Making it up as they go along: A Theory of Reactive Cooperation

Michael Wooldridge[†] and Afsaneh Haddadi[‡]

[†] Department of Electronic Engineering, Queen Mary & Westfield College
University of London, London E1 4NS, United Kingdom
M.J.Wooldridge@qmw.ac.uk

[‡] Daimler-Benz AG, Forschung und Technik, Alt-Moabit 96a,
10559 Berlin, Germany
afsaneh@DBresearch-berlin.de

Abstract. In this article, we present a formal theory of *on-the-fly cooperation*. This is a new model of joint action, which allows for the possibility that a group of cooperating agents will, in general, have neither the information nor the time available to compute an entire joint plan before beginning to work. It proposes that cooperating agents need therefore only reason about what to do *next*; what represents a *believable next action*. Thus, agents literally *make it up as they go along*: a plan only unfolds as cooperation continues. A detailed rationale is presented for the new model, and the components of the model are discussed at length. The article includes a summary of the logic used to formalise the new model, and some remarks on refinements and future research issues.

1 Introduction

A common assumption in early AI planning research was that in order to achieve a goal φ, an agent should first compute an entire plan π for φ, and then execute π [11]. Within the AI planning community, this strategy has long been recognised to be a severe oversimplification [15]. In the real world, the assumptions that underpin plan execution are continually made false, either through the interference of other agents, or else by actions simply failing to have their intended effects. Recognition of this led to work on planning systems that could integrate planning, execution, and, if required, re-planning, in order to achieve their goals [23].

In multi-agent systems research, however, the recognition of this fact has had little impact on theoretical models of cooperative action. For example, in one of the best-known theories of cooperation, [17], it is assumed that a group of agents have an entire, pre-computed joint plan, which they will carry on executing until certain conditions arise — at which point cooperation ends. While the theory *does* recognise that plans can fail, it nevertheless makes some strong assumptions, which, as the AI planning experience shows, must be called into doubt. In brief, the purpose of this paper is to present a new theory of cooperative action, which does not make such limiting assumptions. In this model, which we call *on-the-fly cooperation*, it is not required that agents have an entire, pre-computed joint plan in order to begin work. Nor is it assumed that once a plan fails, cooperation will end. Rather, it is assumed that in order for cooperation to

continue, agents need only know *what to do next* — what makes a believable next move. Hence agents may *make it up as they go along*, and a global plan might only emerge as cooperation progresses. This is not necessarily a good strategy in all situations, but for some dynamic domains, it has been shown to have certain advantages [13].

The remainder of this paper is structured as follows. First, in the following subsection, we present a more detailed rationale for our work. In section 2, we give an overview of the logic we use to express our new theory, and in section 3, we present the new model itself. Finally, some concluding remarks are presented in section 4.

1.1 Background

Perhaps the best-known theory of cooperative action is that due to Levesque-Cohen [17]. The aim of this model is to give an account of the mental states of agents that are engaged in cooperative, team activities[1]. The theory implicitly requires that a group of agents involved in such an activity have a pre-computed joint plan, which they intend to execute until some agent becomes aware either that the goal of the plan is achieved, or else that the plan has failed, and the goal is unachievable. If such a situation arises, then the agent who recognised it is required to make the group mutually aware of the new circumstances. Building on this work, others have attempted more refined theories of cooperation. For example, in earlier work, we extended the theory to account for the whole process of cooperation, from recognition of the potential of cooperation by some agent, through to the collective development of a plan by the group, which they jointly intend to execute [28].

These theories explain many aspects of collective action. However, they also fail to account for several important features of real-world cooperation. Perhaps most importantly, they pre-suppose the development of an entire plan of collective action. We argue that this assumption is often unrealistic. A group of agents committed to cooperation with respect to some goal do not, in general, have sufficient *resources* to develop a complete joint plan. By resources, we mean both informational and computational. With respect to information, Moore [19] observed that agents often need to *find out* how to achieve a goal: they do not always have enough information to develop a complete plan in advance. Consider telephoning a friend. You may have several perfectly good plans for this, which involve either using the phone on your desk, stopping off at a pay-phone on the way home, and so on, but they all require that you know the number before you dial. You might therefore only be able to develop *part* of the plan in advance. The same is true in the multi-agent case. Consider a group of agents writing a joint paper for an important conference. They may not know *all* the required information as they begin — it may not even be possible to obtain some important information until quite late on. For example, the formatting instructions and submission details for the conference may not be available far enough in advance to actually write the paper if this information must be obtained before cooperation can begin. In any case, such information is not usually significant until close to the submission deadline. In this example, not only is it not *necessary* for the agents to have an entire, pre-computed plan, it is not *possible* for them to do so. But this does not stop them writing the paper.

[1] More precisely, the aim is to define the notion of *joint intention*.

With respect to computational resources, developing a joint plan is at least as complex as single-agent planning, which, as Chapman proved, is *hard* [3]. Even if the agents have sufficient pre-compiled plan fragments to achieve the goal, it will often be unrealistic to try to put these fragments together to make a complete plan before execution commences. Consider an (admittedly extreme) example: building a skyscraper. No one ever has a *complete* plan for erecting such a building, and in fact, low-level details are worked out as construction progresses.

In this paper, we present a new model of cooperative action which does not presuppose an entire pre-computed plan. (Though as a special case, this possibility is allowed for.) Rather, the model proposes that agents are only ever *required* to consider *what to do next*. A global plan *may* be available as cooperation starts — in which case the agents use it. But a plan might only emerge in *retrospect*, as the agents go along. More precisely, the model proposes that a group of agents attempting to achieve φ will continually execute a cycle of trying to find an action π that will take them *closer* to φ, (in the sense of "reducing the distance to φ, as is meant in, for example, means-ends analysis [1]) and then executing π. They do this until either they believe that φ is achieved, or that it is unachievable. While this is not always the *best* cooperation strategy, experimental results indicate that it is well-suited to dynamic domains [13].

2 The Logical Framework

The logic $\mathcal{L}$ that we use to represent our theory of on-the-fly cooperation is a many-sorted first-order version of the expressive branching time logic CTL* [9] with functions and equality, enriched by the addition of modal connectives for representing the *beliefs*, *desires*, and *intentions* of a group of *agents*, as well as the *plans* they have available to them, and the *execution* of these plans. The logic is quite complex, and a complete formal presentation is no possible in the space available. However, such a presentation is available in [25]; in this section, we simply give a summary.

The most obvious thing to say about $\mathcal{L}$ is that it contains the usual connectives and quantifiers of sorted first-order logic: we take as primitive the connectives $\neg$ (not) and $\vee$ (or), and the universal quantifier $\forall$ (for all), and define the remaining classical connectives and existential quantifier in terms of these. As $\mathcal{L}$ is based on CTL*, a distinction is made between *state formulae* and *path formulae*. The idea is that $\mathcal{L}$ is interpreted over a tree-like branching time structure. Formulae that express a property of nodes in this structure are known as *state formulae*, whereas formulae that express a property of paths through the structure are known as *path formulae*. State formulae can be ordinary first-order formulae, but various other additional modal connectives are also provided for making state formulae. Thus $(\mathsf{Bel}\ i\ \varphi)$ is intended to express the fact that the agent denoted by i believes φ (where φ is some state formula). The semantics of belief are given in terms of an accessibility relation over possible worlds, in much the standard modal logic tradition [4], with the properties required of belief accessibility relations ensuring that the logic of belief corresponds to the normal modal system KD45 (weak S5). The state formulae $(\mathsf{Goal}\ i\ \varphi)$ and $(\mathsf{Int}\ i\ \varphi)$ mean that agent i has a desire or intention of φ, respectively: the logics of desire and intention correspond to the normal modal system KD.

$\mathcal{L}$ contains various connectives for representing the *plans* possessed by agents. Plans have two components: a *plan descriptor* and a *plan body*. The plan descriptor characterises the pre- and post-conditions of the plan: the pre-condition represents the conditions under which a plan may be executed, and the post-condition represents the effects of the plan. A plan body is the 'program' part of a plan, which specifies a course of action. Each agent has associated with it a plan library, representing its 'procedural know-how': information it has available to it about how to achieve its intentions. The state formula $(\mathsf{Has}\ i\ \pi)$ is used to represent the fact that in the current state, agent i is in possession of the plan denoted by π. Being in possession of a plan simply means that the plan is in that agent's plan library. Restrictions on the logical model mean that an agent cannot execute a plan unless the plan is in its plan library. Note that π is a *term* of the language, not a formula. We have functional and variable terms that can stand for plans, and the ability to quantify over them is in fact crucial when we come to present our model.

The state formulae $(\mathsf{Pre}\ \pi)$ and $(\mathsf{Post}\ \pi)$ represent the fact that the pre- and post-conditions of the plan π respectively are satisfied in the current world-state. The formula $(\mathsf{Body}\ \pi\ \beta)$ is used to represent the fact that β is the body of the plan denoted by π. We also have a connective $(\mathsf{Holds}\ c)$, which means that the *condition* denoted by c is satisfied in the current world state: the use of this connective, and the rationale behind it, is described in [25].

Turning to path formulae, $(\mathsf{Exec}\ \beta)$ means that the plan body denoted by β is executed on the current path. We will sometimes abuse notation by writing $(\mathsf{Exec}\ \pi)$ to mean that body of plan π has been executed on the current path. State formulae may be related to path formulae by using the CTL* *path quantifier* A. This operator means 'on all paths'. It has a dual, existential operator E, meaning 'on some path'. Thus $\mathsf{A}\varphi$ means that the path formula φ is satisfied on all histories originating from the current world state, and $\mathsf{E}\varphi$ means that the path formula φ is satisfied on at least one history that originates from the current world state. Path formulae may be built up from state formulae (or other path formulae) by using two *temporal connectives*: the U connectives means 'until', and so a formula $\varphi \mathsf{U} \psi$ (where φ and ψ are state formulae) means 'φ is satisfied until ψ is satisfied'. The $\bigcirc$ connective means 'next', and so $\bigcirc\varphi$ means that the state formula φ will be satisfied in the next state.

2.1 Derived Connectives

In addition to the basic connectives discussed above, it is useful to introduce some *derived* constructs. These derived connectives do not add to the expressive power of the language, but are intended to make formulae more concise and readable. First, we assume that the remaining connectives of classical logic, (i.e., $\wedge$ — 'and', $\Rightarrow$ — 'if… then…', and $\Leftrightarrow$ — 'if, and only if') have been defined as normal, in terms of $\neg$ and $\vee$. Similarly, we assume that the existential quantifier, $\exists$, has been defined as the dual of $\forall$. Next, we introduce the *existential path quantifier*, E, which is defined as the dual of the universal path quantifier A. Thus a formula $\mathsf{E}\varphi$ is interpreted as 'on some path, φ', or 'optionally, φ':

$$\mathsf{E}\varphi \stackrel{\text{def}}{=} \neg \mathsf{A} \neg \varphi.$$

It is also convenient to introduce further temporal connectives. The unary connective $\Diamond$ means 'sometimes'. Thus the path formula $\Diamond\varphi$ will be satisfied on some path if φ is satisfied at some point along the path. The unary $\Box$ connective means 'now, and always'. Thus $\Box\varphi$ will be satisfied on some path if φ is satisfied at all points along the path. We also have a weak version of the U connective: $\varphi\mathsf{W}\psi$ is read 'φ *unless* ψ'.

$$\Diamond\varphi \stackrel{\text{def}}{=} \mathsf{true}\mathsf{U}\varphi \qquad \Box\varphi \stackrel{\text{def}}{=} \neg\Diamond\neg\varphi \qquad \varphi\mathsf{W}\psi \stackrel{\text{def}}{=} (\varphi\mathsf{U}\psi) \vee \Box\varphi.$$

Thus $\varphi\mathsf{W}\psi$ means that either: (i) φ is satisfied until ψ is satisfied, or else (ii) φ is always satisfied. It is *weak* because it does not require that ψ be eventually satisfied.

Rather than introduce a further primitive modal connective for knowledge, we define it as true belief.

$$(\mathsf{Know}\ i\ \varphi) \stackrel{\text{def}}{=} \varphi \wedge (\mathsf{Bel}\ i\ \varphi)$$

It is often convenient to make use of *mutual* mental states, although such states are idealisations, not realisable in any system that admits the possibility of communication failure [10, pp176–183]. The mutual belief of φ in a group of agents g is written $(\mathsf{M\text{-}Bel}\ g\ \varphi)$; the mutual goal of φ in g is written $(\mathsf{M\text{-}Goal}\ g\ \varphi)$, and the mutual knowledge of φ is written $(\mathsf{M\text{-}Know}\ g\ \varphi)$. We define mutual mental states as the *maximal fixed points* of the following formulae (cf. [10, pp402–411]):

$$\begin{aligned}
(\mathsf{M\text{-}Bel}\ g\ \varphi) &\stackrel{\text{def}}{=} \forall i \cdot (i \in g) \Rightarrow (\mathsf{Bel}\ i\ \varphi \wedge (\mathsf{M\text{-}Bel}\ g\ \varphi)) \\
(\mathsf{M\text{-}Goal}\ g\ \varphi) &\stackrel{\text{def}}{=} \forall i \cdot (i \in g) \Rightarrow (\mathsf{M\text{-}Bel}\ g\ (\mathsf{Goal}\ i\ \varphi)) \\
(\mathsf{M\text{-}Know}\ g\ \varphi) &\stackrel{\text{def}}{=} \varphi \wedge \forall i \cdot (i \in g) \Rightarrow (\mathsf{M\text{-}Bel}\ i\ (\mathsf{M\text{-}Know}\ g\ \varphi))
\end{aligned}$$

Talking about Groups: The language $\mathcal{L}$ provides us with the ability to use simple (typed) set theory to relate the properties of agents and groups of agents. The operators $\subseteq$ and $\subset$ relate groups together, and have the obvious set-theoretic interpretation; $(\mathsf{Singleton}\ g\ i)$ means g is a singleton group with i as the only member; $(\mathsf{Singleton}\ g)$ simply means g is a singleton.

$$\begin{aligned}
(g \subseteq g') &\stackrel{\text{def}}{=} \forall i \cdot (i \in g) \Rightarrow (i \in g') \quad (\mathsf{Singleton}\ g\ i) \stackrel{\text{def}}{=} \forall j \cdot (j \in g) \Rightarrow (j = i) \\
(g \subset g') &\stackrel{\text{def}}{=} (g \subseteq g') \wedge \neg(g = g') \qquad (\mathsf{Singleton}\ g) \stackrel{\text{def}}{=} \exists i \cdot (\mathsf{Singleton}\ g\ i)
\end{aligned}$$

$(\mathsf{Agt}\ \beta\ i)$ means that i is the only agent required to perform plan body β.

$$(\mathsf{Agt}\ \beta\ i) \stackrel{\text{def}}{=} \forall g \cdot (\mathsf{Agts}\ \beta\ g) \Rightarrow (\mathsf{Singleton}\ g\ i)$$

Talking about Plans: Next, we introduce some operators that will allow us to conveniently represent the structure and properties of plans. First, we introduce two constructs, $(\mathsf{Pre}\ \pi\ \varphi)$ and $(\mathsf{Post}\ \pi\ \varphi)$, that allow us to represent the pre- and post-conditions of plans as formulae of $\mathcal{L}$. Thus $(\mathsf{Pre}\ \pi\ \varphi)$ means that φ corresponds to the pre-condition of π — that φ is satisfied in just those situations where the pre-condition of π is satisfied:

$$(\mathsf{Pre}\ \pi\ \varphi) \stackrel{\text{def}}{=} \mathsf{A}\Box((\mathsf{Pre}\ \pi) \Leftrightarrow \varphi).$$

Similarly, $(\mathsf{Post}\ \pi\ \varphi)$ means that φ is satisfied in just those situations in which the post-condition of π is satisfied:

$$(\mathsf{Post}\ \pi\ \varphi) \stackrel{\text{def}}{=} \mathsf{A}\Box((\mathsf{Post}\ \pi) \Leftrightarrow \varphi).$$

We write $(\mathsf{Plan}\ \pi\ \varphi\ \psi\ \beta)$ to express the fact that plan π has pre-condition φ, post-condition ψ, and body β:

$$(\mathsf{Plan}\ \pi\ \varphi\ \psi\ \beta) \stackrel{\text{def}}{=} (\mathsf{Pre}\ \pi\ \varphi) \wedge (\mathsf{Post}\ \pi\ \psi) \wedge (\mathsf{Body}\ \pi\ \beta).$$

It is often useful to be able to talk about the *structure* of plans bodies. We allow such plan bodies to be composed from atomic actions using sequential composition, parallel composition, test, non-deterministic choice, and iteration. We introduce some logical functions, which operate on terms denoting plan bodies, and return other plan bodies corresponding to each of these operations. We introduce one function for each of the plan constructors:

$$\textit{seq} \text{ for } ; \quad \textit{par} \text{ for } \| \quad \textit{test} \text{ for } ? \quad \textit{or} \text{ for } | \quad \textit{iter} \text{ for } *.$$

These functions are required to satisfy certain properties. For example, for all plan bodies β, β', we require that $\textit{seq}(\beta, \beta')$ returns the plan body obtained by conjoining the plan bodies denoted by β and β' with the sequential composition constructor. These functions allow us to construct plan bodies within our language. However, complex plan bodies written out in full using these functions become hard to read. To make such expressions more readable, we introduce a *quoting convention*. The idea is best illustrated by example. We write

$\ulcorner \beta; \beta' \urcorner$	to abbreviate $\textit{seq}(\beta, \beta')$
$\ulcorner \beta; (\beta' \parallel \beta'') \urcorner$	to abbreviate $\textit{seq}(\beta, \textit{par}(\beta', \beta''))$
$\ulcorner \beta; (\beta' \parallel \beta'')* \urcorner$	to abbreviate $\textit{seq}(\beta, \textit{iter}(\textit{par}(\beta', \beta'')))$

and so on. In the interests of consistency, we shall generally use quotes even where they are not strictly required. By means of a rather involved technical construction, we can apply tests to state formula in plan bodies. So, for example, if φ is a state formula, then $\varphi?$ is an acceptable plan body, which will be executed on a path just in case the formula φ is satisfied on the first state of that path. (Readers interested in how we avoid breaking the type rules of our language in using this construction are urged to consult [25].)

The readability of plan body expressions may be further improved by the introduction of derived constructs corresponding to the high-level statement-types one would expect to find in a standard imperative language such as PASCAL. First, the *if... then...* construct:

$$\ulcorner \texttt{if}\ \varphi\ \texttt{then}\ \beta\ \texttt{else}\ \beta'\ \texttt{end-if} \urcorner \stackrel{\text{def}}{=} \ulcorner (\varphi?; \beta) \mid (\neg\varphi; \beta') \urcorner.$$

While and *repeat* loops are similarly easy to define:

$$\ulcorner \texttt{while}\ \varphi\ \texttt{do}\ \beta\ \texttt{end-while} \urcorner \stackrel{\text{def}}{=} \ulcorner (\varphi?; \beta)*; \neg\varphi? \urcorner$$
$$\ulcorner \texttt{repeat}\ \beta\ \texttt{until}\ \varphi \urcorner \stackrel{\text{def}}{=} \ulcorner \beta; \texttt{while}\ \neg\varphi\ \texttt{do}\ \beta\ \texttt{end-while} \urcorner.$$

The `case` structure has a similar use to such statements in languages like PASCAL (cf. the `switch` statement in C).

$$\begin{array}{l}\ulcorner\texttt{case}\\ \quad \varphi_1 : \pi_1\\ \quad \cdots\\ \quad \varphi_n : \pi_n\\ \quad \texttt{else} : \pi_{n+1}\\ \texttt{end-case}\urcorner\end{array} \stackrel{\text{def}}{=} \ulcorner(\varphi_1?;\pi_1)|\cdots|(\varphi_n?;\pi_n)|((\neg\varphi_1 \wedge \cdots \wedge \neg\varphi_n)?;\pi_{n+1})\urcorner$$

Thus exactly one of the actions π_1 to π_{n+1} is executed. The `else` clause is a default action, which is executed if none of the other conditions evaluates to true. Note that the conditions $\varphi_1, \ldots, \varphi_n$ are assumed to be mutually exclusive.

Finally, we define an `await` construct:

$$\ulcorner\texttt{await}\ \varphi\urcorner \stackrel{\text{def}}{=} \ulcorner\texttt{repeat true? until}\ \varphi\urcorner.$$

Thus `await` φ will be executed on a path p if there is some point on p at which φ is true.

3 On-the-Fly Cooperation

In this section, which represents the main contribution of this article, we present our model of on-the-fly cooperation. We begin, in the remainder of this sub-section, with an overview.

Informally, the structure of the model is very easy to understand. Suppose that a group of agents g are cooperating in order to achieve φ (for the sake of argument, we shall assume throughout this article that φ represents a world state). Then the model proposes that g repeatedly perform a cycle of finding a plan π that will in some sense advance them towards φ, and then executing π. They carry on doing this until either they succeed in bringing about φ, or else they become aware that they are no longer able to achieve φ. Of course, figuring out a course of action that will move you towards your goal is far from being trivial: much of AI, (and in particular, the whole of the AI planning paradigm), is directed at precisely this problem. In our model, finding out what to do involves a simple cooperative search of every agent's plan library, in an attempt to find a plan that will reduce the 'distance' to the goal. Agents in our model do no first-principles planning at all. However, we do not assume that agents will *blindly* execute any plan that will reduce the distance to the goal: agents are *autonomous*, with their own desires and intentions, and may *object* to a particular plan. (They may also have strong preferences in *favour* of a particular plan, though we shall not concern ourselves with this possibility here.) The search for a plan must take account of such objections. This leads us to the informal summary of our model in Figure 1, where a group of agents g are attempting to bring about φ.

The remainder of this section is structured as follows. First, in the following section, we informally lay out the assumptions that underpin our model; in section 3.2, we formally define some of the key concepts used in the model, and in section 3.3, we formally define the communication primitives (performatives) that agents can use. Finally, in section 3.4, we present the model itself.

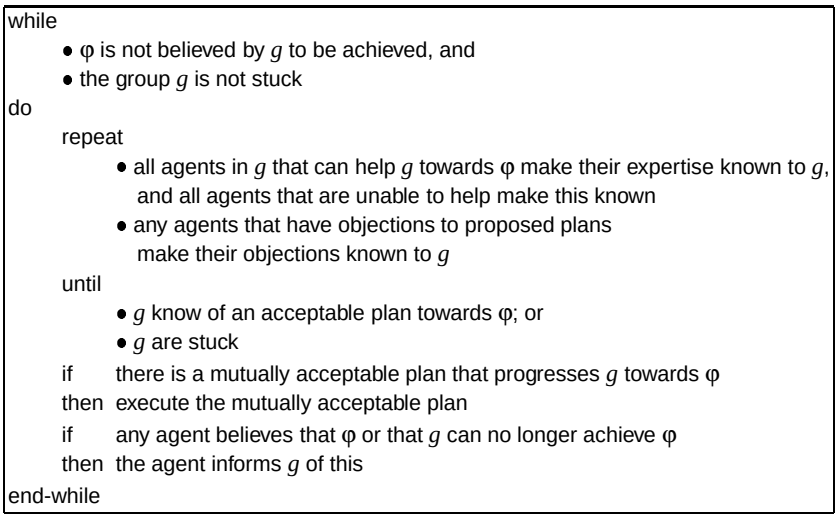

Fig. 1. On-the-Fly Cooperation: An Overview

3.1 Assumptions

It is important to realise that the model we present in this paper is a *first approximation* to a theory of on-the-fly cooperation. *Real* cooperation, of the type that most of us take part in every day of our lives, is, of course, a much more subtle and complex process than our model might indicate. This complexity is likely to prevent complete attempts at formalisation for the foreseeable future. So, rather than attempting to present a complete formalisation, we have selected a simple, stylized, but, we argue, plausible subset of the phenomenon, which we make tractable by the use of some limiting assumptions. Obviously, it is important that these assumptions should strike a balance between being too strong and being too weak. Too strong, and the model becomes trivial; too weak, and it becomes too complex. We believe that the following assumptions are reasonable for a first approximation.

Agents are autonomous: As Castelfranchi shows, [2], autonomy is a slippery subject. For our purposes, we simply take autonomy to mean that agents will not blindly commit to courses of action that conflict with their own intentions. (Agents are therefore not benevolent, since it is not assumed that they share all their intentions [21, p91].)

Agents are helpful: Autonomy does not imply meanness: we assume that agents are happy to perform actions on behalf of a group, provided that the performance of such actions does not conflict with their autonomy.

Agents are accommodating: This is really an aspect of helpfulness: agents will take on board the objections that other agents might have to a plan.

Agents are fanatical: We mean this in the sense of [5]. A group of agents will carry

on trying to achieve their goal until either they have successfully achieved it, or else they believe that it is impossible for them to do so.

Agents communicate: Communication is not universally assumed in multi-agent systems research, but communication via message passing with KQML-like performatives is nevertheless a common assumption, which we adopt here [18].

Agents are veracious: By which we simply mean that agents do not tell lies: they only communicate information that they believe to be true [12, pp159–165].

3.2 Some Concepts and Definitions

In this section, we formally define some of the concepts that will be useful when presenting our model. We begin with *joint ability*, by which we mean the circumstances under which a group of agents can achieve some goal. Reasoning about ability, and in particular, whether it is in principle possible for a group to achieve a goal, is clearly a fundamental aspect of cooperative action. A number of previous definitions of joint ability have appeared in the literature. For example, in earlier work, [27], we defined ability in the sense of 'potential': what an agent could *potentially* bring about. This definition did not refer to the *know-how* of a group at all. The formalisation was based on earlier work by Werner, [24], who defined a notion of joint ability for φ as the group having a 'winning strategy' for φ, in the game-theoretic sense. However, such definitions do not capture the everyday sense of ability, as they ignore what a group of agents need to know in order to *realise* their potential. For example, we might have the *potential* to become millionaires within the next year, by performing some sequence of actions. But this information is, sadly, useless without knowing *which* actions to perform. So, we define a notion of joint ability in line with the definition of single agent ability developed by Moore [19]. Crudely, his definition of ability runs as follows: an agent i can achieve φ if i knows the *identity* of some action α that it can perform, such that it knew that after it did α, either φ would be achieved, or else i would be able to achieve φ. The point is that this definition allows for the possibility of *i finding out* how to do φ. Thus the agent need not have an entire, pre-computed plan.

In order to formalise this concept, we first define what it means for a group to be *sufficient* to carry out a plan. The idea is that group g will be *sufficient* to do plan π, (notation: $(\mathsf{Suff}\ g\ \pi)$), iff the agents required to do π are a subset of g.

$$(\mathsf{Suff}\ g\ \pi) \stackrel{\text{def}}{=} \forall \beta \cdot \forall g' \cdot (\mathsf{Body}\ \pi\ \beta) \wedge (\mathsf{Agts}\ \beta\ g') \Rightarrow (g' \subseteq g).$$

We then adapt the single-agent definition in the following way. A group g can *jointly achieve* φ, (notation: $(\mathsf{J\text{-}Can}\ g\ \varphi)$), iff there exists some plan π, such that it is mutually known in g that π is possessed by some member of g (whose identity is known), that g are sufficient to do π, and either:

1. it is mutually known in g that φ is a post-condition of π; or else
2. it is mutually known in g that the post-condition of π is that g can jointly achieve φ.

Ability is thus defined as the *least fixed point* of the following formula:

$$(\text{J-Can } g\ \varphi) \stackrel{\text{def}}{=} \exists i \cdot \exists \pi \cdot (\text{M-Know } g\ (i \in g) \wedge (\text{Has } i\ \pi) \wedge (\text{Suff } g\ \pi)) \wedge ((\text{M-Know } g\ (\text{Post } \pi\ \varphi)) \vee (\text{M-Know } g\ (\text{Post } \pi\ (\text{J-Can } g\ \varphi)))).$$

Note that the variables i and π are quantified *outside* the scope of the M-Know connectives. This implies that the *identities* of i and π are known to g. Thus if the group do *not* know the identities of the agent and plan, but they know there there is *some* agent and *some* plan, this is not sufficient for joint ability. In the terminology of quantified modal logic, these variables are said to be quantified *de re* [14, pp183–188].

Next, an agent i is said to *object* to plan π, (notation: $(\text{Objects } i\ \pi)$), iff i intends that π is not executed.

$$(\text{Objects } i\ \pi) \stackrel{\text{def}}{=} \forall \beta \cdot (\text{Body } \pi\ \beta) \Rightarrow (\text{Int } i\ \text{A} \neg (\text{Exec } \beta)).$$

A plan will be mutually acceptable to a group if the group is mutually aware that no member of the group objects to it.

$$(\text{Acceptable } \pi\ g) \stackrel{\text{def}}{=} (\text{M-Bel } g\ \forall i \cdot (i \in g) \Rightarrow \neg (\text{Objects } i\ \pi)).$$

Note that it is not possible for an agent within a group to object to a plan, and yet for that plan to be mutually acceptable to the group. Next, we consider the notion of a plan progressing a group towards a goal. Intuitively, we say that a plan does this if either it achieves the goal, or else the plan makes it possible for the group to achieve the goal (cf. the definition of J-Can, above). The former case (where the plan directly achieves the goal) is, in a sense, more desirable than the latter. We formalise this as follows:

$$(\text{Progresses } \pi\ g\ \varphi) \stackrel{\text{def}}{=} (\text{Suff } g\ \pi) \wedge ((\text{Post } \pi\ \varphi) \vee (\text{Post } \pi\ (\text{J-Can } g\ \varphi))).$$

(Note that although the two cases in this definition correspond to those in J-Can, above, it is not possible to define J-Can in terms of Progresses, or Progresses in terms of J-Can.) Next, we define what it means for an agent to be able to *help* the group with its goal. The idea is that agent i can help group g with respect to φ, (notation: $(\text{CanHelp } i\ g\ \varphi)$), iff i has a plan π such that:

1. π progresses g towards φ;
2. i does not object to π; and
3. i does not believe that π is unacceptable to g.

$$(\text{CanHelp } i\ g\ \varphi) \stackrel{\text{def}}{=} \exists \pi \cdot (\text{Has } i\ \pi) \wedge (\text{Progresses } \pi\ g\ \varphi) \wedge \neg (\text{Objects } i\ \pi) \wedge \neg (\text{Bel } i\ \neg (\text{Acceptable } \pi\ g)).$$

Note that i is quantified *de re*, implying that g know the identity of i. Finally, it is useful to identify certain facts as being *communicable*. The idea is that φ will be communicable to g if φ is true, but φ is not (yet) mutually believed to be true in g.

$$(\text{Comm } \varphi\ g) \stackrel{\text{def}}{=} \varphi \wedge \neg (\text{M-Bel } g\ \varphi).$$

Finally, a group will be *stuck* with respect to a goal φ if they are aware of no plan that progresses them towards φ.

$$(\text{Stuck } g\ \varphi) \stackrel{\text{def}}{=} \forall i \cdot (i \in g) \Rightarrow (\text{M-Bel } g\ \neg (\text{CanHelp } i\ g\ \varphi))$$

3.3 Performatives

The cooperative search for a mutually acceptable plan proceeds via the exchange of messages, in which agents make the group aware of both plans they have that might be useful for the group, and any objections they have with respect to plans. The exchange of messages continues until it becomes clear either that there is a mutually acceptable plan that advances the group, (i.e., the group CanProgress), or else that no such plan is available to the group (i.e., the group are Stuck).

Ultimately, the search for a mutually acceptable plan is a simple form of negotiation [22]. In human negotiation, the participants have preferences, goals, and intentions that they may ultimately compromise on, or they may threaten, lie, bluff, appeal, plead, promise, and so on, in an attempt to bring about their desires. Negotiation becomes a game of give-and-take. Rather than attempting to represent this rather complex process, we simply observe assume that agents have certain inviolable intentions; negotiation is an attempt to find a plan that is acceptable to all with respect to their intentions.

In essence, our simple model of negotiation involves agents proposing various plans, and then informing one-another of their preferences with respect to these plans. Proposing and informing are done by agents executing communicative actions: performatives. These performatives are modelled within the logic as functional terms that denote actions. The two performatives are:

$$\begin{array}{ll} \mathit{propose}(i,g,\pi,\varphi) & i \text{ proposes that } g \text{ might do } \pi \text{ to achieve } \varphi \\ \mathit{inform}(i,g,\varphi) & i \text{ informs } g \text{ that } \varphi \end{array}$$

We do not intend that proposing indicates any *preference* on the part of the proposer with respect to the proposed plan: it is simply a way for an agent to communicate to a group one way of achieving a goal. However, we do assume that:

- agents only propose plans that they have no objection to; and
- agents only propose plans that they are in possession of.

This leads us to the following semantics for *propose*:

$$\forall \pi' \cdot (\mathsf{Body}\ \pi'\ \ulcorner \mathit{propose}(i,g,\pi,\varphi)\urcorner) \Rightarrow (\mathsf{Pre}\ \pi'\ p) \wedge (\mathsf{Post}\ \pi'\ (\mathsf{M\text{-}Bel}\ g\ p)) \qquad (1)$$

where

$$p \stackrel{\text{def}}{=} \neg(\mathsf{Objects}\ i\ \pi) \wedge (\mathsf{Has}\ i\ \pi) \wedge (\mathsf{Suff}\ g\ \pi) \wedge (\mathsf{Post}\ \pi\ \varphi).$$

(There are actually some further assumptions hidden within this definition, such as, for example, the fact that communication is guaranteed, and that messages will be delivered instantaneously.) The semantics of the *inform* performative are also quite simple:

$$\forall \pi \cdot (\mathsf{Body}\ \pi\ \ulcorner \mathit{inform}(i,g,\varphi)\urcorner) \Rightarrow (\mathsf{Pre}\ \pi\ (\mathsf{Bel}\ i\ \varphi)) \wedge (\mathsf{Post}\ \pi\ (\mathsf{M\text{-}Bel}\ g\ \varphi)) \qquad (2)$$

Note that in both (1) and (2), the pre-condition represents the sincerity condition associated with the performative, capturing the veracity assumption that we mentioned earlier. Also, note that we are not attempting to define a semantics of speech acts [6]: our aim is simply to define some message types that agents can use when cooperating.

```
while ¬(Bel i φ) ∧ (Bel i (J-Can g φ)) do
  /* phase 1: finding an acceptable plan */
  repeat
    /* phase 1.1: propose any plans that help */
    if (CanHelp i g φ) then
        ∃π · (Comm ((Has i π φ) ∧ ¬(Objects i π)) g) ⇒ (Exec ⌈propose(i, g, π, φ)⌉)?
    else
      ⌈inform(i, g, ¬(CanHelp i g φ))⌉
    end-if
    /* phase 1.2: wait until everyone has announced */
    await ∀j · (j ∈ g) ⇒ (Bel i (CanHelp j g φ)) ∨ (Bel i ¬(CanHelp j g φ))
    /* phase 1.3: veto any unacceptable plans */
    while ∃π · (Progresses π g φ) ∧ (Comm (Objects i π) g) do
      ∃π · (Progresses π g φ) ∧ (Comm (Objects i π) g) ⇒ (Exec ⌈inform(i, g, (Objects i π))⌉)?
    end-while
  until (Bel i (Stuck g φ) ∨ (CanProgress g φ))
  /* phase 2: execute acceptable plan */
  if ∃π · (Bel i (Acceptable π g) ∧ (Progresses π g φ) ∧ (Agt π i) then
    ∃π · (Bel i (Acceptable π g) ∧ (Progresses π g φ) ∧ (Agt π i) ⇒ (Exec ⌈π⌉)?
  end-if
  /* phase 3: post results */
  case
    (Bel i φ)              : inform(i, g, φ)
    (Bel i ¬(J-Can g φ))   : inform(i, g, ¬(J-Can g φ))
    else                   : true?  /* i.e., NOP */
  end-case
end-while
```

Fig. 2. A Model of Reactive Cooperation

3.4 Finally, the Model

We now come to the model itself. This model is formalised as a single-agent plan: each agent individually executes the plan in order to generate the overall behaviour summarised in 1. The model is presented in Figure 2; the agent assumed to be executing the plan is *i*.

The outer loop represents the fanatical commitment that agents have with respect to φ: they will carry on attempting to bring about φ while they believe they have not yet achieved it (the first conjunct), and that they can still achieve it (the second conjunct). We do not claim that one would want these conditions for *every* type of on-the-fly cooperative action: different problem domains call for different types of commitment [20]. Within the main loop are three main phases:

- finding an acceptable plan that "progresses" the group towards the goal (phase 1);
- executing such an acceptable plan (phase 2);

– updating the group with respect to the current status of group action (phase 3).

Phase 1: Finding a mutually acceptable plan. This first phase represents the key problem in on-the-fly cooperation: finding a plan that will progress the agents towards the goal, such that this plan is acceptable to all. The process of finding a plan involves agents repeatedly proposing plans that they believe will move them closer to the goal, and then vetoing any proposed plans that they object to, until eventually either the agents have considered all possible plans and found none to be acceptable (they are Stuck), or else they have found a plan that is acceptable to all (they CanProgress). This phase thus involves repeatedly executing the following steps:

– proposing any plan that may help, or else informing the group of the inability to help (phase 1.1);
– waiting until everyone else has done likewise (phase 1.2);
– vetoing any plans that are unacceptable (phase 1.3).

In stage 1.1, if an agent can help towards a goal, (i.e., it has a plan that progresses the group towards the goal, and it does not believe this plan is unacceptable to the group), then it proposes the plan to the group. Otherwise it informs the group that it is unable to help, i.e., it has no plans that progress the group towards the goal that would be acceptable to the group. Note that stage 1.3 (vetoing unacceptable plans, below), will ensure that an agent does not propose a plan that has been vetoed by some other agent. Thus an agent will eventually either propose all acceptable plans in its plan library, or else will announce that it cannot help.

Stage 1.2, (the `await` loop) represents a simple synchronization condition: an agent will simply wait until every other agent has informed it of the fact that it can or cannot help.

In stage 1.3, the agent repeatedly informs the group of any plans that it objects to. It will carry on doing this until either the group has found a plan that is mutually acceptable, or else the group is aware that no such plan is available.

Phase 2: Executing Acceptable Plans. If the group succeeds in finding a mutually acceptable plan that progresses them towards their goal, then the next thing they should do is execute it. From the point of view of an individual agent, this means simply executing a plan that it is the agent of, such that the agent is the only one required to carry out the plan, the plan is acceptable to all, and the plan progresses the group to the goal.

Phase 3: Posting Results. For this part of the process, we simply require that agents who either believe that the goal has been achieved, or else believe that the goal is unachievable, make the group mutually aware of this fact. If an agent neither believes that the goal is achieved, or that the goal is unachievable, then it does nothing.

4 Conclusions

We conclude with some general remarks on the formalisation. First, we note that the *layered* approach, pioneered by Cohen and Levesque [5], which we have adopted in this

paper, allows us to present the formal model with comparative ease. Another obvious point to make is that presenting an agent's plan directly as a "procedure" by using a dynamic logic-style program logic, and combining this with a BDI logic, allows a number of complex ideas to be succinctly represented.

However, there are a number of points at which the formalisation is weak, and, it could be argued, inadequate. In particular, the fact that our simple plan language does not have assignment statements makes it hard to express a number of concepts, requiring a rather ugly technical kludge to be used instead. (An example is the `if` statement in phase 2 of Figure 2.) An obvious enhancement to the underlying formalism would be the inclusion of such statements into a richer plan language.

With respect to the specifics of the formalism, there are at least two points at which more work needs to be done. First, the notion of "progression", (i.e., the notion of agents moving towards the goal), is not satisfactorily defined. It may be that a utility-theoretic definition is more appropriate: a plan progresses the team towards the goal if the expected cost of achieving the goal after executing the plan is less than the cost of achieving it before. Similarly, we have not attempted to formalise any notion of *preference* with respect to the plans that are executed. A more realistic model would include such notions. The formalisation of *negotiation* could also be refined. Some preliminary attempts to give logical specifications of negotiation appear in [16]. Incorporating such models into our model may prove useful.

With respect to other areas for future work, it may prove helpful to attempt to marry our work on formal models of social action with the various models of cooperation and coordination that have been developed by practitioners of agent systems. For example, building on the work of Durfee on partial global planning [8], Decker has investigated five general techniques for coordinating dynamic multi-agent systems [7]. It would be interesting to investigate the extent to which such techniques could be represented using our modal/dynamic logic approach. Finally, the relationship between our theory of on-the-fly cooperation and implementations of cooperative protocols could also bear further examination.

Acknowledgments: This work was carried out while the first author was a visiting researcher at Daimler-Benz research institute in Berlin. Thanks to Kurt Sundermeyer, who made the visit possible, and everyone at DB for being so friendly.

References

1. J. F. Allen, J. Hendler, and A. Tate, editors. *Readings in Planning.* Morgan Kaufmann Publishers: San Mateo, CA, 1990.
2. C. Castelfranchi. Guarantees for autonomy in cognitive agent architecture. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 56–70. Springer-Verlag: Berlin, Germany, January 1995.
3. D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–378, 1987.
4. B. Chellas. *Modal Logic: An Introduction.* Cambridge University Press: Cambridge, England, 1980.

5. P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
6. P. R. Cohen and H. J. Levesque. Rational interaction as the basis for communication. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 221–256. The MIT Press: Cambridge, MA, 1990.
7. K. Decker and V. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 73–80, San Francisco, CA, June 1995.
8. E. H. Durfee. *Coordination of Distributed Problem Solvers*. Kluwer Academic Publishers: Boston, MA, 1988.
9. E. A. Emerson and J. Y. Halpern. 'Sometimes' and 'not never' revisited: on branching time versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.
10. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. The MIT Press: Cambridge, MA, 1995.
11. R. E. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 5(2):189–208, 1971.
12. J. R. Galliers. *A Theoretical Framework for Computer Models of Cooperative Dialogue, Acknowledging Multi-Agent Conflict*. PhD thesis, Open University, UK, 1988.
13. M. P. Georgeff and F. F. Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 972–978, Detroit, MI, 1989.
14. G. E. Hughes and M. J. Cresswell. *Introduction to Modal Logic*. Methuen and Co., Ltd., 1968.
15. L. P. Kaelbling. An architecture for intelligent reactive systems. In M. P. Georgeff and A. L. Lansky, editors, *Reasoning About Actions & Plans — Proceedings of the 1986 Workshop*, pages 395–410. Morgan Kaufmann Publishers: San Mateo, CA, 1986.
16. S. Kraus, M. Nirke, and K. Sycara. Reaching agreements through argumentation: A logical model. In *Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence (IWDAI-93)*, pages 233–247, Hidden Valley, PA, May 1993.
17. H. J. Levesque, P. R. Cohen, and J. H. T. Nunes. On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 94–99, Boston, MA, 1990.
18. J. Mayfield, Y. Labrou, and T. Finin. Evaluating KQML as an agent communication language. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II (LNAI Volume 1037)*, pages 347–360. Springer-Verlag: Berlin, Germany, 1996.
19. R. C. Moore. A formal theory of knowledge and action. In J. F. Allen, J. Hendler, and A. Tate, editors, *Readings in Planning*, pages 480–519. Morgan Kaufmann Publishers: San Mateo, CA, 1990.
20. A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-91)*, pages 473–484. Morgan Kaufmann Publishers: San Mateo, CA, April 1991.
21. J. S. Rosenschein and M. R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 91–99, Los Angeles, CA, 1985.
22. J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. The MIT Press: Cambridge, MA, 1994.
23. S. Vere and T. Bickmore. A basic agent. *Computational Intelligence*, 6:41–60, 1990.
24. E. Werner. What can agents do together: A semantics of co-operative ability. In *Proceedings of the Ninth European Conference on Artificial Intelligence (ECAI-90)*, pages 694–701, Stockholm, Sweden, 1990.

25. M. Wooldridge. Practical reasoning with procedural knowledge: A logic of BDI agents with know-how. In D. M. Gabbay and H.-J. Ohlbach, editors, *Practical Reasoning — Proceedings of the International Conference on Formal and Applied Practical Reasoning, FAPR-96 (LNAI Volume 1085)*, pages 663–678. Springer-Verlag: Berlin, Germany, June 1996.
26. M. Wooldridge, S. Bussmann, and M. Klosterberg. Production sequencing as negotiation. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-96)*, pages 709–726, London, UK, April 1996.
27. M. Wooldridge and M. Fisher. A first-order branching time logic of multi-agent systems. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, pages 234–238, Vienna, Austria, 1992.
28. M. Wooldridge and N. R. Jennings. Formalizing the cooperative problem solving process. In *Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence (IWDAI-94)*, pages 403–417, Lake Quinalt, WA, July 1994.