Comparing Action Descriptions based on Semantic Preferences *

Thomas Eiter, Esra Erdem, Michael Fink, and Ján Senko

Institute of Information Systems, Vienna University of Technology, Austria
Email: (eiter | esra | michael | jan)@kr.tuwien.ac.at

Abstract. We consider action domain descriptions whose meaning can be represented by transition diagrams. We introduce several semantic measures to compare such action descriptions, based on preferences over possible states of the world and preferences over some given conditions (observations, assertions, etc.) about the domain, as well as the probabilities of possible transitions. This preference information is used to assemble a weight which is assigned to an action description. As an application of this approach, we study the problem of updating action descriptions with respect to some given conditions. With a semantic approach based on preferences, not only, for some problems, we get more plausible solutions, but also, for some problems without any solutions due to too strong conditions, we can identify which conditions to relax to obtain a solution. We conclude with computational issues, and characterize the complexity of computing the semantic measures.

1 Introduction

This paper discusses how to compare action descriptions, whose meaning can be represented by transition diagrams—a directed graph whose nodes correspond to states and edges correspond to transitions caused by action occurrences and non-occurrences, with respect to some given conditions. Comparison of action descriptions is important for applications, when an agent has to prefer one description more than the others. One such application is the action description update problem [1]: when an agent tries to update an action description with respect to some given information, she usually ends up with several possibilities and has to choose one of these action descriptions. Another application is related to representing an action domain in an elaboration tolerant way (for a definition of elaboration tolerance see, e.g., [2, 3]): among several action descriptions representing the same action domain, which one is the most elaboration tolerant one, with respect to some given conditions describing possible elaborations?

The preference of an agent over action descriptions may be based on a syntactic measure, such as the number of formulas: the less the number of formulas contained in an action description, the more preferred it is. A syntactic measure can be defined also in terms of set containment with respect to a given action description D: an action description is more preferred if it is a maximal set among others that is contained in D. For instance, according to the syntactic measure used in [1] for updating an action description D with some new knowledge Q, an action description D' is more preferred if D' is a maximal set among others containing D and contained in $D \cup Q$ is maximum.

* Work supported by the Austrian Science Fund (FWF) under grant P16536-N04.

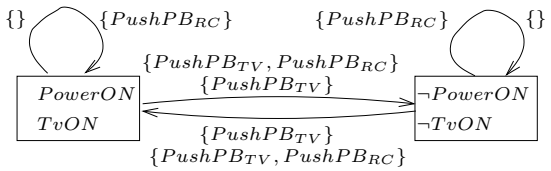

Fig. 1. A transition diagram.

In this paper, we describe the preference of an agent over action descriptions, with respect to some semantic measure. The idea is to describe a semantic measure by assigning weights (i.e., real numbers) to action descriptions, with respect to their transition diagrams and some given conditions; then, once the weights of action descriptions are computed, to compare two descriptions by comparing their weights.

We consider action descriptions, in a fragment of the action language $\mathcal{C}$ [4], which consists of "causal laws." For instance, the causal law

$$\textbf{caused } PowerON \textbf{ after } PushPB_{TV} \wedge \neg PowerON, \tag{1}$$

expresses that the action $PushPB_{TV}$ causes the value of the fluent $PowerON$ to change from f to t; such causal laws describe direct effects of actions. The causal law

$$\textbf{caused } TvON \textbf{ if } PowerON, \tag{2}$$

expresses that if the fluent $PowerON$ is caused to be true, then the fluent $TvON$ is caused to be true as well; such causal laws describe state constraints. The meaning of an action description D can be represented by a transition diagram, like in Fig. 1. In this transition diagram, the nodes of the graph (shown by boxes) denote the states of the world: (s) one where both the power and the TV is on, and (s') the other where both the power and the TV is off. The edges denote action occurrences. For instance, the edge from s to s' labeled by the action of pushing the power button on the TV describes that executing this action at s leads to s'. The edges labeled by the empty set are due to the law of inertia.

Suppose that we are given another action description D' describing the domain above; and that the transition diagram of D' is almost the same as that of D, except that there is no outgoing edge from the state $\{PowerON, TvON\}$ with the label $\{PushPB_{RC}\}$. Which action description should be preferred? In answering this question, we also take given *conditions* (observations, assertions, etc.) on the action domain into account. We describe conditions in an action query language, like in [5], by "queries." For instance,

$$\textbf{ALWAYS } \bigvee_{A \in 2^{\mathbf{A}}} \textbf{executable } A, \tag{3}$$

where $2^{\mathbf{A}}$ denotes the set of all actions, expresses that, at every state, there is some action executable. The query

$$\textbf{SOMETIMES evolves } PowerON; \{PushPB_{RC}\}; PowerON \tag{4}$$

expresses that, at some state when the power is on, pushing the power button on the remote control does not turn the power off.

The question we consider in this paper is then the following:

Given a set $\mathcal{D}$ of action descriptions and a set C of queries, which action description in $\mathcal{D}$ is a most preferred one with respect to C?

Our main contributions are briefly summarized as follows.

- We provide an answer to the above question with respect to mainly four *semantically-oriented* approaches, by assigning weights to action descriptions in $\mathcal{D}$, based on their transition diagrams. The weights express preferences of the agent over possible states of the world and preferences over conditions, as well as the probabilities of possible transitions.
A simple weight measure is to count the number of queries in C which an action description D entails. In the example above, D entails according to its transition diagram (3) and (4), so D has weight 2; D' entails according to its transition diagram only (3), so D' has weight 1. Hence, D is preferred over D'.
- We apply these approaches to the problem of updating an action description, and observe two benefits. First, if a problem has many solutions with the syntactic approach of [1], a semantic approach can be used to pick one. Second, if a problem does not have any solution with any of the approaches due to too strong conditions, a semantic approach can be used to identify which conditions to relax to find a solution.
- We characterize the computational cost of computing the weight assignments, which lays the foundations for efficient computation.

For space reasons, we omit the definitions of transition diagrams and action descriptions. *They are as in [1] and given in an extended version [6],*[1] *which contains further explanation of the examples, additional examples, another application, and a detailed discussion of the complexity results and algorithms.*

2 Action Queries

To talk about observations of the world, or assertions about the effects of the execution of actions, we use an action query language consisting of queries described as follows. We start with *basic queries*: (a) *static queries* of the form

$$\textbf{holds } F, \tag{5}$$

where F is a fluent formula; (b) *dynamic queries* of the form

$$\textbf{necessarily } Q \textbf{ after } A_1; \ldots; A_n, \tag{6}$$

where Q is a basic query and each A_i is an action; and (c) every propositional combination of basic queries. An *existential query* is an expression of the form

$$\textbf{SOMETIMES } Q, \tag{7}$$

where Q is a basic query; a *universal query* is of the form

$$\textbf{ALWAYS } Q, \tag{8}$$

where Q is a basic query. A *query* q is a propositional combination of existential queries and universal queries.

[1] Available at `http://www.kr.tuwien.ac.at/research/ad-cmp.pdf`.

As for the semantics, let $T = \langle S, V, R\rangle$ be a transition diagram, with a set S of states, a value function V mapping, at each state s, every fluent P to a truth value, and a set R of transitions. A *history* of T of length n is a sequence

$$s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n \tag{9}$$

where each $\langle s_i, A_{i+1}, s_{i+i}\rangle$ ($0 \leq i < n$) is in R. We say that a state $s \in S$ *satisfies* a basic query Q' of form (5) (resp. (6)) relative to T (denoted $T, s \models Q'$), if the interpretation $P \mapsto V(P, s)$ satisfies F (resp. if, for every history $s = s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n$ of T of length n, basic query Q is satisfied at state s_n). For other forms of basic queries Q, *satisfaction* is defined by the truth tables of propositional logic. If T is described by an action description D, then the satisfaction relation between s and a basic query Q can be denoted by $D, s \models Q$ as well.

Note that, for every state s and for every fluent formula F, $D, s \models$ **holds** F iff $D, s \models \neg$**holds** $\neg F$. For every state s, every fluent formula F, and every action sequence $A_1, \ldots, A_n$ ($n \geq 1$), if $D, s \models$ **necessarily** (**holds** F) **after** $A_1; \ldots; A_n$ then $D, s \models \neg$**necessarily** ($\neg$**holds** F) **after** $A_1; \ldots; A_n$.

We say that D *entails* a query q (denoted $D \models q$) if one of the following holds:

- q is an existential query (7) and $D, s \models Q$ for some state $s \in S$;
- q is a universal query (8) and $D, s \models Q$ for every state $s \in S$,
- $q = \neg q'$ and $D \not\models q'$;
- $q = q_1 \wedge q_2$ and $D \models q_1$ and $D \models q_2$; or
- $q = q_1 \vee q_2$ and $D \models q_1$ or $D \models q_2$.

For every basic query Q, $D \models$ **SOMETIMES** Q iff $D \models \neg$**ALWAYS** $\neg Q$. For a set C of queries, we say that D *entails* C (denoted $D \models C$) if D *entails* every query in C. Consider, e.g., the action description consisting of (1), (2), and

$$\begin{array}{l} \textbf{caused } \neg PowerON \textbf{ after } PushPB_{TV} \wedge PowerON \\ \textbf{caused } \neg TvON \textbf{ if } \neg PowerON \\ \textbf{inertial } PowerON, \neg PowerON, TvON, \neg TvON \end{array} \tag{10}$$

encoding how a TV system operates; here **inertial** $L_1, \ldots, L_k$ stands for the causal laws **caused** L_i **if** L_i **after** L_i ($1 \leq i \leq k$). It does not entail any set of queries containing

$$\textbf{ALWAYS necessarily } (\textbf{holds } \neg TvON) \textbf{ after } \{PushPB_{RC}\}$$

because this query is not satisfied at the state $\{TvON, PowerON\}$; but, it entails the queries:

$$\begin{array}{l} \textbf{ALWAYS holds } PowerON \equiv TvON, \\ \textbf{ALWAYS holds } PowerON \wedge TvON \supset \\ \qquad \neg\textbf{necessarily } (\textbf{holds } TvON) \textbf{ after } \{PushPB_{TV}\}. \end{array} \tag{11}$$

In the rest of the paper, an expression of the form

$$\textbf{possibly } Q \textbf{ after } A_1; \ldots; A_n,$$

where Q is a basic query and each A_i is an action, stands for the dynamic query $\neg$**necessarily** $\neg Q$ **after** $A_1; \ldots; A_n$; an expression of the form

$$\textbf{evolves } F_0; A_1; F_1; \ldots; F_{n-1}; A_n; F_n, \tag{12}$$

where each F_i is a fluent formula, and each A_i is an action, stands for **holds** $F_0 \wedge$ **possibly** (**holds** $F_1 \wedge$ **possibly** (**holds** $F_2 \wedge ...$) **after** A_2) **after** A_1; and

$$\textbf{executable } A_1; \ldots; A_n,$$

where each A_i is an action, stands for **possibly** $True$ **after** $A_1; \ldots; A_n$. We sometimes drop **holds** from static queries appearing in dynamic queries.

Queries allow us to express various pieces of knowledge about the domain. For instance, we can express the existence of states where a formula F holds by means of the query **SOMETIMES holds** F. Similarly, we can express the existence of a transition from some state where a formula F holds to another state where a formula F' holds, by the execution of an action A:

$$\textbf{SOMETIMES holds } F \wedge \textbf{possibly } F' \textbf{ after } A\ .$$

In general, the existence of a history (9) such that, for each s_i of the history, the interpretation $P \mapsto V(P, s_i)$ satisfies some formula F_i is expressed by the query:

$$\textbf{SOMETIMES evolves } F_0; A_1; F_1; \ldots; F_{n-1}; A_n; F_n. \tag{13}$$

For instance, the query

$$\begin{aligned}\textbf{SOMETIMES evolves } & PowerON; \{PushPB_{TV}\}; \\ & \neg PowerON; \{PushPB_{TV}\}; PowerON.\end{aligned} \tag{14}$$

describes the presence of the following history in Fig. 1:

$$\begin{aligned}&\{PowerON, TvON\}, \{PushPB_{TV}\}, \\ &\{\neg PowerON, \neg TvON\}, \{PushPB_{TV}\}, \{PowerON, TvON\}.\end{aligned} \tag{15}$$

That at some state where formula F holds no action is possible is expressed by

$$\textbf{SOMETIMES holds } F \wedge \bigwedge\nolimits_{A \in 2^{\mathbf{A}}} \textbf{necessarily } False \textbf{ after } A.$$

Like in [1], executability of an action sequence $A_1, \ldots, A_n$ ($n \geq 1$) at every state can be described by **ALWAYS executable** $A_1; \ldots; A_n$; mandatory effects of a sequence $A_1, \ldots, A_n$ ($n \geq 1$) of actions in a given context by **ALWAYS holds** $G \supset$ **necessarily** F **after** $A_1; \ldots; A_n$; and possible effects of a sequence of actions in a context by **ALWAYS holds** $G \supset$ **possibly** F **after** $A_1; \ldots; A_n$. In the last two queries, F describes the effects and G the context.

3 Weight Assignments for Action Descriptions

To compare action descriptions with respect to their semantics, we can assign weights to them, based on their transition diagrams and a given set of conditions. We present below four weight assignments, each with a different motivation expressing some appeal of the action description, however, without an a priori epistemic meaning. They are by no means exhaustive, i.e., many more are conceivable, but allow to specify preferences over the main semantic constituents—states, transitions, queries, and a combination thereof. Corresponding orders are total and, unlike more general preferences (partial orders), beneficial wrt. discrimination of choices or component-wise comparability.

3.1 Weighted states

We can specify our preference over states of a transition diagram $\langle S, V, R\rangle$ by assigning a weight to each state in S, by a function g. Such a function assigning real numbers to

states of the world can be considered as a *utility function*, as in decision theory. If one state of the world is preferred to another state of the world then it has higher utility for the agent; here "utility" is understood as "the quality of being useful" as in [7]. Alternatively, the function g can be viewed as a *reward function*: being at a state s will give a reward of $g(s)$ to the agent.

Given a utility function for a set S of states, the highly preferred states relative to a given number l are states with a weight greater than l. Then, one way to define the weight of an action description D relative to g and l is as follows:

$$weight_s(D) = |\{s : s \in S, g(s) > l\}|.$$

With respect to this definition, the more the number of states that are highly preferred by the agent, the more preferred the action description is.

For instance, consider the transition diagram in Fig. 1 described by D. Take, for each $s \in S$,

$$g(s) = \begin{cases} 2 \text{ if } PowerON \in s \\ 1 \text{ otherwise.} \end{cases} \tag{16}$$

Take $l = 1$. Then $weight_s(D) = 1$.

3.2 Weighted queries

We can assign weights to queries to specify preferences over conditions they express:

Let C be a set of queries, along with a weight function f mapping each query in C to a real number. Then one way to define the weight of D (relative to C and f) is by

$$weight_q(D) = \sum\nolimits_{c \in C, D \models c} f(c).$$

Intuitively, the weight of an action description defined relative to the weights of queries shows how much the set C of given preferable queries are satisfied. (Note that f can easily express a threshold function as well.) *With this definition, the more the highly preferred queries are satisfied, the more preferred the action description is.*

For instance, suppose that C consists of (14) and

$$\textbf{ALWAYS executable } \{PushPB_{RC}\}, \tag{17}$$

with weights 1 and 2 respectively. For the description D with the transition diagram in Fig. 1, $weight_q(D) = 3$.

3.3 Weighted histories

In a transition diagram $T = \langle S, V, R \rangle$, we will say that a history (9) of length n is *desired* with respect to a given query (13), if, for each i, the interpretation $P \mapsto V(P, s_i)$ satisfies F_i.

Let D be an action description, and $T = \langle S, V, R \rangle$ be the transition diagram described by D. Let C be a set of queries, along with a weight function f mapping each condition in C to a number. Let H_C be the set of pairs (w, c) such that w is a desired history in T with respect to the query c of form (13) in C. Let us denote by $st(w)$ the starting state s_0 of a history w of form (9). We define a function h mapping each desired history w appearing in H_C to a real number, in terms of the utility $u(w)$ of state $st(w)$ with respect to w:

$$h(w) = u(w) \times \sum_{(w,c)\in H_C} f(c).$$

The function u mapping a history w of form (9) to a real number can be defined in terms of a sequence of functions u_i. Given a utility function (or a reward function) g mapping each state in S to a real number, and a *transition model* m mapping each transition $\langle s, A, s'\rangle$ in R to a probability (i.e., the probability of reaching s' from s after execution of A):

$$\begin{aligned} u_n(w) &= g(s_n) \\ u_i(w) &= g(s_i) + m(\langle s_i, A_{i+1}, s_{i+1}\rangle) \times u_{i+1}(w)\ (0 \le i < n) \\ u(w) &= u_0(w). \end{aligned}$$

These equations are essentially obtained from the equations used for value determination in the policy-iteration algorithm described in [7, Chapter 17]: take $\{s_0, \ldots, s_n\}$ as the set of states, $\langle s_i, A_{i+1}, s_{i+1}\rangle$ as the possible transitions, the mapping $s_i \mapsto A_{i+1}$ as the fixed policy, U as u, U_i as u_i, R as g, and M as m. Then we can define the weight of D in terms of the weights of desired histories $w_1, \ldots, w_z$ appearing in H_C as follows:

$$weight_h(D) = \sum_{i=1}^{z} h(w_i).$$

The more the utilities of desired histories (or trajectories) satisfied by the action description, the more preferred the action description is.

For instance, suppose that C consists of query (14), with weight 3. Consider the transition diagram $T = \langle S, V, R\rangle$ in Fig. 1. Let us denote history (15) by w, and query (14) by c. Then H_C contains (w, c). Take $g(s)$ as in (16). Take $l = 1$. Suppose that, for each transition $\langle s, A, s'\rangle$ in R,

$$m(\langle s, A, s'\rangle) = \begin{cases} 0.5 \text{ if } s = \{PowerON, TvON\} \wedge\ |A| = 1 \\ 1 \quad \text{otherwise.} \end{cases} \tag{18}$$

Then $u(w)$ is computed as 3.5. and $h(w) = u(w) \times \sum_{(w,c)\in H_C} f(c) = 3.5 \times 3 = 10.5$. Hence $weight_h(D) = 10.5$.

3.4 Weighted queries relative to weighted states

The three approaches above can be united by also considering to what extent each universal query in C is entailed by the action description. The idea is while computing the weight of a description relative to weighted queries, to take into account the states at which these queries are satisfied.

Let D be an action description. Let $T = \langle S, V, R\rangle$ be the transition diagram described by D, along with a weight function g mapping each state in T to a real number. Let C be a set of queries such that every query q in C is an existential query, a universal query, or a disjunction of both.

First, for each state s in S, we compute its new weight $g'(s)$, taking into account utilities of the desired histories starting with s. Let H_C be the set of pairs (w, c) such that w is a desired history in T with respect to the query c of form (13) in C. Let W be the set of histories that appear in H_C. Let u be a function mapping a history w to a real number, describing the utility of state s with respect to w. Then the new weight function g' is defined as follows:

$$g'(s) = \begin{cases} g(s) & \text{if } \nexists w(w \in W \wedge st(w) = s) \\ \sum_{w \in W, st(w)=s} u(w) & \text{otherwise.} \end{cases}$$

Next, for each query c in C, we compute its new weight $f'(c)$. Let f be a function mapping each condition in C to a real number. We will denote by $S_D(B)$ the set of states s such that $D, s \models B$. Then we define f' as follows:

$$f'(q) = \begin{cases} f'(q') + f'(q'') & \text{if } q = q' \vee q'' \\ \beta & \text{if } q = \textbf{ALWAYS } B \\ \gamma & \text{if } q = \textbf{SOMETIMES } B \wedge |S_D(B)| > 0 \\ 0 & \text{if } q = \textbf{SOMETIMES } B \wedge |S_D(B)| = 0, \end{cases}$$

where $\beta = f(q) \times \sum_{s \in S_D(B)} g'(s)$ and $\gamma = f(q) \times [(\sum_{s \in S_D(B)} g'(s))/|S_D(B)|]$. Intuitively, f' describes to what extent each preferable query q is satisfied.

Then the weight of D (relative to C and f') is the sum:

$$weight_{qs}(D) = \sum\nolimits_{q \in C} f'(q).$$

Intuitively, $weight_{qs}(D)$ describes how much and to what extent the given preferable queries are satisfied by D. For instance, suppose C consists of three queries:

$$\textbf{ALWAYS executable } \{PushPB_{TV}\}, \tag{19}$$

$$\textbf{SOMETIMES } \neg\textbf{executable } \{PushPB_{RC}, PushPB_{TV}\}, \tag{20}$$

and query (14), denoted by c_1, c_2 and c_3 respectively. Consider an action description D, with the transition diagram in Fig. 1. Let us denote history (15) by w; then $H_C = \{(w, c_3)\}$. Take the utility function g as in (16), and the transition model m as in (18). Take $f(c_1) = 1,\ f(c_2) = 2,\ f(c_3) = 3$. Then $g'(\{PowerON, TvON\}) = 3.5$, $g'(\{\neg PowerON, \neg TvON\}) = 1$, and $f'(c_1) = 4$, $f'(c_2) = 4$, $f'(c_3) = 10.5$. Therefore, $weight_{qs}(D) = 18.5$.

Further discussion and additional examples considering the weight functions in different action domains are given in the extended version [6].

4 Application: Updating an Action Description

Suppose that an action description D consists of two parts: D_u (unmodifiable causal laws) and D_m (modifiable causal laws); and a set C of conditions is partitioned into two: C_m (must) and C_p (preferable). We define an *Action Description Update (ADU)* problem by an action description $D = (D_u, D_m)$, a set Q of causal laws, a set $C = (C_m, C_p)$ of queries, all with the same signature, and a weight function $weight$ mapping an action description to a number. The weight function can be defined relative to a set of queries, a utility function, or a transition model, as seen in the previous section. We say that a consistent action description D' is a *solution* to the ADU problem $(D, Q, C, weight)$ if

(i) $Q \cup D_u \subseteq D' \subseteq D \cup Q$,
(ii) $D' \models C_m$,

(iii) there is no other consistent action description D'' such that $Q \cup D_u \subseteq D'' \subseteq D \cup Q$, $D'' \models C_m$, and $weight(D'') > weight(D')$.

The definition of an ADU problem in [1] is different from the one above mainly in two ways. First, $C_p = \emptyset$. Second, instead of (iii) above, the following syntactic condition is considered: there is no consistent action description D'' such that $D' \subset D'' \subseteq D \cup Q$, and $D'' \models C$.

The semantic approach above has mainly two benefits, compared to the syntactic approach of [1]. First, there may be more than one solution to some ADU problems with the syntactic approach. In such cases, a semantic approach may be applied to pick one of those solutions. Example 1 illustrates this benefit. Second, for an ADU problem, if no consistent action description D' satisfying (i) satisfies the must queries (C_m), there is no solution to this problem with either syntactic or semantic approach. In such a case, we can use the semantic approach with weighted queries, to relax some must queries in C_m (e.g., move them to C_p). The idea is first to solve the ADU problem $((D_u, D_m), Q, (\emptyset, C'_m), weight)$, where C'_m is obtained from C_m by complementing each query, and where the weights of queries in C'_m are equal to some very small negative integer; and then to identify the queries of C'_m satisfied in a solution and add them C_p, with weights multiplied by -1. This process of relaxing some conditions of C_m to find a solution is illustrated in Example 2.

Example 1. Consider, for instance, an action description $D = (D_m, D_u)$, where $D_m = \{(1), (2)\}$ and D_u is (10), that describes a TV system with a remote control. Suppose that, later the following information, Q, is obtained:

caused $TvON$ **after** $PushPB_{RC} \wedge PowerON \wedge \neg TvON$
caused $\neg TvON$ **after** $PushPB_{RC} \wedge TvON$.

Suppose that we are given the set $C = (C_m, C_p)$ of queries where C_m consists of the queries (3) and

SOMETIMES evolves $\neg TvON; \{PushPB_{TV}\}; \neg TvON$, (21)

and C_p consists of the queries (14), (20), (19), (17), (4), denoted by $c_1, \ldots, c_5$ respectively. When Q is added to D, the meaning of $D \cup Q$ can be represented by a transition diagram almost the same as in that of D (Fig. 1), except that there is no outgoing edge from the state $\{PowerON, TvON\}$ with the label $\{PushPB_{RC}\}$; thus only (3), (21), and (14) in C are entailed by $D \cup Q$. The question is how to update D by Q so that the must conditions, C_m, are satisfied, and the preferable conditions, C_p, are satisfied as much as possible.

The consistent action descriptions for which (i) holds are $D^{(1)} = D \cup Q$, $D^{(2)} = D_u \cup Q \cup \{(2)\}$, $D^{(3)} = D_u \cup Q \cup \{(1)\}$, $D^{(4)} = D_u \cup Q$. With the syntactic approach of [1], we have to choose between $D^{(2)}$ and $D^{(3)}$, since they have more causal laws. Consider the semantic approach based on weighted histories (i.e., $weight = weight_h$), with (16) as the utility function g, (18) as the transition model m, and $f(c_1) = 3, f(c_2) = 1, f(c_3) = 4, f(c_4) = 3, f(c_5) = 2$. Let us consider the states $s_0 = \{PowerON, TvON\}$, $s_1 = \{PowerON, \neg TvON\}$, $s_2 = \{\neg PowerON, \neg TvON\}$; and the histories

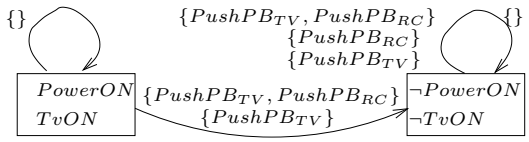

Fig. 2. Transition diagram of $D^{(2)} = D_u \cup Q \cup \{(2)\}$.

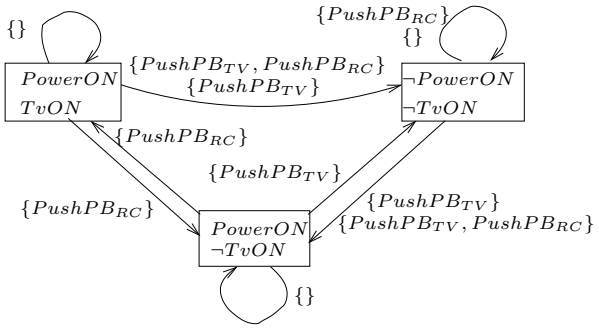

Fig. 3. Transition diagram of $D^{(3)} = D_u \cup Q \cup \{(1)\}$.

$$w_0 = s_0, \{PushPB_{RC}\}, s_1, \qquad w_2 = s_0, \{PushPB_{TV}\}, s_2, \{PushPB_{TV}\}, s_1,$$
$$w_1 = s_1, \{PushPB_{RC}\}, s_0, \qquad w_3 = s_1, \{PushPB_{TV}\}, s_2, \{PushPB_{TV}\}, s_1$$

with utilities $u(w_0) = 3, u(w_1) = 4, u(w_2) = 3.5, u(w_3) = 5$.

For $D^{(2)}$ (Fig. 2), since $H_{C_p} = \emptyset$, $weight_h(D^{(2)}) = 0$. For $D^{(3)}$ (Fig. 3), since H_{C_p} contains (w_0, c_5), (w_1, c_5), (w_2, c_3), and (w_3, c_3), $weight_h(D^{(3)}) = 48$. Thus $D^{(3)}$ is the solution.

Example 2. Let D, Q, C_p, and $D^{(1)}$–$D^{(4)}$ as in Example 1 and C_m consist of

$$\textbf{SOMETIMES} \neg \bigvee\nolimits_{A \in 2^{\mathbf{A}}} \textbf{executable } A, \tag{22}$$

$$\textbf{ALWAYS } \neg\textbf{evolves } \neg TvON; \{PushPB_{TV}\}; \neg TvON, \tag{23}$$

denoted by c_1' and c_2' respectively. None of the descriptions $D^{(1)}$ – $D^{(4)}$ entails C_m. Therefore, there is no solution to the ADU problem above with either the syntactic approach of [1] or any of the semantic approaches above. To identify which queries in C_m we shall move to C_p, first we obtain C_m' from C_m by negating each query in C_m, and assigning a very small negative integer, say –100, as their weights. So C_m' consists of the queries (3) and (21), denoted by c_1'' and c_2'', with weights -100. With the semantic approach based on weighted queries (i.e., $weight = weight_q$),

$$weight_q(D^{(1)}) = f(c_1'') = -100,$$
$$weight_q(D^{(2)}) = weight_q(D^{(3)}) = f(c_1'') + f(c_2'') = -200,$$
$$weight_q(D^{(4)}) = f(c_1'') + f(c_2'') = -200$$

the description $D^{(1)}$ is the solution to the ADU problem given by $((D_u, D_m), Q, (\emptyset, C_m'), weight_q)$. This suggests relaxing the must query (22) (i.e., adding the query (22) to C_p with the weight 100) and solving the new ADU problem, $((D_u, D_m), Q, \{(23)\}, C_p \cup \{(22)\}, weight_q)$, for which the description $D_u \cup Q$ is the solution.

Table 1. Complexity of computing weights (completeness).

Input / Weight	$weight_s$	$weight_q$	$weight_h$	$weight_{qs}$
D, C	#P	FPSPACE	GapP *	FPSPACE
D, C, S	polynomial			
${D_{pol}}^{**}$, C	in $\mathrm{FP}_{\|}^{\mathrm{NP}}$			

* #P for non-negative $g(s)$,$f(q)$; ** $|S|$ is polynomially bounded

Other semantic approaches to action description updates. Given a consistent action description E, condition (iii) of an ADU problem $(D, Q, C, weight)$ can be replaced by

> (iii)′ there is no other consistent D'' such that $Q \cup D_u \subseteq D'' \subseteq D \cup Q$, $D'' \models C_m$, and $|weight(D'') - weight(E)| < |weight(D') - weight(E)|$

to express that, among the consistent action descriptions D' for which (i) and (ii) hold, an action description that is "closest" to (or most "similar" to) E is picked. Here, for instance, E may be $D \cup Q$, to incorporate as much of the new information as possible, although $D \cup Q$ may not entail C. What is meant by closeness or similarity is based on the particular definition of the weight function. For instance, based on the weights of the states only, with $g(s) = 1$ if s is a state of E, and 0 otherwise, the closeness of an action description to E is defined in terms of the common world states.

A further application of weight-based comparison of action descriptions to assess the elaboration tolerance of different representations of an action domain is considered in [6].

5 Computational Aspects

We confine here to discuss the complexity, in order to shed light on the cost of computing the weight measures. We assume that the basic functions $g(s)$, $f(q)$, as well as $m(\langle s, A, s' \rangle)$ are computable in polynomial time. For a background on complexity, we refer to the literature (see e.g. [8]).[2]

Apparently, none of the different weights above is polynomially computable from an input action description D and a set C of queries in general. Indeed, deciding whether S has any states is NP-complete, thus intractable. Furthermore, evaluating arbitrary queries q on D ($D \models q$) is a PSPACE-complete problem. Indeed, q can be evaluated by a simple recursive procedure in polynomial space. On the other hand, evaluating Quantified Boolean Formulas, which is PSPACE-complete, can be reduced to deciding $D \models q$.

Computation given D and C. As it turns out, all four weights are computable in polynomial space. This is because each weight is a sum of (in some cases exponentially many) terms, each of which can be easily computed in polynomial space, using exhaustive enumeration. In some cases, the computation is also PSPACE-hard, but in others supposedly easier:

[2] See also `http://qwiki.caltech.edu/wiki/Complexity_Zoo`

Theorem 1. *Given an action description D, a set C of queries, and polynomial-time computable basic functions $g(s)$, $f(q)$, and $m(\langle s, A, s' \rangle)$,*

(i) Computing $weight_s(D)$ relative to g is, #P-complete;
(ii) Computing $weight_q(D)$ relative to C and f is FPSPACE-complete;
(iii) Computing $weight_h(D)$ relative to C, f, g and m is #P-complete (modulo a normalization, which casts the problem to one on integers), if the range of f and g are nonnegative numbers, and GapP-complete for arbitrary f and g;
(iv) Computing $weight_{qs}(D)$ relative to C, f, g and m is FPSPACE-complete.

These results are also shown in the first row of Table 1. Here #P [8] is the class of the problems where the output is an integer that can be obtained as the number of the runs of an NP Turing machine accepting the input, and GapP [9, 10] is the closure of #P under subtraction (equivalently, the functions expressible as the number of accepting computations minus the number of rejecting computations of an NP Turing machine). These problems are trivially solvable in polynomial time with an oracle #P, and no such problem is believed to be PSPACE-hard.

Computation given D, C, and states S of D. Informally, a source of complexity is that D may specify an exponentially large transition diagram T. If T is given, then all four weights are polynomially computable. In fact, not all of T is needed, but only a *relevant part*, denoted $T_C(D)$, which comprises all states and all transitions that involve actions appearing in C.

Now if the state set S is known (e.g., after computation with CCALC [11]) or computable in polynomial time, then $T_C(D)$ is constructible in polynomial time. Indeed, for each states $s, s' \in S$ and each action A occurring in some query, we can test in polynomial time whether $\langle s, A, s' \rangle$ is a legal transition with respect to D; the total number of such triples is polynomial in $|S|$. Then the following result (the second row of Table 1) holds.

Theorem 2. *Given an action description D, the set S of states described by D, a set C of queries, and polynomial-time computable basic functions $g(s)$, $f(q)$, and $m(\langle s, A, s' \rangle)$. Then $weight_s(D)$ (relative to g), $weight_q(D)$ (relative to C and f), $weight_h(D)$ (relative to C, f, g and m), and $weight_{qs}(D)$ (relative to C, f, g and m), are all computable in polynomial time.*

Intuitively, for $weight_q(D)$ this holds since we can decide whether a query q from C holds with respect to $T_C(D)$ in polynomial time using standard labeling methods from model checking [12]. We can compute $weight_h(D)$ with similar labeling techniques, reshuffling the weight and utility functions $h(w)$ and $u(w)$, respectively, such that considering exponentially many paths in $T_C(D)$ explicitly is avoided.

Computation given D and C for polynomial state set S. Finally, if the state space S is not large, i.e., $|S|$ is polynomially bounded, S is computable with the help of an NP-oracle in polynomial time; in fact, this is possible with parallel NP oracles queries, and thus computing S is in the respective class $\mathrm{FP}_{\|}^{\mathrm{NP}}$. From Theorem 2, we thus obtain the following results (the third row of Table 1):

Theorem 3. *Given an action description D such that $|S|$ is polynomially bounded, a set C of queries, and polynomial-time computable basic functions $g(s)$, $f(q)$, and $m(\langle s, A, s' \rangle)$, Then computing each of the weight functions, $weight_s(D)$ (relative to g), $weight_q(D)$ (relative to C and f), $weight_h(D)$ (relative to C, f, g and m), and $weight_{qs}(D)$ (relative to C, f, g and m), is in $FP_{\|}^{\mathrm{NP}}$.*

On the other hand, tractability of any of the weight functions in the case where $|S|$ is polynomially bounded is unlikely, since solving SAT under the assertion that the given formula F has at most one model (which is still considered to be intractable) is reducible to computing $weight_p(D)$ for each $p \in \{s, q, h, qs\}$.

6 Conclusion

We have presented four ways of assigning weights to action descriptions, based on the preferences over states, preferences over conditions, and probabilities of transitions, so that one can compare the action descriptions by means of their weights. To the best of our knowledge, this paper is the first attempt in this direction. Moreover, we have characterized the computational cost of the weight assignments, providing a basis for efficient algorithms.

We have illustrated the usefulness of such a semantically-oriented approach of comparing action descriptions, on the problem of updating an action description, in comparison with the syntactic approach of [1]. Further examples and applications are considered in the extended version of this paper [6].

Further work will aim at implementations of the weight measures, based on the complexity characterizations and algorithms obtained (cf. [6]) and to investigate restricted problem classes. Another issue is to explore further measures.

References

1. Eiter, T., Erdem, E., Fink, M., Senko, J.: Updating action domain descriptions. In: Proc. IJCAI. (2005) 418–423
2. McCarthy, J.: Elaboration tolerance. In: Proc. CommonSense. (1998)
3. Amir, E.: Towards a formalization of elaboration tolerance: Adding and deleting axioms. In: Frontiers of Belief Revision. Kluwer (2000)
4. Giunchiglia, E., Lifschitz, V.: An action language based on causal explanation: Preliminary report. In: Proc. AAAI. (1998) 623–630
5. Gelfond, M., Lifschitz, V.: Action languages. ETAI **3** (1998) 195–210
6. Eiter, T., Erdem, E., Fink, M., Senko, J.: Comparing action descriptions based on semantic preferences. Extended manuscript. Available at `http://www.kr.tuwien.ac.at/research/ad-cmp.pdf` (2006)
7. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall (1995)
8. Papadimitriou, C.: Computational Complexity. Addison-Wesley (1994)
9. Fenner, S.A., Fortnow, L., Kurtz, S.A.: Gap-definable counting classes. Journal of Computer and System Sciences **48** (1994) 116–148
10. Gupta, S.: Closure properties and witness reduction. Journal of Computer and System Sciences **50** (1995) 412–432
11. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., Turner, H.: Nonmonotonic causal theories. AI **153** (2004) 49–104
12. Clark, E., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (1999)