Safe LTL Assumption-Based Planning

Alexandre Albore
Departament de Tecnologia, Universitat Pompeu Fabra
Passeig de Circumval·lació, 8 - 08003 Barcelona, Spain
alexandre.albore@upf.edu

Piergiorgio Bertoli
ITC-IRST
Via Sommarive, 18 - 38050 Povo, Trento, Italy
bertoli@irst.itc.it

Abstract

Planning for partially observable, nondeterministic domains is a very significant and computationally hard problem. Often, reasonable assumptions can be drawn over expected/nominal dynamics of the domain; using them to constrain the search may lead to dramatically improve the efficiency in plan generation. In turn, the execution of assumption-based plans must be monitored to prevent runtime failures that may happen if assumptions turn out to be untrue, and to replan in that case. In this paper, we use an expressive temporal logic, LTL, to describe assumptions, and we provide two main contributions. First, we describe an effective, symbolic forward-chaining mechanism to build (conditional) assumption-based plans for partially observable, nondeterministic domains. Second, we constrain the algorithm to generate *safe* plans, i.e. plans guaranteeing that, during their execution, the monitor will be able to univocally distinguish whether the domain behavior is one of those planned for or not. This is crucial to inhibit any chance of useless replanning episodes. We experimentally show that exploiting LTL assumptions highly improves the efficiency of plan generation, and that by enforcing safety we improve plan execution, inhibiting useless and expensive replanning episodes, without significantly affecting plan generation.

1 Introduction

A large number of relevant planning problems require the ability to deal with partially observable, nondeterministic domain models. Planning under such conditions is an extremely hard task, and often *strong* plans, i.e. plans that guarantee reaching the goal in spite of nondeterminism and incomplete runtime information, do not exist. In many cases, it is possible to express reasonable assumptions over the expected dynamics of the domain, e.g. by identifying "nominal" behaviors. Using these assumptions to constrain the search may greatly ease the planning task, allowing the efficient construction of assumption-based solutions. Of course, assumptions taken when generating a plan may turn out to be incorrect when executing it. For this reason, assumption-based plans must be executed within reactive architectures such as (Muscettola *et al.* 1998; Myers and Wilkins 1998), where a monitoring component traces the status of the domain, in order to abort plan execution and replan whenever an unexpected behavior has compromised the success of the plan. However, due to the incomplete run-time knowledge on the domain state, it may not be possible for a monitor to establish unambiguously whether the domain status is evolving as expected or not; in this case, replanning must occur whenever a dangerous state *may* have been reached. But if the actual domain state is one of those planned for, replanning is unnecessary and undesired. These situations can only be avoided if states not planned for can unambiguously be identified at plan execution time; in turn, whether this is possible crucially depends on the actions performed by the plan.

In this paper, we consider an expressive language that provides us with the key ability to specify assumptions over the domain dynamics, called linear temporal logics (LTL, (Emerson 1990)), and we provide two main contributions. First, we provide an effective, symbolic mechanism to constrain forward and-or search to generate (conditional) LTL assumption-based solutions for nondeterministic, partially observable domains. Second, we further constrain the search to obtain *safe* LTL assumption-based solutions, i.e. plans that not only guarantee that the goal is reached when the given assumption holds, but also guarantee that, during their execution, the monitor will be able to unambiguously distinguish whether the current domain status has been planned for or not. In this way, we guarantee that, during plan execution, the monitor will not trigger any plan abortion unless really needed. We experimentally show that generating LTL assumption-based solutions can be dramatically more effective than generating strong plans, and that enforcing safety can highly improve plan execution, since it inhibits costly and useless replanning episodes at the cost of a minor overhead in plan generation.

The paper is organized as follows. Section 2 provides the basic background notions regarding planning for partially observable, nondeterministic domains. Section 3 introduces LTL assumptions, and defines LTL assumption-based solutions for a planning problem. Section 4 provides the key notion of safe (LTL) assumption-based plan. Section 5 describes a forward-chaining procedure to generate LTL assumption-based solutions, and in particular safe ones, using symbolic representation techniques. Section 6 provides an experimental evaluation of the approach. Section 7 draws conclusions and illustrates future work directions.

2 Domain, Goals, Plans

We model a planning domain as a finite state machine. Our model follows (Bertoli *et al.* 2001), and allows for initial state uncertainty, non-deterministic action outcomes, and partial observability with noisy sensing:

Definition 1 (Planning domain). *A* nondeterministic planning domain with partial observability *is a 6-tuple* $\mathcal{D} = \langle \mathcal{S}, \mathcal{A}, \mathcal{U}, \mathcal{I}, \mathcal{T}, \mathcal{X} \rangle$*, where:*

- $\mathcal{S}$ *is the set of* states*.*
- $\mathcal{A}$ *is the set of* actions*.*
- $\mathcal{U}$ *is the set of* observations*.*
- $\mathcal{I} \subseteq \mathcal{S}$ *is the set of* initial states*; we require* $\mathcal{I} \neq \emptyset$*.*
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow 2^{\mathcal{S}}$ *is the* transition function*; it associates with each current state* $s \in \mathcal{S}$ *and with each action* $a \in \mathcal{A}$ *the set* $\mathcal{T}(s, a) \subseteq \mathcal{S}$ *of next states.*
- $\mathcal{X} : \mathcal{S} \rightarrow 2^{\mathcal{U}}$ *is the* observation function*; it associates with each state* s *the set of possible observations* $\mathcal{X}(s) \subseteq \mathcal{U}$*, with* $\mathcal{X}(s) \neq \emptyset$*.*

We say that action α is *executable* in state s iff $\mathcal{T}(s, \alpha) \neq \emptyset$, and we denote with $\alpha(s) = \{s' : s \in \mathcal{T}(s, \alpha)\}$ its execution. An action is executable in a set of states B (also called a *belief*) iff it is executable in every state of the set; its execution is denoted $\alpha(B) = \{s' : s' \in \mathcal{T}(s, \alpha),\ s \in B\}$. We indicate with $[\![o]\!]$ the set of states compatible with the observation o: $[\![o]\!] = \{s \in \mathcal{S} : o \in \mathcal{X}(s)\}$. Conversely, $\mathcal{X}(B) = \{o : \exists s \in B : o \in \mathcal{X}(s)\}$ denotes the observations compatible with some state in B. Moreover, we assume the existence of a set of basic propositions $\mathcal{P}$, and of a labeling of states with the set of propositions holding on them. We denote with $\mathcal{P}rop(\mathcal{P})$ the propositional formulæ over $\mathcal{P}$, and with $[\![\varphi]\!]$ the states satisfying $\varphi \in \mathcal{P}rop(\mathcal{P})$.

A planning problem is a pair $\langle \mathcal{D}, \mathcal{G} \rangle$, where $\mathcal{G} \subseteq \mathcal{S}$ is a set of goal states. We solve such problems considering conditional plans that may branch on the basis of observations:

Definition 2 (Conditional Plan). *The set of conditional plans* Π *for a domain* $\mathcal{D} = \langle \mathcal{S}, \mathcal{A}, \mathcal{U}, \mathcal{I}, \mathcal{T}, \mathcal{X} \rangle$ *is the minimal set such that:*

- $\epsilon \in \Pi$*;*
- *if* $\alpha \in \mathcal{A}$ *and* $\pi \in \Pi$*, then* $\alpha \circ \pi \in \Pi$*;*
- *if* $o \in \mathcal{U}$*, and* $\pi_1, \pi_2 \in \Pi$*, then* if o then π_1 else $\pi_2 \in \Pi$*.*

Intuitively, ϵ is the empty plan, $\alpha \circ \pi$ indicates that action α has to be executed before plan π, and $if\ o\ then\ \pi_1\ else\ \pi_2$ indicates that either π_1 or π_2 must be executed, depending on whether the observation o holds. A plan can be thought of as a finite state machine that executes synchronously with the domain.

An execution of a conditional plan can be described as a trace, i.e. a sequence of traversed domain states and associated observations, connected by the actions executed by the plan [1]. Notice that, since a plan may attempt a non-executable action, originating a run-time failure, we have to distinguish between failure traces from non-failure traces.

[1] We use the standard notation $\bar{x} = [x^1 \dots x^n]$ for a sequence of n elements, whose i-th element is indicated with x^i. Concatenations of two sequences $\bar{x_1}$ and $\bar{x_2}$ is denoted $\bar{x_1} \circ \bar{x_2}$. The length of $\bar{x}$ is denoted $|\bar{x}|$, and $\lfloor x \rfloor_L$ is its truncation of length L.

Definition 3 (Traces of a plan). *A trace of a plan is a sequence* $[s^0, o^0, \alpha^0, \dots, s^n, o^n, End]$*, where* s^i, o^i *are the domain state and the related observation at step* i *of the plan execution, and* α^i *is the action produced by the plan on the basis of* o^i *(and of the plan's internal state).* End *can either be* Stop*, indicating that the plan has terminated, or* $Fail(\alpha^n)$*, indicating execution failure of action* α^n *on* s^n*. We also indicate a trace with* $\langle \bar{s}, \bar{o}, \bar{\alpha} \rangle$*, splitting it into sequences of states, observations, and actions, respectively, and omitting the final symbol.*

A trace t *is a* goal trace *for problem* $\langle \mathcal{D}, \mathcal{G} \rangle$ *iff it is not a failure trace, and its final state, denoted* $final(t)$*, belongs to* $\mathcal{G}$*. We indicate with* $Trs(\pi, \mathcal{D})$*, the set of traces associated to plan* π *in domain* $\mathcal{D}$*.* $TrsFail(\pi, \mathcal{D})$ *and* $TrsG(\pi, \mathcal{D}, \mathcal{G})$ *are, respectively, the subsets of the failure and goal traces in* $Trs(\pi, \mathcal{D})$*.*

A plan is said to be a strong solution for a planning problem $\langle \mathcal{D}, \mathcal{G} \rangle$ iff every execution does not fail, and ends in $\mathcal{G}$:

Definition 4 (Strong solution). *A plan* π *is a strong solution for a problem* $\langle \mathcal{D}, \mathcal{G} \rangle$ *iff* $Trs(\pi, \mathcal{D}) = TrsG(\pi, \mathcal{D}, \mathcal{G})$

We observe that tree-structured plans such as the ones we consider can be decomposed into *paths*, and that such paths induce a partitioning on the set of traces of the plan:

Definition 5 (Paths). *The set of paths* $\mathcal{P}aths(\pi)$ *associated to a conditional plan* π *is defined as follows:*

- $\mathcal{P}aths(\epsilon) = \{\epsilon\}$
- $\mathcal{P}aths(\alpha \circ \pi) = \{\alpha \circ p : p \in \mathcal{P}aths(\pi)\}$
- $\mathcal{P}aths(\text{if } o \text{ then } \pi_1 \text{ else } \pi_2) = \{o \circ p : p \in \mathcal{P}aths(\pi_1)\} \cup \{\tilde{o} \circ p : p \in \mathcal{P}aths(\pi_2)\}$

where $\tilde{o}$ *is the symbol denoting the "complementary" observation to* o*, i.e. that the path can be traversed if any observation different from* o *is met.*

Definition 6 (Traces bound to a path). *Given a path* p*, a planning domain* $\mathcal{D}$*, and a belief state* B*, we call* $Trs_{\mathcal{D}}(p, B)$ *the set of traces associated to the path* p*:*

- $Trs_{\mathcal{D}}(\epsilon, B) = \{[s, o, Stop] : s \in B, o \in \mathcal{X}(s)\}$
- $Trs_{\mathcal{D}}(\alpha \circ p, B) =$ $\{[s, o, Fail(\alpha)] : s \in B, o \in \mathcal{X}(s), \alpha(s) = \emptyset\} \cup$ $\{[s, o, \alpha] \circ t : s \in B, o \in \mathcal{X}(s), \alpha(s) \neq \emptyset, t \in Trs_{\mathcal{D}}(p, \alpha(s))\}$
- $Trs_{\mathcal{D}}(o \circ p, B) = Trs_{\mathcal{D}}(p, B \cap [\![o]\!])$
- $Trs_{\mathcal{D}}(\tilde{o} \circ p, B) = Trs_{\mathcal{D}}(p, B \cap [\![\tilde{o}]\!])$*, where* $[\![\tilde{o}]\!] = \bigcup_{o' \in \mathcal{U} \setminus \{o\}} [\![o']\!]$

3 Assumptions, Assumption-based solutions

To express assumptions over the behavior of $\mathcal{D}$, we adopt *Linear Temporal Logic* (LTL) (Emerson 1990), whose underlying ordered structure of time naturally models the dynamic evolution of domain states (Calvanese *et al.* 2002). LTL expresses properties over sequences of states, by introducing the operators $\mathbf{X}$ (next) and $\mathbf{U}$ (until):

Definition 7 (LTL syntax). *The language* $\mathcal{L}(\mathcal{P})$ *of the LTL formulæ* φ *on* $\mathcal{P}$ *is defined by the following grammar, where* $q \in \mathcal{P}$*:*

$$\varphi := q \mid \neg \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\, \varphi \mid \varphi\, \mathbf{U}\, \varphi$$

The derived operators $\mathbf{F}$ (future) and $\mathbf{G}$ (globally) are defined on the basis of $\mathbf{U}$: $\mathbf{G}\varphi = \varphi\,\mathbf{U}\,\bot$ and $\mathbf{F}\varphi = \neg\mathbf{G}\neg\varphi$. The semantics of LTL formulæ are given inductively on infinite state sequences, see (Manna and Pnueli 1992).

Definition 8 (LTL semantics). *Given an infinite state sequence $\sigma = [s^0, \ldots, s^n, \ldots]$, we denote by $(\sigma, i) \models \varphi$ whether a formula $\varphi \in \mathcal{L}(\mathcal{P})$ holds at a position $i \geq 0$ in σ. The semantics for each of the operators and subformulæ provided by the syntax is defined as follows:*

$$
\begin{array}{ll}
(\sigma,i) \models \varphi \quad \textit{iff} & s^i \models \varphi \ \textit{and}\ \varphi \in \mathcal{P}rop(\mathcal{P}) \\
(\sigma,i) \models \neg\varphi \quad \textit{iff} & (\sigma,i) \not\models \varphi \\
(\sigma,i) \models \varphi \wedge \psi \quad \textit{iff} & (\sigma,i) \models \varphi \ \textit{and}\ (\sigma,i) \models \psi \\
(\sigma,i) \models \mathbf{X}\varphi \quad \textit{iff} & (\sigma,i+1) \models \varphi \\
(\sigma,i) \models \varphi\,\mathbf{U}\psi \quad \textit{iff} & \exists k \geq j : (\sigma,k) \models \psi \ \textit{and} \\
\quad \forall i : j \leq i < k, & (\sigma,i) \models \varphi \ \textit{or}\ (\sigma,j) \models \mathbf{G}\varphi
\end{array}
$$

We say that an LTL formula φ is satisfiable over a plan trace $\langle \bar{s}, \bar{o}, \bar{\alpha} \rangle \in Trs(\pi, \mathcal{D})$ iff it holds at position 0 for some infinite prolongation of $\bar{s}$. Thus, given an LTL assumption $\mathcal{H}$, the finite traces $Trs(\pi, \mathcal{D})$ of a plan π can be partitioned into those traces for which $\mathcal{H}$ is satisfiable, and those for which it is not, denoted $Trs_{\mathcal{H}}(\pi, \mathcal{D})$ and $Trs_{\bar{\mathcal{H}}}(\pi, \mathcal{D})$ respectively. The failure traces $TrsFail(\pi, \mathcal{D})$ are partitioned analogously into $TrsFail_{\mathcal{H}}(\pi, \mathcal{D})$ and $TrsFail_{\bar{\mathcal{H}}}(\pi, \mathcal{D})$, and so for $TrsG(\pi, \mathcal{D}, \mathcal{G})$, partitioned into $TrsG_{\mathcal{H}}(\pi, \mathcal{D}, \mathcal{G})$ and $TrsG_{\bar{\mathcal{H}}}(\pi, \mathcal{D}, \mathcal{G})$.

If every possible execution for which $\mathcal{H}$ is satisfiable succeeds, then the plan is a solution under assumption $\mathcal{H}$:

Definition 9 (Solution under Assumption). *A plan π is a solution for the problem $\langle \mathcal{D}, \mathcal{G} \rangle$ under the assumption $\mathcal{H} \in \mathcal{L}(\mathcal{P})$ iff* $Trs_{\mathcal{H}}(\pi, \mathcal{D}) = TrsG_{\mathcal{H}}(\pi, \mathcal{D}, \mathcal{G})$

Example 1. *We introduce a simple navigation domain for explanatory purposes, see Fig. 1. A mobile robot, initially placed in room I, must reach room K_3. The shaded cells K_1, K_2, K_3 are kitchens, while the other ones are normal rooms. The robot is only equipped with a smell sensor **K**, that allows to detect whether it is in a kitchen room. The robot moves at each step in one of the four compass directions (n, e, s, o); moving onto a wall is not possible.*

We do not know the speed of the robot: thus a movement might terminate in any of the rooms in the movement's direction (for instance, moving north from room I may end up in K_1, R_2 or in R_3).

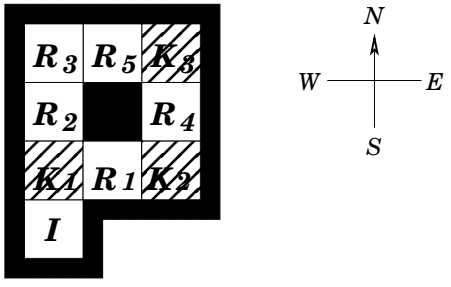

Figure 1: An example domain.

A strong solution for this problem does not exist, as it is easy to see. However, solutions exists if we assume that the robot steps of one room at a time, at least until it reaches the goal. The considered assumption formula is $\mathcal{H}_0 = (\mathbf{X}(\delta = 1)\,\mathbf{U}\,K_3)$, where δ models the Manhattan distance between two successive cells. The simple plan $\pi_1 = n \circ n \circ n \circ e \circ e \circ \epsilon$ is a solution under $\mathcal{H}_0$. This plan has the following possible traces $Trs(\pi_1, \mathcal{D})$:

$t_1 : [I, \overline{K}, n, K_1, K, n, R_2, \overline{K}, n, R_3, \overline{K}, e, R_5, \overline{K}, e, K_3, K, Stop.]$

$t_2 : [I, \overline{K}, n, K_1, K, n, R_3, \overline{K}, Fail(n)]$

$t_3 : [I, \overline{K}, n, R_2, \overline{K}, n, R_3, \overline{K}, Fail(n)]$

$t_4 : [I, \overline{K}, n, R_3, \overline{K}, Fail(n)]$

$t_5 : [I, \overline{K}, n, K_1, K, n, R_2, \overline{K}, n, R_3, \overline{K}, e, K_3, K, Fail(e)]$

Indeed $Trs_{\mathcal{H}_0}(\pi_1, \mathcal{D}) = TrsG_{\mathcal{H}_0}(\pi_1, \mathcal{D}, \mathcal{G}) = \{t_1\}$.

4 Safe assumption-based plans

At run-time, the domain may exhibit a behavior that does not satisfy the assumptions considered at planning-time. For this reason, assumption-based plans are usually executed within a reactive framework (see e.g. (Bertoli *et al.* 2001; Muscettola *et al.* 1998)), where an external beholder of the execution (called *monitor*) observes the responses of the domain to the actions coming from the plan, and triggers replanning when such observations are not compatible with some expected behavior.

However, the monitor may not always be able to decide whether the domain is reacting as expected or not to the stimuli of a plan π. This happens when the observations gathered are not informative enough to decide whether the domain is behaving as expected or not, i.e. when given a sequence of actions $\bar{\alpha}$, the domain produces a sequence of observations $\bar{o}$ compatible both with some assumed domain behavior, and with some behavior falsifying the assumption:

$\exists \bar{s}, t, \bar{s}', t' : \langle \bar{s}, \bar{o}, \bar{\alpha} \rangle \circ t \in Trs_{\mathcal{H}}(\pi, \mathcal{D}) \wedge \langle \bar{s}', \bar{o}, \bar{\alpha} \rangle \circ t' \in Trs_{\bar{\mathcal{H}}}(\pi, \mathcal{D})$.

In situations where such ambiguity on the current state makes the applicability of the next action also ambiguous, the monitor will trigger replanning, in order to rule out any chance of a run-time failure. But the next action might actually be applicable, even though this is not discernable: then, plan execution should not be interrupted, and replanning is unnecessary and undesirable.

Example 2. *Consider the plan π_1 from example 1. Trace t_2 produces the same observations (namely $[\bar{K}, K, \bar{K}]$) of the successful trace t_1, in response to the same actions ($n \circ n$), up to its failure. Thus, after the first n move, the monitor knows the robot is in K_1; after the second, the monitor cannot distinguish if the robot is in R_2 or R_3. In case it is in R_3, the next north action is inapplicable, so the plan execution has to be stopped. This makes π_1 practically useless, since its execution is always halted after two actions, even if the assumption under which it guarantees success holds.*

Similarly, an assumption-based solution plan may terminate without the monitor being able to distinguish whether the goal has been reached or not, therefore triggering replanning at the end of plan execution.

Whether such situations occur depend on the nature of the problem and, crucially, on the plan chosen as an assumption-based solution. We are interested in characterizing and generating assumption-based solutions such that these situations do not occur, i.e. every execution that causes action inapplicability or ends up outside the goal, must be distinguishable from every successful execution.

Thus, we define safe LTL assumption-based plans as those plans that (a) achieve the goal whenever the assumption holds, and (b) guarantee that each execution where an assumption failure compromises the success of the plan is observationally distinguishable (from the monitor's point of view) from any successful execution.

Definition 10 (Distinguishable traces). *Let t and t' be two traces; let $L = min(|t|, |t'|)$, $\lfloor t \rfloor_L = \langle \bar{s}, \bar{o}, \bar{\alpha} \rangle$ and $\lfloor t' \rfloor_L = \langle \bar{s}', \bar{o}', \bar{\alpha}' \rangle$. Then t, t' are distinguishable, denoted $Dist(t, t')$, iff $(\bar{o} \neq \bar{o}') \vee (\bar{\alpha} \neq \bar{\alpha}')$.*

Definition 11 (Safe LTL assumption-based solution). *A plan π is a safe assumption-based solution for assumption $\mathcal{H} \in \mathcal{L}(\mathcal{P})$ iff the conditions below are met:*

$a)$ $Trs_{\mathcal{H}}(\pi, \mathcal{D}) = TrsG_{\mathcal{H}}(\pi, \mathcal{D}, \mathcal{G})$

$b)$ $\forall t \in Trs_{\bar{\mathcal{H}}}(\pi, \mathcal{D}) \backslash TrsG_{\bar{\mathcal{H}}}(\pi, \mathcal{D}, \mathcal{G}),$
$\forall t' \in Trs_{\mathcal{H}}(\pi, \mathcal{D}) \cup TrsG_{\bar{\mathcal{H}}}(\pi, \mathcal{D}, \mathcal{G}) : Dist(t, t')$

Example 3. *Consider the plan $\pi_2 = n \circ e \circ e \circ n \circ n \circ \epsilon$. This is a safe assumption-based solution for the problem of example 1, as it is easy to see. The traces $Trs(\pi_2, \mathcal{D})$ are:*

$t'_1 : [I, \overline{K}, n, K_1, K, e, R_1, \overline{K}, e, K_2, K, n, R_4, \overline{K}, n, K_3, K, Stop]$

$t'_2 : [I, \overline{K}, n, K_1, K, e, R_1, \overline{K}, e, K_2, K, n, K_3, K, Fail(n)]$

$t'_3 : [I, \overline{K}, n, K_1, K, e, K_2, K, Fail(e)]$

$t'_4 : [I, \overline{K}, n, R_2, \overline{K}, Fail(e)]$

$t'_5 : [I, \overline{K}, n, R_3, \overline{K}, e, R_5, \overline{K}, e, K_3, K, Fail(n)]$

$t'_6 : [I, \overline{K}, n, R_3, \overline{K}, e, K_3, Fail(e)]$

We have $TrsG_{\mathcal{H}_0}(\pi_2, \mathcal{D}, \mathcal{G}) = Trs_{\mathcal{H}_0}(\pi_2, \mathcal{D}) = \{t'_1\}$, and every trace $t'_2, t'_3, t'_4, t'_5, t'_6$ is distinguishable from t'_1.

5 Generating safe LTL assumption-based plans

We intend to efficiently generate safe LTL assumption-based plans for partially observable, nondeterministic domains. For this purpose, we take as a starting point the plan generation approach presented in (Bertoli *et al.* 2001), where an and-or graph representing an acyclic prefix of the search space of beliefs is iteratively expanded: at each step, observations and actions are applied to a fringe node of the prefix, removing loops. Each node in the graph is associated with a belief in the search space, and to the path of actions and observations that is traversed to reach it. When a node is marked success, by goal entailment or by propagation on the graph, its associated path is eligible as a branch of a solution plan. The implementation of this approach by symbolic techniques has proved very effective in dealing with complex problems, where uncertainty results in manipulating large beliefs.

To generate plans under an LTL assumption $\mathcal{H}$, using this approach, we have to adapt this schema so that the beliefs generated during the search only contain states that can be reached if $\mathcal{H}$ is satisfiable. Moreover, in order to constrain the algorithm to produce safe plans, success marking of a leaf node n must require the safety conditions of def.11 (recast to those traces in $Trs_{\mathcal{H}}$ and $Trs_{\bar{\mathcal{H}}}$ that can be traversed to reach n). We now describe in detail these adaptations, and the way they are efficiently realized by means of symbolic representation techniques.

5.1 Assumption-induced pruning

In order to prune states that may only be reached if the assumption is falsified, we annotate each state s in a (belief associated to a) search node with an LTL formula φ, representing the current assumption on how s will evolve over the timeline, and we progress the pair $\langle s, \varphi \rangle$ in a way conceptually similar to (Kabanza *et al.* 1997). Thus, a graph node will now be associated to a set $\mathcal{B} = \{\langle s, \varphi \rangle : s \in \mathcal{S}, \varphi \in \mathcal{L}(\mathcal{P})\}$ called *annotated belief*, whose states will be denoted with $St(\mathcal{B}) = \{s : \langle s, \varphi \rangle \in \mathcal{B}\}$. Formulæ will be expressed by unrolling top-level **U**s in terms of their recursive semantic definition, to allow evaluating them on the current state:

$$
\begin{aligned}
&Unroll(\psi) = \psi \quad \text{if } \psi \in \{\top, \bot\} \cup \mathcal{P}rop(\mathcal{P}) \\
&Unroll(\neg\varphi) = \neg Unroll(\varphi) \\
&Unroll(\varphi \vee \psi) = Unroll(\varphi) \vee Unroll(\psi) \\
&Unroll(\varphi \wedge \psi) = Unroll(\varphi) \wedge Unroll(\psi) \\
&Unroll(\mathbf{X}\varphi) = \mathbf{X}\varphi \\
&Unroll(\varphi \,\mathbf{U}\, \psi) = Unroll(\psi) \vee (Unroll(\varphi) \wedge \mathbf{X}(\varphi \,\mathbf{U}\, \psi))
\end{aligned}
$$

Thus, the initial graph node will be

$$\mathcal{B}_{\mathcal{I}} = \{\langle s, Unroll(\mathcal{H})|_s \rangle : \ s \in \mathcal{I}, \ Unroll(\mathcal{H})|_s \neq \bot\} \quad (1)$$

where $\varphi|_s$ denotes the formula resulting from φ by replacing its subformulæ outside the scope of temporal operators with their evaluation over state s. Notice that the formulæ associated to states have the form $\bigwedge \varphi_i \wedge \bigwedge \mathbf{X}\psi_j$, with $\varphi_i, \psi_j \in \mathcal{L}(\mathcal{P})$. When a fringe node in the graph is expanded, its associated annotated belief $\mathcal{B}$ is progressed as follows:

- if an observation o is applied, $\mathcal{B}$ is restricted to

$$\mathcal{E}(\mathcal{B}, o) = \{\langle s, \varphi \rangle \in \mathcal{B} : \ s \in [[o]]\} \quad (2)$$

- if an (applicable) action α is applied, each pair $\langle s, \varphi \rangle \in \mathcal{B}$ is progressed by rewriting φ to refer to the new time instant, unrolling untils, and evaluating it on every state in $\alpha(s)$:

$$
\begin{aligned}
\mathcal{E}(\mathcal{B}, \alpha) = \{&\langle s', X^{-1}(\varphi)|_{s'} \rangle : \\
&\langle s, \varphi \rangle \in \mathcal{B}, \ s' \in \alpha(s), X^{-1}(\varphi)|_{s'} \neq \bot\}
\end{aligned} \quad (3)
$$

where $X^{-1}(\varphi)$ is defined as follows:

$$
\begin{aligned}
&X^{-1}(\psi) = \psi \quad \text{if } \psi \in \{\top, \bot\} \\
&X^{-1}(\neg\psi) = \neg X^{-1}(\psi) \\
&X^{-1}(\varphi \vee \psi) = X^{-1}(\varphi) \ \vee \ X^{-1}(\psi) \\
&X^{-1}(\varphi \wedge \psi) = X^{-1}(\varphi) \ \wedge \ X^{-1}(\psi) \\
&X^{-1}(\mathbf{X}\varphi) = Unroll(\varphi)
\end{aligned}
$$

5.2 Enforcing safety

Def. 11.a is easily expressed as an entailment between the states of the current annotated belief and the goal. To efficiently compute the distinguishability (requirement 11.b), we associate to a search node the sets of indistinguishable final states of $Trs_{\mathcal{H}}(p, \mathcal{D})$ and $Trs_{\bar{\mathcal{H}}}(p, \mathcal{D})$, storing them within a couple of annotated beliefs $\langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle$. When $\mathcal{B}_{\mathcal{H}}$ and $\mathcal{B}_{\bar{\mathcal{H}}}$ only contain goal states, this indicates that the only indistinguishable behaviors lead to success, so the path satisfies requirement 11.b (in particular, if $\mathcal{B}_{\mathcal{H}}$ and $\mathcal{B}_{\bar{\mathcal{H}}}$ are empty, this indicates that the monitor will be able to distinguish any assumption failure along p).

During the search, $\mathcal{B}_{\mathcal{H}}$ and $\mathcal{B}_{\bar{\mathcal{H}}}$ are progressed similarly as the annotated belief representing the search node; but on top of this, they are pruned from the states where the success or failure of the assumption can be distinguished by observations. In fact, while progressing $\langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle$, we detect situations where indistinguishable assumption failures may compromise action executability. These situations inhibit the safety of the plan, and as such we cut the search on these branches of the graph.

Initially, we compute $\langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle$ by considering those states of $\mathcal{I}$ for which $\mathcal{H}$ (resp. $\bar{\mathcal{H}}$) is satisfiable, and by eliminating from the resulting couple the states distinguishable thanks to some initial observation, i.e.

$$\langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle = \big\langle prune\big(\mathcal{B}^0_{\mathcal{H}}, \mathcal{B}^0_{\bar{\mathcal{H}}}\big), prune\big(\mathcal{B}^0_{\bar{\mathcal{H}}}, \mathcal{B}^0_{\mathcal{H}}\big) \big\rangle \quad (4)$$

where

$$\mathcal{B}^0_{\mathcal{H}} = \big\{ \langle s, Unroll(\mathcal{H})|_s \rangle : s \in \mathcal{I},\ Unroll(\mathcal{H})|_s \neq \bot \big\}$$

$$\mathcal{B}^0_{\bar{\mathcal{H}}} = \big\{ \langle s, Unroll(\bar{\mathcal{H}})|_s \rangle : s \in \mathcal{I},\ Unroll(\bar{\mathcal{H}})|_s \neq \bot \big\}$$

$$prune(\mathcal{B}, \mathcal{B}') = \big\{ \langle s, \varphi \rangle \in \mathcal{B} : \exists \langle s', \varphi' \rangle \in \mathcal{B}' : \mathcal{X}(s) \cap \mathcal{X}(s') \neq \emptyset \big\}$$

When a node is expanded, its associated pair is expanded as follows:

- if an observation o is applied, $\mathcal{B}_{\mathcal{H}}$ and $\mathcal{B}_{\bar{\mathcal{H}}}$ are simply pruned from the states not compatible with o:
$$\mathcal{E}(\langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, o) = \langle \mathcal{E}(\mathcal{B}_{\mathcal{H}}, o), \mathcal{E}(\mathcal{B}_{\bar{\mathcal{H}}}, o) \rangle$$
- If the node is expanded by an action α, and α is not applicable on some state of $\mathcal{B}_{\bar{\mathcal{H}}}$, then a "dangerous" action is attempted on a state that can be reached by an indistinguishable assumption failure. This makes the plan unsafe: as such, we mark the search node resulting from this expansion as failure.
- If the node is expanded by an action α, and α is applicable on every state of $\mathcal{B}_{\mathcal{H}}$ and $\mathcal{B}_{\bar{\mathcal{H}}}$, then $\mathcal{E}(\langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, \alpha) = \langle prune(\mathcal{B}'_{\mathcal{H}}, \mathcal{B}'_{\bar{\mathcal{H}}}), prune(\mathcal{B}'_{\bar{\mathcal{H}}}, \mathcal{B}'_{\mathcal{H}}) \rangle$ where $\mathcal{B}'_{\mathcal{H}} = \mathcal{E}(\mathcal{B}_{\mathcal{H}}, \alpha)$ and $\mathcal{B}'_{\bar{\mathcal{H}}} = \mathcal{E}(\mathcal{B}_{\bar{\mathcal{H}}}, \alpha)$.

The safety of a plan π can be evaluated by considering an associated execution tree $\tau(\pi, \mathcal{I}, \mathcal{H})$, built by progressing and pruning annotated beliefs as described so far. The nodes of $\tau(\pi, \mathcal{I}, \mathcal{H})$ are either annotated beliefs or $Fail$ nodes, and we denote its leaves with $leaf\big(\tau(\pi, \mathcal{I}, \mathcal{H})\big)$. For π to be safe, its associated execution tree must be failure-free, and the states in the leaves, unless unambiguously related to the assumption being false, must be associated to goal states:

Theorem 1. *Let π be a conditional plan for a domain $\mathcal{D} = \langle \mathcal{S}, \mathcal{A}, \mathcal{U}, \mathcal{I}, \mathcal{T}, \mathcal{X} \rangle$, let $\mathcal{G} \subseteq \mathcal{S}$ be a goal, and let $\mathcal{H} \in \mathcal{L}(\mathcal{P})$ be an assumption. If $\tau(\pi, \mathcal{I}, \mathcal{H})$ does not contain the Fail node, and $\forall \langle \mathcal{B}, \langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle \rangle \in leaf\big(\tau(\pi, \mathcal{I}, \mathcal{H})\big) : St(\mathcal{B}) \cup St(\mathcal{B}_{\bar{\mathcal{H}}}) \subseteq \mathcal{G}$, then π is a safe solution for $\mathcal{G}$ under $\mathcal{H}$.*

A detailed proof of the theorem is provided in Appendix.

5.3 Symbolic annotated beliefs

Progressing annotated belief states by explicitly enumerating each state becomes soon unfeasible when beliefs contain large sets of states. To scale up on significant problems, we exploit symbolic techniques based on BDDs (Bryant 1986) that allow efficiently manipulating sets of states; such techniques are indeed the key behind the effectiveness of approaches such as (Bertoli *et al.* 2001). To leverage on BDD primitives, we group together states associated with the same formula, and represent annotated beliefs as sets of pairs $\langle B, \varphi \rangle$, where B is a set of states (a belief). Inside a *symbolic annotated belief*, we do not impose that two beliefs are disjoint. A state belonging to two pairs $\langle B_1, \varphi_1 \rangle$ and $\langle B_2, \varphi_2 \rangle$ is in fact associated to $\varphi_1 \vee \varphi_2$.

Symbolic annotated beliefs are progressed and handled by operating on their belief-formula pairs, based on the consideration that LTL formulæ can be represented in a disjunctive normal form:

$$\phi = \bigvee (\bigwedge \psi_i \wedge \bigwedge \neg \rho_i)$$

where ψ_i, ρ_i are either basic propositions, or formulæ whose top-level is either $\mathbf{U}$ or $\mathbf{X}$. In particular, a disjunct t can be split into a purely propositional part $Prop(t)$ (a conjunction of literals), and a temporal part $Time(t)$ (a conjunction of possibly negated $\mathbf{U}$ and $\mathbf{X}$ formulæ):

$$\phi = \bigvee_{t \in Dnf(\phi)} (Prop(t) \wedge Time(t))$$

If no top-level $\mathbf{U}$s are present in a disjunct t, $Prop(t)$ univocally identifies the states for which t is satisfiable, while $Time(t)$ determines which formula remains to be verified in the future. Thus, we can use this representation to identify for which portions of a belief B is an LTL formula ϕ satisfiable, and which LTL formula must they satisfy in the future, by unrolling the $\mathbf{U}$s, and by partitioning the belief according to the propositional parts of the disjuncts of ϕ:

$$\{ \langle B \cap Prop(t), Time(t) \rangle : t \in Dnf(Unroll(\phi)) \}$$

Applying this idea, the initialization and progression of search nodes given in section 5.1 are represented as follows:

$$\mathcal{B}_{\mathcal{I}} = \{ \langle \mathcal{I} \cap Prop(t), Time(t) \rangle : t \in Dnf(Unroll(\mathcal{H})), \mathcal{I} \cap Prop(t) \neq \emptyset \}$$

$$\mathcal{E}(\mathcal{B}, o) = \{ \langle B \cap [\![o]\!], \varphi \rangle : \langle B, \varphi \rangle \in \mathcal{B}, B \cap [\![o]\!] \neq \emptyset \}$$

$$\mathcal{E}(\mathcal{B}, \alpha) = \{ \langle \alpha(B) \cap Prop(t), Time(t) \rangle : \langle B, \varphi \rangle \in \mathcal{B}, t \in Dnf(X^{-1}(\varphi)), \alpha(B) \cap Prop(t) \neq \emptyset \}$$

The indistinguishable sets can also be represented symbolically, and their initialization and progression follow the same ideas above; notice that also the pruning operation is

represented symbolically:
$\langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle = \langle prune(\mathcal{B}^0_{\mathcal{H}}, \mathcal{B}^0_{\bar{\mathcal{H}}}), prune(\mathcal{B}^0_{\bar{\mathcal{H}}}, \mathcal{B}^0_{\mathcal{H}}) \rangle$
where
$\mathcal{B}^0_{\mathcal{H}} = \{\langle \mathcal{I} \cap Prop(t), Time(t) \rangle : t \in Dnf(Unroll(\mathcal{H})), \mathcal{I} \cap Prop(t) \neq \emptyset\}$
$\mathcal{B}^0_{\bar{\mathcal{H}}} = \{\langle \mathcal{I} \cap Prop(t), Time(t) \rangle : t \in Dnf(Unroll(\bar{\mathcal{H}})), \mathcal{I} \cap Prop(t) \neq \emptyset\}$
$prune(\mathcal{B}, \mathcal{B}') = \{\langle prune_D(B, \mathcal{B}'), \varphi \rangle : \langle B, \varphi' \rangle \in \mathcal{B}\}$
$prune_D(B, \mathcal{B}') = B \cap \bigcup_{o \in \mathcal{X}(B) \cap \mathcal{X}(St(\mathcal{B}'))} [\![o]\!]$

6 Experimental evaluation

Our experiments intend to evaluate the impact of exploiting LTL assumptions, and of enforcing safety, when generating plans and executing them in a reactive framework. For these purposes, we took the SYPEM reactive platform (Bertoli *et al.* 2001), and modified the MBP planner inside it so that it generates safe LTL assumption-based plans. We name SLAM (Safe LTL Assumption-based MBP) our extension of MBP; SLAM can also be run with the safety check disabled, thus performing unsafe LTL assumption-based planning.

We also adapted the monitoring component of SYPEM to detect the failure of LTL assumptions; we will name SALPEM the reactive platform including SLAM and the adapted monitor. Basically, the monitor of SALPEM progresses the symbolic annotated belief associated to the assumption via the $\mathcal{E}$ function, and prunes away states inconsistent with the actual observations, signaling assumption failure when such annotated belief is empty.

We tested SALPEM on a set of problems, using a Linux equipped 2GHz Pentium 4 with 224MB of RAM memory.

Grid crossing We first consider a navigation problem where a robot must traverse a square grid, from the south-west corner to the north-east one. There can be obstacles (persons) in the grid, and of course the robot cannot step over them; the number and positions of these obstacles is unknown. The robot can move of one cell at a time in the four compass directions, and it can interrogate its proximity sensors to detect obstacles around him. A strong plan does not exist; however, it we take the assumption that no obstacle is ever met by the robot, a plan can be found rather easily. Fig. 2 reports the timings for safe and unsafe planning. Enforcing safety imposes a small overhead, but pays off at run-time: when the assumption holds, safe plans reach the goal without replanning, while unsafe plans cause infinite replanning episodes. This is because, while safe plans exploit sensing at every step, unsafe plans do not, blindly trusting the assumption; but then, the monitor cannot establish whether moving is actually possible and, to avoid a possible run-time failure, replans. Notice that this can (and does) happen before any progress is made by the plan, preventing termination of the plan-execution process.

Production chain We now consider a production chain which must transform an initially raw item by applying a sequence of manufacture phases to get the item in a final, refined form. Each phase, modeled as an action, prepares

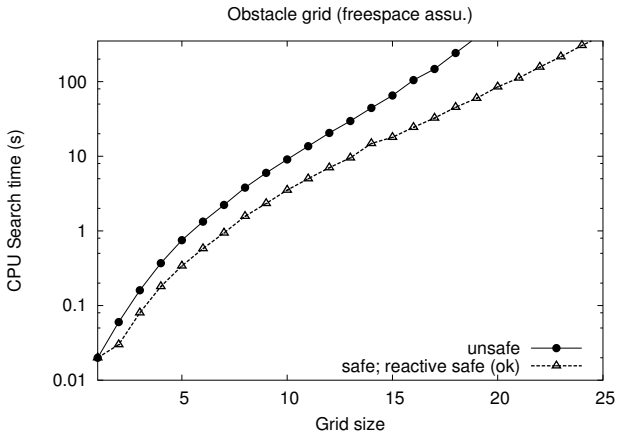

Figure 2: Performances for "free" grid navigation.

the item for the next one, but may also fail, ruining the item and making it unworkable. An item inspection operation can be commanded, which allows detecting whether the item is ruined. Of course no strong plan exists to produce a refined item, unless one assumes that no failure occurs. The only safe solution under this assumption strictly alternates manufacture phases and inspections; unsafe solutions discard some or all of the inspections. But during the execution of an unsafe solution, any attempt to perform a manufacture phase without a prior inspection causes replanning, since the monitor cannot establish whether the item is workable.

The results of our experiments are shown in Fig. 3, considering both the initial plan generation in isolation, and the overall reactive loop execution. Enforcing safety imposes no overhead at plan generation time: the cost of computing safety conditions appears to be balanced by the pruning induced in the search.

Moreover, as expected, when the assumption holds (denoted (ok)), executing the safe solution achieves the goal with no replanning; using unsafe solutions instead causes replanning sequences that, in general, do not terminate, as confirmed by our tests. This is due to the fact that unsafe solutions do not guarantee any progress when executed: indeed, given the assumption, they can attempt a manufacturing phase as the first step, in which case, unless the item is known a priori to be workable, replanning is immediately triggered. To have unsafe replanning sequences terminate, we had to adopt an ad-hoc heuristic that forces inspecting as the first action; in spite of this, replanning sequences can be rather long, and cause an overhead roughly proportional to the length of the chain, as reported in the figure.

We then considered a variation of this domain, where each manufacture phase may fail at most once, and its effects can be reverted by a specific action. Then, a strong plan exists; however, Fig. 4 shows that strong planning becomes very hard but for small domain instances, due to the high number of branches to be considered; safe and unsafe assumption-based plan generation scale up much better, exhibiting similar performances. Once more, when the assumption holds, the execution of safe solutions achieves the goal without re-

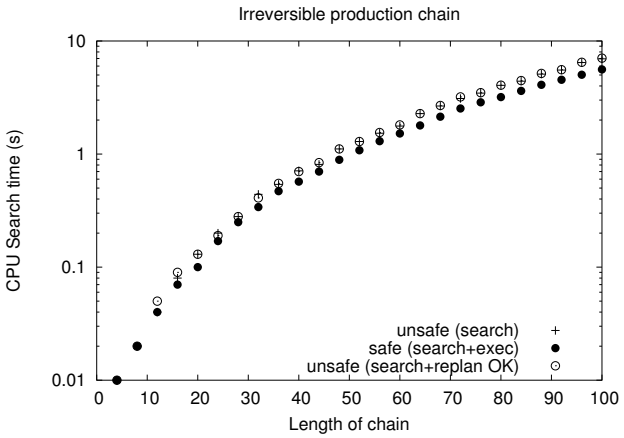

Figure 3: Performances for production chain.

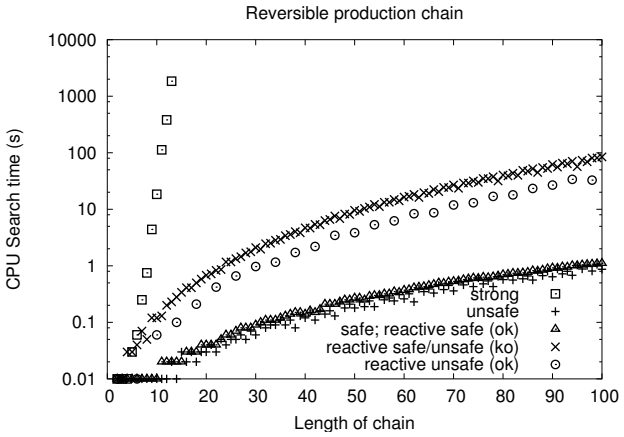

Figure 4: Performances for reversible production chain.

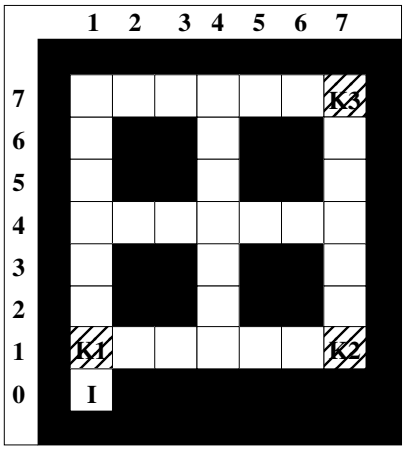

Figure 5: Slippery labyrinth of size 7.

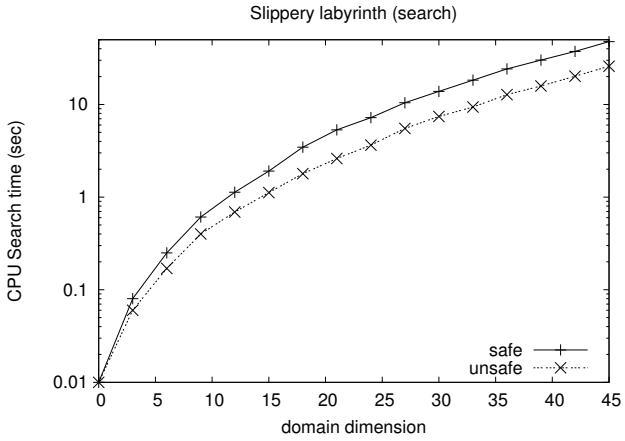

Figure 6: Performances for slippery labyrinth.

planning, while unsafe solutions cause infinite sequences of replannings, unless the ad-hoc heuristic that forces immediate inspecting is used. We observe that instead, enforcing safety guarantees termination of the reactive loop, even when replanning is actually necessary (i.e. when the assumption does not hold, denoted (ko)) [2]. For unsafe planning, our experiments using the ad-hoc heuristic show that the performance of the plan-execution is basically independent on whether the assumption actually holds. In particular, the timings are very close to those obtained by using safe planning when the assumption does not hold.

Slippery labyrinth This testing domain is a generalization of the domain of example 1. A robot can move in the four directions within a grid of slippery corridors; parallel corridors are 3 rooms apart. The rooms in the south-west, south-east and north-east corners are kitchens, which the robot is equipped to recognize by a smell sensor. The robot must reach the north-east corner from the southern-most room at the west end. An instance of the domain is shown in Fig. 5.

In this case, no strong solution exists, while, considering the assumption that the robot never slips, safe and unsafe assumption-based solutions can be found.

Fig.6 reports the time for synthesizing safe and unsafe solutions by SLAM under such assumption, for increasing sizes of the domain. We observe that the safety check imposes a limited overhead over the unsafe LTL generation. At the same time, notice that, in this domain, executing an unsafe plan always causes replanning, even when the assumption holds. That is, when the assumption holds, generating unsafe plans causes the plan-execution loop to never terminate, while enforcing safety guarantees that the goal is reached (and without ever replanning). This is confirmed by our experiments with SALPEM.

When the assumption does not hold, safe and unsafe plans behave similarly, triggering replanning; the results of replanning depend again on whether the plan newly generated is

[2] In this case it is also possible to rule out in a principled way the possibility of infinite replanning episodes for unsafe planning, e.g. by giving up any assumption upon replanning; our tests have shown that then the performance degrades to that of strong planning.

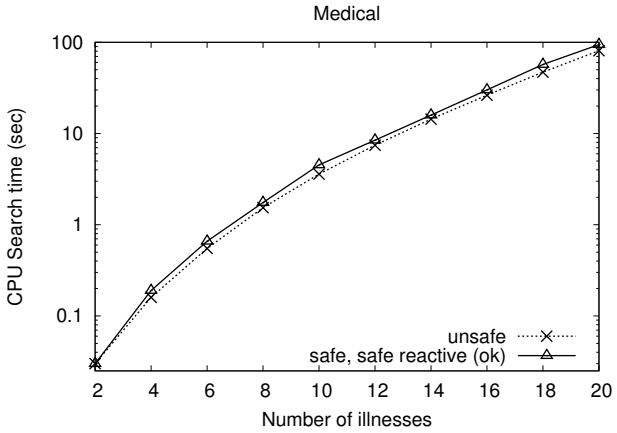

Figure 7: Performances for the medical domain.

safe or unsafe, and on whether the assumption will actually hold.

Medical The medical domain was first introduced in (Weld *et al.* 1998); a patient may suffer of one of several illnesses; every illness as a cure, but the cure is deadly if applied on an illness different from the one it has been designed for. It is possible to diagnose the illness by exploiting diagnosis actions: measuring the white cells in the blood, or using a litmus paper. In our modeling, we also consider the possibility that a cure, although applied on the right illness, may not heal the patient. Of course then, there is no strong plan to heal the patient, and a solution can only be found if we assume that the cure will not fail. The results of our experiments are shown in Fig.7; once more, enforcing safety only adds a very limited overhead on unsafe planning, but unsafe planning is practically useless, since it will cause replanning even when the assumption holds, and such replanning instances may be infinite unless a safe plan is generated by chance.

The results of our experiments confirm the intuitions that motivate our approach:

- using assumptions is extremely advantageous, since strong plans rarely exist, and even when they do, they are very difficult to build.
- in nondeterministic domains, assumptions concern in most cases the dynamics of the domain; thus, the availability of a temporal logics such as LTL to write them is crucial.
- enforcing safety dramatically improves the run-time behavior of plans: the execution of unsafe plans may be interrupted by useless replanning episodes, and in many cases these repeat infinitely, inhibiting goal achievement even though the goal could be reached.
- enforcing safety adds no relevant overhead at plan generation time: the cost of computing safety conditions is alleviated by the usage of symbolic techniques, and appears to be balanced by the safety-induced search pruning.

7 Conclusions

In this paper, we tackled the problem of efficiently building assumption-based solutions partially observable and known domains, using an expressive logics, LTL, to describe assumptions over the domain dynamics. Moreover, we constrained the search to generate safe solutions, which allow a monitor to unambiguously identify, at run-time, domain behaviors unplanned for. Both of our contributions are, to the best of our knowledge, novel, and substantially extend existing works on related research lines.

In particular, LTL has been used as a means to describe search strategies (and goals) in (Bacchus and Kabanza 2000; Calvanese *et al.* 2002). While also that work exploits LTL to restrict the search, it focuses on the much simpler framework of fully deterministic and observable planning. This greatly simplifies the problem: no ambiguous monitoring result is possible (thus safety is guaranteed), and since the nodes of the search space are single states, it is possible to simply evaluate the progression of formulæ over states, without recurring to symbolic representation techniques to progress beliefs. On the contrary, we had to consider safety, and to tailor our algorithms by efficiently exploiting symbolic representation techniques to progress and evaluate formulæover beliefs, rather than over single states.

A first, less general notion of safety is first presented in (Albore and Bertoli 2004), considering a very limited, propositional assumption language, by which it is only possible to state assumptions over the initial state. The ability of representing assumptions over the dynamics of the domain is crucial to the applicability of assumption-based planning; none of the examples presented here could have been handled by such a propositional assumption language. At the same time, using temporal logic assumptions implies a much more complex treatment of the way beliefs are progressed during the search.

While several planning frameworks and algorithms have appeared that deal with (some variety of) nondeterministic, partially observable domains, e.g. (Bonet and Geffner 2000; Rintanen 2002), they are unable so far to deal with assumptions. Their extensions in this direction, especially considering temporal assumptions, is far from trivial, and a direct comparison is currently unfeasible.

The notion of safety is closely related to the one of *diagnosability* (Sampath *et al.* 1995). In essence, a system is diagnosable for a certain condition ϕ if and only if it cannot produce two observationally indistinguishable traces associated with different values of ϕ. In (Cimatti *et al.* 2003; Jiang and Kumar 2004), LTL conditions are considered, and the diagnosability of a system is verified by model-checking techniques. With safe planning, we enforce a specific form of diagnosability by construction: the system composed by the given planning domain and the synthesized safe plan, on top of guaranteeing goal achievement proviso assumptions, is diagnosable for what concerns the executability of actions and the achievement of the goal. An in-depth study of the relationships between diagnosability and safe planning is in our agenda; in particular, we intend to investigate the way in which (safe) planning may take advantage of analysis performed by on-line over the domain.

Several other future directions of work are open. First, in our work, assumptions are an explicit input to the planning process; e.g. they could be provided by a user with a knowledge of the domain. While it is often practically feasible to formulate assumptions (e.g. on nominal behaviors), it would be extremely useful to be able to (semi-)automatically extract and formulate assumptions from the domain description, possibly learning by previous domain behaviors. Finally, in certain cases, a more qualitative notion of safety can be useful to practically tune the search to accept 'almost safe' plans, or score the quality of plans depending on a degree of safety.

References

A. Albore and P. Bertoli. Generating Safe Assumption-Based Plans for Partially Observable, Nondeterministic Domains. In *Proc. of AAAI'04*, pages 495–500, 2004.

F. Bacchus and F. Kabanza. Using Temporal Logic to Express Search Control Knowledge for Planning. *Artificial Intelligence*, 116(1-2):123–191, 2000.

P. Bertoli, A. Cimatti, and M. Roveri. Conditional Planning under Partial Observability as Heuristic-Symbolic Search in Belief Space. In *Proc. of 6th European Conference on Planning (ECP'01)*, 2001.

B. Bonet and H. Geffner. Planning with Incomplete Information as Heuristic Search in Belief Space. In *Proc. of Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000)*, pages 52–61, 2000.

R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

D. Calvanese, G. De Giacomo, and M. Y. Vardi. Reasoning about actions and planning in LTL action theories. In *Proc. of the 8th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2002)*, pages 593–602, 2002.

A. Cimatti, C. Pecheur, and R. Cavada. Formal Verification of Diagnosability via Symbolic Model Checking. In *Proc. of 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.

E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*. Elsevier, 1990.

R. Hähnle and O. Ibens. Improving temporal logic tableaux using integer constraints. In *Proc. of the First International Conference on Temporal Logic (ICTL '94)*, pages 535–539, London, UK, 1994. Springer-Verlag.

S. Jiang and R. Kumar. Failure Diagnosis of discrete event systems with Linear-time Temporal Logic fault specifications. *IEEE Transactions on Automatic Control*, 49(6):934–945, 2004.

F. Kabanza, M. Barbeau, and R. St-Denis. Planning Control Rules for Reactive Agents. *Artificial Intelligence*, 95(1):67–113, 1997.

Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, 1992.

N. Muscettola, P. P. Nayak, B. Pell, and B. C. Williams. Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, 103(1-2):5–47, 1998.

K. L. Myers and D. E. Wilkins. Reasoning about locations in theory and practice. *Computational Intelligence*, 14(2):151–187, 1998.

J. Rintanen. Backward plan construction for planning with partial observability. In *Proc. of 6th International Conference on Artificial Intelligence Planning and Scheduling (AIPS02)*, pages 73–182, 2002.

M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

D. Weld, C. Anderson, and D. Smith. Extending Graphplan to Handle Uncertainty and Sensing Actions. In *Proc. of AAAI'98*, pages 897–904, 1998.

Appendix

We now show how the safety of a plan π can be determined while progressing and pruning annotated beliefs as stated in section 4, checking goal entailment on the leaves of an associated execution (and-or) tree. We do so in three steps.

First, we define the tree structure associated to a plan π and an assumption $\mathcal{H}$ as a collection of nodes of the form $\langle \mathcal{B}, \langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, p \rangle$. The accessibility relation between nodes is implicitly defined by the prefix relation among paths. This structure will be built following the progression and pruning rules described so far. Second, we show that for each node $\langle \mathcal{B}, \langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, p \rangle$, $\mathcal{B}$ contains the final states of the traces along p for which $\mathcal{H}$ is satisfiable. Third, we prove that for each node $\langle \mathcal{B}, \langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, p \rangle$, if t and t' are two indistinguishable traces along p for which $\mathcal{H}$ is respectively satisfiable and unsatisfiable, then $final(t') \in \mathcal{B}_{\bar{\mathcal{H}}}$.

Definition 12 (Tree structure induced by a plan). *Let π be a conditional plan for a domain $\mathcal{D} = \langle \mathcal{S}, \mathcal{A}, \mathcal{U}, \mathcal{I}, \mathcal{T}, \mathcal{X} \rangle$, and let $\mathcal{H} \in \mathcal{L}(\mathcal{P})$. The tree structure induced by π from $\mathcal{I}$, denoted $\tau(\pi, \mathcal{I}, \mathcal{H})$, is defined as follows:*

- $\tau(\pi, \mathcal{I}, \mathcal{H}) = \tau_0(\pi, \mathcal{B}_{\mathcal{I}}, \langle \mathcal{B}^0_{\mathcal{H}}, \mathcal{B}^0_{\bar{\mathcal{H}}} \rangle, \epsilon)$, *where $\mathcal{B}_{\mathcal{I}}$ is defined as in* (1)*, and $\langle \mathcal{B}^0_{\mathcal{H}}, \mathcal{B}^0_{\bar{\mathcal{H}}} \rangle$as in* (4)*.*
- $\tau_0(\epsilon, \mathcal{B}, \langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, p) =$
  $\{\langle \mathcal{B}, \langle prune\big(\mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}}\big), prune(\mathcal{B}_{\bar{\mathcal{H}}}, \mathcal{B}^0_{\mathcal{H}}) \rangle, p \rangle\}$
- $\tau_0(\alpha \circ \pi, \mathcal{B}, \langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, p) = \tau_0(\epsilon, \mathcal{B}, \langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, p) \cup$
  $\tau_0(\pi, \mathcal{E}(\mathcal{B}, \alpha), \mathcal{E}(\langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, \alpha), p \circ \alpha)\}$
  if $\alpha\big(St(\mathcal{B})\big) \neq \emptyset \ \vee \ \alpha\big(St(\mathcal{B}_{\bar{\mathcal{H}}}\big) \neq \emptyset$
- $\tau_0(\alpha \circ \pi, \mathcal{B}, \langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, p) =$
  $\{\langle Fail, \langle Fail, Fail \rangle, p \circ Fail(\alpha) \rangle\}$
  if $\alpha\big(St(\mathcal{B})\big) = \emptyset \ \vee \ \alpha\big(St(\mathcal{B}_{\bar{\mathcal{H}}}\big) = \emptyset$*. We call nodes whose annotated beliefs are $Fail$ "failure nodes".*
- $\tau_0(\text{if } o \text{ then } \pi_1 \text{ else } \pi_2, \mathcal{B}, \langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, p) =$
  $\tau_0(\pi_1, \mathcal{E}(\mathcal{B}, o), \mathcal{E}(\langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, o), p \circ o) \cup$
  $\tau_0(\pi_2, \mathcal{E}(\mathcal{B}, \tilde{o}), \mathcal{E}(\langle \mathcal{B}_{\mathcal{H}}, \mathcal{B}_{\bar{\mathcal{H}}} \rangle, \tilde{o}), p \circ \tilde{o})$

We denote the leaves of $\tau(\pi, \mathcal{I}, \mathcal{H})$ with $leaf\big(\tau(\pi, \mathcal{I}, \mathcal{H})\big)$.

An annotated belief associated to a graph node contains the final states of the traces $Trs_{\mathcal{H}}(p,\mathcal{D})$ which traverse the path p associated to the node.

Lemma 1. *Let $\mathcal{D}=\langle\mathcal{S},\mathcal{A},\mathcal{U},\mathcal{I},\mathcal{T},\mathcal{X}\rangle$ be a planning domain, $\mathcal{H}\in\mathcal{L}(\mathcal{P})$, and π a conditional plan for $\mathcal{D}$. Then $\forall\langle\mathcal{B},\langle\mathcal{B}_{\mathcal{H}},\mathcal{B}_{\bar{\mathcal{H}}}\rangle,p\rangle\in\tau(\pi,\mathcal{I},\mathcal{H}):St(\mathcal{B})=\{final(t):\forall t\in Trs_{\mathcal{H}}(p,\mathcal{D})\}$.*

Proof. The lemma is proved by induction on the length of p. We exploit the fact that tableaux rewriting rules like the $Unroll$ function preserve satisfiability of the rewritten formula (Hähnle and Ibens 1994); thus, since $Unroll(\mathcal{H})|_s\equiv\mathcal{H}|_s$, we have that $Trs_{\mathcal{H}}(p,\mathcal{D})=Trs_{Unroll(\mathcal{H})}(p,\mathcal{D})$.

The *base case* is trivial: the path p is empty, and the root of $\tau(\pi,\mathcal{I},\mathcal{H})$ is built according to (1). Consequently its annotated belief contains every final state of $Trs_{\mathcal{H}}(\epsilon,\mathcal{D})$.

In the *induction step* we assume that for a node $\langle\mathcal{B}',\langle\mathcal{B}'_{\mathcal{H}},\mathcal{B}'_{\bar{\mathcal{H}}}\rangle,p'\rangle$ the lemma holds, and we prove it for a direct descendant, i.e for a node $\langle\mathcal{B},\langle\mathcal{B}_{\mathcal{H}},\mathcal{B}_{\bar{\mathcal{H}}}\rangle,p\rangle$ such that $|p|=|p'|+1$ and p' is a prefix of p. Three cases arise:

- if $p=p'\circ\alpha$, the belief $St(\mathcal{B})$ is progressed folllowing (3), so $St(\mathcal{B})=\{\alpha(s):s\in St(\mathcal{B}'),X^{-1}(\mathcal{H})|_{\alpha(s)}\neq\bot\}$. Given that $St(\mathcal{B}')=\{final(t')\}$, from the latter operation we have that $St(\mathcal{B})=\{s\in final(t):t=t'\circ[s,o,\alpha]:\forall t'\in Trs_{\mathcal{H}}(p',\mathcal{D}),o\in\mathcal{X}(s)\}$; this belief corresponds to the set of final states of the traces bound to path $p=p'\circ\alpha$ (cf. def. 6).
- If $p=p'\circ o$, from expansion rule (2), the belief $St(\mathcal{B})=St(\mathcal{E}(\mathcal{B}',o))=\{final(t')\}\cap[\![o]\!]$, indicating all those states in $\{final(t):\forall t=Trs_{\mathcal{D}}\big(o\circ p',\mathcal{I}\big)=Trs_{\mathcal{D}}\big(p,\mathcal{I}\big)\}$. From def. 6 we have that $t\in Trs_{\mathcal{H}}(p,\mathcal{D})$.
- If $p=p'\circ\tilde{o}$, the case is very similar to the latter. Again, from def. 6 we observe that the expansion of $\mathcal{B}'$ by $\tilde{o}$ corresponds to the final states of the traces $t\in Trs_{\mathcal{H}}(p,\mathcal{D})\subseteq Trs_{\mathcal{D}}\big(p'\circ\tilde{o},\mathcal{I}\big)=Trs_{\mathcal{D}}\big(p,\mathcal{I}\big)$. □

Lemma 2. *Let $\mathcal{D}=\langle\mathcal{S},\mathcal{A},\mathcal{U},\mathcal{I},\mathcal{T},\mathcal{X}\rangle$ be a planning domain, a formula $\mathcal{H}\in\mathcal{L}(\mathcal{P})$, and let π be a conditional plan for $\mathcal{D}$. Let $\langle\mathcal{B},\langle\mathcal{B}_{\mathcal{H}},\mathcal{B}_{\bar{\mathcal{H}}}\rangle,p\rangle$ be a node of $\tau(\pi,\mathcal{I},\mathcal{H})$. Then,*

$$\forall t\in Trs_{\mathcal{H}}(p,\mathcal{D}),\ \forall t'\in Trs_{\bar{\mathcal{H}}}(p,\mathcal{D}):\ \neg Dist(t,t')\implies final(t')\in St(\mathcal{B}_{\bar{\mathcal{H}}})$$

Proof. We prove the lemma by induction of the length of p. We show that in the case of not distinguishable traces, the pruning operation is ineffective, and therefore the path traces end in the beliefs progressed by satisfying the assumption.

From def.10 and the hypothesis, we have that t,t' are bound to a path p, as defined in def. 6, consequently $\forall t=[\overline{s},\overline{o},\overline{\alpha}]\in Trs_{\mathcal{H}}(p,\mathcal{D}),\forall t'=[\overline{s'},\overline{o'},\overline{\alpha'}]\in Trs_{\bar{\mathcal{H}}}(p,\mathcal{D})$

$$\neg Dist(t,t')\Rightarrow\bar{o}=\bar{o'}\wedge\bar{\alpha}=\bar{\alpha'},\quad\text{and}$$
$$\bar{o}=\bar{o'}\Rightarrow\forall n\leq|t|,\forall s'\in t',\forall s\in t:\mathcal{X}(s^n)\cap\mathcal{X}(s'^n)\neq\emptyset \tag{5}$$

The traces t,t' are bound to the same path p, which implies that $\bar{\alpha}=\bar{\alpha'}$.

We consider, for the *base case* of the proof by induction, a pair of annotated beliefs $\langle\mathcal{B}^0_{\mathcal{H}},\mathcal{B}^0_{\bar{\mathcal{H}}}\rangle$, defined as in (4). It is trivial to see that all the final states of t' (resp. t) are in $St(\mathcal{B}^0_{\bar{\mathcal{H}}})$ (resp. $St(\mathcal{B}^0_{\mathcal{H}})$). Following (4), the pair $\langle\mathcal{B}_{\mathcal{H}},\mathcal{B}_{\bar{\mathcal{H}}}\rangle$ is built up by applying $prune$ to $\mathcal{B}^0_{\mathcal{H}}$ and $\mathcal{B}^0_{\bar{\mathcal{H}}}$. From (5) and the definition of $prune$ function, it comes that $prune(\mathcal{B}^0_{\mathcal{H}},\mathcal{B}^0_{\bar{\mathcal{H}}})=\mathcal{B}^0_{\mathcal{H}}$, and that $prune(\mathcal{B}^0_{\bar{\mathcal{H}}},\mathcal{B}^0_{\mathcal{H}})=\mathcal{B}^0_{\bar{\mathcal{H}}}$, which proves the lemma at base step.

By *induction step* we assume that, given a pair of annotated belief $\langle\mathcal{B}_{\mathcal{H}},\mathcal{B}_{\bar{\mathcal{H}}}\rangle$ progressed on p, via the function $\mathcal{E}$: $\forall t'\in Trs_{\bar{\mathcal{H}}}(p,\mathcal{D}):final(t')\in St(\mathcal{B}_{\bar{\mathcal{H}}})$. We now prove the lemma on the progression of $\langle\mathcal{B}_{\mathcal{H}},\mathcal{B}_{\bar{\mathcal{H}}}\rangle$.

- If an observation o is applied, then $\mathcal{E}(\langle\mathcal{B}_{\mathcal{H}},\mathcal{B}_{\bar{\mathcal{H}}}\rangle,o)=\langle\mathcal{B}'_{\mathcal{H}},\mathcal{B}'_{\bar{\mathcal{H}}}\rangle$, i.e. $St(\mathcal{B}'_{\bar{\mathcal{H}}})=\{s\in\mathcal{B}_{\bar{\mathcal{H}}}:s\in[\![o]\!]\}$, as defined in (2). Thus $St(\mathcal{B}'_{\bar{\mathcal{H}}})=\{final(t''):\forall t''\in Trs_{\bar{\mathcal{H}}}(p\circ o,\mathcal{D})\}$.
- If an observation $\tilde{o}$ is applied, then the case is very similar to the latter, and we observe that if $\mathcal{E}(\langle\mathcal{B}_{\mathcal{H}},\mathcal{B}_{\bar{\mathcal{H}}}\rangle,\tilde{o})=\langle\mathcal{B}'_{\mathcal{H}},\mathcal{B}'_{\bar{\mathcal{H}}}\rangle$, then $St(\mathcal{B}'_{\bar{\mathcal{H}}})=\{final(t''):\forall t''\in Trs_{\bar{\mathcal{H}}}(p\circ\tilde{o},\mathcal{D})\}$.
- If an action α is applied, then $\mathcal{E}(\langle\mathcal{B}_{\mathcal{H}},\mathcal{B}_{\bar{\mathcal{H}}}\rangle,\alpha)=\langle\mathcal{B}'_{\mathcal{H}},\mathcal{B}'_{\bar{\mathcal{H}}}\rangle$, where $\mathcal{B}'_{\bar{\mathcal{H}}}=prune\big(\mathcal{E}(\mathcal{B}_{\bar{\mathcal{H}}},\alpha),\mathcal{E}(\mathcal{B}_{\mathcal{H}},\alpha)\big)$. Given that $\forall t'\in Trs_{\bar{\mathcal{H}}}(p,\mathcal{D}):final(t')\in St(\mathcal{B}_{\bar{\mathcal{H}}})$, from the latter operation we have that $St(\mathcal{E}(\mathcal{B}_{\bar{\mathcal{H}}},\alpha))=\{s\in final(t''):t''=t'\circ[s,o,\alpha]:\forall t'\in Trs_{\bar{\mathcal{H}}}(p,\mathcal{D}),o\in\mathcal{X}(s)\}$; this belief corresponds to the set of final states of the traces bound to path $p'=p\circ\alpha$. From the relation (5), applying the pruning operator to build $\mathcal{B}'_{\bar{\mathcal{H}}}$ brings that $St(\mathcal{B}'_{\bar{\mathcal{H}}})=\mathcal{E}(\mathcal{B}_{\bar{\mathcal{H}}},\alpha)$, i.e. no states are pruned because of indistinguishability of the traces. Consequently, this brings to the set of final states of the trace: $\{final(t):\forall t'\in Trs_{\bar{\mathcal{H}}}(p\circ\alpha,\mathcal{D})\}\subseteq St(\mathcal{B}'_{\bar{\mathcal{H}}})$. □

Theorem 2. *Let π be a conditional plan for a domain $\mathcal{D}=\langle\mathcal{S},\mathcal{A},\mathcal{U},\mathcal{I},\mathcal{T},\mathcal{X}\rangle$, let $\mathcal{G}\subseteq\mathcal{S}$ be a goal, and let $\mathcal{H}\in\mathcal{L}(\mathcal{P})$ be an assumption. If $\tau(\pi,\mathcal{I},\mathcal{H})$ does not contain the Fail node, and $\forall\langle\mathcal{B},\langle\mathcal{B}_{\mathcal{H}},\mathcal{B}_{\bar{\mathcal{H}}}\rangle\rangle\in leaf\big(\tau(\pi,\mathcal{I},\mathcal{H})\big):St(\mathcal{B})\cup St(\mathcal{B}_{\bar{\mathcal{H}}})\subseteq\mathcal{G}$, then π is a safe solution for $\mathcal{G}$ under $\mathcal{H}$.*

Proof. Let us consider a generic leaf $\langle\mathcal{B},\langle\mathcal{B}_{\mathcal{H}},\mathcal{B}_{\bar{\mathcal{H}}}\rangle\rangle$ of $\tau(\pi,\mathcal{I},\mathcal{H})$.

From lemma 1, we derive that $\forall t\in Trs_{\mathcal{H}}(\pi,\mathcal{D}):final(t)\in\mathcal{G}$, i.e. that

$$Trs_{\mathcal{H}}(\pi,\mathcal{D})=TrsG(\pi,\mathcal{D},\mathcal{G}) \tag{6}$$

From lemma 2, we derive that $\forall t'\in Trs_{\bar{\mathcal{H}}}(\pi,\mathcal{D}):final(t')\in\mathcal{G}$, i.e. that

$$Trs_{\bar{\mathcal{H}}}(\pi,\mathcal{D})=TrsG_{\bar{\mathcal{H}}}(\pi,\mathcal{D},\mathcal{G}) \tag{7}$$

By conjoining (6) and (7) over every leaf of $\tau(\pi,\mathcal{I},\mathcal{H})$, we obtain def.11. □