On the Dynamics of Delegation, Cooperation, and Control: A Logical Account

Wiebe van der Hoek and Michael Wooldridge
Department of Computer Science
The University of Liverpool
L69 3BX UK
wiebe, mjw@csc.liv.ac.uk

ABSTRACT

We present DCL-PC: a dynamic logic of delegation and cooperation. The logical foundation of DCL-PC is CL-PC, a logic for reasoning about cooperation in which the powers of agents and coalitions of agents stem from a distribution of atomic Boolean variables to individual agents – the choices available to coalitions in CL-PC correspond to the possible truth assignments to the propositions they control. The basic modal constructs of CL-PC are of the form "coalition C can cooperate to bring about φ". DCL-PC extends CL-PC with dynamic logic modalities in which atomic programs are of the form "agent i gives proposition p to agent j". By combining these dynamic delegation operators with cooperation modalities, it is possible to reason about delegation and how it affects the power structure within a society. We give two alternative semantics for the logic, (a "direct" semantics, in which we directly represent the distributions of atomic propositions to agents, and a more conventional Kripke semantics), and prove that these semantics are equivalent. We then present a sound and complete axiomatization, and investigate the computational complexity of the model checking and satisfiability problems for DCL-PC.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Coherence and coordination, Multiagent Systems; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Modal Logic*

General Terms

Verification, Design

Keywords

Propositional control, cooperation, delegation, dynamics, modal logic, dynamic logic, powers of agents and coalitions

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

1. INTRODUCTION

In recent years, there has been a flurry of activity in the development of logics for reasoning about the strategic abilities of agents and coalitions of agents in game-like multi-agent systems. Pauly's Coalition Logic [7] and the Alternating-time Temporal Logic (ATL) of Alur, Henzinger, and Kupferman [1] are perhaps the best-known examples of such work. Although they differ on details, the basic construct in both logics is the *cooperation modality*, a construct which is expressed in ATL as $\langle\langle C \rangle\rangle \varphi$, with the intended meaning that C can cooperate in such a way as to ensure that φ becomes true. However, the *origin* of an agent's powers – where these powers *derive from* – is rarely discussed in the cooperation logic literature: powers are simply defined in the models that underpin the logic.

One exception is the MOCHA system for model checking coalitional power properties, in which powers are specified by defining, for every variable in a system, a unique agent that *controls* this variable [2]. Control, in this sense, means the unique ability to choose a value for this variable. Motivated by this observation, van der Hoek and Wooldridge developed CL-PC, a cooperation logic in which powers are specified by allocating every Boolean variable to a unique agent in the system: the choices (and hence powers) available to a coalition then correspond to the possible assignments of truth or falsity that may be made to the variables under their control [11]. van der Hoek and Wooldridge gave a complete axiomatization of CL-PC, and showed that the model checking and satisfiability problems for the logic are both PSPACE-complete. However, one failing of CL-PC is that the power structures underpinning the logic – that is, the allocation of variables to agents – is *fixed*. Hence, ultimately, coalitional powers remain static in CL-PC.

In this paper we study a variant of CL-PC which allows us to reason about *dynamic* power structures. The logic DCL-PC extends CL-PC with dynamic logic operators [4], in which atomic actions are of the form $a_1 \rightsquigarrow_p a_2$, which is read as "agent a_1 gives variable p to agent a_2". The pre-condition of such an action is that variable p is in agent a_1's allocation of variables, and executing the program has the effect of *transferring control of variable p from agent a_1 to agent a_2*. Thus the dynamic component of DCL-PC is concerned with *delegating control* in systems, and by using the logic, we can reason about how the abilities of agents are affected by the transfer of control of variables in the system. Note that, as in conventional dynamic logic, atomic programs may be combined in DCL-PC with the usual sequential composition

(";"), non-deterministic choice ("∪"), test ("?"), and iteration (" ") operations, to make complex delegation programs. For example, the following DCL-PC formula asserts that, if agent i gives either p or q to j, then j will be able to achieve φ.

$$[(i \rightsquigarrow_p j) \cup (i \rightsquigarrow_q j)]\Diamond_j \varphi$$

In the remainder of the paper, following an introduction to the logic, we prove three main results with respect to DCL-PC. First, we give two alternative semantics for the logic: a "direct" semantics, in which models directly represent the allocation of propositions to the agents that control them, and a more conventional Kripke semantics, and we prove that these two semantics are equivalent. Second, we give an axiomatization of DCL-PC, and show that this axiomatization is complete (with respect to both semantics). Finally, we show that, for fragments of DCL-PC in which delegation programs are "well-behaved" (in the sense of Halpern and Reif's SDPDL [3]), the satisfiability and model checking problems for DCL-PC are both PSPACE-complete, and hence no worse than the corresponding problems for CL-PC. We conclude with some comments on related work and conclusions.

2. THE LOGIC DCL-PC

The language of DCL-PC extends classical propositional logic with *cooperation modalities*, of the form $\Diamond_C \varphi$. These modalities are used to express *contingent ability* [11]: $\Diamond_C \varphi$ means that, under the assumption that the world remains otherwise unchanged, the set of agents C have the ability to achieve φ. As shown in [11], (and as defined below), stronger ability operators, roughly corresponding to Pauly's Coalition Logic cooperation modality [7] may be derived from these: we express the fact that C have a choice such that, no matter what the agents outside C do, φ will become true, as $\langle\!\langle C \rangle\!\rangle_\alpha \varphi$ (the "α" is for "α-effectivity" [7, p.20]).

In DCL-PC, these operators may be combined with *dynamic delegation modalities*, of the form $[\delta]\varphi$, which means "after the delegation program δ is executed, φ will hold" (cf. dynamic logic [4]). Delegation programs express the transfer of control between agents, and are built from an atomic set of delegation expressions of the form $i \rightsquigarrow_p j$, meaning "agent i gives proposition p to agent j". Such a program can be executed iff agent i owns this proposition, and the effect is to transfer ownership to j. These atomic programs may then be combined with the usual program constructs (;, ?, ∪,) of regular dynamic logic [4], and of course, conventional program constructs such as **while** and **if** may be straightforwardly defined in terms of these.

Notice that executing such a program – and hence changing the distribution of ownership of atomic propositions in the structure – changes the abilities of agents and coalitions. Thus we can reason about the effect that such dynamic power allocation programs have on the abilities of agents and coalitions. As an example, the following formula asserts that it is possible for agent i to give away its propositions to agent j, non-deterministically choosing one proposition at a time, until agent j has the ability to achieve φ.

$$\langle \textbf{while } \neg\Diamond_j \varphi \textbf{ do } \bigcup_{p \in \mathbb{P}_i} i \rightsquigarrow_p j \rangle \top$$

2.1 Syntax

Figure 1 defines the syntax of DCL-PC. Thus we use $\top$ as a logical constant for truth, "¬" for negation, and "∨" for disjunction. As usual, we define the remaining connectives of classical propositional logic as abbreviations: $\bot \mathrel{\hat{=}} \neg\top$, implication is defined by $\varphi \rightarrow \psi \mathrel{\hat{=}} \neg\varphi \vee \psi$, conjunction is $\varphi \wedge \psi) \mathrel{\hat{=}} \neg(\varphi \rightarrow \neg\psi)$ and implication is defined as $\varphi \leftrightarrow \psi \mathrel{\hat{=}} (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$. Finally, exclusive or $\triangledown$ is defined by $(\varphi \triangledown \psi) \mathrel{\hat{=}} (\varphi \vee \psi) \wedge \neg(\varphi \wedge \psi)$. With respect to delegation programs, **while** and **if** constructs are defined as follows [4, p.167]:

$$\begin{aligned} \textbf{if } \varphi \textbf{ then } \delta_1 \textbf{ else } \delta_2 &\mathrel{\hat{=}} ((\varphi?;\delta_1) \cup (\neg\varphi?;\delta_2)) \\ \textbf{while } \varphi \textbf{ do } \delta &\mathrel{\hat{=}} ((\varphi?;\delta)*;\neg\varphi?) \end{aligned}$$

Where there is no possibility of confusion, we will omit set brackets for cooperation modalities, for example writing $\Diamond_{1,2}$ rather than $\Diamond_{\{1,2\}}$. A DCL-PC formula containing no modalities is said to be an *objective* formula.

Let $\mathbb{P}(\varphi)$ denote the set of propositional variables occurring in DCL-PC formula φ, and let $\mathbb{A}(\varphi)$ denote the set of all agents named in φ.

2.2 Direct Semantics

We now introduce the first of the two semantics for DCL-PC. Given a (fixed, finite, non-empty) set $\mathbb{A}$ of agents, and a (fixed, finite, non-empty) set $\mathbb{P}$ of propositional atoms, we say an *allocation* of $\mathbb{P}$ to $\mathbb{A}$ is an indexed tuple $\xi = \langle \mathbb{P}_1, \ldots, \mathbb{P}_n \rangle$ where there is an indexed element $\mathbb{P}_i$ for each $i \in \mathbb{A}$, such that $\mathbb{P}_1, \ldots, \mathbb{P}_n$ forms a partition of $\mathbb{P}$. The intended interpretation of an allocation $\xi = \langle \mathbb{P}_1, \ldots, \mathbb{P}_n \rangle$ is that $\mathbb{P}_i \subseteq \mathbb{P}$ is the set of propositional variables under agent i's control. That is, agent i has freedom to allocate whatever Boolean values it sees fit to the members of $\mathbb{P}_i$.

Now, we say a *model* for DCL-PC is a structure:

$$\mathcal{M} = \langle \mathbb{A}, \mathbb{P}, \xi_0, \theta \rangle$$

where:

- $\mathbb{A} = \{1, \ldots, n\}$ is finite, non-empty a set of *agents*;
- $\mathbb{P} = \{p, q, \ldots\}$ is finite, non-empty set of *propositional variables*;
- $\xi_0 = \langle \mathbb{P}_1 \ldots, \mathbb{P}_n \rangle$ is the *initial allocation* of $\mathbb{P}$ to $\mathbb{A}$, with the intended interpretation that $\mathbb{P}_i$ is the subset of $\mathbb{P}$ representing those variables initially under the control of agent $i \in \mathbb{A}$; and finally,
- $\theta : \mathbb{P} \rightarrow \{\mathtt{tt}, \mathtt{ff}\}$ is a propositional valuation function, which determines the initial truth value of every propositional variable.

Some additional notation is convenient in what follows. We define the *size* of a model $\mathcal{M} = \langle \mathbb{A}, \mathbb{P}, \xi_0, \theta \rangle$ to be $|\mathbb{A}|+|\mathbb{P}|$; we denote the size of $\mathcal{M}$ by *size*($\mathcal{M}$). A *coalition*, C is simply a subset of $\mathbb{A}$, i.e., $C \subseteq \mathbb{A}$. For any such $C \subseteq \mathbb{A}$ we denote the *complement* of C, (i.e., $\mathbb{A} \setminus C$) by $\overline{C}$. We will write $\mathbb{P}_C$ for $\bigcup_{i \in C} \mathbb{P}_i$. For two valuations θ and θ', and a set of propositions $\Psi \subseteq \mathbb{P}$, we write $\theta = \theta' \pmod{\Psi}$ if θ and θ' differ at most in the atoms in Ψ, and we then say that θ and θ' are the *same modulo* Ψ. We will sometimes understand the model $\mathcal{M} = \langle \mathcal{F}, \theta \rangle$ to consist of a valuation θ on top of a

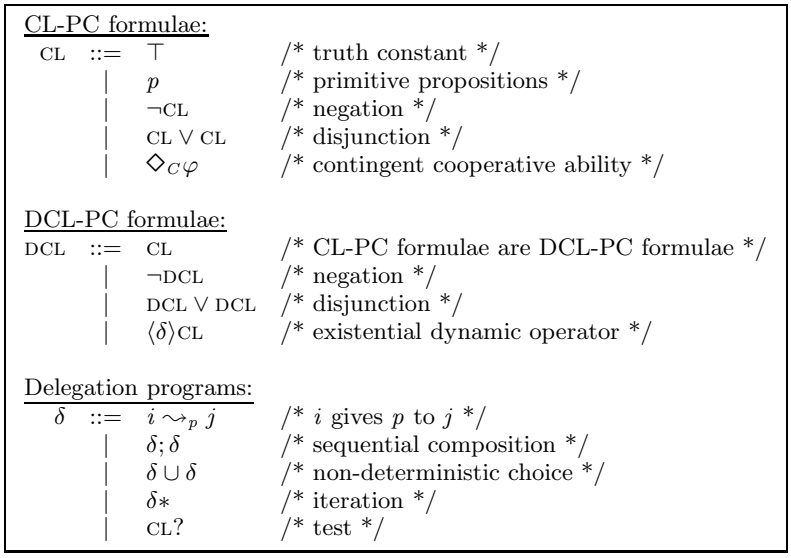

Figure 1: Syntax of DCL-PC: $p \in \mathbb{P}$ is a primitive proposition, $C \subseteq \mathbb{A}$ is a set of agents, and $i, j \in \mathbb{A}$ are agents.

frame $\mathcal{F} = \langle \mathbb{A}, \mathbb{P}, \xi_0 \rangle$. Given a model $\mathcal{M} = \langle \mathbb{A}, \mathbb{P}, \xi_0, \theta \rangle$ and a coalition C in $\mathcal{M}$, a *C-valuation* is a function:

$$\theta_C : \mathbb{P}_C \to \{\mathtt{tt}, \mathtt{ff}\}.$$

Thus a C-valuation is a function that assigns truth values to just the primitive propositions controlled by the members of the coalition C. If $\mathcal{M} = \langle \mathbb{A}, \mathbb{P}, \xi_0, \theta \rangle$ is a model, C is a coalition in $\mathcal{M}$, and θ_C is a C-valuation, then by $\mathcal{M} \oplus \theta_C$ we mean the model $\langle \mathbb{A}, \mathbb{P}, \mathbb{P}_1, \ldots, \mathbb{P}_n, \theta' \rangle$, where θ' is the function defined as follows

$$\theta'(p) \mathrel{\hat{=}} \begin{cases} \theta_C(p) & \text{if } p \in \mathbb{P}_C \\ \theta(p) & \text{otherwise} \end{cases}$$

and all other element of the model are as in $\mathcal{M}$. Thus $\mathcal{M} \oplus \theta_C$ denotes the model that is identical to $\mathcal{M}$ except that the values assigned by its valuation function to propositions controlled by members of C are as determined by θ_C. Obviously, we will use these 'hops' between valuations to interpret the $\Diamond_C$-modalities. We also need a mechanism to interpret the basic delegation construct $i \rightsquigarrow_p j$ and arbitrary programs δ: this is given in the next section.

2.3 Delegation Program Relations

Readers familiar with dynamic logic will have been expecting this section: to give a modal semantics to the dynamic logic constructs of DCL-PC, we must define, for every delegation program δ a binary relation R_δ over models such that $(\mathcal{M}_1, \mathcal{M}_2) \in R_\delta$ iff $\mathcal{M}_2$ is a model that may result from one possible execution of program δ. We start by defining the relation $R_{i \rightsquigarrow_p j}$, for primitive delegation programs of the form $i \rightsquigarrow_p j$, i.e., agent i gives control of proposition p to agent j. Let $\mathcal{M} = \langle \mathbb{A}, \mathbb{P}, \xi_0, \theta \rangle$ and $\mathcal{M}' = \langle \mathbb{A}', \mathbb{P}', \xi_0', \theta' \rangle$, with $\xi_0 = \langle \mathbb{P}_1, \ldots, \mathbb{P}_n \rangle$ and $\xi_0' = \langle \mathbb{P}_1', \ldots, \mathbb{P}_n' \rangle$. Then $(\mathcal{M}, \mathcal{M}') \in R_{i \rightsquigarrow_p j}$ iff either $i = j$, $p \in \mathbb{P}_i$ and $\mathcal{M} = \mathcal{M}'$, or else:

1. $p \in \mathbb{P}_i$ (agent i controls p to begin with)
2. $\mathbb{P}_i' = \mathbb{P}_i \setminus \{p\}$ (agent i no longer controls p afterwards);
3. $\mathbb{P}_j' = \mathbb{P}_j \cup \{p\}$ (agent j controls p afterwards); and
4. all other components of $\mathcal{M}'$ are as in $\mathcal{M}$.

For the remaining constructs of delegation programs, we define the program relations inductively, in terms of the relations for primitive delegation programs, as given above. Let the composition of relations R_1 and R_2 be denoted $R_1 \circ R_2$, and let the reflexive transitive closure (ancestral) of relation R be denoted R^*. Then the accessibility relations for complex programs are defined as follows [4, p.168] ($\models^{\mathsf{d}}$ will be defined shortly):

$$\begin{array}{rcl} R_{\delta_1;\delta_2} & = & R_{\delta_1} \circ R_{\delta_2} \\ R_{\delta_1 \cup \delta_2} & = & R_{\delta_1} \cup R_{\delta_2} \\ R_{\delta^*} & = & (R_\delta)^* \\ R_{\varphi?} & = & \{(\mathcal{M}, \mathcal{M}) \mid \mathcal{M} \models^{\mathsf{d}} \varphi\} \end{array}$$

2.4 Truth Conditions

We interpret formulae of DCL-PC with respect to models, as introduced above. Given a model $\mathcal{M} = \langle \mathbb{A}, \mathbb{P}, \xi_0, \theta \rangle$ and a formula φ, we write $\mathcal{M} \models^{\mathsf{d}} \varphi$ to mean that φ is satisfied (or, equivalently, true) in $\mathcal{M}$, under the "direct" semantics. The rules defining the satisfaction relation $\models^{\mathsf{d}}$ are as follows:

$\mathcal{M} \models^{\mathsf{d}} \top$

$\mathcal{M} \models^{\mathsf{d}} p$ iff $\theta(p) = \mathtt{tt}$ (where $p \in \mathbb{P}$);

$\mathcal{M} \models^{\mathsf{d}} \neg\varphi$ iff $\mathcal{M} \not\models^{\mathsf{d}} \varphi$;

$\mathcal{M} \models^{\mathsf{d}} \varphi \vee \psi$ iff $\mathcal{M} \models^{\mathsf{d}} \varphi$ or $\mathcal{M} \models^{\mathsf{d}} \psi$;

$\mathcal{M} \models^{\mathsf{d}} \Diamond_C \varphi$ iff there exists a C-valuation θ_C such that $\mathcal{M} \oplus \theta_C \models^{\mathsf{d}} \varphi$.

$\mathcal{M} \models^{\mathsf{d}} \langle\delta\rangle\varphi$ iff there exists a model $\mathcal{M}'$ such that $(\mathcal{M}, \mathcal{M}') \in R_\delta$ and $\mathcal{M}' \models^{\mathsf{d}} \varphi$.

We assume the conventional definitions of satisfiability and validity: a DCL-PC formula φ is $^{\mathsf{d}}$-*satisfiable* iff there exists a

DCL-PC model $\mathcal{M}$ such that $\mathcal{M} \models^{\mathsf{d}} \varphi$, and φ is $^{\mathsf{d}}$-*valid* iff for every DCL-PC model $\mathcal{M}$ we have $\mathcal{M} \models^{\mathsf{d}} \varphi$. We write $\models^{\mathsf{d}} \varphi$ to indicate that φ is $^{\mathsf{d}}$-valid.

Let us define the natural box dual "$\Box$..." of the $\Diamond$... cooperation modality:

$$\Box_C \varphi \mathrel{\hat{=}} \neg \Diamond_C \neg \varphi.$$

Where C is a coalition and φ is a formula of DCL-PC, we write $controls(C, \varphi)$ to mean that C can choose φ to be either true or false:

$$controls(C, \varphi) \mathrel{\hat{=}} \Diamond_C \varphi \wedge \Diamond_C \neg \varphi$$

By using the $controls(\ldots)$ construct, we can capture the distribution of propositions among agents in a model.

LEMMA 2.1 ([11]). *Let $\mathcal{M} = \langle \mathbb{A}, \mathbb{P}, \xi_0, \theta \rangle$ be a model for* DCL-PC*, $i \in \mathbb{A}$ be an agent, and $p \in \mathbb{P}$ be a propositional variable in $\mathcal{M}$. Then $\mathcal{M} \models^{\mathsf{d}} controls(i, p)$ iff $p \in \mathbb{P}_i$.*

Although we will not use them in what follows, we note that the basic cooperation operators of ATL [1] and Pauly's Coalition Logic [7] can be defined as follows (where $\bar{C}$ is the complement of C):

$$\langle\!\langle C \rangle\!\rangle_\alpha \varphi \mathrel{\hat{=}} \Diamond_C \Box_{\bar{C}} \varphi$$

Thus $\langle\!\langle C \rangle\!\rangle_\alpha \varphi$ means that C have a choice such that if they make this choice, then no matter what the agents outside C do, φ will be true. We refer the reader to [11] for details.

2.5 A Kripke Semantics

For some purposes, it is more natural to conceive of the semantics for DCL-PC as a multi-modal one. Taking a valuation of $\mathbb{P}$ as a world, there are basically two, "orthogonal" accessibility relations (cf. Figure 2 where, in the model M, $p \in \mathbb{P}_i$) between them: first of all, we have a "horizontal" relation for agent i between two worlds w and u if agent agent i is able, given the valuation in w, to turn it into the valuation as described by u, just by choosing appropriate values for his atoms. In other words, $R_i wu$ iff $w = u \pmod{\mathbb{P}_i}$. Such a relation does not affect the partition ξ. Let us therefore define our Kripke models to be $M = \langle \Theta, R_{i \in \mathbb{A}}, \xi \rangle$ where Θ is the set of all valuations θ over $\mathbb{P}$. We denote the set of all such Kripke models by $\mathcal{K}(\mathbb{A}, \mathbb{P})$.

Secondly, there is a 'vertical' relation between pointed models (M, θ) and (M', θ') (with $M = \langle \Theta, R_{i \in \mathbb{A}}, \xi \rangle, M' = \langle \Theta, R_{i \in \mathbb{A}}, \xi' \rangle \in \mathcal{K}(\mathbb{A}, \mathbb{P})$), which indicates a change of the partition ξ to ξ'. Since these do not affect the valuation, we have for such pairs that $\theta = \theta'$. Slightly abusing notation, we define $(M, \theta)(i \rightsquigarrow_p j)(M', \theta')$ exactly when either $i = j$, $p \in \mathbb{P}_i$ and $M = M'$, or else $p \in \mathbb{P}_i$, $\mathbb{P}'_i = \mathbb{P}_i \setminus \{p\}$ and $\mathbb{P}'_j = \mathbb{P}_j \cup \{p\}$, and all the other sets $\mathbb{P}_h$ remain the same.

The truth relation $\models^{\mathsf{k}}$ interpreting formulae over Kripke structures holds between pairs of the form M, θ and formulae φ. It's definition is obvious: for $\Diamond_i$- formulas, we stay in M, and for $\langle i \rightsquigarrow_p j \rangle$- formulas, we hop to another M', θ (see also Figure 2). The following lemma is then easily established by induction on φ:

LEMMA 2.2. *For any fixed sets of agents $\mathbb{A}$ and atoms $\mathbb{P}$, the direct semantics and the modal semantics are equivalent,*

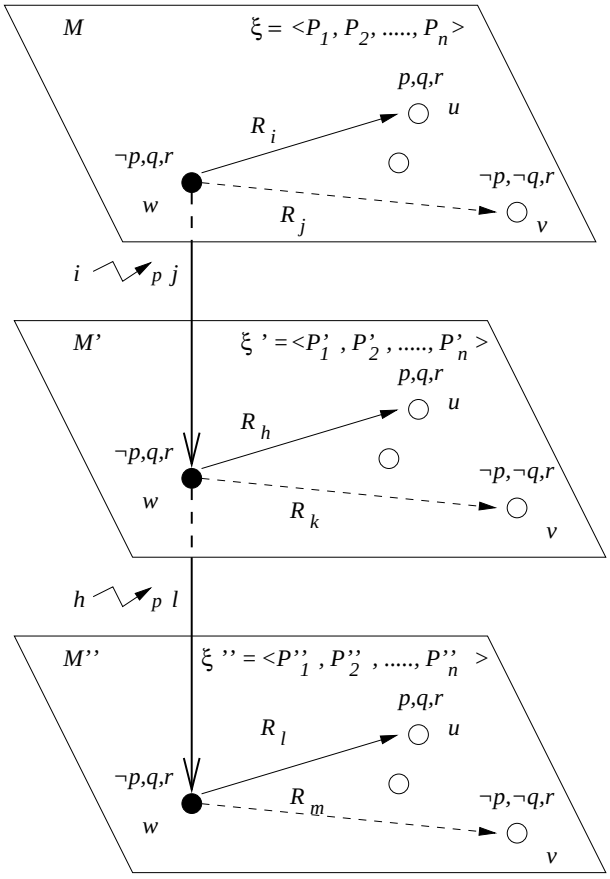

Figure 2: $\mathcal{K}(\mathbb{A}, \mathbb{P})$ for DCL-PC.

i.e., for any φ and any $M \in \mathcal{K}(\mathbb{A})(\mathbb{P})$ with $M = \langle \Theta, R_{i \in \mathbb{A}}, \xi \rangle$ and model $\mathcal{M} = \langle \mathbb{A}, \mathbb{P}, \xi, \theta \rangle$:

$$\mathcal{M} \models^{\mathsf{d}} \varphi \text{ iff } M, \theta \models^{\mathsf{k}} \varphi$$

2.6 Some Observations

It is important to note that $\Diamond_i \varphi$ is read as "agent i has the power, by assigning values to his variables, to make φ true". If i controls p, he also "has the power" to ensure for instance $controls(j, p)$, but this is expressed through the delegation modality: $\langle i \rightsquigarrow_p j \rangle controls(j, p)$ We will in the next section see that these types of "control" are indeed rather orthogonal. For instance, $\langle i \rightsquigarrow_p j \rangle \Diamond_j \varphi$ (i can give p to j, who then can achieve φ) and $\Diamond_{i,j} \varphi$ (i and j can cooperate, to achieve φ) are logically incomparable. For instance, taking $\varphi = \langle j \rightsquigarrow_p i \rangle \top$ gives $\models^{\mathsf{d}} controls(i, p) \rightarrow (\langle i \rightsquigarrow_p j \rangle \varphi \wedge \neg \Diamond_{i,j} \varphi)$, while for $\varphi = \langle i \rightsquigarrow_p j \rangle \top$ we have $\models^{\mathsf{d}} controls(i, p) \rightarrow (\neg \langle i \rightsquigarrow_p j \rangle \varphi \wedge \Diamond_{i,j} \varphi)$. However, if the goal is an objective formula, we can relate atomic control and delegation, as we will shortly see.

Consider the delegation program

$$\alpha_i = (\bigcup_{p \in \mathbb{P}} controls(i, p)?; \bigcup_{j \in \mathbb{A}} i \rightsquigarrow_p j)$$

Then $\langle \alpha_i \rangle \varphi$ would express that i has a way to give one of his atoms to one of the agents (possibly himself) in such a way, that consequently φ holds. Thus, $\langle \alpha_i \rangle \varphi$ means that i can distribute his variables among the agents in such a

way, that afterwards φ holds. So, when reasoning about i's power, the 'strongest' that he can achieve is any φ for which $\Diamond_i\varphi \vee \langle\alpha_i\rangle\varphi$ expressing that i can achieve φ by either choosing an appropriate value for his atoms, or by distributing his atoms over $\mathbb{A}$ in an appropriate way.

The program α can be generalised to incorporate coalitions that can give away atoms, and those that can send: let

$$\alpha_{C\rightsquigarrow D} = \bigcup_{c\in C}\bigcup_{p\in\mathbb{P}_c} \mathit{controls}(c,p)?;\ \bigcup_{d\in D\cup\{c\}} \langle c \rightsquigarrow_p d\rangle$$

This program $\alpha_{C\rightsquigarrow D}$ lets an arbitrary agent c from C either give one of his arbitrary atoms p to an arbitrary member of D, or do nothing (give it to himself). Now, for objective formulas φ, we have the following, where i is a dedicated agent from C: $\Diamond_C\varphi \leftrightarrow \langle\alpha_{C\rightsquigarrow\{i\}}\rangle\Diamond_i\varphi$. In words: a coalition can choose values for their atoms such that φ, if and only if they have a way to give all their atoms to the dedicated agent i, who then can achieve φ. Note that we are in general not able to eliminate all occurrences of $\Diamond$'s, since this is the only way to reason about a "different valuation".

3. A COMPLETE AXIOMATIZATION

A complete axiomatisation for the logic DCL-PC is given in Figure 3. The Dynamic Component is an immediate adaptation of Propositional Dynamic Logic (see [4]). The Control Axioms are inherited from [11]. Note that *partition* only specifies that we have *a* partition, while in contrast, for the fixed partition ξ that was assumed in [11], one could explicitly state that $\mathit{controls}(i,p)$, for every $p \in \mathbb{P}_i$.

For the Delegation & Control Axioms, *atomic permanence* states that no program δ changes the valuation. From this, one easily extends this to arbitrary objective formulas (obtaining *objective permanence*, see Lemma 3.1). The axiom $\mathit{persistence}_1(\mathit{control})$ says that i's control over p is not affected when we move to another valuation, and axiom $\mathit{persistence}_2(\mathit{control})$ specifies how i remains in control over p, even when a delegation program is executed: either the atom passed in that program is not p, or the delegating agent is not i. The axiom $\mathit{precondition}(\mathit{delegation})$ expresses that agents can only give atoms away that they possess, and, finally *func* says that the transition relation associated with an atomic delegation program is functional: at most one resulting state emerges.

LEMMA 3.1. *The schemes of Figure 4 are derivable in DCL-PC. (The occurrence of $\ell(p)$ refers to a literal in p: it is either p or $\neg p$, with the obvious meaning for $\neg\ell(p)$.) Moreover, from [11] we know that the axioms $K(i), T(i), B(i)$ and $\mathit{effect}(i)$ have coalitional counterparts $K(C), T(C), B(C)$ and $\mathit{effect}(C)$ that are derivable for any coalition C.*

We now proceed to prove that the system DCL-PC of Figure 3 is complete. First, some notation. Given the set of atoms $\mathbb{P}$, we denote conjunctions of literals (p or $\neg p$) over them by $\pi, \pi_1, \pi_2 \ldots$, and disjunctions of such $\pi's$ are denoted by $\sigma, \sigma_1, \ldots$. It is well known that every objective formula ψ has an equivalent Disjunctive Normal Form $\mathit{DNF}(\psi)$. For instance, if $\mathbb{P} = \{p,q,r\}$, then $\mathit{DNF}(p \wedge q) = (p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r)$.

Consider the language without programs, in which we only have propositional logic and abilities $\Diamond_C$. Models for these

at-least(*control*) : $(\ell(p) \wedge \mathit{controls}(i,p)) \rightarrow \Diamond_i\neg\ell(p)$
at-most(*control*) : $\ell(p) \rightarrow (\Diamond_i\neg\ell(p) \rightarrow \Box_j\ell(p))$ $\quad (i \neq j)$
non-effect(i) : $(\Diamond_i\ell(p) \wedge \neg\mathit{controls}(i,p)) \rightarrow \Box_i\ell(p)$
inverse : $\mathit{controls}(i,p) \rightarrow (\varphi \leftrightarrow [i \rightsquigarrow_p j; j \rightsquigarrow_p i]\varphi)$
reverse : $([i \rightsquigarrow_p j][k \rightsquigarrow_q h]\varphi) \leftrightarrow ([k \rightsquigarrow_q h][i \rightsquigarrow_p j]\varphi)$ where $(j \neq k$ and $h \neq i)$ or $p \neq q$
persistence(*non − control*) : $(\neg\mathit{controls}(i,p) \leftrightarrow \Box_j\neg\mathit{controls}(i,p))$
objectivepermanence : $\langle\delta\rangle\top \rightarrow (\varphi \leftrightarrow [\delta]\varphi)$ where φ is objective.

Figure 4: Theorems of DCL-PC.

are M, $M' \in \mathcal{K}(\mathbb{A})(\mathbb{P})$ In [11] it was shown that in that program-free language, every formula φ is equivalent to one without any occurrences of coalition operators. For instance, suppose that $p, q \in \mathbb{P}_i$. Then a formula $\Diamond_i(\neg p \wedge r)$ is equivalent to $(p \wedge r) \vee (\neg p \wedge r)$ (we "read off" the current value of atom r outside i's control).

We now establish a similar result for the full language. Any state (M,θ) is completely characterised when we know which atoms are true in it, and what the allocation of atoms to agents is. In such a case, the truth of all objective formulas, formulas involving abilities and delegation programs is completely determined.

Hence, apart from formulas σ in DNF, we also need disjunctions ω over conjunctions ξ, corresponding to partitions of $\mathbb{P}$. To be more precise, for any partition ξ of $\mathbb{P}$ in $\langle\mathbb{P}_1,\ldots,\mathbb{P}_n\rangle$, we will also write ξ for the conjunction $\bigwedge_{i\in\mathbb{A}}\bigwedge_{p_i\in\mathbb{P}_i} p_i$ exactly enumerating this partition. For any set of constraints Γ of the form $\mathit{controls}(i,p)$ and $\neg\mathit{controls}(i,p)$, let ω_Γ be the disjunction of conjunctions ξ that are compatible with Γ. For instance, if $\mathbb{P} = \{p,q,r\}$ and Γ enforces that $p \in \mathbb{P}_1$ and $q \notin \mathbb{P}_2$ ($\mathbb{A} = \{1,2\}$), then $\omega_\Gamma = (\mathit{controls}(1,p) \wedge \mathit{controls}(1,q) \wedge \mathit{controls}(1,r)) \vee (\mathit{controls}(1,p) \wedge \mathit{controls}(1,q) \wedge \mathit{controls}(2,r))$.

LEMMA 3.2. *Let φ be an arbitrary formula and ζ a conjunction of assertions of the form $(\neg)\mathit{controls}(i,p)$. Then, in DCL-PC, we have $\vdash \Diamond_C(\varphi \wedge \zeta) \leftrightarrow (\zeta \wedge \Diamond_C\varphi)$.*

THEOREM 3.1. *Every formula φ is DCL-PC-provably equivalent to either $\bot$ or else a formula of the form $\sigma \wedge \omega$, where σ is an objective formula in DNF, and ω is a disjunction of conjunctions of ξ's, each ξ corresponding to a partition ξ of $\mathbb{P}$.*

PROOF. The proof is an induction over φ. In the case of $\varphi = p$, the result is obvious: take for σ the disjunction of all conjunctions $p \wedge \pi$, where π runs over all possible conjunctions of literals over $\mathbb{P} \setminus \{p\}$. For ω we just can take the disjunction of all possible ξ.

Let us enumerate all possible (full) conjunctions of literals over $\mathbb{P}$ (with $|\mathbb{P}| = k$) with $W = \pi_1, \pi_2, \ldots, \pi_{2^k}$, and all

Propositional Component		
Prop	φ	where φ is any objective tautology
Dynamic Component		
$K(\delta)$	$[\delta](\varphi \to \psi) \to ([\delta]\varphi \to [\delta]\psi)$	
$union(\delta)$	$[\delta \cup \delta']\varphi \leftrightarrow ([\delta]\varphi \wedge [\delta']\varphi)$	
$comp(\delta)$	$[\delta;\delta']\varphi \leftrightarrow ([\delta][\delta']\varphi)$	
$test(\delta)$	$[\varphi?]\psi \leftrightarrow (\varphi \to \psi)$	
$mix(\delta)$	$(\varphi \wedge [\delta][\delta*]\varphi) \leftrightarrow ([\delta*]\varphi)$	
$ind(\delta)$	$(\varphi \wedge [\delta*](\varphi \to [\delta]\varphi)) \to ([\delta*]\varphi)$	
Control Axioms		
$K(i)$	$\Box_i(\varphi \to \psi) \to (\Box_i\varphi \to \Box_i\psi)$	
$T(i)$	$\Box_i\varphi \to \varphi$	
$B(i)$	$\varphi \to \Box_i\Diamond_i\varphi$	
empty	$\Box_\emptyset\varphi \leftrightarrow \varphi$	
control	$controls(i,p) \leftrightarrow (\Diamond_i p \wedge \Diamond_i \neg p)$	
partition	$\bigwedge_{p\in P}(controls(1,p) \triangledown \cdots \triangledown controls(n,p))$	
effect(i)	$(\psi \wedge \ell(p) \wedge controls(i,p)) \to \Diamond_i(\psi \wedge \neg\ell(p))$	where $\begin{cases} p \notin \mathbb{P}(\psi), \text{ and} \\ \psi \text{ is objective} \end{cases}$
Comp-$\cup$	$\Box_{C_1}\Box_{C_2}\varphi \leftrightarrow \Box_{C_1\cup C_2}\varphi$	
Delegation & Control Axioms		
atomic permanence	$\langle\delta\rangle\top \to (p \leftrightarrow [\delta]p)$	
persistence$_1$(*control*)	$(controls(i,p) \to \Box_j controls(i,p))$	
persistence$_2$(*control*)	$controls(i,p) \to [j \rightsquigarrow_q h]controls(i,p)$	where $i \neq j$ or $p \neq q$
precondition(*delegation*)	$\langle i \rightsquigarrow_p j\rangle\top \to controls(i,p)$	
delegation	$controls(i,p) \to \langle i \rightsquigarrow_p j\rangle controls(j,p)$	
func	$controls(i,p) \to (\langle i \rightsquigarrow_p j\rangle\varphi \leftrightarrow [i \rightsquigarrow_p j]\varphi)$	
Rules of Inference		
ModusPonens	$\vdash \varphi, \vdash (\varphi \to \psi) \Rightarrow \vdash \psi$	
Necessitation	$\vdash \varphi \Rightarrow \vdash \Box\varphi$	$\Box = [\delta], [i \rightsquigarrow_p j],$ or $\Box_i$

Figure 3: Axioms of DCL-PC.

possible partitions over $\mathbb{P}$ by $P = \xi_1, \xi_2, \ldots, \xi_m$, where m denotes the number of passions in n sets of k elements. Let us now assume the theorem to be proven for ψ, and let its equivalent be

$$\psi \equiv \sigma \wedge \omega = (\bigvee_{i\in I} \pi_i) \wedge (\bigvee_{j\in J} \xi_j)$$

where I is some index set $\subseteq \{1, 2, \ldots, 2^k\}$ (narrowing down the number of possible 'worlds'), as is $J \subseteq \{1, 2, \ldots, m\}$ (constraining the set of possible partitions). We now consider cases, focusing on those including the dynamic component, i.e., where $\varphi = [\delta]\psi$, with ψ of the mentioned form. We proceed now by induction over the program δ. The basic program yields $\varphi = \langle i \rightsquigarrow_p j\rangle\psi$, i.e., $\varphi \equiv \langle i \rightsquigarrow_p j\rangle(\sigma \wedge \omega)$. By the axiom for *precondition*(*delegation*) and *func*, this is equivalent to $controls(i,p) \wedge [i \rightsquigarrow_p j]\psi$. Given that $\psi \equiv (\sigma \wedge \omega)$ this gives $controls(i,p) \wedge [i \rightsquigarrow_p j]\sigma \wedge [i \rightsquigarrow_p j]\omega$. By *objective permanence*, the conjunct $[i \rightsquigarrow_p j]\sigma$ is equivalent to σ. The remainder is equivalent to $\langle i \rightsquigarrow_p j\rangle\omega$, which is $\langle i \rightsquigarrow_p j\rangle \bigvee_{j\in J} \delta_j$. The latter is equivalent to $\bigvee_{j\in J}\langle i \rightsquigarrow_p j\rangle\delta_j$. Consider an arbitrary $\langle i \rightsquigarrow_p j\rangle\delta_j$. Again, this is equivalent to $controls(i,p) \wedge [i \rightsquigarrow_p j]\delta_j$. Recall that δ_j is a conjunction of basic assertions of the form $\pm controls(h,q)$. Thus, $[i \rightsquigarrow_p j]\delta_j$ is equivalent to a conjunction of $[i \rightsquigarrow_p j] \pm controls(h,q)$. Hence, $controls(i,p) \wedge [i \rightsquigarrow_p j]\delta_j$ is equivalent to a conjunction of formulas of the form $\langle i \rightsquigarrow_p j\rangle \pm controls(h,q)$. We argue how those are reduced:

1. $(q = p, h = j)$
   $\langle i \rightsquigarrow_p j\rangle controls(j,p)$ reduces to $controls(i,p)$ (use *delegation*) and $\langle i \rightsquigarrow_p j\rangle\neg controls(j,p)$ reduces to $\bot$ (use *delegation*, *func* and *partition*).

2. $(q = p, h \neq j)$
   $\langle i \rightsquigarrow_p j\rangle\neg controls(h,p)$ reduces to $controls(i,p)$ (use

delegation and *partition*) and $\langle i \rightsquigarrow_p j \rangle controls(h, p)$ reduces to $\bot$ (use *delegation* and *partition*).

3. $(p \neq q)$
$\langle i \rightsquigarrow_p j \rangle controls(h, q)$ reduces to $controls(h, q)$ '$\leftarrow$' follows from $persistence_2(control)$ and *func*, '$\rightarrow$' follows from *delegation* and *partition*. Also $\langle i \rightsquigarrow_p j \rangle \neg controls(h, q)$ reduces to $\neg controls(h, q)$ ('$\leftarrow$' follows from *func* and the other equivalence proven in this item, and '$\rightarrow$' follows from *partition* and *delegation*)

For programs δ composed by union, sequential composition and test, we can immediately apply the induction hypothesis and use the axioms given in Figure 3, in the Dynamic Component. The case for is more complex, and we omit it due to space restrictions: the key observation is that we can essentially eliminate the star operator by noting that a delegation program can only be applied a finite number of times before revisiting a previous configuration.

□

We can use Theorem 3.1 directly to generate a canonical model for a consistent formula ψ, as follows. First, calculate its equivalent $\sigma \wedge \omega = (\bigvee_{i \in I} \pi_i) \wedge (\bigvee_{j \in J} \xi_j)$. Include this in a maximal consistent set Γ. We then get both $\bigvee_{i \in I} \pi_i \in \Gamma$ and $\bigvee_{j \in J} \xi_j \in \Gamma$. Since Γ is maximal consistent, and the fact that both the π_i's and the ξ_j's exclude each other, we find exactly one i and j such that $\pi_i \in \Gamma$ and $\xi_j \in \Gamma$. But then we can immediately 'read off' in which world M, θ we are in the canonical model: the world (which is a valuation) θ is dictated by π_i, and the partition ξ is completely determined by the formula ξ_j. All in all, we have

THEOREM 3.2. *The system DCL-PC is sound and complete with respect to both the modal and the direct semantics.*

4. EXPRESSIVITY AND COMPLEXITY

We know from the results of van der Hoek and Wooldridge (see [11]) that the model checking and satisfiability problems for CL-PC are PSPACE-complete, and since DCL-PC subsumes CL-PC, this implies a PSPACE-hardness lower bound on the corresponding problems for DCL-PC. The obvious question is then whether the additional expressive power of DCL-PC arising from the dynamic constructs implies a more complex decision problem. We now show that, for fragments of DCL-PC in which delegation programs are "well behaved", the model checking and satisfiability problems are no worse: they are both PSPACE-complete. (When we consider the model checking problem in this section, we consider the problem with respect to *direct* models, not Kripke models. Of course, with respect to satisfiability, it makes no difference: a formula is satisfiable with respect to direct models iff it is satisfiable wrt Kripke models.)

The *$*$-free* fragment of DCL-PC is the fragment in which delegation programs are constrained to contain no $*$ operators (but may contain other complex program constructs). The *deterministic* fragment of DCL-PC is the fragment in which the only choice operator permitted is **if**...**then**...**else**..., and the only loop construct permitted is **while**...**do**... (cf. [3]). Thus, in deterministic DCL-PC, we are not permitted to use arbitrary non-deterministic choice ($\cup$) or iteration ($*$) constructs: we can only use choice and iteration in their well-behaved **if** and **while** forms. Delegation programs in deterministic DCL-PC are *strictly deterministic*: given the same initial configuration, such a program will always behave in *exactly* the same way. With this observation in mind, we can prove the following.

```
1.  function eval(φ, M = ⟨𝔸, ℙ, ξ0, θ⟩) returns tt or ff
2.      if φ ∈ ℙ then
3.          return θ(φ)
4.      elsif φ = ¬ψ then
5.          return not eval(ψ, M)
6.      elsif φ = ψ1 ∨ ψ2 then
7.          return eval(ψ1, ⟨𝔸, ℙ, ξ0, θ⟩)
8.              or eval(ψ2, ⟨𝔸, ℙ, ξ0, θ⟩)
9.      elsif φ = ◇C ψ then
10.         for each C-valuation θC
11.             if eval(ψ, ⟨𝔸, ℙ, ξ0, θ⟩ ⊕ θC) then return tt
12.         end-for
13.         return ff
14.     elsif φ = ⟨δ⟩ψ then
15.         for each model M' over 𝔸, ℙ
16.             if (M, M') ∈ Rδ then
17.                 if eval(φ, M') then return tt
18.         end-for
19.         return ff
20. end-function
```

Figure 5: A model checking algorithm for dcl-pc.

THEOREM 4.1. *The model checking problems for deterministic* DCL-PC *and $*$-free* DCL-PC *(with respect to direct models) are both* PSPACE*-complete.*

PROOF. *Given that both of these fragments subsume* CL-PC*, we only need to prove the upper bound. Consider the function $eval(\cdots)$ in Figure 5. Soundness is obvious by construction. First note that the algorithm is strictly analytic: recursion is always on a sub-formula of the input. That the algorithm is in* PSPACE *follows from the fact that the loops at lines 10–12 and 15-18 involve, in the first case simply binary counting with the variables $\mathbb{P}_C$, and in the second simply looping through all direct models over $\mathbb{A}$ and $\mathbb{P}$: we do not need to record these models once they are checked, and so this can be done in polynomial space. It only remains to justify that the check $(\mathcal{M}, \mathcal{M}') \in R_\delta$ on line 16 can be done in polynomial space. The only non-obvious case is where δ is a loop, i.e., of the form* **while** φ ***do*** δ'*. To see that this relation can be checked in polynomial space, recall that the program must be deterministic, and that there are only exponentially many models over $\mathbb{A}$ and $\mathbb{P}$. So,* **while** φ ***do*** δ' *can only be executed exponentially times before either terminating or re-entering a state that has previously been reached – in which case, since programs are deterministic,* it must be looping. *So, we repeatedly execute the statement, keeping track (by counting) of how many times it has been executed. If, after $O(2^{p(size(\mathcal{M}))})$ executions (for polynomial $p(\cdots)$), the statement has not terminated, then it must be looping, and hence we can assert $(\mathcal{M}, \mathcal{M}') \notin R_\delta$. (We can count to $2^{p(n)}$ using only $O(p(n) + 1)$ bits.) If the program terminates, we simply need to check that the model which results is $\mathcal{M}'$. The proof for the $*$-free case is similar.* □

The proof of the following is identical to the equivalent result of [11].

LEMMA 4.1. *A* DCL-PC *formula φ is satisfiable, iff it is satisfied in a (direct) model $\mathcal{M}$ such that $|\mathbb{A}| = |\mathbb{A}(\varphi)| + 1$ and $\mathbb{P} = \mathbb{P}(\varphi)$.*

The 'additional agent' is needed in e.g., $\neg controls(a, p)$: if agent a does not control p, someone else must. (Having noticed this, it may seem a bigger surprise that only one additional agent is *sufficient*, for every formula.). Note that a consequence of Lemma 4.1 is that a DCL-PC formula φ is satisfiable iff it is satisfied in a model $\mathcal{M}$ such that $size(\mathcal{M}) = |\mathbb{P}(\varphi)| + |Ag(\varphi)| + 1$.

This gives us:

THEOREM 4.2. *The satisfiability checking problems for deterministic* DCL-PC *and $*$-free* DCL-PC *are* PSPACE*-complete.*

PROOF. *Given a formula φ, loop through each model $\mathcal{M}$ containing $\mathbb{P}(\varphi)$ and $\mathbb{A}(\varphi)$ such that $size(\mathcal{M}) = |\mathbb{P}(\varphi)| + |\mathbb{A}(\varphi)| + 1$, and if $\mathcal{M} \models^{\mathsf{d}} \varphi$ then return "yes". If we have considered all such models, return "no". By Theorem 4.1, we can check whether $\mathcal{M} \models^{\mathsf{d}} \varphi$ in polynomial space.* □

5. CONCLUSIONS

In this paper, we have built upon a logic of strategic cooperative ability, in which the control that agents have over their environment is represented by assigning them specific propositional variables, for which the agents that "owns them" can determine their truth value. We added a dynamic component to this logic, thus obtaining an expressive language in which one can reason about what agents (and coalitions of agents) can achieve by setting their assigned variables, or by giving the control of them to others. We gave two related semantics for this language, and provided a complete axiomatisation for them.

The key property that establishes the proof of completeness is the fact that every formula in the language is provably equivalent to a disjunction of conjunctions of literals over atoms p and assertions of the form $controls(i, p)$. We also investigated the complexity of the model checking and satisfiability problems for the logic, and showed that, for "well behaved" delegation programs, these problems were no worse than for the program-free fragment. Although other researchers have developed formal systems for reasoning about delegation (e.g., [6, 5]), to the best of our knowledge DCL-PC is the first such system to have a rigorous semantics, and a complete axiomatization.

The possible extensions of this work are multiple. For instance, we have assumed that agents are *omniscient*: they know 'where they are' and which atoms are under control of which other agents. This assumption can be easily relaxed in a similar way as was done in [10]: it would be interesting to see however what would be 'natural' ways to impose such knowledge (through an accessibility relation) in our Kripke semantics of Section 2.5.

Secondly, we have treated all states in our Kripke semantics as 'equal', i.e., we only considered what agents are *able* to achieve if they pass or have control over certain means that can change the state of the world. But we have not considered *why* certain agents might try to bring about certain states. In [8] we add utilities to ATL-like structures, and there is no reason we could not have them in DCL-PC, so that one could express, for instance, that, regardless whether agent i 'owns' atom p or not, he prefers states in which p is true. Finally, one might, apart from addressing agents' individual strategies, single out certain states that are more *desirable* or *socially preferable* for the whole community. We hope we can apply work that we have recently developped (see [9]) on *knowledge and social laws* in a general setting of ATL to the context of DCL-PC.

6. REFERENCES

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, Sept. 2002.

[2] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Taşiran. Mocha: Modularity in model checking. In *CAV 1998: Tenth International Conference on Computer-aided Verification, (LNCS Volume 1427)*, pages 521–525. Springer-Verlag: Berlin, Germany, 1998.

[3] J. Y. Halpern and J. H. Reif. The propositional dynamic logic of deterministic, well-structured programs. *Theoretical Computer Science*, 27:127–165, 1983.

[4] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. The MIT Press: Cambridge, MA, 2000.

[5] N. Li, B. N. Grosof, and J. Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security*, 6(1):128 – 171, 2003.

[6] T. J. Norman and C. Reed. Group delegation and responsibility. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, pages 491–498, Bologna, Italy, 2002.

[7] M. Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[8] W. van der Hoek, W. Jamroga, and M. Wooldridge. A logic for strategic reasoning, 2005. See elsewhere in these proceedings of *AAMAS05*.

[9] W. van der Hoek, M. Roberts, and M. Wooldridge. Knowledge and social laws, 2005. See elsewhere in these proceedings of *AAMAS05*.

[10] W. van der Hoek and M. Wooldridge. Time, knowledge, and cooperation: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[11] W. van der Hoek and M. Wooldridge. On the logic of co-operation and propositional control. *Artificial Intelligence*, 64(1-2):81–119, 2005.