Non-classical Planning with a Classical Planner: The Power of Transformations

Hector Geffner

ICREA & Universitat Pompeu Fabra
Roc Boronat 138, 08018 Barcelona, Spain
hector.geffner@upf.edu
http://www.dtic.upf.edu/~hgeffner

Abstract. Planning is the model-based approach to autonomous behavior where a predictive model of actions and sensors is used to generate the behavior for achieving given goals. The main challenges in planning are computational as all models, whether featuring uncertainty and feedback or not, are intractable in the worst case when represented in compact form. Classical planning refers to the simplest form of planning where goals are to be achieved by applying deterministic actions to a fully known initial situation. In this invited paper, I review the inferences performed by classical planners that enable them to deal with large problems, and the transformations that have been developed for using these planners to deal with non-classical features such as soft goals, hidden goals to be recognized, planning with incomplete information and sensing, and multiagent nested beliefs.

1 Introduction

At the center of the problem of intelligent behavior is the problem of selecting the action to do next. In AI, three different approaches have been used to address this problem. In the *programming-based approach*, the controller that prescribes the action to do next is given by a programmer, usually in a suitable high-level language. In this approach, the problem is solved by the programmer in his head, and the solution is expressed as a high-level program in behavior-based languages, hierarchical task-networks, rules, or languages such as Golog [1,2]. In the *learning-based approach*, the controller is not given by a programmer but is induced from experience: the agent's own experience, in reinforcement learning, or the experience of a 'teacher' in supervised learning schemes [3]. Finally, in the *model-based approach*, the controller is not learned but is derived automatically from a model of the actions, sensors, and goals.

Planning is the model-based approach to action selection where different types of models are used to make precise the different types of agents, environments, and controllers [4,5]. Classical planning is the simplest form of planning, concerned with the achievement of goals in deterministic environments whose initial state is fully known. POMDP planning, on the other hand, allows for stochastic actions in partially observable environments. The main challenges in planning

E. Fermé and J. Leite (Eds.): JELIA 2014, LNAI 8761, pp. 33–47, 2014.

are computational, as all the models, whether accommodating feedback and uncertainty or not, are intractable in the worst case when models are represented in compact form.

In this paper, I review the inferences performed by classical planners that enable them to deal with large problems, and the transformations that have been developed for using these planners to deal with non-classical features such as soft goals, hidden goals to be recognized, planning with incomplete information and sensing, and multiagent nested beliefs.

2 Planning Models

A wide range of models used in planning can be understood as variations of a *basic state model* featuring

- a finite and discrete state space S,
- a *known initial state* $s_0 \in S$,
- a set $S_G \subseteq S$ of goal states,
- a set $A(s) \subseteq A$ of actions applicable in each state $s \in S$,
- a *deterministic* state transition function $f(a, s)$, $a \in A(s)$, and
- positive *action costs* $c(a, s)$.

This is the model underlying *classical planning* where it also normally assumed that action costs $c(a, s)$ do not depend on the state, and hence $c(a, s) = c(a)$. A solution or *plan* in this model is a sequence of applicable actions that map the initial state into a goal state. More precisely, a plan $\pi = a_0, \ldots, a_{n-1}$ must generate a state sequence $s_0, \ldots, s_n$ such that $a_i \in A(s_i)$, $s_{i+1} = f(a_i, s_i)$, and $s_n \in S_G$, for $i = 0, \ldots, n-1$. The cost of the plan is the sum of the action costs $c(a_i, s_i)$, and a plan is optimal if it has minimum cost over all plans.

A classical plan $a_0, \ldots, a_n$ represents an *open-loop controller* where the action to be done at time step i depends on the step index i. The solution of models that accommodate uncertainty and feedback, on the other hand, produce *closed-loop controllers* where the action to be done at step i depends on the actions and observations collected up to that point. These models can be obtained by relaxing some of the assumptions in the classical model.

In the model for *conformant planning*, the initial state s_0 is not known and it is replaced by a *set* S_0 of possible initial states. Likewise, in non-deterministic conformant planning, the state transition function $f(a, s)$ is replaced by a non-deterministic transition function $F(a, s)$ that denotes the set of states that are possible after doing an action a in the state s. A *conformant plan* is an action sequence that must achieve the goal for any possible initial state and state transition.

In *contingent planning* or *partially observable planning*, a sensor model $O(a, s)$ is assumed that maps the current state s and last action a into a set of possible observation tokens that provide partial information about the true but possibly hidden state s. Contingent plans can be expressed in many forms, for example, as a function (policy) mapping *beliefs* into actions, where a belief represents a

set of states that are deemed as possible. The initial belief state is given by the set of states S_0 that are initially possible, and the successor belief states can be obtained from the actions performed and the observations gathered, using the transition and sensor functions $F(\cdot, \cdot)$ and $O(\cdot, \cdot)$.

Partial Observable Markov Decision Processes (POMDPS) are contingent planning models where uncertainty about the initial situation, the next system state, and the possible token to be observed, are not represented by *sets* but by probability distributions. Beliefs in POMDPs are thus not sets of states but probability distributions over states. Markov Decision Processes are POMDPs where the states are fully observable.

Classical, conformant, contingent, MDP, and POMDP planners accept a compact description of the corresponding models and produce the solutions (controllers) automatically. On-line planners, on the other hand, produce the action to be done next in the current situation. The basic language for modeling classical planning problems is STRIPS, where a problem is a tuple $P = \langle F, I, O, G \rangle$ in which F is a set of atoms, $I \subseteq F$ and $G \subseteq F$ represent the initial and goal situations, and O is a set of actions a with preconditions, add, and delete effects, all part of F. The PDDL language provides a standard syntax for STRIPS and a number of extensions. Similar languages are used to describe the other planning models in compact form. In all cases, a problem involves a number of variables, boolean or not, and the states correspond to the possible valuations of such variables.

3 Classical Planning

A classical planning problem P can be mapped into a *path-finding problem* over a graph $\mathcal{S}(P)$ where the nodes are the states, the initial node and target nodes are the initial and goal states respectively, and a directed edge between two nodes denotes the existence of an action that maps one state into the other. Classical planning problems can thus be solved in theory by path-finding algorithms such as Dijkstra's, but not in practice, as the size of the graph is exponential in the number of problem variables. Current classical planners such as LAMA [6] thus appeal to *three ideas* for scaling up: automatically derived *heuristic functions* for guiding the search [7,8], the inference of implicit goals in the problems called *landmarks* [9], and a structural criterion for distinguishing the applicable actions that are more likely to be relevant called the *helpful actions* [10].

Heuristic functions have been used in AI since the 60s for making graph search goal-directed. An heuristic function $h(s)$ in planning provides a quick but approximate estimate of the cost of solving the problem from the state s. The new development in planning in the 90s was a way for deriving informed heuristic values effectively from STRIPS encodings. Basically, if $P(s)$ is the classical planning problem with initial state s, and $P^+(s)$ is the *delete-free relaxation* of $P(s)$; i.e., the STRIPS problem that results from dropping the "delete lists", the heuristic $h(s)$ is set to the cost of a *relaxed plan* for $P(s)$; namely a *plan for the relaxation* $P^+(s)$ [10]. While computing an optimal plan for the delete-free problem $P^+(s)$

remains NP-hard, computing one possibly non-optimal plan for $P^+(s)$ is easy. This is because delete-free problems are *fully decomposable*, and hence, a plan π that achieves p from s can be appended to a plan π' that achieves p' from s to yield a plan that achieves both p' and p.

As a result, a simple polynomial iterative procedure can be used to compute relaxed plans for achieving each of the atoms in the problem. Basically, an atom p is *reachable* in 0 steps with relaxed plan $\pi(p, s) = \{\}$ if $p \in s$ (p is true in s), while an atom p is *reachable* in $i + 1$ steps with relaxed plan $\pi(p_1, s), \ldots, \pi(p_n, s), a_p$ if p not reachable in i steps or less from s, and there is an action a_p that adds p with preconditions $p_1, \ldots, p_n$ reachable from s in no more than i steps.

It's simple to prove that the procedure terminates in a number of steps bounded by number of problem atoms (when there are no new reachable atoms), and that if an atom p is reachable, $\pi(p, s)$ is a relaxed plan for p from s; i.e. a plan for p in the relaxation $P^+(s)$. Also if an atom p is not reachable from s, there is no plan for p in the original problem $P(s)$. The heuristic $h_{\mathrm{FF}}(s)$ used in the FF and LAMA planners is related to the number of *different* actions in the relaxed plans $\pi(G_i, s)$ for the problem goals G_i. The actions applicable in a state s that are regarded as *helpful* are the actions that are relevant to these relaxed plans; namely, those that add the precondition of an action in $\pi(G_i, s)$ or a goal G_i that is not true in s. Similarly, the *landmarks* in $P(s)$ are identified with the landmarks of the relaxation $P^+(s)$ which can be computed in low polynomial time; indeed, p is a landmark in $P^+(s)$ and hence in $P(s)$, iff the relaxed problem $P^+(s)$ has no plans once the actions that add the atom p are excluded.

State-of-the-art classical planners make use of these three notions, *heuristics*, *landmarks*, and *helpful actions* in different ways. For example, LAMA is a best-first search planner that uses *four queues* [11]: two of these queues are ordered by the h_{FF} heuristic and two are ordered by the number of unachieved landmarks. One queue for each heuristic is restricted to contain the children that result from the application of helpful actions, and the best first search alternates among the four queues. In this way, these planners tend to be robust and do not break down due to the number of atoms or actions in the problem. In the last few years, the SAT approach to classical planning [12], as pushed recently by Rintanen [13], has closed the performance gap with heuristic search planners quite considerably too.

4 Beyond Classical Planning

Classical planners work reasonably well by now, meaning that they can accept problems involving hundreds, and even thousand of actions and variables, often producing plans very quickly.[1] The sheer size of a problem is not an impediment in itself for solving it. The model underlying classical planning is simple but useful. Actions in planning can be activities or policies of any sort that can be characterized deterministically in terms of pre and postconditions. While non-deterministic effects are not accommodated, they can be handled sometimes in

[1] My focus is on satisficing planning, not optimal planning. Satisficing planners search for solutions that are good but not necessarily optimal.

a simple manner too. Some of the best planners in the MDP competitions held so far, for example, are not MDP solvers, but classical planners that *choose* one of the possible outcomes and replan from the current state when the system is observed off its expected trajectory [14].

For dealing with non-classical planning models in a more general way, two types of approaches have been pursued: a *top-down* approach, where *native solvers* are developed for more expressive models, and a *bottom-up* approach, where the power of classical planners is exploited by means of suitable *translations* [15]. MDP and POMDP planners are examples of native solvers for more expressive models. A limitation of these planners in comparison with classical planners is that *inference* is usually performed at the level of *states* and *belief states*, rather than at the level of *variables*. *Translation-based* approaches, on the other hand, leverage on classical planners for solving non-classical planning problems by introducing suitable transformations.

5 Translations and Transformations

Transformations have been developed for dealing with *soft goals*, *goal recognition*, *incomplete information and sensing*, and *multiagent nested beliefs*. The ideas underlying these transformations are reviewed below. Other features addressed in recent years using classical planners and transformations include *temporally extended goals* [16,17,18,19], *probabilistic conformant planning* [20], and *off-line contingent planning* [21,22].

5.1 Soft Goals and Rewards

Soft goals are used to express desirable outcomes that unlike standard hard goals are subject to a cost-utility tradeoff [23]. We consider STRIPS problems extended with positive *action costs* $c(a)$ for each action a, and non-negative *rewards* or *utilities* $u(p)$ for every atom p. The soft-goals of the problem are the atoms with positive utility. In the presence of soft goals, the target plans π are the ones that maximize the utility measure $u(\pi)$ given by the difference between the total utility obtained by the plan and its cost; i.e., $u(\pi) = \sum_{p:\pi \models p} u(p) - c(\pi)$ where $c(\pi)$ is the sum of the action costs in π, and the utility sum ranges over the soft goals p that are true at the end of the plan.

A plan π for a problem with soft goals is optimal when no other plan π' has utility $u(\pi')$ higher than $u(\pi)$. The International Planning Competition held in 2008 featured a *net-benefit optimal* track where the objective was to find $u(\pi)$ optimal plans [24]. Soft goal or net-benefit planning appears to be very different than classical planning as it involves two interrelated problems: deciding which soft goals to pursue and deciding how to achieve them. Indeed, most of the entries in the competition developed native planners for solving these two interrelated problems. More recently, it has been shown that problems P with soft goals can be compiled into *equivalent problems* P' *without soft goals* that can be solved by classical planners able to handle action costs only [25].

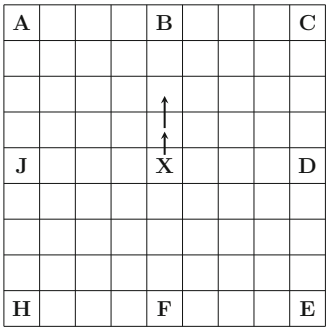

Fig. 1. Goal recognition: Where is the agent moving to?

The idea of the transformation is very simple. For soft-goals p associated with individual atoms, one adds new atoms p' that are made into *hard goals* in P' that are achievable in one of two ways: by the new actions *collect*(p) with precondition p and cost 0, or by the new actions *forgo*(p) with precondition $\overline{p}$, that stands for the negation of p, and cost equal to the utility $u(p)$ of p. Additional bookkeeping is needed in the translation so that these new actions can be done only after the actions in the original problem.

The two problems P and P' are equivalent in the sense that there is a correspondence between the plans for P and P', and corresponding plans are ranked in the same way. More specifically, for any plan π for P, there is a plan π' in P' that extends π with the *end* action and a set of *collect* and *forgo* actions whose cost is $c(\pi') = -u(\pi) + \alpha$ where α is a constant independent of both π and π'. Finding an optimal (maximum utility) plan π for P is therefore equivalent to finding an optimal (minimum cost) plan π' for P'. Interestingly, the cost-optimal planners that entered the optimal sequential track of the 2008 IPC, fed with the translations of the problems in the optimal net-benefit track, do significantly better than the net-benefit planners that entered the latter [25].

5.2 Plan and Goal Recognition

The need to recognize the goals and plans of an agent from observations of his behavior arises in a number of tasks. Goal recognition is like planning but in reverse: while in planning the goal is given and a plan is sought; in plan recognition, part of a plan is observed, and the agent goal is sought. Figure 1 shows a simple scenario of plan recognition where an agent is observed to move up twice from cell X. The question is which is the most likely destination among the possible targets A to J. Clearly, A, B and C appear to be more likely than D, E or F. The reason is that the agent is moving away from these other targets, while it's not moving away from A, B, or C. The second question is whether B can be regarded as more likely than A or C. It turns out that yes. If we adopt a Bayesian formulation, the probability of an hypothesis H given the observation Obs, $P(H|Obs)$ is given by the formula $P(H|Obs) = P(Obs|H)P(H)/P(Obs)$

where $P(Obs|H)$ represents how well the hypothesis H *predicts* the observation Obs, $P(H)$ stands for how likely is the hypothesis H a priori, and $P(Obs)$, which affects all hypotheses H equally, measures how surprising is the observation. In our problem, the hypotheses are about the possible destinations of the agent, and since there are no reasons to assume that one is more likely a priori than the others, Bayes rule yields that $P(H|Obs)$ should be proportional to the likelihood $P(Obs|H)$ that measures *how well* H *predicts* Obs. Going back to the figure, and assuming that the agent is reasonably 'rational' and hence wants to achieve his goals with least cost, it's clear that A, B, and C predict Obs better than D, E, F; and also that B predicts Obs better than A and C. This is because there is a single optimal plan for B that is compatible with Obs, but there are many optimal plans for A and for C, some of which are not compatible with Obs (as when the agent moves first left or right, rather than up). We say that a plan π is compatible with the observed action sequence Obs when the action sequence Obs is embedded in the action sequence π; i.e. when Obs is π but with certain actions in π omitted (not observed).

The reasoning above reduces goal recognition to Bayes' rule and how well each of the possible goals predicts the observed action sequence. Moreover, how well a goal G predicts the sequence Obs turns out to depend on considerations having to do with *costs*, and in particular, *two cost measures:* the cost of achieving G through a plan *compatible* with the observed action sequence Obs, and the cost of achieving G through a plan that is *not* compatible with Obs. We will denote the first cost as $c_P(G + Obs)$ and the second as $c_P(G + \overline{Obs})$, where P along with the observations Obs define the *plan recognition problem*. That is, P is like a classical planning problem but with the actual goal hidden and replaced by a set $\mathcal{G}$ of possible goals G, and a sequence of observed actions. The plan recognition problem is about inferring the probability distribution $P(G|Obs)$ over the possible goals $G \in \mathcal{G}$ where each possible goal G can be a (conjunctive) set of atoms.

The *cost differences* $\Delta(G, Obs) = c_P(G + \overline{Obs}) - c_P(G + Obs)$ for each of the possible goals G, which can range from $-\infty$ to $+\infty$, can be used to define the likelihoods $P(Obs|G)$, and hence, to obtain the goal posterior probabilities $P(G|Obs)$ when the goal priors $P(G)$ are given. Clearly, the higher the cost difference $\Delta(G, Obs)$, the better that G predicts Obs, and hence the higher the likelihood $P(Obs|G)$. The function used to map the Δ-costs into the $P(O|G)$ likelihoods is the sigmoid function, which follows from assuming that the agent is not perfectly rational [26]. The costs $c_P(G + \overline{Obs})$ and $c_P(G + Obs)$ can be computed by calling a classical planner over the two classical problems $P(G + \overline{Obs})$ and $P(G + Obs)$ that are obtained from P, the hypothetical goal G, and the observations. The result is that the goal posterior probabilities $P(G|Obs)$ can be computed through Bayes' rule and $2 \times |\mathcal{G}|$ calls to a classical planner.

5.3 Incomplete Information and Sensing

A (deterministic) *conformant problem* can be expressed as a tuple $P = \langle F, I, O, G \rangle$ where F stands for the fluents or atoms in the problem, O for the actions, I is a set of

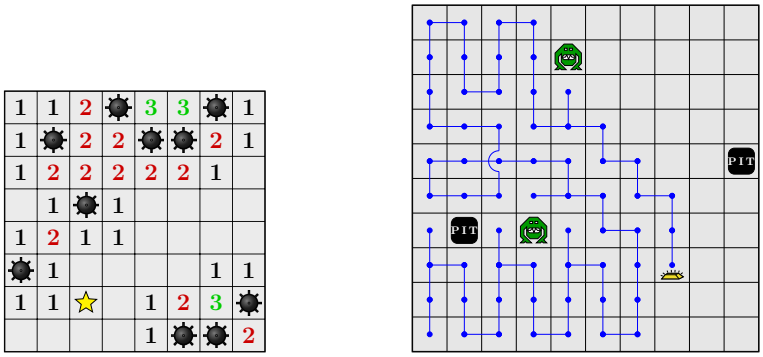

Fig. 2. Example of instances solved by on-line partially observable planner LW1 using linear translations and classical planners [27]. *Left: Minesweeper* instance where the star marks the first cell opened and empty cells have 0 counts. *Right: Wumpus* instance with 2 monsters and 2 pits. Positions of monsters, pits, and gold initially unknown.

clauses over F defining the initial situation, and G is a set of literals over F defining the (conjunctive) goal. The difference to classical problems is the uncertainty in the initial situation which is described by means of clauses. A clause is a disjunction of one or more literals, and a literal is an atom in F or its negation. We assume that the actions are not purely STRIPS but can feature conditional effects and negation; i.e., every action a is assumed to have a precondition given by a set of literals, and a set of *conditional effects* $a : C \rightarrow C'$ where C and C' are sets (conjunctions) of literals, meaning that the literals in C' become true after the action a if the literals in C were true when the action was done. The states associated with the problem P are valuations over the atoms in F, and the set of *possible initial states* are the states that satisfy the clauses in I.

A deterministic conformant problem P defines a conformant state model $\mathcal{S}(P)$ which is like the state model for a classical problem with one difference: there is no single initial state s_0 but a *set* of possible initial states S_0. A solution for P, namely a *conformant plan* for P, is an action sequence that simultaneously solves *all* the *classical state models* $\mathcal{S}'(P)$ that result from replacing the set of possible initial states S_0 in $\mathcal{S}(P)$ by each one of the states s_0 in S_0.

From a computational point of view, conformant planning can be formulated as a *path-finding problem* over a graph where the nodes in the graph do not represent the *states* of the problem as in classical planning but *belief states*, where a belief state is a set of states deemed possible at one point. An alternative approach, however, is to map deterministic conformant planning into classical ones [28]. The basic sound but incomplete translation removes the uncertainty in the problem by replacing each literal L in the conformant problem P by two literals KL and $K\neg L$, to be read as 'L is known to be true' and 'L is known to be false' respectively. If L is known to be true or known to be false in the initial situation, then the translation will contain respectively KL or $K\neg L$. On the other hand, if L is not known, then both KL and $K\neg L$ will be initially false.

The result is that there is no uncertainty in the initial situation of the translation which thus represents a classical planning problem.

More precisely, the basic translation K_0 is such that if $P = \langle F, I, O, G\rangle$ is a deterministic conformant problem, the translation $K_0(P)$ is the classical planning problem $K_0(P) = \langle F', I', O', G'\rangle$ where[2]

- $F' = \{KL, K\neg L \mid L \in F\}$
- $I' = \{KL \mid L$ is a unit clause in $I\}$
- $G' = \{KL \mid L \in G\}$
- $O' = O$ but with each precondition L for $a \in O$ replaced by KL, and each effect $a : C \to L$ replaced by $a : KC \to KL$ and $a : \neg K\neg C \to \neg K\neg L$.

The expressions KC and $\neg K\neg C$ for $C = \{L_1, L_2, \ldots\}$ are abbreviations for the conjunctions $\{KL_1, KL_2, \ldots\}$ and $\{\neg K\neg L_1, \neg K\neg L_2, \ldots\}$ respectively. Recall that in a classical planning problem, atoms that are not part of the initial situation are assumed to be initially false, so if KL is not part of I', KL will be initially false in $K_0(P)$.

The only subtlety in this translation is that each conditional effect $a : C \to L$ in P is mapped into *two* conditional effects in $K_0(P)$: a *support* effect $a : KC \to KL$ that ensures that L is known to be true when the condition C is known to be true, and a *cancellation* effect $a : \neg K\neg C \to \neg K\neg L$ that ensures that L is possible when the condition C is possible.

The translation $K_0(P)$ is *sound* as every classical plan that solves $K_0(P)$ is a conformant plan for P, but is *incomplete*, as not all conformant plans for P are classical plans for $K_0(P)$. The meaning of the KL literals follows a similar pattern: if a plan achieves KL in $K_0(P)$, then the same plan achieves L with certainty in P, yet a plan may achieve L with certainty in P without making the literal KL true in $K_0(P)$.

For completeness, the basic translation K_0 is extended into a general translation scheme $K_{T,M}$ where T and M are two parameters: a set of *tags* t and a set of *merges* m. A tag $t \in T$ is a set (conjunction) of literals L from P whose truth value in the initial situation is not known. The tags t are used to introduce a new class of literals KL/t in the classical problem $K_{T,M}(P)$ that represent the conditional statements: 'if t is initially true, then L is true'. Likewise, a merge m is a non-empty collection of tags t in T that stands for the Disjunctive Normal Form (DNF) formula $\bigvee_{t\in m} t$. A merge m is *valid* when one of the tags $t \in m$ must be true in I; i.e., when $I \models \bigvee_{t\in m} t$. A merge m for a literal L translates into a 'merge action' with effects that capture a simple form of reasoning by cases: $\bigwedge_{t\in m} KL/t \;\longrightarrow\; KL$.

The parametric translation scheme $K_{T,M}$ is the basic translation K_0 'conditioned' with the tags in T and extended with the actions that capture the merges in M. If $P = \langle F, I, O, G\rangle$ is a deterministic conformant problem, then $K_{T,M}(P)$ is the *classical planning problem* $K_{T,M}(P) = \langle F', I', O', G'\rangle$ where

[2] A conditional effect $a : C \to C'$ is assumed to be expressed as a collection of conditional effects $a : C \to L$, one for each literal L in C'. The symbol a stands for the action associated with these effects.

- $F' = \{KL/t, K\neg L/t \,|\, L \in F \text{ and } t \in T\}$,
- $I' = \{KL/t \mid I, t \models L\}$,
- $G' = \{KL \,|\, L \in G\}$,
- $O' = \{a : KC/t \rightarrow KL/t,\ a : \neg K\neg C/t \rightarrow \neg K\neg L/t \mid a : C \rightarrow L \text{ in } P\} \cup$
  $\{a_{m,L} : \left[\bigwedge_{t\in m} KL/t\right] \rightarrow KL \mid L \in P, m \in M\}$.

Two basic properties of the general translation scheme $K_{T,M}(P)$ are that it is always *sound* (provided that merges are valid), and for suitable choice of the sets of tags and merges T and M, it is *complete*. In particular, a complete instance of the general translation $K_{T,M}(P)$ results when the sets of tags T is the set S_0 of possible initial states of P, and $M = T$. While the resulting translation $K_{S0}(P)$ is exponential in the number of unknown atoms in the initial situation in the worst case, there is an alternative choice of tags and merges, called the $K_i(P)$ translation, that is exponential in the non-negative integer i, and that is *complete* for problems P that have a structural parameter $w(P)$, called the *width* of P, bounded by i. In problems defined over multivalued variables, this width stands for the maximum number of variables all of which are relevant to a variable appearing in an action precondition or goal [29]. It turns out that many conformant problems have a bounded and small width, and hence such problems can be efficiently solved by a classical planner after a low polynomial translation [28]. The conformant plans are then obtained from the classical plans by removing the 'merge' actions.

The translation-based approach, introduced initially for deterministic conformant planning, has been extended to deterministic planning with *sensing* [30,31]. Examples of problems solved by the on-line partially observable planner LW1 [27] that uses linear translations for both action selection and belief tracking are shown in Figure 2.

5.4 Finite-State Controllers

Finite-state controllers represent an action selection mechanism widely used in video-games and mobile robotics. In comparison to plans and POMDP policies, to be studied later, finite-state controllers have two advantages: they are often extremely compact, and they are general, applying not just to one problem but to many variations as well. As an illustration, Figure 3(a) depicts a simple problem over a 1×5 grid where a robot, initially at one of the two leftmost positions, must visit the rightmost position, marked B, and get back to A. Assuming that the robot can observe the mark in the current cell if any, and that the actions *Left* and *Right* deterministically move the robot one unit left and right respectively, the problem can be solved by planners that sense and POMDP planners. A solution to the problem, however, can also be expressed as the finite-state controller shown on the right. Starting in the controller state q_0, this controller selects the action *Right*, whether A or no mark ('$-$') is observed, until observing B. Then the controller selects the action *Left*, switches to state q_1, and remains in this state selecting the action *Left* as long as no mark is observed. Later, when a

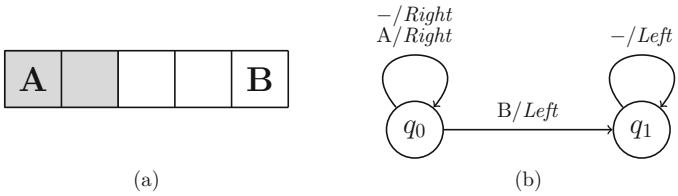

Fig. 3. (a) Agent initially in one of the two leftmost positions has to go to cell marked B and back to A. The marks are observable. (b) A 2-state controller that solves the problem and many variations of it. The circles are the controller states, and an edge $q \to q'$ labeled o/a means to perform action a when the observation is o in state q, switching then to state q'. The initial controller state is q_0.

mark is observed, no further actions are taken as the agent must be back at A, having achieved the goal.

The finite-state controller displayed in the figure has two appealing features: it is very compact (it involves two states only), and it is very general. Indeed, the problem can be changed in a number of ways and the controller would still work, driving the agent to the goal. For example, the *size of the grid* can be changed from 1×5 to $1 \times n$, the agent can be placed *initially* anywhere in the grid (except at B), and the actions can be made *non-deterministic* by adding 'noise' so that the agent can move one or two steps at a time. The controller would work for all these variations. This generality is well beyond the power of plans or policies that are normally tied to a particular state space.

The benefits of finite-state controllers, however, come at a price: unlike plans, they are usually not derived automatically but are written by hand; a task that is non-trivial even in the simplest cases. Recently, however, the problem of deriving compact and general finite-state controllers using planners has been considered [32]. Once again, this is achieved by using classical planners over suitable transformations. We sketch the main ideas below.

A finite-state controller $\mathcal{C}_N$ with N controller states $q_0, \ldots, q_{N-1}$ is fully characterized by the tuples (q, o, a, q') associated with the edges $q \stackrel{o/a}{\to} q'$ in the controller graph. These edges and hence, these tuples, prescribe the action a to do when the controller state is q and the observation is o, switching then to the controller state q' (which may be equal to q or not). A controller solves a problem P if starting in the distinguished controller state q_0, all the executions that are possible given the controller reach a goal state. The key question is how to find the tuples (q, o, a, q') that define such a controller. In [32], the problem P is transformed into a problem P' whose actions are associated with each one of the possible tuples (q, o, a, q'), and where extra fluents p_q and p_o for keeping track of the controller states and observations are introduced. The action $\langle t \rangle$ associated with the tuple $t = (q, o, a, q')$ behaves then very much like the action a but with two differences: first, the atoms p_q and p_o are added to the body of each conditional effect, so that the resulting action $\langle t \rangle$ behaves like the original

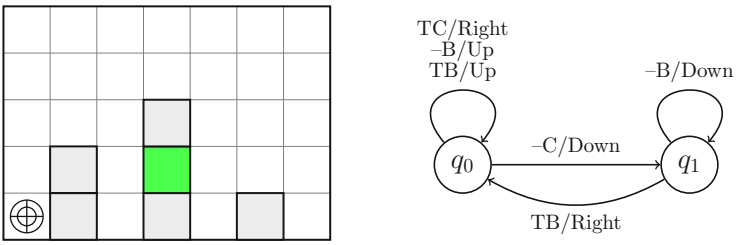

Fig. 4. *Left:* Problem where a visual-mark (on the lower left cell) must be placed on top of a green block whose location is not known, by moving the mark one cell at a time, and observing what's in the marked cell. *Right:* Finite-state controller obtained with a classical planner from translation. The controller solves the problem and *any variation* resulting from changes in either the *number* or *configuration* of blocks [32].

action a but only when the controller state is q and the observation is o; second, the action makes the atom p_q false and the atom $p_{q'}$ true, in accordance with the interpretation of the tuple (unless $q = q'$). Additional bookkeeping is required in the transformed problem P' to prevent plans from executing actions $\langle t \rangle$ and $\langle t' \rangle$ when $t = (q, o, a, q')$, $t' = (q, o, a', q'')$, and $a \neq a'$ or $q' \neq q''$. The reason is that no controller can include such pairs of tuples, as the action and new controller state are always a *function* of the current controller state and observation. Interestingly, the transformation from P into P' eliminates sensing by making the effects of the actions conditional on the current controller state and observation. The result is that while P is a partially observable problem, P' is a *conformant* problem, which as we have seen before, it can be transformed into a classical problem P''. The actions $\langle t \rangle$ that solve such classical problem encode the tuples that define the controller with up to N states that solves P.

As a further illustration of the power of these transformations, Figure 4, on the left, shows a problem inspired in the use of deictic representations where a visual-marker (the circle on the lower left) must be placed on top of a green block by moving it one cell at a time. The location of the green block is not known, and the observations are whether the cell currently marked contains a green block (G), a non-green block (B), or neither (C), and whether this cell is at the level of the table (T) or not (–). The *compact* and *general* controller shown on the right has been computed by running a *classical planner* over a translation obtained following the two steps above [32]. The controller solves the problem shown and any variation resulting from changes in either the *number* or *configuration* of blocks.

5.5 Planning with Other Agents in Mind

The muddy children puzzle is a common example used for illustrating the subtleties that arise when reasoning about the beliefs of other agents. In the problem, there are n children, k of whom have mud on their forehead. The children can see which other children are muddy but can't tell whether they are muddy or not.

The father comes and tells the children that at least one of them is muddy, and then asks the children whether they know whether they are muddy or not. It's possible to show that if the children are good reasoners, those who are muddy will know that they are muddy after the father repeats the question exactly k times [33]. When k is greater than 1, the puzzle is that the children arrive to this conclusion after expressing ignorance $k - 1$ times, and after (apparently) not learning anything new from the parent (they can all see at least one muddy child). A planning version of the problem can be constructed, for example, by asking one of the children to find out whether he is muddy or not by selecting another child X and asking him whether X knows whether he is muddy or not, with everyone listening to the response. The shortest plan in that case is to ask this question to the children seen to be muddy one by one.

The problem of characterizing the state of (nested) knowledge or beliefs in a setting where agents are able to act on the world, observe the world, and communicate their beliefs, has been studied in recent years in the area of dynamic epistemic logics [34,35]. Recently, an expressive and meaningful fragment of this logic has been identified where the methods for handling beliefs in the single agent setting are used to compute *linear plans* akin to multiagent conformant plans, using classical planners and a transformation that is quadratic in the number of possible initial states [36]. The planning version of the muddy children problem is an example of a problem that fits into this fragment. Similar methods have been developed also for computing join plans in a logical version of the multiagent planning models called Descentralized POMDPs [37].

6 Summary

Planning is the model-based approach to autonomous behavior where models come in many forms depending on the presence of uncertainty, the form of the feedback, and whether uncertainty is represented by means of sets or probability distributions. All forms of planning are intractable in the worst case when problems are represented in compact form, including the simplest form of planning, classical planning, where actions are deterministic and the initial state is fully known. In spite of this, significant progress has been achieved in classical planning in the last two decades for scaling up. Like in other AI models and solvers, the key is the exploitation of the problem structure by means of cost-effective forms of inference. In classical planning, this inference takes mainly the form of heuristics derived from the problem, landmarks that uncover implicit subgoals, and a structural criterion for distinguishing the applicable actions that are more likely to be relevant, called the helpful actions. Advances in classical planning have been exploited for dealing with non-classical problems by means of suitable transformations. Even if sound and complete translations are worst-case exponential, as in planning with incomplete information and sensing, compact polynomial but incomplete transformations have been shown to be very powerful as well. We have briefly reviewed transformations for dealing with soft goals, goal recognition, planning with incomplete information and sensing, and nested multiagent beliefs.

Acknowledgments. I have borrowed material from [15]. Some of the work reviewed has been done in collaboration with students and colleagues. I also thank João Leite and Eduardo Fermé for the invitation to talk at JELIA-2014.

References

1. Erol, K., Hendler, J., Nau, D.S.: HTN planning: Complexity and expressivity. In: Proc. 12th Nat. Conf. on Artificial Intelligence, pp. 1123–1123 (1994)
2. Levesque, H., Reiter, R., Lespérance, Y., Lin, F., Scherl, R.: GOLOG: A logic programming language for dynamic domains. J. of Logic Progr. 31, 59–83 (1997)
3. Sutton, R., Barto, A.: Introduction to Reinforcement Learning. MIT Press (1998)
4. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Prentice-Hall (2009)
5. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and practice. Morgan Kaufmann (2004)
6. Richter, S., Westphal, M.: The LAMA planner: Guiding cost-based anytime planning with landmarks. Journal of Artificial Intelligence Research 39, 127–177 (2010)
7. McDermott, D.V.: Using regression-match graphs to control search in planning. Artificial Intelligence 109(1-2), 111–159 (1999)
8. Bonet, B., Geffner, H.: Planning as heuristic search. Artificial Intelligence 129(1-2), 5–33 (2001)
9. Hoffmann, J., Porteous, J., Sebastia, L.: Ordered landmarks in planning. Journal of Artificial Intelligence Research 22, 215–278 (2004)
10. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research 14, 253–302 (2001)
11. Helmert, M.: The Fast Downward planning system. Journal of Artificial Intelligence Research 26, 191–246 (2006)
12. Kautz, H.A., Selman, B.: Pushing the envelope: Planning, propositional logic, and stochastic search. In: Proc. AAAI, pp. 1194–1201 (1996)
13. Rintanen, J.: Planning as satisfiability: Heuristics. Art. Int. 193, 45–86 (2012)
14. Yoon, S., Fern, A., Givan, R.: FF-replan: A baseline for probabilistic planning. In: Proc. 17th Int. Conf. on Automated Planning and Scheduling, pp. 352–359 (2007)
15. Geffner, H., Bonet, B.: A Concise Introduction to Models and Methods in Automated Planning. Morgan & Claypool (2013)
16. Cresswell, S., Coddington, A.M.: Compilation of LTL goal formulas into PDDL. In: Proc. 16th European Conf. on Artificial Intelligence, pp. 985–986 (2004)
17. Edelkamp, S.: On the compilation of plan constraints and preferences. In: Proc. 16th Int. Conf. on Automated Planning and Scheduling, pp. 374–377 (2006)
18. Albarghouthi, A., Baier, J.A., McIlraith, S.A.: On the use of planning technology for verification. In: Proc. ICAPS 2009 Workshop VV&PS (2009)
19. Patrizi, F., Lipovetzky, N., de Giacomo, G., Geffner, H.: Computing infinite plans for LTL goals using a classical planner. In: Proc. 22nd Int. Joint Conf. on Artificial Intelligence, pp. 2003–2008 (2011)
20. Taig, R., Brafman, R.: Compiling conformant probabilistic planning problems into classical planning. In: Proc. ICAPS (2013)
21. Brafman, R., Shani, G.: A multi-path compilation approach to contingent planning. In: Proc. AAAI (2012)
22. Palacios, H., Albore, A., Geffner, H.: Compiling contingent planning into classical planning: New translations and results. In: Proc. ICAPS Workshop on Models and Paradigms for Planning under Uncertainty (2014)

23. Smith, D.E.: Choosing objectives in over-subscription planning. In: Proc. 14th Int. Conf. on Automated Planning and Scheduling, pp. 393–401 (2004)
24. Helmert, M., Do, M.B., Refanidis, I.: 2008 IPC Deterministic planning competition. In: 6th IPC Booklet, ICAPS 2008 (2008)
25. Keyder, E., Geffner, H.: Soft goals can be compiled away. Journal of Artificial Intelligence Research 36, 547–556 (2009)
26. Ramírez, M., Geffner, H.: Probabilistic plan recognition using off-the-shelf classical planners. In: Proc. 24th Conf. on Artificial Intelligence, pp. 1121–1126 (2010)
27. Bonet, B., Geffner, H.: Flexible and scalable partially observable planning with linear translations. In: Proc. AAAI (2014)
28. Palacios, H., Geffner, H.: Compiling uncertainty away in conformant planning problems with bounded width. JAIR 35, 623–675 (2009)
29. Bonet, B., Geffner, H.: Width and complexity of belief tracking in non-deterministic conformant and contingent planning. In: Proc. 26nd Conf. on Artificial Intelligence, pp. 1756–1762 (2012)
30. Albore, A., Palacios, H., Geffner, H.: A translation-based approach to contingent planning. In: Proc. 21st Int. Joint Conf. on Artificial Intelligence, pp. 1623–1628 (2009)
31. Maliah, S., Brafman, R., Karpas, E., Shani, G.: Partially observable online contingent planning using landmark heuristics. In: Proc. ICAPS (2014)
32. Bonet, B., Palacios, H., Geffner, H.: Automatic derivation of memoryless policies and finite-state controllers using classical planners. In: Proc. ICAPS (2009)
33. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning about Knowledge. MIT Press (1995)
34. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Springer (2007)
35. van Ditmarsch, H., Kooi, B.: Semantic results for ontic and epistemic change. In: Logic and the Foundations of Game and Decision Theory (LOFT 7), pp. 87–117 (2008)
36. Kominis, F., Geffner, H.: Beliefs in multiagent planning: From one agent to many. In: Proc. ICAPS Workshop on Distributed and Multi-Agent Planning (2014)
37. Brafman, R., Shani, G., Zilberstein, S.: Qualitative planning under partial observability in multi-agent domains. In: Proc. AAAI (2013)