Multiagent Planning with Partially Ordered Temporal Plans

Technical Report

Michael Brenner
Institut für Informatik
Universität Freiburg
79110 Freiburg, Germany
brenner@informatik.uni-freiburg.de

April 6, 2003

Abstract

This paper discusses the specifics of planning in multiagent environments. It presents the formal framework MAPL ("maple") for describing multiagent planning domains. MAPL allows to describe both qualitative and quantitative temporal relations among events, thus subsuming the temporal models of both PDDL 2.1 and POP. Other features are different levels of control over actions, modeling of agents' ignorance of facts, and plan synchronization with communicative actions. For global planning in multiagent domains, the paper describes a novel forward-search algorithm producing MAPL's partially ordered temporal plans. Finally, the paper describes a general distributed algorithm scheme for solving MAPL problems with several coordinating planners. The different contributions intend to provide a simple, yet expressive standard for describing multiagent planning domains and algorithms that in the future might allow cross-evaluation of Multiagent Planning algorithms on standardized benchmarks.

1 Introduction and Related Work

In this paper, we discuss the specific properties of planning in Multiagent Systems (MAS). With the term Multiagent Planning (MAP), we will denote any kind of planning that happens 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 neither assume cooperativity nor competition among agents, nor do we impose any relation among planning and executing agents: 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*.

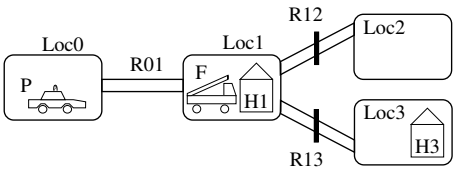

Figure 1: A multiagent planning problem

As a motivating example, Fig. 1 shows a simple MAP problem as it might appear in the RoboCupRescue challenge [Kitano *et al.*, 1999]. Two autonomous agents, police force P and fire brigade F, are working in a city devastated by an earthquake. While F's goal is to extinguish all burning houses, it P has both the capability and the goal to clear the blocked roads. P's position being Loc0 we will assume him unaware of teh fact that R12 and R13 are blocked. The agents' actions have specific durations that may be exactly known only at execution time, sometimes because of specific execution parameters of the agents, sometimes because of intrinsic unpredictability of the environment: while, for example, moving through the town may take between 2 and 4 minutes and depend only on the map distance and speed of the agent, extinguishing a fire may take 1 to 4 hours depending on conditions unknown to the agents. So an agent has some degree of control about the duration of a *move* action, while the duration of *extinguish* can only be guessed.

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) Agents may be *unaware* of parts of the world state (P does not know whether R13 is blocked). (2) *Concurrent acting* is central to MAP (P can move to Loc1 and start clearing R13 while F is extinguishing H1, although both agents using the same road at the same time may be prohibited to avoid collisions). Modeling concurrency necessitates (2a) a description of which events may occur concurrently and which not, (2b) *metric time* for realistic descriptions of action durations and their relations, but (2c) 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 is (3) synchronization on *communicative acts*, as in "F moves to Loc3 after P has informed him that R13 is clear".

In their plans, agents must take other agents actions into account: F may "exploit" P's clearing of R13 in his own plan, but must also assure that he does not try to use a road that is also used by P at the same time. Especially, (4) cannot *control* occurence or duration of other agents' actions.

While many recent planning formalisms allow some degree of concurrency, all fail in providing features for either quantitative (2b) or qualitative (2c) temporal reasoning. PDDL 2.1, for example, supports metric time but the plan semantics enforces planners to assign exact time stamps and durations to all events [Fox and Long, 2002] which might be impossible in dynamic multiagent systems. In contrast, the concurrency model of Boutilier and Brafman[2001] augments partial order plans with concurrency, thus allowing flexible, synchronized execution, but cannot describe, for example, that many sequential *move* actions (maximum duration 4) could executed in parallel with one *extinguish* action (minimum duration 60). None of the planning models known to us features property (3), implicit plan synchronization with communicative acts.

Different degrees of control about actions in a plan where extensively studied in [Vidal and Fargier, 1999; Morris *et al.*, 2001]. However, to our knowlegde the models developed describe only single-agent settings (+ a nondeterministic environment.) This paper is a first step to extending the controllability framework to multiagent domains, and to integrate it into a formal planning language.

To address the representation problems (1)–(4) we introduce the Multiagent Planning Language MAPL ("maple"). Fig. 2 shows part of a MAPL description for the Rescue domain.

Instead of propositional state representations MAPL allows non-boolean state variables (cf. also [Geffner, 2000]). To model ignorance (1) each state variable may have the special value *unknown*, thereby avoiding representation of belief states as sets of possible states. A number of other advantages comes with the introduction of state variables; especially, for feature (2a), an intuitive definition of mutual exclusivity (i.e. the impossibility to execute some actions concurrently, cf. [Blum and Furst, 1997]) can be given that describes mutexes as *read-write locks* on state variables. According to this perspective, distributed planning can then be seen as detection or, even better, prevention of possible read-write locks *before* execution.

```
(: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))
  :control (start: ?a) (end: ?a)
  :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

MAPL's temporal model allows to combine (2b) quantitative and (2c) qualitative temporal information in plans, thus subsuming both the purely metric temporal model

of PDDL 2.1 [Fox and Long, 2002] and the purely qualitative model of Partial Order Planning [Weld, 1994]. At its core MAPL represents a multiagent plan as a Simple Temporal Network [Dechter *et al.*, 1991] in which each durative action is modeled by start and end events, possibly extended by invariant conditions. In the STN, both action durations and qualitative ordering relations are treated as constraints represented by closed, semi-open or open intervals. In so doing, not only can imprecisely known action durations be represented as intervals of the form $[\delta_1, \delta_2]$, but qualitative constraints like "after" can be described by (semi-)open intervals like $(0, \infty)$.

Another central concept of MAPL is the use of communication as action (3). This means that speech acts can be directly put into a plan and are treated like other actions. Such communicative actions serve as reference events for plan synchronization. This kind of synchronization is implicit in the sense that no distinction is made between qualitative or quantitative temporal constraints for ordering an agent's own actions and constraints used for inter-agent coordination.

This treatment of communication (or, more generally, information gathering) is beneficial in many ways. It allows agents to refer to facts (especially those achieved by others) the change of which they do not influence or witness themselves. We can even treat an agent's perceptions as "speech acts by the environment". Another benefit, as explained later, is that speech acts allow agents to reveal only the minimum of information about their plans needed for coordination, thus keeping as much privacy about their knowledge, goals, and plans as possible.

A plan is only fully specified with (4) a *control function* describing which agent (or the environment) controls the occurence of each event. With this function we can describe, e.g., that a specific agent is allowed to add and remove an action from his plan (control of the start event), but has no influence on it its duration (end event controlled by the environment). During planning, having control of an event or not fundamentally changes its possible use and evaluation. For example, being able or unable to control the duration of an action will lead the planner to a fundamentally different heuristic evaluation of its use.

For a plan to be executable, it must be both temporally and logically consistent. The former criterion is reducible to consistency of the underlying STN. The latter, logical consistency, can be defined similarly to POP as the plan having no open conditions and no unsafe links, with the additional criterion that the plan must ensure that no mutex events may occur concurrently. For a plan to solve a certain agent's problem it must achieve his goals and also be consistent with the control function, i.e. only constraints involving events controlled by the respective agent must have been tightened by the planner.

How is planning in MAS carried out? It is obvious that the easiest way is to find a plan alone: assumed that F knows about P's capabilities, F can find a plan that solves his problems. Even if F does not know about P's concrete actions, this plan will provide clues about where help is needed and thus triggers cooperation. We see that (5) the capability for single-agent synthesis of multiagent plans is a basic requirement for MAP. We have developed a plan-space forward-search algorithm that can be used with any standard forward branching scheme and arbitrary plan metrics. This paper presents two such metrics, the well-known makespan and the new minMaxMakespan, the latter of which extends the former by assigning maximal possible duration to uncontrolled durative actions. As MAPL extends both PDDL2.1 and Partial Order Planning the

forward-branching scheme can be used not only to generate valid MAPL plans, but also for PDDL and PO plans.

Heuristic forward planning in the style of FF [Hoffmann and Nebel, 2001] can be extended to find MAPL plans. The current simple algorithm is sound, but not complete, i.e. there is a set of clearly distinguished MAPL problems it cannot solve yet. We are working on a sound, yet more complex version of the algorithm.

When several agents are planning and acting individually in a common environment, they will probably run into one of the following problems: (6) They won't be able to find individual plans solving their problems or (7) the plans found will conflict at execution time. MAP Literature has mostly treated only problem (7), implicitly assuming that plans can be found and that therefore separating planning from coordination is possible ([Tonino *et al.*, 2002], [Ephrati and Rosenschein, 1994]). Distributed hierarchical planning ([Durfee, 1988; Durfee and Montgomery, 1991]) is a special case thereof where the set of possible solutions is already implicitly given in an abstraction hierarchy that is refined in a distributed manner. While in our opinion coordination *during* the planning phase is indispensable for dealing with problem (6), it is at least advantageous for coping with problem (7), i.e. when individually valid plans are conflicting (e.g. when two ambulance teams plan to rescue the same refugee).

We have therefore developed a general distributed planning algorithm that uses single-agent planning to synthesize partial plans and to trigger cooperation and coordination efforts as early as possible. The algorithm is integrates asynchronous distributed search techniques used in Distributed Constraint Satisfaction resarch [Yokoo and Hirayama, 2000] with modern heuristic-search based Planning techniques [Hoffmann and Nebel, 2001]. Our MAP algorithm is reactive in the sense that new events (like communication or exogenous events) can be integrated into the current search state easily so that changes of knowledge do not necessarily enforce replanning. The algorithm thus allows distributed, continual planning [desJardins *et al.*, 2000].

A key concept of our distributed planning techniqeu is the use of a *responsibility function* that assigns to each state variable an agent managing and controlling its changes over time. This agent will detect read-write conflicts in the agents' plans. i.e. possible execution conflicts, but will also provide information when another agent cannot achieve a (sub)goal involving that variable. In the basic form of the algorithm, the responsibility is static, but similar to approaches in Distributed CSP research [Yokoo and Hirayama, 2000] we will relax this assumption in future work. The idea of the algorithm is simple: in a reachability analysis the planning agent detects goals involving state variables he does not know about, cannot manipulate or could if only some earlier condition were satisfied. He contacts the responsible agents to receive more information or delegate a subgoal concerning the variable. The responsible agent answers the question or adopts a temporary goal to help the asking agent.

In the remainder of the paper, we will formally describe the Multiagent Planning Language MAPL as an extension of PDDL (section 2). Section 4 describes single-agent (or central) search techniques in the space of MAPL plans. Section 5 show a distributed planning algorithm based on the concept of state variable responsibility. In section 6 we conclude with our vision for the future of Multiagent Planning research.

2 Multiagent problems and plans in MAPL

2.1 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* $(v = o)$.

2.2 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 $\mathit{eff}(e)$. *For assignments* $(v = o)$ *in the preconditions* [*effects*] *of an event we will also write* $(v{=}{=}o)$ $[(v{:=}o)]$.

```
(:durative-action Move_F_Loc2[Loc1_R12]
  :parameters (?a - agent ?dst - place)
  :duration (:= ?duration (interval 2 4))
  :control (start: ?a) (end: ?a)
  :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 3: Instantiated Move action

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.*

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*

[1] In this paper we assume *ground* events and actions. Instantiation of actions schemas (Fig. 3) 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.

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* $\mathit{eff}(e) = \emptyset$*, called the* invariant event*. An* instantaneous action *is a durative action* $a = (e, e, [0,0], e_{inv})$ *where* $\mathit{pre}(e_{inv}) = \mathit{eff}(e_{inv}) = \emptyset$*. 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, 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.

2.3 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.

Its a fundamental change to the semantics of a plan whether we have control of an event or not. Sophisticated models of different kinds of control are developed in [Vidal and Fargier, 1999; Morris *et al.*, 2001]. At present, we have integrated only a very much simplified version of this framework, but have focussed on extending it to a multiagent setting and to integrate it with MAPL's other features.

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.

2.4 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.* $mutex(e_1, e_2) :\Leftrightarrow$
$(\exists (v:=o) \in \mathit{eff}(e1)\ \exists (v,o') \in \mathit{pre}(e_2) \cup \mathit{eff}(e_2)) \vee$
$(\exists (v:=o) \in \mathit{eff}(e2)\ \exists (v,o') \in \mathit{pre}(e_1) \cup \mathit{eff}(e_1))$

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 section 5.1 we introduce the related concept of state variable *responsibility* among agents to solve lock/mutex conflicts during distributed *planning*.

2.5 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 inconsisten-

cies 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. 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. $(e' \prec e) \in C \wedge (v := o) \in \mathit{eff}(e)$ *(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!

3 Communicative actions

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). The following approach to communication treats speech acts just like other actions: they have preconditions and effects, and once they have been added to a plan they can be referenced in qualitative and quantitative constraints.

At present we propose only one simple communicative act, TELL, which is used to communicate values of state variables from one agent to another one. As state variables cannot be parameters of MAPL actions, we use parameterized speech acts of the form TELL_v. TELL_v is defined as an instantaneous action $\text{TELL}_v(A, B, o)$ where A is the speaker, B the listener, and o the value of state variable v that is communicated.

The conditions and effects of a speech act are different for speaker and listener. For example, we want to ensure that the speaker believes in the state variable assignment

she is going to communicate by making this assignment a precondition. However, for the listener who cannot control the speech act there is no apparent precondition: the communicative event just occurs! To emphasize these different viewpoints and semantics, we will use TELL_v to denote the speech act in the speaker's plan, while writing TOLD_v in the listener's plan. The full definitions therefore are:

$$\begin{aligned} \mathit{pre}(\text{TELL}_v(A, B, o)) &= \{\ (v == o)\ \} \\ \mathit{eff}(\text{TELL}_v(A, B, o)) &= \emptyset \end{aligned}$$

$$\begin{aligned} \mathit{pre}(\text{TOLD}_v(A, B, o)) &= \emptyset \\ \mathit{eff}(\text{TOLD}_v(A, B, o)) &= \{\ (v == o)\ \} \end{aligned}$$

After adding new information into the current plan with TOLD agents can use the new information like any effect of other events: as preconditions of new actions and as temporal reference in constraints.

Having communication explicitly anchored in the plan in that manner brings 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, but may describe learning new knowledge as the effect of a knowledge gathering action.

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.

In this section, we have described the semantics of MAPL domains, problems and plans. In the following sections we will see how such plans can be synthesized by a single agent (section 4) or by multiple agents (section 5).

4 Single agent search for multiagent plans

In the following we look at a basic capability of agents: to plan alone (if possible). To that end we develop a forward search algorithm on partially ordered temporal plans.

A minimal condition for a plan to satisfy a set of goals is that every goal is achieved at some point in the plan and is not removed again after that point. Instead of testing goal satisfaction, we will use this condition to describe if actions can be added to a plan "at the end".

For a given plan $P = (A, E, C, c)$ we will use the following abbreviation: $achieves(e, (v,o)) :\Leftrightarrow (v,o) \in \mathit{eff}(e) \wedge$
$\forall e' \in E \setminus \{e\}\ \forall (v,o') \in \mathit{eff}(e').\ o' \neq o \ \rightarrow\ (e' \prec e) \in C$

Definition 11 *The* frontier F_P *of a plan* $P = (A, E, C, c)$ *is the set of achieved state variable assignments*
$F_P = \{(v,o) \mid \exists e \in E.\ achieves(e, (v,o))\}$

Corollary 1 *For each assignment* $(v,o) \in F_P$ *in a valid plan* P *there is a unique* achiever $e_{(v,o)}$ *with* $achieves(e_{(v,o)}, (v,o))$.

Definition 12 *The set of* enabled actions *for a given plan* P *and a set of possible actions* Act *is*
$enabled_{P,Act} = \{(e_s, e_e, I, e_{inv}) \in Act \mid pre(e_s) \subseteq F_P\}$

Enabled actions can be added to the plan "at the frontier", i.e. after all events achieving their preconditions. But there is potential for conflict: even if no event changes an assignment after the achiever, events later in the plan might "read" that assignment, i.e. it appears in their preconditions although not in their effects. If the newly added action changes the assignment it threatens the "reader" events. E.g. actions *extinguish* and *move* can both be enabled in plan frontier $F_P \ni (isAt(F) = Loc1)$, but both can only be applied by first apply *extinguish* and then add *move* after the reader event *extinguish*. By doing so, we automatically prevent the plan from becoming invalid as *extinguish* and *move* are mutex.

Algorithm 1 apply(a,P)

$E' := E \cup \{e_s, e_e\ e_{inv}\}$
$C' := C \cup \{(e_s, e_e, I), (e_s, e_{inv}, \mathbb{R}^+), (e_{inv}, e_e, \mathbb{R}^+)\}$
for all assignments $(v,o) \in pre(e_s)$ **do**
  $readers := \{e \in E \mid (v,o) \in pre(e)\}$
  if $(readers = \emptyset)$ or $(\neg\exists (v := o') \in \mathit{eff}(e_s) \cup \mathit{eff}(e_e))$ **then**
    // there are no readers of (v,o) *or* a *is a reader itself*
    $C' := C' \cup \{(e_{(v,o)}, e_s, \mathbb{R}^+)\}$
  else
    $C' := C' \cup \bigcup_{e \in readers} \{(e, e_s, \mathbb{R}^+)\}$
return $P' = (A, E', C', c)$

Algorithm 1 enters an enabled action a into a plan at the earliest possible position that causes no safety or mutex threats. Open conditions cannot be produced either[2] because the old plan was valid and the new action was enabled, i.e. preconditions satisfied. Without proof, we can therefore state:

Theorem 2 *If* P *is a valid plan and* $a \in Act$ *is enabled in* P*, i.e.* $a \in enabled_{F_P, Act}$ *then* apply(a,P) *returns a valid plan.*

[2]We assume $pre(e_e) \subseteq pre(e_s) \cup \mathit{eff}(e_s)$ and $pre(e_{inv}) \subseteq pre(e_s) \cup \mathit{eff}(e_s)$. Any other choice would make the semantics of durative actions problematic, e.g. non-terminating durative actions would be possible.

We can now describe single-agent algorithms that search for (multi-agent) plans in the space of plans (like POP algorithms): for every state (=plan), *enabled* gives us a set of possible transitions (=actions) and *apply* describes the transition function from one state to a successor state. That means we can use any state-space search technique to find valid plans. Fig. 4 shows a plan for the Rescue example found by this algorithm with a central planner controlling all (executing) agents.

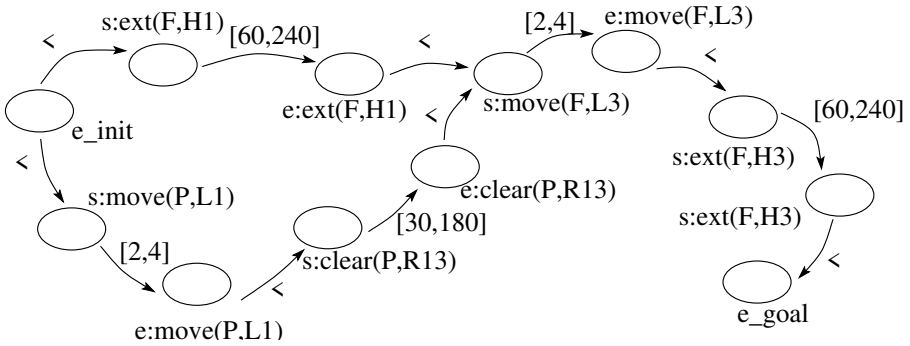

Figure 4: A centrally synthesized multiagent plan

The simplest method, Breadth First Search, produces plans with minimal number of actions. For concurrent temporal plans this is unintuitive because concurrency and action duration is ignored. A better criterion for plan quality is *makespan* (minimal duration of the execution) of a plan. The calculation of a plan's makespan is a side product of the consistency check with the Floyd-Warshall algorithm: makespan corresponds to the length of the longest path in the plan assuming minimal possible duration for durative actions. An even more reasonable quality metrics for MAP is to assume *maximal* duration for *uncontrolled* actions. We will call this metrics *min-max makespan* and denote with $mmdur(a)$ the minimal (maximal) duration of a controlled (uncontrolled) action.

Quality metrics are used in a search method by sorting the search queue according to the metrics. This corresponds to A^* search with heuristic function 0. However, we need not do without heuristics. Goal distance can be estimated in various ways; the following method is based on the FF[Hoffmann and Nebel, 2001] heuristic.

To relax a planning problem we assume that assignments to state variables made anywhere in a plan remain true throughout the plan. Algorithm *r-apply(a,P)* is derived from *apply(a,P)* by simply not checking for "readers" and adding the new action after all its preconditions are achieved, i.e. by adding $(e_{(v,o)}, e_s, \mathbb{R}^+)$ to C for all $(v, o) \in pre(e_s)$. But Corollary 1 does not hold for relaxed plans: when mutex events are allowed there is not necessarily a *unique* achiever $(e_{(v,o)}$ for an assignment. To be sure to find the relaxed plans with minimal (min-max) makespan we must make sure that the *earliest achiever* is used when constraints with the precondition achievers of a new action are added to the plan. This can be accomplished by building the relaxed plan with a simple algorithm that uses Uniform-Cost Search (UCS) when adding new relaxed actions at the relaxed frontier.

4.1 Using relaxed plans to trigger coordination

The "state" a relaxed plan P_P^R is built on is the frontier of a non-relaxed (partial) plan P. This way, the relaxed plan can be used to provide a heuristic evaluation of P. The min-max makespan of the relaxed plan including its non-relaxed prefix can then be used as an admissible heuristic for the goal distance of plan P. However, the heuristic is not very informative (the reason being that unnecessary actions in a plan can usually be executed concurrently with necessary ones and thus will not increase the makespan). We are currently experimenting with variants of this heuristic in combination with different forward search algorithms based on the techniques presented in this section.

For the purpose of this paper we can disregard the exact heuristic function for relaxed plans and use only one basic information it provides: the relaxed (un-)solvability of a problem. Like FF, our relaxed planning algorithm terminates if either no more new actions are applicable in P or if the goals are achieved at the *relaxed frontier* which is defined $F_P^R = \{(v, o) | \exists e \in E.\ (v, o) \in \mathit{eff}(e)\}$. It is clear that if a relaxed plan cannot be found then the non-relaxed problem is also unsolvable. In single-agent planning (e.g. [Blum and Furst, 1997], FF[Hoffmann and Nebel, 2001]), this property is used to prove a problem unsolvable and terminate search. In MAP, we will interpret it as a trigger for cooperation with other agents!

While our algorithm is proven sound by theorem 2, it is, in this simple form, not complete, i.e. for some MAPL problems the algorithm does not find a solution although one exists. The solutions missed can be characterized and found by an extended algorithm that we are currently developing. Discussion of this issue will be left to another paper.

5 Multiagent planning

In a multiagent environment, non-coordinating agents will probably run into one of the following problems: (1) they won't know how to solve their planning problem or (2) they will find a plan but run into execution conflicts (mutex or safety threats) with other agents' plans. In a competitive environment this might be acceptable or even intended; in all other cases it is not. For the rest of the paper we will therefore assume agents willing to coordinate at least as much as to produce a set of conflict-free plans to make plan execution predictable. This does not mean that one common, conflict-free plan must be built. The method proposed is intended to minimize synchronization of both planners and plans.

5.1 Responsibility for state variables

We will tackle the problem of unsolvability and the problem of conflicting plans with the same concept: responsibility for state variables. The agent responsible for a state variable is the one who is asked when another cannot reach a goal involving that variable. And the responsible agent will decide who is allowed to manipulate a variable when actions planned by two agents use a variable in a conflict-producing way.

Currently, MAPL and our algorithmic framework are only at the beginning of development. For now, we therefore make strong simplifying assumptions about responsibility:

(1) Responsibility for state variables is described by a function $resp : V \rightarrow A$. (2) The function $resp$ is fixed during the planning process. (3) The agent $resp(v)$ and only he must know controlled actions that manipulate v. (4) The agent responsible for a state variable v knows the initial value of v.

In future work, these assumptions will be relaxed as follows: (1) We will allow a set of responsible agents negotiating about changes to the variable. (2) We will allow dynamic change of the responsible agent to better reflect who really is "using" the state variable during the course of the planning process. (3) (4) will both not be enforced any more, but be an effect of dynamic change of the responsible agent.

5.2 Access histories

The responsible agent records the changes to "his" state variable in an *access history*. Thus he can give agents accessing the variable informations about the changes and they can create plans consistent with the access history.

Definition 13 *An* access history $H_v = (A, E_v, C_v, c_v)$ *is a plan where all preconditions and effects* $e_v \in E_v$ *use only assignments of one state variable* v.

H_v *is an access history* for *a plan* P *iff there is a bijective mapping* m *from* E_v *to the set* $\{e \in E | \exists o.(v, o) \in pre(e) \cup \mathit{eff}(e)\}$ *such that for all* $e_v, e'_v \in E$*:*

$(v, o) \in pre(e_v) \leftrightarrow (v, o) \in pre(m(e_v)) \wedge$

$(v, o) \in \mathit{eff}(e_v) \leftrightarrow (v, o) \in \mathit{eff}(m(e_v)) \wedge$

$(e_v, e'_v, I) \in C_v \rightarrow (m(e_v), m(e'_v), I) \in C \wedge$

$c_v(e_v) = c(m(e_v))$

By $H_{v;write}$ $[H_{v;read}]$ *we denote the sub-plan of* H_v *that contains only events* e *where* $\mathit{eff}(e) \neq \emptyset$ $[pre(e) \neq \emptyset]$.

5.3 Synchronizing on reference events

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 implicit synchronization among those plans. On the other hand, the causal or temporal relationships of the reference events can be kept private and will be different in the different agents' plans.

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 achieved with *communicative events*, i.e. speech acts. For now, we propose only the simple communicative act $\text{TELL}_{v,o}$ (cf. 3).

Having added the new information to the current plan as an effect of TOLD agents can use this knowledge in the usual ways: as preconditions of new actions and as temporal reference in constraints. It is the latter use that will especially helpful in the following algorithm: the TOLD event provides automatic synchronization with another

agents plan. E.g. fig. 5 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.

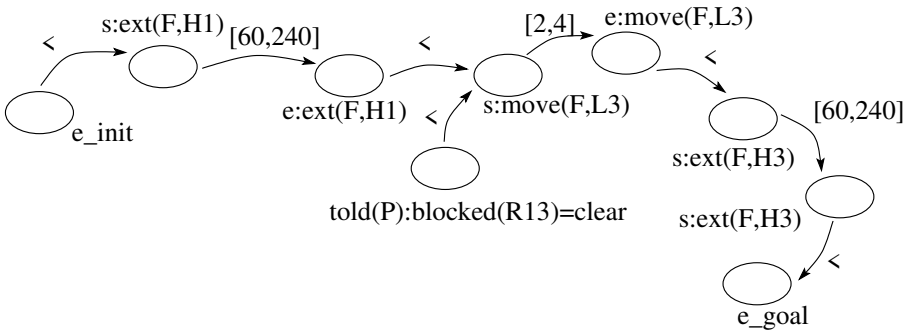

Figure 5: F's plan (including a communicative action by P)

5.4 Distributed planning

The basic distributed planning algorithm for MAPL is shown in alg. 2 and 3. Alg. 2 shows the reactions of a planning agent to various messages and his behavior when no messages are received. Alg. 3 describes the behavior of an agent responsible for a variable, though of course an agent may have both roles (or even responsibility for more than one variable). To simplify presentation, the algorithms are described using the nondeterministic *choose* operator that also defines backtrack points. These are beginning points for future research: how can the nondeterministic choices be heuristically guided?

The idea of the algorithm is simple: by building a relaxed plan, the planning agent detects goals involving state variables he does not know about, cannot manipulate or could if only some earlier condition were satisfied. He contacts the responsible agents to receive more information or delegate a subgoal concerning the variable. The responsible agent answers the question or adopts a temporary goal to help the asking agent. For this simplified version of the algorithm, we assume that the responsible agent can always help. In general, this is of course not true, and when the responsible agent fails to help asynchronous backtracking [Yokoo and Hirayama, 2000] is triggered; but this is out of the scope of this paper.

There are three possibilities for termination: either an agent can backtrack no more, or an agent not responsible for a variable has found a plan, or a responsible agent gets information that all other agents have terminated or found a plan so that he can terminate, too, without "shirking" his responsibility to inform other agents about his variable.

Algorithm 2 MA Planning (agent a, current partial Plan P)

Received no message:
  $P^R :=$ relaxed plan upon current plan frontier F_P
  $unachievable = \{(v,o) \in pre(e_\infty) | (v,o) \notin F_P^R\}$
  if $unachievable \neq \emptyset$ **then**
    choose $(v,o) \in unachievable$
    investigate(v,o)
  else if found Plan P **then**
    if $\exists v$ accessed in P for which $H^r_{v;write}$ is unknown **then**
      send ask(v) **to** $resp(v)$
    else if $H^r_{v;write}$ is consistent with P $\forall v$ accessed in P **then**
      send tell(H_v) **to** $resp(v)$
      output P
      terminate (if not responsible for a state variable)

Received tell($H^r_{v;write}$) **from** agent $r = resp(v)$:
  choose (v,o) for which $\exists e \in H^r_{v;write}.\ (v,o) \in \mathit{eff}\ (e)$
  apply(P,$\text{TOLD}_{v,o}$)

Received unachievable(v,o) **from** agent $r = resp(v)$:
  backtrack

Procedure investigate(v,o):
  if $(v == unknown) \in F_P^R$ **then**
    send ask(v) **to** $resp(v)$
  else if $\neg\exists a \in Act.\ (v,o) \in \mathit{eff}(e_s) \cup \mathit{eff}(e_e)$ **then**
    send askFor(v,o) **to** $resp(v)$
  else
    choose $(v',o') \notin F_P^R$ by regression on (v,o)
    investigate(v',o')

Algorithm 3 MA planning (agent r responsible for v)

Initialize H_v^r with $E_v := \{e_{tr}, e_0^v\}$ and $C_v := \{(e_{tr}, e_0^v, \mathbb{R}^+)\}$

Received no message:
  for all $(v, o, a) \in asked$ **do**
    if $\exists e \in H^r_{v;write}.\ (v, o) \in \mathit{eff}(e)$ **then**
      $asked := asked \setminus \{(v, o, a)\}$
      $pre(e_\infty) := pre(e_\infty) \setminus \{(v, o)\}$
      send tell($H^r_{v;write}$) **to** a
    if all agents (including r) have found a plan or terminated **then**
      terminate

Received ask(v) **from** agent a:
  send tell($H^r_{v;write}$) **to** a

Received askFor(v,o) **from** agent a:
  if $\exists e \in H^r_{v;write}.\ (v, o) \in \mathit{eff}(e)$ **then**
    send tell($H^r_{v;write}$) **to** a
  else if $(v, o) \in F_P^R$ **then**
    $asked := asked \cup \{(v, o, a)\}$
    $pre(e_\infty) := pre(e_\infty) \cup \{(v, o)\}$
  else
    send unachievable(v,o) **to** a

Received tell(H_v^a) **from** agent a:
  find the mapping m from $H^a_{v;write}$ to $H^r_{v;write}$
  forall $c^a \in C_{H_v^a}$ **do** $C_{H_v^r} := C_{H_v^r} \cup \{m(c^a)\}$

6 Conclusion and future work

We have presented a new representation for MAP domains and solutions along with basic single- and multi-agent planning algorithms. The algorithms have been implemented in Java; agents are modeled as threads communicating via pipes. The implementation is preliminary, in particular, we are still in development of a parser to automatically read in problems. At the moment, only hand-coded toy problems (like the Rescue example) are solved by the planners.

A lot of exciting work is possible now: we are already experimenting with different heuristics and search methods on all levels of the algorithms. Dynamic responsibility hierarchies will improve flexibility and performance of the algorithm (comparably to dynamic agent hierarchies in Asynchronous Weak-Commitment Search [Yokoo and Hirayama, 2000]). We will also develop a set of benchmark problems and apply the distributed algorithm in a realistic application (RobocupRescue).

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 that has lead AI Planning researchers to concentrate exclusively on the former while MAS researchers almost as exclusively deal with the latter. This separation is only possible with strong assumptions that narrow the generality of the proposed approaches, for example the assumption that actual planning of each agent

can either be easily done before coordinating in a "classical" way or is eased by a given hierarchical task decomposition. It seems obvious to us that such a separation is artificial: planning and coordination should be interleaved to enable agents both to find plans at all and to detect probable execution conflicts of their (partial) plans as early as possible.

With PDDL 2.1, the AI planning community has only recently fully acknowledged the need for sophisticated models of concurrency (earlier exceptions include most notably work by M. Ghallab[Ghallab and Laruelle, 1994]). MAPL is an attempt to extend that representation in a way that allows flexible execution *after* and easy coordination *during* the planning process. The single- and multi-agent algorithms we propose are only first steps and hopefully not the only possible ways to synthesize MAPL plans. 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

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

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

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

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

[desJardins *et al.*, 2000] M. desJardins, E. Durfee, Jr. C. Ortiz, and M. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 2000.

[Durfee and Montgomery, 1991] E. Durfee and T. Montgomery. Coordination as distributed search in hierarchical behavior space. *IEEE Transactions on Systems, Man, and Cybernetics*, 1991.

[Durfee, 1988] Edmund Durfee. *Coordination of Distributed Problem Solvers*. Kluwer, 1988.

[Ephrati and Rosenschein, 1994] Eithan Ephrati and Jeffrey S. Rosenschein. Divide and conquer in multi-agent planning. In *Proc. AAAI-94*, 1994.

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

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

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

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

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14, 2001.

[Kitano *et al.*, 1999] 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.

[Morris *et al.*, 2001] Paul H. Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal uncertainty. In *Proc. IJCAI '01*, 2001.

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

[Tonino *et al.*, 2002] Hans Tonino, André Bos, Mathijs de Weerdt, and Cees Witteveen. Plan coordination by revision in collective agent based systems. *AIJ*, 142(2), 2002.

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

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

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

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