A Multiagent Planning Language

Michael Brenner
Institute for Computer Science
University of Freiburg, Germany
Georges-Koehler-Allee, Geb. 052
79110 Freiburg
brenner@informatik.uni-freiburg.de

Abstract

This paper discusses specific features of planning in multiagent domains and presents concepts for a multiagent extension of PDDL, the Multiagent Planning Language MAPL ("maple"). MAPL uses non-boolean state variables and thus allows to describe an agent's ignorance of facts as well as a simplified mutex concept. The time model of MAPL is based on Simple Temporal Networks and allows both quantitavive and qualitative use of time in plans, thereby subsuming the plan semantics of both partial order plans and PDDL 2.1.

Introduction

In this paper we describe some properties specific to planning in multiagent systems and, resulting from these properties, propose a multiagent extension of PDDL, the Multiagent Planning Language MAPL (pronocunced "maple"). By Multiagent Planning (MAP) we denote any kind of planning in multiagent environments, meaning on the one hand that the planning process can be distributed among several *planning* agents, but also that individual plans can (and possibly must) take into account concurrent actions by several *executing* agents. We do not impose any relation among planning and executing agents: one planner can plan for a group of concurrent executers (this corresponds roughly to planning with PDDL 2.1 but necessitates extensions allowing more execution flexibility), several planners can devise one shared plan (linear or not) or, in the general case, m planners plan for n executing agents. In the specific, yet common case of n agents, each having both planning and executing capabilities we speak of *autonomous agents*. Note that we do neither assume cooperativity nor competition among agents.

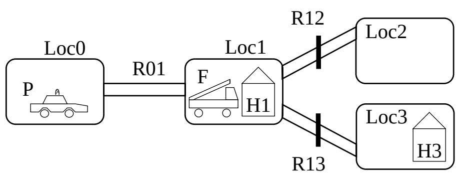

Figure 1: A multiagent planning problem

As a motivating example, fig. 1 shows a simple MAP problem as it appears in the RoboCupRescue simulation (Kitano *et al.* 1999). There are two autonomous agents: police force P and fire brigade F. They have different capabilities: P clears blocked roads, F extinguishes burning houses, both can move on unblocked roads. Each action has a duration which may vary because of specific execution parameters (e.g. location distance, motion speed) and/or intrinsic unpredictability. For this example, we assume a duration of 30 to 180 minutes for *clear*, 1 to 4 hours for *extinguish*, and 2 to 4 minutes for *move*. The speed and thus the duration of *move* is controlled by each agent while the duration intervals for *clear* and *extinguish* can only be estimated. The agents' knowledge and goals are differing, too: P wants the roads to be clear, but is unaware of the state of all roads except $R01$. F wants all burning houses extinguished, knows that $H1$ and $H3$ are burning, but also that it cannot reach $H3$ because road $R13$ is blocked.

Even in this trivial example we can make some general observations about planning in MAS that will motivate the concepts introduced in the rest of the paper.

(1) *Concurrent acting* is central to MAS (P can move to $Loc1$ and start clearing $R13$ while F is extinguishing $H1$). (2) *Metric time* is needed to realistically describe action durations and their relations. (3) Synchronizing on actions of unknown (at least to some agent) duration demands *qualitative* use of time (e.g. "after P has cleared $R13$"). A specific usage of qualitative time in MAP is (4) synchronization on communicative acts, for example "after P has informed me that $R13$ is now clear".

While many recent planning formalisms allow some degree of concurrency, most fail in providing either (2) or (3). PDDL 2.1, for example, supports metric time but enforces planners to assign exact time stamps and durations to all events (Fox and Long 2002). In contrast, the concurrency model of (2001) augments partial order plans with concurrency, thus allowing flexible, synchronized execution, but makes no difference between plans that take seconds and ones that take years. None of the planning models known to us allows to synchronize on communicative acts.

To summarize, PDDL is, in its current form, inadequate for representing MAP problems and their solutions, namely because of the following missing features:

1. **beliefs:** if more than one agent is manipulating the world (unlike assumed by classical planners) facts about it cannot only be true or false, but also simply *unknown* to an agent (e.g. P not knowing whether road $R12$ is clear or

not). Instead of using some kind of possible world semantics, we propose to give up the propositional representation of facts in PDDL and move on to ternary or even n-ary state variables.

2. **model of time:** Not only *quantitative* ("duration is 30 minutes"), but also *qualitative* ("F moves to Loc3 *after* P has cleared R13") models of time are needed to represent and coordinate multiple-agent behavior. To that end, we propose to exchange PDDL's time-point semantics with a semantics of temporal relations among actions that can be both quantitative or qualitative.
3. **degrees of control:** An agent may exploit another agents' actions in her own plan ("F moves to $L3$ after P has cleared $R13$"), but cannot (by removing them from her plan) prevent them from happening. Even her own actions might be only partially controllable by the agent, e.g. duration of *move* (controllable) and *extinguish* (uncontrollable). PDDL must allow to describe controllable and uncontrollable events so that agents can exploit their differing properties during planning.
4. **plan synchronization:** PDDL 2.1's plan semantics forces agents to attribute exact time points to all actions, thus making synchronization of (partial) plans very hard (when trivial merging is impossible). More importantly, for reasons of flexibility and security it is often best to share as little information as possible. To achieve such *minimum synchronization* we suggest not only to change the temporal model but to allow *speech acts* as synchronizing (meta-)actions in a plan. E.g., all F needs to know about P's plan is that at some point P will have cleared $R13$. F can thus enter a speech act TOLD(P,F,R13=CLEAR) into her own plan that has R13=CLEAR as effect and allows F to plan on with that knowledge.

Our extension of PDDL will provide these features. Fig. 2 shows part of a MAPL description for the Rescue domain. Fig. 3 shows a MAPL plan of agent F for the problem given in Fig. 1

```
(:state-variables
  (pos ?a - agent) - location
  (connection ?p1 ?p2 - place) - road
  (clear ?r - road) - boolean)
(:durative-action Move
  :parameters (?a - agent ?dst - place)
  :duration (:= ?duration (interval 2 4))
  :condition
    (at start (clear (connection (pos ?a) ?dst)))
  :effect (and
    (at start (:= (pos ?a) (connection (pos ?a) ?dst)))
    (at end (:= (pos ?a) ?dst))))
```

Figure 2: Excerpt from a MAPL domain description

The remainder of the paper presents MAPL solutions to these problems: first, we show how to describe beliefs conveniently as non-binary state variables. Then we present the temporal model of MAPL and its representation of events and actions. The concepts of control over and mutual exclusivity among events are introduced in the following sections, preparing the ground for the definition of MAPL's plan semantics. Finally, we show how speech acts can provide synchronization between plans of several agents.

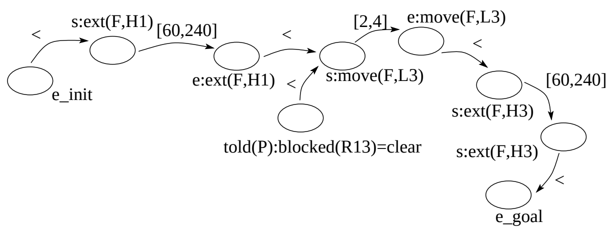

Figure 3: F's plan including a reference speech act by P

Beliefs and other state variables

One main feature distinguishing MAPL from PDDL is the use of non-propositional state variables: in MAP we must dismiss the Closed-World Assumption (CWA) that everything not known to be true is false – the truth value might also be simply *unknown* to an agent. There are several possibilities to represent such belief states, for example sets of possible states (possible worlds) could represent all possible combinations of states for unknown facts. Another possibility is to represent each of the three possible states of a fact (true, false, unknown) by a unique proposition and to assure that exactly of one these propositions hold in any given state. This is similar to the representation of negation proposed in (Gazen and Knoblock 1997): explicit negation of a fact is compiled away in a planning domain by introducing a special proposition representing the negated fact and assuring in the planning domain that only one of the two facts can hold in a state.

However, we do not see any genuine merit in a propositional representation of states; the simplest way to represent beliefs it to allow state variables to have more than just the two values *true* and *false*. We will therefore not only allow ternary state variables (with values *true*, *false* and *unknown*), but *n-ary state variables*, meaning that a state variable v must be assigned exactly one of its n possible values in any given state. Among others, Geffner(2000) uses the same concept and gives an extended formal description and justification.

For example, in our Rescue domain the state variable `(pos F)` could have any of the values `Loc0`, `Loc1`, `Loc2`, `Loc3` or the new "default" value `unknown` that is a possible value for each state variable. Our new CWA will then be that every state variable the value of which is not specified in a state (or cannot be deduced otherwise) is believed to be `unknown.`

Note that a compilation approach similar to the one of (Gazen and Knoblock 1997) is still possible: every n-ary state variable can be compiled down to a set of propositions that must be ensured to be mutually exclusive. This ensurance is implicit in the definition of n-ary state variables and thus gives domain designers a natural way to describe important invariants of a domain, for example that an object can only be at one location at a time or may have only shape or color.

Definition 1 *A* planning domain *is a tuple* $D = (T, O, V, type)$ *where* T *is a set of types,* O *a finite set of objects,* V *the set of state variables.* $type : O \cup V \rightarrow T$ *assigns a type to each object and state variable.* $dom : V \rightarrow \mathcal{P}(O)$ *with* $dom(v) := \{o \in O | type(o) = type(v)\} \cup \{unknown\}$ *gives the possible values for state variable* v*. A* state variable assignment *is a pair* $(v, o) \in V \times dom(v)$*, also written* $(v = o)$*.*

Temporal model

Quantitative models of time are necessary to describe exact temporal relations between actions of differing duration. Level 3 of PDDL 2.1 provides a simple, yet expressive means to model durative actions. However, the time-point semantics for plans proposed in (Fox and Long 2002) is overly restrictive. In forcing planners to assign exact time points to every action in a plan it takes away the execution flexibility offered by plan semantics based on action order. Sequential, Graphplan-like ordered, or partially ordered plan semantics can easily deal with action durations that are unknown (in general or to a specific agent) because they offer qualitative notions of time like "after" or "before". MAPL is an approach to take the best of both worlds and combine quantitative and qualitative models of time. The key idea is to give up the time-point semantics for plans and go back to ordering constraints among events, but to make these constraints more flexible than those of total or even partial-order planning. Precisely, the temporal component of a MAPL plan corresponds to a Simple Temporal Network (Dechter *et al.* 1991) the constraints of which are intervals describing the temporal relation among events (instantaneous state changes). Note that in lieu of the term *action* we use the more neutral *event* here to reflect that state changes are not necessarily actively brought about by an agent but can also be observations of "natural" changes in the environment.

Definition 2 *An* event[1] e *is defined by two sets of state variable assignments: its* preconditions $pre(e)$ *and its* effects $eff(e)$*. For assignments* $(v = o)$ *in the preconditions [effects] of an event we will also write* $(v == o)$ $[(v := o)]$*.*

Relating events by ordering (i.e. temporal) constraints is central to partial-order planning (but is also implicit in classical time-step based planning). To allow for a quantitative model of time, we will extend each constraint with an interval expressing the possible variation in two events' temporal distance.

Definition 3 *A* temporal constraint $c = (e_1, e_2, I)$ *associates events* e_1, e_2 *with an interval* I *over the real numbers, describing the values allowed for the temporal distance between the occurrence times* t_{e_1} *and* t_{e_2} *of the events:* (e_1, e_2, I) *is satisfied iff* $t_{e_2} - t_{e_1} \in I$*.* I *can be open, closed or semi-open.*

```
(:durative-action Move_F_Loc2[Loc1_R12]
  :parameters (?a - agent ?dst - place)
  :duration (:= ?duration (interval 2 4))
  :condition (and
       (at start (== (pos F) Loc1))
       (at start (== (connection Loc1 Loc1) R12))
       (at start (clear R12))
  :effect (and (at start (:= (pos ?a) R12))
               (at end (:= (pos ?a) Loc2))))
```

Figure 4: Instantiated Move action

Using intervals, we can express that the duration of an action is undetermined that an agent is ignorant of it. The main advantage of the interval constraints, however, is that we can express *quantitative* relations in a quantitative manner: "e_x occurs *after* e_y" is expressed by the constraint $(e_x, e_y, \mathbb{R}^+)$; "e_x occurs *at the same time as* e_y" by $(e_x, e_y, [0, 0])$. To give qualitative descriptions of concrete, quantitative constraints we will use the abbreviation $(e_1 \prec e_2) \in C$ for the expression $\forall I.\ (e_1, e_2, I) \in C \rightarrow I \subseteq \mathbb{R}^+$, i.e. e_1 occurs sometime before e_2. $(e_1 \preceq e_2) \in C$ is defined similarly for sub-intervals of $\mathbb{R}_0^+$.

With such constraints we do not need definite time points any more: all that is important to describe a plan is the *relations* among the actions and events. As usual in partial-order planning, the initial state can be represented by a special event e_0 such that constraints with e_0 can be seen as absolute times. However, in MAP, there may be a different initial event for every agent. To be able to synchronize on absolute times if necessary, we can (but need not) assume a common clock. It is modeled as a special event e_{tr}, the temporal reference point, also called the Big Bang event because it lies before all other events and is thus the point where time starts. All agents know e_{tr} and thus can describe *absolute times* as constraints with e_{tr}.

Definition 4 *A* durative action *is a tuple* $a = (e_s, e_e, I, e_{inv})$ *where* e_s, e_e *are events (called the start and end event),* $I \subseteq \mathbb{R}^+$ *is an interval representing the temporal constraint* (e_s, e_e, I) *of the form* $e_s \preceq e_e$*, and* e_{inv} *is an event with* $eff(e) = \emptyset$*, called the* invariant event*. An* instantaneous action *is a durative action* $a = (e, e, [0, 0], e_{inv})$ *where* $pre(e_{inv}) = eff(e_{inv}) = \emptyset$*. For a set of actions* Act*,* E_{Act} *denotes the set of start and events of actions in* Act*.*

It is clear that when only using instantaneous actions and constraints of the form $(e_x, e_y, \mathbb{R}^+)$ between them, we come back to partial-order plans. On the other hand, when using durative actions with constraints of the form $(e_x, e_y, [d, d])$, i.e. exact durations and delays, we will create PDDL 2.1 plans. Thus, MAPL subsumes both partial-order and PDDL plans.

Before describing the semantics of MAPL plans we will introduce two more concepts describing events: the first,

[1]In this paper we assume *ground* events and actions. Instantiation of actions schemas (Fig. 4) includes instantiation of the state variable schemas (like `pos(?a)`) as well. When a state variable is used functionally, i.e. it represents its value in a given state (like `(pos ?a)` in `(connection (pos(?a) ?p))`, instantiation implies creation of ground actions for every possible value $o \in dom(v)$. There, v is replaced by o and $(v == o)$ is added to the preconditions.

control, allowing planners to distinguish between endogenous and exogenous events, the second, mutual exclusiveness (or, relatedly, read-write locks) describing events that must not occur concurrently.

Control

There are two kinds of durative actions: those in which duration is controlled by the executing agent (e.g. reading a book) and those in which the environments determines the duration (e.g. boiling water). In the former case, the agent (or its corresponding planner) can *choose* the delay from start to end event, in the latter case the end event may happen *at any time* during the interval given by the constraint. For any set of actions Act_a of an agent a we assume there is a *control function* $c_a : E_{Act} \rightarrow \{a, env\}$ describing whether the agent or the environment controls the occurrence time of an event. As agents can normally decide at least the start time of an action we assume that $c_a(e) = a$ for start events e_s.

When multiple planners communicate and share parts of their plans, a planner has to store for each event in a plan the executing agent controlling the event. As each planner will plan for at least one agent, the control concept is a natural way to model which events the planner can influence and how. Durations of actions where both start and end events are controlled by the planner (i.e. executing agents associated with the planner) can be manipulated in the limits of the constraining interval. Actions in which only the start event is controlled by the planner can at least be added or removed from the plan at will. Actions and events not under control of the planner cannot simply be removed from the plan; that would be self-deception because removal would not prevent their occurence. Their occurence must be taken into account during planning and plans should be valid for every possible duration in the limits of the constraining interval. (Similar, but more sophisticated concepts are developed in (Vidal and Fargier 1999; Tsamardinos *et al.* 2002).)

Mutex events and variable locks

Concurrency is a key notion in MAS. In Multiagent Planning it appears at two levels: as concurrent actions in a *plan* (or distributed over several plans by different agents) and as concurrent *planning*. Both levels are closely related: concurrency conflicts at the plan level must be detected and resolved during planning. For the plan level we define:

Definition 5 *Two events are* mutually exclusive (mutex) *if one affects a state variable assignment that the other relies on or affects, too.* $mutex(e_1, e_2) :\Leftrightarrow$
$(\exists (v\!:=\!o) \in \mathit{eff}(e1)\ \exists (v,o') \in pre(e_2) \cup \mathit{eff}(e_2)) \vee$
$(\exists (v\!:=\!o) \in \mathit{eff}(e2)\ \exists (v,o') \in pre(e_1) \cup \mathit{eff}(e_1))$

This definition corresponds to mutex concepts in single-agent Planning, e.g. in PDDL 2.1 or Graphplan(Blum and Furst 1997). From a Distributed Systems point of view, however, the mutex definition describes a *read-write lock* on the state variable v that will prevent concurrent access to the same resource v because this may lead to indeterminate values of v. Interestingly, the correspondence between mutual exclusive events and locks on state variable is more visible in a formalism like MAPL that, by the use of non-boolean state variables, seems to be a step closer to "imperative" distributed programming than the more declarative style of STRIPS and PDDL in which the state variable concept is hidden behind the Closed World Assumption and ADD/DEL effects instead of state variable updates.

In the next section, we will use the mutex definition to describe non-interference in concurrent plans. In another paper we introduce the related concept of state variable *responsibility* among agents to solve lock/mutex conflicts during distributed *planning* (Brenner 2003).

Plans

Definition 6 *A* multiagent plan *is a tuple* $P = (A, E, C, c)$ *where* A *is a set of* agents, E *a set of events,* C *a set of temporal constraints over* E*, and* $c : E \rightarrow A$ *is the* control function *assigning to each event an agent controlling its execution.*

We can now start to describe when a plan is valid, i.e. executable. We will split this definition into two aspects: temporal validity, meaning that there are no inconsistencies among temporal constraints in the plan, and logical consistency, meaning that no actions do logically interfere or are disabled when they shall be executed in the plan.

To simplify the next definitions we assume the set C of temporal constraints to be always *complete*, i.e. $\forall e_1, e_2 \in E \exists I.\ (e_1, e_2, I) \in C$. This is no restriction because we can assume C to contain the trivial constraints $(e, e, [0, 0])$ for all events $e \in E$ and $(e_1, e_2, (-\infty, \infty))$ for unrelated events $e_1 \neq e_2$.

Definition 7 *A set of temporal constraints C is* consistent *if* $\neg \exists e_1, e_2, \ldots, e_n.(e_1 \prec e_2) \in C \wedge (e_2 \prec e_3) \in C \wedge \cdots \wedge (e_n \prec e_1) \in C$*. A multiagent plan* $P = (A, E, C, c)$ *is* temporally consistent *if* C *is consistent.*

This is a reformulation of the consistency condition for Simple Temporal Networks (STNs) (Dechter *et al.* 1991) as (E, C) is in fact an STN[2]. Using the Floyd-Warshall algorithm (Cormen *et al.* 1992), consistency of an STN can be checked in $O(n^3)$. In planning, new events and constraints are repeatedly added to a plan while consistenty must be kept. To check this, we have developed an incremental variant of the algorithm (omitted from this paper) that checks for consistency violations caused by a constraint newly entered into the plan. This algorithm is in $O(n^2)$ (for every addition of a constraint).

Definition 8 *A multiagent plan* $P = (A, E, C, c)$ *is* logically valid *if the following conditions hold:*

1. *No mutex events* $e', e'' \in E$ *can occur simultaneously:*
$\forall e', e'' \in E.mutex(e', e'') \rightarrow (e' \prec e'') \in C \vee (e'' \prec e') \in C$

For any assignment $(v == o)$ *in the precondition of any event* $e \in E$ *there is a* safe achieving event $e' \in E$*:*

[2]We are aware that STN consistency is not adequate for plans with uncontrollable action durations. We are working to integrate the concept of *dynamic controllability* into our framework (Vidal and Fargier 1999).

2. $(e' \prec e) \in C \wedge (v := o) \in \mathit{eff}(e)$ *(achieving event)*
3. $\forall e'' \in E\ \forall (v := o') \in \mathit{eff}(e'').\ o' \neq o \ \rightarrow\ (e'' \prec e') \in C \ \vee\ (e \prec e'') \in C$ *(safety)*

Conditions 2 and 3 define plans as valid if there are no open conditions and no unsafe links, an approach well-known from partial order planning(Nguyen and Kambhampati 2001; Weld 1994). Condition 1 (similarly used in GraphPlan(Blum and Furst 1997)) describes threats caused by conflicting effects that do not necessarily cause unsafe links. This happens especially when events violate invariants of durative actions.

Definition 9 *A* planning problem *for an agent a is a tuple $Prob_a = (Act, c_a, e_0, e_\infty)$ where Act is a set of actions, c_a is the control function for Act, and e_0, e_∞ are special events describing the initial and goal conditions.*

We will now define when a plan solves a problem. We do not need to and cannot use happening sequences like PDDL 2.1 because of MAPL's plans being partially ordered. Instead we will reduce the question to a check for temporal and logical validity of a new plan that is obtained as a combiniation of the problem with the solution plan.

Definition 10 *A multiagent plan $P = (A, E, C, c)$ is* valid *if it is both temporally consistent and logically valid. A plan P is a* solution *to a problem $Prob_a = (Act, c_a, e_0, e_\infty)$ of agent a if the following conditions are satisfied*

1. *c is consistent with c_a: $c_a(e) = x \ \rightarrow\ c(e) = x$ and $\forall (e_s, e_e, I, e_{inv}) \in Act.$*
   $[c(e_e) = a \ \rightarrow\ \forall (e_s, e_e, I') \in C.\ I' \subseteq I] \ \wedge$
   $[c(e_e) = env \ \rightarrow\ \forall (e_s, e_e, I') \in C.\ I' = I]$
2. $\forall (e_s, e_e, I, e_{inv}) \in Act.$
   $e_s \in E \ \rightarrow\ (e_e \in E \wedge e_{inv} \in E) \ \wedge$
   $(e_s \prec e_{inv}) \in C \wedge (e_{inv} \prec e_e) \in C$
3. *for $C' = C \cup \bigcup_{e \in E} \{(e_0, e, \mathbb{R}^+), (e, e_\infty, \mathbb{R}^+)\}$*
   $P' = (A, E \cup \{e_0, e_\infty\}, C', c)$ is valid.

In words these conditions can be described as follows:

(1) the plans uses actions controlled by the agent in the way they are specified in the problem: the agent controlling an event is the same in the problem and in the plan; only actions in which the planner can control start and end event can be tightened during planning (complete control).

(2) durative actions and their invariants are used as expected: for each action appearing in a plan, its start, end, and invariant event must all appear in the plan as well as constraints describing their appearance in the natural order: $e_s \prec e_{inv} \prec e_e$. Note that no "pseudo" time points must be associated with invariants but that it suffices to have constraints forcing them to hold anytime between the start and end events.

(3) executing the plan in the initial state reaches the goals. Though looking simple, this last condition is the most important: when initial and goal events are added to the plan with constraints describing that the initial event (goal event) happens before (after) all others in the plan, then temporal and logical validity of the resulting plan signifies that the plan solves the problem.

Note that the solution plan is not required to contain only actions from Act: a plan can solve an agent's problem even if it contains not a single action of that agent!

Speech acts as synchronizing events between plans

An agent using a fact in his plan need not know how, why or by whom it has been achieved. In temporally uncertain domains the agent must even plan not knowing *when* exactly the fact will become true. To enable planning under these different kinds of ignorance, we will allow agents to use different kinds of possibly virtual *reference events* in their plans. As the same event may appear in plans of different agents this provides an implicit coordination among those plans while still allowing the knowledge about causal or temporal links of the event with others to vary largely from agent to agent.

A basic reference event that we will only briefly mention here is e_{tr}, the temporal reference point lying before all other events. All agents know e_{tr} and thus can describe *absolute times* as constraints with e_{tr}.

For MAP it is most important that agents can coordinate and exchange knowledge about the domain and their plans. This can be done with *communicative events* (i.e. speech acts). For now, we propose only the simple communicative act of the form $\text{TELL}_{v,o}$ with $pre(\text{TELL}_{v,o}) = \{(v, o)\}$ and $\mathit{eff}(\text{TELL}_{v,o}) = \emptyset$ and its counterpart $\text{TOLD}_{v,o}$ with $pre(\text{TOLD}_{v,o}) = \emptyset$ and $\mathit{eff}(\text{TOLD}_{v,o}) = \{(v, o)\}$.

By entering new information into the current plan with TOLD agents can use it like any effects of other events: as preconditions of new actions and as temporal reference in constraints. It is the latter use that is especially helpful: the TOLD event provides automatic synchronization with another agents plan. E.g. fig. 3 shows how the fire brigade synchronizes on the police clearing a road without knowing when or how this is done. Only the minimum of information necessary for coordinated action is communicated. This is important both for privacy reasons and to keep individual knowledge bases conveniently small.

Having communication explicitly anchored in the plan has several advantages. First and foremost, "being told something" is one of the simplest means for modeling "observations" of world changes not brought about by an agent himself. This way, we do not need complex semantics for information gathering or conditional plan execution.

For the speaking agent, the communicative act represents a commitment to inform the other of a specific fact during execution. It is not enough, for example, that a police agent promises to clear a road *during planning*, but that also the fire agent somehow has to be informed *during execution* that this promise has been realized. Anchoring the speech act in the plan thus is a "physical" representation of the link between the commitment made during planning and its fulfillment during execution.

During distributed planning this means, on the other hand, that plans synchronized by speech acts also commit the agents to coordinate changes to their plans. If, for example, the police agent decides at some point during planning that he must revise his decision to clear R13, the TELL event will remind him to inform the fire brigade of this change. The speech act can thus represent a distributed backtracking point, a concept similarly used in Distributed CSP solving (Yokoo and Hirayama 2000).

The basics for a distributed planning algorithm using speech acts both to exchange missing information and to synchronize are presented in (Brenner 2003).

Conclusion and future work

We have presented basic concepts for the Multiagent Planning Language MAPL, an extension of PDDL that supports planning for and by Multiagent Systems. MAPL's temporal model can be used to describe exact, quantitative temporal relations as well as flexible, quantitative ones. It might therefore be useful not only for multiagent scenarios but for every domain where execution flexibility is important after planning has been completed. MAPL's use of non-boolean state variables makes it easier for domain designers to describe basic invariants like "an object can have only one location at a time". It also sheds some light on the relation between mutually exclusive actions in Planning and similar concept in Distributed Computing like read-write locks on variables.

We have defined temporal and logical validity of MAPL plans as well as what it means to solve a specified planning problem. As, in contrast to PDDL 2.1, MAPL plans are partially ordered we cannot and do not need to define happening sequences or induced simple plans for MAPL plans. This also avoids associating invariants with "pseudo" time points.

In another paper (Brenner 2003), we present the first single agent and distributed planning algorithms for MAPL domains. These algorithms are as preliminary as the definition of MAPL's syntax and semantics. Exciting future work is possible now: we are currently working on a parser and a small domain suite to test both the expressivity of the language and the powers and limits of our algorithms.

MAP has been a topic of interest in AI for quite some time. However, not much work has been published, neither in the field of Multiagent Systems (MAS) nor in Planning; furthermore, what has been published is mostly stand-alone work that has not led to a steady development in MAP research. In our view, this is due to an unfavorable separation of the (single-agent) planning phase and the (multi-agent) coordination and execution phase, resulting in AI Planning researchers concentrating mostly on the former and MAS researchers almost exclusively dealing with the latter. This separation is only possible with strong assumptions that narrow the generality of the proposed approaches, for example the assumption in MAS research that the actual planning of each agent can either be handled by classical single-agent planning methods or is eased by a given hierarchical task decomposition. The AI planning community, on the other hand, has only recently fully acknowledged the need for sophisticated models of concurrent plan execution (earlier exceptions include most notably work by M. Ghallab(Ghallab and Laruelle 1994)). MAPL is an attempt to show possible extensions of PDDL 2.1's representation in a way that allows flexible execution *after* and easy coordination *during* the planning process. We hope that our representation will allow to conveniently describe largely differing MAP domains for which researchers can propose and cross-evaluate very different algorithmic approaches, thus promoting the field of Multiagent Planning.

References

Avrim Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2), 1997.

Craig Boutilier and Ronen Brafman. Partial order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research*, 2001.

Michael Brenner. Multiagent planning with partially ordered temporal plans. In *Proc. IJCAI*, 2003.

T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1992.

Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49, 1991.

Maria Fox and Derek Long. *PDDL 2.1: an Extension to PDDL for Expressing Temporal Planning Domains*, 2002.

B. Gazen and C. Knoblock. Combining the expressiveness of UCPOP with the efficiency of Graphplan. In *Proc. ECP '97*, 1997.

H. Geffner. Functional STRIPS: a more flexible language for planning and problem solving. In Jack Minker, editor, *Logic-Based Artificial Intelligence*. Kluwer, 2000.

M. Ghallab and H. Laruelle. Representation and control in IxTeT, a temporal planner. In *Proc. of AIPS '94*, 1994.

H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjoh, and S. Shimada. RoboCupRescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *Proc. 1999 IEEE Intl. Conf. on Systems, Man and Cybernetics*, 1999.

XuanLong Nguyen and Subbarao Kambhampati. Reviving partial order planning. In *Proc. IJCAI '01*, 2001.

Ioannis Tsamardinos, Thierry Vidal, and Martha Pollack. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints Journal*, 2002.

Thierry Vidal and Hélène Fargier. Handling contingency in temporal constraint networks. *Journal of Experimental and Theoretical Artificial Intelligence*, 11, 1999.

Daniel Weld. An introduction to least commitment planning. *AI Magazine*, 15(4), 1994.

Makoto Yokoo and Katsutoshi Hirayama. Algorithms for distributed constraint satisfaction: a review. *Autonomous Agents and Multi-Agent Systems*, 3(2), 2000.