi = \exists x_1 \forall x_2 \cdots \exists x_{m-1} \forall_m x_m \, \varphi$$

where φ is a propositional formula in CNF containing no variables other than $x_1, \dots, x_m$. (If necessary an even m can be obtained by adding a dummy variable x_m.) Let $V = \emptyset$ be the PDL-PA valuation over the set of propositional variables $\mathbb{P} = \{x_1, \dots, x_m\}$ that sets all variables to false. Let

$$\begin{array}{lll} \Phi^{\mathsf{PDL\text{-}PA}} & = & \langle +x_1 \cup -x_1\rangle[+x_2 \cup -x_2] \cdots \\ & & \langle +x_{m-1} \cup -x_{m-1}\rangle[+x_m \cup -x_m]\, \varphi \end{array}$$

It is readily checked that the QBF Φ is true iff $V \models \Phi^{\mathsf{PDL\text{-}PA}}$. Since both the size of $\Phi^{\mathsf{PDL\text{-}PA}}$ and the size of the model are linear in the size of Φ, we conclude that model checking is PSPACE-hard.

The proof of *membership* requires the following recursive definition of the set of sequences of atomic events that are *admitted* by a complex event π.

$$\begin{array}{lll} adm(+p) & = & \{+p\} \\ adm(-p) & = & \{-p\} \\ adm(\pi_1;\pi_2) & = & \{\alpha_1;\alpha_2 : \alpha_1 \in adm(\pi_1) \,\&\, \alpha_2 \in adm(\pi_2)\} \\ adm(\pi_1 \cup \pi_2) & = & adm(\pi_1) \cup adm(\pi_2) \\ adm(\pi^*) & = & \{\alpha_1;\dots;\alpha_k : k \geq 0 \,\&\, \alpha_1, \dots, \alpha_k \in adm(\pi)\} \\ adm(\varphi?) & = & \{\varphi?\} \end{array}$$

In the fifth line we suppose $\alpha_1;\dots;\alpha_k$ is `skip` when $k = 0$.

The main point in the proof is that every possible update of V by a complex event π can also be reached by a sequence that is admitted by π and that is of length at most exponential in the length of π. Precisely, it can be reached in at most $2^{|\pi|^2}$ steps. This is the case despite the presence of the arbitrary iteration operator *, the reason being that any π^* can only bring about a finite number of valuation updates.

In detail, the proof is by induction on the structure of π. The only nontrivial case is $\pi = \sigma^*$. Suppose $VR_{\sigma^*}V'$. Then there are $V_0, \dots, V_n$ such that $V_0 = V$, $V_n = V'$ and $V_{k-1}R_\sigma V_k$ for every $k \leq n$. As R_{σ^*} is transitive we may suppose that $V_k \neq V_l$ for $k \neq l$, i.e. there are no loops. By induction hypothesis σ admits $\alpha_1, \dots \alpha_n$ such that $V_{k-1}R_{\alpha_k}V_k$ and such that for every k, $|\alpha_k| \leq 2^{|\pi|^2}$. The sequence $\alpha = \alpha_1;\dots;\alpha_n$ is admitted by π. All valuations reachable from V differ from V by the values of the variables in $\mathbb{P}_\pi$. Hence there are at most $2^{Card(\mathbb{P}_\pi)} \leq 2^{|\pi|}$ such models. Notice that n cannot exceed $2^{Card(\mathbb{P}_\pi)}$; otherwise the sequence $V_0, \dots, V_n$ would contain loops (contradicting the previous assumption). We have:

$$\begin{array}{lll} |\alpha| & \leq & n \times \max_{k \leq n}\{|\alpha_k|\} + n - 1 \\ & \leq & 2^{|\pi|} \times 2^{|\pi|^2} + 2^{|\pi|} \\ & \leq & 2^{(|\pi|+1)^2} \\ & \leq & 2^{(|\pi^*|)^2} \end{array}$$

Based on that one can prove that membership of a couple of valuations (V, V') in R_π can be decided in polynomial space. Finally one can prove that a formula φ can be evaluated in space polynomial in the size of φ; in particular, when evaluating $\langle\pi^*\rangle\varphi$ one may decide in polynomial space whether one of the (at most $2^{(|\pi^*|)^2}$) valuations V' is accessible from V. ■

Theorem 2 *The problem of PDL-PA satisfiability checking is PSPACE-complete.*

Proof (sketch). Just as for model checking, hardness can be proved by encoding QBFs.

Membership can be proved as follows: given φ we guess a valuation V. A valuation of size polynomial in φ exists because we can always restrict our attention to the propositional variables occurring in φ. Then we check whether $V \models \varphi$. By Theorem 1 this can be done in polynomial time. Hence PDL-PA satisfiability can be checked in NPSPACE = PSPACE. ■

3 Embedding CL-PC

We now show that PDL-PA generalises Coalition Logic of Propositional Control CL-PC [Hoek and Wooldridge, 2005].

Let $\mathbb{A} = \{i, j, \ldots\}$ be a finite set of agents. A *coalition* is a subset of agents $J \subseteq \mathbb{A}$. We define $\mathbb{A}_\varphi$ and $\mathbb{A}_\pi$ to be the set of agents from $\mathbb{A}$ occurring in formula φ (resp. event π).

We suppose that beyond a set of basic propositional variables $\mathbb{P}^0$, for every $i \in \mathbb{A}$ and $p \in \mathbb{P}^0$ there are propositional variables $\mathrm{A}_i(+p)$ and $\mathrm{A}_i(-p)$ denoting that an agent i has the ability to perform $+p$ and $-p$. Assignments of variables in $\mathbb{P}^0$ are called *basic*. Hence

$$\begin{aligned}\mathbb{P} \quad = \quad & \mathbb{P}^0 \cup \{\mathrm{A}_i(+p) \ : \ i \in \mathbb{A} \text{ and } p \in \mathbb{P}^0\} \\ & \cup \{\mathrm{A}_i(-p) \ : \ i \in \mathbb{A} \text{ and } p \in \mathbb{P}^0\}\end{aligned}$$

Let us extend the language of PDL-PA by modal operators of ontic capability $\Diamond_J^{\mathrm{A}^0}$, one for every coalition $J \subseteq 2^{\mathbb{A}}$. The formula $\Diamond_J^{\mathrm{A}^0}\varphi$ reads "the agents in J are able to achieve φ by appropriately performing the actions in their repertoires". The truth condition is:

$$\begin{aligned}V \models \Diamond_J^{\mathrm{A}^0}\varphi \quad \text{iff} \quad & \text{there are } \textit{basic} \text{ assignments } \alpha_1, \ldots, \alpha_n \text{ s.t.} \\ & V \models \langle \alpha_1; \ldots; \alpha_n \rangle \varphi \text{ and for every } \alpha_k \\ & \text{there is } i \in J \text{ with } \mathrm{A}_i(\alpha_k) \in V\end{aligned}$$

The *language of CL-PC* is built from $\mathbb{P}^0$ by means of the Boolean connectives plus operators $\Diamond_J^{\mathrm{A}^0}$. The class of *CL-PC models* is the set of PDL-PA models satisfying:

1. $\mathrm{A}_i(+p) \in V$ iff $\mathrm{A}_i(-p) \in V$ (symmetry);
2. for every $\alpha \in \mathcal{A}$ there is $i \in \mathbb{A}$ such that $\mathrm{A}_i(\alpha) \in V$ (exhaustivity);
3. for every $\alpha \in \mathcal{A}$, if $\mathrm{A}_i(\alpha) \in V$ and $\mathrm{A}_j(\alpha) \in V$ then $i = j$ (exclusivity).

Theorem 3 *A CL-PC formula φ is CL-PC satisfiable iff $Sym_\varphi \wedge Exh_\varphi \wedge Excl_\varphi \wedge \varphi$ is PDL-PA satisfiable, where:*

$$\begin{aligned}Sym_\varphi \quad &= \quad \textstyle\bigwedge_{i \in \mathbb{A}_\varphi, p \in \mathbb{P}_\varphi} \mathrm{A}_i(+p) \leftrightarrow \mathrm{A}_i(-p) \\ Exh_\varphi \quad &= \quad \begin{cases} \bigwedge_{p \in \mathbb{P}_\varphi} \bigvee_{i \in \mathbb{A}} \mathrm{A}_i(-p) & \textit{when } \mathbb{A}_\varphi = \mathbb{A} \\ \bigwedge_{p \in \mathbb{P}_\varphi} \bigvee_{i \in \mathbb{A}_\varphi \cup \{j_0\}} \mathrm{A}_i(-p) & \textit{when } \exists j_0 \in \mathbb{A} \setminus \mathbb{A}_\varphi \end{cases} \\ Excl_\varphi \quad &= \quad \textstyle\bigwedge_{i,j \in \mathbb{A}_\varphi, i \neq j, p \in \mathbb{P}_\varphi} \neg\big(\mathrm{A}_i(-p) \wedge \mathrm{A}_j(-p)\big)\end{aligned}$$

The case analysis for Exh_φ accounts for the case where a variable in φ is controlled by an agent not occurring in φ. Only the case of assignments $-p$ is mentioned in Exh_φ and $Excl_\varphi$; the case $+p$ follows by Sym_φ. We observe that the formulations of the three properties on CL-PC models all have length quadratic in the length of φ.

The above theorem says that if we extended our language by modal operators $\Diamond_J^{\mathrm{A}^0}$ then we could do the same kind of reasoning as in CL-PC. The next theorem says that actually we do not need to extend our language because $\Diamond_J^{\mathrm{A}^0}$ can already be defined in the language of PDL-PA.

Theorem 4 *Let φ be a formula in the language of PDL-PA extended by modal operators $\Diamond_J^{\mathrm{A}^0}$, for every $J \subseteq 2^{\mathbb{A}}$. Let $\mathbb{P}_\varphi = \{p_1, \ldots, p_n\}$. Then the formula $\Diamond_J^{\mathrm{A}^0}\varphi$ is PDL-PA equivalent to*

$$\begin{aligned}&\langle \mathtt{skip} \cup (\textstyle\bigvee_{i \in J} \mathrm{A}_i(+p_1)?; +p_1) \cup (\bigvee_{i \in J} \mathrm{A}_i(-p_1)?; -p_1)\rangle \ldots \\ &\langle \mathtt{skip} \cup (\textstyle\bigvee_{i \in J} \mathrm{A}_i(+p_n)?; +p_n) \cup (\bigvee_{i \in J} \mathrm{A}_i(-p_n)?; -p_n)\rangle\, \varphi\end{aligned}$$

In the above formula, for every variable p_k occurring in φ there is a nondeterministic event which either does nothing (`skip`) or checks whether one of the agents in J can modify p_k and does so. Observe that the length of that formula is linear in the length of φ. Therefore our language is at least as succinct as that of CL-PC.

The logic DCL-PC extends CL-PC by delegation events $i \overset{p}{\leadsto} j$ [Hoek *et al.*, 2010] of i transferring to j his control over p. Such an event corresponds to the following event in PDL-PA:

$$\mathrm{A}_i(+p)?; \mathrm{A}_i(-p)?; -\mathrm{A}_i(+p); -\mathrm{A}_i(-p); +\mathrm{A}_j(+p); +\mathrm{A}_j(-p)$$

4 Ability, permission and capability

In this section we study various forms of ability, permission and capability. We first consider higher-order ability and permission allowing to express that an agent is able (resp. authorized) to perform a certain assignment (possibly of a propositional variable about ability or permission). We then consider the capability of an agent or a coalition of agents to ensure some outcome, distinguishing ontic and deontic capability.

4.1 Higher-order ability and permission

Let us suppose that for every $p \in \mathbb{P}$ and $i \in \mathbb{A}$, the set $\mathbb{P}$ not only contains the propositional variables $\mathrm{A}_i(+p)$ and $\mathrm{A}_i(-p)$ that we have introduced in the preceding section, but also $\mathrm{P}_i(+p)$ and $\mathrm{P}_i(-p)$. $\mathrm{P}_i(+p)$ is read "i is authorized to make p true" (or "i has the permission to make p true"), and similarly for $\mathrm{P}_i(-p)$. Contrarily to the preceding section, though, we apply these constructions *recursively*, considering that p itself may be a special propositional variable. We can now express e.g. that i is allowed to enable j to set p to false, written $\mathrm{P}_i(+\mathrm{A}_j(-p))$.

Assignments of the special propositional variables allow to model change of first-order abilities and permissions as basic assignments such as $+\mathrm{A}_i(+p)$ and $+\mathrm{P}_i(+p)$, etc. For example, $+\mathrm{A}_i(+p)$ makes that i can perform the action of making p true, and $+\mathrm{A}_i(-p)$ makes that i can perform the action of making p false; $-\mathrm{A}_i(+p)$ makes that i cannot make p true, and $-\mathrm{A}_i(-p)$ likewise, $+\mathrm{P}_i(+p)$ makes that i is authorized to perform the action of making p true, and so on. Beyond, assignments also allow to model change of higher-order abilities and permissions. For instance, $+\mathrm{A}_i(+\mathrm{A}_j(+p))$ makes that i can enable j to perform a certain action. For example, if i wins \$100,000 in the lottery then i can give to another agent j the ability to buy a new car (by giving a part of the prize to j). $+\mathrm{P}_i(+\mathrm{P}_j(+p))$ makes that i is authorized to authorize j to perform a certain action. For example, suppose i is nominated president by the parliament. Then, *qua* president, i is authorized to authorize (by the act of promulgation) the minister of justice to start the process of publishing a new law in the official journal of record. $+\mathrm{P}_i(-\mathrm{A}_j(+p))$ makes that i is authorized to hinder j to perform a certain action. For example, suppose i is nominated judge at some court. Then, *qua* judge of the court, i is authorized to sentence some defendant j to imprisonment and, consequently, to deprive j of his freedom.

Our language accounts for the notion of normative system $\mathcal{NS}_V$ defined as a set of permissions about the agents' behavior which belong to the valuation V, that is,

$$\begin{aligned}\mathcal{NS}_V \quad = \quad & \{\mathrm{P}_i(+p) \in V \;:\; i \in \mathbb{A} \text{ and } p \in \mathbb{P}\} \cup \\ & \{\mathrm{P}_i(-p) \in V \;:\; i \in \mathbb{A} \text{ and } p \in \mathbb{P}\}\end{aligned}$$

As the set of prohibitions is the complement of the set of permissions, a normative system can also be seen as a set of prohibitions about the agents' behavior. This definition of normative system therefore matches the definition accepted by several authors in the area of MASs according to whom a normative system is a set of constraints on the agents' behaviour, which may or may not be followed by the agents; see e.g. [Ågotnes *et al.*, 2007; Shoham and Tennenholtz, 1996].

4.2 Capabilities beyond CL-PC

In Section 3 we have captured a notion of basic capability as expressible in CL-PC where one quantifies only about physical actions, viz. the assignment of basic propositional variables in $\mathbb{P}^0$. The operator $\Diamond_J^{\mathrm{A}^0}$ can be generalised to an operator $\Diamond_J^{\mathrm{A}}$ quantifying over *all* possible assignments $\mathcal{A}$:

$$\begin{array}{lll} V \models \Diamond_J^{\mathrm{A}}\varphi & \text{iff} & \text{there are assignments } \alpha_1, \ldots, \alpha_n \in \mathcal{A} \text{ s.t.} \\ & & V \models \langle \alpha_1; \ldots; \alpha_n \rangle \varphi \text{ and for every } \alpha_k \\ & & \text{there is } i \in J \text{ with } \mathrm{A}_i(\alpha_k) \in V \end{array}$$

Similarly, one may extend the language of PDL-PA by modal operators of deontic capability $\Diamond_J^{\mathrm{P}}$, one for every coalition $J \subseteq 2^{\mathbb{A}}$. $\Diamond_J^{\mathrm{P}}\varphi$ reads "the agents in J are allowed to achieve φ by legally performing the actions in their repertoires". In the same spirit we introduce a combined operator $\Diamond_J^{\mathrm{AP}}$, with the obvious reading. The truth conditions are:

$$\begin{array}{lll} V \models \Diamond_J^{\mathrm{P}}\varphi & \text{iff} & \text{there are assignments } \alpha_1, \ldots, \alpha_n \in \mathcal{A} \text{ s.t.} \\ & & V \models \langle \alpha_1; \ldots; \alpha_n \rangle \varphi, \text{ and for every } \alpha_k \\ & & \text{there is } i \in J \text{ with } \mathrm{P}_i(\alpha_k) \in V \\ V \models \Diamond_J^{\mathrm{AP}}\varphi & \text{iff} & \text{there are assignments } \alpha_1, \ldots, \alpha_n \in \mathcal{A} \text{ s.t.} \\ & & V \models \langle \alpha_1; \ldots; \alpha_n \rangle \varphi, \text{ and for every } \alpha_k \\ & & \text{there is } i \in J \text{ with } \mathrm{A}_i(\alpha_k), \mathrm{P}_i(\alpha_k) \in V \end{array}$$

These operators will be handy in order to express capabilities of agents and coalitions of agents. Just as for $\Diamond_J^{\mathrm{A}}$ they neither add expressiveness nor succinctness: there are formulas similar to that of Theorem 4 allowing to rewrite $\Diamond_J^{\mathrm{P}}\varphi$ and $\Diamond_J^{\mathrm{AP}}\varphi$ to PDL-PA formulas of length polynomial in the length of φ. Therefore complexity of reasoning with such operators stays in PSPACE.

5 An example: managing water resources

This section illustrates an application of the logic PDL-PA in a concrete normative system of water management.

There are two farmers f_1 and f_2 working in a certain area close to a town called Thirstytown who need water in order to irrigate their fields. In this area there are three different exploitable water basins 1, 2 and 3. Only water basins 1 and 2 can be safely used by the farmers; basin 3 provides drinkable water to the population of Thirstytown, and if it is exploited for irrigation then Thirstytown will fall short of water. There are two other actors in this scenario: c is the chief of the Water Authority which has the jurisdiction over the area, and p is a local policeman working in Thirstytown.

The propositional variables $\{b_1, b_2, b_3\}$ indicate whether the level of water in a given basin is high or low: b_1 means that "the level of water in the basin 1 is high", $\neg b_1$ means that "the level of water in the basin 1 is low", etc. Furthermore, for every farmer $i \in \{f_1, f_2\}$ and for every propositional variable b_h with $h \in \{1, 2, 3\}$, $\mathrm{A}_i(-b_h)$ expresses that basin h is physically connected to the field of farmer i so that i is able to exploit the water of the basin h, and $\mathrm{P}_i(-b_h)$ expresses that i is authorized to exploit the water of basin h.

$$\boxed{\begin{aligned}\varphi_1 &= \textstyle\bigwedge_{i\in\{f_1,f_2\}}(\mathrm{P}_i(-b_1) \wedge \mathrm{P}_i(-b_2) \wedge \neg\mathrm{P}_i(-b_3)) \\ \varphi_2 &= \mathrm{A}_{f_1}(-b_1) \wedge \mathrm{A}_{f_1}(-b_2) \wedge \neg\mathrm{A}_{f_1}(-b_3) \\ \varphi_3 &= \neg\mathrm{A}_{f_2}(-b_1) \wedge \neg\mathrm{A}_{f_2}(-b_2) \wedge \mathrm{A}_{f_2}(-b_3) \\ \varphi_4 &= \textstyle\bigwedge_{i\in\{f_1,f_2\},h\in\{1,2,3\}}(\mathrm{A}_c(+\mathrm{P}_p(+\mathrm{A}_i(-b_h))) \wedge \mathrm{A}_c(+\mathrm{P}_p(-\mathrm{A}_i(-b_h)))) \\ \varphi_5 &= \textstyle\bigwedge_{i\in\{f_1,f_2\},h\in\{1,2,3\}}(\mathrm{P}_c(+\mathrm{P}_p(+\mathrm{A}_i(-b_h))) \wedge \mathrm{P}_c(+\mathrm{P}_p(-\mathrm{A}_i(-b_h)))) \\ \varphi_6 &= \textstyle\bigwedge_{i\in\{f_1,f_2\},h\in\{1,2,3\}}(\mathrm{A}_c(+\mathrm{A}_i(-b_h)) \wedge \mathrm{A}_p(-\mathrm{A}_i(-b_h)) \wedge \\ &\qquad\qquad \neg\mathrm{P}_p(+\mathrm{A}_i(-b_h)) \wedge \neg\mathrm{P}_p(-\mathrm{A}_i(-b_h)))\end{aligned}}$$

Table 1: Formulas describing the initial situation.

We suppose that the initial situation *INIT* is defined by the conjunction of the following six facts, characterized by the formulas $\varphi_1, \ldots, \varphi_6$ of Table 1:

1. The farmers can legally only exploit basins 1 and 2 but not basin 3 (formula φ_1).
2. Farmer f_1 has access to basins 1 and 2, but not to basin 3 (formula φ_2).
3. Farmer f_2 has access to basin 3, but not to basin 1 and 2 (formula φ_3).
4. The chief of the Water Authority is *able* to authorize the policeman of Thirstytown to add and remove connections between basins and fields (imagine that a connection between a basin and a field can be added or removed by opening or closing a sluice gate) (formula φ_4).
5. The chief of the Water Authority is *authorized* to authorize the policeman of Thirstytown to add and remove connections between basins and fields (formula φ_5).
6. The policeman of Thirstytown is able to add and remove the connections between basins and fields, but is not authorized to do that (formula φ_6).

The problem to be solved is to find an executable and authorized procedure such that, given the initial situation *INIT*, it ensures that the situation *GOAL* will be achieved, where *GOAL* is defined by the conjunction of the following two facts ψ_1 and ψ_2.

1. Every farmer has exclusive control over at least one basin in $\{1, 2\}$: each farmer's field is connected to at least one water basin in $\{1, 2\}$ which is not connected to the other farmer's field. Formally:

$$\textstyle\bigvee_{h\in\{1,2\}}(\mathrm{A}_{f_1}(-b_h) \wedge \neg\mathrm{A}_{f_2}(-b_h)) \wedge \bigvee_{h\in\{1,2\}}(\mathrm{A}_{f_2}(-b_h) \wedge \neg\mathrm{A}_{f_1}(-b_h))$$

2. No field is connected to basin 3: $\neg\mathrm{A}_{f_1}(-b_3) \wedge \neg\mathrm{A}_{f_2}(-b_3)$.

Let $\alpha_{i,h}^{+} = +\mathrm{P}_p(+\mathrm{A}_i(-b_h))$ and $\alpha_{i,h}^{-} = +\mathrm{P}_p(-\mathrm{A}_i(-b_h))$, and $\pi_{i,h} = \alpha_{i,h}^{+}; \alpha_{i,h}^{-}$ be the event of the policeman getting authorization to connect/disconnect the field of farmer i to basin h.

Now, consider the following sequence of events.

1. The chief of the Water Authority authorizes the policeman to modify the connections between basins and fields. Formally: $\pi_1 =$

$(\mathrm{A}_c(\alpha^+_{f_1,1}) \wedge \mathrm{P}_c(\alpha^+_{f_1,1}) \wedge \mathrm{A}_c(\alpha^-_{f_1,1}) \wedge \mathrm{P}_c(\alpha^-_{f_1,1}))?; \pi_{f_1,1}\ ;\ \ldots\ ;$
$(\mathrm{A}_c(\alpha^+_{f_2,3}) \wedge \mathrm{P}_c(\alpha^+_{f_2,3}) \wedge \mathrm{A}_c(\alpha^-_{f_2,3}) \wedge \mathrm{P}_c(\alpha^-_{f_2,3}))?; \pi_{f_2,3}$

2. The policeman removes the connection between basin 1 and f_1's field. Formally: $\pi_2 =$
   $(\mathrm{A}_p(-\mathrm{A}_{f_1}(-b_1)) \wedge \mathrm{P}_p(-\mathrm{A}_{f_1}(-b_1)))?; -\mathrm{A}_{f_1}(-b_1)$
3. The policeman adds the connection between basin 1 and f_2's field. Formally: $\pi_3 =$
   $(\mathrm{A}_p(+\mathrm{A}_{f_2}(-b_1)) \wedge \mathrm{P}_p(+\mathrm{A}_{f_2}(-b_1)))?; +\mathrm{A}_{f_2}(-b_1)$
4. The policeman removes the connection between basin 3 and f_2's field. Formally: $\pi_4 =$
   $(\mathrm{A}_p(-\mathrm{A}_{f_2}(-b_3)) \wedge \mathrm{P}_p(-\mathrm{A}_{f_2}(-b_3)))?; -\mathrm{A}_{f_2}(-b_3)$

It can be shown that the formula

$$INIT \rightarrow \langle \pi_1; \pi_2; \pi_3; \pi_4 \rangle GOAL$$

is PDL-PA valid: given the initial situation *INIT*, the sequence of events $\pi_1; \pi_2; \pi_3; \pi_4$ is an executable and authorized (legal) procedure which ensures that *GOAL* will be achieved. We may also quantify over an agent's actions: the valid formula

$$INIT \rightarrow \Diamond^{\mathrm{A}}_{\{p\}}(\neg \mathrm{A}_{f_1}(-b_3) \wedge \neg \mathrm{A}_{f_2}(-b_3))$$

expresses that the policeman has the ontic capability to guarantee that no farmer has access to basin 3.

The formula $INIT \rightarrow \Diamond^{\mathrm{AP}}_{\{c\}} GOAL$ however is invalid: the chief cannot ensure *GOAL* alone. In contrast, the formula $INIT \rightarrow \Diamond^{\mathrm{AP}}_{\{c,p\}} GOAL$ is valid, meaning that the policeman and the chief can cooperate to achieve the goal.

6 Integrating constitutive rules

According to several authors working in legal theory and in the field of normative MASs (see, e.g., [Alchourrón and Bulygin, 1971; Boella and Torre, 2004]), normative systems are based both on regulative as well as constitutive (i.e. non-regulative) components. That is, normative systems are not only defined in terms of sets of permissions, obligations, and prohibitions (i.e. *norms of conduct*) but also in terms of rules which specify and create new forms of behavior and concepts. According to Searle for instance "[. . .] regulative rules regulate antecedently or independently existing forms of behavior [. . .]. But constitutive rules do not merely regulate, they create or define new forms of behavior" [Searle, 1969, p. 33]. In Searle's theory [Searle, 1969; 1995], constitutive rules are expressed by means of "counts-as" assertions of the form "*X* counts as *Y* in context *C*" where the context *C* refers to the normative system in which the rule is specified. Constitutive rules relate "brute" physical facts, actions, etc. with institutional facts, actions, etc. For example, in the US, receiving a piece of paper with a certain shape, color, etc. (a physical action) counts as receiving a five-dollar bill (an institutional action). [1]

While our logic PDL-PA allows to model and reason about some regulative aspects of normative systems (permissions and authorizations), it does not account for constitutive rules which are an essential component to the modelling of social reality. In this section, we show how PDL-PA can be extended in order to describe them.

We postpone to future work an analysis of more subtle aspects of "counts-as" like for instance the relationship between "counts-as" and the notion of institutional power, and the distinction between the "counts-as" relation and a notion of dependence between events in the causal sense as in [Thielscher, 1997]. Note that the distinction between "counts-as" and causality is somehow explicit in Goldman's theory of action [Goldman, 1970] where "causal generation" is opposed to "conventional generation". According to Goldman, physical actions are causally generated, that is, they just consist in an agent bringing about (i.e. causing) a certain state of affairs to be true. On the other hand, institutional actions are conventionally generated, by which he meant that actions such as signalling before making a turn, and checkmating one's opponent, exist in virtue of rules or conventions relating physical actions with institutional effects.

We add to the language a primitive $\alpha_1 \rightsquigarrow \alpha_2$ that reads that the occurrence of α_1 *counts as* the occurrence of α_2, where α_1 and α_2 are assignments. For example, suppose we are modelling driving laws. Then, the formula $+80mph \rightsquigarrow +overspeed$ means that if a car runs at 80mph this counts as an overspeeding that has to be sanctioned. (We abstract away from the institutional context in which the constitutive rule applies.) Technically, our modelling is inspired from Thielscher's solution to the ramification problem in reasoning about actions [Thielscher, 1997].

Definition 2 *A model with constitutive rules is a tuple* (V, C) *such that V is a valuation and* $C \subseteq \mathcal{A} \times \mathcal{A}$ *is a* reflexive *and* transitive *relation over assignments that satisfies the following coherence constraint:*

$$(Coh) \quad (\alpha, +q) \notin C \text{ or } (\alpha, -q) \notin C$$

When $(\alpha_1, \alpha_2) \in C$ then the occurrence α_1 counts as the occurrence of α_2. The constraint (*Coh*) says that an event cannot have inconsistent ramifications. Note that it follows from (*Coh*) together with reflexivity of *C* that $(+q, -q) \notin C$, $(+\mathrm{P}_i(\alpha), -\mathrm{P}_i(\alpha)) \notin C$, etc.

Remark 1 *It is worth noting that there is some disagreement whether the "counts-as" relation should satisfy transitivity. For a discussion on this matter see [Jones and Sergot, 1996; Grossi* et al.*, 2006; Lorini* et al.*, 2009]. In addition, some authors argue that the "counts-as" should satisfy contraposition [Grossi* et al.*, 2006], while others have a different opinion on this matter [Jones and Sergot, 1996]. We did not include contraposition in the list of properties of Definition 2; however, the constraint "if* $(+p, +q) \in C$ *then* $(-q, -p) \in C$*" could be added to C.*

The truth condition for the "counts-as" construction is:

$$(V, C) \models \alpha_1 \rightsquigarrow \alpha_2 \quad \text{iff} \quad (\alpha_1, \alpha_2) \in C$$

We have to modify the definition of the relation R_π in order to take the relation *C* into account; we only change the case where π is an atomic event α:

$$R_\alpha \quad = \quad \{(V, (V \setminus \{q : (\alpha, -q) \in C\}) \cup \{q : (\alpha, +q) \in C\})\}$$

[1] We only consider here the "counts-as" relation between events. We note that Searle's "counts-as" relation is also between objects (such as pieces of paper and money).

Note that we are now able to define R_α without a case analysis as we did previously. Note also that C is never updated.

We note the new logic PDL-PA$_{\leadsto}$. Validity and satisfiability in PDL-PA$_{\leadsto}$ are defined in the standard way. For example, the following schemas are valid:

$$\begin{array}{l} \alpha \leadsto \alpha \\ (\alpha_1 \leadsto \alpha_2) \wedge (\alpha_2 \leadsto \alpha_3) \rightarrow (\alpha_1 \leadsto \alpha_3) \\ \neg((\alpha \leadsto +q) \wedge (\alpha \leadsto -q)) \\ \langle\alpha\rangle q \leftrightarrow ((\alpha \leadsto +q) \vee (q \wedge \neg(\alpha \leadsto -q))) \end{array}$$

The last equivalence is reminiscent of Reiter's successor state axioms in the Situation Calculus [Reiter, 2001]. It follows from it that $(\alpha \leadsto +q) \rightarrow \langle\alpha\rangle q$. For example, if driving at 80mph counts as overspeeding (i.e. $+80mph \leadsto +overspeed$) then, after starting to drive at 80mph, the driver will be over the speed limit (i.e. $\langle +80mph\rangle overspeed$).

The proofs of the complexity results for PDL-PA can be adapted straightforwardly to PDL-PA$_{\leadsto}$.

Theorem 5 *The problems of model checking and of satisfiability checking in PDL-PA$_{\leadsto}$ are both PSPACE-complete.*

7 Conclusion

We have presented the propositional dynamic logic of assignments PDL-PA, with assignments of the form $+p$ and $-p$ that are combined by the standard program constructions of PDL. The logic allows to reason about practical possibilities (*alias* abilities) and deontic possibilities (*alias* permissions) of agents and coalitions of agents, as well as the dynamics of abilities and permissions. The abilities and permissions may be higher-order. We have demonstrated the logic on an example about water resource management. Finally, we have introduced an extension of PDL-PA that allows to reason about constitutive rules.

We have shown that the model checking problem and the satisfiability problem are PSPACE-complete, both for PDL-PA and its extension with the "counts-as" operator. This contrasts with other formalisms that were proposed in the literature. In most of them complexity results are not mentioned. The only complexity result we are aware of are by Ågotnes *et al.* In [Ågotnes *et al.*, 2007] a normative extension of CTL is presented. Its symbolic model checking problem is proved to be PSPACE complete in the interpreted case, and EXPTIME hard in the uninterpreted case. In the former, agents' abilities and permissions are completely specified, while in the latter only the abilities are completely specified. In that respect, symbolic model checking in our logic provides more flexibility. We note however that their framework has a richer temporal ontology than ours. We leave further investigation of differences and similarities to future work.

A prominent benefit of our dynamic logic evaluated over simple symbolic models is that their extension with further notions relevant to social reality appears like a simpler task than in any other formalism of the literature. We have demonstrated that with the integration of constitutive rules. An interesting direction of future research that we did not address in Section 6 is the dynamics of constitutive rules in an institution. This could be done in a way that is in the very spirit of this paper by integrating two new kinds of events: $\pi_1 + \pi_2$ adds π_2 as a ramification of π_1, and $\pi_1 - \pi_2$ withdraws π_2 as a ramification of π_1. The only difficulty here is to design a model update preserving the constraints on C. This is subject of ongoing work.

References

[Ågotnes *et al.*, 2007] T. Ågotnes, W. van van der Hoek, J.A. Rodriguez-Aguilar, C. Sierra, and M. Wooldridge. On the logic of normative systems. In *IJCAI'07*, pages 1181–1186. AAAI Press, 2007.

[Alchourrón and Bulygin, 1971] C. Alchourrón and E. Bulygin. *Normative systems*. Springer, New York, 1971.

[Boella and Torre, 2004] G. Boella and L. van der Torre. Regulative and constitutive norms in normative multiagent systems. In *KR 2004*, pages 255–266. AAAI Press, 2004.

[Goldman, 1970] A. Goldman. *A Theory of Human Action*. Prentice-Hall, Englewood Cliffs NJ, 1970.

[Grossi *et al.*, 2006] D. Grossi, J.-J. Ch. Meyer, and F. Dignum. Classificatory aspects of counts-as: An analysis in modal logic. *Journal of Logic and Computation*, 16(5):613–643, 2006.

[Harel *et al.*, 2000] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, 2000.

[Hoek and Wooldridge, 2005] W. van der Hoek and M. Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence*, 164(1-2):81–119, 2005.

[Hoek *et al.*, 2010] W. van der Hoek, D. Walther, and M. Wooldridge. Reasoning about the transfer of control. *JAIR*, 37:437–477, 2010.

[Jones and Sergot, 1996] A. Jones and M. J. Sergot. A formal characterization institutionalised power. *J. of the IGPL*, 4:429–445, 1996.

[Lorini *et al.*, 2009] E. Lorini, D. Longin, B. Gaudou, and A. Herzig. The logic of acceptance: grounding institutions on agents' attitudes. *J. of Logic and Computation*, 19(6):901–940, 2009.

[Reiter, 2001] R. Reiter. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT Press, Cambridge, 2001.

[Searle, 1969] J. R. Searle. *Speech acts: An essay in the philosophy of language*. Cambridge University Press, New York, 1969.

[Searle, 1995] J. R. Searle. *The Construction of Social Reality*. The Free Press, New York, 1995.

[Shoham and Tennenholtz, 1996] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. In P. E. Agre and S. J. Rosenschein, editors, *Computational theories of interaction and agency*, pages 597–618. MIT Press, Cambridge, M., 1996.

[Thielscher, 1997] M. Thielscher. Ramification and causality. *Artificial Intelligence*, 89:317–364, 1997.