Reasoning about Action and Cooperation

Luigi Sauro
Jelle Gerbrandy
Dipartimento di Informatica
University of Torino, Italia
jelle@gerbrandy.com
sauro@di.unito.it

Wiebe van der Hoek
Michael Wooldridge
Department of Computer Science
University of Liverpool, United Kingdom
wiebe@csc.liv.ac.uk
mjw@csc.liv.ac.uk

ABSTRACT

We present a logic for reasoning both about the ability of agents to cooperate to execute complex actions, and how this relates to their ability to reach certain states of affairs. We show how the logic can be obtained in a modularised way, by combining a model for reasoning about actions and their effects with a model that describes what actions an agent can perform. More precisely, we show how one can combine an action logic which resembles Propositional Dynamic Logic with a cooperation logic which resembles Coalition Logic. We give a sound and complete axiomatisation for the logic, illustrate its use by means of an example, and discuss possible future extensions to it.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*coherence and coordination, multiagent systems*

General Terms

Theory

Keywords

Agent and multi-agent architectures, Cooperation and coordination among agents, Logics for agent systems, (Multi-)Agent planning

1. INTRODUCTION

When reasoning about cooperation in multi-agent systems, at least two questions seem important: *Who* can achieve some state of affairs and *How* can they obtain it? Historically, the latter question has received far more attention than the former. In the AI planning community, for example, it is known as the means-ends reasoning problem, and indeed the AI planning project is directed at developing computer programs that can answer it efficiently [4]. In the logic community, the question has been studied largely through the medium of action and program logics, of which PDL is perhaps the best known example [7]. A basic issue in such logics is the construction of complex programs from simpler ones. Typical constructs to build complex programs include sequential composition, non-deterministic choice and tests. However, there are limits to what one can say in such logics. For example, in standard PDL, one cannot express that an action is *not* executed; nor is not straightforward to say that one complex action *subsumes*, or *implies* another. Last but not least, although there is a notion of concurrency in PDL, all processes are expected to behave as specified by the program under consideration, and there is no notion of competition or cooperation, and hence no need to express *who* can achieve something. Building on work in PDL and related formalisms, some early attempts in the multi-agent systems community to formalise the ability of agents to execute plans and bring about states of affairs include [8, 14, 9].

The question of *who* can achieve some state of affairs has received much less attention within the AI and logic community. Although Moore presented a modal analysis of ability as long ago as 1977 [11], and a number of other authors built on this work over the intervening three decades, it is only recently that the logical analysis of ability has entered the mainstream of multi-agent systems research. Two closely related formalisms have received particular attention: the Alternating-time Temporal Logic (ATL, [1]), in which one can express that a coalition can enforce a certain temporal property, and Coalition Logic ([12]), in which the basic expression states that a coalition is effective for some outcome. Another closely related formalism is the Coalition Logic for Propositional Control (CL-PC, [15]), where the propositional atoms under the control of an agent determine what he, or any coalition he is a member of, can achieve. All these approaches revolve around a model in which a set of agents, who may have conflicting interests, may or may not work together to obtain a certain state of affairs. Central is the notion of so-called α-ability: the capability of a group of agents to enforce a certain state of affairs, no matter what actions the other agents take. As in [15], we will call such logics *Cooperation Logics*.

Notice that in all of these approaches, the ability of groups of agents to obtain a state of affairs is modeled without an explicit representation of *how* these agents obtain them. In other words, these logics do not represent actions and plans in the object language, even though in some cases, actions or choices are present in the underlying semantics. However, such an *explicit* representation of action in the object language is important in many domains. For example, certain actions may be more costly than others, and the costs may be distributed among the agents helping to achieve the goal according to the actions they perform. Also, in some situa-

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

tions, it seems a good heuristic to formulate a plan before organising a coalition to execute it [16, 13]. Finally, using such a language for verification of a MAS would not only provide the user — or indeed, the agents themselves— with a proof *that* a desired goal of the system can be achieved, but it would also explicitly capture a *plan* of how to achieve it.

In this paper, we are interested in how the ability to obtain a state of affairs relates to the ability to execute certain, possibly complex, and possibly collective, actions. We develop a logic that combines operators from cooperation logics with an explicit representation of actions. We achieve this by dividing the problem in two parts: a logic for reasoning about actions and their effects (*à la* dynamic logic) together with a variant of coalition logic to reason about (groups of) agents and the actions that they can perform. In more detail, we study multiagent systems that consist of two separate parts: an *environment module* that describes actions and their effects, and an *agents module* describing how the agents can interact with the environment by performing certain actions.

We use the agents module to represent the ability of a coalition G to *execute* certain type of actions. Where β is a joint action (i.e., it denotes a set of actions to be executed concurrently), we write $\langle\langle G\rangle\rangle\beta$ to indicate that coalition G can execute the complex action β. In the environment, then, we can reason about the effect of actions using our environment logic, and write, for example $[\beta]\phi$ for 'every successful execution of β results in state where ϕ is true' (cf. PDL).

The cooperation modality of Coalition Logic expresses the ability of a group to obtain a certain state of affairs. This kind of expression can now be analysed using the two modules described above: a group G of agents has the ability to obtain a state of affairs ϕ just in case there exists a plan β that G is able to execute, which is guaranteed to result in ϕ.

Such an approach presumes that the effect of actions can be modeled more or less independently from the behaviour of the agents that perform these actions. This approach seems applicable to many real life scenarios. For example, an environment may be a communication channel via which agents send and receive messages. Or, the environment may be a Database Management System, defining actions and constraints in a general way, and the agents are users querying and updating the database.

Separating actions from the agents that perform them is not only practical from an operative viewpoint, but from a logical standpoint as well. It allows us to break up the problem of reasoning about planning in a multi-agent environment into two sub-problems, and to study the logic of each of these 'modules' in isolation before putting them together. This approach is also interesting, because it allows us to 'reuse' existing logics and exploit them as 'modules' in a richer system.

In Section 2 we will present the two building blocks of our framework. In section 2.1 we define a logic for the environment, in which one can reason about the effect of executing joint actions. The language can express that an action is not taken, but also, in a natural way, compares joint actions in relative 'strength': if the joint action β is stronger than α, then we also have that whatever is true as a result of α, is true when executing β. We then (in section 2.2) define our agents' module, in which the α-ability of coalitions can be expressed — but now, not with respect to states of affairs that can be reached, but for joint actions that can be guaranteed.

Section 2.3 puts these blocks together in a module called a multi-agent system. Here, we define $\langle\langle G\rangle\rangle\phi$, the α-ability of coalition G to bring about state of affairs ϕ, in terms of $\langle\langle G\rangle\rangle\alpha$ and $[\alpha]\phi$. Apart from the completeness results for the separate building blocks, which we judge to provide insight in the systems themselves, we believe that one of the interesting technical developments of our work relates to the fact that the completeness result for the *combined* logic is straightforward given the completeness of the two logics of the underlying modules. We see this as indicative of the success of the modularisation approach we adopt in this paper, and a good indication that it might be usefully applied elsewhere.

Section 3 is devoted to an example, illustrating how the logic can be used to represent properties of a multi-agent system. The example is a scenario in which agents in a coalition can collectively set variables which they are unable to change on their own. Section 4 indicates a few of the many possible extensions to our multi-agent system, and Section 5 concludes.

2. ACTION AND COOPERATION

In our setting, a multiagent system is composed of two *modules*: one module describes the *agents* in terms of the actions they can execute, another describes the *environment* that is populated by these agents, and in particular the effects of performing certain actions.

2.1 The environment module

There are many ways of modeling an environment. One very general way to model actions and their effects is by using a Labeled Transition System (LTS) [6, 2, 10]. An LTS is a directed graph in which the edges are labeled by elements from a set of atomic actions and the vertices are labeled with sets of propositional variables. Edges are called transitions, the vertices are called states. The idea is that if a state s is connected to another state s' by a transition labeled with an action a, this means that s' is a possible result of executing action a in state s. The sets of variables at each end of the transition tell us how the performance of the action changes the state of the environment, in terms of the propositions that change value.

One of the drawbacks of this basic LTS model is that it does not allow for modeling concurrency directly: there is no straightforward way to model the fact that performing certain actions together may have a very different effect than performing each of them separately.

In the present context, we are interested in the way that the choice of actions of different agents interact, so we propose to model the effect of actions being performed concurrently directly in the model. We do this by labeling the transitions not with single actions, but with *sets* of actions. Intuitively, a transition labeled with a set of actions A represents the effect of executing exactly those actions together, and no others. This approach to deal with concurrency is similar to that taken in [5].

DEFINITION 1 (ENVIRONMENT MODULE).
An environment module Env is a Set-Labeled Transition System *(SLTS) is a tuple* $\langle S, Ac, (\rightarrow)_{A\subseteq Ac}, P, \pi\rangle$ *where:*

1. S *is a set of states.*
2. Ac *is a finite set of actions.*
3. *For each* $A \subseteq Ac$, $\rightarrow_A$ *is a relation over* S.
4. P *is a set of propositional variables*
5. $\pi : (S \times P) \rightarrow \{1, 0\}$ *is an interpretation function that assigns to each state and each propositional variable a truth value.*

We require that the accessibility relations $\rightarrow_A$ *are* serial *for each set of actions* A*, i.e., that for each state* s *there is always an* s' *with* $s \rightarrow_A s'$. □

A state s is characterised by the propositional variables that are true and false in it, as given by $\pi(s)$, and by the possible effects of executing combinations of actions A concurrently, as defined by the accessibility relations $\rightarrow_A$. We will refer to a set of actions A as a *concurrent action*. Actions may be *non-deterministic*, in the sense that the outcome of performing a set of actions may not always determine a unique result.

To reason about actions and their effect in an SLTS, we use a language that is similar to that of Propositional Dynamic Logic. PDL has a language that consists of two parts: an action language to describe actions, and a 'static' language to describe properties of states. However, rather than directly incorporating the program constructs of PDL, we use *Boolean combinations* of atomic actions to reason about sets of actions.

DEFINITION 2 (ACTION EXPRESSIONS).
Given the set of atomic actions Ac, the set of action expressions *is defined by the grammar:*

$$\alpha ::= a \mid \alpha \wedge \alpha \mid \neg\alpha$$

where $a \in Ac$.

To interpret this action language, we define a relation $\models^p$ between subsets of Ac and action expressions. $A \models^p \alpha$ is to be read as 'concurrent action A is of type α', or 'α is true of A'.

DEFINITION 3 (INTERPRETATION OF ACTIONS).

1. $A \models^p a$ *iff* $a \in A$
2. $A \models^p \neg\alpha$ *iff* $A \not\models^p \alpha$
3. $A \models^p \alpha \wedge \beta$ *iff* $A \models^p \alpha$ *and* $A \models^p \beta$.

So, for example, the concurrent action $\{a, b\}$ is of type $a \wedge \neg c$, but it is not of type $\neg b$. Formally, the action language is of course simply classical Boolean propositional logic, with Boolean variables corresponding to atomic actions. We can define additional connectives, such as $\vee$ or $\rightarrow$, in the usual way.

DEFINITION 4 (LANGUAGE OF ENVIRONMENT LOGIC).
Given a set of actions Ac and a set of propositional variables P, the language of environment logic is given by:

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid [\alpha]\phi$$

where $p \in P$, and α is an expression of the action language.

In effect, the resulting language is Boolean Modal Logic [3]. The truth of sentences of environment logic is defined in the standard way, as follows.

DEFINITION 5 (SEMANTICS OF ENVIRONMENT LOGIC). *Let $Env = \langle S, Ac, (\rightarrow)_{A \subseteq Ac}, P, \pi\rangle$ be a SLTS, and s is a state in S. Truth of sentences of the language of environment logic is defined inductively as follows:*

1. $Env, s \models^e p$ *iff* $\pi(s, p) = 1$
2. $Env, s \models^e \phi_1 \wedge \phi_2$ *iff* $Env, s \models^e \phi_1$ *and* $Env, s \models^e \phi_2$.
3. $Env, s \models^e \neg\phi$ *iff* $Env, s \not\models^e \phi$
4. $Env, s \models^e [\alpha]\phi$ *iff for all* $A \subseteq Ac$ *and* $s' \in S$, *if* $A \models^p \alpha$ *and* $s \rightarrow_A s'$ *then* $Env, s' \models^e \phi$.

As usual we write $Env \models^e \phi$ when for all states $s \in S$, $Env, s \models^e \phi$. For example, if a is an atomic action, then $[a]\phi$ is true just in case executing a will always result in a state where ϕ is true, no matter what other actions are performed concurrently.

A sentence $[\neg a]\phi$ expresses that *not* doing a will always result in a state where ϕ is true – performing any concurrent action that does not involve a will results in a state where ϕ is true.

In a precise sense, the formula $[\alpha]\phi$ expresses that being of type α is a sufficient condition for a set of actions to guarantee that ϕ be true; the formula $[\neg\alpha]\neg\phi$ expresses that doing an action of type α is a *necessary* condition for reaching a state where ϕ is true – if the action is not of type α, for all the states in which the system will result ϕ is false.

The logic that results from this semantics is given by the following set of axioms and rules.

DEFINITION 6 (ENVIRONMENT LOGIC).
The environment logic *($\vdash^e$) has the following axioms*

1. *All axioms of propositional logic*
2. $[\alpha](\phi \rightarrow \psi) \rightarrow ([\alpha]\phi \rightarrow [\alpha]\psi)$ *for each α (distribution)*
3. *All axioms of the form $[\alpha]\phi \rightarrow [\beta]\phi$, if $\vdash \beta \rightarrow \alpha$ in propositional logic.*
4. $([\alpha]\phi \wedge [\beta]\phi) \rightarrow [\alpha \vee \beta]\phi$ *(additivity)*
5. $\neg[\alpha]\neg\top$ *if α is consistent in propositional logic (seriality)*
6. $[\bot]\ \bot$

Rules:

1. *from $\vdash^e \phi$ and $\vdash^e \phi \rightarrow \psi$, derive $\vdash^e \psi$ (modus ponens)*
2. *from $\vdash^e \phi$, derive $\vdash^e [\alpha]\phi$ (necessitation for each action expression α)*

We write $\Sigma \vdash^e \phi$ if ϕ can be derived from the set of sentences Σ using these rules and axioms.

Axiom 2 and the rule of necessitation state that the $[\alpha]$-operators are normal modal operators. Axiom 3 says that if β implies α in propositional logic, then, if an α-action always results in a ϕ state, then a β-action will as well (since any β-action is an α-action). Axiom 4 says that if all α-actions and all β-actions result in a state where ϕ is true, then also all $\alpha \vee \beta$-actions will. Finally, the seriality axiom reflects that any consistent combination of actions will lead to some resulting state.

This logic is sound and complete with respect to the semantics.

PROPOSITION 7 (SOUNDNESS AND COMPLETENESS).
Environment logic is sound and complete with respect to environment models.

$$\vdash^e \phi \quad \textit{iff} \quad Env \models^e \phi \textit{ for each environment model } Env.$$

proof: We give the main steps of a proof. Construct a canonical model in the usual way, by taking maximal consistent sets of the logic as states in the model. The following *truth lemma* holds for each maximal consistent Σ:

$$\phi \in \Sigma \text{ iff } \Sigma \models^e \phi \text{ in the canonical model}$$

So, suppose $[\alpha]\psi \in \Sigma$. We need to show that $\Sigma \models^e [\alpha]\psi$. If α is not consistent this holds trivially. So, assume that $\Sigma \rightarrow_A \Gamma$ for some A such that $A \models^p \alpha$. Then $\psi \in \Gamma$ by definition of the

canonical model, and by induction hypothesis, $\Gamma \models^e \psi$. Since Γ was arbitrary, it follows that $\Sigma \models^e [\alpha]\psi$.

For the other direction. Before proving the general case, we first consider the case where our action expression is of the form $\alpha_A = \bigwedge A \wedge \bigwedge\{\neg a \mid a \in Ac \text{ and } a \notin A\}$ for some set of actions A.

Suppose $[\alpha_A]\psi \notin \Sigma$ for some A. With axiom 2 and the consistency of Σ, this means that $\{\chi \mid [\alpha_A]\chi \in \Sigma\} \cup \{\neg\psi\}$ is consistent. This set can be extended to a maximal consistent Γ. We show that $\Sigma \rightarrow_A \Gamma$. To see this, take an arbitrary $[\alpha]\chi \in \Sigma$ such that $A \models^p \alpha$. Then, in propositional logic, $\vdash \alpha_A \rightarrow \alpha$, and by axiom 3, it must hold that $[\alpha_A]\chi \in \Sigma$. But then, by construction of Γ, $\chi \in \Gamma$. Putting everything together, we have that $\Sigma \rightarrow_A \Gamma$, $A \models^p \alpha_A$, and, by induction hypothesis, $\Gamma \not\models^e \psi$. We conclude that $\Sigma \not\models^e [\alpha_A]\psi$.

Now we move to the general case. Suppose that $\Sigma \models^e [\alpha]\psi$ for arbitrary α. If α is not consistent, then we can use axioms 6 and 3 to conclude that $[\alpha]\psi$ is in Σ. If α is consistent, then the sentence α is equivalent in propositional logic to the sentence $\alpha^\vee := \bigvee\{\alpha_A \mid A \models^p \alpha\}$. Clearly, we have that $\Sigma \models^e [\alpha^\vee]\psi$; and also that, for axiom 3, $\Sigma \models^e [\alpha_A]\psi$ for each A such that $A \models^p \alpha$. By the previous argument, we have that $[\alpha_A]\psi \in \Sigma$ for each A such that $A \models^p \alpha$. With axiom 4, it follows then that $[\alpha^\vee]\psi \in \Sigma$. We can now use axiom 3 and the fact that $\alpha^\vee \rightarrow \alpha$, and conclude that $[\alpha]\psi \in \Sigma$. □

So, what we have now is a dynamic logic in which we can reason about *concurrent* actions. That is, we have a way of reasoning about the effect of performing *combinations* of actions, as well as about the effect of *not* performing certain actions. This dynamic logic will serve as the environment module of our multi-agent system. We will now proceed to define the agents that will populate this environment.

2.2 Agents Module

Agents are identified with the set of actions that they can perform.

DEFINITION 8 (AGENTS MODULE).
Given a set of agents Ag and a set of actions Ac, an agents *module is a tuple $\langle Ag, Ac, \mathsf{act}\rangle$, where act is a function $Ag \mapsto \mathcal{P}(Ac)$ that assigns to each agent i a subset $\mathsf{act}(i)$ of actions from Ac.*
We require that $\bigcup_{i\in Ag} \mathsf{act}(i) = Ac$.

We will abuse notation somewhat, and write 'act' as a *pars pro toto* for $\langle Ag, Ac, \mathsf{act}\rangle$. We denote with $\mathsf{act}(G)$ the set $\bigcup_{i\in G} \mathsf{act}(i)$ of actions that the group G of agents can perform.

To represent the ability of agents to perform actions of a certain type, we will use some notation from cooperation logic. For an action expression α, a sentence of the form $\langle\!\langle G\rangle\!\rangle\alpha$ expresses the fact that G has the power to bring it about that a set of actions of type α will be performed; we say that G is *effective for* α, that G is able to enforce α, or simply that G is able to do α. It is important to note that α here is an action type: we are dealing with the ability to *carry out a complex action*, not the ability to *bring about some state of affairs*.

A group G is able to enforce α exactly if there is a subset from the actions under their control that, no matter what actions the other agents choose, is such that the resulting concurrent action is of type α.

The way the semantics is defined is similar to that of the Coalition Logic of Propositional Control of [15], except that we explicitly allow for different agents to 'control' the same action.

DEFINITION 9 (ABILITY FOR ACTIONS).
Let $\langle Ag, Ac, \mathsf{act}\rangle$ be an agents module, let $G \subseteq Ag$ be a set of agents.

$\langle Ag, Ac, \mathsf{act}\rangle \models^a \langle\!\langle G\rangle\!\rangle\alpha$ *iff there is an $A \subseteq \mathsf{act}(G)$ such that for all $B \subseteq \mathsf{act}(Ag \setminus G)$ it holds that $A \cup B \models^p \alpha$*

So, for example, if act is an agents module in which an agent i can execute action a, then $\mathsf{act} \models^a \langle\!\langle\{i\}\rangle\!\rangle a$: i can enforce a. However, i can only enforce $\neg a$ if he is the only one that can execute a: $\mathsf{act} \models^a \langle\!\langle\{i\}\rangle\!\rangle\neg a$ only if $a \notin \mathsf{act}(j)$ for any j other than i. Similarly, $\mathsf{act} \models^a \langle\!\langle G\rangle\!\rangle(a \wedge \neg b)$ exactly when at least one agent in G can execute a, while no agent outside of G can do b.

DEFINITION 10 (COALITION LOGIC FOR ACTIONS).
The coalition logic for actions is given by the following axioms, together with all axiom schemes of propositional logic, and modus ponens. We denote derivability in this logic with $\vdash^a$.

1. $\langle\!\langle G\rangle\!\rangle\top$, *for all* $G \subseteq Ag$
2. $\langle\!\langle G\rangle\!\rangle\alpha \rightarrow \neg\langle\!\langle Ag \setminus G\rangle\!\rangle\neg\alpha$
3. $\langle\!\langle G\rangle\!\rangle\alpha \rightarrow \langle\!\langle G\rangle\!\rangle\beta$ *if* $\vdash \alpha \rightarrow \beta$ *in propositional logic.*
4. $(\langle\!\langle G_1\rangle\!\rangle\alpha \wedge \langle\!\langle G_2\rangle\!\rangle\beta) \rightarrow \langle\!\langle G_1 \cup G_2\rangle\!\rangle(\alpha\wedge\beta)$ *for all* $G_1, G_2 \subseteq Ag$ *such that* $G_1 \cap G_2 = \emptyset$
5. $\langle\!\langle G\rangle\!\rangle a \rightarrow \bigvee_{i\in G} \langle\!\langle\{i\}\rangle\!\rangle a$, *for all* $G \subseteq Ag$ *and atomic* $a \in Ac$
6. $(\langle\!\langle G\rangle\!\rangle\alpha \wedge \langle\!\langle G\rangle\!\rangle\beta) \rightarrow \langle\!\langle G\rangle\!\rangle(\alpha\wedge\beta)$ *if* α *and* β *have no atomic actions in common.*
7. $\langle\!\langle G\rangle\!\rangle\neg a \rightarrow \langle\!\langle G\rangle\!\rangle a$ *for atomic* $a \in Ac$.
8. $\langle\!\langle G\rangle\!\rangle\alpha \rightarrow \bigvee\{\langle\!\langle G\rangle\!\rangle \bigwedge\Phi \mid \Phi$ *is a set of literals such that* $\bigwedge\Phi \rightarrow \alpha\}$

An explanation of some of the axioms is in order.

The first axiom states that any group of agents is able to enforce some trivial action. Axiom 2 is typical of cooperation logics, and states that if a group G is able to enforce an action α, then the agents outside that group cannot enforce that an action that is not of type α will happen. This follows directly from the definitions.

Axiom 3 says that if a group of agents is able to enforce an α-action, and all α-actions are also of type β, then G is also able to enforce a β-action. Note that the axioms 1 and 3 give us a kind of necessitation, even for actions: If α is a classical tautology, then we also have $\vdash^a \langle\!\langle G\rangle\!\rangle\alpha$.

Axiom 4 states that if two independent groups of agents have the ability to enforce, respectively, actions of type α and of type β, then they can combine forces to enforce an $\alpha\wedge\beta$-action. This holds only if the two groups are disjoint: if this is not the case, and the two groups have an agent in common, say agent i, then it may be that i needs to execute action a to (help) enforce α, while he needs to refrain from doing a to enforce β: there is no single choice of actions that guarantees that $\alpha \wedge \beta$ will happen.

Axiom 5 says that if a group of agents has the ability to enforce an atomic action, then at least one of them must have that ability by himself.

Axiom 6 says that if a group can enforce actions α and β, and α and β involve completely different atomic actions, then the group can also enforce $\alpha\wedge\beta$ (by choosing the actions needed to guarantee α and performing them together with those that guarantee β).

The validity of axiom 7 is a consequence of the assumption in definition 8 that any atomic action can be executed by at least one agent. It says that if a group of agents can guarantee that an *atomic*

action is *not* performed, then one of the agents in that group must be able to execute that action.

Axiom 8 says that if a group of agents can perform an α-action, then there must be a set of literals (that is, formulas of the form a or $\neg a$) that together imply α, that they can enforce – that is, there must be a 'recipe' that tells them which actions to execute and which not to execute, that guarantees that α results.

PROPOSITION 11 (SOUNDNESS AND COMPLETENESS).
Coalition logic for actions is sound and complete with respect to agent models.

proof: Soundness is relatively straightforward. To prove completeness, we define, given a maximal $\vdash^a$-consistent set of sentences Σ, a function act for which the following truth lemma holds:

$$\mathsf{act} \models^a \phi \text{ iff } \phi \in \Sigma$$

which gives us an immediate completeness result. We define act by setting $\mathsf{act}(i) = \{a \mid \langle\langle i\rangle\rangle a \in \Sigma\}$. Note that with axioms 4 and 5, we have for atomic actions a that $a \in \mathsf{act}(G)$ iff $\langle\langle G\rangle\rangle a \in \Sigma$, a fact that we will use in the proof.

Before proving the truth lemma, we define, for a group of agents G and a set of actions $A \subseteq \mathsf{act}(G)$, the set $\Phi(A, G)$ to consist of all literals $A \cup \{\neg a \mid a \in \mathsf{act}(G), a \notin A \text{ and } a \notin \mathsf{act}(Ag \setminus G)\}$. The sentence $\Phi(A, G)$ characterises exactly those possible courses of action that might result when G chooses to do A.

We prove the truth lemma by induction on ϕ, where the only interesting case is the one where ϕ is of the form $\langle\langle G\rangle\rangle\alpha$.

Suppose first that $\mathsf{act} \models^a \langle\langle G\rangle\rangle\alpha$. That means that there is an $A \subseteq \mathsf{act}(G)$ such that for all $B \subseteq \mathsf{act}(Ag \setminus G) : A \cup B \models^p \alpha$. Clearly, then, $\vdash \bigwedge \Phi(A, G) \rightarrow \alpha$ in propositional logic.

By definition of act, it holds that $\langle\langle G\rangle\rangle a \in \Sigma$ for each $a \in A$. Similarly, we have for each $a \notin \mathsf{act}(Ag \setminus G)$ that $\langle\langle Ag \setminus G\rangle\rangle a \notin \Sigma$, from which it follows with maximality and axiom 2 that $\langle\langle G\rangle\rangle \neg a \in \Sigma$. Axiom 6 allows us to conclude that $\langle\langle G\rangle\rangle \bigwedge \Phi(A, G) \in \Sigma$, and by the monotony axiom 3 it follows that $\langle\langle G\rangle\rangle\alpha \in \Sigma$.

For the other direction, assume that $\langle\langle G\rangle\rangle\alpha \in \Sigma$. Since Σ includes axiom 8, there must be a set of literals Φ such that $\bigwedge \Phi \vdash \alpha \in \Sigma$ and $\langle\langle G\rangle\rangle \bigwedge \Phi \in \Sigma$. With axiom 3, we know that for each of the literals l in Φ it holds that $\langle\langle G\rangle\rangle l \in \Sigma$.

For positive literals a, we immediately have that $\langle\langle G\rangle\rangle a$ implies that $a \in \mathsf{act}(G)$.

For negative literals $\neg a$, we have with axiom 7 that $\langle\langle G\rangle\rangle \neg a$ implies that $a \in \mathsf{act}(G)$. Moreover, for the negative literals we can use axiom 2 to conclude that $a \notin \mathsf{act}(Ag \setminus G)$.

But this means that $\mathsf{act} \models^a \langle\langle G\rangle\rangle \bigwedge \Phi$, and since $\bigwedge \Phi \rightarrow \alpha$, that $\mathsf{act} \models^a \langle\langle G\rangle\rangle\alpha$. □

2.3 A multi-agent system – agents acting in an environment

In the previous two subsections we defined a module describing an environment, and a separate module describing the actions that agents can choose to perform, either by themselves or together. These two modules can be related by identifying the set of actions in the two respective modules. Such a combination provides us with a semantics for reasoning about agents that act in an environment by way of performing certain actions.

Formally, a multi-agent system is an environment together with an agent module that shares its action repertoire.

DEFINITION 12 (MULTI-AGENT SYSTEM).
A multi-agent system MaS is a tuple

$$\langle S, Ac, (\rightarrow)_{A \subseteq Ac}, P, \pi, Ag, \mathsf{act}\rangle$$

where:

1. $\langle S, (\rightarrow_{A'})_{A \subseteq Ac}, P, \pi\rangle$ *is an environment (in the sense of definition 1).*
2. $\langle Ac, Ag, \mathsf{act}\rangle$ *is an agent module (in the sense of definition 8).*

Of course, everything that we could express with the languages we used to talk about the environment and about the capabilities of the agents remains interesting in the combined system as well, so a logic for reasoning about a multi-agent system should include both. But in the new, more complex system, there are other notions of interest that did not make sense in the separate modules.

In particular, we are interested in the more standard operators of coalition logic that reason not about ability to enforce complex *actions*, but ability to enforce certain *results*. We will overload the operators $\langle\langle G\rangle\rangle$ for this purpose, and add sentences of the form $\langle\langle G\rangle\rangle\phi$ to the language, where ϕ can be any sentence. Intuitively, $\langle\langle G\rangle\rangle\phi$ means that the agents in G have the power to obtain a state where ϕ is true.

This leads to the following definition of a language for multi-agent systems:

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid [\alpha]\phi \mid \langle\langle G\rangle\rangle\alpha \mid \langle\langle G\rangle\rangle\phi$$

where $p \in P$, and α is an expression of the action language, as in definition 2. The satisfaction of formulae of the type $\langle\langle G\rangle\rangle\phi$ is defined as follows.

DEFINITION 13 (ABILITY).
MaS, $s \models^m \langle\langle G\rangle\rangle\phi$ *iff there is a set of actions* $A \subseteq \mathsf{act}(G, s)$ *such that for all* $B \subseteq \mathsf{act}(Ag \setminus G)$ *and for all states* t, *if* $s \rightarrow_{A \cup B} t$, *then MaS*, $t \models^m \phi$.

So, $\langle\langle G\rangle\rangle\phi$ is true in a certain state just in case the agents in G are able to execute a concurrent action that guarantees, independently from what the other agents do, the truth of ϕ. As for the environment module, *MaS* $\models^m \phi$ means that for all $s \in S$, *MaS*, $s \models^m \phi$.

The following proposition provides a precise link between the ability to obtain a certain state of affairs, and the ability to execute an action that is guaranteed to result in such a state. It states that a group has the ability to enforce the truth of a sentence just in case it is able to enforce a concurrent action that is guaranteed to result in a state in which that sentence is true:

OBSERVATION 14. *Given a multi-agent system MaS and a state s of its environment:*
MaS, $s \models^m \langle\langle G\rangle\rangle\phi$ *iff there exists an action expression* α *with MaS*, $s \models^m \langle\langle G\rangle\rangle\alpha$ *and MaS*, $s \models^m [\alpha]\phi$

proof: Assume that *MaS*, $s \models^m \langle\langle G\rangle\rangle\phi$, and let A be a witness for this fact. The action expression $\bigwedge \Phi(A, G)$ (the set of sentences we used in the proof of proposition 11) is the witness we are looking for: it holds that *MaS*, $s \models^m \langle\langle G\rangle\rangle \bigwedge \Phi(A, G)$, and that *MaS*, $s \models^m [\bigwedge \Phi(G, A)]\phi$.

For the other direction, let α be such that *MaS*, $s \models^m \langle\langle G\rangle\rangle\alpha$ and *MaS*, $s \models^m [\alpha]\phi$. By definition it follows from *MaS*, $s \models^m \langle\langle G\rangle\rangle\alpha$ that there must be a set of actions $A \in \mathsf{act}(G, s)$ such that $A \cup B \models^m \alpha$ for each $B \subseteq \mathsf{act}(Ag \setminus G, s)$. Take this A, and take any $B \subseteq \mathsf{act}(Ag \setminus G)$ and t such that $s \rightarrow_{A \cup B} t$. Since $A \cup B \models^p \alpha$ and *MaS*, $s \models^m [\alpha]\phi$, it must hold that $t \models^m \phi$. So *MaS*, $s \models^m \langle\langle G\rangle\rangle\phi$. □

The logic for the multiagent system obviously contains all axioms we already had from the two separate models. To obtain a complete axiom system, it suffices to add just two axioms that relate the ability to obtain a state of affairs with the ability to execute concurrent actions.

DEFINITION 15 (COOPERATION LOGIC WITH ACTIONS). *The cooperation logic with actions, $\vdash^m$, is given by:*

1. *All axioms and rules from the environment logic of section 2.1*
2. *All axioms and rules from the agent logic of section 2.2*
3. $(\langle\langle G\rangle\rangle\alpha \wedge [\alpha]\phi) \rightarrow \langle\langle G\rangle\rangle\phi$ *for each* α
4. $\langle\langle G\rangle\rangle\phi \rightarrow \bigvee\{\langle\langle G\rangle\rangle\alpha \wedge [\alpha]\phi \mid \alpha$ *is the conjunction of a set of action literals* $\}$

where 'action literals' are atomic actions a or their negations $\neg a$.

The two new axioms relate the ability to execute actions with the ability to enforce the truth of propositions, in the same way as is done in observation 14.

PROPOSITION 16 (SOUNDNESS AND COMPLETENESS). *Cooperation logic with actions is sound and complete with respect to action models:*

$$\vdash^m \phi \quad \textit{iff} \quad \textit{MaS} \models^m \phi \textit{ for all models MaS}$$

proof: Soundness follows from the soundness of the two modules and the previous observation 14.

Completeness is relatively straightforward with the completeness results of sections 2.1 and 2.2. We first construct a model *MaS* combining the constructions of the previous proofs in the straightforward way.

We need to prove a truth lemma stating that $\textit{MaS}, \Sigma \models^m \phi$ iff $\phi \in \Sigma$. This is proven by induction on ϕ, leaving the proofs of propositions 11 and 7 practically unchanged. The only interesting case is the new operator. So suppose that

ϕ is of the form $\langle\langle G\rangle\rangle\psi$

Suppose $\textit{MaS}, \Sigma \models^m \langle\langle G\rangle\rangle\psi$. With observation 14, there is an action expression α such that $\Sigma \models^m \langle\langle G\rangle\rangle\alpha$ and $\Sigma \models^m [\alpha]\psi$. By induction hypothesis and the previous two cases, we have that $\langle\langle G\rangle\rangle\alpha \in \Sigma$ and $[\alpha]\phi \in \Sigma$, and therefore, with our axiom 3 and the fact that Σ is maximal, that $\langle\langle G\rangle\rangle\phi \in \Sigma$.

For the other direction, assume that $\langle\langle G\rangle\rangle\psi \in \Sigma$. Then, with the axiom 4 and maximality of Σ, there is a set of action literals Φ such that $[\bigwedge \Phi]\psi \in \Sigma$ and that $\langle\langle G\rangle\rangle \bigwedge \Phi$ in Σ. We can conclude (by induction hypothesis and the previous cases) that $\Sigma \models^m [\bigwedge \Phi]\psi$ and $\Sigma \models^m \langle\langle G\rangle\rangle \bigwedge \Phi$. We then use observation 14 again to conclude that $\Sigma \models^m \langle\langle G\rangle\rangle\psi$. □

Summarising, we have defined an environment as a Set-Labelled Transition System, and defined a PDL-like logic for representing the performance of complex actions in such an environment. We gave a sound and complete logic for this logic. We then defined agents as actors that can choose to do certain sets of actions, and defined a sound and complete logic for reasoning about the ability of groups of agents to enforce that a certain type of concurrent action is executed. We then defined a notion of cooperative ability. This *modularisation* of a multi-agent system paid off: it turned out to be relatively easy to find a sound and complete axiomatisation for the combination of these two logics.

3. AN EXAMPLE

Often in a multi-agent system, certain variables can be changed only when two or more agents cooperate, while other variables can be changed by the agents acting by themselves. For example, a database may contain crucial data that can be changed only with the consent of all operators, while data of a lesser importance can be changed by any agent. For another example, a website may accept a request for information without further ado, but for accepting a transaction with a credit card, it will need the consent of both the customer and the credit card company. Similarly, a website such as Ebay will accept advertisements that are submitted by a single agent acting alone, but will not accept a transaction if this is not confirmed by all parties involved.

All of these examples are instances of a case in which certain variables are under control of single agents, while other variables can only be changed by groups of agents acting together. We will consider an instance of this schema, in which two agents control two variables. To make the example concrete, we will talk about prison guards that control two doors (see Figure 1). A prison is composed of two interiors, a dormitory and a courtyard. One door opens to the outside, and it is important that it is not opened by accident. It can therefore only be opened if both guards act together; if it is open, each of the agents can close it at will. The other door separates the cell block from the courtyard, and each guard can open that door by himself. The fact that the inner gate is open is expressed by a proposition $\mathsf{in\_open}$; if the outer gate is open, $\mathsf{out\_open}$ is true.

The state of the gates is ruled by four buttons: a_1, b_1, a_2 and b_2. The gate $\mathsf{in\_open}$ toggles its state if and only if at least one of a_1 and a_2 is pressed. The outer gate $\mathsf{out\_open}$ toggles its state if and only if both b_1 and b_2 are pressed.

This description can be captured by an environment model *Env* with four states determined by the fact whether the gates are open or not. For example, we will write 01 for the state in which the inner door is closed and the outer door is open. Each of the buttons corresponds to an atomic action in the model, and the transitions are as described (so, for example, it will hold that $01 \rightarrow_{\{a_2,b_2\}} 11$).

With our environment language, we can express properties of the environment such that the fact that the action $a_1 \wedge b_1 \wedge b_2$ ensures, in the state 00, that in the next state both the gates will be open:

$$\textit{Env}, 00 \models^e [a_1 \wedge b_1 \wedge b_2](\mathsf{in\_open} \wedge \mathsf{out\_open})$$

Similarly, the fact that if the outer gate is closed, the action $b_1 \wedge b_2$ is both a sufficient and necessary condition to open the outer gate is captured by:

$$\textit{Env} \models^e \neg\mathsf{out\_open} \rightarrow ([b_1 \wedge b_2]\mathsf{out\_open} \wedge [\neg(b_1 \wedge b_2)]\neg\mathsf{out\_open})$$

Two guards, gd_1 and gd_2, control the buttons. The first guard gd_1 has access to the buttons a_1 and b_1, whereas gd_2 has access to a_2 and b_2. This situation can be captured by an agents model with the same set of atomic actions as the environment, and a function act such that $\mathsf{act}(gd_1) = \{a_1, b_1\}$ and $\mathsf{act}(gd_2) = \{a_2, b_2\}$.

We can express that the action b_1 can be executed only by the guard gd_1 as $\langle\langle gd_1\rangle\rangle b_1 \wedge \neg\langle\langle gd_2\rangle\rangle b_1$ (strictly speaking, we should have written $\langle\langle gd_1\rangle\rangle$ instead of $\langle\langle\{gd_1\}\rangle\rangle$, but we are omitting the curly brackets for readability).

Together, the agents can push both b_1 and b_2, that is, $\mathsf{act} \models^a \langle\langle gd_1, gd_2\rangle\rangle(b_1 \wedge b_2)$. Analogously, both of them can, individually, execute the disjunction of a_1 and a_2: $\mathsf{act} \models^a \langle\langle gd_1\rangle\rangle(a_1 \vee a_2) \wedge \langle\langle gd_2\rangle\rangle(a_1 \vee a_2)$.

Combining the environment and the agents modules in multiagent system *MaS*, we obtain a model of the guards controlling the prison.

For example, since $b_1 \wedge b_2$ is a necessary condition to open the outer gate, and each guard gd_i can prevent the execution of b_i, it follows that both guards are needed to open the outer gate. Indeed, it holds that $\textit{MaS} \models^m \neg\mathsf{out\_open} \rightarrow (\neg\langle\langle gd_1\rangle\rangle\mathsf{out\_open} \wedge$

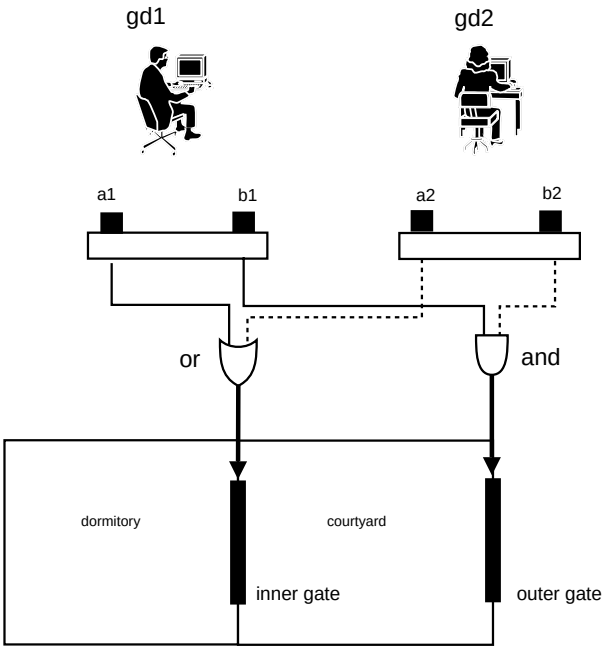

Figure 1: ***The prison example.***

$\neg\langle\langle gd_2\rangle\rangle$out_open)). Together, they can, indeed, open the door: *MaS* $\models^m \langle\langle gd_1, gd_2\rangle\rangle$out_open. To escape, the prisoner will need to take both of the guards hostage to open the outer door.

On the other hand, we want to be sure that when either one of the guards notes that there's a fire in the dormitory, he can open the inner gate to let the prisoners into the courtyard. Since both the guards are able to execute at least one of the actions a_1 and a_2, act $\models^a \langle\langle gd_1\rangle\rangle(a_1 \vee a_2) \wedge \langle\langle gd_2\rangle\rangle(a_1 \vee a_2)$, and an action of type $a_1 \vee a_2$ opens the inner gate, *Env* $\models^e \neg$in_open $\rightarrow [a_1 \vee a_2]$in_open, each of the guards has the power to open the inner gate if required:

$$\textit{MaS} \models^m \neg\mathsf{in\_open} \rightarrow (\langle\langle gd_1\rangle\rangle\mathsf{in\_open} \wedge \langle\langle gd_1\rangle\rangle\mathsf{in\_open}).$$

4. RICHER SYSTEMS

We have studied a system in which 'plans' of agents to obtain certain goals (i.e., complex action expressions) never look more than one transition ahead. There are different ways of enriching the system. First of all, one may enrich the model itself, and, for example, take a more sophisticated view of agents, taking the information they have and/or their goals into account as well. But also if we leave the semantical model the way it is, there are ways to enrich the language to allow us to express other interesting properties of a system.

One obvious choice is to enrich the language on the level of the multi-agent system with other operators borrowed from ATL [1].

Let a strategy for a group of agents G be a function that assigns to each state a subset of actions from act(G). A strategy for G together with a strategy for the agents not in G defines a unique path through the model, and we can define logical operators that not only talk about those states that can be reached by performing a single action, but also about strategies that guarantee that a sentence will always remain true, or will eventually become so:

1. *MaS*, $s \models \langle\langle G\rangle\rangle\Box\phi$ iff there is a strategy for the agents in G such that, no matter what the other agents do, ϕ will be true in all the states along the resulting paths.
2. *MaS*, $s \models \langle\langle G\rangle\rangle\Diamond\phi$ iff there is a strategy for the agents in G such that, no matter what the other agents do, will eventually result in a state where ϕ is true.

What we want to do is 'decompose' these notions into independent statements about the environment and about the agent module, to obtain a theorem similar to that of observation 14. We will focus on the second construct: the ability of a coalition to *eventually* bring about some state of affairs.

To express the type of actions that guarantee that some state will obtain at some point arbitrarily far in the future, we need a richer action language. A more program-like language is suitable here, in which we can express things like 'continue doing α until ϕ is true.' One well-studied set of operators for expressing these kinds of properties are those of Propositional Dynamic Logic [7]. We obtain an action repertoire like the following:

$$\alpha, \beta ::= \gamma \mid ?\phi \mid \alpha;\beta \mid \alpha \cup \beta \mid \alpha^*$$

where γ is a simple action expression (a Boolean combination of atomic actions), as in definition 2 and ϕ is a sentence as in definition 4. These operators can be given their standard semantics, which we do not repeat here.

We will call these types of action expression 'plans', in conformance with the usage we make of them. Now, what we are looking for is the following type of 'decomposition' of the meaning of $\langle\langle G\rangle\rangle\Diamond\phi$ — which is defined in terms of strategies — into a statement about the ability to execute a certain action and a statement about the effect of this action, in a way that is similar to observation 14:

Let MaS consist of an agent module act *and an environment module Env. Then:*

MaS, $s \models \langle\langle G\rangle\rangle\Diamond\phi$ *iff there is a plan* α *such that* G *has the ability to perform* α *(in the action module* act*), and* α *is guaranteed to reach a state where* ϕ *is true (in the environment module Env).*

With action expressions to express complex plans, then, we need a way of expressing that a plan is guaranteed to result in a state where ϕ is true. Somewhat disappointingly, PDL does not provide us with a way of doing that: the expression $[\alpha]\phi$ captures that all halting executions end up in a ϕ-state. The problem is that α might not halt at all, in which case ϕ may never become true, even if $[\alpha]\phi$ is.

One way of remedying this lack of expressive power is to add a new predicate to the language to obtain sentences of the form halt(α), that are true in a state s exactly when all executions of α are guaranteed to halt.

We also need to extend the language of the agent module of the previous section with expressions that express the ability to perform these complex actions. If we assume that all agents can observe all static sentences, the following definitions are reasonable:

1. act $\models \langle\langle G\rangle\rangle ?\phi$ (since agents can observe anything)
2. act $\models \langle\langle G\rangle\rangle\alpha;\beta$ iff act $\models \langle\langle G\rangle\rangle\alpha$ and act $\models \langle\langle G\rangle\rangle\beta$
3. act $\models \langle\langle G\rangle\rangle\alpha \cup \beta$ iff act $\models \langle\langle G\rangle\rangle\alpha$ and act $\models \langle\langle G\rangle\rangle\beta$
4. act $\models \langle\langle G\rangle\rangle\alpha^*$ iff act $\models \langle\langle G\rangle\rangle\alpha$

Putting all of this together, we obtain the following conjecture:

CONJECTURE 17. *MaS*, $s \models \langle\langle G\rangle\rangle F\phi$ *iff there is a plan* α *(in the language extended with PDL-operators) such that MaS*, $s \models$

$[\alpha]\phi$ *(if* α *halts, it is in a* ϕ*-state, MaS,* $s \models \langle\alpha\rangle\top$ *(*α *is executable), MaS,* $s \models \mathsf{halts}(\alpha)$ *(*α *is guaranteed to halt), and MaS,* $s \models \langle\langle G\rangle\rangle\alpha$ *(the agent in* G *are able to enforce* α*).*

5. CONCLUSIONS

Many logic-based formalisms – cooperation logics – have been developed in order to represent the power of groups of agents to achieve certain states of affairs.

Pauly [12] provides a modal logic, Coalition Logic, which axiomatises α-ability – a notion of power developed in game theory. A state of affairs is denoted as a set of states that satisfy a propositional formula ϕ and, informally, a group of agents G is α-effective for ϕ if and only if there exists a joint strategy for G which, no matter what the other agents do, ensures the satisfaction of ϕ. Another example is CL-PC, van der Hoek et al. [15]. In CL-PC each agent controls the truth value of a set of variables. In this way, the notion of α-ability can be formalised a standard multi-modal Kripke model and the usual semantics for box and diamond operators can be used. Finally another well-known logic to reason about the α-ability is ATL, in which the expressivity of the previous approaches is enriched with CTL temporal formulas [1].

All these approaches directly describe the achievement power of groups of agents without explicitly representing what they actually do in order to achieve a certain state of affairs. This fact has two drawbacks. First, it may not be easy to reason about meta-information such as the costs required to achieve a certain state of affairs or how these costs are distributed. Second, even if these logics correctly describe whether a group of agents has the power achieve a state of affairs, they do not provide, as required in cooperative problem solving or social reasoning mechanism, any clue about which group of agents has the power to achieve a desired state of affairs.

Thus, inspired by work on cooperative problem solving [16] and social reasoning mechanisms [13], we have formalised a notion of power in two modules: in the first module, the environment module, we addressed the problem to describe a causal model of an environment in which several agents act at the same time. In this module it is possible to describe the fact that a plan, intended as a set of concurrent actions, assures a certain state of affairs. In the second module, the agents module, the capabilities of the agents are modeled and in particular the possibility for a group of agents to execute a plan no matter what the other agents do. Combining these two modules provides a straightforward way to describe the achievement power a group of agents: a group of agents has the power to achieve a state of affairs ϕ if there exists a plan α assuring ϕ and the group can execute α without being obstructed by the other agents.

We have provided a complete axiomatisation for the two separated modules as well as for their combination. We noted that the axiomatisation of the combined module is straightforward, given the completeness of the two logics of the underlying modules. This is evidence that the modularisation of the notion of power is possible.

Finally we also provided an indication of how to enrich the expressiveness of our framework, maintaining the modularisation uninjured, with operators borrowed from Dynamic Logic. Our future work is devoted to provide formal results in this sense.

6. REFERENCES

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

[2] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, 1982.

[3] G. Gargov and S. Passy. A note on boolean modal logic. In P. Petkov, editor, *Mathematical Logic, Proc. of Heyting88*, pages 311–321. Plenum Press, 1990.

[4] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers: San Mateo, CA, 2004.

[5] Laura Giordano, Alberto Martelli, and Camilla Schwind. Dealing with concurrent actions in modal action logic. In Henri Prade, editor, *ECAI 98. 13th European Conference on Artificial Intelligence*, 1998.

[6] D. Harel. Dynamic logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic Volume II — Extensions of Classical Logic*, pages 497–604. D. Reidel Publishing Company: Dordrecht, The Netherlands, 1984.

[7] David Harel, Jerzy Tiuryn, and Dexter Kozen. *Dynamic Logic*. MIT Press, Cambridge, MA, USA, 2000.

[8] D. Kinny, M. Ljungberg, A. S. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems — Selected Papers from the Fourth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-92 (LNAI Volume 830)*, pages 226–256. Springer-Verlag: Berlin, Germany, 1992.

[9] B. van Linder, W. van der Hoek, and J.J-Ch. Meyer. Formalizing abilities and opportunities of agents. *Fundamenta Informaticae*, 34(1,2):53–101, 1998.

[10] K.L. McMillan. *Symbolic Model Checking*. PhD thesis, CMU University, 1992.

[11] R. C. Moore. Reasoning about knowledge and action. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, Cambridge, MA, 1977.

[12] M. Pauly. A Modal Logic for Coalitional Power in Games. *Journal of Logic and Computation*, 12:146–166, 2002.

[13] J. S. Sichman and Y. Demazeau. On social reasoning in multi-agent systems. *Revista Iberoamericana de Inteligencia Artificial*, 13:68–84, 2001.

[14] M. P. Singh. *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications (LNAI Volume 799)*. Springer-Verlag: Berlin, Germany, 1994.

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

[16] M. Wooldridge and J. Jennings. Towards a theory of cooperative problem solving. In *Procs. of the Sixth European Workshop on Modelling Autonomous Agents in Multi-Agent Worlds (MAAMAW-94)*, 1994.