Three Traditions in the Logic of Action: Bringing them Together

Andreas Herzig, Tiago de Lima, Emiliano Lorini and Nicolas Troquard

Abstract We propose a Dynamic Logic of Propositional Control (DL-PC) that is equipped with two dynamic modal operators: one of ability and one of action. We integrate into DL-PC the concept of 'seeing to it that' (abbreviated by stit) as studied by Belnap, Horty and others. We prove decidability of DL-PC satisfiability and establish the relation with the logic of the Chellas stit opertor.

1 Introduction

Krister Segerberg's favourite logic of action is clearly dynamic logic [1–3]. However, there is another important tradition focusing on 'rival' modal logics, such as Pörn's logic of bringing-it-about [4–7] and Belnap et col.'s logic of seeing-to-it-that [8–10]. The latter logics should be called more precisely *logics of agency*: they allow to reason about whether an agent is agentive for a proposition.Beyond dynamic logic and logics of agency, other quite different logical approaches to action were developed in artificial intelligence (AI). There, the aim is to design practically usable formalisms that allow knowledge representation e.g. for automated planning. In Thomason's words, "to formalize realistic planning domains, to provide knowledge representation support for automated planning systems [...] requires an axiomatization of what Segerberg called the change function, which tells us what to expect when an action is performed" [11]. AI formalisms such as the situation calculus [12] focus on the

A. Herzig (✉) · E. Lorini
IRIT, University of Toulouse, 118 Route de Narbonne, 31062 Toulouse Cedex 9, France
e-mail: Andreas.Herzig@irit.fr

T. de Lima
University of Artois and CNRS, Rue Jean Souvraz SP 18,
62307 Lens Cedex, France

N. Troquard
LOA-ISTC, Trento, Italy

problem of defining such change functions, which became known under the denomination 'frame problem' and was considered to be one of the major challenges of AI. By far the most popular solution is in terms of Reiter's basic action theories which axiomatise the change function in terms of so-called successor state axioms [13, 14]. The definition of such axioms requires quantification over actions, which is a feature distinguishing these formalisms from dynamic logic and logics of agency that do not provide such a facility.

It is the aim of the present chapter to bring together the above three traditions in logics of action: dynamic logic, seeing-to-it-that (stit) logic, and situation calculus. We start from dynamic logic, into which we embed the situation calculus à la Reiter and integrate a stit operator of agency. More precisely, we are going to resort to a variant of dynamic logic that we call *dynamic logic of propositional control* (DL-PC).

As far as the embedding of situation calculus is concerned we build on the previous work of van Ditmarsch et al. [15]. There, basic action theories were mapped to a dynamic logic of propositional assignments. Let us call that logic DL-PA. It is a version of dynamic logic whose atomic programs are sets of assignments of propositional variables each of which is of the form $p \leftarrow \varphi$ where p is a propositional variable and φ is a formula. Such an assignment is always executable. DL-PA does not have quantification over actions, thus demonstrating that Reiter's solution to the frame problem actually does not require quantification over actions (contrarily to what Reiter had claimed). While agents play no particular role in DL-PA —that may actually be said to be rather about events than about actions—our logic DL-PC has 'true' actions: assignments performed by agents. An agent can only perform an assignment if he *controls* that assignment, in other words, if it is in his *repertoire*.[1]

Things are more involved if we want to embed logics of agency into our logic. The difficulties are threefold.

- Just as the above DL-PA, dynamic logic is about events rather than actions: agents do not play a role in dynamic logic. As we have said above, this can be overcome by associating repertoires of assignments to agents.
- In stit logic the agents act simultaneously, while (at least in the basic version of) dynamic logic actions are performed in sequence. We therefore need a version of dynamic logic with *parallel* actions. The above DL-PA actually already provides for sets of assignments; in DL-PC these are generalised to sets of authored assignments.
- The dynamic logic operator $\langle\alpha\rangle$ talks about the *possibility* of the occurrence of program α and not about the occurrence of α itself: instead of actual performance of an action, dynamic logic is rather about the opportunity to perform an action.

In order to overcome the third difficulty we are going to add to dynamic logic a second kind of dynamic operator, noted $\langle\langle\alpha\rangle\rangle$: while $\langle\alpha\rangle$ talks about the opportunity of performance of α, the new dynamic operator $\langle\langle\alpha\rangle\rangle$ is about the performance of

[1] Our syntax is actually a bit more restrictive: instead of $p \leftarrow \varphi$ it only allows for assignments to either true or false, written $+p$ and $-p$. The more general assignment $p \leftarrow \varphi$ can however be simulated by the dynamic logic program $(\varphi?; +p) \cup (\neg\varphi?; -p)$, where '?' is test, ';' is sequential composition, and '; $\cup$' is nondeterministic composition.

α. In the semantics we add a *successor function* modelling the next actions that are going to take place.

To sum it up: the language of our dynamic logic of propositional control DL-PC has a language in terms of two kinds of dynamic operators; the arguments of the dynamic operators are group actions; group actions are sets of assignments performed by agents. The semantics of DL-PC has a repertoire function and a successor function that both associate sets of assignments to agents. An obvious requirement is that if a group action α takes place according to the successor function then each of the individual actions in α must be executable, i.e. each individual assignment must be in the repertoire of the agent performing it.

The chapter is organised as follows. Section 2 we introduce dynamic logic of propositional control DL-PC and establish a decidability result. In Sect. 3 we study the fragment without stit operators and give a decision procedure in NP. In Sect. 4 we give reduction axioms for the fragment without the 'next' operator. In Sect. 5 we relate DL-PC to a discrete version of the Chellas stit logic.

2 Dynamic Logic of Propositional Control DL-PC

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

2.1 Syntax

The vocabulary of the Dynamic Logic of Propositional Control (DL-PC) contains a set P of propositional variables and a finite non-empty set Ag of agent names.

Given a propositional variable $p \in \mathsf{P}$, $+p$ 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 $P \subseteq \mathsf{P}$, the set of all positive assignments of 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 *all* assignments of variables in P is $\pm P = +P \cup -P$. The set of all assignments is therefore $\pm\mathsf{P} = +\mathsf{P} \cup -\mathsf{P}$. We use e for elements of $\pm\mathsf{P}$.

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 $\mathsf{Act} = \mathsf{Ag} \times \pm\mathsf{P}$. A *group action* is a finite set of actions from Act. The set of all group actions is $\mathsf{GAct} = 2^{\mathsf{Act}}$. 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. For a group action α and a group of agents $G \subseteq \mathsf{Ag}$ we define G's *part in* α as follows:

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

In particular, $\alpha_\emptyset = \emptyset$ and $\alpha_{\mathsf{Ag}} = \alpha$. 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 P, G ranges over 2^{Ag}, 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 $\perp$. When α is a singleton $\{(i, e)\}$ we write the more convenient $\langle\!\langle i, e\rangle\!\rangle\varphi$ instead of $\langle\!\langle\{(i, e)\}\rangle\!\rangle\varphi$. The set of propositional variables occurring in a formula φ is noted P_φ and the set of agents occurring in φ is noted Ag_φ. For example, $\mathsf{P}_{\langle i,-p\rangle q} = \{p, q\}$ and $\mathsf{Ag}_{\langle i,-p\rangle q} = \{i\}$.

2.2 Models

While the semantics of PDL is in terms of Kripke models the semantics of DL-PC is not (cf. [16]). Models for DL-PC 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:

- $\mathcal{R} \subseteq \mathsf{Ag} \times \pm\mathsf{P}$
- $\mathcal{S} : \mathsf{GAct}^* \longrightarrow \mathsf{GAct}$ such that $\mathcal{S}(\sigma) \subseteq \mathcal{R}$ for every $\sigma \in \mathsf{GAct}^*$
- $\mathcal{V} \subseteq \mathsf{P}$

The valuation $\mathcal{V}$ provides the set of propositional variables from 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 on $\mathcal{S}$ ensures 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 expect e to be in i's repertoire, i.e., we expect $(i, e) \in \mathcal{R}$. Note that the group action $\emptyset$ is consistent with every repertoire. According to our definitions $(\mathcal{S}(\mathrm{nil}))_G$ 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 [17], 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 might stipulate that such a group action cannot be performed. We take a different route: the new 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:

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

Hence $\mathcal{S}^\alpha(\text{nil})$ (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 *Succ* 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; 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:

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

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.

Let us consider the two cases when G is empty and when it is the set of all agents Ag. First, $\text{Stit}_\emptyset \varphi$ means "φ is true whatever the agents choose to do". This is a modal operator of historic necessity just as in stit logics. Second, $\text{Stit}_{\mathsf{Ag}} \varphi$ means "φ is true given the current strategies of all agents". This is a modal operator of historic possibility.

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 $\not\models \neg\varphi$. For example, the schema $\models \langle\langle \alpha_G \rangle\rangle \top \to \langle \alpha_G \rangle \top$ is valid (because $\mathcal{S}(\text{nil}) \subseteq \mathcal{R}$). This is a 'do implies can' principle: if α is going to be performed then α can be performed. Moreover, $\langle \emptyset \rangle \top$ and $\langle\langle \emptyset \rangle\rangle \top$ are both DL-PC valid. If φ is a Boolean formula then $\langle \{(i, +p), (j, -p)\} \rangle \varphi \to \varphi$ is valid. It is not valid in general; to see this take e.g. $\langle\langle (i, +q) \rangle\rangle \top$ for φ. Moreover, the converse is invalid, e.g. because $+p$ might not be in i's repertoire. Observe that 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. Observe also that the modal operators Stit_G are normal modal box operators; in particular, the schemas $\text{Stit}_G \top$ and $\text{Stit}_G(\varphi \wedge \psi) \leftrightarrow (\text{Stit}_G \varphi \wedge \text{Stit}_G \psi)$ are valid. A DL-PC validity that we are going to discuss later is $\text{Stit}_i (p \vee q) \to (\text{Stit}_i\, p \vee \text{Stit}_i\, q)$. Note that $\langle\langle \alpha \rangle\rangle \varphi \to \text{X}\varphi$ is invalid. (To see this, note that $\varphi \to \langle\langle \emptyset \rangle\rangle \varphi$ is valid and that φ should not imply $\text{X}\varphi$.)

2.6 Replacement of Equivalents

The rule of replacement of valid equivalents will be useful in Sects. 3 and 4. It is based on the following proposition.

Proposition 1 (Rules of equivalents for $\langle\alpha\rangle$, $\langle\!\langle\alpha\rangle\!\rangle$, and Stit_G)

1. *If* $\models \varphi_1 \leftrightarrow \varphi_2$ *then* $\models \langle\alpha\rangle\varphi_1 \leftrightarrow \langle\alpha\rangle\psi\varphi_2$ *(rule of equivalents for $\langle\alpha\rangle$)*
2. *If* $\models \varphi \leftrightarrow \psi$ *then* $\models \langle\!\langle\alpha\rangle\!\rangle\varphi_1 \leftrightarrow \langle\!\langle\alpha\rangle\!\rangle\varphi_2$ *(rule of equivalents for $\langle\!\langle\alpha\rangle\!\rangle$)*
3. *If* $\models \varphi_1 \leftrightarrow \varphi_2$ *then* $\models \mathrm{Stit}_G\varphi_1 \leftrightarrow \mathrm{Stit}_G\varphi_2$ *(rule of equivalents for Stit_G)*

Proposition 1 (plus the rules of equivalents for the Boolean connectives) allows to prove that the rule of replacement of equivalents preserves validity. Let $\varphi[p/\psi]$ denote the formula φ where all occurrences of the propositional variable p are replaced by ψ.

Proposition 2 (Rule of replacement of valid equivalents) *If* $\models \varphi_1 \leftrightarrow \varphi_2$ *then* $\models \psi[p/\varphi_1] \leftrightarrow \psi[p/\varphi_2]$.

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, '$\langle$', '$+$', 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 length $|\sigma|$ of a sequence of group actions σ as follows:

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

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. At first glance there are no such models because each model is infinite: 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.

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

Proposition 3 (Strong fmp) *For every* DL-PC *formula φ, if φ is* DL-PC *satisfiable then φ is satisfiable in a model of size $\mathcal{O}((|\varphi|)^{2|\varphi|})$.*

Proof Let $\mathcal{M} = \langle \mathcal{R}, \mathcal{S}, \mathcal{V}\rangle$, let φ be a formula, and let $n \in \mathbb{N}_0$ be an integer with $n \geq 0$. We do two things in order to turn $\mathcal{M}$ into a finite model: we restrict the vocabulary that is interpreted in $\mathcal{M}$ to that of φ, and we restrict the depth of the successor function by setting $\mathcal{S}(\sigma)$ to the empty set when the length of σ is greater than n. So let us define the model $\mathcal{M}_{\varphi,n} = \langle \mathcal{R}_\varphi, \mathcal{S}_{\varphi,n}, \mathcal{V}_\varphi\rangle$ by:

$$\mathcal{R}_\varphi = \mathcal{R} \cap (\mathsf{Ag}_\varphi \times \pm\mathsf{P}_\varphi)$$
$$\mathcal{S}_{\varphi,n}(\sigma) = \begin{cases} \mathcal{S}(\sigma) & \text{if } |\sigma| < n \\ \emptyset & \text{if } |\sigma| \geq n \end{cases}$$
$$\mathcal{V}_\varphi = \mathcal{V} \cap \mathsf{P}_\varphi$$

Each of $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{V}$ is finite (where finiteness of $\mathcal{S}$ is understood as having value $\emptyset$ almost everywhere), and therefore $\mathcal{M}_{\varphi,n}$ is finite. Observe that $(\mathcal{M}^\alpha)_{\varphi,n} = (\mathcal{M}_{\varphi,n+1})^\alpha$ $(*)$; moreover, observe that for every n and φ, the set of models $\mathcal{N}_{\varphi,n}$ such that $\mathcal{N} \sim_G \mathcal{M}$ equals the set of models $\mathcal{N}_{\varphi,n}$ such that $\mathcal{N} \sim_G \mathcal{M}_{\varphi,n}$ $(**)$. Basically, the last property says that 'a G-variant of $\mathcal{S}$ cut at heigth n and restricted to the vocabulary of φ' is the same thing as 'a G-variant of $\mathcal{S}_{\varphi,n}$ cut at heigth n and restricted to the vocabulary of φ'.

We prove that we have $\mathcal{M} \models \chi$ if and only if $\mathcal{M}_{\varphi,\delta(\chi)} \models \chi$ for every formula χ whose language is included in that of φ, i.e. such that $\mathsf{Ag}_\chi \subseteq \mathsf{Ag}_\varphi$ and $\mathsf{P}_\chi \subseteq \mathsf{P}_\varphi$. The proof is by induction on the structure of χ. The only delicate cases are those of the modal operators. We only give those of $\langle\alpha\rangle$ and Stit_G, the others are similar. For the former we prove:

$$\begin{aligned} \mathcal{M} \models \langle\alpha\rangle\chi \text{ iff } & \alpha \subseteq \mathcal{R} \text{ and } \mathcal{M}^\alpha \models \chi \\ \text{iff } & \alpha \subseteq \mathcal{R}_\varphi \text{ and } (\mathcal{M}^\alpha)_{\varphi,\delta(\chi)} \models \chi && \text{(by I.H.)} \\ \text{iff } & \alpha \subseteq \mathcal{R}_\varphi \text{ and } (\mathcal{M}_{\varphi,\delta(\chi)+1})^\alpha \models \chi && \text{(by } (*)) \\ \text{iff } & \alpha \subseteq \mathcal{R}_\varphi \text{ and } (\mathcal{M}_{\varphi,\delta(\langle\alpha\rangle\chi)})^\alpha \models \chi \\ \text{iff } & \mathcal{M}_{\varphi,\delta(\langle\alpha\rangle\chi)} \models \langle\alpha\rangle\chi \end{aligned}$$

For the stit operators we have to apply the induction hypothesis twice:

$$\begin{aligned}\mathcal{M} \models \mathrm{Stit}_G\chi \text{ iff } & \mathcal{N} \models \chi \text{ for every } \mathcal{N} \text{ such that } \mathcal{N} \sim_G \mathcal{M} \\ \text{iff } & \mathcal{N}_{\varphi,\delta(\chi)} \models \chi \text{ for every } \mathcal{N} \text{ such that } \mathcal{N} \sim_G \mathcal{M} && \text{(by I.H.)} \\ \text{iff } & \mathcal{N}_{\varphi,\delta(\chi)} \models \chi \text{ for every } \mathcal{N} \text{ such that } \mathcal{N} \sim_G \mathcal{M}_{\varphi,\delta(\chi)} && \text{(by (**))} \\ \text{iff } & \mathcal{N} \models \chi \text{ for every } \mathcal{N} \text{ such that } \mathcal{N} \sim_G \mathcal{M}_{\varphi,\delta(\chi)} && \text{(by I.H.)} \\ \text{iff } & \mathcal{M}_{\varphi,\delta(\chi)} \models \mathrm{Stit}_G\chi\end{aligned}$$

This ends the proof. ■

Proposition 4 *(Decidability of satisfiability) The* DL-PC *satisfiability problem is decidable.*

Proof This follows by [18, Theorem 6.7] from the above strong fmp (Proposition 3) and the fact that the set of DL-PC models of a given size is recursive (plus the fact that model checking is decidable). ■

We can prove that the satisfiability problem is PSPACE hard by encoding the QBF satisfiability problem: this is already the case for the fragment of the DL-PC language without the next operator, as we will establish in Sect. 4.3. We conjecture that it is also PSPACE complete, but leave a formal proof to future work.

3 The Fragment Without Stit Operators

We now investigate the fragment of DL-PC without stit operators. We provide a decision procedure in terms of reduction axioms.

3.1 *Simplifying* $\langle\langle\alpha\rangle\rangle$, $\langle\alpha\rangle$, and X

The first step is to simplify formulas of the form $\langle\langle\alpha\rangle\rangle\varphi$.

Proposition 5 (Simplification of $\langle\langle\alpha\rangle\rangle$)

1. $\models \langle\langle\alpha\rangle\rangle\varphi \leftrightarrow (\langle\alpha\rangle\varphi \wedge \langle\langle\alpha\rangle\rangle\top)$
2. $\models \langle\langle\emptyset\rangle\rangle\top \leftrightarrow \top$
3. $\models \langle\langle\alpha \cup \beta\rangle\rangle\top \leftrightarrow (\langle\langle\alpha\rangle\rangle\top \wedge \langle\langle\beta\rangle\rangle\top)$

Proof

1. First, $\langle\langle\alpha\rangle\rangle\varphi \to \langle\alpha\rangle\varphi$ is valid because $\mathcal{S}(\text{nil}) \subseteq \mathcal{R}$. Second, $(\langle\alpha\rangle\varphi \wedge \langle\langle\alpha\rangle\rangle\top) \to \langle\langle\alpha\rangle\rangle\varphi$ is valid because updates are functional.
2. This follows from the observation that for every model $\mathcal{M}$, $\mathcal{M} \models \langle\langle\emptyset\rangle\rangle\top$.
3. This follows from the observation that for every model $\mathcal{M}$, $\mathcal{M} \models \langle\langle\alpha\rangle\rangle\top$ iff $\alpha \subseteq \mathcal{S}(\text{nil})$. ■

According to the preceding proposition we can suppose w.l.o.g. that every occurrence of $\langle\!\langle \alpha \rangle\!\rangle$ is followed by $\top$ and that α is a singleton.

The second step is to put formulas of the form $\langle \alpha \rangle \varphi$ without Stit operators in φ in a particular normal form.

Proposition 6 (Simplification of $\langle \alpha \rangle$)

1. $\models \langle \alpha \rangle p \leftrightarrow \begin{cases} \langle \alpha \rangle \top & \text{if there is } i \text{ s.th. } (i, +p) \in \alpha \text{ and there is no } j \text{ s.th. } (j, -p) \in \alpha \\ \bot & \text{if there is } i \text{ s.th. } (i, -p) \in \alpha \text{ and there is no } j \text{ s.th. } (j, +p) \in \alpha \\ \langle \alpha \rangle \top \wedge p & \text{either if there are } i, j \text{ such that } (i, +p), (j, -p) \in \alpha \\ & \text{or if there are no } i, j \text{ such that } (i, +p), (j, -p) \in \alpha \end{cases}$

2. $\models \langle \alpha \rangle \neg\varphi \leftrightarrow (\langle \alpha \rangle \top \wedge \neg \langle \alpha \rangle \varphi)$

3. $\models \langle \alpha \rangle (\varphi \wedge \psi) \leftrightarrow (\langle \alpha \rangle \varphi \wedge \langle \alpha \rangle \psi)$

4. $\models \langle \alpha \rangle \langle \beta \rangle \top \leftrightarrow (\langle \alpha \rangle \top \wedge \langle \beta \rangle \top)$

5. $\models \langle \emptyset \rangle \top \leftrightarrow \top$

6. $\models \langle \alpha \cup \beta \rangle \top \leftrightarrow (\langle \alpha \rangle \top \wedge \langle \beta \rangle \top)$

Proof

1. This is clear from the definition of valuation update.
2. From the left to the right, $\langle \alpha \rangle \neg\varphi \rightarrow \neg \langle \alpha \rangle \varphi$ because updates are functions (as opposed to relations). From the right to the left, suppose $\mathcal{M} \models \langle \alpha \rangle \top \wedge \neg \langle \alpha \rangle \varphi$; then $\alpha \subseteq \mathcal{R}$ and $\mathcal{M}^{\alpha} \not\models \varphi$, i.e. $\mathcal{M} \models \langle \alpha \rangle \neg\varphi$.
3. From the left to the right, $\langle \alpha \rangle (\varphi \wedge \psi) \rightarrow (\langle \alpha \rangle \varphi \wedge \langle \alpha \rangle \psi)$ is valid because $\langle \alpha \rangle$ is a normal diamond operator. From the right to the left, $(\langle \alpha \rangle \varphi \wedge \langle \alpha \rangle \psi) \rightarrow \langle \alpha \rangle (\varphi \wedge \psi)$ is valid because updates are functions.
4. $\langle \alpha \rangle \langle \beta \rangle \top \leftrightarrow (\langle \alpha \rangle \top \wedge \langle \beta \rangle \top)$ is valid because the repertoire is not modified by the update. ■

For example, the formula $\langle (i, +p), (j, -q) \rangle \langle (i, -r) \rangle p$ can be rewritten as follows:

$$\begin{aligned} \langle (i, +p), (j, -q) \rangle \langle i, -r \rangle p &\leftrightarrow \langle (i, +p), (j, -q) \rangle (\langle i, -r \rangle \top \wedge p) \\ &\leftrightarrow \langle (i, +p), (j, -q) \rangle \langle i, -r \rangle \top \wedge \langle (i, +p), (j, -q) \rangle p \\ &\leftrightarrow \langle (i, +p), (j, -q) \rangle \top \wedge \langle i, -r \rangle \top \wedge \langle (i, +p), (j, -q) \rangle \top \\ &\leftrightarrow \langle i, +p \rangle \top \wedge \langle j, -q \rangle \top \wedge \langle i, -r \rangle \top \end{aligned}$$

The third step deals with the ‘next’ operator and relies on finiteness of the set of agents Ag.

Proposition 7 (Simplification of X)

1. $\models \mathrm{X}p \leftrightarrow \Big(((\bigvee_{i \in \mathsf{Ag}} \langle\!\langle i, +p \rangle\!\rangle \top) \wedge \neg(\bigvee_{j \in \mathsf{Ag}} \langle\!\langle j, -p \rangle\!\rangle \top)) \vee (p \wedge \neg(\bigvee_{j \in \mathsf{Ag}} \langle\!\langle j, -p \rangle\!\rangle \top)) \Big)$

2. $\models \mathrm{X}\neg\varphi \leftrightarrow \neg \mathrm{X}\varphi$

3. $\models \mathrm{X}(\varphi \wedge \psi) \leftrightarrow (\mathrm{X}\varphi \wedge \mathrm{X}\psi)$

4. $\models \mathrm{X}\langle \alpha \rangle \top \leftrightarrow \langle \alpha \rangle \top$

Proof The first equivalence is clear from the definition of valuation update. The second and third are familiar from linear-time temporal logic. The last equivalence is valid because the repertoire is not modified by the update. ■

3.2 Modal Atoms and Successor Function Atoms

Rewriting a formula without stit operators by applying the equivalences of propositions 5, 6, and 7 we obtain a Boolean combination of *modal atoms*. A modal atom is either a propositional variable from P, or a *repertoire atom* $\langle i, e\rangle\top$, or a *successor atom* $\langle\langle i, e\rangle\rangle\top$ that is preceded by a sequence of operators either $\langle\alpha\rangle$ or X. We call the latter kind of modal atoms *successor function atoms*, abbreviated SFA, and write $\mu\langle\langle i, e\rangle\rangle\top$ for such successor atoms. The sequence μ of operators $\langle\alpha\rangle$ and X is called a *modality*. The grammar of modal atoms π is therefore:

$$\pi ::= p \mid \langle i, e\rangle\top \mid \mu\langle\langle i, e\rangle\rangle\top$$

For a given SFA $\mu\langle\langle i, e\rangle\rangle\top$, the formula $\mu\langle i, e\rangle\top$ denotes the result of the replacement of $\langle\langle i, e\rangle\rangle$ by $\langle i, e\rangle$.

For a given Boolean combination of modal atoms φ, the set SFA_φ denotes the set of successor function atoms of φ. For example, for

$$\varphi = \neg(p \wedge (\langle i, +p\rangle\top \rightarrow \neg \mathrm{X}\langle\langle j, +q\rangle\rangle\top))$$

we have

$$\mathrm{SFA}_\varphi = \{\mathrm{X}\langle\langle j, +q\rangle\rangle\top\}.$$

Proposition 8 *Let φ be a formula without stit operators. Then φ is equivalent to a Boolean combination of modal atoms of length quadratic in the length of φ.*

Proof Every formula φ without stit operators can be transformed into an equivalent formula by applying the equivalences of propositions 5, 6, and 7 from the left to the right.

All the resulting formulas are equivalent to the original formula due to the rule of replacement of valid equivalents (Proposition 2).

The resulting formula is of length quadratic in the length of φ because the procedure basically consists in shifting the modal operators $\langle.\rangle$, $\langle\langle.\rangle\rangle$, and X in front of the atomic formulas: in the worst case every atom gets prefixed by a sequence of modal operators whose length is in the order of the length of φ. ■

3.3 Decision Procedure

We now translate formulas that are Boolean combinations of modal atoms into formulas of classical propositional logic as follows:

$$\begin{aligned}
\tau(\top) &= \top \\
\tau(p) &= \nu_p \\
\tau(\langle i, e\rangle\top) &= \nu_{\langle i,e\rangle\top} \\
\tau(\mu\langle\!\langle i, e\rangle\!\rangle\top) &= \nu_{\mu\langle\!\langle i,e\rangle\!\rangle\top} \\
\tau(\neg\varphi) &= \neg\tau(\varphi) \\
\tau(\varphi \wedge \psi) &= \tau(\varphi) \wedge \tau(\psi)
\end{aligned}$$

where ν_p ,$\nu_{\langle i,e\rangle\top}$ and $\nu_{\mu\langle\!\langle i,e\rangle\!\rangle\top}$ are fresh propositional variables that do not occur in the formula to be translated. Our translation therefore just identifies modal atoms with distinct propositional variables.

Proposition 9 *Let φ be a* DL-PC *formula that is a Boolean combination of modal atoms. φ is* DL-PC *satisfiable if and only if $\tau(\varphi) \wedge (\bigwedge \Gamma_\varphi)$ is satisfiable in classical propositional logic, where*

$$\Gamma_\varphi = \{\nu_{\mu\langle\!\langle i,e\rangle\!\rangle\top} \to \nu_{\langle i,e\rangle\top} : \mu\langle\!\langle i, e\rangle\!\rangle\top \in \text{SFA}_\varphi\}$$

Proof Let φ be a DL-PC formula that is a Boolean combination of modal atoms.

From the left to the right, suppose $\mathcal{M} \models \varphi$. We transform $\mathcal{M}$ into an interpretation $\mathcal{I}_\mathcal{M}$ of classical propositional logic by associating the 'right' truth values to the propositional variables that stand for modal atoms: we set

$$\mathcal{I}_\mathcal{M}(\nu_\pi) = 1 \text{ if and only if } \mathcal{M} \models \pi$$

where π is a modal atom. It is then straightforward to prove by induction on the form of ψ that $\mathcal{M} \models \psi$ if and only if $\mathcal{I}(\tau(\psi)) = 1$, for every DL-PC formula ψ. Moreover, $\mathcal{I}(\bigwedge \Gamma_\varphi) = 1$ because the successor function $\mathcal{S}$ respects the repertoire function $\mathcal{R}$: whenever $\mathcal{M} \models \mu\langle\!\langle i, e\rangle\!\rangle\top$ for some SFA $\mu\langle\!\langle i, e\rangle\!\rangle\top$ then we have $\mathcal{M} \models \langle i, e\rangle\top$.

From the right to the left, suppose $\mathcal{I}(\tau(\varphi) \wedge (\bigwedge \Gamma_\varphi)) = 1$. We may suppose w.l.o.g. that $\mathcal{I}(\nu_{\mu\langle\!\langle i,e\rangle\!\rangle\top}) = 0$ for all those $\nu_{\mu\langle\!\langle i,e\rangle\!\rangle\top}$ such that the SFA $\mu\langle\!\langle i, e\rangle\!\rangle\top$ does not belong to SFA_φ. We build a DL-PC model $\mathcal{M}_\mathcal{I} = \langle \mathcal{R}_\mathcal{I}, \mathcal{S}_\mathcal{I}, \mathcal{V}_\mathcal{I}\rangle$ by setting $\mathcal{R}_\mathcal{I} = \{(i, e) : \mathcal{I}(\nu_{\langle i,e\rangle\top}) = 1\}$, $\mathcal{V}_\mathcal{I} = \{p \in \mathsf{P} : \mathcal{I}(\nu_p) = 1\}$, and by inductively defining $\mathcal{S}_\mathcal{I}$ as follows:

$$\mathcal{S}_\mathcal{I}(\text{nil}) = \{(i, e) : \mathcal{I}(\nu_{\langle\!\langle i,e\rangle\!\rangle\top}) = 1\}$$

$$\mathcal{S}_\mathcal{I}(\alpha \cdot \sigma) = \begin{cases} \mathcal{S}_{\mathcal{I}^\alpha}(\sigma) & \text{if } \alpha \neq \mathcal{S}_\mathcal{I}(\text{nil}) \\ \mathcal{S}_{\mathcal{I}^\text{X}}(\sigma) \cup \{(i_0, +\nu)\} & \text{if } \alpha = \mathcal{S}_\mathcal{I}(\text{nil}) \end{cases}$$

for somei_0 and some fresh ν, where updates of the SFA part of interpretation $\mathcal{I}$ (more precisely, updates of the fresh variables $\nu_{\mu\langle\langle i,e\rangle\rangle\top}$ associated to the SFAs of φ) are defined in the obvious way:

$$\mathcal{I}^{\alpha} = \{\nu_{\mu\langle\langle i,e\rangle\rangle\top} : \nu_{\langle\alpha\rangle\mu\langle\langle i,e\rangle\rangle\top} \in \mathcal{I}\}$$
$$\mathcal{I}^{\mathrm{X}} = \{\nu_{\mu\langle\langle i,e\rangle\rangle\top} : \nu_{\mathrm{X}\mu\langle\langle i,e\rangle\rangle\top} \in \mathcal{I}\}$$

Note that $\mathcal{S}_{\mathcal{I}^{\alpha}} = (\mathcal{S}_{\mathcal{I}})^{\alpha}$ and $\mathcal{S}_{\mathcal{I}^{\mathrm{X}}} = (\mathcal{S}_{\mathcal{I}})^{\mathcal{S}_{\mathcal{I}}(\mathrm{nil})}$. Note also that in the inductive definition of $\mathcal{S}_{\mathcal{I}}$, when $\alpha = \mathcal{S}_{\mathcal{I}}(\mathrm{nil})$ then the 'fresh action' $(i_0, +\nu)$ makes that $\mathcal{S}_{\mathcal{I}}$ is well-defined: it avoids a conflict between e.g. $\mathcal{S}_{\mathcal{I}}(\mathcal{S}_{\mathcal{I}}(\mathrm{nil}))$ and $\mathcal{S}_{\mathcal{I}}(\alpha)$ for some SFA $\langle\alpha\rangle\mu \in \mathrm{SFA}_{\varphi}$ because $\mathcal{S}_{\mathcal{I}}(\mathrm{nil})$ differs from any group action α coming from φ.[2] The triple $\mathcal{M}$ that we have constructed in this way is indeed a DL-PC model: it satisfies the constraint that every $\mathcal{S}(\sigma)$ is included in $\mathcal{R}$ because (1) $\mathcal{I}(\bigwedge \Gamma_{\varphi}) = 1$ and because (2) $\mathcal{I}(\nu_{\mu\langle\langle i,e\rangle\rangle\top}) = 0$ for all those $\mu\langle\langle i, e\rangle\rangle\top$ not in SFA_{φ}. Now we prove, first, that for every modal atom π occurring in φ we have $\mathcal{M} \models \pi$ if and only if $\mathcal{I}(\pi) = 1$. The case of successor function atoms $\mu\langle\langle i, e\rangle\rangle\top$ is proved by induction on its length. In the induction step we use that (1) $\mathcal{I}^{\alpha}(\nu_{\mu\langle\langle i,e\rangle\rangle\top}) = 1$ iff $\mathcal{I}(\nu_{\langle\alpha\rangle\mu\langle\langle i,e\rangle\rangle\top}) = 1$ and that (2) $\mathcal{I}^{\mathrm{X}}(\nu_{\mu\langle\langle i,e\rangle\rangle\top}) = 1$ iff $\mathcal{I}(\nu_{\mathrm{X}\mu\langle\langle i,e\rangle\rangle\top}) = 1$. Second, the formula φ being a Boolean combination of modal atoms it clearly follows that $\mathcal{M} \models \varphi$. ■

3.4 Complexity

We have just defined a decision procedure for DL-PC formulas without stit operators. We are now going to show that that procedure works in nondeterministic polynomial time.

Proposition 10 *The problem of satisfiability of* DL-PC *formulas without the stit operator is NP complete.*

Proof The problem is clearly NP hard, given that DL-PC is a conservative extension of propositional logic.

In what concerns membership, by Proposition 8 we know that every DL-PC formula φ without stit operators is equivalent to a Boolean combination of modal atoms φ' whose length is quadratic in that of φ. According to Proposition we may check satisfiability of φ' by checking satisfiability of $\tau(\varphi') \wedge (\bigwedge \Gamma_{\varphi'})$. The length of $\tau(\varphi')$ is linear in the length of φ', and the length of $\Gamma_{\varphi'}$ is linear in the length of φ'; together, they make up a linear transformation. Overall, the length of the propositional formula $\tau(\varphi') \wedge (\bigwedge \Gamma_{\varphi'})$ is quadratic in the length of the original φ. Therefore DL-PC satisfiability is in NP.

[2] To see this, suppose that $\langle\langle i, +p\rangle\rangle\top$ is the only SFA of φ such that $\mathcal{I}(\nu_{\langle\langle i,+p\rangle\rangle\top}) = 1$. Then $\mathcal{S}_{\mathcal{I}}(\mathrm{nil}) = \{(i, +p)\}$. Now suppose $\mathcal{I}$ is such that $\mathcal{I}(\nu_{\mathrm{X}\langle\langle i,+p\rangle\rangle\top}) = 1$ and $\mathcal{I}(\nu_{\langle i,+p\rangle\langle\langle i,+p\rangle\rangle\top}) = 0$: then $\mathcal{S}_{\mathcal{I}}$ would be ill-defined if we hadn't we introduced the fresh action $(i_0, +\nu)$.

4 The Fragment Without the 'next' Operator

We now give a decision procedure for the fragment of the language of DL-PC without the temporal 'next'. The procedure amounts to the elimination of stit operators and uses some of the results of the preceding section.

4.1 G-Determinate Formulas

A formula φ is G-determinate if and only if for all DL-PC models $\mathcal{M}$ and $\mathcal{M}'$ such that $\mathcal{M} \sim_G \mathcal{M}'$ we have $\mathcal{M} \models \varphi$ iff $\mathcal{M}' \models \varphi$. Note that propositional variables are G-determinate, for every group G. The same is the case for formulas of the form $\langle\alpha\rangle\top$. Moreover, $\langle\!\langle(i, e)\rangle\!\rangle\top$ is G-determinate if $i \in G$. Note also that when a formula φ is G-determinate then the equivalence $\mathrm{Stit}_G\varphi \leftrightarrow \varphi$ is valid.

The next two propositions generalise these observations.

Proposition 11 (Some G-determinate formulas) *Let G be a group of agents.*

1. *Every propositional variable is G-determinate.*
2. *Every formula $\langle\alpha\rangle\top$ is G-determinate.*
3. *If $i \in G$ then $\langle\!\langle(i, e)\rangle\!\rangle\top$ is G-determinate.*
4. *If φ is G-determinate then $\neg\varphi$, $\langle\alpha\rangle\varphi$, and $\mathrm{X}\varphi$ are G-determinate.*

Proposition 12 (Properties of G-determinate formulas) *Let φ be G-determinate. Then* $\mathrm{Stit}_G(\varphi \vee \psi) \leftrightarrow (\varphi \vee \mathrm{Stit}_G\psi)$ *is valid.*

Here are some examples of formulas that are not G-determinate. First, when G is the set of all agents Ag then every formula is G-determinate. Second, the formula $\mathrm{X}p$ is G-determinate only when G is the set of all agents Ag. Third, $\langle\!\langle\alpha\rangle\!\rangle\top$ is not G-determinate when α_i is non-empty for some $i \notin G$.

4.2 Eliminating the Stit Operators

Consider any subformula $\mathrm{Stit}_G\psi$ of a formula φ such that ψ is a Boolean combination of modal atoms. We may suppose w.l.o.g. that ψ is in conjunctive normal form, i.e. that ψ is a conjunction of clauses, where clauses are disjunctions of modal atoms or negations thereof. For example, a conjunctive normal form of the above

$$\neg(p \wedge (\langle i, +p\rangle\top \rightarrow \neg\langle i, -p\rangle\langle\!\langle j, +q\rangle\!\rangle\top))$$

is

$$(\neg p \vee \langle i, +p\rangle\top) \wedge (\neg p \vee \langle i, -p\rangle\langle\!\langle j, +q\rangle\!\rangle\top)$$

Given a Stit_G operator followed by a formula in conjunctive normal form, we may apply the following reduction axioms.

Proposition 13 (Reduction axioms for Stit_G**)**

1. $\models \text{Stit}_G\top \leftrightarrow \top$
2. $\models \text{Stit}_G(\varphi_1 \wedge \varphi_2) \leftrightarrow (\text{Stit}_G\varphi_1 \wedge \text{Stit}_G\varphi_2)$
3. $\models \text{Stit}_G(p \vee \varphi) \leftrightarrow (p \vee \text{Stit}_G\varphi)$
4. $\models \text{Stit}_G(\neg p \vee \varphi) \leftrightarrow (\neg p \vee \text{Stit}_G\varphi)$
5. $\models \text{Stit}_G(\langle\alpha\rangle\top \vee \varphi) \leftrightarrow (\langle\alpha\rangle\top \vee \text{Stit}_G\varphi)$
6. $\models \text{Stit}_G(\neg\langle\alpha\rangle\top \vee \varphi) \leftrightarrow (\neg\langle\alpha\rangle\top \vee \text{Stit}_G\varphi)$
7. *Let* $i \in G$. *Then*

$$\models \text{Stit}_G(\mu\langle\langle i, e\rangle\rangle\top \vee \varphi) \leftrightarrow \mu\langle\langle i, e\rangle\rangle\top \vee \text{Stit}_G\varphi$$
$$\models \text{Stit}_G(\neg\mu\langle\langle i, e\rangle\rangle\top \vee \varphi) \leftrightarrow \neg\mu\langle\langle i, e\rangle\rangle\top \vee \text{Stit}_G\varphi$$

8. *Let P and Q be two finite sets of successor function atoms that are all of the form* $\mu\langle\langle i, e\rangle\rangle\top$ *with* $i \notin G$ *and that do not contain* X. *Then*

$$\models \text{Stit}_G\Big((\bigvee P) \vee \neg(\bigwedge Q)\Big) \leftrightarrow \begin{cases}\top & \textit{if } P \cap Q \neq \emptyset \\ \neg \bigwedge_{\mu\langle\langle i,e\rangle\rangle\top \in Q} \mu\langle i, e\rangle\top & \textit{if } P \cap Q = \emptyset\end{cases}$$

Proof As to Item 1, $\models \text{Stit}_G\top \leftrightarrow \top$ is valid because $\text{Stit}_G\top$ is valid (Stit_G being a normal modal box).

As to Item 2, $\models \text{Stit}_G(\varphi_1 \wedge \varphi_2) \leftrightarrow (\text{Stit}_G\varphi_1 \wedge \text{Stit}_G\varphi_2)$ is valid because Stit_G is a normal modal box.

Items 3–6 are valid because Boolean formulas and formulas of the form $\langle\alpha\rangle\top$ and $\neg\langle\alpha\rangle\top$ are G-determinate (Proposition 11, items 1 and 2) and therefore Proposition 12 applies.

For Item 7, let $i \in G$. Then according to Proposition 11, both $\mu\langle\langle i, e\rangle\rangle\top$ and $\neg\mu\langle\langle i, e\rangle\rangle\top$ are G-determinate. Therefore the following schemas are both valid:

$$\text{Stit}_G\,(\mu\langle\langle i, e\rangle\rangle\top \vee \varphi) \leftrightarrow (\mu\langle\langle i, e\rangle\rangle\top \vee \text{Stit}_G\varphi)$$
$$\text{Stit}_G\,(\neg\mu\langle\langle i, e\rangle\rangle\top \vee \varphi) \leftrightarrow (\neg\mu\langle\langle i, e\rangle\rangle\top \vee \text{Stit}_G\varphi)$$

For Item 8 we examine the two cases. First, let $P \cap Q \neq \emptyset$. As P collects the SFAs of the positive literals and Q collects the SFAs of the negative literals, the formula $(\bigvee P) \vee \neg(\bigwedge Q)$ is valid in classical propositional logic; and as Stit_G is a normal modal box, $\text{Stit}_G\big((\bigvee P) \vee \neg(\bigwedge Q)\big)$ is DL-PC valid. For the second case where $P \cap Q = \emptyset$ we prove the two directions of the equivalence separately.

- For *the left-to-right direction*, let $\mathcal{M} = \langle \mathcal{R}, \mathcal{S}, \mathcal{V}\rangle$ be a DL-PC model such that $\mathcal{M} \not\models \neg \bigwedge_{\mu\langle\langle i,e\rangle\rangle\top\in Q} \mu\langle i, e\rangle\top$, i.e. $\mathcal{M} \models \mu\langle i, e\rangle\top$ for every $\mu\langle\langle i, e\rangle\rangle\top \in Q$. Let us define a successor function $\mathcal{S}_Q$ by:

$$\mathcal{S}_Q(\text{nil}) = (\mathcal{S}(\text{nil}))_G \cup \{(i,e) : \langle\!\langle i,e\rangle\!\rangle\top \in Q\}$$
$$\mathcal{S}_Q(\alpha\cdot\sigma) = (\mathcal{S}(\alpha\cdot\sigma))_G \cup \mathcal{S}_{Q^{\langle\alpha\rangle}}(\sigma)$$

where the set $Q^{\langle\alpha\rangle}$ is defined as follows:

$$Q^{\langle\alpha\rangle} = \{\mu\langle\!\langle i,e\rangle\!\rangle\top : \langle\alpha\rangle\mu\langle\!\langle i,e\rangle\!\rangle\top \in Q\}$$

The function $\mathcal{S}_Q$ respects $\mathcal{R}$: clearly, $(\mathcal{S}(\text{nil}))_G$ respects $\mathcal{R}$, and we can prove by induction on the length of σ that $\mathcal{S}_{Q^\mu}(\sigma)$ respects $\mathcal{R}$ for every modality μ, where the set of modal atoms Q^μ generalises the set $Q^{\langle\alpha\rangle}$ in the obvious way.
Let $\mathcal{M}_Q = \langle\mathcal{R},\mathcal{S}_Q,\mathcal{V}\rangle$. First, we have $\mathcal{M} \sim_G \mathcal{M}_Q$ because $(\mathcal{S}(\text{nil}))_G = (\mathcal{S}_Q(\text{nil}))_G$. Second, we can prove that $\mathcal{M}_Q \models \mu\langle\!\langle i,e\rangle\!\rangle\top$ iff $\mu\langle\!\langle i,e\rangle\!\rangle\top \in Q$, for every successor function atom $\mu\langle\!\langle i,e\rangle\!\rangle\top$ such that $i \notin G$. It follows that $\mathcal{M}_Q \models \mu\langle\!\langle i,e\rangle\!\rangle\top$ for every $\mu\langle\!\langle i,e\rangle\!\rangle\top \in Q$, and as $P\cap Q = \emptyset$ we also have $\mathcal{M}_Q \not\models \mu\langle\!\langle i,e\rangle\!\rangle\top$ for every $\mu\langle\!\langle i,e\rangle\!\rangle\top \in P$. Therefore $\mathcal{M}_Q \models (\bigwedge Q)\wedge\neg(\bigvee P)$, i.e. $\mathcal{M}_Q \not\models (\bigvee P)\vee\neg(\bigwedge Q)$. According to the truth condition for Stit_G this means that $\mathcal{M} \not\models \text{Stit}_G\big((\bigvee P)\vee\neg(\bigwedge Q)\big)$.

- For *the right-to-left direction*, suppose $\mathcal{M} \models \neg\bigwedge_{\mu\langle\!\langle i,e\rangle\!\rangle\top\in Q}\mu\langle i,e\rangle\top$, i.e. $\mathcal{M} \not\models \mu\langle i,e\rangle\top$ for some $\mu\langle\!\langle i,e\rangle\!\rangle\top \in Q$. So either $(i,e) \notin \mathcal{R}$, or $\alpha \not\subseteq \mathcal{R}$ for some dynamic operator $\langle\alpha\rangle$ of the sequence μ. In the first case $\mathcal{M} \not\models \mu\langle\!\langle i,e\rangle\!\rangle\top$ because the successor function respects the repertoire; and in the second case $\mathcal{M} \not\models \mu\top$ because of the truth condition for $\langle\alpha\rangle$, and therefore $\mathcal{M} \not\models \mu\langle\!\langle i,e\rangle\!\rangle\top$, too. The formula $\mu\langle\!\langle i,e\rangle\!\rangle\top$ is actually false in *every* model $\mathcal{M}'$ such that $\mathcal{M} \sim_G \mathcal{M}'$ (in the first case because every $\mathcal{S}'$ respects $\mathcal{R}$; in the second case because when interpreting μ the truth condition for $\langle\alpha\rangle$ checks whether $\alpha \subseteq \mathcal{R}$). It follows that $\mathcal{M}' \not\models \bigwedge Q$ for every model $\mathcal{M}'$ such that $\mathcal{M} \sim_G \mathcal{M}'$. Hence $\mathcal{M}' \models (\bigvee P)\vee\neg(\bigwedge Q)$ for every model $\mathcal{M}'$ such that $\mathcal{M} \sim_G \mathcal{M}'$, from which it follows that $\mathcal{M} \models \text{Stit}_G\big((\bigvee P)\vee\neg(\bigwedge Q)\big)$.

This concludes the proof of Item 8. ∎

For example, the formula $\text{Stit}_i\,(\neg p \vee \langle i,+p\rangle\top \vee \langle i,+p\rangle\langle\!\langle j,+q\rangle\!\rangle\top)$ can be rewritten as follows:

$$\begin{aligned}\text{Stit}_i\,(\neg p \vee \langle i,+p\rangle\top \vee \langle i,+p\rangle\langle\!\langle j,+q\rangle\!\rangle\top) &\leftrightarrow \neg p \vee \langle i,+p\rangle\top \vee \text{Stit}_i\,\langle i,+p\rangle\langle\!\langle j,+q\rangle\!\rangle\top\\ &\leftrightarrow \neg p \vee \langle i,+p\rangle\top \vee \bot\end{aligned}$$

Anticipating a bit, we observe that the first two items of Proposition 13 are also valid in the logic of the Chellas stit, while the third and the fourth item are only valid if the values of the propositional variables are moment-determinate. (Validity in the logic of the Chellas stit and moment-determinateness are going to be defined in Sect. 5.)

Applying the above equivalences from the left to the right allows to entirely eliminate the stit operators. It follows that we can transform every formula without the 'next' operator into an equivalent Boolean combination of modal atoms.

Theorem 14 *Every* DL-PC *formula without* X *is equivalent to a Boolean combination of modal atoms.*

Note that Item 7 of Proposition 13 also holds for the more general case where μ contains the temporal X. Item 8 does not: let $P = \{\langle i,e\rangle\langle\!\langle i,e\rangle\!\rangle\top\}$ and let $Q = \{\langle\!\langle i,e\rangle\!\rangle\top, \mathrm{X}\langle\!\langle i,e\rangle\!\rangle\top\}$. Then the formula $\mathrm{Stit}_\emptyset\big((\bigvee P) \vee \neg(\bigwedge Q)\big)$ is not equivalent to $\neg(\langle i,e\rangle\top \wedge \mathrm{X}\langle i,e\rangle\top)$, i.e., to $\neg\langle i,e\rangle\top$. To see this consider any model $\mathcal{M}$ where $\mathcal{R} = \mathcal{S}(\mathrm{nil}) = \mathcal{S}((i,e)\cdot\mathrm{nil}) = \{(i,e)\}$: while $\mathrm{Stit}_\emptyset\big((\bigvee P) \vee \neg(\bigwedge Q)\big)$ is true in $\mathcal{M}$, $\neg\langle i,e\rangle\top$ is not. We were not able to find reduction axioms for the whole language of DL-PC.

4.3 Complexity Of Satisfiability: A Lower Bound

Proposition 15 (Complexity, lower bound) *The* DL-PC *satisfiability problem is PSPACE hard even for formulas without the* X *operator.*

Proof We establish the proof by encoding the quantified Boolean formula (QBF) satisfiability problem into the fragment of DL-PC without the next operator. We view an interpretation of classical propositional logic as a mapping $\mathcal{I}$ from the set of propositional variables into {0, 1} (that is extended to evaluate any Boolean formula in the standard way).

Let φ_0 be a QBF to be translated. Define a translation t from the language of QBFs to the language of DL-PC as follows:

$$t(p) = \langle\!\langle p, +p\rangle\!\rangle\top$$
$$t(\forall p\varphi) = \mathrm{Stit}_{\mathsf{P}_{\varphi_0}\setminus\{p\}}t(\varphi)$$

and homomorphic for the other connectives. (We therefore translate propositional variables into agent names, supposing therefore that there are at least as many agent names in Ag as there are propositional variables in P.)

Define the set Γ_{φ_0} as:

$$\Gamma_{\varphi_0} = \{\langle p, +p\rangle\top : p \in \mathsf{P}_{\varphi_0}\} \cup \{\langle p, -p\rangle\top : p \in \mathsf{P}_{\varphi_0}\}$$

We prove that the QBF φ_0 is satisfiable if and only if $t(\varphi_0) \wedge (\bigwedge \Gamma_{\varphi_0})$ is satisfiable.

From the left to the right, suppose I is an interpretation of classical propositional logic such that $I(\varphi_0) = 1$. We define a DL-PC model $\mathcal{M}_I = \langle \mathcal{R}_I, \mathcal{S}_I, \mathcal{V}_I\rangle$ such that

$$\mathcal{R}_I = \{(p, +p) : p \in \mathsf{P}_{\varphi_0}\} \cup \{(p, -p) : p \in \mathsf{P}_{\varphi_0}\}$$

$$\mathcal{S}_I(\sigma) = \begin{cases} \{(p, +p) : p \in \mathsf{P}_{\varphi_0} \text{ and } I(p) = 1\} & \text{if } \sigma = \text{nil} \\ \mathcal{S}_I(\sigma) = \emptyset & \text{if } \sigma \neq \text{nil} \end{cases}$$
$$\mathcal{V} = \emptyset$$

Clearly $\mathcal{M}_I \models \bigwedge \Gamma_{\varphi_0}$. It then suffices to prove by induction that $I(\varphi) = 1$ iff $\mathcal{M}_I \models t(\varphi)$, for every subformula φ of φ_0.

From the right to the left, suppose $\mathcal{M}$ is a DL-PC model such that $\mathcal{M} \models t(\varphi_0) \wedge (\bigwedge \Gamma_{\varphi_0})$. We define an interpretation $I_\mathcal{M}$ of the propositional variables p occurring in φ_0 by: $I_\mathcal{M}(p) = 1$ iff $(p, +p) \in \mathcal{S}(\text{nil})$. We then prove by induction that $\mathcal{M} \models t(\varphi)$ iff $I_\mathcal{M}(\varphi) = 1$, for every subformula φ of φ_0. ■

5 Relation with Chellas Stit

We now investigate the relationship between our Stit operator and the Chellas stit logic [8–10]. The language of that logic has a stit operator just as DL-PC. It moreover has temporal operators that are not part of DL-PC. We therefore extend the language of DL-PC by the simplest temporal operator, viz. the temporal ‘next’ operator, and compare that extension with a discrete version of the Chellas stit logic as introduced in [19].

5.1 *Chellas Stit Logic*

The language of the discrete Chellas stit logic is nothing but the fragment $\mathcal{L}_{\text{Stit},\text{X}}$ of the language of DL-PC without the dynamic operators. The set of formulas φ is defined by the following BNF:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \text{Stit}_G\varphi \mid \text{X}\varphi$$

The reading of $\text{Stit}_G\varphi$ and $\text{X}\varphi$ is the same as before.

The formulas are interpreted in discrete *Branching Time models with Agent Choice functions* (discrete BT+AC models). These models are defined in two steps.

First, a discrete branching time structure (BT) is a pair $\langle \mathsf{Mom}, < \rangle$, where:

- Mom is a non-empty set of moments.
- $<$ is a tree-like partial ordering that is irreflexive and discrete. We recall that an ordering $<$ is discrete if and only if for every $m \in \mathsf{Mom}$ there is a set of closest moments $\text{succ}(m)$ such that for every $m' \in \text{succ}(m)$, $m < m'$ and there is no $m'' \in \mathsf{Mom}$ with $m < m'' < m'$.

Then a *history* is a maximally $<$-ordered set of moments. We use $\mathcal{H}$ to denote the set of all histories and $\mathcal{H}_m$ to denote the set of histories passing through the moment m, i.e., the set of histories h such that $m \in h$. The successor function can be extended to moment-history pairs: $\mathrm{succ}(m, h)$ is the moment m' such that $\mathrm{succ}(m) \cap h = \{m'\}$. Two histories $h_1, h_2 \in \mathcal{H}_m$ are *undivided* at m if and only if both histories have the same successor to m, i.e., if and only if $\mathrm{succ}(m, h_1) = \mathrm{succ}(m, h_2)$.[3]

Second, a discrete BT+AC model is a quadruple of the form

$$\mathcal{M} = \langle \mathsf{Mom}, <, \mathcal{C}, \mathsf{Val} \rangle$$

where $\langle \mathsf{Mom}, < \rangle$ is a discrete branching time structure and where $\mathcal{C}$ and Val are as follows.

- $\mathcal{C}$ is function from $\mathsf{Ag} \times \mathsf{Mom}$ to $\mathcal{H} \times \mathcal{H}$ such that each $\mathcal{C}(i, m)$ is an equivalence relation on $\mathcal{H}_m$.[4] It is assumed that $\mathcal{C}$ satisfies the following constraints:
  1. Independence of agents: for every moment m and for every mapping $H : \mathsf{Ag} \longrightarrow \mathcal{H}_m$ there is a history $h \in \mathcal{H}_m$ such that $(H(i), h) \in \mathcal{C}(i, m)$ for every $i \in \mathsf{Ag}$.[5]
  2. No choice between undivided histories: if two histories h_1 and h_2 are undivided at m then $(h_1, h_2) \in \mathcal{C}(i, m)$ for every agent i.
- Val is a valuation function from $\mathsf{Mom} \times \mathcal{H}$ to 2^{P}.

The constraint of independence of agents says that any individual choice is compatible with the other choices. The constraint of no choice between undivided histories says that if two histories are undivided at m, then no possible choice for any agent at m distinguishes between the two histories: for every agent i, both h_1 and h_2 belong to the same choice cell at m.

Choice functions are extended from agents to groups of agents by stipulating:

$$\mathcal{C}(G, m) = \bigcap_{i \in G} \mathcal{C}(i, m)$$

Note that with this definition the above 'no choice between undivided histories' constraint can be formulated as: if $m \in h_1 \cap h_2$ and $m_0 < m$ then $(h_1, h_2) \in \mathcal{C}(\mathsf{Ag}, m_0)$.

The values of the propositional variables are said to be *moment-determinate* if $\mathsf{Val}(m, h) = \mathsf{Val}(m, h')$ for every $h, h' \in \mathcal{H}_m$.

Let $\mathcal{M} = \langle \mathsf{Mom}, <, \mathcal{C}, \mathsf{Val} \rangle$ be a BT+AC model as defined above. A *pointed BT+AC model* is a pair $(\mathcal{M}, m/h)$ such that where $m \in h$ and $h \in \mathcal{H}$. The satisfaction relation $\models$ is defined between the formulas and pointed BT+AC models as follows:

[3] The original definition is equivalent to ours in the case of discrete BT structures: it stipulates that there is some m' such that $m < m'$ and m' belongs to both h_1 and h_2.

[4] The original definition is equivalent: $\mathcal{C}$ is function from $\mathsf{Ag} \times \mathsf{Mom}$ to $2^{2^{\mathcal{H}}}$ mapping each agent and each moment into a partition of $\mathcal{H}_m$.

[5] The original definition is: for every moment m, if H_i is some set in $\mathcal{C}(i, m)$ for every $i \in \mathsf{Ag}$ then $\bigcap_{i \in \mathsf{Ag}} H_i \neq \emptyset$.

$$\begin{array}{lll}
\mathcal{M}, m/h \models p & \text{iff} & p \in \mathsf{Val}(m,h) \\
\mathcal{M}, m/h \models \mathrm{Stit}_G\varphi & \text{iff} & \text{for all } h' \text{ such that } (h,h') \in \mathcal{C}(G,m),\ \mathcal{M}, m/h' \models \varphi \\
\mathcal{M}, m/h \models \mathrm{X}\varphi & \text{iff} & \mathcal{M}, \mathrm{succ}(m,h)/h \models \varphi
\end{array}$$

and as usual for the Boolean operators.

A formula φ is *valid* if and only if $\mathcal{M}, m/h \models \varphi$ for every BT+AC model $\mathcal{M}$, every history h of $\mathcal{M}$, and every moment $m \in h$.

For example, the schema $\mathrm{Stit}_\emptyset\varphi \to \mathrm{Stit}_G\varphi$ is valid, and the schema $\mathrm{Stit}_{G_1}\mathrm{Stit}_{G_2}\varphi \to \mathrm{Stit}_\emptyset\varphi$ is valid if $G_1 \cap G_2 = \emptyset$. Each of the modal operators Stit_G is an S5 operator: the schemas $\mathrm{Stit}_G\varphi \to \varphi$, $\mathrm{Stit}_G\varphi \to \mathrm{Stit}_G\mathrm{Stit}_G\varphi$, and $\neg\mathrm{Stit}_G\varphi \to \mathrm{Stit}_G\neg\mathrm{Stit}_G\varphi$ are all valid, and the rule of necessitation preserves validity.

5.2 DL-PC Models as Particular BT+AC Models

We are now going to relate the discrete Chellas stit logic to DL-PC: we show that DL-PC models can be viewed as particular discrete BT+AC models. A similar technique has been used in [20].

Let $\mathcal{M} = \langle \mathcal{R}, \mathcal{S}, \mathcal{V} \rangle$ be a DL-PC model. The translation of $\mathcal{M}$ into a discrete BT+AC model is the structure $\mathrm{tr}(\mathcal{M}) = \langle \mathsf{Mom}, <, \mathcal{C}, \mathsf{Val} \rangle$, where:

- $\mathsf{Mom} = (2^{\mathcal{R}})^*$ (the set of sequences of group actions respecting $\mathcal{R}$)
- $\sigma < \sigma'$ if and only if there is $\sigma'' \neq \mathrm{nil}$ such that $\sigma' = \sigma \cdot \sigma''$ (prefix relation)
- for every agent $i \in \mathsf{Ag}$ and every moment $\sigma \in \mathsf{Mom}$,

$$\mathcal{C}(i,\sigma) = \{(h,h') : \text{ there are } \alpha, \alpha' \text{ such that } \sigma \cdot \alpha \in h, \sigma \cdot \alpha' \in h', \text{ and } \alpha_i = \alpha'_i\}$$

- Val is recursively defined by:

$$\begin{array}{ll}
\mathsf{Val}(\mathrm{nil}, h) = \mathcal{V} & \text{(for every } h) \\
\mathsf{Val}(\sigma \cdot \alpha, h) = (\mathsf{Val}(\sigma, h))^\alpha & \text{(for every } h)
\end{array}$$

In the last line, $(\mathsf{Val}(\sigma, h))^\alpha$ is the update of the valuation $\mathsf{Val}(\sigma, h)$ by α as defined in Sect. 2.3.

Note that the successor function of $\mathcal{M}$ does not play any role in the definition. We therefore have the following.

Proposition 16 *Let* $\langle \mathcal{R}, \mathcal{S}, \mathcal{V} \rangle$ *and* $\langle \mathcal{R}, \mathcal{S}', \mathcal{V} \rangle$ *be two* DL-PC *models. Then* $\mathrm{tr}(\langle \mathcal{R}, \mathcal{S}, \mathcal{V} \rangle) = \mathrm{tr}(\langle \mathcal{R}, \mathcal{S}', \mathcal{V} \rangle)$.

Note further that $\mathrm{succ}(\sigma) = \{\sigma \cdot \alpha : \alpha \in \mathcal{R}\}$.

Proposition 17 *If* $\mathcal{M}$ *is a* DL-PC *model then* $\mathrm{tr}(\mathcal{M})$ *is a discrete BT+AC model.*

Proof First, $(\mathsf{Mom}, <)$ is a discrete BT structure because the prefix relation is a tree-like partial ordering. Let us show that $\mathrm{tr}(\mathcal{M}) = \langle \mathsf{Mom}, <, \mathcal{C}, \mathsf{Val}\rangle$ satisfies the two constraints for choice functions: 'independence of agents' and 'no choice between undivided histories'.

Let $\sigma \in \mathsf{Mom}$ be some moment and let $H : \mathsf{Ag} \longrightarrow \mathcal{H}_\sigma$ be some mapping. For every i, let $\alpha(i)$ be such that $\mathrm{succ}(\sigma, H(i)) = \sigma \cdot \alpha(i)$. Let $\alpha = \bigcup_{i \in \mathsf{Ag}} (\alpha(i))_i$. α is composed of the agents' choices at 'their' history $H(i)$. Clearly $\alpha \subseteq \mathcal{R}$, and therefore $\sigma \cdot \alpha \in \mathsf{Mom}$. Let $h \in \mathcal{H}_\sigma$ be any history such that $\sigma \cdot \alpha \in h$. (Such a history exists; take for example the history where none of the agents acts after $\sigma \cdot \alpha$.) Then $(h, H(i)) \in \mathcal{C}(i, \sigma)$. for every agent i. Therefore, $\mathrm{tr}(\mathcal{M})$ satisfies the constraint of independence of agents.

In order to see that the 'no choice between undivided histories' constraint is satisfied suppose that the moment σ is both on h_1 and on h_2, i.e. $\sigma \in h_1 \cap h_2$, and suppose that $\sigma_0 < \sigma$, i.e. $\sigma = \sigma_0 \cdot \sigma'$ for some $\sigma' \neq \mathrm{nil}$. Due to the latter there must be a group action $\alpha \in \mathsf{GAct}$ such that $\sigma = \sigma_0 \cdot \alpha \cdot \sigma''$ for some $\sigma'' \in \mathsf{GAct}^*$. Therefore $\sigma_0 \cdot \alpha \in h_1 \cap h_2$. It then follows from the definition of $\mathcal{C}$ that for every agent i, both h_1 and h_2 belong to the choice cell of i in $\mathcal{C}(i, \sigma_0)$ that is defined by α_i (which is i's part of α). In other words, $(h_1, h_2) \in \mathcal{C}(i, \sigma_0)$. ■

Let $\mathcal{M} = \langle \mathcal{R}, \mathcal{S}, \mathcal{V}\rangle$ be a DL-PC model. We recursively define the history associated to its successor function $\mathcal{S}$ as follows:

$$\begin{aligned} h_{\mathcal{M}}^{\leq 0} &= \{\mathrm{nil}\} \\ h_{\mathcal{M}}^{\leq n+1} &= h_{\mathcal{M}}^{\leq n} \cup \{\mathcal{S}(\sigma) : \sigma \in h_{\mathcal{M}}^{\leq n}\} \\ h_{\mathcal{M}} &= \bigcup_{n \in \mathbb{N}_0} h_{\mathcal{M}}^{\leq n} \end{aligned}$$

The set $h_{\mathcal{M}}$ is a history from $\mathcal{H}_{\mathrm{nil}}$. Observe that $\mathrm{succ}(\mathrm{nil}, h_{\mathcal{M}}) = \mathcal{S}(\mathrm{nil})$.

Proposition 18 *Let* $\mathcal{M} = \langle \mathcal{R}, \mathcal{S}, \mathcal{V}\rangle$ *be a* DL-PC *model. Then for every* $\mathcal{L}_{\mathrm{Stit,X}}$ *formula* φ *we have*

$$\mathcal{M} \models \varphi \textit{ if and only if } \mathrm{tr}(\mathcal{M}), \mathrm{nil}/h_{\mathcal{M}} \models \varphi$$

Proof We prove by induction on the structure of φ that for every model $\mathcal{M}$ and for every sequence σ we have

$$\mathcal{M}^\sigma \models \varphi \text{ if and only if } \mathrm{tr}(\mathcal{M}^\sigma), \mathrm{nil}/h_{\mathcal{M}^\sigma} \models \varphi$$

where $\mathcal{M}^\sigma$ is defined recursively as expected:

$$\begin{aligned} \mathcal{M}^{\mathrm{nil}} &= \mathcal{M} \\ \mathcal{M}^{\alpha \cdot \sigma} &= (\mathcal{M}^\alpha)^\sigma \end{aligned}$$

Observe that $\mathrm{succ}(\sigma, h_{\mathcal{M}^\sigma}) = \sigma \cdot \mathcal{S}^\sigma(\mathrm{nil})$.

The only interesting cases are the operators Stit_G and the operator X. For the 'next' operator we have:

$$\begin{aligned}
\mathcal{M}^\sigma \models \mathrm{X}\psi \text{ iff } & (\mathcal{M}^\sigma)^{\mathcal{S}^\sigma(\text{nil})} \models \psi \\
\text{iff } & \mathcal{M}^{\sigma \cdot \mathcal{S}^\sigma(\text{nil})} \models \psi \\
\text{iff } & \text{tr}(\mathcal{M}^{\sigma \cdot \mathcal{S}^\sigma(\text{nil})}), \text{nil}/h_{\mathcal{M}^{\sigma \cdot \mathcal{S}^\sigma(\text{nil})}} \models \psi && \text{(by I.H.)} \\
\text{iff } & \text{tr}(\mathcal{M}^\sigma), \mathcal{S}^\sigma(\text{nil})/h_{\mathcal{M}^\sigma} \models \psi && (*) \\
\text{iff } & \text{tr}(\mathcal{M}^\sigma), \text{succ}(\text{nil}, h_{\mathcal{M}^\sigma})/h_{\mathcal{M}^\sigma} \models \psi \\
\text{iff } & \text{tr}(\mathcal{M}^\sigma), \text{nil}/h_{\mathcal{M}^\sigma} \models \mathrm{X}\psi
\end{aligned}$$

The step $(*)$ is correct because for every model $\mathcal{M}$ and formula ψ, $\text{tr}(\mathcal{M}), \mathcal{S}(\text{nil})/h_{\mathcal{M}} \models \psi$ if and only if $\text{tr}(\mathcal{M}^\alpha), \text{nil}/h_{\mathcal{M}^\alpha} \models \psi$.

For the agency operator we have:

$$\begin{aligned}
& \mathcal{M}^\sigma \models \text{Stit}_G\psi \\
\text{iff } & (\mathcal{M}^\sigma)' \models \psi \text{ for every } (\mathcal{M}^\sigma)' \text{ such that } (\mathcal{M}^\sigma)' \sim_G \mathcal{M}^\sigma \\
\text{iff } & \text{tr}((\mathcal{M}^\sigma)'), \text{nil}/h_{(\mathcal{M}^\sigma)'} \models \psi \text{ for every } (\mathcal{M}^\sigma)' \text{ such that } (\mathcal{M}^\sigma)' \sim_G \mathcal{M}^\sigma \\
& \text{(by I.H.)} \\
\text{iff } & \text{tr}(\mathcal{M}^\sigma), \text{nil}/h_{(\mathcal{M}^\sigma)'} \models \psi \text{ for every } (\mathcal{M}^\sigma)' \text{ such that } (\mathcal{M}^\sigma)' \sim_G \mathcal{M}^\sigma \\
& \text{(Prop.16)} \\
\text{iff } & \text{tr}(\mathcal{M}^\sigma), \text{nil}/h' \models \psi \text{ for every } h' \text{ such that } (h_{\mathcal{M}^\sigma}, h') \in \mathcal{C}^\sigma(G, \text{nil}) \\
& (**) \\
\text{iff } & \text{tr}(\mathcal{M}^\sigma), \text{nil}/h_{\mathcal{M}^\sigma} \models \text{Stit}_G\psi
\end{aligned}$$

The step $(**)$ is correct because there is a history h' such that $(h_{\mathcal{M}^\sigma}, h') \in \mathcal{C}^\sigma(G, \text{nil})$ if and only if there is a successor function $(\mathcal{S}^\sigma)'$ such that for every sequence of group actions σ_1 we have $(\mathcal{S}^\sigma)'(\sigma_1) \subseteq \mathcal{R}$ and $((\mathcal{S}^\sigma)'(\sigma_1))_G = ((\mathcal{S}^\sigma)'(\sigma_1))_G$.

Corollary 19 For every formula $\varphi \in \mathcal{L}_{\text{Stit,X}}$, if φ is valid in the discrete Chellas stit logic then φ is valid in DL-PC.

We note that there exist $\mathcal{L}_{\text{Stit,X}}$ formulas that are DL-PC valid but invalid in the Chellas stit logic. An example is $\text{Stit}_i(p \vee q) \to (\text{Stit}_i p \vee \text{Stit}_i q)$. Among the $\mathcal{L}_{\text{Stit,X}}$ formulas, those that are valid in BT+AC are therefore a strict subset of those that are valid in DL-PC models. We leave it as an open question whether there is a set of schematic validities distinguishing DL-PC from discrete BT $+$ AC models.

6 Conclusion

We have introduced a Dynamic Logic of Propositional Control DL-PC having a stit operator. We have axiomatised DL-PC and have shown that the problem of satisfiability in models of propositional control is decidable. Our result is interesting because we know that in the set of BT+AC models, satisfiability of 'pure stit' formulas (formulas from $\mathcal{L}_{\mathrm{Stit,X}}$ without X) is already undecidable [21]. This makes DL-PC an interesting alternative to stit logics.

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. Their integration remains to be done.

Our logic is related to Segerberg's logic of bringing it about [22]. There, an operator μ is introduced whose argument is a formula. The expression $\mu\varphi$ denotes an action leading to states where φ holds, and the formula $[\mu\varphi]\psi$ reads "after an agent brings it about that φ it is the case that ψ". In Segerberg's logic the recursive structure of actions can be easily captured. For example, Jack's action of killing Joe by shooting him can be described by the formula $[\mu \mathit{JoeShot}]\mathit{JoeDead}$. The interesting aspect of Segerberg's logic—distinguishing it from other logics of agency such as the logic of seeing-to-it-that or the logic of bringing-it-about-that— is that it provides a clear separation between the result of the action and the means for achieving the result. This perspective is similar in spirit both to our logic, which also includes in the object language action labels making reference to the means leading to the result of the (individual or group) action: in the case of a single agent, our group actions α may be viewed as the bringing about of a conjunctions of literals. For example the group action $\{(i, +p), (i, -q)\}$ may be identified with $\mu(p \wedge \neg q)$.

7 Perspectives: Bringing Them All Together

The title of the present chapter is inspired from Krister Segerberg's chapter "Two traditions in the logic of belief: bringing them together" [3]. The aim of that work was to reconcile two different logical approaches to belief: epistemic logics à la Hintikka [23] and belief revision theory à la Alchourrón, Gärdenfors and Makinson [24]. His strategy was to couch the latter in the former by extending epistemic logic with modal operators from dynamic logic, where the programs of the latter are nothing but operations of belief revision. Obviously, a continuation of the present chapter would be to bring together DL-PC and Segerberg's approach. We leave this to future work.

Acknowledgments The first and third author acknowledge the support of the EU coordinated action SINTELNET. The fourth author acknowledges the support of the program Marie Curie People Action Trentino (project LASTS).

References

1. Segerberg, K. (1992). Getting started: Beginnings in the logic of action. *Studia Logica*, *51*, 347–378.
2. Segerberg, K. (2000) Outline of a logic of action. In F. Wolter, H. Wansing, M. de Rijke, & M. Zakharyaschev (Eds.), *Advances in modal logic* (pp. 365–387). New Jersey: World Scientific.
3. Segerberg, K. (1999). Two traditions in the logic of belief: Bringing them together. In H. Jürgen Ohlbach, & U. Reyle (Eds.), *Logic, language and reasoning: Essays in honour of Dov Gabbay, volume 5 of Trends in Logic* (pp. 135–147). Dordrecht: Kluwer Academic Publishers.
4. Pörn, I. (1977). *Action theory and social science: Some formal models. Synthese library 120*. D. Reidel: Dordrecht.
5. Elgesem, D. (1993). Action theory and modal logic (Ph.D. thesis, Institut for filosofi, Det historiskfilosofiske fakultetet, Universitetet i Oslo, 1993).
6. Elgesem, D. (1997). The modal logic of agency. *Nordic Journal of Philosophy and Logic*, *2*(2), 1–46.
7. Governatori, Guido, & Rotolo, Antonino. (2005). On the axiomatization of elgesem logic of agency and ability. *Journal of Philosophical Logic*, *34*, 403–431.
8. Horty, John, & Belnap, Nuel. (1995). The deliberative stit: A study of action, omission, ability and obligation. *Journal of Philosophical Logic*, *24*(6), 583–644.
9. Horty, J. F. (2001). *Agency and deontic logic*. Oxford: Oxford University Press.
10. Belnap, N., Perloff, M., & Xu, M. (2001). *Facing the future: Agents and choices in our indeterminist world*. Oxford: Oxford University Press.
11. Thomason, R. H. (2012). Krister Segerberg's philosophy of action. *In this volume*.
12. McCarthy, J., & Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer, & D. Mitchie (Eds.), *Machine intelligence* (Vol. 4, pp. 463–502). Edinburgh: Edinburgh University Press.
13. Reiter, Raymond. (1991). The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz (Ed.), *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy* (pp. 359–380). San Diego: Academic Press.
14. Reiter, R. (2001). *Knowledge in action: Logical foundations for specifying and implementing dynamical systems*. Cambridge: The MIT Press.
15. van Ditmarsch, Hans, Herzig, Andreas, & de Lima, Tiago. (2011). From situation calculus to dynamic logic. *Journal of Logic and Computation*, *21*(2), 179–204.
16. van Eijck, Jan. (2000). Making things happen. *Studia Logica*, *66*(1), 41–58.
17. Hans P., van Ditmarsch, Wiebe van der Hoek, & Barteld, P. Kooi. (2005). Dynamic epistemic logic with assignment. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, & M. Wooldridge (Eds.), *Proceedings of the 4th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)* (25–29 July 2005 , pp. 141–148). Utrecht, The Netherlands: ACM.
18. Blackburn, Patrick, de Rijke, Maarten, & Venema, Yde. (2001). *Modal logic. Cambridge tracts in theoretical computer science*. Cambridge: University Press.
19. Broersen, Jan, Herzig, Andreas, & Troquard, Nicolas. (2006). Embedding alternating-time temporal logic in strategic STIT logic of agency. *Journal of Logic and Computation*, *16*(5), 559–578.
20. Herzig, Andreas, & Lorini, Emiliano. (2010). A dynamic logic of agency I: STIT, abilities and powers. *Journal of Logic, Language and Information*, *19*(1), 89–121.
21. Herzig, A., & Schwarzentruber, F. (2008). Properties of logics of individual and group agency. In C. Areces, & R. Goldblatt (Eds.), *Advances in modal logic (AiML)* (pp. 133–149), Nancy: College Publications.
22. Segerberg, Krister. (1989). Bringing it about. *Journal of Philosophical Logic*, *18*(4), 327–347.
23. Hintikka, Jaakko. (1962). *Knowledge and belief*. Ithaca: Cornell University Press.
24. Alchourrón, Carlos, Gärdenfors, Peter, & Makinson, David. (1985). On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, *50*, 510–530.