Reasoning about Actions with Sensing under Qualitative and Probabilistic Uncertainty

Luca Iocchi [1] and **Thomas Lukasiewicz** [1 2] and **Daniele Nardi** [1] and **Riccardo Rosati** [1]

Abstract. We focus on the aspect of sensing in reasoning about actions under qualitative and probabilistic uncertainty. We extend an $\mathcal{A}$-related action language by actions with nondeterministic and probabilistic effects, and define a formal semantics in terms of deterministic, nondeterministic, and probabilistic transitions between epistemic states. We then introduce the notions of a conditional plan and its goodness in this framework, and we formulate the conditional planning problem. We present an algorithm for solving it, which is proved to be sound and complete in the sense that it produces all optimal plans. We also report on a first prototype implementation of this algorithm. An application in a robotic-soccer scenario underlines the usefulness of our formalism in realistic applications.

1 INTRODUCTION

In reasoning about actions for mobile robots in real-world environments, one of the most crucial problems that we have to face is uncertainty, both about the initial situation of the robot's world and about the results of the actions taken by the robot. One way of adding uncertainty to reasoning about actions is based on qualitative models, in which all possible alternatives are equally considered. Another way is based on quantitative models, where we have a probability distribution on the set of possible alternatives, and thus can numerically distinguish between possible alternatives.

Well-known first-order formalisms for reasoning about actions, such as the situation calculus, easily allow for expressing qualitative uncertainty about the initial situation of the world and the effects of actions through disjunctive knowledge. Similarly, recent formalisms for reasoning about actions that are inspired by the action language $\mathcal{A}$, such as the action language $\mathcal{C}+$ [9] and the planning language $\mathcal{K}$ [6], allow for qualitative uncertainty in the form of incomplete initial states and nondeterministic effects of actions.

There are a number of formalisms for probabilistic reasoning about actions. In particular, Bacchus et al. [1] propose a probabilistic generalization of the situation calculus, which is based on first-order logics of probability, and which allows to reason about an agent's probabilistic degrees of belief and how these beliefs change when actions are executed. Poole's independent choice logic [18] is based on acyclic logic programs under different "choices". Each choice along with the acyclic logic program produces a first-order model. By placing a probability distribution over the different choices, one then obtains a distribution over the set of first-order models. Boutilier et al. [3] introduce and explore an approach to first-order Markov decision processes (MDPs) that are formulated in a probabilistic generalization of the situation calculus. A companion paper [4] presents a generalization of Golog, called DTGolog, that combines robot programming in Golog with decision-theoretic planning in MDPs. Other probabilistic extensions of the situation calculus and Golog are given in [16, 10]. A probabilistic extension of the action language $\mathcal{A}$ is given by Baral et al. [2], which aims especially at an elaboration-tolerant representation of MDPs and at formulating observation assimilation and counterfactual reasoning.

Even though there is extensive work on reasoning about actions under qualitative and probabilistic uncertainty separately, there is only few work that orthogonally combines qualitative and probabilistic uncertainty in a uniform framework for reasoning about actions. One important such approach is due to Halpern and Tuttle [11], which combines nondeterminism and probabilistic uncertainty in a game-theoretic framework. Halpern and Tuttle argue in particular that "some choices in a distributed system must be viewed as inherently nondeterministic (or, perhaps better, nonprobabilistic), and that it is inappropriate, both philosophically and pragmatically, to model probabilistically what is inherently nondeterministic". This underlines the strong need for explicitly modeling qualitative uncertainty in addition to probabilistic uncertainty in reasoning about actions.

Example 1.1 In robotic soccer, the action "align to ball" may succeed (resp., fail) with probability 0.7 (resp., 0.3), while the goalkeeper's action "open legs" may either save the goal or not. In the latter case, the effect is nondeterministic rather than probabilistic, as it is not possible to assign probabilities to the possible effects, which in fact depend on external factors (e.g., speed and kind of kick performed by an opponent robot) and cannot be evaluated a priori. □

The work [7] is among the few papers that orthogonally combine qualitative and probabilistic uncertainty in a uniform framework for reasoning about actions. It presents the language P$\mathcal{C}+$ for probabilistic reasoning about actions that allows for expressing probabilistic and nondeterministic effects of actions as well as probabilistic and qualitative uncertainty about the initial situation of the world. A formal semantics of P$\mathcal{C}+$ is defined in terms of probabilistic transitions between sets of states. This work, however, does not consider the crucial issue of sensing and thus of conditional planning.

In this paper, we aim at filling this gap. We develop a formalism that additionally allows for *sensing in reasoning about actions under qualitative and probabilistic uncertainty*, and thus to formulate the problem of *conditional planning under qualitative and probabilistic uncertainty*. In particular, we elaborate a sound and complete algorithm for solving it. For ease of presentation, the base action language that we use in this paper is an $\mathcal{A}$-related language that is less expressive than $\mathcal{C}+$, but our results can be easily extended to $\mathcal{C}+$ and related languages as base action language. The main contributions of

[1] DIS, Università di Roma "La Sapienza", Via Salaria 113, 00198 Rome, Italy; e-mail: {iocchi, lukasiewicz, nardi, rosati}@dis.uniroma1.it.
[2] Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, 1040 Vienna, Austria; e-mail: lukasiewicz@kr.tuwien.ac.at.

the present paper are the following:

- We extend a language for reasoning about actions with sensing by actions with nondeterministic and probabilistic effects. We define a formal semantics of action descriptions through systems of deterministic, nondeterministic, and probabilistic transitions between epistemic states, which are sets of possible world states.
- We formulate the problem of conditional planning under qualitative and probabilistic uncertainty and define conditional plans. Based on the concept of a belief tree, we then define the goodness of a conditional plan relative to a goal and an initial observation.
- We present an algorithm for conditional planning under qualitative and probabilistic uncertainty, and we prove in particular that this algorithm is sound and complete in the sense that it generates the set of all optimal conditional plans. We also describe a first prototype implementation of the above algorithm.
- A formulation of a robotic-soccer scenario gives evidence of the usefulness of our formalism in realistic applications.

Note that all proofs and further details (especially on implementation and experimental results) are given in the extended report [12].

2 ACTION LANGUAGE

In this section, we describe the base formalism that we use for reasoning about actions. It is related to the action language $\mathcal{A}$, but any other action language with similar expressiveness could also be used. As main features, it allows for sensing actions and for modeling an agent's *epistemic state*, which encodes what the agent knows about the world, in contrast to what is true in the world. Following the literature (e.g. [15, 20]), the knowledge of the agent is characterized through the set of possible states which satisfy the known properties. Reasoning in presence of sensing is done by modeling the dynamics of the agent's epistemic state, rather than the dynamics of the world.

A dynamic system is specified through an initial state description and an action description, which allow for modeling what an agent knows about the initial properties of the world and how this knowledge changes through the execution of actions. We now describe the syntax and the semantics of the initial state and action descriptions.

Syntax. We assume a nonempty finite set of variables $\mathcal{F}$, called *fluents*, which are divided into *static* and *dynamic* fluents. We use $\bot$ and $\top$ to denote the constants $false$ and $true$, respectively. A *fluent literal* is either a fluent f or its negation $\neg f$. A *fluent conjunction* is of the form $l_1 \wedge \cdots \wedge l_n$, where $l_1, \ldots, l_n$ are fluent literals and $n \geqslant 1$. The set of *fluent formulas* is the closure of $\mathcal{F} \cup \{\bot, \top\}$ under the Boolean operators $\neg$, $\wedge$, and $\vee$ (that is, if ϕ and ψ are fluent formulas, then also $\neg\phi$, $\phi \wedge \psi$, and $\phi \vee \psi$). We assume a set of *actions* $\mathcal{A}$, which are divided into *effect* and *sensing* actions.

A *precondition axiom* is of the form **executable** α **if** ϕ, where ϕ is a fluent formula, and α is an action. Informally, α is executable in every state that satisfies ϕ. If $\phi = \top$, then α is always executable.

A *conditional effect axiom* has the form **caused** ψ **after** α **when** ϕ (abbreviated as **caused** ψ **after** α, when $\phi = \top$), where ϕ is a fluent formula, ψ is a fluent conjunction, and α is an action. Informally, if the current state satisfies ϕ, then executing α produces the effect ψ.

A *sensing effect axiom* is of the form **caused to know** ω **or** $\neg\omega$ **after** α, where ω is a fluent conjunction, and α is a sensing action. Informally, after executing α, the agent knows that ω is either true or false. That is, sensing actions modify the epistemic state of the agent without affecting the state of the world [15].

A *default frame axiom* has the form **inertial** ϕ **after** α, where ϕ is a fluent conjunction, and α is an action. Informally, if ϕ holds in the current state, then ϕ holds also after the execution of α, if it is consistent with the effects of α.

A *domain constraint axiom* is of the form **caused** ψ **if** ϕ, where ϕ and ψ are fluent formulas. It represents background knowledge, which is invariant w.r.t. the execution of actions.

Definition 2.1 An *initial state description* ϕ_I is a fluent formula. An *action description* KB is a finite set of precondition, conditional effect, sensing effect, default frame, and domain constraint axioms. A *goal description* ψ_G is a fluent formula.

Semantics. An initial state description ϕ represents an epistemic state (that is, a set of possible states), while an action description KB encodes a system of state transitions between epistemic states (a directed graph, where epistemic states serve as nodes, and the outgoing arrows of each node are labeled with pairwise distinct actions). We first define states, epistemic states, the executability of an action, and the transition between epistemic states by executing an action.

A *state* s is a truth assignment to the fluents in $\mathcal{F}$. It is *admissible* with an action description KB iff s satisfies all domain constraint axioms in KB (that is, s satisfies $\phi \Rightarrow \psi$ for all **caused** ψ **if** ϕ in KB). An *epistemic state* (or *e-state*) is a set S of admissible states. An e-state S *satisfies* a fluent formula ϕ iff every $s \in S$ satisfies ϕ.

An action α is *executable* in an e-state S iff S satisfies ϕ, for every precondition axiom **executable** α **if** ϕ in KB. Given an e-state S and an effect action α executable in S, we denote by *direct*(α, S) the conjunction of all ψ such that **caused** ψ **after** α **when** ϕ is in KB and ϕ is satisfied by S. Then, the *successor e-state* of S under an effect action α, denoted $\Phi(S, \alpha)$, is the set S' of all admissible states that satisfy (i) *direct*(α, S), and (ii) every ϕ such that **inertial** ϕ **after** α is in KB, ϕ is satisfied by S, and $\Psi \wedge$ *direct*$(\alpha, S) \wedge \phi$ is satisfiable, where Ψ is the conjunction of all $\phi \Rightarrow \psi$ for each **caused** ψ **if** ϕ in KB. Intuitively, S' encodes the direct effects of α (since it satisfies *direct*(α, S)), the indirect effects due to the domain constraint axioms, and the propagation of inertial properties that are consistent with the above direct and indirect effects.

The *successor e-state* of S under a sensing action α with outcome $\sigma \in \{\omega, \neg\omega\}$, denoted $\Phi(S, \alpha_\sigma)$, is the set of all admissible states that satisfy σ and every ϕ such that **inertial** ϕ **after** α is in KB, ϕ is satisfied by S, and the formula $\Psi \wedge \sigma \wedge \phi$ is satisfiable, where Ψ is the conjunction of all $\phi \Rightarrow \psi$ for each **caused** ψ **if** ϕ in KB.

Definition 2.2 An action description KB encodes the directed graph $G_{KB} = (N, E)$, where N is the set of all e-states, and E contains $S \to S'$ labeled with an effect action α (resp., sensing action α with outcome $\sigma \in \{\omega, \neg\omega\}$) iff (i) α is executable in S and (ii) $S' = \Phi(S, \alpha)$ (resp., $S' = \Phi(S, \alpha_\sigma)$). An initial state description ϕ encodes the set S_ϕ of all admissible states that satisfy ϕ.

3 NONDETERMINISM AND PROBABILITIES

We now extend the action language of the previous section by actions with nondeterministic and probabilistic effects.

Syntax. We divide the set of effect actions into *deterministic*, *nondeterministic*, and *probabilistic* actions. The possible effects of the latter two types of actions are encoded in dynamic context formulas. A *nondeterministic dynamic context formula* has the form

$$\textbf{caused } \psi_1, \ldots, \psi_n \textbf{ after } \alpha \textbf{ when } \phi, \tag{1}$$

where ϕ is a fluent formula, $\psi_1, \ldots, \psi_n$ are fluent conjunctions, α is an action, and $n \geqslant 2$. Informally, if the current state satisfies ϕ, then

executing α has at least one of the effects ψ_i. A *probabilistic dynamic context formula* is an expression of the form

$$\textbf{caused } \psi_1 : p_1, \ldots, \psi_n : p_n \textbf{ after } \alpha \textbf{ when } \phi, \qquad (2)$$

where additionally $p_1, \ldots, p_n > 0$ and $p_1 + \cdots + p_n = 1$. Informally, if the current state satisfies ϕ, then executing α has the effect ψ_i with the probability p_i. We omit "**when** ϕ" in (1) and (2), when $\phi = \top$.

Definition 3.1 An *extended action description* $D = (KB, C)$ consists of an action description KB and a finite set C containing exactly one nondeterministic (resp., probabilistic) dynamic context formula for each nondeterministic (resp., probabilistic) action in KB.

Semantics. We define the semantics of an extended action description $D = (KB, C)$ through a system of deterministic, nondeterministic, and probabilistic transitions between e-states. To this end, we add to the transition system of KB a mapping that assigns to each pair (S, α) of a current e-state S and a nondeterministic (resp., probabilistic) action α executable in S, a set (resp., a probability distribution on a set) of successor e-states after executing α.

Note that *probabilistic transitions* are like in partially observable Markov decision processes (POMDPs) [13], but they are between epistemic states and thus *sets of states* rather than *single states*.

Each nondeterministic (resp., probabilistic) action α in KB, which has its dynamic context formula (1) (resp., (2)) in C, is associated with a set of *contexts* $V_\alpha = \{v_1, \ldots, v_n\}$, where each v_i has the probability $Pr_\alpha(v_i) = p_i$, if α is probabilistic. We use $KB_\alpha(v_i)$ to denote KB enlarged by **caused** ψ_i **after** α **when** ϕ. If α is executable in an e-state S, then the *successor e-state* of S after executing α in its context v, denoted $\Phi_v(S, \alpha)$, is the e-state $\Phi(S, \alpha)$ under $KB_\alpha(v)$.

Definition 3.2 Let α be an action that is executable in an e-state S. If α is nondeterministic, then the *set of successor e-states* of S under α is defined as $F_\alpha(S) = \{\Phi_v(S, \alpha) \mid v \in V_\alpha\}$. If α is probabilistic, then the *probability distribution on the successor e-states* of S under α, denoted $Pr_\alpha(\cdot\,|S)$, is defined by $Pr_\alpha(S'|S) = \sum_{v \in V_\alpha,\, S' = \Phi_v(S,\alpha)} Pr_\alpha(v)$. We say D is *consistent* iff $\emptyset \notin F_\alpha(S)$ (resp., $Pr_\alpha(\emptyset|S) = 0$) for every nondeterministic (resp., probabilistic) action α and every e-state S in which α is executable.

Intuitively, executing a nondeterministic action α in an e-state S nondeterministically leads to some $S' \in F_\alpha(S)$, while executing a probabilistic action α in S leads to $S' = \Phi_v(S, \alpha)$, $v \in V_\alpha$, with probability $Pr_\alpha(S'|S)$. In the rest of this paper, we implicitly assume that every action description $D = (KB, C)$ is consistent.

Example 3.3 We describe the actions of a goalkeeper in robotic soccer, specifically in the RoboCup Four-Legged League. The extended action description is shown in Fig. 1. It includes the fluents cb (the robot is close to the ball), ba (the ball is in the penalty area), fa (the space ahead the goalkeeper is free), ip (the goalkeeper is in the correct position), bm (the ball is moving towards its own goal), ab (the goalkeeper is aligned with the direction of the ball), and gs (the goal has been saved). The actions are gotoball (a movement towards the ball, which possibly touches the ball and moves it outside the penalty area), bodykick, straightkick, and sidekick (three different kinds of kicks with different capabilities), openlegs (a position for intercepting a ball kicked towards its own goal), aligntoball (a movement for aligning to the direction of the ball moving towards its own goal), and several sensing actions for some of the properties.

Note that the action openlegs has both the deterministic effect that the goalkeeper is able to save the goal when it is aligned to the ball direction, as well as nondeterministic effects, which encode a possible capability of saving the goal even when the alignment is not known. In addition, if the robot is assumed to be always in its own area, then the axiom **caused** ba **if** cb allows for defining indirect effects of actions (e.g., in several actions, the effect $\neg$ba indirectly implies $\neg$cb). Finally, all the fluents are inertial in this example. □

executable gotoball **if** ba$\wedge\neg$bm
executable bodykick **if** cb
executable straightkick **if** cb$\wedge$fa
executable sidekick **if** cb$\wedge\neg$fa
executable aligntoball **if** bm
executable openlegs **if** bm
executable sensealignedtoball **if** bm
caused gs **after** openlegs **when** ab
caused to know cb **or** $\neg$cb **after** senseballclose
caused to know fa **or** $\neg$fa **after** sensefreeahead
caused to know ab **or** $\neg$ab **after** sensealignedtoball
inertial l **after** α (for every fluent literal l and action α)
caused ba **if** cb
caused gs, $\neg$gs **after** openlegs
caused cb:0.8, $\neg$ba:0.1, $\neg$cb:0.1 **after** gotoball
caused $\neg$ba$\wedge\neg$ip:0.1, $\neg$ba$\wedge$ip:0.5, $\neg$ip:0.1, $\top$:0.3 **after** bodykick
caused $\neg$ba:0.9, $\top$:0.1 **after** straightkick
caused $\neg$ba:0.7, $\top$:0.3 **after** sidekick
caused ab:0.7, $\neg$ab:0.3 **after** aligntoball

Figure 1. Extended action description

4 CONDITIONAL PLANS

The conditional planning problem can be described as follows. Given an extended action description, an initial state description ϕ_I, and a goal description ψ_G, compute the best conditional plan to achieve ψ_G from ϕ_I. In this section, we first define conditional plans in our framework. We next introduce belief trees, which are then used to define the goodness of a conditional plan for achieving ψ from ϕ.

Conditional plans. A conditional plan is a binary directed tree where each arrow represents an action, and each branching expresses the two outcomes of a sensing action, which can thus be used to select the proper actions. Recall that a directed tree is a directed acyclic graph (DAG) in which every node has exactly one parent, except for the *root*, which has no parents; nodes without children are *leaves*. Formally, a *conditional plan* Π is either (i) the *empty conditional plan*, denoted λ, or (ii) of the form $\alpha; \Pi'$, or (iii) of the form $\beta; \textbf{if } \omega \textbf{ then } \{\Pi_\omega\} \textbf{ else } \{\Pi_{\neg\omega}\}$, where α is an effect action, β is a sensing action of outcomes ω and $\neg\omega$, and Π', Π_ω, and $\Pi_{\neg\omega}$ are conditional plans. We often abbreviate "$\pi; \lambda$" by "π".

Example 4.1 We use the domain of Example 3.3 for defining two planning problems. The first specifies an initial situation $\phi_I = \text{ba} \wedge \text{ip} \wedge \neg\text{bm}$, in which the robot is in its standard position and the ball is in its own area and it is not moving, and a goal description $\psi_G = \neg\text{ba} \wedge \text{ip}$, which requires the robot to kick away the ball and to remain in its position. Two conditional plans that solve this problem are $\Pi_1 = \text{gotoball}; \text{bodykick}$ and $\Pi_2 = \text{gotoball}; \text{sensefreeahead}; \textbf{if } \text{fa } \textbf{then } \{\text{straightkick}\} \textbf{ else } \{\text{sidekick}\}$.

The second problem specifies an initial situation $\phi_I = \text{bm}$, in which the ball is moving, and a goal description $\psi_G = \text{gs}$, where the goal has been saved. Some conditional plans that solve this second problem are $\Omega_1 = \text{openlegs}$, $\Omega_2 = \text{aligntoball}; \text{openlegs}$, and $\Omega_3 = \text{sensealignedtoball}; \textbf{if } \text{ab } \textbf{then } \{\text{openlegs}\} \textbf{ else } \{\text{aligntoball}; \text{openlegs}\}$. □

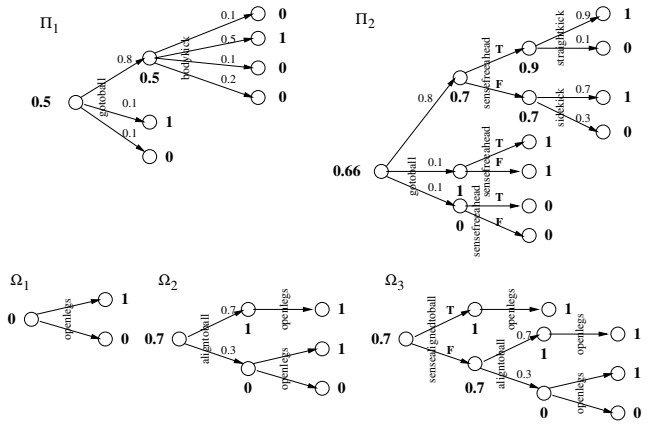

Figure 2. Computing the goodness of a conditional plan

Belief trees. A belief tree is a directed tree over epistemic states as nodes. Each arrow (eventually with a probability) represents a transition, and every branching represents the different effects (resp., outcomes) of some effect (resp., sensing) action. Formally, the *belief tree* $T=(G, Pr)$ for Π under ϕ, denoted $T_{\phi\Pi}$, consists of a directed tree $G=(V, E)$, where the nodes are pairs (S, Ω) of an epistemic state S and a conditional plan Ω, and a mapping $Pr\colon E \rightarrow [0, 1]$, which are constructed by the following steps (1)–(3):

(1) Initially, T is only the node (S_ϕ, Π).

(2) Let $T=(G, Pr)$ be the tree built thus far. For each leaf (S, Ω), where the first action α of Ω is executable in S, enlarge T by:

(2.1) If α is a sensing action, and Ω has the form α; **if** ω **then** $\{\Omega_\omega\}$ **else** $\{\Omega_{\neg\omega}\}$, then add to T each arrow $(S, \Omega) \rightarrow (S', \Omega_\sigma)$ such that $S' = \Phi(S, \alpha_\sigma) \neq \emptyset$ and $\sigma \in \{\omega, \neg\omega\}$.

(2.2) If α is a deterministic (resp., nondeterministic) effect action, and Ω has the form $\alpha; \Omega'$, then add to T every arrow $(S, \Omega) \rightarrow (S', \Omega')$ with $S' = \Phi(S, \alpha)$ (resp., $S' \in F_\alpha(S)$).

(2.3) If α is a probabilistic effect action, and $\Omega = \alpha; \Omega'$, then add to T all $e = (S, \Omega) \rightarrow (S', \Omega')$ with $S' = \Phi_v(S, \alpha)$ for some $v \in V_\alpha$ along with $Pr(e) = \sum_{v \in V_\alpha,\, S' = \Phi_v(S,\alpha)} Pr_\alpha(v)$.

(3) Repeat (2) until T is free of leaves (S, Ω) such that the first action α of Ω is executable in S.

Goodness. The goodness of a conditional plan Π for achieving a goal ψ under an initial state description ϕ is defined using the belief tree $T_{\phi\Pi} = ((V, E), Pr)$ as follows. The success (resp., failure) leaves of $T_{\phi\Pi}$ have goodness 1 (resp., 0). We then propagate the goodness to every node of $T_{\phi\Pi}$, using the goodness of the children and the probabilities associated with an arrow. The goodness of Π is the goodness of the root of $T_{\phi\Pi}$. Formally, the *goodness* of Π for achieving ψ under ϕ is defined as $g_{\phi,\psi}(\Pi) = g(R)$, where R is the root of $T_{\phi\Pi}$, and the function $g\colon V \rightarrow [0, 1]$ is defined by:

- $g(v) = 1$ (resp., $g(v) = 0$) for every leaf $v = (S, \Omega) \in V$ (*success leaf* (resp., *failure leaf*)) such that $\Omega = \lambda$ and $\forall s \in S\colon s \models \psi$ (resp., $\Omega \neq \lambda$ or $\exists s \in S\colon s \not\models \psi$);
- $g(v) = \min_{v \rightarrow v' \in E} g(v')$ for every node $v = (S, \Omega) \in V$ such that the first action of Ω is either a sensing action, or a deterministic effect action, or a nondeterministic effect action;
- $g(v) = \sum_{v \rightarrow v' \in E} Pr(v \rightarrow v') \cdot g(v')$ for all $v = (S, \Omega) \in V$ such that the first action of Ω is a probabilistic effect action.

Example 4.2 The belief trees and the goodness of the conditional plans given in Example 4.1 are shown in Fig. 2. □

Algorithm Plan Generation.
Input: G_{KB}, initial state description ϕ_I, goal description ψ_G.
Output: $SP = \{\Pi_i = \{(S_i, A, S_j)\}\}$: set of conditional plans with positive goodness.

S_0 is an initial e-state in which ϕ_I holds;
$SP = \mathit{findAllPaths}(G_{KB}, S_0, \psi_G, \emptyset)$;
while $\exists P \in SP\colon (S_i, A_i, S_j) \in P \ \wedge\ (S_i, \neg A_i, S_k) \notin P$ **do**
  $SP_{aux} = \mathit{findAllPaths}(G_{KB}, S_k, \psi_G, \mathcal{F}_P(S_0, S_k))$;
  $SP = SP - \{P\}$;
  for each $P_{aux} \in SP_{aux}$ **do**
    $P_{new} = P \cup \{(S_i, \neg A_i, S_k)\} \cup P_{aux}$;
    $SP = SP \cup \{P_{new}\}$
  end for
end while;
$SP = \mathit{unify}(SP)$;
return SP.

Figure 3. Plan Generation Algorithm

5 CONDITIONAL PLANNING

The problem of conditional planning in our framework can be defined as follows. Given an extended action description $D = (KB, C)$, an initial state description ϕ_I, and a goal description ψ_G, compute a conditional plan Π that, when executed from an e-state in which ϕ_I holds, leads to an e-state in which the goal ψ_G holds (for any possible outcome of sensing) with maximum goodness.

We now present a planning method for solving this problem, which is divided into two steps: (1) computing all the valid conditional plans that are solutions to the planning problem; (2) evaluating the goodness for each of these conditional plans and selecting the best one (that is, the conditional plan with maximum goodness).

The first step of the planning method is achieved by the algorithm shown in Fig. 3 for extracting plans from the graph G_{KB}, representing a planning problem, as shown in the previous sections. The output of this algorithm is the set of all conditional plans that are solutions of the planning problem. Each plan is a directed acyclic graph represented as a set of tuples (S_i, A_i, S_j), whose meaning is that from the e-state S_i it is possible to execute the action A_i leading to the successor e-state S_j. When the action A_i is a sensing action, then the tuple (S_i, A_i, S_j) denotes an execution of A_i whose outcome is true, while the tuple $(S_i, \neg A_i, S_k)$ denotes an execution of A_i whose outcome is false. Now, the definition of conditional plan implies that, for each tuple (S_i, A_i, S_j) in P there exists a tuple of the form $(S_i, \neg A_i, S_k)$ in P. If this condition is not satisfied in a given (partially built) plan P, then the e-state S_i in P must be further expanded.

The algorithm uses the function $\mathit{findAllPaths}(G_{KB}, S, \psi_G, \mathcal{F})$ that returns the set of all possible paths (without cycles) in G_{KB} from the e-state S to an e-state in which ψ_G holds, without considering any of the e-states specified in $\mathcal{F}$. Since a path is a sequence of actions, in this function, only one outcome of a sensing action or one context of an action with either nondeterministic or probabilistic effects is considered. The returned linear path may have e-states that must be further expanded, when they lack the other outcome of sensing.

Moreover, since we are only interested in generating conditional plans (without cycles), it is necessary to limit the search of the paths in $\mathit{findAllPaths}$, by excluding the e-states that have already been considered in the path from the initial e-state to the current e-state: this is obtained by the computation of the set of e-states $\mathcal{F}_P(S_0, S_k)$, which includes all the e-states in the current plan P such that there exists a path from S_0 to S_k. In this way, the function $\mathit{findAllPaths}$ never returns a path that can produce cycles.

Observe that the first part of the algorithm only finds those plans that are valid by considering a single context for each nondetermin-

istic or probabilistic action. However, it is also necessary to derive plans that consider at the same time multiple contexts. To this end, it is possible to generate more general plans by combining pairs of previously computed ones. This is performed by a unification operation (implemented by the $unify$ procedure) that unifies terms that suitably represent conditional plans.

The second step of the planning method is a procedure that computes the goodness for all the conditional plans retrieved in the previous step and returns the best one. Observe that, for efficiency reasons, the planning algorithm given above generally builds conditional plans in the form of DAGs, while the goodness of such plans is defined on conditional plans expressed in the form of trees. Thus, for the computation of the goodness a transformation of the DAG into a tree (by recursively duplicating those subgraphs in the DAG whose root has more than one parent) is needed.

Correctness of the algorithm is formally expressed as follows.

Theorem 5.1 (Plan Generation) *Given an extended action description $D = (KB, C)$, an initial state description ϕ_I, and a goal description ψ_G, the algorithm Plan Generation terminates and is sound and complete, i.e., it returns all the solutions (and only solutions) to the conditional planning problem.*

Our current implementation of the Plan Generation algorithm considers the following aspects: (i) G_{KB} is generated on-line during the search for the plan, according to Definition 2.2; (ii) the algorithm finds a plan that has an overall goodness greater than a given threshold γ and thus the computation of the optimal plan (the one with the maximum goodness) can be obtained by a small number of executions of this procedure; (iii) the unification process is not implemented as a separate step, but within the process of plan generation, and, for efficiency reasons, it makes use of an heuristic that considers only a particular but significative portion of the possible unification forms. These implementation choices allowed us to realize an efficient implementation of the planner that has been used for generating significant plans both in abstract domains (taken from the literature) and in the mobile robot domain described before.

6 RELATED WORK

With regard to the literature on reasoning about actions with probabilistic effects, the most closely related approach is Poole's independent choice logic (ICL) [18], which uses a similar way of adding probabilities to an approach based on acyclic logic programs. But the central conceptual difference is that Poole's ICL does not allow for qualitative uncertainty in addition to probabilistic uncertainty. Poole circumvents the problem of dealing with qualitative uncertainty by imposing the strong condition of acyclicity on logic programs.

From a more general perspective, our approach is also related to planning under uncertainty in AI, since it can roughly be understood as a combination of conditional planning under nondeterministic uncertainty in AI [8] with conditional planning under probabilistic uncertainty in AI, both in partially observable environments.

Generalizations of classical planning in AI including actions with probabilistic effects, see for example [5, 17, 14], typically consider the problem of determining a sequence of actions given a success threshold, with some extension that considers also sensing and conditional plans. On the other hand, decision-theoretic planning in AI considers fully or partially observable Markov decision processes (MDPs [19] or POMDPs [13]), which also include costs and/or rewards associated with actions and/or e-states, and their solutions are mappings from situations to actions of high expected utility, rather than courses of actions achieving a goal with high probability.

Our approach can be seen as combining conditional planning under nondeterministic and under probabilistic uncertainty, where the latter is perhaps closest to generalizations of classical planning in AI. But instead of giving a threshold for the success probability of a plan, we aim at all plans with highest possible success probability. In contrast to the decision-theoretic framework, we do not assume costs and/or rewards associated with actions and/or states. Furthermore, sensing actions in our approach are more flexible than observations in POMDPs, since they allow for preconditions, and they can be performed at any time point when executable.

Acknowledgments. This work has been partially supported by the Austrian Science Fund under project Z29-N04 and by a Marie Curie Individual Fellowship of the European Union programme "Human Potential" under contract number HPMF-CT-2001-001286 (disclaimer: The authors are solely responsible for information communicated and the European Commission is not responsible for any views or results expressed). We are grateful to the reviewers for their constructive comments, which helped to improve our work.

References

[1] F. Bacchus, J. Y. Halpern, and H. J. Levesque, 'Reasoning about noisy sensors and effectors in the situation calculus', *Artif. Intell.*, **111**(1–2), 171–208, (1999).

[2] C. Baral, N. Tran, and L.-C. Tuan, 'Reasoning about actions in a probabilistic setting', in *Proc. AAAI/IAAI-2002*, pp. 507–512, (2002).

[3] C. Boutilier, R. Reiter, and B. Price, 'Symbolic dynamic programming for first-order MDPs', in *Proc. IJCAI-2001*, pp. 690–700, (2001).

[4] C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun, 'Decision-theoretic, high-level agent programming in the situation calculus', in *Proc. AAAI/IAAI-2000*, pp. 355–362, (2000).

[5] D. Draper, S. Hanks, and D. S. Weld, 'Probabilistic planning with information gathering and contingent execution', in *Proc. AIPS-1994*, pp. 31–36, (1994).

[6] T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres, 'A logic programming approach to knowledge-state planning, II: The DLV$^{\mathcal{K}}$ system', *Artif. Intell.*, **144**(1–2), 157–211, (2003).

[7] T. Eiter and T. Lukasiewicz, 'Probabilistic reasoning about actions in nonmonotonic causal theories', in *Proc. UAI-03*, pp. 192–199, (2003).

[8] H. Geffner, 'Perspectives on artificial intelligence planning', in *Proc. AAAI/IAAI-2002*, pp. 1013–1023, (2002).

[9] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner, 'Nonmonotonic causal theories', *Artif. Intell.*, **153**(1–2), 49–104, (2004).

[10] H. Grosskreutz and G. Lakemeyer, 'Belief update in the pGOLOG framework', in *Proc. KI/ÖGAI-2001*, pp. 213–228, (2001).

[11] J. Y. Halpern and M. R. Tuttle, 'Knowledge, probability, and adversaries', *J. ACM*, **40**(4), 917–962, (1993).

[12] L. Iocchi, T. Lukasiewicz, D. Nardi, and R. Rosati, 'Reasoning about actions with sensing under qualitative and probabilistic uncertainty', Technical Report INFSYS RR-1843-03-05, Institut für Informationssysteme, TU Wien, (2003).

[13] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, 'Planning and acting in partially observable stochastic domains', *Artif. Intell.*, **101**(1–2), 99–134, (1998).

[14] L. Karlsson, 'Conditional progressive planning under uncertainty', in *Proc. IJCAI-2001*, pp. 431–438, (2001).

[15] H. J. Levesque, 'What is planning in presence of sensing?', in *Proc. AAAI-1996*, pp. 1139–1149, (1996).

[16] P. Mateus, A. Pacheco, J. Pinto, A. Sernadas, and C. Sernadas, 'Probabilistic situation calculus', *Ann. Math. Artif. Intell.*, **32**, 393–431, (2001).

[17] N. Onder and M. E. Pollack, 'Conditional, probabilistic planning: A unifying algorithm and effective search control mechanisms', in *Proceedings AAAI/IAAI-1999*, pp. 577–584, (1999).

[18] D. Poole, 'The independent choice logic for modelling multiple agents under uncertainty', *Artif. Intell.*, **94**(1–2), 7–56, (1997).

[19] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, 1994.

[20] T. C. Son and C. Baral, 'Formalizing sensing actions: A transition function based approach', *Artif. Intell.*, **125**(1–2), 19–91, (2001).