A Computationally Grounded Dynamic Logic of Agency, with an Application to Legal Actions

Andreas Herzig[1], Tiago de Lima[2], Emiliano Lorini[1], Nicolas Troquard[3]

[1] University of Toulouse and CNRS, IRIT, Toulouse, France
[2] University of Artois and CNRS, Lens, France
[3] LOA-ISTC-CNR, Trento, Italy

Abstract. In this article, we propose a Dynamic Logic of Propositional Control DL-PC in which the concept of 'seeing to it that' (abbreviated stit) as studied by Belnap, Horty and others can be expressed; more precisely, we capture the concept of the so-called Chellas stit theory and the deliberatibe stit theory, as opposed to Belnap's original achievement stit. In this logic, the sentence 'group G sees to it that φ' is defined in terms of dynamic operators: it is paraphrased as 'group G is going to execute an action now such that whatever actions the agents outside G can execute at the same time, φ is true afterwards'. We also prove that the satisfiability problem is decidable. In the second part of the article we extend DL-PC with operators modeling normative concepts, resulting in a logic DL-PC$^{\mathrm{Leg}}$. In particular, we define the concepts of 'legally seeing to it that' and 'illegally seeing to it that'. We prove that the decidability result for DL-PC transfers to DL-PC$^{\mathrm{Leg}}$.

Keywords: Modal logic, dynamic logic, action, agency, propositional control

1 Introduction

The study of formalisms enabling reasoning about individual and group abilities is a very active research field in AI and in multiagent systems. For instance, in the last decade, Alternating-time Temporal Logic (ATL) [5] has been prominent in the logical analysis of multiagent systems [28, 20, 1]. Albeit complex, ATL satisfiability is decidable, and there exist model representation languages, algorithms, and tools for model checking ATL formulas [4, 3, 21, 27]. These tools allow to verify properties of multiagent systems such as whether a set of services can cooperate in a way to achieve some desired goal.

However, logics of strategic ability are not appropriate when we want to study multiagent systems involving *actual agency*. Actual agency is ubiquitous in the description of agents' behaviors: the strategising of competing or cooperating agents is based on what they observe (know, believe) about what the other agents are actually doing. For instance, a Nash equilibrium is realized when every agent plays the best response to what is the actual action of the other players. This at least partly explains why logics of 'seeing to it that' —abbreviated stit— are most prominent in philosophy of action [19, 10]. Its basic construction are formulas of the form $\mathrm{Stit}_i\varphi$ that read "agent i sees to it

that φ". The focus is therefore on the result of the action, i.e., φ (and not on the action itself).

A drawback that is common to ATL and stit logics is that their models are not what has been called 'computationally grounded'. That is, the semantics of computations is given in terms of traditional possible worlds semantics, at a level of description that is not related to what would be an actual computer implementation of it [34]. In particular, model checking a stit formula is easy, but representing the models is unfeasible in practice for any non-trivial multiagent system. This contrasts with, e.g., *logic of propositional control* CL-PC, within which the coalition logic fragment of ATL can be reconstructed [18, 17, 16, 13]. The models of these logics were inspired by the model representation languages for ATL [4]. They not only have a function associating to each propositional variable a truth value, but also a function associating to each propositional variable the agent controlling it, where 'control' means that the agent is able to set the value of the variable to true or false. These logics may legitimately be called computationally grounded because they are equipped with models that can be readily captured by a concise syntactic description. This makes it interesting to represent multiagent systems in terms of models of propositional control.

Our aim in this paper is to provide a logic that both (i) allows to reason about actual agency and (ii) has models that are computationally grounded.

We use dynamic logics as our starting point. The latter were invented and used in computer science and artificial intelligence in order to reason about actions performed by computer programs and artificial agents. Their basic construction are formulas of the form $\langle\pi\rangle\varphi$, read "there is a possible execution of π such that φ is true afterwards". The focus is therefore both on the means (viz. the program π) and the result (viz. the proposition φ). Beyond means-end reasoning, one of the advantages of dynamic logics is that they can have a rich ontology of program operators.[1] These operators allow to express standard program constructions such as `if`...`then`...`else`... or `while`. However, dynamic logics also have some shortcomings as a logic of agency. First of all, agents are not part of its ontology. Second, in its standard reading the modal operator $\langle\pi\rangle$ is not about the performance of an action, but rather about the *opportunity* to perform an action, cf. the above reading of $\langle\pi\rangle\varphi$. Third and just as ATL and stit logics, the standard dynamic logics with Kripke semantics (such as PDL) are not computationally grounded.

The first thing we do is to add agents to the picture. It is straightforward to replace the atomic programs in the grammar of actions by atomic actions of the form (i, e), where i is the agent performing action e. This alone would be mere cosmetics: inspired by logics of propositional control, we moreover introduce in the semantics the notion of an agent's action repertoire of atomic programs: i can execute e only if e is in i's repertoire.

The second thing we do is to complement the 'can' modality of dynamic logic by a 'do' operator. The latter operator was used before e.g. in [11, 12]. We then can write

[1] Sequence of events, non-determinism, interleaving, unbounded finite iterations, infinite iterations, intersection, negation, converse, etc.

both $\langle\alpha\rangle\varphi$, reading "there is a possible execution of α such that φ true afterwards", and $\langle\!\langle\alpha\rangle\!\rangle$, reading "$\alpha$ is going to be executed".

The third thing we do is to use a computationally grounded version of dynamic logic whose atomic programs are assignments of propositional variables to either true or false: $(i, +p)$ is i's action of setting the truth value of p to true, and $(i, -p)$ is i's action of setting the truth value of p to false. Hence, the name of an action is not a simple program name variable like in PDL. The semantics of an assignment action $(i, -p)$ is hard-coded into its syntactic representation. Therefore, it relieves the system designer from the tedious (and otherwise impractical) task of specifying manually the transition system. This is of course what would be sought after in the implementation of a sensible modelling tool for dynamic logics. In this paper we study our logic directly over grounded models.

In the dynamic logic literature there are only very few papers about assignments, that are viewed there as particular programs [25, 33, 32]. More recently assignments were studied in dynamic epistemic logics [26, 31, 29]. We here combine them with agents into actions, just as in logics of propositional control. A notion of actual agency of a coalition of agents G is captured by what is true of the world whatever the valuation of the variables under the control of the agents outside G is.

Altogether, we provide (i) a logic of actual agency based on models that are (ii) computationally grounded, but (iii) still as rich as stit models, while (iv) enjoying decidability of satisfiability. The latter property contrasts with stit logics: reasoning about coalitional agency in the original branching-time stit models is undecidable [14]. (Some interesting variants are however decidable [6].)

The rest of the paper is organized as follows. In the next section we present the syntax, semantics and some mathematical properties of our dynamic logic of propositional control DL-PC; in particular we prove decidability of the satisfiability problem. In Section 3, we illustrate the logic on a case study of voting and legal voting. We are naturally led to propose an extension of our logic to model legal and illegal action. We conclude in Section 4.

2 Dynamic logic of propositional control DL-PC

We now introduce the dynamic logic of propositional control DL-PC by defining its syntax and semantics.

2.1 Syntax

The vocabulary of the DL-PC contains a set $\mathbb{P}$ of propositional variables and a finite non-empty set $\mathbb{A}$ of agent names.

Given a propositional variable $p \in \mathbb{P}$, $+p$ denotes the *positive assignment* of p, i.e., the event of setting the value of p to true, and $-p$ denotes the *negative assignment* of p, i.e., the event of setting the value of p to false. Given a set of propositional variables $P \subseteq \mathbb{P}$, the set of all positive assignments of the elements of P is

$$+P = \{+p : p \in P\}$$

and the set of all negative assignments is

$$-P = \{-p : p \in P\}$$

The set of assignments of variables from P (for short: the set of P-assignments) is

$$\pm P = +P \cup -P$$

The set of $\mathbb{P}$-assignments is therefore $\pm\mathbb{P} = +\mathbb{P} \cup -\mathbb{P}$. We use e for elements of $\pm\mathbb{P}$.

An *individual action* is a couple made up of an agent name and the assignment of a propositional variable. The set of all individual actions is $\mathsf{Act} = \mathbb{A} \times \pm\mathbb{P}$. A *group action* is a finite set of actions from Act. The set of all group actions is noted GAct. The set of *sequences of group actions* is noted GAct^*. The empty sequence is noted nil, and the typical elements of GAct^* are noted σ, σ_1, etc. Given a group action $\alpha \in \mathsf{GAct}$ and a group of agents $G \subseteq \mathbb{A}$, we define *G's part in α* as follows:

$$\alpha_G = \alpha \cap (G \times \pm\mathbb{P}) = \{(i, e) : (i, e) \in \alpha \text{ and } i \in G\}$$

In particular, $\alpha_\emptyset = \emptyset$ and $\alpha_\mathbb{A} = \alpha$. Clearly, every α_G is also a group action from GAct.

The *language* of DL-PC is the set of formulas φ defined by the following BNF:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle\alpha\rangle\!\rangle\varphi \mid \langle\alpha\rangle\varphi \mid \mathrm{Stit}_G\varphi \mid \mathrm{X}\varphi$$

where p ranges over $\mathbb{P}$, G ranges over $2^\mathbb{A}$, and α ranges over GAct. The modal operators $\langle\alpha\rangle$ and $\langle\!\langle\alpha\rangle\!\rangle$ are both dynamic operators. The former is about opportunity while the latter is about agency: $\langle\!\langle\alpha\rangle\!\rangle\varphi$ reads "α is going to be performed and φ will be true after updating by α", while $\langle\alpha_G\rangle\varphi$ reads "α_G can be performed and φ will be true after updating by α". The modal operator Stit stands for "seeing-to-it-that": the formula $\mathrm{Stit}_G\varphi$ reads "group G sees to it that φ is true". X is a temporal 'next' operator: the formula $\mathrm{X}\varphi$ is read "next φ".

We use the common abbreviations for $\vee$, $\rightarrow$, $\leftrightarrow$ and $\bot$. When α is a singleton $\{(i, e)\}$ we write $\langle\!\langle i, e\rangle\!\rangle\varphi$ instead of $\langle\!\langle\{(i, e)\}\rangle\!\rangle\varphi$. The set of propositional variables occurring in a formula φ is noted $\mathbb{P}_\varphi$ and the set of agents occurring in φ is noted $\mathbb{A}_\varphi$. For example, $\mathbb{P}_{\langle i,-p\rangle q} = \{p, q\}$ and $\mathbb{A}_{\langle i,-p\rangle q} = \{i\}$.

2.2 Models

While the semantics of PDL is in terms of Kripke models the semantics of DL-PC is not, and models are simply valuations of propositional logic that are augmented by two further ingredients: first, every agent has a *repertoire of assignments* that is available to him; second, there is a *successor function* which for every sequence of group actions tells us which group action is going to take place next. Such models consist therefore of tuples $\langle\mathcal{R}, \mathcal{S}, \mathcal{V}\rangle$, where:

- $\mathcal{R} \subseteq \mathbb{A} \times \pm\mathbb{P}$ (the action repertoire)
- $\mathcal{S} : \mathsf{GAct}^* \longrightarrow \mathsf{GAct}$ such that $\mathcal{S}(\sigma) \subseteq \mathcal{R}$ for every $\sigma \in \mathsf{GAct}^*$ (the successor function)
- $\mathcal{V} \subseteq \mathbb{P}$ (the valuation)

The valuation $\mathcal{V}$ provides the set of propositional variables from $\mathbb{P}$ that are true. The repertoire $\mathcal{R}$ is a set of group actions: when $(i, e) \in \mathcal{R}$ then agent i is able to perform e. $\mathcal{S}$ associates to every finite sequence of group actions $\sigma \in \mathsf{GAct}$ the group action $\mathcal{S}(\sigma) \in \mathsf{GAct}$ that will occur after σ. So $\mathcal{S}(\mathrm{nil})$ is the group action that is going to be performed now. Our constraint that $\mathcal{S}(\sigma) \subseteq \mathcal{R}$ makes that every $\mathcal{S}(\sigma)$ respects $\mathcal{R}$: for example, when $(i, e) \in \mathcal{S}(\mathrm{nil})$ then according to $\mathcal{S}$ agent i performs e next; we then require e to be in i's repertoire, i.e., $(i, e) \in \mathcal{R}$. Note that the group action $\emptyset$ is consistent with every repertoire. The set $(\mathcal{S}(\mathrm{nil}))_G$ is group G's part of the next action, i.e., it is the group action that G will execute now.

2.3 Updating valuations

Just as in dynamic epistemic logic with assignments [30], the dynamic operators are interpreted as *model updates*.

The language of DL-PC allows for group actions with conflicting assignments, like $\alpha = \{(i, +p), (j, -p)\}$, where two agents disagree on the new value of the variable p; actually these two agents might even be identical. We could stipulate that such a group action cannot be performed. We take a different route: the value of a variable p changes only if the agents trying to assign p agree on the new value. The other way round, if the agents disagree on the new value of a variable then this variable keeps its old truth value.

The update of the model $\mathcal{M} = \langle \mathcal{R}, \mathcal{S}, \mathcal{V} \rangle$ by the group action $\alpha \in \mathsf{GAct}$ is the new model $\mathcal{M}^\alpha = \langle \mathcal{R}^\alpha, \mathcal{S}^\alpha, \mathcal{V}^\alpha \rangle$, where:

$$
\begin{aligned}
\mathcal{R}^\alpha &= \mathcal{R} \\
\mathcal{S}^\alpha(\sigma) &= \mathcal{S}(\alpha \cdot \sigma) \quad \text{(where the symbol `$\cdot$' stands for concatenation of lists)} \\
\mathcal{V}^\alpha &= (\mathcal{V} \setminus \{p : \text{ there is } (i, -p) \in \alpha \text{ and there is no } (j, +p) \in \alpha\}) \cup \\
&\qquad \{p : \text{ there is } (i, +p) \in \alpha \text{ and there is no } (j, -p) \in \alpha\}
\end{aligned}
$$

Hence $\mathcal{S}^\alpha(\mathrm{nil})$ (the group action that will be executed now in $\mathcal{M}^\alpha$) is the group action that will be executed after α in $\mathcal{M}$; and $\mathcal{V}^\alpha$ (the set of variables that are true in $\mathcal{M}^\alpha$) is $\mathcal{V}$ without those variables that have been set to false by α, plus the new variables that have been set to true by α.

Clearly, the update $\mathcal{M}^\alpha$ of a DL-PC model $\mathcal{M}$ is also a DL-PC model; in particular, the successor function $\mathcal{S}^\alpha$ respects $\mathcal{R}$.

2.4 Varying the successor function

The stit operator will be evaluated by varying the successor function.

Given two successor functions $\mathcal{S}$ and $\mathcal{S}'$, we say that *$\mathcal{S}$ and $\mathcal{S}'$ agree on G's strategy*, noted $\mathcal{S} \sim_G \mathcal{S}'$, if and only if $(\mathcal{S}'(\sigma))_G = (\mathcal{S}(\sigma))_G$ for every sequence of group actions σ. We also say that $\mathcal{S}'$ is a G-variant of $\mathcal{S}$.

This extends to models: two models $\mathcal{M} = \langle \mathcal{R}, \mathcal{S}, \mathcal{V} \rangle$ and $\mathcal{M}' = \langle \mathcal{R}', \mathcal{S}', \mathcal{V}' \rangle$ *agree on G's strategy*, noted $\mathcal{M} \sim_G \mathcal{M}'$, if and only if $\mathcal{R} = \mathcal{R}'$, $\mathcal{V} = \mathcal{V}'$, and $\mathcal{S} \sim_G \mathcal{S}'$. Clearly, when $\mathcal{M}$ is a DL-PC model and $\mathcal{M} \sim_G \mathcal{M}'$ then $\mathcal{M}'$ is also a DL-PC model (simply because the agreement relation only relates DL-PC models); in particular, its successor function $\mathcal{S}'$ respects $\mathcal{R}$.

2.5 Truth conditions

Let $\mathcal{M} = \langle \mathcal{R}, \mathcal{S}, \mathcal{V} \rangle$ be a DL-PC model. The satisfaction relation $\models$ between DL-PC models and formulas is defined as usual for the Boolean operators, plus:

$$\begin{array}{lll}
\mathcal{M} \models p & \text{iff} & p \in \mathcal{V} \\
\mathcal{M} \models \langle\!\langle \alpha \rangle\!\rangle \varphi & \text{iff} & \alpha \subseteq \mathcal{S}(\mathrm{nil}) \text{ and } \mathcal{M}^\alpha \models \varphi \\
\mathcal{M} \models \langle \alpha \rangle \varphi & \text{iff} & \alpha \subseteq \mathcal{R} \text{ and } \mathcal{M}^\alpha \models \varphi \\
\mathcal{M} \models \mathrm{Stit}_G \varphi & \text{iff} & \mathcal{M}' \models \varphi \text{ for every } \mathcal{M}' \text{ such that } \mathcal{M}' \sim_G \mathcal{M} \\
\mathcal{M} \models \mathrm{X} \varphi & \text{iff} & \mathcal{M}^{\mathcal{S}(\mathrm{nil})} \models \varphi
\end{array}$$

In words, in model $\mathcal{M}$, group G sees to it that φ if and only if φ is true in every DL-PC model that agrees with G's strategy in $\mathcal{M}$. In other words, G sees to it that φ if and only if φ obtains due to the actions selected by G, whatever the other agents choose to do. This is a stit operator *à la Chellas* [8].

Let us consider the two cases when G is empty and when it is the set of all agents $\mathbb{A}$. First, $\mathrm{Stit}_\emptyset \varphi$ means "φ is true whatever the agents choose to do". This is a modal operator of historic necessity (quantifying over all possible combinations of agents' choices) just as in stit logics. Second, $\mathrm{Stit}_\mathbb{A} \varphi$ is equivalent to φ because the only model $\mathcal{M}'$ such that $\mathcal{M} \sim_\mathbb{A} \mathcal{M}'$ is $\mathcal{M}$ itself.

2.6 Validity

As usual, a formula φ is valid in DL-PC (notation: $\models \varphi$) if and only if every DL-PC model satisfies φ. A formula φ is satisfiable in DL-PC if and only if $\not\models \neg\varphi$.

In the sequel we discuss some DL-PC validities. Both $\langle\!\langle \alpha \rangle\!\rangle$ and $\langle \alpha \rangle$ are normal modal diamond operators; in particular the schemas

$$\langle\!\langle \alpha \rangle\!\rangle (\varphi \wedge \psi) \to (\langle\!\langle \alpha \rangle\!\rangle \varphi \wedge \langle\!\langle \alpha \rangle\!\rangle \psi)$$
$$\langle \alpha \rangle (\varphi \wedge \psi) \to (\langle \alpha \rangle \varphi \wedge \langle \alpha \rangle \psi)$$

are valid. The modal operators Stit_G are normal modal box operators; in particular, the schemas $\mathrm{Stit}_G \top$ and $\mathrm{Stit}_G(\varphi \wedge \psi) \leftrightarrow (\mathrm{Stit}_G \varphi \wedge \mathrm{Stit}_G \psi)$ are valid.

The formulas $\langle \emptyset \rangle \top$ and $\langle\!\langle \emptyset \rangle\!\rangle \top$ are both DL-PC valid. The formula schema

$$\langle\!\langle \alpha_G \rangle\!\rangle \top \to \langle \alpha_G \rangle \top$$

is valid (because $\mathcal{S}(\mathrm{nil}) \subseteq \mathcal{R}$). This is a 'do implies can' principle: if α is going to be performed then α can be performed.

If φ is a Boolean formula then $\langle \{(i,+p),(j,-q)\} \rangle \varphi \to \varphi$ is valid. The schema is however not valid in general; to see this take e.g. $\langle\!\langle (i,+q) \rangle\!\rangle \top$ for φ. Its converse $\varphi \to \langle \{(i,+p),(j,-q)\} \rangle \varphi$ is invalid even when φ is restricted to Boolean formulas: the group action $\{(i,+p),(j,-q)\}$ is inexecutable as soon as $+p$ is not in i's repertoire or $+q$ is not in j's repertoire.

An interesting DL-PC validity is $\mathrm{Stit}_i(p \vee q) \to (\mathrm{Stit}_i p \vee \mathrm{Stit}_i q)$. Indeed, when $\mathcal{M} \models \mathrm{Stit}_i(p \vee q)$ then either p must be in i's repertoire, or q, or both. This validity

distinguishes our logic from the logic of the Chellas stit, where the above principle is invalid. We refer the reader to [13] for further discussions on this issue.

We also observe that $\langle\!\langle\alpha\rangle\!\rangle\varphi \rightarrow \mathrm{X}\varphi$ is invalid. (To see this, note that $\varphi \rightarrow \langle\!\langle\emptyset\rangle\!\rangle\varphi$ is valid and that φ should not imply $\mathrm{X}\varphi$.)

Finally, the following valid schema expresses the independence of agents:

$$\mathrm{Stit}_G\mathrm{Stit}_H\varphi \rightarrow \mathrm{Stit}_\emptyset\varphi, \quad \text{for } G \cap H = \emptyset$$

It was shown in [15] that the above schema is the group counterpart to Xu's axiom schema of independence of agents AIA which is central in stit logics (that was formulated for the language where the argument of the stit operator is restricted to singletons) [8, 7].

2.7 Decidability

We now prove that satisfiability is decidable.

Here are some definitions that we need for our results. The *length* of a formula is the number of symbols we need to write it down, including parentheses, '(', '+', etc. We denote the length of a formula φ by $|\varphi|$. For example, $|\langle i, -p\rangle q| = 2 + 4 + 1 = 7$. Moreover, we define the size $|\sigma|$ of a sequence of group actions σ as follows:

$$\begin{aligned} |\mathrm{nil}| &= 0 \\ |\alpha \cdot \sigma| &= \mathrm{card}(\alpha) + |\sigma| \end{aligned}$$

where $\mathrm{card}(\alpha)$ is the cardinality of the set α.

The *dynamic depth* of a formula is the maximal number of nested dynamic operators and 'next' operators, defined inductively as:

$$\begin{aligned} \delta(\top) = \delta(p) &= 0 \\ \delta(\neg\varphi) = \delta(\mathrm{Stit}_G\varphi) &= \delta(\varphi) \\ \delta(\varphi \wedge \psi) &= \max(\delta(\varphi), \delta(\psi)) \\ \delta(\langle\alpha\rangle\varphi) = \delta(\langle\!\langle\alpha\rangle\!\rangle\varphi) = \delta(\mathrm{X}\varphi) &= 1 + \delta(\varphi) \end{aligned}$$

We are now going to define the *size* of a finite DL-PC model. Clearly, a finite DL-PC model should have a finite repertoire and a finite valuation; but how can a successor function be finite? *A priori*, the representation of the function $\mathcal{S}$ is an infinite set of couples $\langle\sigma, \mathcal{S}(\sigma)\rangle$, one per sequence $\sigma \in \mathsf{GAct}^*$. A way out is to consider that a model is finite when $\mathcal{R}$ and $\mathcal{V}$ are finite and the value of the successor function $\mathcal{S}$ is $\emptyset$ almost everywhere. Such functions can be represented in a finite way if we drop those couples $\langle\sigma, \mathcal{S}(\sigma)\rangle$ where $\mathcal{S}(\sigma) = \emptyset$ and view $\mathcal{S}$ as a partial function. Then the size of the finite DL-PC model $\mathcal{M} = \langle\mathcal{R}, \mathcal{S}, \mathcal{V}\rangle$ can be defined as the sum of the cardinalities of each of its elements, i.e.

$$\mathrm{size}(\mathcal{M}) = \mathrm{card}(\mathcal{R}) + \Sigma_{\{\sigma:\mathcal{S} \text{ is defined on } \sigma\}}|\sigma \cdot \mathcal{S}(\sigma)| + \mathrm{card}(\mathcal{V})$$

Proposition 1 (Strong fmp). *For every* DL-PC *formula* φ*, if* φ *is* DL-PC *satisfiable then* φ *is satisfiable in a model of size* $\mathcal{O}((|\varphi|)^{2|\varphi|})$.

The above proposition can be proved by (1) restricting the depth of the successor function to the dynamic depth $\delta(\varphi)$ of the formula φ, setting each $\mathcal{S}(\sigma)$ such that the length of the sequence σ is greater than $\delta(\neg\varphi)$ to $\emptyset$, and (2) by restricting repertoire, successor function and valuation to the vocabulary of φ.

Proposition 2 (Decidability of satisfiability). *The* DL-PC *satisfiability problem is decidable.*

Proof. This follows by [9, Theorem 6.7] from the above strong fmp (Proposition 1) and the fact that the set of DL-PC models of a given size is recursive (plus the fact that model checking is decidable).

One can prove that the satisfiability problem is PSPACE hard by encoding the QBF satisfiability problem. (This is actually already the case for the fragment of the DL-PC language without the next operator.) We conjecture that it is also PSPACE complete, but leave a formal proof to future work.

3 Case study: voting and legal voting

In this section, we present an example to illustrate the kind of scenario that can be formalized in DL-PC. Our logic allows to distinguish three different concepts: (1) the concept of action repertoire (i.e., which propositional atoms are controlled by a given agent or coalition of agents), (2) the concept of agency (i.e., what a given agent or coalition of agents brings about), and (3) the concept of capability (i.e., what a given agent or coalition of agents can bring about). Moreover, in DL-PC the concepts of agency and capability are grounded on the notion of action repertoire, in the sense that what a given agent (or coalition) does and can do is determined by its action repertoire. This contrasts with existing logics of actions and capabilities such as ATL and Coalition Logic CL, stit theories, Coalition Logic of Propositional Control CL-PC) [18, 17, 16, 13]. For instance, CL and ATL only consider the concept of capability, while stit theories consider both agency and capability but do not allow to model action repertoires. Finally, in CL − PC one can represent both action repertoires and capabilities, but it is not clear how the concept of agency can be expressed in this logic. The aim of the example is to instantiate the three different concepts that are expressible in DL-PC in a concrete scenario.

We start with a very simple example of voting. The set of agents is $\mathbb{A} = \{1, 2, 3, 4, 5\}$. Each agent is a voter who has to choose between two candidates for an election, candidate A and candidate B.

We assume that the set of propositional variables $\mathbb{P}$ is made up of variables $p_{i,B}$ and $p_{i,A}$, one per agent $i \in \mathbb{A}$:

$$\mathbb{P} = \{p_{i,A} : i \in \mathbb{A}\} \cup \{p_{i,B} : i \in \mathbb{A}\}$$

The variables $p_{i,B}$ and $p_{i,A}$ describe the two voting options of agent i: $p_{i,B}$ means 'agent i has voted for candidate A' and $p_{i,A}$ means 'agent i has voted for candidate B'.

We assume that the candidate obtaining the majority of votes is going to be elected. This latter statement can be formally expressed by the following two abbreviations:

$$\mathrm{elected}(A) \stackrel{\mathrm{def}}{=} \bigvee_{G\subseteq\mathbb{A},\mathrm{card}(G)>\mathrm{card}(\mathbb{A}\setminus G)} \left(\bigwedge_{i\in G} p_{i,A}\right)$$

$$\mathrm{elected}(B) \stackrel{\mathrm{def}}{=} \bigvee_{G\subseteq\mathbb{A},\mathrm{card}(G)>\mathrm{card}(\mathbb{A}\setminus G)} \left(\bigwedge_{i\in G} p_{i,B}\right)$$

So $\mathrm{elected}(A)$ and $\mathrm{elected}(B)$ have to be read respectively 'candidate A is elected' and 'candidate B is elected'. Moreover, we assume that every agent has his voting options in his action repertoire, namely the action of voting for candidate A and the action of voting for candidate B, that is,

$$\mathcal{R} = \{+p_{i,A} \in \mathbb{P} : i \in \mathbb{A}\} \cup \{+p_{i,B} \in \mathbb{P} : i \in \mathbb{A}\}$$

Finally, we assume that in the initial state of the system all variables $p_{i,A}$ and $p_{i,B}$ are set to false as nobody has voted, i.e., $\mathcal{V} = \emptyset$.

First of all, the concept of capability to achieve φ can be modelled in DL-PC by formulas of the form

$$\Diamond \mathrm{Stit}_G \varphi$$

where the formula $\Diamond\psi$ abbreviates $\neg\mathrm{Stit}_\emptyset\neg\psi$. The operator $\Diamond$ quantifies over all variants of the successor function $\mathcal{S}$: it is a modal operator of historic possibility. In our example we may e.g. express that "the group of agents $\{1,2,3\}$ has the capability of making candidate B elected" by means of the formula $\Diamond\mathrm{Stit}_{1,2,3}\mathrm{Xelected}(B)$. For any model $\mathcal{M} = \langle\mathcal{R},\mathcal{S},\mathcal{V}\rangle$

$$\mathcal{M}, w \models \Diamond\mathrm{Stit}_{1,2,3}\mathrm{Xelected}(B)$$

Let us now suppose that the agents 1, 2 and 3 decide to vote for candidate A whereas agents 4 and 5 decide to vote for B, that is,

$$\mathcal{S}(\mathrm{nil}) = \{+p_{1,A}, +p_{2,A}, +p_{3,A}, +p_{4,B}, +p_{5,B}\}$$

It follows that in the next state of the system candidate A is going to be elected:

$$\mathcal{M} \models \mathrm{Xelected}(A)$$

Furthermore, the group of agents $\{1,2,3\}$ sees to it that in the next state candidate A is elected:

$$\mathcal{M} \models \mathrm{Stit}_{\{1,2,3\}}\mathrm{Xelected}(A)$$

In fact, no matter what the agents 4 and 5 have chosen, the joint action of agents 1, 2, and 3 ensures that in the next state candidate A will be elected. On the contrary, for every subgroup G of $\{1,2,3\}$ it is not the case that G sees to it that in the next state A will be elected, that is:

$$\mathcal{M} \models \bigwedge_{G\subset\{1,2,3\}} \neg\mathrm{Stit}_G\mathrm{Xelected}(A)$$

For instance, the group $\{1, 2\}$ does not see to it that in the next state candidate A will be elected because the result of the election depends on agent 3's choice (i.e., if agent 3 had not voted for A, A would not have been elected).

Any common voting procedure raises questions about whether a ballot is valid or not. Probably the most (in)famous recent example is the *Florida election recount* in the 2000 US presidential elections. What cannot be denied is that in that year, the American voters saw to it that G.W. Bush was elected. The simple problem we concern ourselves here is to be able to recognise in logical terms when the actual actions of a group of agents yield that the group sees to something legally or not, independently of whether it actually sees to it.

Before proceeding to our case of legal and illegal voting, we propose next an extension of DL-PC which formalises the concepts of legal action and illegal action.

3.1 Legal and illegal action

We extend the language of DL-PC with three new constructions. The first two express legal and illegal action, respectively. The formula $\mathrm{LStit}_G\varphi$ reads "G legally sees to it that φ is true", and the formula $\mathrm{IStit}_G\varphi$ reads "G illegally sees to it that φ is true". The last construction is an atom of legal action: Leg_G reads "the group action currently chosen by G is legal". We write Leg_i instead of $\mathrm{Leg}_{\{i\}}$.

In order to interpret these operators we extend the definition of DL-PC models of Section 2 as follows. A *model* is now a quadruple $\langle \mathcal{R}, \mathcal{LR}, \mathcal{S}, \mathcal{V}\rangle$, where:

- $\langle \mathcal{R}, \mathcal{S}, \mathcal{V}\rangle$ is a DL-PC model;
- $\mathcal{LR} \subseteq \mathsf{Act} = \mathbb{A} \times \pm\mathbb{P}$ is a *repertoire of legal actions*.

The repertoire of legal actions $\mathcal{LR}$ represents the actions that the agents are authorised to perform: $(i, e) \in \mathcal{LR}$ means that agent i is authorised to perform e. So it may be the case that i has the ontic ('physical') ability to perform e but does not have the legal ability to perform e; and the other way round.

A group action $\alpha \in \mathsf{GAct}$ is authorised when $\alpha \subseteq \mathcal{LR}$.

The *update* of $\mathcal{M} = \langle \mathcal{R}, \mathcal{LR}, \mathcal{S}, \mathcal{V}\rangle$ by the group action $\alpha \in \mathsf{GAct}$ is the new model $\mathcal{M}^\alpha = \langle \mathcal{R}^\alpha, \mathcal{LR}^\alpha, \mathcal{S}^\alpha, \mathcal{V}^\alpha\rangle$, where $\mathcal{R}^\alpha$, $\mathcal{S}^\alpha$, and $\mathcal{V}^\alpha$ are defined as in Section 2.3, and $\mathcal{LR}^\alpha = \mathcal{LR}$. Moreover, when varying the successor function we require that not only the ontic repertoires but also the legal repertoires are kept constant.

The truth conditions for the new primitives are as follows:

$$
\begin{array}{lll}
\mathcal{M} \models \mathrm{Leg}_G & \text{iff} & (\mathcal{S}(\mathrm{nil}))_G \subseteq \mathcal{LR} \\
\mathcal{M} \models \mathrm{LStit}_G\varphi & \text{iff} & \mathcal{M} \models \mathrm{Stit}_G\varphi \text{ and } \mathcal{M} \models \mathrm{Leg}_G \\
\mathcal{M} \models \mathrm{IStit}_G\varphi & \text{iff} & \mathcal{M} \models \mathrm{Stit}_G\varphi \text{ and } \mathcal{M} \models \neg\mathrm{Leg}_G
\end{array}
$$

Hence, G legally sees to it that φ is true if and only if the action G will execute now is legal and, with that action, G sees to it that φ. Similarly, G illegally sees to it that φ is true if and only if, the action G will execute now is not legal and, with that action, G sees to it that φ.

Validity and satisfiability are then defined as usual. We call the resulting logic $\text{DL-PC}^{\text{Leg}}$.

The next proposition allows to eliminate the operators of legal action and illegal action.

Proposition 3 (Reduction of $\text{IStit}_G\varphi$ and $\text{LStit}_G\varphi$).

1. $\models \text{LStit}_G\varphi \leftrightarrow (\text{Stit}_G\varphi \wedge \text{Leg}_G)$
2. $\models \text{IStit}_G\varphi \leftrightarrow (\text{Stit}_G\varphi \wedge \neg\text{Leg}_G)$

Our decidability result for DL-PC can be adapted straightforwardly.

Proposition 4 (Strong fmp). *For every $\text{DL-PC}^{\text{Leg}}$ formula φ, if φ is $\text{DL-PC}^{\text{Leg}}$ satisfiable then φ is satisfiable in a $\text{DL-PC}^{\text{Leg}}$ model of size $\mathcal{O}((|\varphi|)^{2|\varphi|})$.*

Proposition 5 (Decidability of satisfiability). *The $\text{DL-PC}^{\text{Leg}}$ satisfiability problem is decidable.*

Let us examine some of the properties of the operators of legal action and illegal action.

First of all, differently from the normal modal box operators Stit_G, operators of legal and illegal action do not satisfy the axioms of necessity $\text{LStit}_G\top$ and $\text{IStit}_G\top$. This means that an agent or group of agents does not necessarily bring about tautologies in a legal or illegal way. These stit operators resemble the deliberative stit operator in this respect, which does not satisfy the axiom of necessity either. However, LStit and IStit differ from the deliberative stit operator in that they satisfy closure under conjunction: the schemas

$$\text{LStit}_G(\varphi \wedge \psi) \leftrightarrow (\text{LStit}_G\varphi \wedge \text{LStit}_G\psi)$$
$$\text{IStit}_G(\varphi \wedge \psi) \leftrightarrow (\text{IStit}_G\varphi \wedge \text{IStit}_G\psi)$$

are both $\text{DL-PC}^{\text{Leg}}$ valid.

Another important difference between the Chellas stit operators Stit_G and legal action operators IStit_G is that the former satisfy group monotony while the latter don't: if $G \subseteq H$ then the schema $\text{Stit}_G\varphi \rightarrow \text{Stit}_H\varphi$ is valid, whereas $\text{LStit}_G\varphi \rightarrow \text{LStit}_H\varphi$ is not. This means that when the group G sees to it that φ in a legal way, then this does not necessarily imply that every supergroup H of G sees to it that φ in a legal way. Indeed, there might be some agent in $H \setminus G$ whose chosen action is illegal. On the contrary, group monotony is valid for illegal action:

$$\models \text{IStit}_G\varphi \rightarrow \text{IStit}_H\varphi, \quad \text{for } G \subseteq H$$

Therefore, when a group G sees to it that φ in an illegal way then every supergroup H of G sees to it that φ in a illegal way.

3.2 Legal/illegal ballots

We are now able to take over our voting example and enrich it the normative consideration of Section 3.1. We distinguish not only feasible from unfeasible ballots, but also legal from illegal ballots.

Let us suppose that agent 1 does not meet the legal requirements for voting in the election even though he participates in the election. For example, we might imagine that agent 1 falsely claims to be a citizen of the state but actually does not have citizenship. This implies that the repertoire $\mathcal{LR}$ of legal actions does not coincide with the repertoire $\mathcal{R}$ of ontic actions: we have

$$\mathcal{LR} = \mathcal{R} \setminus \{+p_{1,A}, +p_{1,B}\}$$

As before, we assume that the agents 1, 2, and 3 vote for candidate A, while agents 4 and 5 vote for candidate B:

$$Succ(\text{nil}) = \{+p_{1,A}, +p_{2,A}, +p_{3,A}, +p_{4,B}, +p_{5,B}\}$$

Therefore group $\{1, 2, 3\}$ sees to it in an illegal way that in the next state A is elected:

$$\mathcal{M} \models \text{IStit}_{\{1,2,3\}}\text{Xelected}(A)$$

Indeed, 1's vote counts as an illegal vote and, consequently, the result of the election is not legal.

4 Conclusion

We have presented a dynamic logic of propositional control DL-PC which accounts for both individual and group agency, i.e., the fact that an agent or a group of agents sees to it that a given state of affairs φ is true. We have shown that satisfiability in DL-PC models is decidable. Our decidability result for DL-PC is interesting because we know that stit logic with individual and group agency is already undecidable [15]. This makes DL-PC an interesting alternative to stit logics.

We have presented an extension of DL-PC with the concepts of legal and illegal action and have proven its decidability. Our logic allows to account for the notion of normative system, defined as a set of prohibitions about the agents' behavior. As the set of illegal actions is the complement of the set of legal actions, a normative system can also be seen as a set of permissions about the agents' behavior. This definition of normative system therefore matches the definition accepted by several authors in the area of normative systems according to whom such a system is a set of constraints on the agents' behaviour, which may or may not be followed by the agents; see e.g. [23, 2, 24].

While we provided a decidability result for our logics, we did not axiomatise their validities. We leave this to future work.

As the reader may have noticed, our logic is not a dynamic logic in the strict sense because it lacks sequential and nondeterministic composition, iteration and test. This may be compared to the situation in dynamic epistemic logics, where the extension of public announcement logic by the Kleene star leads to undecidability [22]. We conjecture that this is not the case for DL-PC, but again leave this to future work.

Acknowledgements

We would like to thank the reviewers of DEON 2012 for their careful reading of the paper and their detailed suggestions.

References

1. T. Ågotnes. Action and knowledge in alternating-time temporal logic. *Synthese*, 149(2):377–409, 2006.
2. T. Ågotnes, W. van der Hoek, and M. Wooldridge. Robust normative systems and a logic of norm compliance. *Logic Journal of IGPL*, 18(1):4–30, 2009.
3. R. Alur, L. de Alfaro, Th.A. Henzinger, S.C. Krishnan, F.Y.C. Mang, S. Qadeer, S.K. Rajamani, and S. Taşiran. MOCHA user manual. University of Berkeley Report, 2000.
4. R. Alur and Th.A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.
5. R. Alur, Th.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. of the ACM*, 49(5):672–713, 2002.
6. Ph. Balbiani, O. Gasquet, A. Herzig, F. Schwarzentruber, and N. Troquard. Coalition games over Kripke semantics. In C. Dégremont, L. Keiff, and H. Rückert, editors, *Dialogues, Logics and Other Strange Things – Essays in Honour of Shahid Rahman*, pages 11–32. College Publications, http://www.collegepublications.co.uk/tributes/?00007, 2008.
7. Ph. Balbiani, A. Herzig, and N. Troquard. Alternative axiomatics and complexity of deliberative stit theories. *J. of Philosophical Logic*, 37(4):387–406, 2008.
8. N. Belnap, M. Perloff, and M. Xu. *Facing the Future: Agents and Choices in Our Indeterminist World*. Oxford University Press, 2001.
9. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. University Press, 2001.
10. J. Broersen. Modeling attempt and action failure in probabilistic stit logic. In Toby Walsh, editor, *IJCAI*, pages 792–797. IJCAI/AAAI, 2011.
11. Ph.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2–3):213–261, 1990.
12. A. Herzig and E. Lorini. A dynamic logic of agency I: STIT, abilities and powers. *Journal of Logic, Language and Information*, 19:89–121, 2010.
13. A. Herzig, E. Lorini, F. Moisan, and N. Troquard. A dynamic logic of normative systems. In Toby Walsh, editor, *International Joint Conference on Artificial Intelligence (IJCAI)*, Barcelona, 2011. Morgan Kaufmann Publishers.
14. A. Herzig and F. Schwarzentruber. Properties of logics of individual and group agency. In Carlos Areces and Robert Goldblatt, editors, *Advances in Modal Logic (AiML)*, pages 133–149, Nancy, 2008. College Publications.
15. A. Herzig and F. Schwarzentruber. Properties of logics of individual and group agency. In C. Areces and R. Goldblatt, editors, *AiML'08*, pages 133–149, Nancy, 2008. College Publications.
16. W. van der Hoek, D. Walther, and M. Wooldridge. On the logic of cooperation and the transfer of control. *JAIR*, 37:437–477, 2010.
17. W. van der Hoek and M. Wooldridge. On the dynamics of delegation, cooperation and control: a logical account. In *Proc. AAMAS'05*, 2005.
18. W. van der Hoek and M. Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence*, 164(1-2):81–119, 2005.
19. J. Horty. *Agency and deontic logic*. Oxford University Press, 2001.

20. W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2-3):185–219, 2004.
21. A. Lomuscio and F. Raimondi. MCMAS: a tool for verifying multi-agent systems. In *Proceedings of The Twelfth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS-2006)*. Springer Verlag, 2006.
22. J.S. Miller and L.S. Moss. The undecidability of iterated modal relativization. *Studia Logica*, 79(3):373–407, 2005.
23. M. Sergot and R. Craven. The deontic component of action language $nc+$. In *Deontic Logic and Artificial Normative Systems*, volume 4048 of *LNCS*, pages 222–237. Springer-Verlag, 2006.
24. 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.
25. M.L. Tiomkin and J.A. Makowsky. Propositional dynamic logic with local assignments. *Theor. Comput. Sci.*, 36:71–87, 1985.
26. J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.
27. W. van der Hoek, A. Lomuscio, and M. Wooldridge. On the complexity of practical ATL model checking. In *Proc. AAMAS 2006*, 2005.
28. W. van der Hoek and M. Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75:125–157, 2003.
29. H. van Ditmarsch, A. Herzig, and T. de Lima. From situation calculus to dynamic logic. *J. of Logic and Computation*, 21(2):179–204, 2011.
30. H. P. van Ditmarsch, W. van der Hoek, and B. P. Kooi. Dynamic epistemic logic with assignment. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, and M. Wooldridge, editors, *Proc. of AAMAS'05*, pages 141–148. ACM, 2005.
31. H.P. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Kluwer Academic Publishers, 2007.
32. J. van Eijck. Making things happen. *Studia Logica*, 66(1):41–58, 2000.
33. A. Wilm. Determinism and non-determinism in PDL. *Theor. Comput. Sci.*, 87(1):189–202, 1991.
34. M. Wooldridge. Computationally grounded theories of agency. In *4th International Conference on Multi-Agent Systems (ICMAS 2000)*, pages 13–22. IEEE Computer Society, 2000.