Assumption-Based Planning: Generating Plans and Explanations under Incomplete Knowledge

Sammy Davis-Mendelow
Department of Computer Science
University of Toronto
Toronto, Canada

Jorge A. Baier
Depto. de Ciencia de la Computación
Pontificia Universidad Católica de Chile
Santiago, Chile

Sheila A. McIlraith
Department of Computer Science
University of Toronto
Toronto, Canada

Abstract

Many practical planning problems necessitate the generation of a plan under incomplete information about the state of the world. In this paper we propose the notion of Assumption-Based Planning. Unlike conformant planning, which attempts to find a plan under all possible completions of the initial state, an assumption-based plan supports the assertion of additional assumptions about the state of the world, often resulting in high quality plans where no conformant plan exists. We are interested in this paradigm of planning for two reasons: 1) it captures a compelling form of *commonsense planning*, and 2) it is of great utility in the generation of explanations, diagnoses, and counter-examples – tasks which share a computational core with planning. We formalize the notion of assumption-based planning, establishing a relationship between assumption-based and conformant planning, and prove properties of such plans. We further provide for the scenario where some assumptions are more preferred than others. Exploiting the correspondence with conformant planning, we propose a means of computing assumption-based plans via a translation to classical planning. Our translation is an extension of the popular approach proposed by Palacios and Geffner and realized in their T0 planner. We have implemented our planner, A0, as a variant of T0 and tested it on a number of expository domains drawn from the International Planning Competition. Our results illustrate the utility of this new planning paradigm.

Introduction

Many real-world planning problems provide limited information about the actual state of the world, and yet for a number of tasks it is necessary to generate a reasonable plan prior to execution, without the benefit of run-time sensing. Planning a major project in which a reasonable plan must be delivered to stakeholders is one such example. Planning your day is another. Such forms of commonsense planning are prevalent. A third example of this paradigm of planning is found in the generation of explanations or diagnoses for dynamical systems.

When information is limited and plans must be generated without the benefit of run-time sensing, a common approach is conformant planning, which computes a plan that relies only on what is known. This can make planning difficult and can lead to poor-quality plans or no plan at all.

In this paper we define the notion of *assumption-based planning* (ABP). Assumption-based planning endeavors to find a middle-ground between conformant and classical planning wherein the planner computes a set of calculated assumptions about aspects of the world that support the generation of a plan. Assumption-based planning is well-suited to scenarios where resolving uncertainty directly is impossible, difficult, or expensive.

Our interest in assumption-based planning is two-fold: 1) it captures a compelling form of *commonsense planning*[1], and 2) it is of great utility for tasks that share a computational core with planning, such as explanation generation (e.g., Göbelbecker et al. 2010, Sohrabi, Baier, and McIlraith 2011), plan recognition (e.g., Ramírez and Geffner 2010), diagnosis of dynamical systems (e.g., Sohrabi, Baier, and McIlraith 2010, Haslum and Grastien 2011) and counter-example generation for verification (e.g., Albarghouthi, Baier, and McIlraith 2009). To illustrate, consider planning your trip home at the end of a work day. You don't know for certain that the subway will be running, you have no way of finding out, but it's reasonable to assume so. Making this assumption supports generation of a reasonable plan. The conformant plan might have you walking home! Similarly consider a sequence of observations and the task of finding a sequence of actions (a plan/explanation) that account for them. In such a scenario, there is no opportunity to sense, because the observations occurred in the past. An explanation typically necessitates reasonable assumptions about unobserved state or events. For the related task of counter-example generation in service of verification, *any* assumptions that can be consistently made to generate the counter-example are desirable.

The term assumption-based planning has been coined for a number of diverse planning activities that broadly relate to assumptions. Albore and Bertoli (2004) used the term to describe planning with linear temporal logic "assumptions" provided in advance. Pellier and Fiorino's basic formulation (2004) shares some commonalities with ours but the approach to plan generation is fundamentally different. Our characterization of assumption-based planning is related to

[1]Coined previously but with different meaning in (Faletti 1982).

abduction as theory formation (e.g., Poole, Goebel, and Aleliunas 1987) wherein additional facts about the world are conjectured to explain an observation.

Assumption-based planning shares commonalities with contingent and probabilistic planning. Indeed, the translation-based approach to conformant planning we exploit for assumption-based planning has been extended in different ways for probabilistic planning (Brafman and Taig 2011) and for contingent planning (Albore, Palacios, and Geffner 2009). Some contingent and probabilistic planners have also exploited a form of assumptions. Brafman and Shani (e.g., 2012) make implicit assumptions informed by a probability distribution, and by online sensing. Göbelbecker, Gretton, and Dearden (2011) do something similar for decision-theoretic planning. Bonet and Geffner (2011) make optimistic assumptions about the value of unknown hidden variables, executing a classical plan given these assumptions, and replanning if an assumption is refuted during execution. Finally, Albore and Geffner (2009) incorporate assumptions into the CLG+ contingent planner in order to deal with dead-end states. Rather than failing when there is the possibility of such a state, the planner assumes that these states are not possible, and only fails when the dead-end is a certainty. Beyond the fundamental difference in planning paradigm, a major difference between these works and ours is that they all combine some notion of assumption making with online execution. While this can be an important component of assumption-based planning, it need not be, and is not the focus of this paper.

Here we provide a formal characterization of assumption-based planning that can be realized offline or online and that is applicable to a diversity of tasks involving assumption-based reasoning in dynamical systems. By establishing and exploiting a correspondence between assumption-based planning and conformant planning we provide a translation of assumption-based planning to a classical planning problem, building on the popular translation developed by Palacios and Geffner (2009). We prove the soundness and completeness of our translation. This provides us with a means of generating assumption-based plans using classical planners. We also argue for the merit of preferred assumption-based plans and propose a means of realizing such plans via cost-based planning. We implement these two approaches and perform experiments to illustrate their viability and assess some of their properties. While we have implemented assumption-based planning using a translation-based approach, the established correspondence to conformant planning supports the realization of assumption-based planning via minor modification of a diversity of conformant planners.

Characterization

In this section, we formally define an assumption-based plan, analyze the complexity of plan generation, and relate it to other well-known notions of planning.

Background

Following Palacios and Geffner (2006), a *Planning Problem* is a tuple $P = (F, O, I, G)$ where F is a finite set of fluent symbols, O is a finite set of action operators, I is a set of clauses over F, defining the set of possible initial states, and G is a boolean formula over symbols in F, that defines a goal condition. Every action $a \in O$ is defined by a conjunction of fluent literals, $prec(a)$ (preconditions) and a set of conditional effects $C \rightarrow L$ where L is a fluent literal that is made true when the action is executed and the conjunction of fluent literals C holds.

Example Returning to our example of planning your trip home from work, let $P = (F, O, I, G)$, where F includes propositions describing your location, $at(loc)$, and the location of subway stations, $station(loc)$, among other properties. O includes the locomotion operators $walk(orig, dest)$, $bicycle(orig, dest)$, and $subway(orig, dest)$ that take you from origin to destination. Each of these operators has the precondition $at(orig)$; $bicycle(orig, dest)$ has the additional preconditions $haveBike$ and $dryRoad$; whereas $subway(orig, dest)$ has the additional preconditions $station(orig)$ and $station(dest)$. $walk(orig, dest)$ and $bicycle(orig, dest)$ have the effects $\neg at(orig)$ and $at(dest)$, and $subway(orig, dest)$ has the effects $subwayRunning \rightarrow at(dest), \neg at(orig)$. The initial state contains the location of all subway stations, $at(Office)$ and $\neg haveBike$. It is not known whether the subway is operational, so there is no mention of $subwayRunning$. The goal is $G = at(Home)$.

A *planning state* s is defined by a set of fluent symbols, which represent all that is true. Each system state s induces a propositional valuation $M_s : F \rightarrow \{\mathsf{true}, \mathsf{false}\}$ that maps any fluent literal in s to true, and all other literals to false.

We say a state s is *consistent* with a set of clauses $\mathcal{C}$, if $M_s \models c$, for every $c \in \mathcal{C}$. Intuitively, $M_s \models \phi$ stands for "boolean formula ϕ holds true in state s".

An action a is *executable* in a state s if $M_s \models prec(a)$. If a is executable in a state s, we define its successor state as $\delta(a, s) = (s \setminus Del) \cup Add$, where Add contains a fluent f iff $C \rightarrow f$ is an effect of a and $M_s \models C$. On the other hand Del contains a fluent f iff $C \rightarrow \neg f$ is an effect of a, and $M_s \models C$. We define $\delta(a_0 a_1 \ldots a_n, s) = \delta(a_1 \ldots a_n, \delta(a_0, s))$, and $\delta(\epsilon, s) = s$. A sequence of actions α is *executable* in s if $\delta(\alpha, s)$ is defined. Furthermore α is executable in P iff it is executable in s, for any s consistent with I.

Next we define an *execution trace* which intuitively characterizes maximal state trajectories that could result from the execution of an action sequence when performed in some of the possible initial states of a planning problem.

Definition 1 (Execution Trace) *A sequence of planning states* $\sigma = s_0 s_1 \cdots s_k$ *is an* execution trace *of* $\alpha = a_0 a_1 \ldots a_n$ *in planning problem* $P = (F, O, I, G)$ *iff (1)* s_0 *is consistent with* I*, (2)* $\delta(a_i, s_i) = s_{i+1}$*, for all* $i < k$*, and (3) either* $k = n + 1$ *or* $k < n + 1$ *and* $\delta(s_k, a_k)$ *is undefined.*

Definition 2 (Successful Execution Trace) *An execution trace* σ *for* α *is successful iff* $|\sigma| = |\alpha| + 1$.

Naturally, we are interested in execution traces that lead to goal satisfaction, i.e., for which the goal holds in the final state of the sequence of planning states. Formally,

Definition 3 (Leads to) *An execution trace $\sigma = s_0 \cdots s_k$ leads to (goal formula) G, iff $M_{s_k} \models G$.*

A standard definition of conformant plan follows.

Definition 4 (Conformant Plan) *A sequence of actions α is a conformant plan for $P = (F, O, I, G)$ iff every execution trace of α is successful and leads to G.*

Example: In our subway example, all conformant plans require you to walk home, e.g., $walk(\mathit{Office}, \mathit{Home})$. Such plans are safe at the expense of quality. For the authors' home and office, however, a plan that reasonably assumes the subway is operational and takes the subway is best.

Assumption-Based Planning

In real-world planning scenarios, where incomplete information is the norm, humans will often construct a commonsense plan based on reasonable assumptions. In the case of the subway example, they may do so in order to coordinate with friends or family. In a project planning scenario, such a plan might be necessary for stakeholder coordination. Analogously, when generating explanations for a narrative of observed state, assumptions are posited in support of generating a reasonable explanation. Indeed, the positing of assumptions is at the core of abductive reasoning. Here it is extended to dynamical systems.

Given a planning problem with an incomplete initial state, the task of assumption-based planning requires computing two elements: (1) a set of assumptions that are made at different states during the execution of the plan, and (2) a sequence of actions that, given the assumptions, is guaranteed to reach the goal. As such, the main difference between assumption-based planning and conformant planning is the computation of assumptions.

Formally an *Assumption-Based Planning Problem* is a tuple $P = (F, O, I, G, U)$. F, O, I, and G follow the previously described standard definition, but here P is augmented with U, a subset of F denoting the set of *assumable fluents* used to contruct assumptions. U may be equal to F or it may be restricted to an application-specific subset of fluents that are reasonable to assume. For example, in the case of explanation generation or diagnosis of a component-based system, U might include fluents denoting the (ab)normal functioning of system components (e.g., $AB(c_i)$). In the subway example, U would include $subwayRunning$. An assumption-based plan is a pair (ρ, α) where $\alpha = a_0 \cdots a_k$ is a sequence of actions and $\rho = h_0 \cdots h_{k+1}$ is a sequence of boolean formulae over fluents in U representing assumptions made about the i-th state visited when performing α.

The execution traces of interest are those that *conform to* ρ; i.e., those consistent with the assumptions. Formally,

Definition 5 (Conforms to) *An execution trace $\sigma = s_0 \cdots s_k$ conforms to a sequence of boolean formulae $\rho = h_0 \cdots h_n$ with $k \leq n$ iff $M_{s_i} \models h_i$, for every $i \in \{0, \ldots, k\}$.*

Finally, each of the execution traces of α that conforms to ρ must actually lead to the goal. A formal definition of an assumption-based plan follows.

Definition 6 (Assumption-Based Plan) *The pair (ρ, α), where α is a sequence of k actions, and ρ is a sequence of $k+1$ boolean formulae over U is an assumption-based plan for $P = (F, O, I, G, U)$ iff any execution trace of α that conforms to ρ is successful and leads to G, and furthermore at least one such execution trace exists.*

Intuitively for every consistent completion of the initial state the execution trace is either successful and leads to the goal or is pruned by ρ. Assumption-based planning reduces to yielding a conformant plan when assumptions are entailed by the states in which they are made.

Example: Assumption-based planning allows for the generation of the reasonable plan, that we assume the subway is running when we need to take it. I.e., $\pi = (\rho, \alpha)$, where $\rho = \mathsf{true}; subwayRunning; \mathsf{true}; \mathsf{true}$, and $\alpha = walk(\mathit{Office}, \mathit{StnA}); subway(\mathit{StnA}, \mathit{StnB}); walk(\mathit{StnB}, \mathit{Home})$.

The astute reader will note that restricting the vocabulary of assumptions via U may not be sufficient to ensure reasonable plans. At the extreme one could imagine simply assuming the goal is already true in I. This issue can be addressed by defining a notion of quality over assumption-based plans. We discuss this in more detail later in the paper.

Initial-State Assumption-Based Planning

In many settings it is convenient or sufficient to restrict assumptions to the initial state of the world, i.e., to make $h_i = \mathsf{true}$ for every $i > 0$. We call this class of problems *initial-state assumption-based planning*. An initial-state assumption-based plan is denoted by (h_0, α), where h_0 is a boolean formula over U that corresponds to an assumption with respect to the initial state.

The formal relation between conformant planning and initial-state assumption-based planning is straightforward, and is established in the following proposition.

Proposition 1 *The tuple (h_0, α) is an initial-state assumption-based plan for planning problem $P = (F, O, I, G, U)$ iff α is a conformant plan for $P' = (F, O, I \cup \{h_0\}, G)$.*

Note that this proposition *does not* imply that an assumption-based plan can be directly computed using a conformant planner, since a conformant planner is not able to compute assumptions. In addition, a relation between assumption-based planning and initial-state assumption-based planning can be established.

Theorem 1 *If $P = (F, O, I, G, U)$, $U = F$, and (ρ, α) is an assumption-based plan for P, then there exists an h_0 such that (h_0, α) is an initial-state assumption-based plan for P. Furthermore, h_0 can be computed from ρ, P and α in time $2^{\mathcal{O}(|\alpha|)}$.*

Proof sketch: Via regression, we obtain a condition ϕ_{prec} that corresponds to the conditions under which α is executable in the initial state. Likewise, we obtain a condition ϕ_G under which α leads to the goal G. h_0 is made equal to $\phi_{prec} \wedge \phi_G$. Regression is worst-case exponential in $|\alpha|$, but is linear in $|\alpha|$ if there are no actions with conditional effects in α. $\square$

As a consequence of this theorem, if α is a sequence of actions for which there is some ρ such that (ρ, α) is an assumption-based plan, then we can construct an assumption h_0 on the initial state using α such that (h_0, α) is an assumption-based plan. The proof of the above theorem (omitted here for space) actually gives a constructive algorithm for h_0 that relies on regressing G over α. Under certain conditions, one can similarly construct an arbitrary assumption-based plan (ρ, α) from an initial-state assumption-based plan (h_0, α) by *progressing* aspects of h_0 (Lin and Reiter 1997). Intuitively, this provides a means of generating an assumption-based plan that makes assumptions at the point at which they are needed, and no sooner. This supports monitoring of the continued validity of assumption-based plans during execution, dovetailing nicely with techniques for annotating plans with the necessary conditions for their continued validity (e.g., Fritz and McIlraith 2007), monitoring continued plan validity, and repairing or replanning when necessary.

As it turns out, the definition of assumption-based planning is general enough that its complexity seems to lie across a spectrum of complexity classes, depending on which literals are allowed to be assumed. Below we provide two complexity results showing that assumption-based planning is complete for two complexity classes. Our first result follows directly from the fact that conformant planning is EXPSPACE-complete (Haslum and Jonsson 1999).

Theorem 2 *Given an assumption-based planning problem $P = (F, O, I, G, U)$, where U contains no fluents mentioned in non-unary clauses of I, deciding whether or not an assumption-based plan exists is EXPSPACE-complete.*

However, as more information can be assumed, the complexity moves down to that of classical planning.

Theorem 3 *Given an assumption-based planning problem $P = (F, O, I, G, U)$, where U contains all fluents mentioned in non-unary clauses of I, deciding whether or not an assumption-based plan exists is PSPACE-complete.*

Proof sketch: For membership, we propose the following NPSPACE algorithm: guess the assumptions h_0 such that $I \cup h_0$ has a unique model, then call a PSPACE algorithm (like the one suggested by De Giacomo and Vardi (1999)) to decide (classical) plan existence. Then we use the fact that NPSPACE=PSPACE. Hardness is given by the fact that classical planning, a PSPACE-complete problem (Bylander 1994), can be straightforwardly reduced to assumption-based planning. $\square$

Theorem 3 implies that when the set of assumable fluents contain all fluents appearing in non-unary clauses of I, assumption-based planning can be reduced to classical planning. A *Naive Translation* is to construct a classical problem P' by augmenting P with an exponential (in the number of unknown fluents in the initial state, k) number of additional actions, selected at the outset of planning to complete the initial state, consistent with I. If $a_0 a_1 \ldots a_n$ is a classical plan for P', we construct the initial-state assumption-based plan as follows. h_0 is constructed with the facts true in the state s_1 that a_0 generates and α is simply set to $a_1 \ldots a_n$. Such a commitment to a single initial state can be excessive, making unnecessary assumptions. Alternatively, some domains may lend themselves to achieving the same completion effect by applying a sequence of actions. In our example, each sequence of these actions generates one of the possible 2^k states. Both approaches outlined above have been used in the past to tackle diagnosis problems in which the initial state is incomplete or unknown (Sohrabi, Baier, and McIlraith 2010; Haslum and Grastien 2011).

A Translation-Based Approach

Here we propose an alternative translation of assumption-based planning to classical planning that builds on top of Palacios and Geffner's $K_{T,M}$ translation (2009) – henceforth denoted by P&G– which translates conformant planning into classical planning. The main objective of our translation is to avoid the excessive commitment exhibited by the naive translation of assumption-based planning to classical planning.

The $K^A_{T,M}$ Translation

Given an assumption-based planning problem $P = (F, O, I, G, U)$, we generate a new planning problem $P' = (F', O', I', G')$; we call this process the $K^A_{T,M}$ translation, which builds on P&G. For each literal L we associate a set of *merges*, M_L. Each merge is a finite set of *tags*, which in turn are conjunctions of literals that are unknown in the initial state. Each merge characterizes a partition of the initial state in the sense that $I \models \bigvee_{t \in m} t$ is required to hold for each merge m.

A tag intuitively represents a partial completion of the initial state in which every $L \in t$ is initially true – it is a "case" in which L is initially true. Problem P' contains fluents of the form KL, for each $L \in F$, Kt and $K\neg t$ for each tag t, and KL/t for each $L \in F$ and each tag t in a merge of M_L. KL intuitively represents that L is *known*. KL/t represents the fact that L is known given that t is true in the initial state.

The main difference between P&G and our translation is that we consider a set of *assumption actions*, which allow the planner to assume that a tag t was true in the initial state. More specifically, given a set $\mathcal{T}$ of *assumable tags*, there is an assumption action associated to each $t \in \mathcal{T}$ that assumes t is true in the initial state. Instead of using the standard P&G merge actions, we use the contingent merge actions introduced by Albore, Palacios, and Geffner (2009). Furthermore, our translation augments P&G with additional conditional effects to handle assumptions. More precisely, P' is such that the following holds.

(1) $I' = \{KL \mid I \models L\} \cup \{KL/t \mid I, t \models L\} \cup \{Kt, K\neg t \mid I, t \models L, \text{for some } L\} \cup \{ok\}$. I' differs from P&G in the ok fluent which is added to keep track of consistency and is explained in detail later, and in the fluents of the form Kt and $K\neg t$, with t a tag, which represent that t is true and, respectively, false in the initial state.

(2) For each literal L, and each tag t in some merge of M_L, O contains the so-called *contingent merges* proposed originally by Albore, Palacios, and Geffner (2009), of the form $[\bigwedge_{t \in m}(KL/t \vee K\neg t)] \to KL$. These generalize the P&G merge actions for the case where t is refuted by assumptions.

(3) Like in P&G, for each action a with conditional effect $C \rightarrow L$, O' contains the conditional effects $[\bigwedge_{c \in C} Kc/t] \rightarrow KL/t$ and $[\bigwedge_{c \in C} \neg K \neg c/t] \rightarrow \neg K \neg L/t$, for each tag t in some merge of M_L.

(4) In addition, for each tag t in the set of assumable tags, $\mathcal{T}$, we create an assumption action $Assume(t)$, with precondition $\neg K \neg t \wedge \neg Kt \wedge \neg K \neg t' \wedge \neg Kt'$ and effects Kt, $\neg K \neg t$, $K \neg t'$, $\neg Kt'$ for every tag t' that is inconsistent with t, i.e., contains the complement of a literal in t.

(5) For each merge set M_L that contains tag t, and each merge $m \in M_L$, the conditional effects $KL/t \rightarrow KL$, and $KL/t \wedge K \neg L \rightarrow \neg ok$ are added to the $Assume(t)$ action. The first conditional effect makes L known if it is the case that KL/t. The second conditional effect takes care of potential inconsistencies that could arise when assuming a literal that implies that L is known, when $\neg L$ is already known. In such cases the action deletes the fluent ok signaling inconsistency.

(6) For each action $a \in O$ the version of a in O' contains the precondition $ok \wedge \bigwedge_{L \in prec(a)} KL$.

(7) The planner should not make inconsistent assumptions. Thus whenever we assume a tag t, we may need to update the knowledge about other tags. To illustrate this, consider that L_1, L_2, and L_3 are literals, each of which corresponds to a tag, and that both $c_1 = \neg L_1 \vee L_2$ and $c_2 = \neg L_3 \vee L_1$ are clauses in I, and suppose a plan contains the action $Assume(L_3)$. Then action $Assume(\neg L_2)$ cannot consistently occur after $Assume(L_3)$ because that would imply that c_1 would be contradicted (L_1 is forced to be true by c_2 and the assumption of L_3). Thus whenever we assume a tag t, we may need to update the knowledge about other tags. We achieve this by adding specific conditional effects to assumption actions. Such effects reflect logical inferences among clauses defining the initial state and we obtain them by performing resolution. In our example, if one carries out a resolution step between c_1 and c_2, then we obtain the clause $c_3 = \neg L_3 \vee L_2$, from which it is straightforward that L_2 is forced to be true after assuming L_3. Using c_3 we write a new effect for $Assume(\neg L_3)$ that states KL_2 as a new effect. This idea can be extended further by computing *all* possible resolution steps with the clauses in the initial state.

In the general case, however, tags may be conjunctions of literals, and thus the relationship between different tags may not be entirely obvious by just looking at the clauses that result from resolution. In such a case, for each tag t, we add to I the clauses corresponding to the formula $t \leftrightarrow [\bigwedge_{L \in t} L]$, where t is a new variable that represents a tag t. After carrying out all possible resolutions there will be clauses that only contain variables of the form t, and we only consider these clauses to generate the effects.

Let us denote by I^+ the set of clauses that result after performing resolution. Now we are ready to specify the conditional effects that are going to be added to $Assume(t)$. For each clause in $c \in I^+$ of the form $\{\neg\tau_0, \tau_1, \tau_2, \ldots, \tau_n\}$, with each τ_i of the form t or $\neg t$, for some tag t, we add the conditional effect $[\bigwedge_{\tau_r \in c \setminus \{\tau_0, \tau_i\}} K \neg \tau_r] \rightarrow K\tau_i$, for every $i \in \{1, \ldots, n\}$, to action $Assume(\tau_0)$. In addition, effects similar to those described in step (5) are also added to account for the addition of each $K\tau_i$.

Note that the construction of I^+ is clearly worst-case exponential. Nevertheless, in practice, there are usually few resolution steps that can be made between clauses in I as usually I is formed by groups of clauses relatively independent of each other. Furthermore, when tags are of size 1, we do not need to add the clauses involving additional variables t as literals themselves represent their tags.

(8) Finally, $G' = \{KL \mid L \in G\} \cup \{ok\}$.

Just like P&G, our $K^A_{T,M}$ translation is *sound* in the following sense.

Theorem 4 *The $K^A_{T,M}$ translation is sound; i.e., if α is a plan for $K^A_{T,M}(P)$, then there is an assumption-based plan (ρ, α') for the original problem. Furthermore, (ρ, α') can be computed from α in linear time.*

The $K^A_i(P)$ Translation

As with P&G's $K_{T,M}$ translation, the $K^A_{T,M}$ translation does not explicitly define how the merges/tags are computed from the original problem. In addition, it provides no completeness guarantees. A practical realization of P&G's $K_{T,M}$ is given by the so-called K_i translation (Palacios and Geffner 2009). K_i defines an explicit way to compute merges. It is a sound translation (in the sense defined above). In addition, if i is not greater than the so-called *width* of the problem P, then it is also complete.

We have defined an analogous version of the K_i translation, that we call K^A_i. K^A_i is a version of K_i in which merges and tags are computed using the same procedure as for the case of K_i. Due to lack of space we cannot elaborate on this process, but we refer the reader to Palacios and Geffner's paper (2009) for reference. After the tags and merges are determined, however, it may be that the set of tags does not capture the set of assumable fluents. In such a case, we create additional tags for those assumable fluents that are not captured. Since K^A_i is a particular form of the translation $K_{T,M}$, we obtain that it is *sound* as a corollary of Theorem 4. Furthermore,

Theorem 5 *Given an assumption-based planning problem $P = (F, O, I, G, U)$, with width $w(P) \leq i$, the K^A_i translation is complete; i.e. if there exists an assumption-based plan (ρ, α) for P, in which ρ are conjunctions of literals in U, then a plan exists for $K^A_i(P)$.*

In the previous result, $w(P)$ is defined analogously to P&G.

Negative Results Given P, $K_i(P)$ is polynomial in the width of P (Palacios and Geffner 2009). Since our implementation involves a step in which previously we do a resolution fixpoint computation (Step (7)), we cannot guarantee that the K^A_i translation is polynomial on the width of P.

Preferred Assumption-Based Planning

The definition of an assumption-based plan allows the planner to assume any aspect of the state that can be constructed from the subset of assumable literals and consistently assumed. However, some assumptions will be more reasonable than others. E.g., in our subway example, if it's raining, it is much more reasonable to assume

$subwayRunning$ than $dryRoad$. To define the notion of a preferred assumption-based plan, we employ a preference relation $\preceq$, a transitive and reflexive relation in $\Pi \times \Pi$, where Π is the set of all assumption-based plans for a particular problem (following Baier and McIlraith 2008). Plan optimality is defined in the obvious way given relation $\preceq$.

For the purposes of this paper, we will appeal to the uniform notion of action cost in order to characterize preferred assumption-based plans, rather than defining $\preceq$ directly. Specifically, given an assumption-based planning problem P, we build its translated instance $K^A_{T,M}(P) = (F, O, I, G)$, and then augment this instance to produce a *cost-based planning problem* $P_{\mathcal{C}} = (F, O, I, G)$ such that each action $a \in O$ has a non-negative cost $\mathcal{C}(a)$. Note that this means that actions of the form $Assume(t)$, as well as other domain actions, will have a cost associated with them. When a particular assumption is reasonable, as is the case with assuming that the subway is running, the cost of the corresponding $Assume(t)$ will be low relative to domain action costs. In cases where the assumptions are not reasonable, the cost of their corresponding actions will be high.

Specifying how a domain expert would specify these preferences in the original problem specification and ensuring that the corresponding cost-based planning problem respects the induced $\prec$ relation can be achieved in a variety of ways. Detailed discussion of this is beyond the scope of this paper.

Implementation and Experiments

The K^A_1 translation was implemented in our A0 planner as an augmentation of Palacios and Geffner's T0 planner. In the absence of the specification of assumables, the assumables are set to all the fluents less those involved in G, precluding assumption-based plans that assume G. We use FF (Hoffmann and Nebel 2001) to generate classical plans with the translated domains and convert them back into assumption-based plans. To generate preferred assumption-based plans, we associate a cost with each action in the (translated) classical planning problem. The resulting cost-based planning problem is solved using the latest versions of Metric-FF (Hoffmann 2003) and LAMA (Richter, Helmert, and Westphal 2008).

Since the notion of assumption-based planning is new, there are no systems to benchmark against. We sought instead to evaluate the running time of A0 + FF compared to an implementation of the previously introduced *naive* assumption-based planning, and to various cost distributions for cost-based assumption-based planning. We also assessed the proportion of solution time taken by translation.

Domains: We exploited four domains from the International Planning Competition (IPC) benchmark suite: *logistics*, *raokeys*, *coins*, and *storage*. The well-known logistics domain was converted into an assumption-based planning problem by making some routes uni-directional (corresponding to one-way streets, temporary closures, etc.) or closed completely. Trucks were provided with 4 levels of gas, which are decremented by one with each movement. Certain locations are designated as having gas stations, permitting refueling. We refer to this modified domain as *alogistics*. The 12 instances we constructed vary in the number of cities and trucks (2-4), and locations within a city. Varying amounts of uncertainty were introduced into the initial state of each instance via unknown truck gas levels and locations, and the connectivity within cities. The second domain we used was *raokeys*, a conformant planning IPC-2008 benchmark. The problem requires reasoning about n locks with n different possible keys in n different possible locations, making the number of initial states combinatorial explosive. We experimented with 4 instances of *raokeys*. These problem instances were left unchanged to see how the planner would address standard conformant planning problems. The *storage* classical IPC-2006 planning benchmark which requires storing different crates into depots using hoists, was converted into an assumption-based planning domain by adding uncertainty about whether or not a depot was available. Crates could only be stored in available depots. All 12 instances we generated have four depots, and the number of crates is varied. Because one of the legal initial states is such that no depot is available, none of these instances have a conformant plan. Finally, the *coins* domain, a conformant planning benchmark, in which the objective is to collect coins in different floors, was converted into an assumption-based planning domain by letting the planner assume the location of the coins, and the floor at which an elevator is initially located.

Experiments: We ran 7 different experimental configurations on the 16 problem instances described above and 4 preliminary results from *coins*. **(1)** We ran A0 + FF on the translated domains. **(2)** We generated a naive assumption-based planning problem by augmenting each problem instance with actions that create each of the different consistent completions of the initial state, then solved with FF. **(3)–(7)** These configurations all relate to generating preferred assumption-based plans. The configurations differ with respect to the cost of the assumption actions relative to the domain actions. E.g., $x = 0.5$ denotes that assumption actions are twice as expensive as domain actions. All merge actions were assigned equal (low) cost. Instances solved via A0 + Metric-FF or LAMA.

Table 1 shows the results obtained on the seven configurations for a representative subset of all instances. On the classical settings, A0 does not take much more time than the naive method but makes fewer assumptions in general. In *alogistics*, A0 and the naive approach are comparable time-wise, but A0 makes fewer assumptions (see e.g., alog-4). This domain is contrasted by *raokeys* in which A0 seems to take exponentially more time while the naive method works fairly well. The reason the naive method is able to handle the small instances (raokeys2, raokeys3, and raokeys4) is because the conditional dependencies are not considered, it merely needs to set the unknown fluents and find a plan from there. In the easier assumption-based planning *coins* domain, we observe a behavior similar to alogistics. For larger instances FF failed due to a too large number of predicates (even after recompiling it to accept 10 times more predicates than the default). This does not seem to be a limitation of the planner, however; we believe that by increasing even more its hard-coded parameters, more instances should be

Table 1: A comparison of seven planner configurations. The total time to solve in seconds is on the left and plan length on the right. The number of assumptions made appears in parentheses. (STO: time out during planning. PRED: FF error, more than 5,000 predicates needed.). All experiments were run on a 2.80GHz machine with 2GB memory and a 30 minute timeout.

Prob	**Classical** (t(s)/len)		**A0 + Cost-Based Metric-FF** (t(s)/len)					**A0 + Cost-Based LAMA** (t(s)/len)				
	A0 + FF	naive + FF	x=0.1	x=0.5	x=1	x=2	x=10	x=0.1	x=0.5	x=1	x=2	x=10
alog-01	0.05/42 (2)	0.00/38 (4)	0.01/34 (2)	0.01/34 (2)	0.01/34 (2)	0.01/34 (2)	0.01/34 (2)	214.59/37 (1)	255.05/37 (1)	445.31/35 (2)	0.02/36 (3)	0.01/36 (3)
alog-02	0.07/42 (2)	0.01/50 (16)	0.02/34 (2)	0.02/34 (2)	0.01/34 (2)	0.02/34 (2)	0.01/34 (2)	1.27/38 (2)	418.33/35 (2)	0.02/36 (3)	95.17/34 (2)	128.16/34 (2)
alog-04	0.2/42 (2)	0.43/128 (92)	0.02/34 (2)	0.02/34 (2)	0.02/34 (2)	0.01/34 (2)	0.02/34 (2)	0.32/38 (2)	0.01/36 (3)	0.03/36 (3)	0.03/36 (3)	0.02/36 (3)
alog-06	0.07/48 (2)	0.01/49 (8)	0.02/40 (3)	0.02/40 (3)	0.02/40 (3)	0.02/40 (3)	0.02/40 (3)	6.08/41 (1)	31.99/43 (1)	1.45/41 (4)	1.46/41 (4)	1.46/41 (4)
alog-10	0.07/34 (5)	0.01/35 (6)	0.03/37 (3)	0.03/37 (3)	0.03/37 (3)	0.03/37 (3)	0.03/37 (3)	8.11/32 (1)	22/32 (1)	0.02/33 (4)	601.62/30 (6)	149/29 (4)
coins-01	0.01/5 (3)	0.01/9 (4)	0.00/6 (3)	0.01/6 (3)	0.00/6 (3)	0.01/6 (3)	0.00/6 (3)	0.03/12 (0)	0/5 (3)	0.01/5 (3)	0/5 (3)	0.01/5 (3)
coins-07	0.05/8 (5)	0.01/14 (6)	0.02/8 (5)	0.02/8 (5)	0.03/8 (5)	0.02/8 (5)	0.03/8 (5)	4.91/31 (0)	0.08/8 (5)	0.08/8 (5)	0.06/8 (5)	0.04/8 (5)
coins-13	0.7/10 (7)	0.10/18 (8)	0.51/10 (7)	0.51/10 (7)	0.54/10 (7)	0.52/10 (7)	0.55/10 (7)	117.71/12 (6)	4.33/10 (7)	4.29/10 (7)	3.83/10 (7)	3.43/10 (7)
coins-19	0.79/15 (7)	0.12/25 (8)	0.56/15 (7)	0.57/15 (7)	0.57/15 (7)	0.57/15 (7)	0.57/15 (7)	311.82/18 (6)	248.39/14 (7)	214.96/14 (7)	82.83/14 (7)	3.69/15 (7)
coins-25	PRED	24.32/62 (20)	PRED	PRED	PRED	PRED	PRED	131.86/45 (16)	132.06/45 (16)	129.12/45 (16)	143.87/40 (16)	156.69/40 (16)
coins-30	PRED	67.50/71 (25)	PRED	PRED	PRED	PRED	PRED	399.1/52 (21)	398.42/52 (21)	398.32/52 (21)	392.55/52 (21)	399.85/52 (21)
raokeys-2	0.04/8 (2)	0.01/10 (9)	0.02/9 (2)	0.02/9 (2)	0.02/9 (2)	0.02/9 (2)	0.03/9 (2)	0.02/8 (2)	0.01/8 (2)	0.02/8 (2)	0.02/8 (2)	0.02/8 (2)
raokeys-3	36.68/13 (3)	0.04/23 (13)	8.24/15 (4)	8.57/15 (4)	8.50/15 (4)	8.16/15 (4)	8.22/15 (4)	57.37/14 (3)	57.29/14 (3)	57.52/14 (3)	57.35/14 (3)	57.44/14 (3)
raokeys-4	NOTRAN	5.14/39 (17)	NOTRAN	NOTRAN	NOTRAN	NOTRAN	NOTRAN	NOTRAN	NOTRAN	NOTRAN	NOTRAN	NOTRAN
raokeys-5	NOTRAN	STO	NOTRAN	NOTRAN	NOTRAN	NOTRAN	NOTRAN	NOTRAN	NOTRAN	NOTRAN	NOTRAN	NOTRAN
storage-01	0.44/7 (2)	0.02/11 (4)	0.02/7 (2)	0.01/7 (2)	0.01/7 (2)	0.01/7 (2)	0.02/7 (2)	0.02/9 (1)	0.07/7 (2)	0.02/7 (2)	0.03/7 (2)	0.07/7 (2)
storage-04	1.80/33 (4)	125.47/49 (4)	0.14/35 (4)	0.15/35 (4)	0.14/35 (4)	0.14/35 (4)	0.18/35 (4)	3.93/29 (4)	4.71/29 (4)	4.56/29 (4)	4.84/29 (4)	5.15/29 (4)
storage-07	1.06/9 (3)	0.06/12 (3)	0.02/9 (3)	0.02/9 (3)	0.02/9 (3)	0.02/9 (3)	0.02/9 (3)	0.04/13 (1)	0.08/9 (3)	0.08/9 (3)	0.08/9 (3)	0.08/9 (3)
storage-09	1.20/35 (3)	100.53/48 (3)	0.12/39 (3)	0.13/39 (3)	0.14/39 (3)	0.11/39 (3)	0.11/39 (3)	1.64/33 (3)	0.98/33 (3)	0.88/33 (3)	0.76/33 (3)	0.67/33 (3)
storage-10	3.44/59 (3)	STO	6.51/85 (3)	6.50/85 (3)	7.12/83 (3)	7.16/83 (3)	7.04/83 (3)	6.17/49 (3)	5.92/49 (3)	5.15/49 (3)	5.75/49 (3)	5.83/49 (3)
storage-11	STO	STO	STO	STO	STO	STO	STO	705.62/167 (4)	637.31/165 (4)	635.51/165 (4)	632.32/157 (4)	621.2/157 (4)
storage-12	STO	STO	STO	STO	STO	STO	STO	STO	STO	STO	STO	STO

solved. It is interesting to note that when no assumptions can be made, the very same instances we evaluated here are very challenging (To, Son, and Pontelli 2010). In the *storage* domain, we observe that our translation approach can significantly outperform the naive approach. In this domain, the standard delete relaxation is a weak heuristic, thus the planner may choose the wrong set of assumptions initially, leading the search astray.

Finally, we observe that the number of assumptions decreases as they are penalized further, at the expense of increased solving time. In the instances we tried it was not possible to have a large difference in the number of assumptions because on the one hand a few assumptions were *required* for each instance, and on the other hand only a few assumptions were possible to make in one plan without inconsistencies. With further testing on problem instances that either allow or require many more assumptions we believe the trend will be more apparent and interesting.

We also evaluated the proportion of solution time dedicated to translation. For the *alogistics* domains this ranged from 5–25%, whereas with the *raokeys* domain, it was closer to 50%. In the *coins* and *storage* domains, on the other hand, translation time is negligible compared to solving time.

Summary and Concluding Remarks

Our concern in this paper is with real-world planning scenarios where incomplete information is prevalent, and where a plan must be generated prior to execution and/or without the benefit of run-time sensing. Motivated by so-called commonsense planning, and by the paradigm of planning that is at the computational core of explanation, diagnosis, or counter-example generation, we introduce the notion of assumption-based planning. We provide a formal characterization of assumption-based planning, establishing a correspondence to conformant planning. Exploiting this correspondence, we provide a translation of an assumption-based planning problem to a classical planning problem, building on the popular translation developed by P&G. We prove the soundness and completeness of our translation. This provides us with a means of generating assumption-based plans using classical planners. We also argue for the merit of preferred assumption-based plans and propose a means of realizing such plans using cost-based planning. We describe A0, a planner that addresses the subset of initial state assumption-based planning problems and present experiments that illustrate the viability of our approach and that assess some properties of our translation.

While this paper explores the generation of assumption-based plans via a translation to classical planning, the correspondence to conformant planning opens the door to adapting a variety of conformant planners for assumption-based planning (e.g., To, Son, and Pontelli 2010). Beyond planning, the assumption-based planning paradigm has compelling applications in diagnosis, explanation generation, and verification of dynamical systems.

Acknowledgements: We wish to thank Alexandre Albore and Héctor Palacios for assistance with the CLG and TO planner code bases. We also gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC). Jorge Baier was partly funded by Fondecyt grant no. 11110321. Earlier versions of this work appeared at ICAPS and AAAI workshops (Davis-Mendelow, Baier, and McIlraith 2012a; 2012b).

References

Albarghouthi, A.; Baier, J. A.; and McIlraith, S. A. 2009. On the use of planning technology for verification. In *Proc. of ICAPS Workshop on Verification and Validation of Planning and Scheduling Systems*.

Albore, A., and Bertoli, P. 2004. Generating safe assumption-based plans for partially observable, nondeterministic domains. In *Proc. of the 19th National Conference on Artificial Intelligence (AAAI)*, 495–500.

Albore, A., and Geffner, H. 2009. Acting in partially observable environments when achievement of the goal cannot be guaranteed. In *Proc. of ICAPS Workshop on Planning and Plan Execution for Real-World Systems*.

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In *Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 1623–1628.

Baier, J. A., and McIlraith, S. A. 2008. Planning with preferences. *Artificial Intelligence Magazine* 29(4):25–36.

Bonet, B., and Geffner, H. 2011. Planning under partial observability by classical replanning: Theory and experiments. In *Proc. of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 1936–1941.

Brafman, R. I., and Shani, G. 2012. Replanning in domains with partial information and sensing actions. *Journal of Artificial Intelligence Research* 45:565–600.

Brafman, R. I., and Taig, R. 2011. A translation based approach to probabilistic conformant planning. In *Proceedings of the Second International Conference on Algorithmic Decision Theory (ADT)*, 16–27.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1-2):165–204.

Davis-Mendelow, S.; Baier, J. A.; and McIlraith, S. A. 2012a. Making reasonable assumptions to plan with incomplete information. In *Proc. of the ICAPS Workshop on Heuristics and Search for Domain-Independent Planning (HSDIP)*.

Davis-Mendelow, S.; Baier, J. A.; and McIlraith, S. A. 2012b. Making reasonable assumptions to plan with incomplete information: Abridged report. In *Proc. of the AAAI Workshop on Problem Solving Using Classical Planners (CP4PS)*.

De Giacomo, G., and Vardi, M. Y. 1999. Automata-theoretic approach to planning for temporally extended goals. In Biundo, S., and Fox, M., eds., *ECP*, volume 1809 of *LNCS*, 226–238. Durham, UK: Springer.

Faletti, J. 1982. Pandora: A program for doing commonsense planning in complex situations. In *AAAI82*, 185–188.

Fritz, C., and McIlraith, S. A. 2007. Monitoring plan optimality during execution. In *Proc. of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*, 144–151.

Göbelbecker, M.; Keller, T.; Eyerich, P.; Brenner, M.; and Nebel, B. 2010. Coming up with good excuses: What to do when no plan can be found. In *Proc. of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*, 81–88.

Göbelbecker, M.; Gretton, C.; and Dearden, R. 2011. A switching planner for combined task and observation planning. In *Proc. of the 26th AAAI Conference on Artificial Intelligence (AAAI)*.

Haslum, P., and Grastien, A. 2011. Diagnosis as planning: Two case studies. In *Proc. of the International Scheduling and Planning Applications workshop (SPARK)*.

Haslum, P., and Jonsson, P. 1999. Some results on the complexity of planning with incomplete information. In *Proc. of the 5th European Conference on Planning (ECP)*, 308–318.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Hoffmann, J. 2003. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research* 20:291–341.

Lin, F., and Reiter, R. 1997. How to progress a database. *Artif. Intell.* 92(1-2):131–167.

Palacios, H., and Geffner, H. 2006. Compiling uncertainty away: Solving conformant planning problems using a classical planner (sometimes). In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI)*.

Palacios, H., and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research* 35:623–675.

Pellier, D., and Fiorino, H. 2004. Assumption-based planning. In *In Proceedings of the International Conference on Advances in Intelligence Systems Theory and Applications, Luxemburg*.

Poole, D.; Goebel, R.; and Aleliunas, R. 1987. Theorist: a logical reasoning system for defaults and diagnosis. In Cercone, N., and McCalla, G., eds., *The Knowledge Frontier: Essays in the Representation of Knowledge*. New York: Springer Verlag. 331–352.

Ramírez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proc. of the 25th AAAI Conference on Artificial Intelligence (AAAI)*.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *Proc. of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, 975–982.

Sohrabi, S.; Baier, J.; and McIlraith, S. A. 2010. Diagnosis as planning revisited. In *Proc. of the 12th International Conference on Knowledge Representation and Reasoning (KR)*, 26–36.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2011. Preferred explanations: Theory and generation via planning. In *Proc. of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, 261–267.

To, S. T.; Son, T. C.; and Pontelli, E. 2010. A New Approach to Conformant Planning Using CNF*. In *Proc. of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*, 169–176.