# Planning for Multiagent Environments

## From Individual Perceptions to Coordinated Execution

**Michael Brenner**
Institute for Computer Science
Albert-Ludwigs-University
79110 Freiburg
Germany
brenner@informatik.uni-freiburg.de

**Abstract**

In this paper, we discuss the particular characteristics of planning for multiagent systems, and present a rich formal model for describing features like concurrency, individual and mutual beliefs of agents, acting under incomplete knowledge, control, perception, and communication. Our model allow agents to execute their individual plan fragments as autonomously as possible while provably guaranteeing synchronized behavior where necessary. Synchronization can be achieved by as different methods as communication, metric or quantitative temporal constraints, or copresence. We show the semantic relation of multiagent plans to classical plans, and informally describe a sound and complete variant of a POCL algorithm for multiagent planning.

## 1 Introduction

In this paper, we discuss the particular characteristics of planning for multiagent systems, and present a rich formal model for describing features like concurrency, individual and mutual beliefs of agents, acting under incomplete knowledge, control, perception, and communication. While previous research has acknowledged most of these characteristics to be relevant for multiagent planning (MAP), we are not aware of prior work modelling and integrating all of them and, above all, giving them a clear formal semantics that can be used to prove properties of both plans and planning algorithms.

Our model allow agents to execute their individual plan fragments as autonomously and flexibily as possible while provably guaranteeing synchronized behavior where necessary. Synchronization can be achieved by as different methods as communication, metric or quantitative temporal constraints, individual perception or copresence.

The purpose of this article is unusual in so far that it does not contain algorithmical or empirical results, but "only" provides a thorough discussion of MAP characteristics and, consequently, a thoroughly defined logical model that allows, e.g., to prove that a multiagent plan can by executed by multiple agents without further coordination or external synchronization. We believe that only based on such formal qmodels, there can be theoretical, algorithmical, and empirical progress in MAP. As one tool for this development we have designed a variant of PDDL for the semantics defined in this article. A parser and several sample domains (including the one presented in the next section) will be made available for download.

The next section will motivate and discuss the concepts formalized in the remainder of the article. At the end of the paper, we also sketch a sound and complete algorithm for planning in our formalism.

## 2 Motivation

**Example 1** *A person wants to visit a friend. The friend's house can only be entered once its door has been opened.*

Consider the scenario described in Example 1. We can model it using the simple STRIPS-like operators given below. If we assume $closed \wedge outside \wedge \neg atHouse$ as the initial state of the world (intuitively stating that the door is closed and that the visitor has not yet reached the house) the following is a valid STRIPS plan for the scenario: $\langle$*move2house; open; enter*$\rangle$.

| **action** | **precondition** | **effect** |
|---|---|---|
| move2house | $outside \wedge \neg atHouse$ | $atHouse$ |
| open | $closed$ | $\neg closed$ |
| enter | $outside \wedge atHouse \wedge \neg closed$ | $\neg outside$ |

As the reader may have noticed, we have tried to obfuscate an important aspect of the scenario both in the verbal and the formal description, namely *who* is performing which action. In fact, in classical STRIPS-like planning there is no direct way to model the *agent* of an action (even if telling action names suggest a specific reading). This is unproblematic for Classical Planning which assumes centralized control of plan execution, but for MAP one must at least distinguish the different capabilities of different agents. Most prior MAP formalizations have recognized this and allow actions to be associated with an *executing* or *controlling agent*. For our example, let us assume that the visitor $x$ can move to the house and enter it, but that only her friend $y$ can open the door. Then, most existing MAP formalisms would accept the following as a valid plan for the scenario (where each action is annotated with the controlling agent): $\langle move2house_x; open_y; enter_x \rangle$.

However, the main point we will elaborate in the rest of the paper is that this plan may actually not be executable by the two autonomous agents! The reason is that the plan

constrains two autonomous agents to a specific temporal order of actions, but does not guarantee that they can actually *synchronize* their behavior accordingly. Note that this is not merely a problem of total-order (TO) vs. partial-order (PO) plans: even a less constrained PO representation like $\{move2house_x \prec enter_x, open_y \prec enter_x\}$ demands agent $x$ to synchronize her *enter* action with a previous *open* action by another agent. How is this synchronization achieved?

The example is so simple that the solution seems obvious: $x$ *knows* when she can enter the house because she can *perceive* the door to be open once she is outside the house. However, semantically there is no relation between the proposition $outside$ describing where $x$ is and the proposition $closed$ describing what she is supposed to perceive. There is not even a distinction between the door being open and $x$ being aware of it. The key to synchronization thus lies in modeling not only the state of the world, but also the beliefs that agents may have about it. Since facts can not only be believed to be true or false, but also *unknown*, we use multi-valued state variables in our framework instead of propositions. This has the additional advantage of leading to smaller state spaces. In our scenario, the propositions *outside* and *atHouse* could be fused to one state variable $loc_x$ with possible values *inHouse, nearHouse,* and *elsewhere* (plus *unknown* in case of beliefs), thereby eliminating the combination $\neg atHouse \wedge \neg outside$ which is meaningless in our scenario.

Actually, many existing MAP formalisms, e.g. SharedPlans (Grosz & Kraus 1996), also employ beliefs and even mutual beliefs among agents. However, to the best of our understanding these concepts are only used there to describe beliefs *about* a plan (e.g. mutual beliefs about the joint commitment to the plan), but not for describing how (mutual) beliefs about the world develop and change *in* the plan. The most basic way to change one's beliefs is through *perception*. (In Section 5 we will also describe communication.) We model perceptions by *sensor rules* that are automatically triggered when the necessary conditions are satisfied. The key realization underlying this approach is that perception is not a consequence of one single action, but is an event emerging from, firstly, something happening and, secondly, somebody being there to watch it. In this sense, perception is a special case of concurrency. Indeed, we will use other kinds of *domain rules* to describe non-trivial effects of concurrent events, as in the following variation of our scenario:

**Example 2** *Person $y$ lives in a multi-storey building where she can operate the entrance door by a buzzer. $x$ can only enter the house while the door is temporarily unlocked by the buzzer. Furthermore, $x$ must ring the doorbell first to notify $y$ that she is there.*

This example is fairly common in reality and introduces a number of new aspects. Firstly, actions must be performed concurrently in this scenario to achieve a nontrivial joint effect, namely $x$ must push the door *while* the buzzer is activated to cause the door to swing open. Boutilier and Brafman (2001) show how such concurrent interacting actions can be modelled using concurrency constraints and special conditional effects. The authors note that post-planning synchronization will be necessary to ensure the concurrency constraints are obeyed by the executing agents. Since in our model the plan itself is intended to guarantee synchronization, we present an alternative model here which we call the *physical forces approach* to concurrency. The underlying idea is that action have only individual effects that do not directly interact, but which in combination may create a kind of "force" or "instability" that causes a natural event according to the *causal rules* of the domain. Fig. 1 shows the Causal Domain Rule (CDR) of our scenario in PDDL-like syntax. The rule is modeled as an action caused by the unique agent env representing the environment.

```
(:action swing-open
 :agent env
 :precondition (and (doorstate = pushed)
                    (buzzer))
 :effect (doorstate = open))
```

Figure 1: Causal Domain Rule

When synchronization by means of sensing is not possible, an alternative may be to agree on absolute time points for action execution. This is just one reason for including metric time in our model. For most practical problems featuring concurrency it is important to reason not only about whether some actions can be parallelized, but also about the quantitative relation between their durations. However, we also want to describe flexible or unknown durations; to ensure synchronized behavior in that case of uncertainty we must be able to reason about the qualitative relations between events. The temporal model used in this paper allows to describe both qualitative and quantitative temporal relations between events.

**Example 3** *The door to $y$'s house often stands open. If this is the case when $x$ arrives she can simply walk in instead of ringing.*

One major reason for the difficulty of multiagent planning is the high dynamics of MAS and, consequently, the many facts that may be unknown at *planning time* – even if the planner in question centralizes knowledge of several executing agents. Usually, however, many things unknown at planning time will become perceivable to at least one executing agent at *execution time*. This fact can be exploited by a planner, since perception models form an explicit part of our model, and plans can thus include actions for *active knowledge gathering*. However, since the actual perceptions to be made are unknown at planning time, *conditional action execution* must be possible depending on the outcome of the perception. In Ex. 3, $x$ must first move to the entrance of $y$'s house to perceive the state of the door (open or locked) before she can decide whether she can simply walk in or must ring first. A multiagent plan for Ex. 3 is shown in Fig. 2; the sensor rule describing the circumstances under which the agent can perceive the state of the door is shown in Fig. 3 in the extended PDDL syntax we have defined for our model.

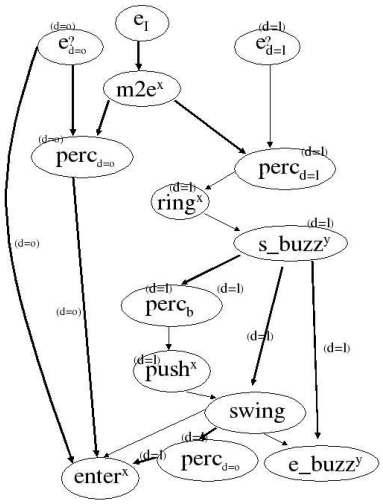


Figure 2: A multiagent plan for scenario 3.
For clearness, temporal constraints and facts supported by causal links have been omitted. Labels $d = l$ ($d = o$) denote the door being initally locked (open). Events labeled $perc$ are perception rules, the CDR $swing$ corresponds to the door swinging open when being pushed and kept unlocked (by buzzing) simultaneously.

## 3 Integrating Agency with Planning

In this section, we will describe how basic notions of agency can be integrated into a Planning formalism. We will, however, not attempt a definition of what constitues an "agent". Instead, an unspecified set of agents $\mathcal{A}$ will be the basic building block for all further definitions. $\mathcal{A}$ is always assumed to include the unique agent $\mathsf{env}$, the **environment agent** with special characteristics described in Sec. 5.

Some necessary components of agency (like beliefs and capabilities for sensing and acting) will be defined and attributed to agents $a \in \mathcal{A}$. For this we use a function $agt(x) := a$ where $x$ can be any such component. We will use the index notation $x_a$ to denote $agt(x) = a$ and also extend this notation straightforwardly to sets.

**Facts, beliefs, and mutual beliefs** Instead of the propositional representation used in most Planning formalisms, our model uses **non-boolean state variables** (cf. (Bäckström & Nebel 1995; Helmert 2004) for a discussion of the SAS$^+$ formalism where this extension is borrowed from). There are several reasons for this design choice:

1. Multi-valued state variables occur naturally in most planning domains. For example, the positions of an agent $a$ can be encoded by one state variable $loc_a$ with a set of possible values, the **domain** of $loc_a$, $dom_{loc_a}$. Not only becomes modeling such domains considerably easier, also the size of the state space can often be dramatically reduced (Helmert 2004). Moreover, since propositions are state variables over the domain $\{\mathsf{true},\mathsf{false}\}$, propositional planning formalisms like STRIPS, ADL and PDDL are subsumed by state-variable model, anyway. Syntactically, compatibilty with propositional planning can be maintained by allowing the notations `(prop)` and `(not (prop))` instead of `(prop = true)` and `(prop = false)`.
2. *Beliefs* of agents can straightforwardly be modeled, without the need for a possible world semantics, by simply allowing state variables to assume a specific additional value *unknown*[1]. We will call such variables **belief state variables**.
3. Distributed Systems are usually modelled by means of private and shared variables. Classical concepts like *read-write conflicts* or variable *locks* can be easily recognized in multiagent plans when using a state variable representation. For example, the Classical Planning concept of *mutually exclusive* propositions (Blum & Furst 1997) can then be expressed as *read-write locks* between state variables. This shift in perspective is helpful especially when applying Distributed Algorithms concepts to Multiagent Planning (Brenner 2003).

Let $\mathcal{V}$ be a set of **state variables**, each $v \in \mathcal{V}$ with an associate finite **domain** $dom_v$. A **partial variable assignment (PVA)** over $\mathcal{V}$ is a function $s$ on some subset of $\mathcal{V}$ such that $s(v) \in dom_v$ wherever $s(v)$ is defined. $\mathsf{undef}_s$ is the set of **undefined variables** in $s$. If $s(v)$ is defined for all $v \in \mathcal{V}$, $s$ is called a **state**. If $s(v)$ is defined (with value $x$) then the pair $(v, x)$ is called an **assignment** (also written $v \doteq x$). Two PVAs $s$ and $s'$ are called **consistent** if the following holds: if both $s(v)$ and $s'(v)$ are defined then $s(v) = s'(v)$.

STRIPS, PDDL, and other languages based on propositional logic use sets of propositions where our model (due to having non-boolean variables) must use PVAs. To make this relation easier to see, we will often use set notations for PVAs, too, e.g. we write $(v \doteq x) \in pre_e$ instead of $pre_e(v) = x$, and denote the completely undefined PVA by $\emptyset$. In particular, we define the **union** of two *consistent* PVAs $s_1$ and $s_2$ as the PVA $s = s_1 \cup s_2$ in which if $s_1(v) = x$ or $s_2(v) = x$ then also $s(v) = x$.

For a given agent $a \in \mathcal{A} \setminus \{\mathsf{env}\}$, a set $\mathcal{V}$ of state variables induces a set of **belief state (BS) variables** $\mathcal{V}_a$ where for each $v \in \mathcal{V}$ there is a $v_a \in \mathcal{V}_a$ with $dom_{v_a} = dom_v \cup \{\mathsf{unknown}\}$. The function $agt$, defined as $agt(v_a) := a$, returns the **owner** of a BS variable. The environment $\mathsf{env}$ does not have beliefs; to keep some of the following definitions simple, we define $\mathcal{V}_{\mathsf{env}} := \mathcal{V}$ and $agt(v) = \mathsf{env}$ for $v \in \mathcal{V}$.

We can further generalize this concept to beliefs shared among subgroups of $\mathcal{A}$: the sets $\mathcal{V}$ and $\mathcal{A}$ induce a set of **mutual belief state (MB) variables** $\mathcal{V}_\mathcal{A}$ where for each $v \in \mathcal{V}$ and each subgroup $A \subseteq \mathcal{A}$ there is a $v_A \in \mathcal{V}_\mathcal{A}$ with $dom_{v_A} = dom_v \cup \{\mathsf{unknown}\}$. The definition of MB variables includes mutual belief among a singleton set of agents which is equivalent to individual belief. Thus the definition

[1] It must be noted that some beliefs can be expressed within a possible world semantics, but not in our model, in particular constraints *between* state variables. For example, the constraint $loc_a \doteq x \leftrightarrow loc_b \not\doteq x$ could describe that no two agents can be believed to be at the same position at the same time. Often, though, such constraints can be modeled by introducing complementary state variables, for example $occupant_x$ would describe who is standing at position $x$ and could have value $a$ or $b$ (plus some dummy $\mathsf{unoccupied}$), but not both.

of MB variables subsumes the one for BS variables of individual agents. For convenience, however, we will keep the distinction between individual and mutual beliefs, and also continue to use the notation $v_a := v_{\{a\}}$ for individual beliefs. Since furthermore $\mathcal{V}_{\mathsf{env}} = \mathcal{V}$, the PVAs over $\mathcal{V}_{\mathcal{A}}$ enumerate all possible states, beliefs, and mutual beliefs for a given domain[2]. A PVA $s$ is **knowledge consistent** if all mutual beliefs correspond to the facts, i.e. they are actually **common knowledge**. Formally, a PVA $s$ is knowledge consistent if $s(v_A) = x$ implies that also $s(v) = x$ for all variables $v$ and all $A \in \mathcal{A}$. In particular, knowledge consistency implies **MB consistency**, i.e. if $s(v_A) = x$ then $s(v_{A'}) = x$ for all $A' \subseteq A$.

## 4 Modelling Multiagent planning domains

We can now define what constitutes a MAP domain. While the definition contains many agent-specific particularities that will be explained in the rest of this section, it was nevertheless designed to be compatible wherever possible with PDDL 2.1 (Fox & Long 2003). Roughly, our definitions extend PDDL 2.1, level 1 and 3, and by "compatibility" we mean that there is a large class of domains that are both MAP and PDDL domains. The main difference, apart from the notions of agency described in the previous sections, is a temporal model that allows more "qualitative" relations between events than the solely metric time model of PDDL 2. For a similar treatment of time, see (Younes & Simmons 2003). We have discussed the importance of a "qualitative" temporal framework in addition to a quantitative one like PDDL 2 in (Brenner 2003).

**Definition 1** *A **multiagent planning domain** is a tuple $\mathcal{D} = (\mathcal{A}, \mathcal{V}, \mathcal{E}, \mathcal{O})$ where*

- *$\mathcal{A}$ is the set of **agents***
- *$\mathcal{V}$ is the set of **state variables**, each $v \in \mathcal{V}$ with an associate finite **domain** $dom_v$*
- *$\mathcal{E}$ is the set of **events**, each $e \in \mathcal{E}$ of the form $e = (a, pre, \mathit{eff})$ where*
  - *$a \in \mathcal{A}$ is the **controlling agent***
  - *$pre$ is a knowledge consistent PVA over $\mathcal{V} \cup \mathcal{V}_a$ called the **precondition***
  - *$\mathit{eff}$ is a knowledge consistent PVA over $\mathcal{V} \cup \mathcal{V}_{\mathcal{A}}$ called the **effect** of $e$.*
- *$\mathcal{O}$ is the set of **processes**, each $o \in \mathcal{O}$ of the form $o = (a, e^s, e^e, \Delta, inv)$ where*
  - *$a \in \mathcal{A}$ is the **controlling agent***
  - *$e^s \in \mathcal{E}$ (with $agt(e^s) = a$) is the **start event***
  - *$e^e \in \mathcal{E}$ (with $agt(e^e) \in \{a, \mathsf{env}\}$) is the **end event***
  - *$inv$ is a PVA over $\mathcal{V} \cup \mathcal{V}_a$ called the **process invariant***
  - *the interval $\Delta \subseteq \mathbb{R}^+$ is called the **duration range** of $o$*

[2]The reader will note that we only define individual and mutual beliefs, but do not attempt to model arbitrary nested beliefs (like "A believes that B believes that C believes that x"). Firstly, this would lead to an infinite number of nested belief variables. Secondly, nested beliefs (other than mutual beliefs) almost never seem to play a role in multiagent behavior. Thirdly, if, for specific problems or domains, beliefs nested to some finite level were needed, the model could easily be extended.

**Events** Roughly, events corresponds to instantaneous actions in PDDL. We use the neutral term "event" to hint at the fact that what for one agent constitutes an action that she can execute at will is an uncontrollable event for another.

Events differ from classical actions in having *knowledge preconditions* and *knowledge effects*. Interestingly, knowledge preconditions are more restricted than knowledge effects. The reason for this is that the controlling agent can only refer to her own beliefs when checking whether she can execute an action. In contrast, we allow agents to *change* the beliefs of other agents directly, at least in principle: this is the most basic way to model *communication*, i.e. actions with knowledge effects can be regarded as speech acts.

We have already seen that for an action to be executable by an agent $a$ not only must its usual preconditions be satisfied, but the agent $a$ must also know about it[3]. We therefore demand the following for all $e_a \in \mathcal{E}$: if $(v \doteq x) \in pre(e_a)$ then also $(v_a \doteq x) \in pre(e_a)$. For effects, we will enforce a similar constraint: if $(v \doteq x) \in \mathit{eff}(e_a)$ then also $(v_a \doteq x) \in \mathit{eff}(e_a)$. The meaning, however, is somewhat different: an agent will know when it has executed an action and therefore will believe in its effects to have occured. Of course, both kinds of constraints can be automatically computed and need not be specified explicitly.

**Processes** Processes are similar to durative actions in PDDL, but must be extended for MAP with the notion of **control** which was introduced by Vidal and Fargier (1999) and is extended to the multiagent case here. Control describes the kind of influence that an agent has on a process. For some process $o$ where $a = agt(e^s)$ we say that $a$ has **occurrence control** over $o$. If, additionally, $a = agt(e^e)$ then $a$ also has **duration control** over $o$. The key semantic difference can be illustrated by the two processes of reading a book and boiling water with a kettle: I can decide for both processes whether I want to execute them in my plan (and thus have occurence control over both), but I have duration control only of my reading the book, i.e. I can tighten duration interval $\Delta$ at will in my plan. In contrast, for boiling the water the plan must be guaranteed to work for all possible durations $\delta \in \Delta$.

The process invariant $inv$ is used just as in PDDL to describe facts that must hold throughout the whole process. To model this semantics an artifical event $e^{inv} = (\mathsf{env}, inv, \emptyset)$ will be used. $e^{inv}, e^s$, and $e^e$ together form the set $\mathcal{E}^o$ of events appearing in process $o$. The set of all events appearing in a set of operators $O$ is denoted $\mathcal{E}^O$.

To simplify our later definition of multiagent plans, we model instantaneous actions as processes, too: if $e_a \notin \mathcal{E}_{\mathcal{O}}$ then we extend $\mathcal{O}$ by $(a, e_a, e_a, \emptyset, [0, 0])$.

**Semantics of events** The semantics of events is defined exactly as in other planning formalisms: given a state $s$ and an event $e$, $e$ is **applicable** in $s$ if whenever $(v \doteq x) \in pre_e$ then also $(v \doteq x) \in s$. Applying an applicable event $e$ in a state $s$ results in state $app(s, e)$ where $(v \doteq x) \in app(s, e)$ iff $(v \doteq x) \in \mathit{eff}_e$ or $[(v \doteq x) \in s$ and

[3]Although sometimes one may want to give up this constraint, resulting in a "leap-before-you-look" approach (Golden 1998).

$v \in \mathsf{undef}_{\mathit{eff}_e}]$. The occurence of a **sequence** of events can be defined inductively in the usual manner: $res(s, \langle\rangle) := s$ and $res(s, \langle e_1, ..., e_n\rangle) := app(res(s, \langle e_1, ..., e_{n-1}\rangle), e_n)$ if $e_n$ is applicable in $res(s, \langle e_1, ..., e_{n-1}\rangle)$, otherwise $res(s, \langle e_1, ..., e_n\rangle)$ is undefined. We will later show how this Classical Planning semantics relates to our complex temporal multiagent plans.

## 5 Modeling Causal Laws of MA Systems

The events and processes controlled by the environment agent env differ from those of all other agents in one crucial aspect: the environment does not act deliberately and willfully; instead events necessarily occur according to the "physical laws" of the domain, its **causal domain rules** (CDRs). Formally the CDRs simply consist of all processes $\mathcal{O}_{\mathsf{env}}$ controlled by env. The semantic difference is the fact that preconditions of normal actions describe conditions *necessary* for an action to be exectubable, while conditions of CDRs are *sufficient* to trigger the corresponding event or process. (The concept of automatically triggered events was inspired by research in the Theory of Actions community, in particular by Thielscher (1995)).

Causal domain rules are meant to model the "laws of nature" of a domain. Whenever a rule is triggered the world is considered to be in an *unstable* state leading to an event or the start of a process which in turn removes the instability (but might create a new one). To capture that aspect and to prevent the same rule to be triggered repeatedly with infinitesimal delays, we enforce rules to destroy their own triggering conditions. $pre(e^s)$ and $\mathit{eff}(e^s)$ must be inconsistent, i.e. $e^s$ destroys one of its preconditions.

Furthermore, two rules $r_1, r_2 \in \mathcal{O}_{\mathsf{env}}$ that are triggered by the same situation could have inconsistent effects, thereby introducing nondeterminism into our model. Just as in Classical Planning we will forbid this, and formally constrain: If $\mathit{eff}(r_1)$ and $\mathit{eff}(r_2)$ are inconsistent, then $pre(r_1)$ and $pre(r_2)$ must be inconsistent, too.

Within the constraints just defined (destroying the own precondititon, no rules leading to nondeterminism), causal domain rules are a powerful tool. In particular, we can use them to naturally model interactions between concurrent actions as was demonstrated in Ex. 2. Due to lack of space, a detailed comparison to alternative models of concurrency like the one of Boutilier and Brafman (2001) will be done in an extended version of this paper.

### Perception, communication, and mutual belief

Causal domain rules are also used to describe sensor models of agents. In contrast to other "physical laws" of a domain, **perception rules** have *knowledge effects*. To simplify reasoning about perception rules we will enforce the following format for them: perception rules $r \in \mathcal{O}_{\mathsf{env}}$ must be instantaneous actions, i.e. $r = (\mathsf{env}, e, e, \emptyset, [0, 0])$. Furthermore $e$ has exactly one effect $(v_a \doteq x)$ where $a \in \mathcal{A}$ and $(v \doteq x) \in pre_e$. We call $pre_e \setminus \{(v \doteq x)\}$ the **perception condition** for $(v \doteq x)$. Usually, we can assume that perception does not depend on a specific value $x$ of $v$ and that there are corresponding rules for all $x \in dom_v$. The set of these rules is called a **sensor model** and we write $\mathsf{sensor}(a, v, cond)$ to denote that for all $x \in dom_v$ there is a rule $(\mathsf{env}, cond \cup \{(v \doteq x)\}, \{(v_a \doteq x)\})$. Fig. 3 shows how the sensor model $\mathsf{sensor}(a, doorstate, \{loc_a \doteq entrance\})$ is described in PDDL-like syntax. It specifies that an agent will perceive the state of the door (*open* or *closed*) when she is at the entrance.

```
(:sensor door-sensor
 :agent ?a
 :precondition (loc ?a = entrance)
 :sense (doorstate))
```

Figure 3: Perception rule

We have previously explained how knowledge effects can be used as an easy means to model speech acts. Additionally, we have assumed that an agent executing an action will believe its effect to be true afterwards. In combination, those premises lead to an interesting effect. Assuming that agent $a$ communicates a fact $p = (v \doteq x)$ to agent $b$, the effect $v_b \doteq x$ could be expressed as $B_b p$ in some standard epistemic logic. However, since $a$ knows this to be the effect of his action also $B_a B_b p$ will be true. We have explicitly not included such nested beliefs in our framework, but we can do something else: If me make the additional assumption (not yet explicit in the semantics) that $b$ will know who has communicated $p$ to her, she will be able to infer $B_b B_a B_b p$, which in turn $a$ may infer, etc. In short, under the assumption of perfect communication and speaker detection, our modeling of speech acts induces mutual belief. This is not surprising (Fagin *et al.* 1995), yet welcome, since it allows us to replace simple knowledge effects with mutual belief effects (among the speaker and hearer) in speech acts.

Communication is not the only way to achieve mutual belief. Another possibility, *copresence* (or coperception) was described already by Lewis (1969). Informally, agents are copresent when they are in a common situation where they can not only perceive the same things but also each other. Such a situation can lead to mutual belief since the agents can mutually infer their perceptions, the beliefs about other agents' perceptions, etc.

We can describe copresence situations as special kinds of sensor models $\mathsf{sensor}(A, v, cond)$ that have effects on a mutual belief variable $v_A$ for a group of agents $A$. A basic example could be a copresence model stating that agents achieve mutual belief about their respective locations whenever those are identical. Based on this "precursory" MB more MB can be inferred wherever a perception rule is triggered the condition of which does not only hold, but is already mutual belief. In that situation all copresent agents could infer the perceptions of the others, plus their inferences, etc. A more formal treatment of this topic will be given in a future publication where we also describe an approach to automatically deriving copresence models from individual sensor models.

## 6 Plans, Problems, and Solutions

**Definition 2** *A **multiagent plan** for a domain $\mathcal{D} = (\mathcal{A}, \mathcal{V}, \mathcal{E}, \mathcal{O})$ is a tuple $P_\mathcal{D} = (A, O, T, L, B)$ where*

- $A \subseteq \mathcal{A}$ *is the set of **agents***
- $O \subseteq \mathcal{O}$ *is the set of **operators***
- $T$ *is a set of **temporal constraints** of the form $t = (e, e', \Delta)$ where $e, e' \in \mathcal{E}^O$ and $\Delta \subseteq \mathbb{R}$.*
- $L = L^+ \cup L^-$ *is a set of **positive** and **negative causal links** of the form $l = (e, v \doteq x, e')$ where $(v \doteq x) \in pre(e')$ and*
  - *$(v \doteq x) \in \mathit{eff}(e)$ if $l \in L^+$*
  - *$(v \doteq x') \in \mathit{eff}(e)$ if $l \in L^-$ (for some $x' \neq x$)*
- $B$ *is a function labeling each event and each causal link with a PVA. It is called the **branching context**.*

This definition of MA plans is related to single-agent formalisms for conditional temporal planning (Tsamardinos, Pollack, & Horty 2000), but extends prior work with multiple agents, causal domain rules, and (mutual) beliefs. To show the relation to classical PO representations of plans, we say an event $e$ is **precedes** another one $e'$ if $(e, e', \Delta) \in T$ and $\Delta \subseteq \mathbb{R}^+$. In that case, we also write $(e \prec e) \in T$.

In the following, we will assume that in every given plan $P$ the set of constraints $T$ is complete and unambiguous, i.e. that there is exactly one constraint $(e, e', \Delta)$ for all $e, e' \in \mathcal{E}^O$. This is no restriction, but can be achieved easily by extending $T$ with $(e, e, [0, 0])$ for all events $e \in \mathcal{E}^O$ and with $(e, e', (-\infty, \infty))$ for previously unrelated events $e \neq e$. We further assume that $T$ is pairwise consistent, i.e. $(e, e', \Delta) \in T$ iff. $(e', e, -\Delta) \in T$. If $(e, e', \Delta) \in T$ and $\Delta \subseteq \mathbb{R}^+$, we say that $e$ must occur **before** $e'$ and write $(e \prec e) \in T$.

The temporal constraints $T$ of a plan $P$ form a Simple Temporal Network (STN) (Dechter, Meiri, & Pearl 1991). A basic prerequisite for giving the plan a meaningful semantics is that the underlying STN is **consistent**. Consistency can be checked in small polynomial time; cf. (Younes & Simmons 2003) for a description of how STNs can be used in a state-of-the-art single-agent planner. In the following we will assume only plans with consistent underlying STN.

Temporal constraints in a plan must not only form consistent STNs, they must also not violate the duration range defined for the processes $\mathcal{O}$ as defined in $\mathcal{D}$. The duration range, however, has a different semantics depending on who has *duration control* of a process $o$: if env controls the duration, the plan must be be valid for any possible duration in the duration range. If, on the other hand, the agent controlling the duration of $o$ is different from env the planner may tighten the duration constraints at will[4]. Formally, we define a plan $P_\mathcal{D}$ to be **process-consistent** with $\mathcal{D}$ if for all processes $o = (a, e^s, e^e, \Delta, inv)$ in $P_\mathcal{D}$, $T \supseteq \{(e^s, e^e, \Delta_1), (e^s, e^{inv}, \Delta_2), (e^{inv}, e^e, \Delta_3)\}$ where $\Delta_1, \Delta_2, \Delta_3 \subseteq \Delta$. If $agt(e^e) = \mathsf{env}$ then $\Delta_1 = \Delta$.

Following the literature on conditional planning we demand that $B$ must be defined such that labels are propagated along temporal constraints in the plan (Peot & Smith 1992; Tsamardinos, Pollack, & Horty 2000; Tsamardinos, Vidal, & Pollack 2002).

The reader may have noted that, in contrast to, e.g, PDDL 2, there is no way in our formalism to specify absolute time points for events. However, absolute time points can be described by referring to a special, mutually known **temporal reference event** $e_0$, virtually occuring before all other actions. Note, however, that in many domains exact time points will only complicate plan monitoring, since in general it cannot be determined whether a plan should still be considered correct when some event occured with a slight temporal difference to its precise scheduled time. The qualitative model we propose is thus more flexible than the metric one of PDDL 2.

We can now generalize the POCL notions of threats and open conditions to our metric temporal and conditional plan formalism.

**Definition 3** *In a plan $P = (A, O, T, L, B)$, an event $e$ has an **open condition** $c = (v \doteq x)$ if $c \in pre(e)$ and there is no causal link $l \in L$ which supports $c$, i.e. which is of the form $l = (e', c, e)$ for some $e'$. An event $e_t \in \mathcal{E}^o$ **threatens** a causal link $l = (e_p, v \doteq x, e_c) \in L$ if*

- *$e_t$ has an effect $v \doteq x'$ where $x' \neq x$ if $l \in L^+$, and $x' = x$ if $l \in L^-$*
- *$e_t$ might occur between $e_p$ and $e_c$, i.e. there exist $\Delta, \Delta' \subseteq \mathbb{R}^+$ for which $T \cup \{(e_p, e_t, \Delta), (e_t, e_c, \Delta')\}$ is consistent*
- *$e_p$ and $e_t$ occur in consistent branching contexts*

Natural events *necessarily* follow the causal rules defined for the domain. As a consequence, valid plans must not only contain actions that achieve the goals, but must also ensure that no harmful natural events can be triggered. An operator $o$ is said to **enable** a rule $r$ if it achieves some trigger condition of $r$. Formally, we define a relation $\mathsf{enables} \subseteq \mathcal{O} \times \mathcal{O}_\mathsf{env}$ where $\mathsf{enables}(o, r)$ if there exists $(v \doteq x)$ with $(v \doteq x) \in \mathit{eff}(o)$ and $(v \doteq x) \in pre(r)$. Note that $o$ might itself be a causal rule which enables another one. Note further that since there may be several trigger conditions for $r$, occurence of $o$ alone is not sufficient to actually trigger $r$.

**Definition 4** *A plan $P_\mathcal{D} = (A, O, T, L, B)$ for a domain $\mathcal{D} = (\mathcal{A}, \mathcal{V}, \mathcal{E}, \mathcal{O})$ is **closed wrt. the domain rules** $\mathcal{O}_\mathsf{env}$ if the following holds:*

- *if $\mathsf{env} \in \mathcal{A}$ then $\mathsf{env} \in A$*
- *if $\mathsf{enables}(o, r)$ then $r' \in O$ and $(o, r', \mathbb{R}^+) \in T$ (where $o \in O, r \in \mathcal{O}_\mathsf{env}$ and $r'$ is a unique copy of $r$)*[5].

In words, a closed plan contains instances of all rules that might possibly be triggered during its execution. Since rules may themselves trigger other rules, computing the closure

[4] This definition of *control* is sufficient for the situation assumed in this paper where there are basically only two *planning* (but many *executing*) agents: a centralized planner who can add and remove actions for all executing agents, and the environment agent env. Our model of *control* is, however, designed to be used also in a Distributed Planning paradigm where some planner *Planner*$_a$ is responsible but for one executing agent $a$. In that case, *Planner*$_a$ may not change the duration range of any process $o$ controlled by agents $b \neq a$. In fact, *Planner*$_a$ can not even simply remove $o$ from its current plan, since this would not force the planner controlling $o$, namely *Planner*$_b$, to do likewise.

[5] Such a copy of (or mapping to) a base action is usually called a *step*. Following most of the planning literature, we will ignore the distinction between steps and base events wherever possible.

for a given plan $P$ amounts to a fixpoint computation, i.e. $P$ is extended with the enabled rules repeatedly until a stable plan is reached[6].

A **MAP problem instance** is a tuple $\Pi = (\mathcal{D}, I, G)$ where $\mathcal{D} = (\mathcal{A}, \mathcal{V}, \mathcal{E}, \mathcal{O})$ is a MAP domain. $I$ and $G$ are knowledge consistent PVAs over $\mathcal{V}_\mathcal{A}$ called the **initial** and **goal knowledge distribution** and which, as usual, will be represented by the dummy actions $e_I = (\mathsf{env}, \emptyset, I)$ and $e_G = (\mathsf{env}, G, \emptyset)$. $I$ might be incompletely specified (although the deman for knowledge consistency at least enforces that there are no false beliefs). Therefore, a solution plan for $\Pi$ must be valid for all possible undefined values. To ensure this we define the set of possible additional initial events $\mathcal{E}_I^? := \{(\mathsf{env}, \emptyset, \{(v \doteq x)\}) \mid v \in \mathsf{undef}_I \wedge x \in dom_v\}$. All of these events will conditionally appear in the plan, labeled with their own effect, thereby defining the only branching context where they can occur.

**Definition 5** *A plan $P_\mathcal{D} = (A, O, T, L, B)$ is a **solution** to $\Pi = (\mathcal{D}, I, G)$ if*

- $O = O' \cup \{e_I, e_G\} \cup \mathcal{E}_I^?$ *where* $O' \subseteq O_A$
- $e_I \prec e \prec e_G$ *and* $e^? \prec e$ *for all* $e \in O'$ *and all* $e^? \in \mathcal{E}_I^?$
- $B(e_I) = B(e_G) = \emptyset$ *and* $B(e) = \mathit{eff}(e)$ *for all* $e \in \mathcal{E}^?$
- $T$ *is process-consistent with* $\mathcal{D}$ *and forms a consistent STN*
- $P$ *is closed wrt. the domain rules* $\mathcal{O}_\mathsf{env}$
- $P$ *contains no threatened causal links*
- *the only open conditions in* $P$ *are in rules* $r \in \mathcal{O}_\mathsf{env}$ *not supporting causal links*

This definition is a straightforward extension of what constitutes solutions in POCL planning. One major difference is the role of individual beliefs in a plan, expressed by the knowledge preconditons and effects of events. A solution plan must, in particular, not contain open knowledge conditions. Thus, the definition forces planners to make sure that knowledge necessary for synchronized actions is shared among the executing agents. Either agents must be brought into positions to perceive changes themselves or communicative actions must be previewed in a plan.

Another novelty is the role of control (embedded in the notion of process-consistency) that different agents have over different actions in the plan, and especially the role that natural events play for modelling concurrency, perception, and, generally, complex ramifications of events caused by agents. Since natural events need not happen necessarily, the definition allows conditions of CDRs not used in causal links to be left open. In this respect, CDRs are similar to conditional effects in POCL planning whose conditions must only be supported if their effect is needed in a causal link.

[6]Since we do not prevent cyclic triggering of rules, the closure of a plan $P$ might be infinite. For example, *day* may be a durative action with an end event triggering another process, *night*, which in turn triggers *day* again. This is a natural way to model recurring events. For space reason, we will not discuss it in detail here, but assume that either cyclic rules do not exist or that the planning and execution horizon is restricted to some *time window* within which the closure is finite.

As the final result of all formalizations, the only theorem in this paper confirms our definition of a "solution" to a MAP problem to be consistent with the state transition model of Classical Planning.

To show this relation, we firstly need complete states to compute transitions on. For a problem instance $\Pi = (\mathcal{D}, I, G)$, a completely defined PVA $s$ is a **possible initial state** if $s(v) = I(v)$ whenever $I(v)$ is defined. $\mathcal{I}_\Pi$ is the set of **possible initial states** for $\Pi$.

Secondly, we must clarify the relation between the temporal constraint networks of MA plans and the transition sequences of Classical Planning. We first note that each possible initial state $s \in \mathcal{I}_\Pi$ is a branching context for the execution of a solution plan $P$, i.e. $s$ induces an unconditional plan $P_s = (A, O', T', L', \emptyset)$ which only contains those processes that in the original solution $P$ where labelled consistently with $I'$. Formally: $e \in \mathcal{E}^{O'}$ iff $B(e)$ is consistent with $s$. An execution schedule for an unconditional plan $P_s = (A, O', T', L', \emptyset)$ is a plan $sched(P_s) = (A, O', T'', L', \emptyset)$ where $T''$ is an extension of $T'$ such that for each pair of events $e_1, e_2 \in \mathcal{E}^{O'}$ either $(e_1 \prec e_2) \in T''$, $(e_2 \prec e_2) \in T''$, or $(e1, e2, [0, 0]) \in T$. An execution schedule is valid, if, despite the new constraints, the underlying STN remains consistent and process-consistent. A valid execution schedule describes a possible sequence of events when executing $P_s$. This schedule, however, may still include simultaneous events. This is, however, unproblematic since the definition of threats (Def. 3) prevents simultaneous occurence of conflicting events. Therefore, to construct a transition sequence, it is possible to allow those events to virtually occur in arbitrary order: we define the set of totally ordered transition sequences of $P_s$ to consist of all sequences $seq = \langle e_1, \ldots, e_n \rangle$ that are topological sortings of valid execution schedules $sched(P_s)$.

Given the set of possible initial states and the induced set of possible transition sequences for a plan $P$, we can finally relate the semantics of multiagent plans to classical plans by the following theorem:

**Theorem 1** *Let $P_\mathcal{D} = (A, O, T, L, B)$ be a solution to a MAP problem $\Pi = (\mathcal{D}, I, G)$. Then, for all possible initial states $s \in \mathcal{I}_\Pi$, $G \subseteq res(s, p)$ for all $p \in TO(P_s)$.*

Proof sketch: Analogously to the semantics of classic POCL plans which is also defined in terms of topological sortings of a partially ordered sequence of events, we use valid execution schedules $sched(P_s)$ to define what Fox & Long (2003) call a "happening sequence" of a temporal plan. Def. 5 explicitly orders the possible initial events $\mathcal{E}_I^?$ before all other events in a solution plan. Therefore, since those events do not threaten any others in $P_s$ (otherwise $P_s$ would violate Def. 5 and thus would not be a solution), each $sched(P_s)$ must be executable in $s$. Since $P_s$ must also contain neither open conditions nor threats, $P_s$ is executable in $s$ (Penberthy & Weld 1992). In particular, the goal event $e_G$ that is scheduled after all other events will be the last event occuring in a topological sortings of $sched(P_s)$. Therefore $G$ is true in the final state of the execution.

We have left out another, more interesting theoretical result, namely how individually executable plan fragments can

be generated from a global plan that are guaranteed to be jointly executable. While the result is rather obvious (since knowledge preconditions ensure that agents wait until they *perceive* the satisfied preconditions, even if they don't know about events that have caused the perception) its description in terms of mergeable individual plans is beyond the space of this paper.

## 7 Planning for Multiple Agents

To show that planning is indeed possible for MAP domains we will now sketch an algorithm for planning in our formalism. It is, however, not specifically tailored to MA Planning, but mainly consists in a transformation of the planning problem to a well-known representation (POCL plans with conditional effects) and the subsequent application of a standard POCL Planning algorithm. As such, the algorithm is certainly less efficient than the special-purpose MAP algorithm we will present in forthcoming work.

In a preprocessing step, all induced belief and mutual belief variables are generated, and all knowledge preconditions and effects a added explicitly to events. In particular, mutual belief effects are added to speech acts, and copresence rules are derived from sensor models. Then for each instantaneous CDR $r$ that is enabled by an event $e$ we extend $e$ by a conditional effect corresponding to $r$. Afterwards, the original CDRs are removed from the MAP domain.

The actual planning algorithm works like UCPOP (Weld 1994): it resolves open conditions by supporting them with causal links, thereby adding processes to the plan if necessary, and threats by promotion, demotion, or confrontation. In particular, confrontation will ensure that no harmful CDRs are triggered. If the current partial plan enables non-instantaneous CDRs its closure must explicitly be computed for threat and validity checking. Generally, when a non-instantaneous process $o$ is added, this means that all events $\mathcal{E}_o$ and constraints between them must be added to the plan. Similarly, when promoting or demoting processes, their duration constraints must be preserved by moving the start, end, and invariant event simultaneously.

Based on the soundness and completeness of UCPOP we can easily prove soundness and completeness of the modified algorithm for the case where duration constraints are merely ordering constraints. For metric duration constraints, we must further guarantee that the Simple Temporal Network underlying the plan is consistent. This can be achieved in low polynomial time (Dechter, Meiri, & Pearl 1991; Younes & Simmons 2003). The definition of threats from Def. 3 which uses temporal instead of relational ordering constraints, ensures that every plan found by the algorithm can indeed be executed in every possible total order without endangering causal links.

## 8 Related Work

This work integrates ideas from several research communities, in particular Classical and Distributed Planning, Multiagent Systems, Epistemic Logic, and Reasoning about Actions and Change.

Boutilier & Brafman (Boutilier & Brafman 2001) developed a formalism for multi-actuator plans and a planning algorithm based on classical POCL techniques. They model interacting effects of concurrent actions by specific kinds of conditional effects of the individual agents. A plan must provide simultaneity constraints ensuring that the interaction really takes place as planned. The authors assume that an external synchronization mechanism will ensure that during execution the constraints are met by the agents. Our formalism, however, rests on the assumption that executing agents are truly autonomous and there is no external instance to synchronize them. Therefore it must allow agents to synchronize on their own. This is achieved by explicit representation of changing knowledge and reasoning about individual and joint perceptions.

The events and temporal constraints in multiagent plans form a Simple Temporal Network (STN) (Dechter, Meiri, & Pearl 1991). Earlier work using this approach to extending PO plans with quantitative temporal constraints include (Ghallab & Laruelle 1994; Younes & Simmons 2003). These approaches subsume the temporal model of PDDL 2 (Fox & Long 2003), but extend it with flexible action durations that are necessary for our "qualitative" approach multiagent synchronization based on perception, rather than on absolute time points.

Conditional single-agent plans based on STNs were used by Tsamardinos et al. (Tsamardinos, Pollack, & Horty 2000; Tsamardinos, Vidal, & Pollack 2002). The notion of different kinds of *control* over intervals in a temporal constraint network was introduced by Vidal and Fargier (Vidal & Fargier 1999). In this paper we provide an extension to these approaches by specifying conditional temporal *multiagent* plans. However, we permit flexible action durations, but no other temporal constraints, which, for the time being, allows us to abstract from the subtler points of control and plans with observation nodes.

We do not know of other work on (multiagent) planning that formalizes the notion of causal domain laws or provides a similar approach to describing complex ramifications of concurrent multiagent actions. Our approach is inspired by work of Thielscher (Thielscher 1995) in the Theory of Actions community.

None of the above mentioned research describes execution time synchronization, sensor modeling, or communication. Our approach to planning in the presence of sensing is inspired by work of Etzioni, Weld & colleagues (Etzioni *et al.* 1992; Golden & Weld 1996; Smith & Weld 1999), Levesque (Levesque 1996), and Petrick & Bacchus (Petrick & Bacchus 2002; 2004). Again, we extend previous work to the multiagent case, thereby providing the basis for synchronized action at execution time. In particular, our explicit modeling of sensing and communication in multiagent environments complements BDI-inspired MAP models like (Grosz & Kraus 1996) that describe the role of (mutual) beliefs in necessary conditions for planful MA behavior, but do not explain how these conditions can be achieved during plan execution.

Since the focus of this paper is Distributed Plan Execution rather Distributed Planning, we will only briefly relate

our representation to some of the formal models used in that area. An excellent survey on techniques for Distributed Planning can be found in the paper by desJardins et. al. (DesJardins *et al.* 2000). Within this field there is a huge body of work relying on a hierarchical representation of multiagent plans (Durfee & Lesser 1987; Durfee & Montgomery 1991; DesJardins & Wolverton 1999; Clement & Durfee 1999b; 1999a). Hierarchical plans are very important in practical applications and therefore we are planning to extend our formalism to account for action decompositions. We believe that this extension should prove not to be complicated, since durative actions and their "invariant conditions", as used in our model, may be employed to "hide" an action decomposition and its "inconditions".

## 9 Conclusion and Future Work

We have presented a rich formal model of multiagent planning that includes and clarifies many important characteristics of MAP missing or underspecified in previous work. In particular, our model describes sensing and communication, and how both explain the evolution of (common) knowledge during a plan. Perceptions and knowledge provide the basis for "qualitative" synchronization of plans, i.e. quantitative notions like exact time points and durations become less important, thereby giving multiagent plans the flexibility needed by truly autonomous agents.

We have sketched a planning algorithm for MAP domains. A more elaborate algorithm will be presented soon. It is based on an extension of state-space forward-search techniques to POCL planning which we call Progressive Partial-Order Planning. This technique also allows to easily reason about triggered domain rules (or prevent their firing).

Although efficient algorithms and empirical results are yet missing in this paper, it is our hope that by defining not only the formal semantics, but also providing sample domains, a parser, and a plan validator for a concrete PDDL-like syntax, we can provide helpful tools for other MAP researchers and thus help to advance this interesting field of research.

## References


Bäckström, C., and Nebel, B. 1995. Complexity results for $SAS^+$ planning. *Computational Intelligence* 11(4):625–655.

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

Boutilier, C., and Brafman, R. 2001. Partial order planning with concurrent interacting actions. *JAIR*.

Brenner, M. 2003. Multiagent planning with partially ordered temporal plans. In *Proc. IJCAI '03*. Extended version: TR 190, Inst. for Computer Science, Freiburg Univ.

Clement, B., and Durfee, E. 1999a. Top-down search for coordinating the hierarchical plans of multiple agents. In *Agents-99*.

Clement, B. J., and Durfee, E. H. 1999b. Theory for coordinating concurrent hierarchical planning agents using summary information. In *Proc. of AAAI '99*, 495–502. Menlo Park, CA: AAAI Press.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49.

DesJardins, M., and Wolverton, M. 1999. Coordinating a distributed planning system. *AI Magazine* 20(4).

DesJardins, M.; Durfee, E.; C. Ortiz, J.; and Wolverton, M. 2000. A survey of research in distributed, continual planning. *AI Magazine*.

Durfee, E. H., and Lesser, V. R. 1987. Using partial global plans to coordinate distributed problem solvers. In *Proc. IJCAI-87*.

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

Etzioni, O.; Hanks, S.; Weld, D.; Draper, D.; Lesh, N.; and Williamson, M. 1992. An approach to planning with incomplete information. In *Proc. of KR '92*, 115–125.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning About Knowledge*. MIT Press.

Fox, M., and Long, D. 2003. PDDL 2.1: an extension to PDDL for expressing temporal planning domains. *JAIR* 20:61–124.

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

Golden, K., and Weld, D. S. 1996. Representing sensing actions: The middle ground revisited. In *Proc. of KR '96*, 174–185.

Golden, K. 1998. Leap before you look: Information gathering in the puccini planner. In *AIPS*, 70–77.

Grosz, B. J., and Kraus, S. 1996. Collaborative plans for complex group action. *Artificial Intelligence* 86(2):269–357.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proc. ICAPS 2004*, 161–170.

Levesque, H. J. 1996. What is planning in the presence of sensing? In *Proc. AAAI*, 1139–1146.

Lewis, D. 1969. *Convention. A Philosophical Study*. Cambridge, Massachusetts: Harvard University Press.

Penberthy, J. S., and Weld, D. 1992. UCPOP: a sound, complete partial order planner for adl. In *Proc. KR'92*.

Peot, M. A., and Smith, D. E. 1992. Conditional nonlinear planning. In *Proc. AIPS-92*.

Petrick, R., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. ICAPS-02*.

Petrick, R. P. A., and Bacchus, F. 2004. Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proc. of ICAPS '04*, 2–11.

Smith, D., and Weld, D. 1999. Temporal planning with mutual exclusion reasoning. In *Proc. IJCAI*.

Thielscher, M. 1995. The logic of dynamic systems. In *Proc. IJCAI-95*.

Tsamardinos, I.; Pollack, M. E.; and Horty, J. F. 2000. Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches. In *Proc. AIPS-00*.

Tsamardinos, I.; Vidal, T.; and Pollack, M. 2002. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints Journal*.

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks. *Journal of Experimental and Theoretical AI*.

Weld, D. S. 1994. An introduction to least commitment planning. *AI Magazine* 15(4):27–61.

Younes, H., and Simmons, R. 2003. VHPOP: Versatile heuristic partial order planner. *JAIR* 20:405–430.