A model for integrating dialogue and the execution of joint plans

Yuqing Tang
Dept. of Computer Science
Graduate Center
City University of New York
365 Fifth Avenue
New York, NY 10016, USA
ytang@gc.cuny.edu

Timothy J. Norman
Dept of Computing Science
The University of Aberdeen
Aberdeen, AB24 3UE, UK
t.j.norman@abdn.ac.uk

Simon Parsons
Dept of Comp & Info Science
Brooklyn College
City University of New York
2900 Bedford Avenue
Brooklyn, NY 11210 USA
parsons@sci.brooklyn.cuny.edu

ABSTRACT

Coming up with a plan for a team that operates in a non-deterministic environment is a complex process, and the problem is further complicated by the need for team members to communicate while the plan is being executed. Such communication is required, for example, to make sure that information critical to the plan is passed in time for it to be useful. In this paper we present a model for constructing joint plans for a team of agents that takes into account their communication needs. The model builds on recent developments in symbolic non-deterministic planning, ideas that have not previously been applied to this problem.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and co-ordination; multiagent systems.*

General Terms

Languages, theory.

Keywords

Agent interaction, planning, dialogue.

1. INTRODUCTION

One of the fundamental problems in multiagent systems is how to get a team of agents to coordinate their behavior. While there are situations in which agents can do this without needing to communicate [10], in general coordination requires communication. Another important part of coordination is having the agents decide what to do. Since [2], the process of deciding what to do is considered to break down into two parts — deciding what goals to achieve, what [2] calls *deliberation*, and then deciding how those goals might best be achieved, which is usually described as *planning*. In this paper we are interested in the planning part of the process. We assume the existence of a set of goals to be achieved, in a form such as a set of joint intentions [8].

We are also greatly concerned with communication. Much recent work on agent communication uses argumentation-based dialogue [12], and the long term goal of our work is to extend existing work on multiagent planning by developing models by which a team of agents can, in the course of an argumentation-based dialogue — by which we mean a process during which agents put forward suggested partial plans backed by reasons, as in [18] — develop a plan for the team. We want this to be done in a way that respects the non-deterministic nature of the world, and which yields efficient implementation. This paper takes several steps towards this goal.

In particular, this paper gives a mechanism, albeit a centralised mechanism, by which a multiagent team can construct plans that take into account the need to communicate to ensure that the plan is executed correctly [13]. By developing a representation language that is an extension of languages used in non-deterministic planning, our approach can make use of new techniques from model-checking to provide efficient implementations. The extension incorporates the elements necessary to take multiple agents, and the necessary communication, into account. The use of a symbolic model makes it possible to turn the plan construction process into an argumentation-based dialogue in the future.

Building our approach on top of work in planning has advantages beyond ease and efficiency of implementation. By appropriating the underlying formal models, we can easily acquire suitable formal guarantees for the planning model that we construct. It is straightforward, for example, to show that given an adequate description of the world, any plan that our planning process will construct is both a feasible and, in a specific sense an optimal, way to achieve the goals of the plan.

Cite as: A Model for Integrating Dialogue and the Execution of Joint Plans, Yuqing Tang, Timothy J. Norman, Simon Parsons, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 883–890

2. REPRESENTATION LANGUAGE

We use a state-space model as a basis for our formalisation. This model is an adaptation of a model commonly used in non-deterministic planning [6]. *States* are objects that capture some aspect of a system, and *actions* are transitions between states. States and actions together define a *state-space*. When action effects are non-deterministic [6] then what one seeks for any state-space is a *policy*: i.e. a state-action mapping that specifies which actions one should take in a given state. We define a non-deterministic domain to be a tuple $\mathcal{M} = \langle \mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ where:

- $\mathcal{P} = \mathcal{P}_S \cup \mathcal{P}_A$ is a finite set of propositions;
- $\mathcal{S} \subseteq 2^{\mathcal{P}_S}$ is the set of all possible states;
- $\mathcal{A} \subseteq 2^{\mathcal{P}_A}$ is the finite set of actions; and
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the state-transition relation.

A propositional language $\mathcal{L}$ with quantification extension can be defined by allowing standard connectives $\wedge, \vee, \rightarrow, \neg$ and quantifiers $\exists, \forall$ over the proposition variables. The resulting language is a logic of quantified boolean formulae (QBF) [3]. A *symbol renaming operation*, which we use below, can be defined on $\mathcal{L}$, denoted by $\mathcal{L}[\mathcal{P}/\mathcal{P}']$, which means that a new language is obtained by substituting the symbols of $\mathcal{P}$ with the symbols of $\mathcal{P}'$ where $\mathcal{P}'$ contains the same set of propositions as that of $\mathcal{P}$ but using different symbol names (notice that $|\mathcal{P}'| = |\mathcal{P}|$). Similarly for a formula $\xi \in \mathcal{L}$, if $\vec{x}$ is a vector of propositional variables for $\mathcal{P}$, then a variable renaming operation can be defined by $\xi[\vec{x}/\vec{x}']$ which means that all the appearances of variables $\vec{x} = x_1 x_2 \ldots x_n$ are substituted by $\vec{x}' = x_1' x_2' \ldots x_n'$ which is a vector of the corresponding variables or constants in $\mathcal{P}'$. In QBF, propositional variables can be universally and existentially quantified: if $\phi[\vec{x}]$ is a QBF formula with propositional variable vector $\vec{x}$ and x_i is one of its variables, the existential quantification of x_i in ϕ is defined as $\exists x_i \phi[\vec{x}] = \phi[\vec{x}][x_i/FALSE] \vee \phi[\vec{x}][x_i/TRUE]$; the universal quantification of x_i in ϕ is defined as $\forall x_i \phi[\mathbf{x}] = \phi[\vec{x}][x_i/FALSE] \wedge \phi[\vec{x}][x_i/TRUE]$. Here *FALSE* and *TRUE* are two propositional constants representing "true" and "false" in the logic. The introduction of quantification doesn't increase the expressive power of propositional logic but allows us to write concise expressions whose quantification-free versions have exponential sizes [9].

Based on the two disjoint sets of propositions, $\mathcal{P}_S$ and $\mathcal{P}_A$, two sub-languages $\mathcal{L}_S$ and $\mathcal{L}_A$ for states and actions can be defined respectively. A state $s = \{p_1, p_2, \ldots, p_k\}$, $s \subseteq \mathcal{P}_S$, means that the propositions $p_1, p_2, \ldots, p_k$ are true in state s and all other propositions in $\mathcal{P}_S$ are false — we therefore make some form of closed-world assumption. In other words, each state s is explicitly encoded by a conjunction composed of all proposition symbols in $\mathcal{P}_S$ in either positive or negative form

$$\psi = \bigwedge_{p_i \in s} p_i \wedge \bigwedge_{p_j \notin s \text{ and } s \in \mathcal{P}_S} \neg p_j$$

We denote that a formula γ is true in state s by $s \models \gamma$. Then a set of states can be characterized by a formula $\gamma \in \mathcal{L}_S$, with the set denoted by $S(\gamma)$, where $S(\gamma) = \{s | s \models \gamma\}$.[1] Actions are encoded in a similar way to states. Action $a = \{p_1, p_2, \ldots, p_l\}$, $a \subseteq \mathcal{P}_A$ means that propositions $p_1, \ldots, p_l$ are true and all other formula in $\mathcal{P}_A$ are false. We denote that a formula α is true in an action a by $a \models \alpha$, and a set of actions can be characterized by a formula $\alpha \in \mathcal{L}_A$ with the set denoted by $A(\alpha) = \{a | a \models \alpha\}$. Given a set of states S, and a set of actions A, the corresponding formulae in $\mathcal{L}$ are denoted by $\xi(S)$ and $\xi(A)$ respectively. With these notions we can have a mapping between the set operations on states and the boolean operations on formulae as shown in Table 1 when X_1 and X_2 are interpreted as two sets of states. Similarly for actions, we can have the same operation mapping in Table 1 when X_1 and X_2 are interpreted as two sets of actions.

With states and actions defined, the state-transition relationship can then be specified by a set SR of triples: $SR = \{\langle \gamma, \alpha, \gamma' \rangle\}$ where $\gamma, \gamma' \in \mathcal{L}_S$ and $a \in \mathcal{L}_A$. Each triple $\langle \gamma, \alpha, \gamma' \rangle$ corresponds to a transition $R_{\langle \gamma, a, \gamma' \rangle} = \{\langle s, a, s' \rangle | s \models \gamma, a \models \alpha, s' \models \gamma\}$, and together:

$$\mathcal{R}_{SR} = \bigcup_{\langle \gamma, \alpha, \gamma' \rangle \in SR} R_{\langle \gamma, \alpha, \gamma' \rangle}.$$

Using the renaming operation, we can extend the state and action language $\mathcal{L} = \mathcal{L}_S \cup \mathcal{L}_A$ to be $\mathcal{L} = \mathcal{L}_S \cup \mathcal{L}_A \cup \mathcal{L}_{S'}$ where $\mathcal{L}_{S'} = \mathcal{L}_S[\mathcal{P}/\mathcal{P}']$. $\mathcal{P}_S$ is for the current state, $\mathcal{P}_A$ is for the action, and $\mathcal{P}_{S'}$ is for the next state in the representation of a state transition. Now a triple $\langle \gamma, \alpha, \gamma' \rangle$ can be rewritten by one formula in $\mathcal{L}$ as $\gamma \wedge \alpha \wedge \gamma'$ where $\gamma \in \mathcal{L}_S$, $\alpha \in \mathcal{L}_A$ and $\gamma' \in \mathcal{L}_{S'}$. Correspondingly, a state transition $r = \langle s, a, s' \rangle$ is said to satisfy a formula $\delta = \gamma \wedge \alpha \wedge \gamma'$, denoted by $r \models \delta$, if $s \models \gamma$, and $a \models a$, and $s' \models \gamma'$. A set of state transitions R can be characterized by a formulae $\delta = \xi(R)$ and the corresponding set of state transitions $R(\delta) = \{r | r \models \delta\}$. We can capture the meaning of δ more easily if we expand δ into a disjunction, $\delta = \bigvee_i (\gamma_i \wedge \alpha_i \wedge \gamma_i')$, in which each state transition is explicitly encoded as a conjunction. It conforms to the mapping between the set operations on state transitions and boolean operations on the formulae in Table 1 by interpreting X_1 and X_2 with two sets of state transitions.

Set operator	QBF operator
$X_1 \cap X_2$	$\xi(X_1) \wedge \xi(X_2)$
$X_1 \cup X_2$	$\xi(X_1) \vee \xi(X_2)$
$X_1 \setminus X_2$	$\xi(X_1) \wedge \neg\xi(X_2)$
$x \in X$	$\xi(x) \rightarrow \xi(X)$
$X_1 \subseteq X_2$	$\xi(X_1) \rightarrow \xi(X_2)$

Table 1: The mapping between set operators and QBF operators

3. POLICIES

The state-space model described above gives us a way of describing the world in which an agent finds itself, and the actions it can undertake. We can then turn to considering what the output of the planning process will be. We call this output a *policy*, and we consider it to simply be a set of state-action pairs,

$$\pi = \{\langle s_i, a_i \rangle\}$$

where $s_i \in \mathcal{S}$ and $a_i \in \mathcal{A}(s)$ with

$$\mathcal{A}(s) = \{a | \exists \langle s, a, s' \rangle \in \mathcal{R}\}$$

that is the set of actions that are applicable in s. A policy π is a *deterministic policy*, if for a given state s, there is no more than one action is specified by π, otherwise it is a *non-deterministic policy*. What we are calling a policy is the state-action table used in [6]. It is also related to what the literature on MDPs calls a policy [1], but we allow a policy to only specify actions for a subset of all possible states.

A policy can be specified by a set of pairs composed of a formula, $\gamma \in \mathcal{L}_S$, and an action, $\alpha \in \mathcal{L}_A$: $SA = \{\langle \gamma, \alpha \rangle\}$. Each pair $\langle \gamma, \alpha \rangle$ corresponds to a policy segment: $\pi_{\langle \gamma, \alpha \rangle} = \{\langle s, a \rangle | s \models \gamma \text{ and } a \models \alpha\}$, and together

$$\pi_{SA} = \bigcup_{\langle \gamma, \alpha \rangle \in SA} \pi_{\langle \gamma, \alpha \rangle}$$

A state-action pair $\langle s, a \rangle$ is said to satisfy a formula of the form $\gamma \wedge \alpha$ where $\gamma \in \mathcal{L}_S$ and $\alpha \in \mathcal{L}_A$, denoted by $\langle s, a \rangle \models \gamma \wedge \alpha$. We can characterize a set π of state-action pairs, namely a policy, by a formula of the form $\tau = \bigvee_i \gamma_i \wedge \alpha_i$ and its equivalents. We denote this by $\pi(\tau) = \{\langle s, a \rangle | \langle s, a \rangle \models \tau\}$. In the same way that we represent state transitions as propositions, we can have a propositional representation $\xi(SA)$ for a set SA of state-action pairs, and the mapping between the set operations on policies and boolean operations on the formulae given in Table 1 applies if we interpret X_1 and X_2 as two sets of state-action pairs. We can represent the constraint $\mathcal{A}(s)$ by a formula in $\mathcal{L}$: $\xi(\mathcal{A}(S(\gamma)) = \exists \vec{x}' \xi(R(S, A, S')) \wedge \gamma$ where

[1] Note that $S(p_1 \wedge p_2 \wedge \ldots \wedge p_k) \neq \{s\}$ where $s = \{p_1, p_2, \ldots, p_k\}$ because S_γ doesn't make the closed world assumption; that is, we don't assume that the unspecified propositions are false when using a formula $\gamma \in \mathcal{L}$ to specify the set of states $S(\gamma)$.

$\vec{x}'$ is the vector of variables for S' and $\xi(R(S,A,S'))$ is the formula representation of the state transition relation in the system. Then we can conjoin the formula $\xi(\mathcal{A}(S(\gamma)))$ to each policy expression of the form $\gamma \wedge \alpha$ to be $\gamma \wedge \alpha \wedge \xi(\mathcal{A}(S(\gamma)))$. For simplicity, we will omit the formula component $\xi(\mathcal{A}(S(\gamma)))$ in the representation of policy below.

The space of all policies is denoted by Π. The set of states in a policy π is $S_\pi = \{s|\langle s,a\rangle \in \pi\}$. Adapting from [6], we have the following definition:

Definition 1. An *execution structure* induced by the policy π from a set of initial states I is a directed graph $\Sigma_\pi(I) = (V_\pi, E_\pi)$ which can be recursively defined as

- if $s \in I$, then $s \in V_\pi$, and
- if $s \in V_\pi$ and there exists a state-action pair $\langle s,a\rangle \in \pi$ such that $\langle s,a,s'\rangle \in \mathcal{R}$, then $s' \in V_\pi$ and $a : \langle s,s'\rangle \in E_\pi$ where the action a is the label of the edge.

Definition 2. An *execution path* of a policy π from a set of states I is a possibly infinite sequence $s_0, s_1, s_2, \ldots$ of states in the execution structure $\Sigma_\pi(I) = \langle V_\pi, E_\pi\rangle$ such that for all states s_i in the sequence:

- either s_i is the last state of the sequence, in which case s_i is a *terminal state* of $\Sigma_\pi(I)$, or
- $\langle s_i, s_{i+1}\rangle \in E_\pi$.

A state s' is said to be *reachable* from s in the execution structure Σ_π if there is a path from s to s' in Σ_π. Σ_π is an *acyclic execution* iff all its execution paths are finite.

These ideas then give us a way to classify policies:

Definition 3. Given a set of initial states I and a set of goal states G for a nondeterministic domain $\mathcal{M} = \langle \mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{R}\rangle$, let π be a policy for $\mathcal{M}$ with execution structure $\Sigma_\pi(I)$, then

- π is a *weak solution* to achieve G iff for any state $s_0 \in I$ there is some terminal state s' of $\Sigma_\pi(I)$ such that $s' \in G$ and it is reachable from s_0;
- π is a *strong solution* to achieve G iff $\Sigma_\pi(I)$ is acyclic and all terminal states of $\Sigma_\pi(I)$ are also in G;
- π is a *strong cyclic solution* to achieve G iff from any state s_0 in $\Sigma_\pi(I)$ some terminal state s is reachable and all the terminate states of $\Sigma_\pi(I)$ are in G.

With a weak solution policy, we have a path to the goal in a finite number of steps, but no guarantee that in a non-deterministic world the goal will be achieved; with a strong solution policy, we have a guarantee that the goal can be achieved in a finite number of steps despite actions being non-deterministic if the state space is acyclic; and with a strong cyclic solution, we are guaranteed that the goal will be achieved even in the face of non-determinism and cycles in the state-space so long as the cycle can be broken non-deterministically.

4. JOINT POLICIES

To describe the behavior of a team, we need to prescribe more structure over the actions available to an agent. We assume that there is a set of n agents labeled by $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ in the system. We call the actions in the set $\mathcal{A}$ *joint actions* of these agents. Each action $a \in \mathcal{A}$ is a tuple of actions of individual agents, so $a = [a_1, \ldots, a_n]$. That is each action $a \in \mathcal{A}$ can be further decomposed into n actions $a_i \in \mathcal{A}_i$ of individual agents T_i. Each $\mathcal{A}_i$ is defined to be a subset $\mathcal{P}_{\mathcal{A}_i}$ of the propositions in $\mathcal{P}_\mathcal{A}$. By overloading the notion, we also denote $a \models a_i$ if agent T_i's action is a_i in a joint action a. In total, we have:

$$\mathcal{A} = \prod_i \mathcal{A}_i$$

Similarly, each state $s \in \mathcal{S}$ is a tuple of states combined from the perception of individual agents, so $s = [s_1, \ldots, s_n]$. That is each state $s \in \mathcal{S}$ can be further decomposed into n states $s_i \in \mathcal{S}_i$ of individual agents T_i. Each $\mathcal{S}_i$ is defined to be a subset $\mathcal{P}_{\mathcal{S}_i}$ of the propositions in $\mathcal{P}_\mathcal{S}$. By overloading this notion, we also denote $s \models s_i$ if agent T_i's perception of a (joint) state s is s_i in.

Overall, we have:

$$\mathcal{S} = \prod_i \mathcal{S}_i$$

Given these ideas, we can generate the set of join(t) and projection operations on an agent T_i's actions, states and state transitions as shown in Tables 2 and 3 respectively. Joint operations combinations the states and actions that concern individual agents into the states and actions that concern a set of agents, while projection operations extract states and actions of individual agents from those of a set of agents.

An additional formula $\beta \in \mathcal{L}$ can be introduced to constrain possible combinations so that $\mathcal{A}(\beta) = \{a \in \mathcal{A} | a \models \beta\}$. For

Representation	Meaning
$joint(a_i)$	$\{a \in \mathcal{A} \mid a \models a_i\}$
$joint(\vec{a})$	$\bigcap_{a_k \in a} joint(a_k)$
$joint(s_i)$	$\{s \in \mathcal{S} \mid s \models s_i\}$
$joint(\vec{s})$	$\bigcap_{s_k \in \vec{s}} joint(s_k)$
$joint(R_i)$	$\{\langle s,a,s'\rangle \mid \langle s_i,a_i,s_i'\rangle \in R_i$, and $s \in joint(s_i), a \in joint(a_i), s' \in joint(s_i')\rangle\}$
$joint(\pi_i)$	$\{\langle s,a\rangle \mid \langle s_i,a_i\rangle \in \pi_i$, and $s \in joint(s_i), a \in joint(a_i)\rangle\}$
$joint(S_i)$	$\bigcup_{s_i \in S_i} joint(s_i)$
$joint(\{\langle s_i,s_i'\rangle\})$	$\{\langle joint(s_i), joint(s_i')\rangle\}$
$joint(\Sigma_{\pi_i})$	$\langle joint(V_{\pi_i}), joint(E_{\pi_i})\rangle$
$joint(\{R_i\})$	$\bigcap_i joint(R_i)$
$joint(\{\pi_i\})$	$\bigcap_i joint(\pi_i)$

Table 2: Joint operations

Representation	Meaning
$proj_i(a)$	$\{a_i \in \mathcal{A}_i \mid a \models a_i\}$
$proj_i(s)$	$\{s_i \in \mathcal{S}_i \mid s \models s_i\}$
$proj_i(R)$	$\{\langle s_i,a_i,s_i'\rangle \mid \langle s,a,s'\rangle \in R$, and $s_i \in proj_i(s), a_i \in proj_i(a), s' \in proj_i(s')\rangle\}$
$proj_i(\pi)$	$\{\langle s_i,a_i\rangle \mid \langle s,a\rangle \in \pi$, and $s_i \in proj_i(s), a_i \in proj_i(a)\rangle\}$
$proj_i(S)$	$\bigcup_{s_k \in S} proj_i(s_k)$
$proj_i(\{\langle s,s'\rangle\})$	$\bigcup_{s_k \in S} \{\langle proj_i(s), proj_i(s')\rangle\}$
$proj_i(\Sigma_{\pi_i})$	$\langle proj_i(V_{\pi_i}), proj_i(E_{\pi_i})\rangle$

Table 3: Projection operations

example, this constraint:

$$\beta = \bigwedge_{i=1}^{n} \bigwedge_{j \neq i} a_i \rightarrow \tau_j$$

where τ_j is a special symbol for an empty action, captures a situation in which agents are not allowed to carry out actions concurrently. The corresponding constrained joint state transition relationship is:

$$\begin{aligned} joint(\{\mathcal{R}_i\}, \beta) &= \{\langle s, a, s' \rangle \mid \langle s, a, s' \rangle \in joint(\{\mathcal{R}_i\}), \\ &\qquad \text{and } a \models \beta, s \models \beta \rangle\} \end{aligned}$$

and the corresponding constrained joint policy is:

$$\begin{aligned} joint(\{\pi_i\}, \beta) &= \{\langle s, a, s' \rangle | \langle s, a \rangle \in joint(\{\pi_i\}), \\ &\qquad \text{and } a \models \beta, s \models \beta \rangle\} \end{aligned}$$

It is should be noted that in practice we need to be careful exactly how we specify formulae like the constraint β since they can adversely affect the complexity of reducing the the formulae into a form in which they can be fed into the BDD implementation. We will discuss this briefly in Section 8.

5. POLICY AND COMMUNICATION

At this point we have a language that is sufficiently rich to construct plans that just involve the physical actions that agents carry out. However, we want to create plans that include communications that permit the necessary sharing of information, so we need to add a dialogue model to the model we already have. As the basis of the dialogue model, we will use the same kind of state space model as we use for the world model. To distinguish the two state transition models, we will denote these two models and their elements with subscripts. We write $_{|D}$ to denote elements of the dialogue model, for example, $M_{|D}$ denotes the state transition model for a dialogue and $\mathcal{S}_{|D}$ denotes the states of a dialogue. We write $_{|W}$ to denote elements of the world model, for example, $M_{|W}$ denotes the external world model and $\mathcal{S}_{|W}$ the states of the world. However, when the state transition model is obvious from the context, we will omit the subscripts.

As before, we assume that, in the dialogue, there is a set of n agents labeled $T_1, T_2, \ldots, T_n$ where each agent T_i has a model of the world $\mathcal{M}_{i|W} = \langle \mathcal{P}_{i|W}, \mathcal{S}_{i|W}, \mathcal{A}_{i|W}, \mathcal{R}_{i|W} \rangle$ and for which it has a policy $\pi_{i|W} = \{\langle s_i, a_i \rangle\}$. Given this, a dialogue model is then a state transition system $\mathcal{M}_{|D} = \langle \mathcal{P}_{|D}, \mathcal{S}_{|D}, \mathcal{A}_{|D}, \mathcal{R}_{|D} \rangle$ for which there is a policy for conducting dialogues $\pi_{|D}$. The dialogue language $\mathcal{P}_{|D}$ contains elements from language $\mathcal{P}_{i|W}$ that individual agents use to describe the world, along with auxiliary language elements such as a proposition to mark the differences between two world states. The dialogue information is induced from $\mathcal{P}_D$. The set of dialogue acts $\mathcal{A}_{|D}$ are those available to the agents. How these dialogues change the information state will be specified by the dialogue state transition relationship of these dialogue acts: $\mathcal{R}_{|D} \subseteq \mathcal{S}_{|D} \times \mathcal{A}_{|D} \times \mathcal{S}_{|D}$. Depending on the specific dialogue, we may distinguish a set of initial dialogue states $I_{|D} \subseteq \mathcal{S}_{|D}$ and a set of goal dialogue states $G_{|D} \subseteq \mathcal{S}_{|D}$ (see [14] for an example).

Definition 4. Agent T_i's behavior model is a joint model of its external world $\langle \mathcal{M}_{i|W}, \pi_{i|W} \rangle$ and its dialogue model $\langle \mathcal{M}_i, \pi_i \rangle = \langle \mathcal{M}_{i|D}, \pi_{i|D} \rangle$ defined as:

$$\langle \mathcal{M}_i, \pi_i \rangle = \langle joint(\mathcal{M}_{i|W}, \mathcal{M}_{i|D}), joint(\pi_{i|W}, \pi_{i|D}) \rangle.$$

The whole system behavior model is $joint_{\{T_i\}}(\{\langle \mathcal{M}_i, \pi_i \rangle\})$.

As before, a policy for a dialogue, $\pi_{|D} = \{\langle s_{|D}, a_{|D} \rangle\}$, specifies what dialogue action should be taken in a given dialogue state

$$\begin{aligned} EXEC(s,a) &= \{s' | \langle s, a, s' \rangle \in \mathcal{R}\} \\ StatesOf(\pi) &= \{s | \langle s, a \rangle \in \pi\} \\ GetAction(s,\pi) &= \{a | \langle s, a \rangle \in \pi\} \\ ComputeWeakPreImage(S) &= \{\langle s, a \rangle | Exec(s,a) \wedge S \neq \emptyset\} \\ ComputeStrongPreImage(S) &= \{\langle s, a \rangle | \emptyset \neq Exec(s,a) \subseteq S\} \\ ComputeNextImage(S) &= \{s' | Exec(s,a) \wedge S\} \\ PrunStates(\pi, S) &= \{\langle s, a \rangle \in \pi | s \notin S\} \end{aligned}$$

Figure 1: Operations on transition relations and policies

Set representation	QBF implementation
$EXEC(s,a)$	$\xi(s) \wedge \xi(a) \wedge \xi(\mathcal{R})[\vec{x'}/\vec{x}]$
$StatesOf(\pi)$	$\exists \vec{a} \xi(\pi)$
$GetAction(s,\pi)$	$\xi(s) \wedge \xi(\pi)$
$ComputeWeakPreImage(S)$	$\exists \vec{x'} \xi(S)[\vec{x}/\vec{x'}] \wedge \xi(\mathcal{R})$
$ComputeStrongPreImage(S)$	$\forall \vec{x'}(\xi(\mathcal{R}) \rightarrow \xi(S)[\vec{x}/\vec{x'}]) \wedge \exists \vec{x'} \xi(\mathcal{R})$
$ComputeNextImage(S)$	$\exists \vec{x} \xi(S) \wedge \xi(\mathcal{R})$
$PrunStates(\pi, S)$	$\xi(\pi) \wedge \neg \xi(S)$

Table 4: The mapping between set representation and QBF implementation of some transition relation and policy functions

to reach the goal states $G_{|D}$ from the initial states $I_{|D}$ at the least expected cost. To distinguish such policies from the policies that govern an agent's actions in the world, we call the policies that govern an agent's actions in a dialogue a *conversation policy* and a policy that governs an agent's actions in the world a *world policy*.

Before we go on to give the description of the algorithm for executing world and conversation policies, we need to take a look at some properties that capture the interaction between the execution of actions in the world and communication between team members.

Definition 5.

- A state-action pair $\langle s, a \rangle \in \pi_{|W}$ is called *totally autonomous*, if for every agent T_i there is no other $\langle s'_i, a'_i \rangle \in proj_i(\pi_{|W})$ such that $\langle s_i, a_i \rangle \in proj_i(\langle s, a \rangle)$, and $s_i = s'_i$ but $a_i \neq a'_i$. In other words, action-state pairs are totally autonomous if for every agent involved there is no confusion about which action it should take. A team policy $\pi_{|W}$ is called totally autonomous if all its constituent joint state-action pairs are totally autonomous. In this case, an individual agent can choose what it should do based only on local information about the world.

- A state-action pair $\langle s, a \rangle \in \pi_{|W}$ is called *state communication sufficient*, if there is no other state-action pair $\langle s', a' \rangle \in \pi_{|W}$ such that $s = s'$ but $a \neq a'$. A team policy $\pi_{|W}$ is called a *state communication sufficient* if all its joint state-action pairs are state communication sufficient (making it equivalent to a deterministic joint policy). In this case, each individual agent can choose correctly what it should do based only on knowledge of the global state.

- A state-action pair $\langle s, a \rangle \in \pi_{|W}$ is called a *state and action communication sufficient*, if there is another $\langle s', a' \rangle \in \pi_{|W}$ such that $\langle s', a' \rangle \in \pi_{|W}$ such that $s = s'$ but $a \neq a'$. A team policy $\pi_{|W}$ is called *state and action communication sufficient* if some of its joint state-action pairs are state and action communication sufficient (making it equivalent to a non-

Algorithm 5.1 Execution of world and conversation policies

procedure *ExecPolicy*$(\mathcal{M}_{|W}, \pi_{|W}, \mathcal{M}_{|D}, \pi_{|D})$ {
(1) $\mathcal{M}_{|W}$: Joint external world model ,
(2) $\pi_{|W}$: Joint external world policy ,
(3) $\mathcal{M}_{|D}$: Joint dialogue model ,
(4) $\pi_{|D}$: Joint dialogue policy }
$\mathcal{M}_{i|W} \leftarrow proj_i(\mathcal{M}_{|W})$
$\pi_{i|W} \leftarrow proj_i(\pi_{|W})$
$\mathcal{M}_{i|D} \leftarrow proj_i(\mathcal{M}_{|D})$
$\pi_{i|D} \leftarrow proj_i(\pi_{|D})$
$s_{i|W} \leftarrow SenseCurrentState()$
$s_{i|D} \leftarrow ReceiveCommunication() \wedge ComputeDialState(s_{i|W} \wedge \pi_{i|W})$
while $s_{i|W} \in StatesOf(\pi_{i|W}) \vee s_{i|D} \in StatesOf(\pi_{i|D})$ **do**
  if $|joint(GetAction(s_{i|W}, \pi_{i|W}))| > 1$ **then**
    $WorldSA \leftarrow ComputeJointSA(s_{i|D})$
    if $|WorldSA| = 1$ **then**
      $a_{i|W} \leftarrow proj_i(GetAction(WorldSA))$
      $Execute(a_{i|W})$
    else
      $a_{i|D} \leftarrow GetAction(s_{i|D}, \pi_{i|D})$
      if $a_{i|D} \neq \emptyset$ **then**
        $Execute(a_{i|D})$ {Communicate to resolve the ambiguity about which action to select}
      else
        $WorldSA \leftarrow RetrieveExternalDecision(WolrdSA)$ {Communication cannot help, ask for external decision}
        $s_{i|D} \leftarrow ComputeDialState(WorldSA)$ {Update the external decision into the information state}
      end if
    end if
  else
    $a_{i|W} \leftarrow GetAction(s_{i|W}, \pi_{i|W})$
    $Execute(a_{i|W})$
  end if
  $s_{i|W} \leftarrow SenseCurrentState()$
  $s_{i|D} \leftarrow ReceiveCommunication() \wedge ComputeDialState(s_{i|W} \wedge \pi_{i|W})$
end while
end procedure

deterministic joint policy). In this case, individual agents need to decide what to do during policy execution by picking among the set of all possible actions given by the joint policy, and need to communicate with one another to come to a decision.

- A policy π is called a *out of usage* in a state s if there is no $\langle s, a\rangle \in \pi$. In this case, agents need to replan.

A procedure to execute a combined world policy and conversation policy is given in Algorithm 5.1. It is adapted from the corresponding procedure in [6] and with the addition of steps to execute the conversation policy. It uses the transition operations defined in Figure 1 and assumes that the these operators, as well as the *joint* and $proj_i$ operations, operate on the world and the dialogue transition model according to the symbols $|W$ and $|D$ respectively. *ComputeDialState*, *ComputeJointSA* and *RetrieveExternal Decision* are application dependent, and define how the dialogue is related to the external world model, as in Section 6. In essence the procedure steps through the world policy, executing the steps of a communication policy when communication is required.

6. GENERATING POLICIES

Given the general model of dialogue defined in Section 5, we can define a specific conversation policy which will ensure that the correct information is exchanged during world policy execution. In this section we describe an algorithm for generating policies that combine world policies and conversation policies.

We start by assuming that each agent T_i maintains a model of the external world $\mathcal{M}_{i|W}$ and its finite propositional language $\mathcal{P}_{i|W}$ will depend on the application. T_i's dialogue model $\mathcal{M}_{i|D}$ is based on a propositional language

$$\mathcal{P}_{i,\mathcal{S}|D} = \mathcal{P}_{\mathcal{S}|W} \cup \mathcal{P}_{\mathcal{A}|W} \cup \mathcal{P}_{AL} \cup \mathcal{P}_{CM}$$

where $\mathcal{P}_{AL}$ contains a boolean variable for every variable in $\mathcal{P}_{\mathcal{S}|W} \cup \mathcal{P}_{\mathcal{A}|W}$ to indicate its validity in dialogue state, $\mathcal{P}_{CM}$ contains a boolean variable for every variable in $\mathcal{P}_{\mathcal{S}_i|W} \cup \mathcal{P}_{\mathcal{A}_i|W}$ of the agent T_i's (the information T_i can effectively known) to indicate whether its value has been communicated in dialogue state, $j = 1 \ldots N$ and N is the number of agents in the system, and

$$\mathcal{P}_{i,\mathcal{A}|D} = \{tell(i, j, x_k, v)\}$$

where $j = 1 \ldots N$, $x_k \in \mathcal{P}_{i|W}$ and $v = \{0, 1\}$. $tell(i, j, x_k, v)$ means that T_i tells T_j that the boolean variable x_k representing some bit of the state and action information is in the value v. We denote variables in $\mathcal{P}_{\mathcal{S}_i|D}$ by $x_{i,j,k}$, $lx_{i,j,k}$ and $cx_{i,j,k}$ for T_i's information about T_j on state variable k, about its validity and whether it has been communicated to T_j, and those in $\mathcal{P}_{\mathcal{A}_i|D}$ by $y_{i,j,l}$, $ly_{i,j,l}$ and $cy_{i,j,l}$ for T_i's information about T_j on action variable l, its validity and whether it has been communicated to T_j where $j = 1, \ldots, N$, $k = 1 \ldots K = |\mathcal{P}_{\mathcal{S}_j}|$ and $l = 1 \ldots L = |\mathcal{P}_{\mathcal{A}_j}|$. In total, we have $3N * N * (K + L)$ variables for the dialogue system of the whole team[2].

The mapping between agent T_i's current state and its information state in the dialogue can be described by a β (connection) conditions. For example,

$$\beta(s_{i|W}, s_{i|D}) = \bigwedge_{j=1}^{N} \bigwedge_{k=1}^{K} [vx_{i,j,k} \rightarrow (x_{i,j,k} \leftrightarrow x'_{i,j,k})]$$

where $x_{i,i,k} \in \mathcal{P}_{\mathcal{S}_i|W}$ and $x'_{i,i,k} \in \mathcal{P}_{i|D}$. More complex mappings can be defined using representation languages such as a restricted linear time logic or a computation tree logic, representations that are used in the symbolic model checking literature [4].

Similarly, there is a mapping between agent T_i's next action decision and its information state in the dialogue. This mapping can be described by the β condition, for example,

$$\beta(a_{i|W}, s_{i|D}) = \bigwedge_{j=1}^{N} \bigwedge_{l=1}^{L} [vx_{i,j,l} \rightarrow (y_{i,j,l} \leftrightarrow y'_{i,j,l})]$$

where $y_{i,i,l} \in \mathcal{P}_{\mathcal{A}_i|W}$ and $y'_{i,i,l} \in \mathcal{P}_{i|D}$. Please notice that the above two β conditions depends on the validity variables in the dialogue information states. These validity variables will be initialized by

$$\nu_0(s_{i|D}) = \bigwedge_{k=1}^{K} [vx_{i,i,k}] \wedge \bigwedge_{l=1}^{L} [vy_{i,i,l}].$$

As shown in the dialogue state transitions below, the value of these validity variables will also been changed by the dialogue acts.

Using the mappings of states and actions, we can compute a set of initial dialogue states — those that exist before taking into account the effects of any communication — from the fragments of world policy that individual agents possess:

$$ComputeDialState(WorldSA) = \exists_{\vec{x} \in \mathcal{P}_{|W}} [WorldSA \wedge \bigwedge_i^N [\beta(s_{i|w}, s_{i,d}) \wedge \beta(a_{i|w}, s_{i|D}) \wedge \nu_0(s_{i|D})]],$$

[2] This can be improved by encoding the indices of $x_{i,j,k}$ and $y_{i,j,l}$ with $logN + logN + logK + logL$ boolean variables, and maintain the information using a relation to map these indices to the values they correspond to.

Algorithm 6.1 Dialogue goal computation

1: **function** *ComputeDialGoal*(*NewWorldSA*) {
(1) *NewWorldSA*: A new external policy segment,
(2) *IJMAP*: The global variable holding the dialogue states to world joint states mapping
}
2: Set *ComputeNextImage* to use R_D
3: $NewDialS \leftarrow ComputeDialState(NewWorldSA)$
4: **repeat**
5: $DialS \leftarrow NewDialS$
6: $NewDialS \leftarrow ComputeNextImage(DialS)$
7: $DJMAP \leftarrow ComputeDJMAP(NewDialS, NewWorldSA)$
8: **until** $DialS = NewDialS \vee GoodExe(DJMAP)$
9: **return** *NewDialS*
10: **end function**

We can also compute a mapping, denoted by *DJMAP*, between the dialogue states and the corresponding fragments of world policy:

$$ComputeDJMAP(DialS, NewWorldSA) = \left[DialS \wedge NewWorldSA \wedge \bigwedge_i^N [\beta(s_{i|w}, s_{i,d}) \wedge \beta(a_{i|w}, s_{i|D})]\right]$$

and conversely we ca compute a joint external world state and its policy action out of a dialogue state using the *DJMAP* mapping:

$$ComputeJointSA(s_{i|D}) = \exists_{\vec{x} \in \mathcal{P}_{|D} \cup \bigcup_j \mathcal{P}_{j \neq i|W}} \left[s_{i|D} \wedge DJMAP\right]$$

The set of dialogue state transitions associated with $\mathcal{A}_{i|D}$ is:

$$\begin{aligned}\mathcal{R}_{i|D} = \{&\langle x_{i,i,k} = v \wedge cx_{i,j,k} = 0, tell(i,j,x_k,v),\\ &x_{j,i,k} = v \wedge cx_{i,j,k} = 1 \wedge vx_{i,j,k} = 1\rangle,\\ &\langle y_{i,i,l} = v \wedge cy_{i,j,l} = 0, tell(i,j,y_l,v),\\ &y_{j,i,k} = v \wedge cy_{i,j,k} = 1 \wedge vy_{i,j,k} = 1\rangle\}\end{aligned}$$

For now, we assume that the execution of communication actions will be much faster than that of actions in the external world — for example assuming that communication is carried out on a high speed network while external actions are carried out under the usual limitations of the physical world. This assumptions enables us to be sure that agents can always carry out the necessary communication before performing the external world actions that required the communication. This assumption can be relaxed, however, by adding variables that capture temporal information. This consideration is a topic for our future research.

By adding communication conditions to the nondeterministic planning algorithms proposed in [6], we obtain the communication-aware policy planning algorithm of Algorithm 6.2. In the algorithm, *I* will be set to the initial states which the team of agents will start with, and *G* will be set to the goal states which the team is intended to end up with, and *ComputePreImage* can be either *ComputeWeakPreImage* or *ComputeStrongPreImage* defined in Figure 1 in Section 2, corresponding to the weak and strong solution concepts respectively. Strong acyclic solutions can be similarly constructed following the approaches used in [6] but omitted here for lack of space.

As for dialogue policy synthesis, the set of initial dialogue states can be computed using *ComputeDialState* from the set of the new world policy segments, *NewSA*, and the set of dialogue goal states can be computed using the function *ComputeDialGoal* are defined in Algorithm 6.1 where the function good for execution is defined as:

$$\begin{aligned}GoodExec(DJMAP) = [&IJMAP \wedge IJMAP[\vec{x},\vec{y}/\vec{x}',\vec{y}'] \wedge\\ &\bigwedge_i^N \left[(\neg \bigwedge_{x_i \in \mathcal{P}_{|W}} (x_i \leftrightarrow x_i')) \wedge (\bigwedge_{y_i \in \mathcal{P}_{|D}} (y_i \leftrightarrow y_i'))\right]\\ &\leftrightarrow FALSE\end{aligned}$$

which means that the *DJMAP* has evolved into a mapping table in which different joint world policy items won't be mapped into one dialogue state. The *DJMAP* table with this *GoodExec* property can be used by every agent with *ComputeJointSA* to obtain an unique external state-action pair. However, *ComputeDialGoal* may return a goal dialogue state without satisfying *GoodExec* property. This means that the external policy is non-deterministic, and need an external decision maker to choose an action.

Algorithm 6.2 World policy generation

1: **function** *ComputeWorldPolicy*(*I*, *G*, *ComputePreImage*) {
(1) *I*: Initial states,
(2) *G*: Goal states,
(3) *ComputePreImage* : A pre-image function }
2: $DJMAP \leftarrow \emptyset$
3: $OldSA \leftarrow Fail$
4: $SA \leftarrow \emptyset$
5: $SA_D \leftarrow \emptyset$
6: **while** $OldSA \neq SA \wedge I \not\subseteq (G \cup StatesOf(SA))$ **do**
7: $PreImage \leftarrow ComputePreImage(G \cup StatesOf(SA))$
8: $NewSA \leftarrow PruneStates(PreImage, G \cup StatesOf(SA))$
9: **if** $\exists i | joint(GetAction(proj_i(NewSA)))| > 1$ **then**
10: $I_D \leftarrow ComputeDialState(NewSA)$
11: $G_D \leftarrow ComputeDialGoal(NewSA)$
12: $NewSA_D \leftarrow ComputePolicy(I_D, G_D, \mathcal{R}_D, ComputePreImage)$
13: **if** $NewSA_D = \emptyset$ **then**
14: **return** *Fail*
15: **end if**
16: $SA_D \leftarrow SA_D \cup NewSA_D$
17: **end if**
18: $OldSA \leftarrow SA \cup NewSA$
19: **end while**
20: **if** $I \subseteq (G \cup StatesOf(SA))$ **then**
21: **return** $\langle SA, SA_D \rangle$
22: **else**
23: **return** *Fail*
24: **end if**
25: **end function**

Algorithm 6.3 General policy generation

1: **function** *ComputePolicy*(*I*, *G*, *ComputePreImage*) { (1) *I*: Initial states,
(2) *G*: Goal states,
(3) *ComputePreImage* : A pre-image function }
2: $OldSA \leftarrow Fail$
3: $SA \leftarrow \emptyset$
4: **while** $OldSA \neq SA \wedge I \not\subseteq (G \cup StatesOf(SA))$ **do**
5: $PreImage \leftarrow ComputePreImage(G \cup StatesOf(SA))$
6: $SA \leftarrow PruneStates(PreImage, G \cup StatesOf(SA))$
7: $OldSA \leftarrow SA \cup SA$
8: **end while**
9: **if** $I \subseteq (G \cup StatesOf(SA))$ **then**
10: **return** *SA*
11: **else**
12: **return** Fail
13: **end if**
14: **end function**

PROPOSITION 1 (CORRECTNESS). *If Algorithm 6.2 returns a policy π, then π is a weak or a strong solution to achieve the goals G from initial states I. If the algorithm returns FAIL, then there is no weak or strong solution.*

PROOF. The algorithm does a backward breadth first search from the goal states with respect to the *ComputePreImage* being set to weak pre-image or strong pre-image function. There is an additional step of computing dialogue policy to combine information from different agents to determine the current state and additional action decision so that every agent can determine the next action uniquely. The correctness of the policy computation can be found

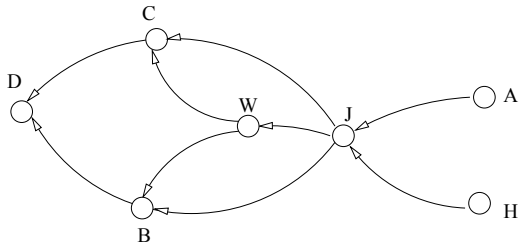

Figure 2: An NGO team task

in work on non-deterministic planning [6]. If the procedure failed, either there is no weak or strong solution to the joint transition model or the dialogue policy synthesis failed. The dialogue policy synthesis is guaranteed to succeed, because in worst case the joint state and additional decision of joint action is fully communicated, and the application of *PruneStates* in Algorithm 6.2 and the way in which the dialogue states and *DJMAP* are constructed in Algorithm 6.1 guarantees that no two dialogue states will be the same in the dialogue policy so we avoid conflicting dialogue action prescriptions. Therefore if the procedure to construct a solution fails, it is because there is no weak or strong solution. □

7. AN EXAMPLE

Consider the following example, based on the example in [5]. Two agents, one representing an NGO (N) and one representing a peace keeping force (F), are working in a conflict zone. The agents (and the organizations they represent) work independently and have different agenda. N is based at A in Figure 2. F is based at point H. N's goal is to reach D to help the villagers there. F's goal is keeping the peace in general in the area, but it also has to protect N while N is carrying out its work. At any time, with some probability, some disruption may flare up at W. If it happens, only F has the surveillence data to know this is happening, and F must go to W to suppress the disturbance. The routes between different points are shown as arcs in Figure 2. N cannot traverse the routes $(J,W),(W,C),(W,B)$, when there is a disturbance at W, and it is only able to traverse (C,D) and (B,D) without harm when it is accompanied by F. N can traverse the rest of the routes independently and F can traverse any route. The goal of the agents is to have N reach D and to have F put down the conflict in W if it happens.

We can formalise this as $\mathcal{P}_{\mathcal{S}_N} = \{l_{N,L}\} \cup \{health\}$, $\mathcal{P}_{\mathcal{S}_F} = \{l_{F,L}\} \cup \{war\}$, $\mathcal{P}_{\mathcal{A}_N} = \{stay_N, move(N,L',L'')\}$, and $\mathcal{P}_{\mathcal{A}_F} = \{move(F,L_1,L_2)$ where $L,L',L'' \in \{A,H,J,W,B,C,D\}$, *conflict* means that there is a disturbance in point W, and *health* means that N is not harmed.

Initially, $I = l_{N,A} \wedge l_{F,H} \wedge health \wedge (conflict \vee \neg conflict)$. The goal $G = l_{N,D} \wedge \neg conflict \wedge health$. The joint transition model $\mathcal{R}$ for the scenarios is as follows

$$\begin{aligned}
&\mathcal{R}_{move} \\
&\quad = \langle l_{F,x} \wedge l_{N,y}, move(F,x,x') \wedge move(N,y,y'), l'_{F,x'} \wedge l'_{N,y'} \rangle \\
&\mathcal{R}_{stay} \\
&\quad = \langle TRUE, stay_N, TRUE \rangle\} \\
&\mathcal{R}_{health} \\
&\quad = \langle TRUE, \neg[move(F,B,D)] \wedge move(N,B,D), \neg health \rangle, \\
&\qquad \langle TRUE, \neg[move(F,C,D)] \wedge move(N,C,D), \neg health \rangle,
\end{aligned}$$

We have additional conditions β_w and $\beta_{agent,route}$:

$$\begin{aligned}
\beta_w &= l_{F,W} \rightarrow \neg conflict \\
\beta_{F,route} &= \bigwedge_{(x,x') \in Route} [l_{F,x} \wedge l'_{F,x'}] \\
\beta_{N,route} &= \bigwedge_{(x,x') \in Route \setminus \{(J,W),(W,C),(W,B)\}} [l_{N,x} \wedge l'_{N,x'}]
\end{aligned}$$

where *Route* is the directed graph of the routes showed in Figure 2. Overall,

$$\mathcal{R} = \mathcal{R}_{action} \wedge \mathcal{R}_{stay} \wedge \mathcal{R}_{health} \wedge \beta_w \wedge \beta_{F,route} \wedge \beta_{N,route}.$$

Algorithm 6.2 will generate the necessary individual world and dialogue policies. Started backward from the set G of goal states, although G does specify only F's location, but the system only allows N to travel to the destination D if it is accompanied by F, and no route is available from D back to W. This indicates that, at the end, F must also be in D. Therefore from the desired goal states, the backward chaining search will trace back to the state where either both F and N are in C or both are in B. Rolling back from these two joint states, if there is conflict in W, F must come from W where it can resolve the conflict; otherwise, F can come from either W or J. As for N, no matter whether it is in C or B, it must come from J. Therefore if there is no conflict at W, the algorithm will force F come directly from J (because *PruneStates* will prune the longer paths). However, when F and N are in both at J (with a conflict at W), the algorithm will produce two valid joint actions: either both going to C or both going to B. Here let's assume one of them, for example F, seeks an external decision to decide the next step, say the result is going to B, then it must communicate the decision with N, so that they can both go to B to guarantee a chance of success. If there is conflict in W, F will go to W to resolve it, while N will reach B. As N and F don't know each other's positions, although they have a valid joint plan, they must communicate with each other so that N knows it will need to stay in B and wait for F to come, and F will know it will need to go to B instead of C. The same kind of communication about positions will be needed for all other locations except at A and B where they can decide by themselves to go to J without needing to communicate with each other.

8. THE BDD IMPLEMENTATION

In the above, we have showed the natural connections between set paradigm on state transitions and its implicit representation using QBF formulae. There is a data structure called a Binary Decision Diagram (BDD) [3] that represents QBF formulae and makes it possible to perform efficient operations over them. A BDD is a rooted directed acyclic graph used to encode the set of truth assignments to a QBF. BDDs guarantee that the basic boolean operations on QBFs can be computed in quadratic time [3, 9] as summarized in in Table 5. The intuition behind this efficiency is that BDD representation is actually a form of minimal description of the information encoded — the BDD for a QBF is actually the minimum automaton that accepts the corresponding set of truth assignments with respect to a specific variable ordering [4], and this minimality can be preserved across the basic boolean operations.

9. CONCLUSIONS

This paper has presented a model of individual and joint action, suitable for describing the behavior of a multiagent team, including communication actions. The model is symbolic, and capable of handling non-deterministic actions. In addition to the model, we

QBF/Set operator	BDD operator	Complexity
$\neg\xi$	$\neg G(\xi)$	$O(\|\|\xi\|\|)$
$\exists x_i(\xi)$	$G(\xi_{x_i=0}) \vee G(\xi_{x_i=1})$	$O(\|\|\xi\|\|^2)$
$\forall x_i(\xi)$	$G(\xi_{x_i=0}) \wedge G(\xi_{x_i=1})$	$O(\|\|\xi\|\|^2)$
$\xi_1 \wedge \xi_2$	$G(\xi_1) \wedge G(\xi_2)$	$O(\|\|\xi_1\|\| \cdot \|\|\xi_2\|\|)$
$\xi_1 \vee \xi_2$	$G(\xi_1) \vee G(\xi_2)$	$O(\|\|\xi_1\|\| \cdot \|\|\xi_2\|\|)$
$\xi_1 \rightarrow \xi_2$	$G(\xi_1) \rightarrow G(\xi_2)$	$O(\|\|\xi_1\|\| \cdot \|\|\xi_2\|\|)$
$\|X\| = 1$	*Sat-one*$(G(\xi(X)))$	$O(\|\vec{x}\|)$

Table 5: The mapping between QBF operators and BDD operators. ξ, ξ_1, ξ_2 are formulae in QBF; $G(\xi), G(\xi_1), G(\xi_2)$ are BDD representations for these formulae; $||\cdot||$ is the number of nodes used in the BDDs.

have provided procedures for creating joint plans, plans that include the communication necessary for plan execution — that is the detection and communication of information relevant to the execution of the plan. We believe this is the first time that this kind of planning model, drawn from the literature of non-deterministic planning, has been combined with a communication model and then applied to multiagent teams.

As discussed by [16], teamwork requires requires the establishment of joint intentions and the determination of which goals to achieve, the creation of a plan, the sharing of knowledge about the environment in which the team is operating, and the ability to monitor plan execution. While we do not claim that what we have described in this paper is a comprehensive model of teamwork — it is much less powerful and comprehensive than Teamcore [17] or Retsina [15], for example — it marks a useful step towards our overall goal of constructing a model of argumentation-based dialogue that can support many of the important aspects of teamwork. In particular, it deals with planning, albeit in a centralised way, the sharing of information,and a limited form of plan monitoring.

One obvious area of future work is moving from a centralised planning process, which just hands every agent a policy that will help the team achieve its goals, to a decentralised process in which agents can engage in a discussion of the best plan. For that we plan to combine our prior work on argumentation-based planning [18], which assumes a simple, deteministic model of actions, with the work we have described here. Another area of future work, which addresses the main area in which our model falls short of a model of teamwork, is to consider the formation of joint intentions. Here there is a rich vein of work to draw on, for instance [7, 11], and we will seek to incorporate this into our model.

Acknowledgments

Research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

10. REFERENCES

[1] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[2] M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4, 1988.

[3] R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surveys*, 24(3):293–318, 1992.

[4] J. R. Burch, E. M. Clarke, D. E. Long, K. L. Mcmillan, and D. L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13:401–424, 1994.

[5] C. Burnett, D. Masato, M. McCallum, T. J. Norman, J. Giampapa, M. J. Kollingbaum, and K. Sycara. Agent support for mission planning under policy constraints. In *Proceedings of the Second Annual Conference of the ITA*, Imperial College, London, 2008.

[6] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1-2):35–84, 2003.

[7] P. Cohen and H. Levesque. Intention is choice with commitment. *Artficial Intelligence*, 42:213–261, 1990.

[8] P. Cohen and H. Levesque. Teamwork. *Nous*, 25(4), 1991.

[9] O. Coudert and J. C. Madre. The implicit set paradigm: a new approach to finite state system verification. *Formal Methods in System Design*, 6(2):133–145, 1995.

[10] M. R. Genesereth, M. L. Ginsberg, and J. S. Rosenschein. Cooperation without communication. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1986.

[11] B. Grosz and S. Kraus. The evolution of sharedplans. In A. Rao and M. Wooldridge, editors, *Foundations and Theories of Rational Agency*. Kluwer, 2003.

[12] S. Parsons and P. McBurney. Argumentation-based dialogues for agent coordination. *Group Decision and Negotiation*, 12(5), 2003.

[13] S. Parsons, S. Poltrock, H. Bowyer, and Y. Tang. Analysis of a recorded team coordination dialogue. In *Proceedings of the Second Annual Conference of the ITA*, Imperial College, London, 2008.

[14] C. Sierra, N. R. Jennings, P. Noriega, and S. Parsons. A framework for argumentation-based negotiations. In M. P. Singh, A. Rao, and M. J. Wooldridge, editors, *Intelligent Agents IV*, pages 177–192. Springer Verlag, Berlin, Germany, 1998.

[15] K. Sycara, M. Paolucci, J. Giampapa, and M. van Velsen. The RETSINA multiagent infrastructure. *Journal of Autonomous Agents and Multiagent Systems*, 7(1), 2003.

[16] K. Sycara and G. Sukthankar. Literature review of teamwork. Technical Report CMU-RI-TR-06-50, Carnegie Mellon University, November 2006.

[17] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7, 1997.

[18] Y. Tang and S. Parsons. Argumentation-based dialogues for deliberation. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 552–559, New York, NY, USA, 2005. ACM Press.