Contingency Planning in Linear Time Logic

Andrea Orlandini

Università di Roma Tre
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79
00146 Rome (Italy)
Tel. +39-0655173220
orlandin@dia.uniroma3.it

Abstract

The "planning as satisfiability" approach for classical planning establishes a correspondence between planning problems and logical theories, and, consequently, between plans and models. This work proposes a similar framework for contingency planning: considering contingent planning problems where the sources of indeterminism are incomplete knowledge about the initial state, non-inertial fluents and non-deterministic actions, it shows how to encode such problems into Linear Time Logic. Exploiting the semantics of the logic, and the notion of *conditioned model* introduced in this work, formal characterizations are given of the notions of contingent plan (a plan together with the set of conditions that ensure its executability).

This work has to be considered as the beginning of a research project in which investigate applications in realistic scenarios.

Introduction

Classical planning is based on a number of simplifying assumptions that are too restrictive to model a realistic environment, where: the description of the initial state may be incomplete; actions may have non-deterministic effects; the environment may be changed by exogenous events; the agent interacts with the environment and gathers information from sensors.

Planning under uncertainty can be solved by *conditional planning*, whose aim is the synthesis of IF/THEN/ELSE programs whose execution requires sensing. Considering the heavy computational complexity of conditional planning (Rintanen 1999), an alternative approach is based on *re-planning*: instead of synthesizing the whole conditional plan off-line, and then executing it, planning and execution (with sensing actions) are interleaved.

Most approaches to contingency planning are based on the manipulation of knowledge states and the explicit representation of sensing actions (Pryor & Collins 1996; Levesque 1996; Anderson, Weld, & Smith 1998; Bonet & Geffner 2000; Bertoli *et al.* 2001; Petrick & Bacchus 2002). An exception is represented by (Rintanen 1999), where conditional planning is reduced to truth evaluation of quantified boolean formulae.

This work proposes an approach to contingency planning where the agent's knowledge and sensing actions are not represented explicitly. It represents an extension of the ideas presented in (Cialdea Mayer *et al.* 2000), where the whole planning problem is modelled in linear time logic (LTL) and planning is reduced to model search. The expressive power of LTL allows one to represent domain restrictions, intermediate tasks, domain and control knowledge (useful to guide the search), as well as temporally extended goals. Moreover, the logical encoding of planning problems provides a formal semantics of the planning language.

In this paper we show that both exogenous events and actions with non-deterministic effects can be represented in a natural way. Making use of a general planning language allowing for different forms of non-determinism (incomplete knowledge about the initial state, non-deterministic operators and non-determinism in the environment) and extended goals, we show how to encode the specification of a planning problem into a set S of LTL formulae, by suitably modifying the analogous translation for classical planning problems defined in (Cerrito & Cialdea Mayer 1998) and used by the system presented in (Cialdea Mayer *et al.* 2000).

Obviously, when the environment is not completely known and predictable, the correspondence between plans and models, established by the "planning as satisfiability" approach, fails. However, we show that a similar correspondence can be established between a *conditioned model* (a notion introduced in this work) and a *contingent plan*, that is a pair $\langle plan, conditions \rangle$, i.e. a plan together with the set of conditions that ensure its executability. In other words, exploiting the semantics of the logic, the notion of contingent plan is extended so that observations gathered during execution are taken into account. Such a characterization can be used when re-planning is needed. The planning procedure approach in this work is complete only under the assumption that actions are reversible.

A Planning Language for Non-Deterministic Domains and its Encoding in LTL

In this section, we define the kind of planning problems we are dealing with and their encoding in LTL. The problems are presented by means of a PDDL-like syntax. The language we use can be seen as an extension of the planning

language used by the system presented in (Cialdea Mayer *et al.* 2000), that allows also for the specification of different forms of domain-specific knowledge, such as heuristic knowledge and information on domain invariants. Although such a language allows also for universal and existential quantification (over finite and fixed domains) – that are however treated as conjunctions or disjunction of propositional formulae –, for simplicity we restrict here the language to be propositional.

The language we consider is built over an alphabet $\mathbb{P} = \mathcal{A} \cup \mathcal{F}$ of propositional letters, partitioned into the two sets $\mathcal{A}$ of action names and $\mathcal{F}$ of fluents. In turn, $\mathcal{F}$ is partitioned into the two sets $\mathcal{I}$ of inertial fluents and $\mathcal{F} - \mathcal{I}$ of non-inertial ones. Inertial fluents (Giunchiglia, Kartha, & Lifschitz 1997) are subject to the "commonsense law of inertia": if no action affects them, their value stays unchanged. The value of a non-inertial fluent, on the contrary, cannot be always predicted from its previous value and the agent's actions, but it may be subject to exogenous events. It can also be affected by the agent's actions, but the effect of actions on non-inertial fluents is guaranteed for the next state only.

The specification of a planning problem is divided into different sections. The declaration of fluents, partitioned into inertial and non-inertial, is contained in a specific section.

The `init` section contains a set of classical formulae describing the initial state. Such a section may contain disjunctive information and is not required to be complete, according to the hypothesis that the agent's knowledge of the initial state may be incomplete.

The goal of the problem is described in a corresponding section, in which it may contain also the temporal operator $\Diamond$ ("eventually in the future").

Each action is described in an `action` section, containing the operator's name, its preconditions and effects. Preconditions are classical formulae of any form. Conditional effects (`when <condition> <effect>`) are allowed, as well as non-deterministic ones, expressed as disjunctions, or exclusive disjunctions, of literals (atoms or negated atoms).

An example of non-deterministic action is $take\_file$ in Example 1: it non-deterministically causes one to obtain a pdf or postscript file (and not both). The symbol $\oplus$ is used for exclusive OR.

Example 1 *The agent has to get a file and print it. He does not know whether a postscript or pdf version of the document is available. Depending on this, he will use either Acrobat Reader or GhostView to print it.*

```
(:fluents
      (:inertial (printed, have_ps, have_pdf)))
(:action gv
      :preconditions have_ps
      :effects printed)
(:action acroread
      :preconditions have_pdf
      :effects printed)
(:action take_file
      :preconditions ¬have_ps ∧ ¬have_pdf
      :effects have_ps ⊕ have_pdf)
(:init ¬have_ps, ¬have_pdf, ¬printed)
(:goal printed)
```

In the rest of this section we show how to encode a planning problem into a set of LTL formulae. The encoding is an extension of the encoding of classical planning problems proposed in (Cerrito & Cialdea Mayer 1998) and used in (Cialdea Mayer *et al.* 2000).

We consider the language of LTL over the set $\mathbb{P} = \mathcal{A} \cup \mathcal{F}$ built by means of the connectives $\neg$, $\wedge$, $\vee$ and the future time operators $\Box$ (always), $\Diamond$ (eventually), $\bigcirc$ (next). Implication and double implication are defined symbols. For convenience, we restrict the language to formulae in negation normal form.

For our aims, a non standard way of describing the semantics of LTL is more convenient. It makes use of *labelled formulae*, i.e. expressions of the form $n : A$ where $n \in \mathbb{N}$ and A is a formula. If ℓ is a literal, then $n : \ell$ is a labelled literal. Intuitively, the label over a literal identifies the state where the literal is true. A set S of labelled literals is complete if for all $n \in \mathbb{N}$ and all $p \in \mathbb{P}$, either $n : p \in S$ or $n : \neg p \in S$. It is consistent if it does not contain both $n : p$ and $n : \neg p$ for any $n \in \mathbb{N}$ and $p \in \mathbb{P}$.

Definition 1 *A temporal interpretation $\mathcal{M}$ is a consistent and complete set of labelled literals.*

Intuitively, if $n : p \in \mathcal{M}$ then p is true at state n, and if $n : \neg p \in \mathcal{M}$ then p is false at state n. If $\mathcal{M}$ is an interpretation of $\mathbb{P}$, $n \in \mathbb{N}$ and A is a formula, the relation $\mathcal{M} \models n : A$ corresponds to the usual satisfiability relation $\mathcal{M}_n \models A$ (the n-th state of $\mathcal{M}$ satisfies A) and is recursively defined as follows:

1. $\mathcal{M} \models n : \ell$ iff $n : \ell \in \mathcal{M}$, when ℓ is a literal;
2. $\mathcal{M} \models n : A \wedge B$ iff $\mathcal{M} \models n : A$ and $\mathcal{M} \models n : B$;
3. $\mathcal{M} \models n : A \vee B$ iff either $\mathcal{M} \models n : A$ or $\mathcal{M} \models n : B$;
4. $\mathcal{M} \models n : \bigcirc A$ iff $\mathcal{M} \models n + 1 : A$;
5. $\mathcal{M} \models n : \Box A$ iff for all $k \geq n$, $\mathcal{M} \models k : A$;
6. $\mathcal{M} \models n : \Diamond A$ iff for some $k \geq n$, $\mathcal{M} \models k : A$.

A formula A is true in $\mathcal{M}$ (and $\mathcal{M}$ is a model of A) iff $\mathcal{M} \models 0 : A$. If S is a set of formulae, $\mathcal{M} \models S$ iff $\mathcal{M} \models A$ for all $A \in S$. From now on, a set S of LTL-formulae is considered the same as $\{0 : A \mid A \in S\}$.

The specification of a planning problem in the language described above is encoded by the set S_0 of LTL formulae obtained as follows.

- All the formulae describing the initial state (in the `init` section) are included in S_0. This results in a (possibly incomplete) description of the initial state.
- The goal G (stated in the `goal` section) is represented by the formula $\Diamond G$.
- For each operator a having preconditions $A_1, ..., A_n$, S_0 contains an *action precondition axiom*, i.e.

$$\Box(a \rightarrow A_1 \wedge ... \wedge A_n)$$

- For each action a, having a disjunctive effect $\ell_1 \vee ... \vee \ell_k$, under conditions C, (when C $(\ell_1 \vee ... \vee \ell_k)$) a *non-deterministic effect axiom* is in S_0:

$$\Box(a \wedge C \rightarrow \bigcirc\ell_1 \vee ... \vee \bigcirc\ell_k)$$

(if the effect is unconditioned, C is omitted).

- If f is a non-inertial fluent or a fluent that can be non-deterministically affected by some action, let G_f^+ be the disjunction of all the $a_i \wedge C_i$ that have f as a deterministic effect (G_f^+ specifies all the conditions that deterministically lead to change the truth value of f from false to true), and G_f^- the disjunction of all the $a_i \wedge C_i$ that have $\neg f$ as a deterministic effect (G_f^- specifies all the conditions that deterministically lead to change the truth value of f from true to false). Then S_0 contains the following *effect axiom* for each of such fluents:

$$\Box(G_f^+ \rightarrow \bigcirc f), \Box(G_f^- \rightarrow \bigcirc\neg f)$$

- For each inertial fluent f, such that:

(i) $\neg f$ is a non-deterministic effect of some actions $a_1, .., a_k$, under conditions, respectively, $C_1, ..., C_k$;

(ii) f is a non-deterministic effect of some actions $b_1, .., b_n$, under conditions, respectively, $D_1, ..., D_n$,

S_0 contains the following *inertia axioms* for f:

$$\Box((f \wedge \neg G_f^- \wedge \neg(a_1 \wedge C_1) \wedge ...\neg(a_k \wedge C_k)) \rightarrow \bigcirc f)$$
$$\Box(\bigcirc f \rightarrow (f \vee G_f^+ \vee (b_1 \wedge D_1) \vee ... \vee (b_n \wedge D_n)))$$

where G_f^+ and G_f^- are as specified in the previous item.

- For each inertial fluent f, that is never a non-deterministic effect of any action, S_0 contains the following paraphrases of Reiter's Successor State Axiom (Reiter 1991):

$$\Box(\bigcirc f \equiv G_f^+ \vee (f \wedge \neg G_f^-))$$

- S_0 contains also a set of axioms describing incompatibility relations between actions, of the form:

$$\Box(\neg a \vee \neg b)$$

What has to be explicitly encoded, however, are the incompatibilities of actions a and b such that a deletes a precondition of b, or vice-versa. In fact, if two actions a and b have conflicting preconditions or effects, such an incompatibility is taken into account by the logic itself, since a and b cannot be true in the same state.

For convenience, we also declare two actions a and b to be incompatible whenever a has a non-deterministic effect f and b a (deterministic or non-deterministic) effect $\neg f$ (or vice-versa).

The adequacy of the analogous encoding for classical planning problems has been shown in (Cerrito & Cialdea Mayer 1998). In the new setting, the encoding can be recognized correct when we consider that a successor state axiom is equivalent to the conjunction of the following formulae:

(1) $\Box(G_f^+ \rightarrow \bigcirc f)$ (2) $\Box(G_f^- \rightarrow \bigcirc\neg f)$
(3) $\Box(f \wedge \neg G_f^- \rightarrow \bigcirc f)$ (4) $\Box(\neg f \wedge \neg G_f^+ \rightarrow \bigcirc\neg f)$

Non-inertial fluents clearly do not satisfy (3) and (4), while (1) and (2) are present in S_0 as effect axioms. If f is a non-deterministic effect of some action, (3) and (4) are in S_0 as inertia axioms, (1) and (2) are replaced by the effect and non-deterministic effect axioms.

Here we give the complete encoding of the example presented before, filtered by application of some simple logical simplifications.

Example 1

Initial state:
$\neg have\_ps,\ \neg have\_pdf,\ \neg printed$

Goal:
$\Diamond printed$

Preconditions:
$\Box(take\_file \rightarrow \neg have\_ps \wedge \neg have\_pdf)$
$\Box(acroread \rightarrow have\_pdf), \Box(gv \rightarrow have\_ps)$

Fluents behaviour:
$\Box(take\_file \rightarrow (\bigcirc have\_ps \wedge \bigcirc\neg have\_pdf)$
$\vee\ (\bigcirc have\_pdf \wedge \bigcirc\neg have\_ps)))$
$\Box(have\_ps \rightarrow \bigcirc have\_ps)$
$\Box(have\_pdf \rightarrow \bigcirc have\_pdf)$
$\Box(\bigcirc have\_ps \rightarrow have\_ps \vee take\_file)$
$\Box(\bigcirc have\_pdf \rightarrow have\_pdf \vee take\_file)$
$\Box(\bigcirc printed \equiv (gv \vee acroread \vee printed))$

Contingent Plans

In this section, we identify a planning problem with its temporal logic encoding and formalize the notion of contingent plan in terms of temporal interpretations.

Instead of producing the whole conditional plan off-line, and then executing it, we can consider the possibility that planning and execution (with sensing actions) are interleaved. In that case, a single contingent plan can be generated at first. During its execution, its conditions are checked by means of sensing actions. When a condition turns out to be false, a different contingent plan is generated and considered.

A plan can be represented by a set of labelled action names. If S is a set of labelled formulae, then $lits(S)$ is the set of labelled literals in S.

Let S_0 be a set of formulae encoding a planning problem, as illustrated before. From a model $\mathcal{M}$ of S_0 we want to extract a contingent plan solving the problem represented by S_0, i.e. a significant subset of $\mathcal{M}$, describing a plan, together with a set of conditions ensuring the executability of the plan. In general, in fact, it is not the whole model $\mathcal{M}$ that is of interest and has to be explained, but only a finite fragment of the model, up to the achievement of the goals. Even the description of $\mathcal{M}$ up to such a "final" state need not be complete. However, it must be rich enough: it must contain enough information on how to reach the goal and which conditions ensure its reachability. The following definition formalizes this concept. For simplicity we assume here that S_0 contains formulae in negation normal form.

Definition 2 *Let S_0 be a set of labelled formulae, $S \supseteq S_0$ and $m \in \mathbb{N}$. Then S is a saturated extension of S_0 up to m if for every $n : A \in S$ such that $n \leq m$:*

1. *if* $A = A_0 \wedge A_1$, *then both* $n : A_0 \in S$ *and* $n : A_1 \in S$;
2. *if* $A = A_0 \vee A_1$, *then either* $n : A_0 \in S$ *or* $n : A_1 \in S$;
3. *if* $A = \Box A_0$ *then for all* k *such that* $n \leq k \leq m$, $k : A_0 \in S$;
4. *if* $A = \Diamond A_0$ *then for some* k *such that* $n \leq k \leq m$, $k : A_0 \in S$;
5. *if* $A = \bigcirc A_0$ *and* $n < m$ *then* $n + 1 : A_0 \in S$.

If $\mathcal{M}$ *is a model of* S_0 *and* S *a saturated extension of* S_0 *up to* m, *such that* $lits(S) \subset \mathcal{M}$, *then* $lits(S)$ *is called a* significant description of $\mathcal{M}$ as a model of S_0.

Note that, mainly because of clause 4 above, any significant description of a model of the encoding of a planning problem implies that the goals are achieved.

It can easily be proved, by use of the same techniques used to prove soundness of tableau methods, that if $\mathcal{M}$ is a model of S_0 then there exists a significant description of $\mathcal{M}$ as a model of S_0.

The following definitions are introduced to the aim of characterizing an *adequate* set of conditions ensuring the executability of a plan.

Definition 3 *If* S *is a set of labelled literals. The* history *of* S *up to* n, $h(S, n)$, *is* $\{k : \ell \mid k : \ell \in S \text{ and } k \leq n\}$.

Let S_0 *be a set of labelled formulae, and* K *and* U *sets of labelled literals. Then* U *is a* set of conditions *explaining* K *in the context of* S_0 *(briefly,* U *explains* K *in* S_0*) iff* U *is a minimal set (w.r.t. set inclusion) such that for all* $n : \ell \in K$: $S_0, h(U, n-1) \models n : \ell$.

The condition above amounts to saying that, for all $n : \ell \in K$ and for every model $\mathcal{M}'$ of S_0, if $\mathcal{M}'$ is also a model of the subset $h(U, n-1)$ of the history of U up to $n-1$, then $\mathcal{M}' \models n : \ell$, too. In informal terms, $h(U, n-1)$ "explains", in the context of S_0, all that happens, according to K, up to the n-th state. Moreover, no subset of U has such a property. If K implies, in the context of S_0, that the goals are achieved within the n-th step, then $h(U, n-1)$ actually guarantees the achievement of the goals.

Definition 4 *Let* S_0 *be a set of labelled formulae representing a planning problem. A* conditioned model *of* S_0 *is a pair* $\langle K, U\rangle$, *where* $K \cup U$ *is a significant description of some model* $\mathcal{M}$ *of* S_0 *and* U *explains* K *in* S_0.

If $\langle K, U\rangle$ *is a conditioned model of* S_0 *then the contingent plan corresponding to* $\langle K, U\rangle$ *solving the problem represented by* S_0 *is the set* $\langle \mathcal{P}, \mathcal{C}\rangle$, *where:*

$$\begin{array}{rcl} \mathcal{P} & = & \{n : a \mid n : a \in U \text{ and } a \in \mathcal{A}\} \\ \mathcal{C} & = & \{n : p \mid n : p \in U \text{ and } p \in \mathcal{F}\} \\ & & \cup \{n : \neg p \mid n : \neg p \in U \text{ and } p \in \mathcal{F}\} \end{array}$$

In simpler words, a contingent plan solving the problem encoded by S_0 can be extracted from the set U of conditions of a conditioned model of S_0, by dropping negative action literals, i.e. labelled literals of the form $n : \neg a$ for $a \in \mathcal{A}$, and splitting the rest of U into the set of action names and the fluents (contingencies guaranteeing the executability of the plan).

Note that, according to Definition 4, the success of the plan could also be conditioned by contingencies regarding the same state where the goal is achieved. For instance, let us consider the following simplification of the problem presented as Example 1:

```
(:fluents
      (:inertial have_ps, have_pdf))
(:action take_file
      :preconditions ¬have_ps ∧ ¬have_pdf
      :effects have_ps ⊕ have_pdf)
(:init ¬have_ps, ¬have_pdf)
(:goal have_ps)
```

A conditioned model of its encoding is:

$$\begin{array}{rcl} K & = & \{0 : \neg have\_ps,\ 0 : \neg have\_pdf\} \\ U & = & \{0 : take\_file,\ 1 : \neg have\_pdf,\ 1 : have\_ps\} \end{array}$$

The goal $have\_ps$ is achieved at state 1, but it is conditioned by contingencies relatives to state 1 itself. In this case, although for $n = 1$, $S_0, U \models n : have\_ps$, the achievement of the goal is not guaranteed only by $S_0 \cup h(U, n-1)$, but the whole set U has to be taken into account.

If this situation is to be excluded, then we must also require for a pair $\langle K, U\rangle$ to be a conditioned model of the encoding S_0 of a problem, that $K \models 0 : \Diamond G$, where $\Diamond G$ is the representation of the goal.

Let $\langle K, U\rangle$ be a conditioned model of the encoding S_0 of a planning problem and let us assume that, during the execution of the contingent plan extracted from $\langle K, U\rangle$, the set $\mathcal{O}$ of observations has been gathered. Let us assume, moreover, that $n : \ell$ is the result of the last sensing action and the only observation that is inconsistent with U. Then we want to synthesize a plan that is "complementary" to the previous one with respect to $n : \ell$. The new plan must take $\mathcal{O}$ into account, i.e. the observations should not appear as conditions in the new plan, nor should whatever is derivable from the observations and S_0.

Definition 5 *Let* S_0 *be the encoding of a planning problem, and* $\mathcal{O}$ *a set of observations, i.e. a set of labelled literals that is consistent with* S_0. *A conditioned model of* S_0 *in the presence of* $\mathcal{O}$ *is a conditioned model of* $S_0 \cup \mathcal{O}$, *i.e. a pair* $\langle K, U\rangle$ *such that* $K \cup U$ *is a significant description of* $S_0 \cup \mathcal{O}$ *and for all* $n : \ell \in K$: $S_0, \mathcal{O}, h(U, n-1) \models n : \ell$.

Let $\langle K, U\rangle$ *be a conditioned model of* S_0, $\mathcal{O}$ *a set of observations, and* n *the maximum label of the literals in* $\mathcal{O}$. *Let us assume, moreover, that* $n : \ell$ *is the only labelled literal that is inconsistent with* U. *Then a pair* $\langle K', U'\rangle$ *is complementary to* $\langle K, U\rangle$ *in the presence of* $\mathcal{O}$ *if it is a conditioned model of* S_0 *in the presence of* $\mathcal{O}$ *and* $K \cup U$ *and* $K' \cup U'$ *describe the same model up to step* $n-1$, *i.e. for all* $k < n$: $\{k : \ell \mid k : \ell \in K \cup U\} = \{k : \ell \mid k : \ell \in K' \cup U'\}$.

For instance, let S_1 be the encoding of the problem presented as Example 1, where the $take\_file$ action has a non-deterministic effect, and let $\langle K, U\rangle$ be the conditioned

model of S_1 where:

$$\begin{aligned} K = \quad & \{\, 0 : \neg have\_ps,\ 0 : \neg have\_pdf,\ 0 : \neg printed, \\ & \ 0 : \neg gv,\ 0 : \neg acroread, \\ & \ 1 : \neg printed,\ 1 : \neg take\_file, \\ & \ 2 : printed,\ 2 : have\_ps,\ 2 : \neg have\_pdf\ , \\ & \ 2 : \neg acroread,\ 2 : \neg take\_file\} \\ U = \quad & \{\, 0 : take\_file, \\ & \ 1 : have\_ps,\ 1 : \neg have\_pdf,\ 1 : gv, \\ & \ 1 : \neg acroread,\ 2 : \neg gv\ \} \end{aligned}$$

Note that both $1 : have\_ps$ and $1 : \neg have\_pdf$ occur as conditions in U, although they are inter-derivable in the context of S_1; in fact, none of the two literals is derivable from S_1 and $h(U, 0)$, since, at that step, the effect of $0 : take\_file$ cannot be predicted.

Let us assume that, during the execution of this plan, after performing the $take\_file$ action at step 0, $have\_ps$ turns out to be false at step 1, so the observation $\mathcal{O} = \{1 : \neg have\_ps\}$ must be taken into account when synthesizing the new plan. A suitable new plan is unconditioned:

$$\langle \{0 : take\_file,\ 1 : acroread\}, \emptyset \rangle.$$

In fact, this is the plan corresponding to the conditioned model $\langle K', U' \rangle$ of $S_1 \cup \mathcal{O}$, where:

$$\begin{aligned} K' = \quad & \{\, 0 : \neg have\_ps,\ 0 : \neg have\_pdf,\ 0 : \neg printed, \\ & \ 0 : \neg gv,\ 0 : \neg acroread, \\ & \ 1 : \neg printed,\ 1 : \neg have\_ps,\ 1 : have\_pdf, \\ & \ 1 : \neg gv,\ 1 : \neg take\_file, \\ & \ 2 : printed,\ 2 : \neg have\_ps,\ 2 : have\_pdf, \\ & \ 2 : \neg gv,\ 2 : \neg take\_file\ \} \\ U' = \quad & \{\, 0 : take\_file, 1 : acroread, 2 : \neg acroread\ \} \end{aligned}$$

In fact, $S_1, \mathcal{O}, 0 : take\_file \models 1 : have\_pdf$.

Concluding remarks

The "planning as satisfiability" approach for classical planning establishes a correspondence between planning problems and logical theories, and, consequently, between plans and models. This work proposes a similar framework for contingency planning: considering contingent planning problems where the sources of indeterminism are incomplete knowledge about the initial state, non-inertial fluents and non-deterministic actions, it shows how to encode such problems into LTL. Exploiting the semantics of the logic, and the notion of conditioned model, we give a formal characterization of the notions of contingent plan and replanning in the presence of a set of observations.

Much work still remains to be done. Let us consider at first the implementation side. A first prototype system was presented in (Cialdea Mayer & Limongelli 2002), but its performance are poor and it has to be re-implemented. The use of appropriate structure, e.g. BDD's for represent propositional formulae, can improve the efficiency of the system.

Our studies are involving in order to exploiting the expressive power of the language proposed: refining the description method of the planning domains can lead to a more efficient behaviour of the planner. In such a way, the choice of *good* control knowledge it's a crucial problem.

Moreover, we are actually studying the application of this tecnique in more complex domain. A realistic scenario can be a robot in a simulated environment in which a sensors system and a planning module interact.

References

Anderson, C.; Weld, D.; and Smith, D. 1998. Extending Graphplan to handle uncertainty and sensing actions. In *Proc. of the 15th National Conf. on Artificial Intelligence (AAAI-98)*, 897–904.

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2001. Planning in non deterministic domains under partial observability via symbolic model checking. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence IJCAI 2001*, 473–478.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proc. 5th Int. Conf. on AI Planning and Scheduling (AIPS 2000)*, 52–61.

Cerrito, S., and Cialdea Mayer, M. 1998. Using linear temporal logic to model and solve planning problems. In Giunghiglia, F., ed., *Proceedings of the 8th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'98)*, 141–152. Springer.

Cialdea Mayer, M., and Limongelli, C. 2002. Linear time logic, conditioned models and planning with incomplete knowledge. In Fermüller, C., and Egly, U., eds., *Proc. of the Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods*, volume 2381 of *LNAI*, 70–84. Springer.

Cialdea Mayer, M.; Orlandini, A.; Balestreri, G.; and Limongelli, C. 2000. A planner fully based on linear time logic. In Chien, S.; Kambhampati, S.; and Knoblock, C., eds., *Proc. of the 5th Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS-2000)*, 347–354. AAAI Press.

Giunchiglia, E.; Kartha, G. N.; and Lifschitz, V. 1997. Representing action: Indeterminacy and ramifications. *Artificial Intelligence* 95(2):409–438.

Levesque, H. 1996. What is planning in the presence of sensing? In *Proc. of the 13th National Conference on Artificial Intelligence, AAAI-96*, 1139–1146. AAAI Press.

Petrick, R., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. of the 6th Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS-2002)*.

Pryor, L., and Collins, G. 1996. Planning for contingencies: a decision-based approach. *Journal of Artificial Intelligence Research* 4:287–339.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., ed., *Artificial Intelligence and mathematical theory of computation: Papers in honor of John McCarthy*. Academic Press. 359–380.

Rintanen, J. 1999. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intellingence Research* 10:323–352.