Comparing Action Descriptions based on Semantic Preferences

Thomas Eiter and **Esra Erdem** and **Michael Fink** and **Ján Senko**[1]

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 further study computational issues, and give a characterization of the computational complexity of the 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 nonoccurrences, 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 [2]: 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 [9, 1]: 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 [2] 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.

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

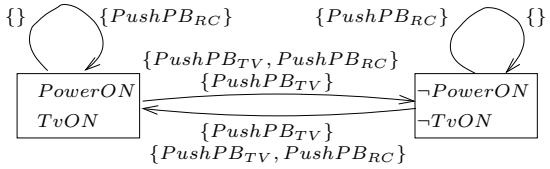

Figure 1. A transition diagram.

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 $\mathcal{C}$ [6], which consists of "causal laws." For instance, the causal law

$$\textbf{caused } PowerON \textbf{ after } PushPB_{TV} \wedge \neg PowerON, \qquad (1)$$

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

$$\textbf{caused } TvON \textbf{ if } PowerON, \qquad (2)$$

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 Figure 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? To answer this question, we assign weights to these two action descriptions, based on their transition diagrams, and given conditions (observations, assertions, etc.).

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 [2], 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 additional examples and another application, we refer the reader to an extended version [3], available at `http://www.kr.tuwien.ac.at/research/ad-cmp.pdf`.

2 TRANSITION DIAGRAMS AND ACTION DESCRIPTIONS

We start with a *(propositional) action signature* that consists of a set $\mathbf{F}$ of fluent names, and a set $\mathbf{A}$ of action names. An *action* is a truth-valued function on $\mathbf{A}$, denoted by the set of action names that are mapped to t. A *(propositional) transition diagram* of an action signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$ consists of a set S of *states*, a function $V : \mathbf{F} \times S \to \{f, t\}$, and a subset $R \subseteq S \times 2^{\mathbf{A}} \times S$ of *transitions*. We say that $V(P, s)$ is the *value* of P in s. The states s' such that $\langle s, A, s' \rangle \in R$ are the possible *results of the execution* of the action A in the state s. We say that A is *executable* in s, if at least one such state s' exists. A transition diagram can be thought of as a labeled directed graph. Every state s is represented by a vertex labeled with the function $P \mapsto V(P, s)$ from fluent names to truth values. Every triple $\langle s, A, s' \rangle \in R$ is represented by an edge leading from s to s' and labeled A. An example of a transition diagram is shown in Figure 1.

We consider a subset of the action description language $\mathcal{C}$ [6] that consists of two kinds of expressions (called *causal laws*): *static laws* of the form

$$\textbf{caused } L \textbf{ if } G, \tag{5}$$

where L is a fluent literal and G is a fluent formula and *dynamic laws* of the form

$$\textbf{caused } L \textbf{ if } G \textbf{ after } U, \tag{6}$$

where L and G are as above, and U is a formula; the part **if** G can be dropped if G is $True$. An *action description* is a set of causal laws. For instance, the action description consisting of the causal laws (1), (2), and

$$\begin{array}{l} \textbf{caused } \neg PowerON \textbf{ after } PushPB_{TV} \wedge PowerON \\ \textbf{caused } \neg TvON \textbf{ if } \neg PowerON \\ \textbf{inertial } PowerON, \neg PowerON, TvON, \neg TvON. \end{array} \tag{7}$$

encodes how a TV system operates; **inertial** $L_1, \ldots, L_k$ stands for **caused** L_i **if** L_i **after** L_i $(1 \leq i \leq k)$.

The meaning of an action description can be represented by a transition diagram. Let D be an action description with a signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$. The transition diagram $\langle S, V, R \rangle$ *described* by D is defined as follows: S is the set of all interpretations s of $\mathbf{F}$ such that, for every static law (5) in D, s satisfies $G \supset L$; $V(P, s) = s(P)$; and R is the set of all triples $\langle s, A, s' \rangle$ such that s' is the only interpretation of $\mathbf{F}$ which satisfies the heads L of all static laws (5) in D for which s' satisfies G, and dynamic laws (6) in D for which s' satisfies G and $s \cup A$ satisfies U. For instance, the transition diagram described by $\{(1), (2)\} \cup (7)$ is presented in Figure 1. Note that there is a unique transition diagram described by an action description. We say that an action description is *consistent* if its transition diagram is with nonempty state set.

3 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

$$\textbf{holds } F, \tag{8}$$

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

$$\textbf{necessarily } Q \textbf{ after } A_1; \ldots; A_n, \tag{9}$$

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

$$\textbf{SOMETIMES } Q, \tag{10}$$

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

$$\textbf{ALWAYS } Q, \tag{11}$$

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

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

$$s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n \tag{12}$$

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 (8) (resp. (9)) 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 query 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 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 (10) and $\exists s \in S\ (D, s \models Q)$; q is a universal query (11) and $\forall s \in S\ (D, s \models Q)$; q is of the form $\neg Q$ and $D \not\models Q$; q is of the form $Q \wedge Q'$, and $D \models Q$ and $D \models Q'$; or q is of the form $Q \vee Q'$, and $D \models Q$ or $D \models Q'$. 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. For instance, consider the action description consisting of (1), (2), and (7) encoding how a TV system operates; **inertial** $L_1, \ldots, L_k$ stands for **caused** L_i **if** L_i **after** L_i $(1 \leq i \leq k)$. This action description 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:

$$\textbf{ALWAYS holds } PowerON \equiv TvON,$$

$$\textbf{ALWAYS holds } PowerON \wedge TvON \supset \neg\textbf{necessarily } (\textbf{holds } TvON) \textbf{ after } \{PushPB_{TV}\}. \tag{13}$$

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

$$\textbf{evolves } F_0; A_1; F_1; \ldots; F_{n-1}; A_n; F_n, \tag{14}$$

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 an expression of the form

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

where each A_i is an action, stands for the dynamic query **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: **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 (12) such that, for each s_i of the history, the interpretation $P \mapsto V(P, s_i)$ satisfies some formula F_i can be expressed by the query:

$$\textbf{SOMETIMES evolves } F_0; A_1; F_1; \ldots; F_{n-1}; A_n; F_n. \tag{15}$$

For instance, query

$$\textbf{SOMETIMES evolves } PowerON; \{PushPB_{TV}\}; \neg PowerON; \{PushPB_{TV}\}; PowerON. \tag{16}$$

describes the presence of the following history in Figure 1:

$$\{PowerON, TvON\}, \{PushPB_{TV}\}, \{\neg PowerON, \neg TvON\}, \{PushPB_{TV}\}, \{PowerON, TvON\}. \tag{17}$$

Also we can express that there is no transition from any state where a formula F holds:

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

Like in [2], 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.

4 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 several weight assignments, each with a different motivation expresses some appeal of the action description.

4.1 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 [11]. 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, we can identify the highly preferred states relative to a given number l: a state with the weight greater than l is highly preferred. 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 Figure 1 described by D. Take, for each $s \in S$,

$$g(s) = \begin{cases} 2 & \text{if } PowerON \in s \\ 1 & \text{otherwise.} \end{cases} \tag{18}$$

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

4.2 Weighted queries

We can assign weights to queries to specify preferences over conditions they express. Based on such weighted queries, we can define the weight of an action description D as follows.

Let C be a set of queries, along with a weight function f mapping each condition 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. *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 (16) and

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

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

4.3 Weighted histories

In a transition diagram $T = \langle S, V, R\rangle$, we will say that a history (12) of length n is *desired* with respect to a given query (15), 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 (15) in C. Let us denote by $st(w)$ the starting state s_0 of a history w of form (12). 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:

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

The function u mapping a history w of form (12) 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 [11, 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:

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

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 (16), with weight 3. Consider the transition diagram $T = \langle S, V, R\rangle$ in Figure 1. Let us denote history (17) by w, and query (16) by c. Then H_C contains (w, c). Take $g(s)$ as in (18). Take $l = 1$. Suppose that, for each transition $\langle s, A, s'\rangle$ in R,

$$m(\langle s, A, s'\rangle) = \begin{cases} 0.5 & \text{if } s = \{PowerON, TvON\} \\ & \quad \wedge\, |A| = 1 \\ 1 & \text{otherwise.} \end{cases} \tag{20}$$

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 $weight_h(D) = 10.5$.

4.4 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 (15) 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:

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

where $\alpha = f'(q') + f'(q'')$; $\beta = f(q) \times \sum_{s \in S_D(B)} g'(s)$; $\gamma = f(q) \times [(\sum_{s \in S_D(B)} g'(s))/|S_D(B)|]$. 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, it describes how much and to what extent the given preferable queries are satisfied.

For instance, suppose that C consists of three queries:

$$\textbf{ALWAYS executable } \{PushPB_{TV}\}, \tag{21}$$

$$\textbf{SOMETIMES } \neg\textbf{executable } \{PushPB_{RC}, PushPB_{TV}\}, \tag{22}$$

and query (16), denoted by c_1, c_2 and c_3 respectively. Consider an action description D, with the transition diagram in Figure 1. Let us

denote history (17) by w; then $H_C = \{(w, c_3)\}$. Take the utility function g as (18), and the transition model m as (20). 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 $f'(c_1) = 4,\ f'(c_2) = 4, f'(c_2) = 10.5$. Therefore, $weight_{qs}(D) = 18.5$.

5 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 [2] 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 [2]. 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, the action description $D = (D_m, D_u)$, where $D_m = \{(1), (2)\}$ and D_u is (7), 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 $\neg TvON; \{PushPB_{TV}\}; \neg TvON$, (23)

and C_p consists of the queries (16), (22), (21), (19), (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 (Figure 1), except that there is no outgoing edge from the state $\{PowerON,\ TvON\}$ with the label $\{PushPB_{RC}\}$; thus only (3), (23), and (16) 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.

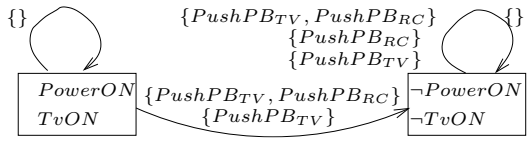

Figure 2. Transition diagram of $D^{(2)} = D_u \cup Q \cup \{(2)\}$.

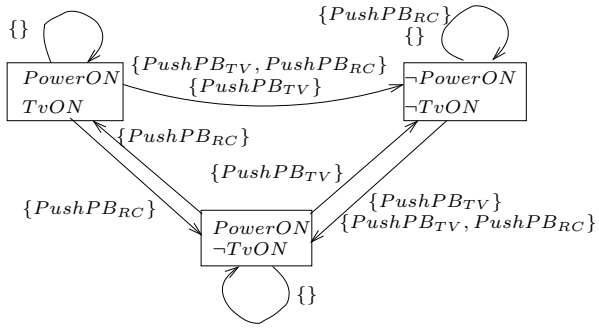

Figure 3. Transition diagram of $D^{(3)} = D_u \cup Q \cup \{(1)\}$.

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 [2], 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 (18) as the utility function g, (20) 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

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

whose utilities, $u(w_i) = u_0(w_i)$, can be computed as shown in Table 1. That is,

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

For $D^{(2)}$ (Figure 2), since $H_{C_p} = \emptyset$, $weight_h(D^{(2)}) = 0$.

Table 1. Utilities of histories in Example 1.

w	i	$u_i(w)$
w_0	1	$g(s_1) = 2$
w_0	0	$g(s_0) + m(\langle s_0, \{PushPB_{RC}\}, s_1\rangle) \times u_1(w_0) = 3$
w_1	1	$g(s_0) = 2$
w_1	0	$g(s_1) + m(\langle s_1, \{PushPB_{RC}\}, s_0\rangle) \times u_1(w_1) = 4$
w_2	2	$g(s_1) = 2$
w_2	1	$g(s_2) + m(\langle s_2, \{PushPB_{TV}\}, s_1\rangle) \times u_2(w_2) = 3$
w_2	0	$g(s_0) + m(\langle s_0, \{PushPB_{TV}\}, s_2\rangle) \times u_1(w_2) = 3.5$
w_3	2	$g(s_1) = 2$
w_3	1	$g(s_2) + m(\langle s_2, \{PushPB_{TV}\}, s_1\rangle) \times u_2(w_3) = 3$
w_3	0	$g(s_1) + m(\langle s_1, \{PushPB_{TV}\}, s_2\rangle) \times u_1(w_3) = 5$

For $D^{(3)}$ (Figure 3), since H_{C_p} contains (w_0, c_5), (w_1, c_5), (w_2, c_3), and (w_3, c_3),

$$weight_h(D^{(3)}) = u(w_0) \times f(c_5) + u(w_1) \times f(c_5) + u(w_2) \times f(c_3) + u(w_3) \times f(c_3) = 3 \times 2 + 4 \times 2 + 3.5 \times 4 + 5 \times 4 = 48.$$

Thus $D^{(3)}$ is the solution.

Example 2 Take D, Q, C_p, and $D^{(1)}$–$D^{(4)}$ as in Example 1, and C_m as the set consisting of the queries

$$\textbf{SOMETIMES} \neg \bigvee_{A \in 2^{\mathbf{A}}} \textbf{executable}\ A, \tag{24}$$

$$\textbf{ALWAYS} \neg \textbf{evolves}\ \neg TvON; \{PushPB_{TV}\}; \neg TvON, \tag{25}$$

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 [2] 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 (23), 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 $((D_u, D_m), Q, (\emptyset, C_m'), weight_q)$. This suggests relaxing the must query (24) (i.e., adding the query (24) to C_p with the weight 100) and solving the new ADU problem, $((D_u, D_m), Q, \{(25)\}, C_p \cup \{(24)\}, weight_q)$, for which the description $D_u \cup Q$ is the solution.

5.1 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 action description 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.

6 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 $g(s)$, $f(q)$, and $m(\langle s, A, s'\rangle)$ are computable in polynomial time. For a background on complexity, we refer to the literature (see e.g. [10]). [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 $D \models q$.

Table 2. 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

6.1 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:

Theorem 1 *Suppose that we are given an action description D, a set C of queries, a function g mapping every state a number, a function f mapping every query in C to a number, and a function m mapping every transition to a probability. Suppose that these functions are computable in polynomial time. Then the following hold:*

(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 (modulo some normalization) #P-complete, if the range of f and g are non-negative numbers, and GapP-complete for arbitrary f and g;
(iv) Computing $weight_{qs}(D)$ relative to C, f, g and m is FPSPACE-complete.

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

These results are also shown in the first row of Table 2. Here #P [10] 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 which accepts the input; problems polynomially solvable with an #P oracle are believed not to be PSPACE-hard. GapP [4, 8] is the closure of #P under subtraction (equivalently, it contains the functions which are expressible as number of accepting computation minus the number of rejecting computations of an NP Turing machine).

Informally, corresponding proof ideas for Theorem 1 can be sketched as follows:

ad (*i*). Computing $weight_s(D)$ amounts to counting the number of states s such that $g(s) > l$. This problem is thus easily seen to be in #P. Moreover, it is also #P-complete, since the canonical #P-complete problem #SAT of counting the models of a propositional formula is readily reduced to it.

ad (*ii*). As for $weight_q(D)$, we must evaluate each query $q{\in}C$ on D and then take a sum. As testing $D{\models}q$ is PSPACE-complete, computing $weight_q(D)$ is in FPSPACE, i.e., the class of functions computable in polynomial space. Moreover, the problem can also be shown to be hard for this class.

ad (*iii*). Computing $weight_h(D)$ modulo some normalization (which casts the problem to one with integer values), can like computing $weight_s(D)$ be seen to be in #P, if the functions $g(s)$ and $f(q)$ are non-negative. Indeed, each relevant history w can be nondeterministically generated in polynomial time, and $u(w)$ and $h(w)$ are easily computed from w; to account for $h(w)$, simply that many accepting computation branches are nondeterministically generated. On the other hand, #SAT is reducible to computing $weight_h(D)$.

We sketch here a simple reduction, which is as follows. Suppose that E is a SAT instance on propositional atoms $x_1, \ldots, x_n$, which without loss of generality is not satisfied if all atoms are assigned false. We let $x_1, \ldots, x_n$ be the fluents and a the single action symbol in an action description D, which consists of all statements

$$\textbf{caused} \quad x_i \ \textbf{if} \quad x_i \ \textbf{after} \ \top,$$
$$\textbf{caused} \ \neg x_i \ \textbf{if} \ \neg x_i \ \textbf{after} \ \top,$$

where $\top$ stands for a tautology, and let C consist of the single query

$$c = \textbf{evolves} \bigwedge_{i=1}^{n} \neg x_i; \emptyset; \top.$$

Informally, the transition diagram of D for the empty action $\emptyset$ the complete graph whose nodes are all truth assignments to $x_1, \ldots, x_n$, and c captures the transitions from the assignment in which all atoms are false to some arbitrary assignment via the empty action $\emptyset$. Now we define that $g(s) = 2^n$ if s satisfies E, and $g(s) = 0$ if s does not satisfy E, for each s. Furthermore, we define that transitions have uniform probability, i.e., $m(\langle s, A, s' \rangle) = 1/2^n$ for each transition $\langle s, A, s' \rangle$ in the transition diagram described by D. Let $f(c) = 1$.

It is easy to see that H_C contains all pairs (w, c) where $w = s_0, \emptyset, s_1$ such that s_0 is the state in which all x_i are false and s_1 is an arbitrary state. Furthermore, $h(w) = 1$ if s_1 satisfies E and $h(w) = 0$ otherwise. Therefore, $weight_h(D)$ is the number of satisfying assignments of E. Since D, C, m, f, and g are obviously constructible in polynomial time, and since moreover m, f, and g are computable in polynomial time, we obtain #P-hardness of computing $weight_h(D)$.

In case of arbitrary (possibly negative) $g(s)$ and $f(q)$, $weight_h(D)$ is computable as the difference of two #P functions. Therefore, computing $weight_h(D)$ is in the class GapP. Indeed, we have that

$$weight_h(D) = \sum_{(w,c) \in H_C^+} u(w){\times}f(c) \ - \sum_{(w,c) in H_C^-} -u(w){\times}f(c),$$

where H_C^+ contains all pairs (w, c) from H_C such that $u(w){\times}f(c)$ is positive and H_c^- contains all pairs (w, c) from H_C such that $u(w){\times}f(c)$ is non-negative. Both $\sum_{(w,c) \in H_C^+} u(w){\times}f(c)$ and $\sum_{(w,c) \in H_C^-} -u(w){\times}f(c)$ can be computed in #P. On the other hand, computing the difference $f_1 - f_2$ of two #P functions f_1 and f_2 can be polynomially reduced to computing $weight_h(D)$ for some action description D in polynomial time. More precisely, with a slight adaption of the above construction, we can reduce computing the difference of the number of satisfying assignments $\#(E_1)$ and $\#(E_2)$ of two SAT instances E_1 and E_2 on atoms $x_1, \ldots, x_n$, respectively, (which is GapP-hard) to computing $weight_h(D)$. For this, we assume without loss of generality that both E_1 and E_2 are not satisfied if all atoms x_i are false, and redefine $g(s)$ to

$$g(s) = \begin{cases} 2^n & \text{if } s \text{ satisfies } E_1 \wedge \neg E_2, \\ -2^n & \text{if } s \text{ satisfies } \neg E_1 \wedge E_2, \\ 0 & \text{otherwise.} \end{cases}$$

This has the effect that any history $w = s_0, A, s_1$ where $(w, c) \in C$, will contribute zero to $weight_h(D)$ if E_1 and E_2 have the same value for the assignment s_1, and contribute $h(w){\times}f(c) = 1$ (resp., $h(w){\times}f(c) = -1$) if E_1 is satisfied but not E_2 (resp. E_2 is satisfied but not E_1). In total, $weight_h(D)$ amounts then to $\#(E_1) - \#(E_2)$. As consequence, computing $weight_h(D)$ for general f and g is (modulo some normalization) complete for GapP.

ad (*iv*). Computing $weight_{qs}(D)$ is more involved than computing $weight_h(D)$. Here, we must take modified state rewards $g'(s)$ into account and normalize with $|S_D(B)|$ for certain queries. However, both values are computable in polynomial space, and thus also $f'(q)$ for each query q. Consequently, computing $weight_{qs}(D)$ is in FPSPACE; like computing $weight_q(D)$, it is also FPSPACE-complete.

In comparison, $weight_s(D)$ and $weight_h(D)$ are of the same computational degree of difficulty, while $weight_q(D)$ and $weight_{qs}(D)$ are harder under common complexity hypotheses. For queries where nesting of formulas is bounded by a constant, the complexity drops below FPSPACE.

6.2 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 each of the four weights can be calculated in polynomial time. In fact, not the whole transition diagram 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 [7]) 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 2) holds.

Theorem 2 *Suppose that we are given an action description D, the set S of states described by D, a set C of queries, a function g mapping every state in S to a number, a function f mapping every query*

in C to a number, and a function m mapping every transition to a probability. Suppose that these functions are computable in polynomial time. Then each weight function, $weight_s(D)$ (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), can be computed in polynomial time.

Obviously, computing $weight_s(D)$ on $T_C(D)$ is polynomial. Similarly, computing $weight_q(D)$ is polynomial since for each query q, testing $D \models q$ is polynomial on $T_C(D)$: label each state $s \in S$ bottom up with the subformulas q' of q that are true at s, and evaluate every dynamic query of form (9) by considering all reachable nodes at distance n.

For computing $weight_h(D)$, we can also exploit a labeling technique to avoid considering exponentially many paths in $T_C(D)$ explicitly. First, for a query q of form (15), we label all states s with p_i, $i \in \{0, \ldots, n\}$, such that $s = s_i$ for some history $w = s_0, A_1, s_1, \ldots, A_n, s_n$ satisfying q, in polynomial time. Here is a two pass procedure for labeling the states:

1) First label, for each state s, all states s' at distance $i = 0, 1, \ldots, n$ with r_i^s that respect the prefix of some w desired with respect to q such that $s = s_0$ and $s' = s_i$.
2) Then, going backwards from states labeled with r_s^n, turn each r_i^s ($i = n, n-1, \ldots, 0$) into p_i.

Now for $i = n, n-1, \ldots, 0$, we can for each state s labeled with p_i compute the sum of the utilities $u(w')$ of all suffixes $w' = s_i, A_{i+1}, s_{i+1}, \ldots, A_n, s_n$ of some history w satisfying c such that $s_i = s$, easily. In particular, $u_0^*(s)$ is the sum of all utilities $u(w)$ of histories that start at s and satisfy q. Exploiting this, $weight_h(D)$ is then readily computed by rearranging the sum of its definition: For each relevant query c of form (15), sum up the the $u_0^*(\cdot)$ values at all states and multiply the result with $f(c)$. This gives one summand of a sum to build over all relevant queries (i.e., queries of form (15)).

Example 3 Consider, for instance, the action description $D^{(3)}$ (Figure 3) in Example 1; take s_0 and s_1 as specified in Example 1. For query (4), in the first pass of the labeling process, state s_0 is labeled with $r_0^{s_0}$, $r_1^{s_1}$; and state s_1 is labeled with $r_0^{s_1}$, $r_1^{s_0}$; in the second pass, both states s_0 and s_1 are labeled with p_0 and p_1. Given the utility function and transition model as in Example 1 (i.e., as (18) and as (20), respectively), and assuming a weight of $f(c) = 3$ for the query, summing up we obtain:

$$\begin{aligned} u_1^*(s_0) &= g(s_0) = 2, \\ u_1^*(s_1) &= g(s_1) = 2, \\ u_0^*(s_0) &= g(s_0) + m(\langle s_0, \{PushPB_{RC}\}, s_1\rangle) \times u_1^*(s_1) = 3, \\ u_0^*(s_1) &= g(s_1) + m(\langle s_1, \{PushPB_{RC}\}, s_0\rangle) \times u_1^*(s_0) = 4. \end{aligned}$$

And in total

$$f(c) \times (u_0^*(s_0) + u_0^*(s_1)) = 21,$$

as the summand for the query, c, considered (and as the value for $weight_h(D^{(3)})$ as c is the only query considered in this example).

Using the same techniques as for $weight_h(D)$, we can compute $g'(s)$ for each state s in polynomial time on $T_C(D)$ and also $|S_D(B)|$. Therefore, also $weight_{qs}(D)$ is computable in polynomial time in this case.

Finally, if the state space S is not large, i.e., $|S|$ is polynomially bounded, S can be computed 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 $\mathrm{FP}_{\|}^{\mathrm{NP}}$. The following theorem summarizes these results (the third row of Table 2):

Theorem 3 *Suppose that we are given an action description D, the set S of states described by D, a set C of queries, a function g mapping every state in S to a number, a function f mapping every query in C to a number, and a function m mapping every transition to a probability. Suppose that $|S|$ is polynomially bounded, and the functions f, g, m are computable in polynomial time. Then computing each weight function, $weight_s(D)$ (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 $\mathrm{FP}_{\|}^{\mathrm{NP}}$.*

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 $p \in \{s, q, h, qs\}$.

7 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. 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 [2]. Further examples and applications are considered in an extended version of this paper [3].

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

Acknowledgments

This work is supported by the Austrian Science Fund (FWF) grant P16536-N04.

REFERENCES

[1] Eyal Amir, 'Towards a formalization of elaboration tolerance: Adding and deleting axioms', in *Frontiers of Belief Revision*, Kluwer, (2000).

[2] T. Eiter, E. Erdem, M. Fink, and J. Senko, 'Updating action domain descriptions', in *Proc. IJCAI*, pp. 418–423, (2005).

[3] T. Eiter, E. Erdem, M. Fink, and J. Senko, 'Comparing action descriptions based on semantic preferences'. Extended manuscript. Available at `http://www.kr.tuwien.ac.at/research/ad-cmp.pdf`, 2006.

[4] Stephen A. Fenner, Lance Fortnow, and Stuart A. Kurtz, 'Gap-definable counting classes', *Journal of Computer and System Sciences*, **48**(1), 116–148, (1994).

[5] M. Gelfond and V. Lifschitz, 'Action languages', *ETAI*, **3**, 195–210, (1998).

[6] E. Giunchiglia and V. Lifschitz, 'An action language based on causal explanation: Preliminary report', in *Proc. AAAI*, pp. 623–630, (1998).

[7] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner, 'Nonmonotonic causal theories', *AI*, **153**(1-2), 49–104, (2004).

[8] Sanjay Gupta, 'Closure properties and witness reduction', *Journal of Computer and System Sciences*, **50**(3), 412–432, (1995).

[9] John McCarthy, 'Elaboration tolerance', in *Proc. CommonSense*, (1998).

[10] C. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.

[11] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995.