Alternating-time Dynamic Logic

Nicolas Troquard
Department of Computer Science
University of Liverpool
Liverpool L69 3BX, UK
nico@liverpool.ac.uk

Dirk Walther
Department of Artificial Intelligence
Faculty of Informatics
Technical University of Madrid (UPM)
E-28660 Boadilla del Monte, Madrid, Spain
dirk.walther@upm.es

ABSTRACT

We propose Alternating-time Dynamic Logic (ADL) as a multi-agent variant of Dynamic Logic in which atomic programs are replaced by coalitions. In ADL, the Dynamic Logic operators are parametrised with regular expressions over coalitions and tests. Such regular expressions describe the dynamic structure of a coalition. This means that, when moving from Dynamic Logic to ADL, the focus shifts away from describing what is executed and when, toward describing who is acting and when. While Dynamic Logic provides for reasoning about complex programs, ADL facilitates reasoning about coalitions with an inner dynamic structure, so-called coordinated coalitions. The semantics for such coalitions involves partial strategies and a variety of ways to combine them. Different combinations of partial strategies give rise to different semantics for ADL. In this paper, we mainly focus on one version of the semantics but we provide a discussion on other semantic variants of ADL together with possible syntactic extensions. We see ADL to be suitable for the specification and the verification of scheduling and planning systems, and we therefore present a model checking algorithm for ADL and investigate its computational complexity.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*modal logic*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems*

General Terms

Theory

Keywords

Logic for MASs, Dynamic Logic, Partial Strategies, Model Checking

1. INTRODUCTION

Following Alternating-time Temporal Logic (ATL) [4], logics for strategic abilities have received much attention in multi-agent systems. Their purpose is to provide formal methods for reasoning about dynamic systems of interacting and autonomous agents.

Cite as: Alternating-time Dynamic Logic, Nicolas Troquard and Dirk Walther, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

However, the notions of cooperation and of abilities are often idealised. First, in general, strategies are taken as total: agents are allowed to plan their actions for every situation they could encounter. Second, it is possible to reason about the abilities of a coalition, but one has no way to describe how the agents in the coalition coordinate.

We investigate abilities of what we name *coordinated coalitions*. A coordinated coalition describes a set of agents that act as is specified by a schedule. The strategies of a scheduled coalition are naturally partial strategies that are defined whenever a member of the coalition is asked to act by the schedule.

The logic.

In this paper, we introduce Alternating-time Dynamic Logic (ADL). It brings new perspectives in logics of agents at the frontier between dynamic logics (Propositional Dynamic Logic PDL) and alternating-time logics (Coalition Logic CL [7], ATL).

First, ADL can be seen as a variant of Dynamic Logic replacing atomic programs with coalitions which yields an abstraction away from particular actions toward acting entities. While in Dynamic Logic the complex programs are regular expressions over atomic programs, in ADL the Dynamic Logic operators are parametrised with regular expressions over coalitions. The latter can be seen as representing dynamic coalitions in the sense that a regular expression over coalitions describes how the initial coalition changes over time. For instance, we understand the sequence $A;B;C$ of three coalitions as describing how a coalition changes over time (the construct ; is the sequential operator): the initial coalition is A that is changing to B after one time step which in turn is changing to C after the second time step.

Second, ADL can be seen as a variant of Coalition Logic with coordinated coalitions. Within a coordinated coalition, the members act according to a schedule which defines the inner dynamics of the organisation of the coalition. The ADL-formula $\langle A;B;C\rangle\varphi$ means that property φ can be achieved when the coalitions A, B and C are acting in turns in this order, and this, no matter what $(B \cup C) \setminus A$ do at the first step, whatever $(A \cup C) \setminus B$ do at the second step and whatever $(A \cup B) \setminus C$ do at the third step.

Like in PDL, these patterns of behaviour can be more sophisticated than simple sequences. They can specify an indeterminate coalition to act using the construct '$\sqcup$'. We can also specify iterations of acting coalitions using '$*$'. Finally, the test operator '?' permits to test the truth value of a formula.

It allows to obtain more complex schedules by combining the different constructs. The formula

$$\langle (A;B)^*\rangle\varphi$$

states that "the coalition $A \cup B$ has a strategy to achieve φ consisting in A and B acting alternatively for an indeterminate time".

We can also characterise conditional schedules:

$$\langle (\psi?;A) \sqcup (\neg\psi?;B)\rangle\varphi$$

"the coalition $A \cup B$ can ensure φ by lending the responsibility to act to A if ψ is true and by letting the responsibility to B otherwise".

Remark 1.1. *It is important to make the difference between the schedule $A \cup B$, that is equivalent to the atomic schedule containing the agents of A and B, and $A \sqcup B$, that is the compound non-deterministic schedule.*

Talking about partial strategies.

To model superposing actions of agents, ADL-formulae are evaluated over alternating transition systems (ATS), which are the models for ATL. We will use ATSs *with endpoints*, where a maximal computation can be finite.

Unlike in ATL, though, strategies in ADL are *partial*. Consider again the formula $\langle A;B;C\rangle\varphi$. At the moment of evaluation of the formula, we will be looking for partial strategies in which the responsibility of acting to obtain φ is incumbent upon the coalition A at first, upon the coalition B next and upon C to finish. We then can see the union of A, B and C becomes a coordinated coalition. They have to act together in a concerted way to achieve φ, by following a pattern of behaviour that is specified by the schedule $A;B;C$.

It is important to note that though we use regular expressions like in PDL, our logic is not purported to reason explicitly about actions or strategies. The strategies at play are not themself represented in the object language of ADL. A regular expression over coalitions can be interpreted as a collection of *partial* strategies sharing a particular *structure*. It enables the reasoning about structures of partial strategies.

In the logic ATL, the formula $\langle\langle A\rangle\rangle\Phi$ means that the agents in A have a (total) strategy to ensure the temporal property Φ. One approach to the reasoning about partial strategies within an ATL-like framework, is to explicitly specify the size of the partial strategies we are considering. In the logic Alternating-time Temporal Logic with Bounded Memory (ATLBM) [2], the formula $\langle\langle A\rangle\rangle^n\Phi$ is intended to capture that A have a partial strategy – specifying what they do in no more than n states – to ensure Φ.

We are in need of a logic where we can distinguish the efficiency of partial strategies that may differ in their inner structure. Here, we propose ADL as one possible approach to the study of partial strategies, by providing complex modalities describing the dynamics of a coalition. For instance, the formula $\langle(\{a,b\} \sqcup \{c\});\{b\}\rangle\Phi$ will trigger partial strategies defined in at most two states. At the first state, the strategies will be defined only for agents a and b, or only for agent c. At the second state, the strategy will be specified only for agent b. Unlike in ATLBM, in ADL we do not only reason about the size of a partial strategy, but also about its inner dynamic structure.

Outline.

In Section 2, we define the logic ADL. We pay particular attention to introducing the notions that are used to describe schedules and partial strategies. In Section 3, we consider some examples to provide more intuitions about the non-trivial interplay of the schedules and the partial strategies. We propose an algorithm to model check ADL-formulae against ATSs with endpoints in Section 4 and show that the model checking problem for ADL is PTime-complete. In Section 5, we describe an application to cooperative game theory capitalising on the notion of complexity of coordinated coalitions. We discuss possible variants of ADL in Section 6 and we conclude in Section 7.

2. ALTERNATING-TIME DYNAMIC LOGIC

In this section we describe the logic ADL. We start with introducing the syntax.

Definition 2.1. *[*ADL *Syntax] Let* $\mathbf{\Pi}$ *be a countably infinite set of atomic propositions and* $\mathbf{\Sigma}$ *a countably infinite set of agents. A* coalition *is a finite set* $C \subset \mathbf{\Sigma}$ *of agents.* ADL*-formulae* φ *and schedules* R *are simultaneously defined with the following grammar, where* p *ranges over atomic propositions and* A *ranges over coalitions of agents:*

$$\begin{array}{lcllllllllll}\varphi & ::= & p & \mid & \neg\varphi & \mid & \varphi\vee\varphi & \mid & \langle R\rangle\varphi & & & \\ R & ::= & \epsilon & \mid & A & \mid & R\sqcup R & \mid & R;R & \mid & R^* & \mid \ \varphi?\end{array}$$

Let $\mathfrak{R}$ *be the set of all schedules.* ◁

Let $R \in \mathfrak{R}$ be a schedule. For convenience, R^k with $k > 0$ denotes the sequence $\underbrace{R;\ldots;R}_{k}$ and R^0 denotes the empty schedule ϵ.

To transition smoothly to the semantics of ADL, we provide some intuitions about the language and some aspects of the truth values of the operators.

The empty schedule ϵ will be interpreted as the strategy with an empty domain. That is, it does not determine any choice to be taken by the agents. Therefore, applying the empty schedule does not yield any transition in the state transition system. We will have that $\langle\epsilon\rangle\varphi \equiv \varphi$.

An atomic schedule test $\varphi?$ that is successful at a state q (i.e., if φ is true in q) will be interpreted as the empty schedule. That is, we assume that performing a test does not involve any transition in the transition system $\mathcal{S}$ (upon which the semantics will be based). The failure of $\varphi?$ at q (i.e., if φ does not hold in q), however, will be interpreted as the absence of any strategy at q: we will note $\Upsilon_{\mathcal{S}}(\varphi?, q) = \emptyset$.

An atomic schedule coalition A will give rise to a set of strategies $\Upsilon_{\mathcal{S}}(A, q)$ at a state q in a transition system $\mathcal{S}$. Each such strategy σ determines a choice for every agent in A and *only for them*. Moreover, such strategies will be defined at q and *only at* q. A choice will be modelled as a set of states. By taking a particular choice, an agent can enforce the successor state q' of the entire system to lie in its choice. Moreover, an agent can prevent the system to go into q' by taking a choice that does not contain q'.

Two schedules R_1 and R_2 can be executed sequentially using the operator ';'. The schedule $R_1; R_2$ has the obvious intended meaning that first a strategy for R_1 and then strategy for R_2 is executed. The semantics of $R_1; R_2$ requires to carefully *combine* the strategies resulting from R_1 with the ones resulting from R_2. We will *complete* the partial strategies resulting from R_1 at q, with the ones resulting from R_2 at the states reached from q (*sequential completion*). The resulting strategies will be collected in the set $\Upsilon_{\mathcal{S}}(R_1; R_2, q)$.

Remark 2.2. *The combination of partial strategies can be done in more than one way. In this paper, we will define this operation (Def. 2.4) such that the combined strategy does not cause any infinite computation. This way, we make sure that the length of a strategy mirrors the length of a schedule, and a schedule can only give rise to strategies that terminate. We will comment more on this issue in Section 6.*

Finally, the compound schedule composed of the constructs '*' and '$\sqcup$' should not pose a problem of interpretation.

Before we formally define the models of our logic, we introduce some abstract notions that are going to be useful to talk about alternating transition systems and partial strategies within.

Let Q be a set of states in a system. A *choice* Q' is a non-empty subset of Q. Intuitively, a choice is a selection of states that are candidates of system successor states. Let Σ be the set of all agents in the system. State transitions are determined by a transition function. A (Q,Σ)-*transition function* is a function $\delta : Q \times \Sigma \to 2^{2^Q}$ that maps a state q and an agent a to the set $\delta(q,a)$ of choices that are available to a at q (according to δ). A state q is called a δ-*endpoint* if there is an agent without a choice at q (i.e. $\delta(q,a) = \emptyset$ for some a).

The notion of a choice is generalised to a group of agents (*collective choice*) using a choice function. A *choice function for Q and Σ* is a partial function $\kappa : \Sigma \to 2^Q \setminus \{\emptyset\}$ mapping an agent a to a choice $\kappa(a)$. A choice function κ is called *total* if it is defined for all agents in Σ (i.e. $\mathsf{dom}(\kappa) = \Sigma$). That is, a total choice function represents a collective choice of the grand coalition Σ. Clearly, if $\kappa(a)$ is undefined (i.e. $a \notin \mathsf{dom}(\kappa)$), then κ does not specify any choice for a.

A *strategy* σ is a (partial) mapping from a state q to a choice function $\sigma(q)$. Intuitively, σ prescribes the collective choice $\sigma(q)$ for the agents in $\mathsf{dom}(\sigma(q))$ at q, where $\sigma(q)(a)$ is the choice made by agent a. We say that σ is *undefined at q* if $q \notin \mathsf{dom}(\sigma)$. Given a (Q,Σ)-transition function δ, a strategy σ is a δ-strategy if

(i) if $q \in \mathsf{dom}(\sigma)$ and $a \in \mathsf{dom}(\sigma(q))$, then $\sigma(q)(a) \in \delta(q,a)$; and
(ii) if q is a δ-endpoint, then $\sigma(q)(a)$ is undefined for all $a \in \Sigma$.

A δ-strategy σ is called *complete* if it determines total choice functions exactly at all non-δ-endpoints. We are now ready to introduce the models of ADL.

Formulae of ADL are evaluated in ATSs: A system is modelled as a set of states Q together with a set Σ of agents that populate the system, and a (Q,Σ)-transition function. The latter accounts for superposing actions of agents. Additionally, each state is labelled with propositions representing facts that hold at this state.

Notice, however, that in this paper, we consider structures that allow for endpoints. That is, these are not the same structures that are used for Alternating-time Temporal Logic, which requires the system to be never-ending, i.e., at every state there must be an outgoing transition to a successor state.

Definition 2.3. *[Alternating Transition Systems with endpoints] Let $\Sigma = \{a_1, \ldots, a_n\} \subseteq \mathbf{\Sigma}$ with $n \geq 1$ be a finite non-empty set of agents. An ATS for Σ is a tuple $\mathcal{S} = \langle \Pi, Q, \pi, \delta \rangle$ where*

- $\Pi \subseteq \mathbf{\Pi}$ *is a finite, non-empty set of* atomic propositions,
- Q *is a finite, non-empty set of* states,
- $\pi : Q \to \mathbf{2}^{\Pi}$ *is a* valuation function *which assigns to every state a set of propositions which are true there, and*
- δ *is a (Q,Σ)-transition function such that for every complete δ-strategy σ, it holds that for all states $q \in Q$ that are not δ-endpoints, there is a state $q' \in Q$ such that*

$$\bigcap_{a \in \Sigma} \sigma(q)(a) = \{q'\}.$$

$\lhd$

Notice that the condition on the transition function states that the entire system is deterministic: at each non-terminal state, the collective choice of the grand coalition (one choice for each agent), overlap in a singleton set containing the unique successor state of the entire system.

It seems to suggest itself to consider endpoints in ATSs for ADL, due to its similarity to PDL where the structures also allow points without successors. In the context of Coalition Logic, endpoints are a special case of the notion of terminal states in weak playability structures as discussed in [6].

Given a schedule, we are now going to define what partial strategies are triggered and what is their effectivity within ATSs.

We assume a (Q,Σ)-transition function δ. Given a δ-strategy σ, a state q' is a σ-*successor* of a state q if there is a complete δ-strategy σ' that contains σ such that $\{q'\} = \bigcap_{a\in\Sigma} \sigma'(q)(a)$. Note that σ', as a complete strategy, prescribes a total choice function at q (it specifies a choice for each agent). Intuitively, q' is a σ-successor of q if the collective choice of the grand coalition can force the system to go into q' at q whilst respecting the choices prescribed in σ. Clearly, q' being a σ-successor of q implies that q is not a δ-endpoint.

A state q' is a δ-*successor* of q if q' is a σ-successor of q for some δ-strategy σ. Let $\Delta_{\delta,q}$ be the set of all δ-successors of q.

When q is not a δ-endpoint, a strategy σ is a (δ, q, A)-*strategy* if

- σ is defined exactly at q, i.e. $\mathsf{dom}(\sigma) = \{q\}$; and
- if $A \neq \emptyset$ then $\mathsf{dom}(\sigma(q)) = A$ and $\sigma(q)(a) \in \delta(q,a)$;
- if $A = \emptyset$, the choice function $\sigma(q)$ is defined as $\sigma(q)(a) = \Delta_{\delta,q}$, for all $a \in \Sigma$.

The case $A = \emptyset$ needs some clarification. We expect to have a system transition resulting of the choices of some agents whenever a coalition is given to act in a schedule.

This includes the empty coalition. However, we have no agent that represent the empty coalition. Thus, we simulate the choice of the empty coalition by making every agent in Σ choose the vacuous choice $\Delta_{\delta,q}$ selecting the set of δ-successors of q. It can be noted that whenever $A \neq \emptyset$, a (δ, q, A)-strategy is a δ-strategy.

The set $\mathsf{out}(q, \sigma)$ of *outcomes* of a strategy σ starting at a state q is the set of all states $q' \in Q$ for which there is a finite path $q_0 q_1 \cdots q_m$ with $m \geq 0$ such that $q_0 = q$ and $q_m = q'$, and one of the following conditions holds:

- q_{i+1} is a σ-successor of q_i, for all positions $i < m$; and
- the choice function $\sigma(q_m)$ is undefined for all agents $a \in \Sigma$.

Next we formalise how partial strategies are combined. In this paper, we choose that combined partial strategies are complementing each other without causing non-terminating computations, cf. Remark 2.2. We define the operator $\oplus$ for combining two partial strategies as follows.

Definition 2.4. *[strategy combination] Let σ_1 and σ_2 be two strategies in an ATS $\mathcal{S}$ such that*

- *σ_1 and σ_2 have disjoint domains of definition ($\mathsf{dom}(\sigma_1) \cap \mathsf{dom}(\sigma_2) = \emptyset$);*
- *there is no infinite path $q_0 q_1 q_2 \dots$ in $\mathcal{S}$ where q_{i+1} is a σ_1- or σ_2-successor of q_i, for all $i \geq 0$.*

Then $\sigma_1 \oplus \sigma_2$ is defined as: For all states q in $\mathcal{S}$,

$$(\sigma_1 \oplus \sigma_2)(q) = \begin{cases} \sigma_1(q) & \text{if } q \in \mathsf{dom}(\sigma_1) \\ \sigma_2(q) & \text{if } q \in \mathsf{dom}(\sigma_2) \\ \text{undefined} & \text{otherwise} \end{cases}$$

$\lhd$

We denote with $\Upsilon_{\mathcal{S}}(R, q)$ the set of partial strategies for a schedule R at a state q in the ATS $\mathcal{S}$. Take a strategy $\sigma_1 \in \Upsilon_{\mathcal{S}}(R_1, q)$. For every schedule R_2, we now define the set of strategies that can be obtained by combining σ_1 with one strategy in $\Upsilon_{\mathcal{S}}(R_2, q')$, for every state q' that can be reached by σ_1 from q (i.e. $q' \in out(q, \sigma_1)$). Notice that, by using the operator $\oplus$ from Definition 2.4, the resulting strategies will not allow any looping computations. We collect this set of strategies in $\mathsf{completions}(q, \sigma_1, R_2)$ defined as follows.

Definition 2.5. *[strategy completion] Let σ_1 be a partial strategy starting at q, and let R_2 be a schedule.*

Set $\mathsf{completions}(q, \sigma_1, R_2) =$

$$\left\{ \sigma_1 \bigoplus_{q' \in out(q,\sigma_1)} \sigma_{q'} \mid \sigma_{q'} \in \Upsilon_{\mathcal{S}}(R_2, q') \right\}$$

$\lhd$

We are ready to define the meanings of schedules and ADL formulae by mutual induction.

Definition 2.6. *[ADL Semantics] Let Σ be a finite non-empty set of agents and let $\mathcal{S} = \langle \Pi, Q, \pi, \delta \rangle$ be an ATS for Σ. The set of partial strategies $\Upsilon_{\mathcal{S}}(R, q)$ for a schedule R at a state q and the satisfaction relation $\models$ are defined by simultaneous induction as follows: Let q range over states in Q, A over coalitions of agents and a over agents in Σ, R over schedules in $\mathfrak{R}$, and φ, φ_1, φ_2 over* ADL*-formulae.*

- *$\mathcal{S}, q \models p$ iff $p \in \pi(q)$ for all propositions $p \in \Pi$;*
- *$\mathcal{S}, q \models \neg\varphi$ iff $\mathcal{S}, q \not\models \varphi$;*
- *$\mathcal{S}, q \models \varphi_1 \vee \varphi_2$ iff $\mathcal{S}, q \models \varphi_1$ or $\mathcal{S}, q \models \varphi_2$;*
- *$\mathcal{S}, q \models \langle R \rangle \varphi$ iff there is a strategy $\sigma \in \Upsilon_{\mathcal{S}}(R, q)$ at q such that for all states $q' \in \mathsf{out}(q, \sigma)$, it holds that $\mathcal{S}, q' \models \varphi$;*

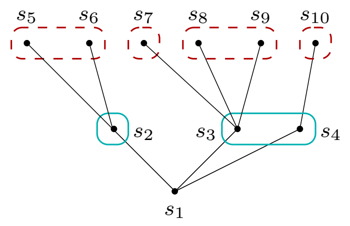
- *$\Upsilon_{\mathcal{S}}(\epsilon, q) = \{\sigma\}$, where $\mathsf{dom}(\sigma) = \emptyset$;*
- *$\Upsilon_{\mathcal{S}}(\varphi?, q) = \begin{cases} \Upsilon_{\mathcal{S}}(\epsilon, q) & \text{if } \mathcal{S}, q \models \varphi \\ \emptyset & \text{otherwise} \end{cases}$;*
- *$\Upsilon_{\mathcal{S}}(A, q) = \{\sigma \mid \sigma \text{ is a } (\delta, q, A)\text{-strategy}\}$;*
- *$\Upsilon_{\mathcal{S}}(R_1; R_2, q) = \bigcup_{\sigma \in \Upsilon_{\mathcal{S}}(R_1, q)} \mathsf{completions}(q, \sigma, R_2)$;*
- *$\Upsilon_{\mathcal{S}}(R_1 \sqcup R_2, q) = \Upsilon_{\mathcal{S}}(R_1, q) \cup \Upsilon_{\mathcal{S}}(R_2, q)$;*
- *$\Upsilon_{\mathcal{S}}(R^*; q) = \bigcup_{k \geq 0} \Upsilon_{\mathcal{S}}(R^k, q)$.*

If for some state q of some ATS $\mathcal{S}$ for Σ it holds that $\mathcal{S}, q \models \varphi$, then the ADL*-formula φ is* true *at q, and $\mathcal{S}$ is called a* model *of φ. An* ADL*-formula is* satisfiable *if it has a model, and it is* valid *if it is true at all states in any ATS for Σ.* $\lhd$

3. EXAMPLES

The following example illustrates in details the process of evaluation of a simple schedule.

Example 3.1. *Consider the ATS with endpoints $\mathcal{S} = \langle \Pi, Q, \pi, \delta \rangle$.*



Let $Q = \{s_1, \dots, s_{10}\}$. For the matter of the example, the important bits of the transition function are explicitly represented on the figure above and are as follows:

- *$\delta(a, s_1) = \{\{s_2\}, \{s_3, s_4\}\}$*
- *$\delta(b, s_2) = \{\{s_5, s_6\}\}$*
- *$\delta(b, s_3) = \{\{s_7\}, \{s_8, s_9\}\}$*
- *$\delta(b, s_4) = \{\{s_{10}\}\}$*

Moreover, we assume $\delta(b, s_1) = \{\{s_2, s_3\}, \{s_2, s_4\}\}$ and $\delta(a, s_2) = \{\{s_5, s_7\}, \{s_6, s_7\}\}$ and $\delta(a, s_3) = \{\{s_7, s_8\}, \{s_7, s_9\}\}$ and $\delta(a, s_4) = \{\{s_9, s_{10}\}\}$. δ is undefined otherwise. Π and π are not relevant here.

The aim of the example is to construct the set of partial strategies induced by the schedule $\{a\}; \{b\}$ at the set s_1, that is, the strategies in $\Upsilon_{\mathcal{S}}(s_1, \{a\}; \{b\})$.

$\Upsilon_{\mathcal{S}}(s_1, \{a\}) = \{\sigma_{a,1}, \sigma_{a,2}\}$ where $\sigma_{a,1}(s_1)(a) = \{s_2\}$ and undefined otherwise and $\sigma_{a,2}(s_1)(a) = \{s_3, s_4\}$ and undefined otherwise. We have precisely $out(s_1, \sigma_{a,1}) = \{s_2\}$ and $out(s_1, \sigma_{a,2}) = \{s_3, s_4\}$. It holds that:

$\Upsilon_{\mathcal{S}}(s_1, \{a\}; \{b\}) = \bigcup_{\sigma \in \{\sigma_{a,1}, \sigma_{a,2}\}} \mathsf{completions}(s_1, \sigma, \{b\})$.

We need to find the completions of $\sigma_{a,1}$ and the completions of $\sigma_{a,2}$. We then proceed. Let σ_{b_1} such that $\sigma_{b_1}(s_2)(b) = \{s_5, s_6\}$ and undefined otherwise. The outcome of $\sigma_{a,1}$ is uniquely s_2 and b only has one strategy at s_2, namely σ_{b_1}. Hence, $\mathsf{completions}(s_1, \sigma_{a,1}, \{b\}) = \{\sigma_{a,1} \oplus \sigma_{b_1}\}$.

How about completions$(s_1, \sigma_{a,2}, \{b\})$*? Since* $out(s_1, \sigma_{a,2}) = \{s_3, s_4\}$ *we have that the completions are the partial strategies consisting in the completion of* $\sigma_{a,2}$ *with exactly one* $(\delta, s_3, \{b\})$*-strategy plus exactly one* $(\delta, s_4, \{b\})$*-strategy.*

We start with s_4*. At* s_4*, b has only the choice* $\{s_{10}\}$*, hence the strategy* $\sigma_{b,2}$ *such that* $\sigma_{b,2}(s_4)(b) = \{s_{10}\}$ *will always be a brick of all the completions we are looking for.*

Let us look at s_3*. Let* $\sigma_{b,3}(s_3)(b) = \{s_7\}$ *and undefined otherwise and let* $\sigma_{b,4}(s_3)(b) = \{s_8, s_9\}$ *and undefined otherwise. They are the only two strategies of b at* s_3*. Then, for every completion either* $\sigma_{b,3}$ *or* $\sigma_{b,4}$ *is a constituent.*

We can now define completions$(s_1, \sigma_{a,2}, \{b\}) = \{\sigma_{a,2} \oplus \sigma_{b,3} \oplus \sigma_{b,2},\ \sigma_{a,2} \oplus \sigma_{b,4} \oplus \sigma_{b,2}\}$

We can conclude that $\Upsilon_{\mathcal{S}}(s_1, \{a\}; \{b\}) = \{\sigma_{a,1} \oplus \sigma_{b_1},\ \sigma_{a,2} \oplus \sigma_{b,3} \oplus \sigma_{b,2},\ \sigma_{a,2} \oplus \sigma_{b,4} \oplus \sigma_{b,2}\}$*.*

Example 3.1 makes clear how the schedules give rise to partial strategies constructed from smaller chunks.

Now, we present another example to explain the role played by the test operation.

Example 3.2. *Consider the simple ATS with endpoints* $\mathcal{S} = \langle \Pi, Q, \pi, \delta \rangle$ *such that* $\Pi = \{p, q\}$*,* $Q = \{s_1, s_2, s_3, s_4\}$*,* $\delta(a, s_1) = \{\{s_1, s_2\}, \{s_3, s_4\}\}$*,* $\delta(b, s_1) = \{\{s_1, s_3\}, \{s_2, s_4\}\}$ *and* $\pi(p) = \{s_3\}$ *and* $\pi(q) = \{s_1, s_3\}$*.*

We want to evaluate the formula $\langle A; p?\rangle q$ *at* s_1*.*

$\Upsilon_{\mathcal{S}}(s_1, \{a, b\}; p?) = \bigcup_{\sigma \in \Upsilon_{\mathcal{S}}(s_1, \{a,b\})} \text{completions}(s_1, \sigma, p?)$

There are four partial strategies in $\Upsilon_{\mathcal{S}}(s_1, \{a, b\})$*. They exhaustively yield every state in Q. We assume that from* s_1 *every* $\sigma_i \in \Upsilon_{\mathcal{S}}(s_1, \{a, b\})$ *lead to* s_i*.*

Since for every $i \in \{1, 2, 4\}$ *we also have that* $\mathcal{S}, s_i \not\models p$*, we have* $\Upsilon_{\mathcal{S}}(s_i, p?) = \emptyset$*. Hence,* completions$(s_i, \sigma_i, p?) = \emptyset$*. However,* $\mathcal{S}, s_3 \models p$*. Thus we have* $\Upsilon_{\mathcal{S}}(s_3, p?) = \{\sigma\}$ *such that* dom$(\sigma) = \emptyset$*. Clearly* dom$(\sigma_3) \cap$ dom$(\sigma) = \emptyset$ *and* $\sigma_3 \oplus \sigma = \sigma_3$*.*

Hence, we have $\Upsilon_{\mathcal{S}}(s_1, \{a, b\}; p?) = \{\sigma_3\}$*. It is also the case that* $out(s_1, \sigma_3) = \{s_3\}$ *and* $\mathcal{S}, s_3 \models q$*.*

So there is a strategy in $\Upsilon_{\mathcal{S}}(s_1, \{a, b\}; p?)$ *(viz. where a chooses* $\{s_3, s_4\}$ *b chooses* $\{s_1, s_3\}$*) and such that every outcome (viz.* s_3*) satisfies p. As a consequence we have* $\mathcal{S}, s_1 \models \langle \{a, b\}; p?\rangle q$*.*

In Example 3.2, the interested reader can also check that $\mathcal{S}, s_1 \models \langle \{a, b\}\rangle p \wedge q$. In fact, it is the case that in ADL, $\langle \{a, b\}; p?\rangle q$ is equivalent to $\langle \{a, b\}\rangle\langle p?\rangle q$, which in turn as we have seen before is equivalent to $\langle \{a, b\}\rangle(p \wedge q)$.

4. VERIFICATION

In this section, we present a model-checking algorithm for ADL. The *model-checking problem* for ADL is, given an ADL-formula φ and an ATS $\mathcal{S} = \langle \Pi, Q, \pi, \delta \rangle$ with endpoints for Σ, to compute the extension $[\![\varphi]\!]_{\mathcal{S}}$ of φ in $\mathcal{S}$, where $[\![\varphi]\!]_{\mathcal{S}}$ is the set of states in $\mathcal{S}$ that all satisfy φ. For computing the extension of a formula, we employ a modified version of the symbolic model-checking algorithm for ATL from [4]. The algorithm recursively computes, for each subformula ψ of φ, its extension $[\![\psi]\!]_{\mathcal{S}}$ in $\mathcal{S}$. For computing the extension of formulas of the form $\langle R\rangle\psi$, we employ a modified pre-image operator $\mathsf{Pre}_{\mathcal{S}}$ that can operate with schedules on ATSs. Then the function $\mathsf{Pre}_{\mathcal{S}}$ yields the extension of $\langle R\rangle\psi$ when called with the parameters R and $[\![\psi]\!]_{\mathcal{S}}$. Recall that schedules are regular expressions over coalitions and tests.

```
1. function $[\![\varphi]\!]_{\mathcal{S}}$ returns a set of states in $\mathcal{S}$
2.     case $\varphi = p$:        return $\{q \in Q \mid q \in \pi(p)\}$
3.     case $\varphi = \neg\psi$:       return $Q \setminus [\![\psi]\!]_{\mathcal{S}}$
4.     case $\varphi = \psi \vee \theta$: return $[\![\psi]\!]_{\mathcal{S}} \cup [\![\theta]\!]_{\mathcal{S}}$
5.     case $\varphi = \langle R\rangle\psi$: $\ell = \mathsf{Pre}_{\mathcal{S}}(R, (\epsilon, [\![\psi]\!]_{\mathcal{S}}, NIL))$
6.                        return $Q'$, where $\ell = (\cdot, Q', \cdot)$
7. end-function
```

Figure 1: Extension function

As tests range over ADL-formulas, the function $\mathsf{Pre}_{\mathcal{S}}$ is computed by simultaneaous recursion together with the function computing the extension $[\![\cdot]\!]_{\mathcal{S}}$. Figure 1 presents the algorithm computing $[\![\cdot]\!]_{\mathcal{S}}$ and Figure 2 the algorithm computing $\mathsf{Pre}_{\mathcal{S}}$.

Consider the algorithm computing the function $[\![\cdot]\!]_{\mathcal{S}}$. The algorithm takes as input an ADL-formula and an ATS with endpoints. The Lines (2), (3) and (4) in Figure 1 cover the base case of propositional variables and the Boolean cases. Lines (5) and (6) deal with formulas of the form $\langle R\rangle\varphi$ by calling the function $\mathsf{Pre}_{\mathcal{S}}$. This function accepts two parameters: a schedule and a data structure. The data structure is a linked list, which is initialised using $[\![\varphi]\!]_{\mathcal{S}}$. The role of the linked list will be explained below. Then $\mathsf{Pre}_{\mathcal{S}}$ yields the set of states from which R can be executed to ensure that all outcome states satisfy φ.

The algorithm in Figure 2 has five cases, one for each schedule operator; cf. the definition of schedules in Definition 2.1. Lines (2) to (5) cover the base case where the schedule is a coalition. The set ρ computed in Line (3) contains the states at which there is an A-choice that lies inside Q'. In Line (4), the set of 'looping states' is computed by calling the function loop-states$_{\mathcal{S}}$ in Figure 3, and subsequently this set is removed from ρ. With 'looping states' we mean states that necessarily give rise to self-loops when certain linear schedules are executed at them. Note that rejecting looping states corresponds to the semantics of ADL. The function completions in Definition 2.5 is responsible for constructing complex strategies for sequentially composed schedules. Observe that completions does not yield any strategy which allows for looping paths.

Lines (6) and (7) in Figure 2 cover the other base case where the schedule is a test ψ?. Here a cross function call $[\![\psi]\!]_{\mathcal{S}}$ is made to compute the extension of ψ?. Lines (8) to (13) cover the cases for complex schedules with choice and sequential composition as topmost operator. Here the application of $\mathsf{Pre}_{\mathcal{S}}$ follows the semantics in a straightforward way. The notation in Lines (8), (9) and (18) is used for convenience. That is, each dot in the expression $(\cdot, Q_1, \cdot)$ means that at this point we do not care about the value of the component of the tuple. The remaining lines cover the last case where the input schedule is S^*. Here the control structure is similar to the one of the until-case in the model checking algorithm for ATL. The set ρ contains the set of states at which the schedule S can be executed n times, for any $n \geq 0$. The set ρ is initialised as empty set in Line (15). The following while-loop in Lines (16) to (19) successively adds more states to ρ until a least fixpoint is reached, i.e., until no more states are added. Finally, Line (20) returns the result.

The function $\mathsf{Pre}_{\mathcal{S}}$ takes a linked list together with a coalition as argument and returns a linked list. A linked list is

```
function $\mathsf{Pre}_{\mathcal{S}}(R,\ell)$ with $\ell=(C,Q',\ell')$ returns a linked list
  case $R=A$:
    $\rho=\{q\in Q \mid \Delta\subseteq Q'$ for some $A$-choice $\Delta$ at $q\}$
    $\rho=\rho\setminus$ loop-states$_{\mathcal{S}}(A,\rho,\epsilon,\ell)$
    return $(A,\rho,\ell)$
  case $R=\varphi?$:
    return $(\epsilon,[\![\varphi]\!]_{\mathcal{S}}\cap Q',NIL)$
  case $R=R_1\sqcup R_2$:
    $(\cdot,Q_1,\cdot)=\mathsf{Pre}_{\mathcal{S}}(R_1,\ell)$
    $(\cdot,Q_2,\cdot)=\mathsf{Pre}_{\mathcal{S}}(R_2,\ell)$
    return $(\epsilon,Q_1\cup Q_2,NIL)$
  case $R=R_1;R_2$:
    return $\mathsf{Pre}_{\mathcal{S}}(R_1,\mathsf{Pre}_{\mathcal{S}}(R_2,\ell))$
  case $R=S^*$:
    $\rho=\emptyset$
    while $Q'\not\subseteq\rho$ do
      $\rho=\rho\cup Q'$
      $\ell=(\cdot,Q',\cdot)=\mathsf{Pre}_{\mathcal{S}}(S,\ell)$
    done
    return $(\epsilon,\rho,NIL)$
end-function
```

Figure 2: Pre-image function

```
function loop-states$_{\mathcal{S}}(A,Q',R,\ell)$ returns a set of states in $\mathcal{S}$
  case $\ell=NIL$:
    return $\emptyset$
  case $\ell=(C,Q_C,\ell')$:
    $\rho=\{q\in Q'\cap Q_C \mid q\in\Delta$ for all $A;R$-choices $\Delta$ at $q\}$
    return $\rho\cup$ loop-states$_{\mathcal{S}}(A,Q',R;C,\ell')$
end-function
```

Figure 3: Function to compute 'looping states'

an ordered list that is used to ensure the uniform execution of schedules at states. Each element of a linked list contains a set of states together with a coalition and a pointer to the successor element. The pointer has the value NIL if no successor element exists. The use of a linked list is as follows: Let $\ell=(A,Q_A,\ell')$ and $\ell'=(\epsilon,Q_\varphi,NIL)$, where ℓ is a linked list of length two, and ℓ' is one of length one. The set Q_A contains states q at which there is an A-choice that lies inside the set Q_φ, which is the extension of φ in $\mathcal{S}$. That is, we have $\mathcal{S},q\models\langle A\rangle\varphi$.

Figure 3 presents the algorithm that computes the function loop-states$_{\mathcal{S}}$. It takes a coalition A, a set of states Q', a schedule R, and a linked list ℓ as arguments. The function returns a set of 'looping states'. A state q is a *looping state* wrt. R if for every strategy $\sigma\in\Upsilon_{\mathcal{S}}(R,q)$, it holds that $q\in k^n(\sigma,q)$. Each computed looping state is contained in $Q'\cap Q_i$, for $i=1..n-1$, where $Q_1,\ldots,Q_n$ are the sets of states stored in this order in ℓ.

The fact that $\mathsf{Pre}_{\mathcal{S}}$ operates on schedules and the simultaneous recursion of this function with $[\![\cdot]\!]_{\mathcal{S}}$ does not affect the complexity of the model-checking algorithm. That is, ADL model-checking is no more complex than that for ATL [4]. Moreover, the problem of model-checking ADL trivially contains the problem of model-checking Coalition Logic [7].

THEOREM 4.1. *The model-checking problem for* ADL *is* PTIME-*complete.*

5. APPLICATION

We informally explain how ADL can help at optimising the coalition structures within a multi-agent system.

In cooperative game theory, coalitions are allowed to form and disassemble depending on the incentives of the individuals and on the effectivities of the coalitions. There are three activities of coalition formation that are interconnected. (i) Generating the coalition structure, that is deciding who is going to act together. (ii) Solving the optimisation problem of each coalitions. The coalition objective is to maximise their utility: the utility received from the system minus the cost of using resources. (iii) Dividing the value of the activity of coalitions.

Coalition formation can be studied using a *characteristic function* (e.g., [9]). A characteristic function maps every coalition A to a value $v(A)\in\mathbb{R}$, which represents the performance of this coalition. A value of a coalition A is assumed the same whatever how the players outside A are teamed up. This common assumption is problematic, though. For instance a large monopolistic coalition competing only with a number of small coalitions may perform better than if they are competing within an oligopoly against a few other large coalitions. A more general model is then based on *partition functions* and allows to model these particular externalities (e.g., [8]). The performance of a coalition also depends on the other coalitions. The value of a coalition depends on the coalition partition.

Coordinated coalitions as introduced in this paper are another generalisation. It allows to lift the study of coalitional games from coalitions (which coalition should an agent join in order to maximise her utility) to scheduled coalitions (which coalition should an agent join and under which agreement about how to coordinate).

Such concepts are of primary importance in the study of organisations where agents form coalitions upon their utility to be part of a particular group, interacting with respect to a precise procedural contract. An agent will joint a coordinated coalition if it helps her out to achieve her goals and that she only has to provide a reasonable workload.

We now see how ADL can be regarded as an appropriate framework for the formal analysis of the problem of optimisation. The next example illustrate how ADL model checking can be used as a semi-automatic method of optimising the coordination of a coalition within a system.

EXAMPLE 5.1. *Suppose that the activity of a startup company is conceptualised by a pointed model $\mathcal{M}$ (as an ATS or Reactive Modules [3]). The employees of the company are a, b and c. The goal of the company is to reach a good quality of production (noted* `good`*) within thirty days. Their goal is then to maintain the quality for two years (735 days) and sell the company for a good price (noted* `windfall`*) to a large holding company for instance. They can verify that they are able to do so:* $\mathcal{M}\models\langle\{a,b,c\}^{30};(\mathtt{good}?;\{a,b,c\})^{735}\rangle(\mathtt{good}\wedge\mathtt{windfall})$.

However, they realise that they can be more efficient: $\mathcal{M}\models\langle\{a,b,c\}^{30};((\mathtt{good}?;\{a,b,c\})^{6};\emptyset)^{105}\rangle(\mathtt{good}\wedge\mathtt{windfall})$. *After one month, they all can have a day off every week and still make a good production.*[1]

[1]The formula also capture the fact that the good quality of production will not be ensured at the resting day.

Even better: $\mathcal{M} \models \langle \{a,b,c\}^{30}; ((\texttt{good}?; (\{a,b\} \sqcup \{a,c\} \sqcup \{b,c\}))^{6}; \emptyset)^{105}\rangle(\texttt{good} \wedge \texttt{windfall})$. *Six days a week, they can work in teams of two, and all can rest the seventh day, still making good product.*

Finally, they figure out that their first goal to sell the company after two years is in fact pessimistic, as they could obtain a good price in about 20 *months (*$88 \times 7 = 616$ *days) after the initial month, provided they work constantly together five days a week:* $\mathcal{M} \models \langle \{a,b,c\}^{30}; ((\texttt{good}?; \{a,b,c\})^{5}; \emptyset^{2})^{88}\rangle$ $(\texttt{good} \wedge \texttt{earlywindfall})$.

From a modelling perspective, we need a method to generate the value of each coalitions. In coalitional games, this is often considered given in the input of the problem. But this is not satisfying from a practical point of view. In [10] for instance, the proposed model takes into account the cost of computation in terms of CPU cycles.

In our situation, the performance of a coalition can be correlated to a positive and a negative value that respectively depends on (i) the effectivity of the coordinated coalition to achieve some states preferred by the members of the coalition (ii) and the resource consumption of the coordinated coalition.

Concerning (i), we will need to introduce the preferences of the agents in the framework. This is beyond the scope of this paper. However, ADL provides the adequate formalisation to reason about effectivities of coordinated coalitions. For this reason we focus our attention to (ii). The cost of resources consumption may well be regarded as a function of the *complexity* of the coordinated coalition: length of the schedule, workforce used at every step, etc.

In Example 5.1 we kept this notion of complexity of a coordinated coalition rather abstract. The next example illustrates a possible notion of schedule complexity.

Example 5.2. *We consider a unit of production that is common in project management:* man-day. *The costs of the different schedules are as follows:*

- *1st schedule:* $(30 \times 3) + (3 \times 735) = 2295$ *man-days;*
- *2nd schedule:* $(30\times3)+((6\times105)*3) = 1980$ *man-days;*
- *3rd schedule:* $(30 \times 3) + ((6 \times 105) \times 2) = 1350$ *man-days;*
- *4th schedule:* $(30\times3)+((5\times88)\times3) = 1410$ *man-days.*

The first, second and third schedules have the same length (schedule three is non deterministic but only produces schedules of fixed length). As such, the third schedule is arguably the most 'desirable' among them. They are somewhat incomparable with the schedule four. Even though the cost of schedule four is greater than the cost of the third schedule, the fourth schedule still appears appealing. Their respective desirability will depend on the preferences of the agents regarding `windfall` and `earlywindfall`.

We would need to introduce for the agents a simple operator of preference. Similar work exist in logics of multi-agent ability (see [1] or [11]) and could be a source of inspiration. There is no reason for it to be problematic. However, though needed it may be more challenging to introduce group preferences. This is a non-trivial problem and a hot topic in social choice theory [5].

Concerning the 'negative correlation of desirability' it is still not clear what should be the cost of a schedule in general. The measure of complexity of a schedule may be particularly difficult to characterise when we consider non-deterministic schedules involving '$*$' and '$\sqcup$' since they can lead to schedules of very different length.

6. DISCUSSION

Axiomatisation.

An axiomatisation can help to clarify the relationship of ADL with other logics like CL, ATL and PDL.

(TAUT)	Propositional tautologies
(Sequence)	$\langle R_1; R_2\rangle\varphi \rightarrow \langle R_1\rangle\langle R_2\rangle\varphi$
(Choice)	$\langle R_1 \sqcup R_2\rangle\varphi \leftrightarrow \langle R_1\rangle\varphi \vee \langle R_2\rangle\varphi$
(Star)	$\langle R^*\rangle\varphi \leftrightarrow \varphi \vee \langle R; R^*\rangle\varphi$
(Test)	$\langle\psi?\rangle\varphi \leftrightarrow \psi \wedge \varphi$
(Ind)	$\langle R^*\rangle\varphi \rightarrow \varphi \vee \langle R^*\rangle(\neg\varphi \wedge \langle R\rangle\varphi)$
($\bot$)	$\neg\langle A\rangle\bot$
(Σ)	$\neg\langle\emptyset\rangle\neg\varphi \rightarrow \langle\Sigma\rangle\varphi$
(S)	$(\langle A\rangle\varphi \wedge \langle B\rangle\psi) \rightarrow \langle A \cup B\rangle(\varphi \wedge \psi)$ when $A \cap B = \emptyset$
(Modus Ponens)	$\dfrac{\varphi, \varphi \rightarrow \psi}{\psi}$
($\neg\langle A\rangle\neg$-Necessitation)	$\dfrac{\varphi}{\neg\langle A\rangle\neg\varphi}$
($\langle R\rangle$-Monotonicity)	$\dfrac{\varphi \rightarrow \psi}{\langle R\rangle\varphi \rightarrow \langle R\rangle\psi}$

Figure 4: Axiom scheme for ADL.

Figure 4 presents a candidate axiomatic system for ADL.

As the 'next fragment' of ATL, it is straightforward that ADL is an extension of Coalition Logic. As such, the axiomatisation of CL is an obvious starting point. The axioms (TAUT), ($\bot$), (Σ) and (S), and the inference rules (Modus Ponens), and ($\langle R\rangle$-Monotonicity) are principles of Coalition Logic.

Analogously, ADL has a lot in common with PDL. (Sequence), (Choice), (Star), (Test), (Ind) and ($\neg\langle A\rangle\neg$-Necessitation) are inspired from PDL. However, there are some little yet important differences. In PDL, the direction from right to left of the axiom (Sequence) holds. This is not the case here. For this reason, we rewrote the axiom (Star) in a 'safe' way (the direct adaptation from PDL would be $\langle R^*\rangle\varphi \leftrightarrow \varphi \vee \langle R\rangle\langle R^*\rangle\varphi$, which does not hold.)

As yet, we do not say anything about the completeness of the axioms of Figure 4.

Memoryless vs. perfect recall strategies.

In this paper, we have assumed *memoryless strategies*: the domain of a strategy is Q. Alternatively, we can define a *perfect recall strategy* at q_0, as a strategy whose domain is the set $Q^+ = \{q_0 \cdots q_k \mid \exists\sigma,\ q_{i+1} \text{is a } \sigma\text{-successor at } q_i\}$ of all state sequences. In this situation, the agents can take their strategic decisions based on a perfect view on the past. In fact, working with perfect recall strategies is like restricting the ATSs to the class of tree-like ATSs. For ATL, it is known that it does not affect the logic. In ADL, however, there are satisfiable formulae that are not satisfiable in a tree-like model. The formula $\langle A\rangle\langle B\rangle p \wedge \neg\langle A;B\rangle p$ is one of them. Interestingly, the logic ADL on tree-like ATSs satisfies the right-to-left direction of the axiom (Sequence). ADL restricted to the class of tree-like ATSs then satisfies all the theorems of PDL. As a consequence, ADL with perfect recall strategies is a *proper extension* of both Coalition Logic and PDL. Moreover, this extension is *conservative* if the axiom system is complete for the class of tree-like ATSs.

Further variants.

As already mentioned in Remark 2.2, there are several ways of combining partial strategies. Each way conveys a very different interpretation of partial strategies. Which combination method is more appealing or suitable depends on the requirements and on the application at hand.

In Definition 2.4, we defined the operator $\oplus$ that combines two partial strategies provided they satisfy two conditions: they are disjoint, and they complement each other without causing infinite computations. That is, when using $\oplus$, we obtain strategies for a schedule that only allow computations of the same length as the length of the schedule. As a consequence, in this ADL-variant, no schedule gives rise to a non-terminating computation.

For an alternative strategy combination, we need to redefine $\oplus$. For instance, we could relax the conditions on partial strategies by allowing them to overlap and the combined strategy to cause infinite computations. Instead, we impose as condition that the partial strategies are not conflicting, i.e., they determine the same choice functions at points of overlap.

Definition 6.1. *[relaxed strategy combination] Let σ_1 and σ_2 be two strategies in an ATS $\mathcal{S}$ such that σ_1 and σ_2 are not conflicting ($\sigma_1(q) = \sigma_2(q)$, for all states $q \in \mathsf{dom}(\sigma_1) \cap \mathsf{dom}(\sigma_2)$). Then $\sigma_1 \oplus' \sigma_2$ is defined as: For all states q in $\mathcal{S}$ and all agents a in Σ,*

$$(\sigma_1 \oplus' \sigma_2)(q)(a) = \begin{cases} \sigma_1(q)(a) & \textit{if } a \in \mathsf{dom}(\sigma_1(q)) \\ \sigma_2(q)(a) & \textit{if } a \in \mathsf{dom}(\sigma_2(q)) \\ \textit{undefined} & \textit{otherwise} \end{cases}$$

$\lhd$

Another variant relaxes the condition even further and allows partial strategies to prescribe different but consistent choice functions. Formally, σ_1 and σ_2 can be combined if $\sigma_1(q)(a) = \sigma_2(q)(a)$, for all states $q \in \mathsf{dom}(\sigma_1) \cap \mathsf{dom}(\sigma_2)$ and all agents $a \in \mathsf{dom}(\sigma_1(q)) \cap \mathsf{dom}(\sigma_2(q))$. This strategy combination would refine choice functions. We see this method to be suitable for concurrent schedules.

A more incisive change of the semantics is achieved by employing non-deterministic strategies. Such strategies may prescribe a number of choices for an agent at a state. The strategy combination would then facilitate the addition of alternative choices. Analogously, one could think of the possibility to remove choices or to exclude agents from the coalition. The latter features, however, are likely to require additional schedule operators in the syntax of the language, whereas the former variations were purely semantic.

7. CONCLUSION

In this paper, we have proposed a logic reminiscent of Propositional Dynamic Logic and Coalition Logic, but also Alternating-time Temporal Logic. The language is similar to PDL, but we interpret atomic programs as coalitions, and compound programs as coordinated coalitions. The logic restricted to atomic programs trivially corresponds to Coalition Logic.

We have seen in the previous section that there are more than one way to interpret the strategies triggered by coordinated coalitions. We have chosen to offer a preliminary analysis of the framework by focusing on memoryless strategies that cannot cause a loop. We have shown that in this situation, the model checking problem is Ptime-complete. Moreover, we have seen that in the case of perfect recall strategies, the logic results in a proper extension of PDL and CL.

In the future, we plan to investigate thoroughly the different possible variants of ADL. We can change the interpretation of strategies, and we can augment the language with program constructs that have already been studied in PDL.

8. ACKNOWLEDGEMENTS

We are grateful to the reviewers of AAMAS'10. Nicolas Troquard is supported by the EPSRC grant EP/E061397/1. Dirk Walther was formerly supported by the EPSRC grant EP/E065279/1 in the UK and he is now supported by a 'Juan de la Cierva' postdoctoral fellowship in Spain.

9. REFERENCES

[1] T. Ågotnes, W. van der Hoek, and M. Wooldridge. On the logic of coalitional games. In *AAMAS'06*, pages 153–160, New York, NY, USA, 2006. ACM.

[2] T. Ågotnes and D. Walther. A logic of strategic ability under bounded memory. *J. of Logic, Lang. and Inf.*, 18(1):55–77, 2009.

[3] R. Alur, L. de Alfaro, T. A. Henzinger, S. Krishnan, F. Mang, S. Qaader, S. Rajamani, and S. Taşiran. mocha user manual, University of Berkeley, 2000.

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

[5] K. J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare*, volume 1. Elsevier, 2002.

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

[7] M. Pauly. A modal logic for coalitional power in games. *J. of Logic and Computation*, 12(1):149–166, 2002.

[8] T. Rahwan, T. Michalak, N. Jennings, M. Wooldridge, and P. McBurney. Coalition structure generation in multi-agent systems with positive and negative externalities. In *IJCAI'09*, pages 257–263, Pasadena, USA, 2009.

[9] T. Sandholm, K. Larson, M. Anderson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2):209–238, 1999.

[10] T. Sandholm and V. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1-2):99–137, 1997.

[11] N. Troquard, W. van der Hoek, and M. Wooldridge. A logic of games and propositional control. In *AAMAS'09*, pages 961–968. IFAAMAS, 2009.