An Extended Query Language for Action Languages (and its Application to Aggregates and Preferences)

James P. Delgrande
School of Computing Science
Simon Fraser University
Burnaby, B.C.
Canada V5A 1S6
jim@cs.sfu.ca

Torsten Schaub*
Institut für Informatik
Universität Potsdam
Postfach 90 03 27
D–14439 Potsdam, Germany
torsten@cs.uni-potsdam.de

Hans Tompits
Institut für Informationssysteme 184/3
Technische Universität Wien
Favoritenstraße 9–11
A–1040 Vienna, Austria
tompits@kr.tuwien.ac.at

Abstract

This paper continues our earlier work in representing arbitrary preferences in causal reasoning and planning systems, albeit in an oblique fashion. Previously, we defined a very general query language relative to histories; from this we specified a second language in which preferences on histories are defined. This in turn allowed us to define the notion of a most preferred history in a set of histories. In this paper, we extend these languages in two directions. First, we add a conditional construct that allows one to select between terms. Second, we add a capability for defining macros. With these two added constructs, one can now define aggregate quantities, such as the total cost of actions in a history, the maximum value of a fluent in a history, or a count of the number of times a fluent goes to zero in a history. Via the preference language, one can then express a preference for histories (or, plans) with minimum action cost, maximum value of a fluent, or in which a fluent is most often zero. We argue that this substantially increases the range of concepts about which one can express preferences.

Introduction

The traditional formulation of planning involves determining how a given goal can be attained, beginning from some initial state and by means of a given set of actions which alter the state of the world. A plan succeeds just when it is executable and attains the goal; otherwise, it fails. However, in realistic situations, things are not quite so simple. Thus, there may be requirements specifying that a plan should be as short as possible or that total cost, where costs are associated with actions, be minimised. As well, there may be *preferred* conditions, that are desirable to attain, but not necessary. Thus, in getting to the airport, the goal is to indeed arrive at the airport in good time; I may prefer to be able to pick up a coffee en route, but this preference is subordinate to the overall goal of getting to the airport. As well, there may be other preferences, such as preferring to take transit to driving, going by a particular route, etc. Each preference partitions the space of successful plans into those that satisfy the preference and those that do not. The goal of a planning problem now shifts to determining a *preferred* plan, in which a maximal set of preferences is satisfied, along with the goal. Such preferences also make sense outside of planning domains, and in fact apply to arbitrary sequences of temporal events. Hence, it is perfectly rational to prefer that one's favourite sport's team wins the championship, even though in the typical course of events one has no control over how the team performs.

In earlier work (Delgrande, Schaub, & Tompits 2004; 2005), we considered the problem of using general preferences over (fluent and action) formulas to determine preferences among temporal histories, or plans. While we focussed on histories as they are used in action description languages (Gelfond & Lifschitz 1998), our approach was, and is, applicable to any planning formalism. A history is defined as an interleaved sequence of states (of the world), and actions that take one state in a sequence to the next. We first specified a query language, $\mathcal{Q}_{\Sigma,n}$, (over some signature Σ and histories of maximum length n) in which one can determine whether an arbitrary expression is true in a given history. Given this language, we defined a *preference-specification language*, $\mathcal{P}_{\Sigma,n}$, that allows the definition of preference relations between histories. Via the language $\mathcal{P}_{\Sigma,n}$, we showed how to specify general preferences in temporal, causal, or planning frameworks. As well, the approach provided a very general language in which other "higher-level" constructs could be encoded, and in which other approaches could be expressed and so compared.

In the present paper, our motivating interest lies with being able to express preferences over aggregate quantities—that is, quantities that in some sense express a collective property of a set of fluent values. If actions come with a measure of their cost and duration, then two corresponding aggregate quantities of these measures would be the total action cost of a plan and the total duration of a plan. Clearly, in many cases these are quantities that one would want to minimise or, in a preference framework, prefer those plans with the minimal action cost or duration.

*Affiliated with the School of Computing Science at Simon Fraser University, Burnaby, Canada.

Such aggregates can be defined in our query language quite simply, given the addition of two capabilities. First, we extend the query language with a conditional term-forming operator. Syntactically, this operator is of the form $(\phi\ ?\ t_1 : t_2)$. This term denotes t_1 if ϕ is true; otherwise, it denotes t_2. Second, we add a "macro definition" capability, external to the language. That is, assertions in the query language can now contain macros that look exactly like fluents; given an appropriate set of macro definitions, these macros "compile out" so that one obtains assertions in the original language. Given this macro capability along with the conditional construct, one can define complex aggregate notions that can then take part in preference expressions. Thus, in our preference language we can now express a preference for histories with minimum action cost, maximum value of a fluent, or in which a fluent is zero at the most time points.

The next section briefly covers related work. This is followed by a section on the history-specific query language $\mathcal{R}_{\Sigma,n}$. This language extends our earlier query language $\mathcal{Q}_{\Sigma,n}$ with the new conditional operator. The following section describes adding a macro capability to this language and illustrates how aggregates can now be defined. The penultimate section describes the definition and use of our preference language $\mathcal{P}_{\Sigma,n}$ in this extended setting. The last section provides a brief discussion.

Background

Reasoning with preferences is an active area that is receiving increasing attention in AI. Hence, while the main topic of this paper is expressing aggregate quantities in our extended query language, our primary goal is the definition of a general language for specifying preferences on histories (or plans or other temporal sequences). Since histories and plans (implicitly or explicitly) involve sequences of states of the world, expressing aggregates and preferences over aggregates are of particular interest in planning systems.

In AI, Wellman & Doyle (1991) earlier suggested that the notion of *goal* is a relatively crude measure for planners to achieve, and instead that a relative preference over possible plan outcomes constitutes (or should constitute) a fundamental objective for planning. They show how to define goals in terms of preferences and, conversely, how to define (incompletely) preferences in terms of sets of goals. Here, we maintain a strict division between (hard) goals and (soft) preferences, although a framework along the lines of Wellman and Doyle is expressible in our method simply by taking the overarching goal as being $\top$—i.e., all valid histories are the subject of the preferences.

Myers & Lee (1999) assume that there is a set of desiderata, such as affordability or time, whereby successful plans can be ranked. A small number of plans is generated, where the intent is to generate divergent plans. The best plan is then chosen, based on a notion of Euclidean distance between these select attributes. Hence, they deal implicitly with aggregate quantities, with a concrete, quantitative notion of preference, applied to a small set of successful and presumably representative plans. In related work, Haddawy & Hanks (1992) use a utility function to guide a planner.

One approach to preferences in planning has been proposed by Son & Pontelli (2004), where a language for specifying preferences between histories is presented. This language is an extension of action language $\mathcal{B}$ (Gelfond & Lifschitz 1998), and is subsequently compiled into logic programs under the answer-set semantics. The notion of preference explored is based on so-called *desires* (what we call *absolute preferences* in previsous work (Delgrande, Schaub, & Tompits 2004)), expressed via formulas built by means of propositional as well as temporal connectives such as *always*, *until*, etc. From desires, preferences among histories are induced as follows: Given a desire ϕ, a history H is preferred to H' if $H \models \phi$ but $H' \not\models \phi$.

Similarly, Bienvenu & McIlraith (2005) address planning with preferences in the situation calculus. Preferences are founded on the notion of *basic desire formulas*, whose members are somewhat analogous to formulas in our language $\mathcal{Q}_{\Sigma,n}$. These formulas in turn are used in the composition of *atomic preference formulas* (essentially chains of preferences) and *general preference formulas*. As the authors note, this approach extends and modifies that of Son & Pontelli (2004) although expressed in terms of the situation calculus rather than an action language. Based on a concrete means of explicitly combining preferences, a best-first planner is given. Fritz & McIlraith (2005) employ a subset of this language in an approach to compile preferences into DT-Golog.

Eiter *et al.* (2003) describe planning in an answer-set programming framework where action costs are taken into account. The approach allows the specification of desiderata such as computing the shortest plan, the cheapest plan, or some combination of these criteria. This is achieved by employing weak constraints, which filter answer sets, and thus plans, based on *quantitative* criteria.

A General Query Language for Histories

Histories and Queries on Histories

In specifying histories, we extend the notation of Gelfond & Lifschitz (1998) in their discussion of *transition systems*. As described, virtually any general planning system (or indeed causal-reasoning formalism) could be used to provide a setting for our approach; as well, the approach is more broadly applicable than just to planning problems.

Definition 1 *An* action signature Σ *is a quadruple* $\langle D, F, V, A\rangle$, *where* D *is a set of* value names, F *is a set of* fluent names, $V : F \rightarrow 2^D \setminus \emptyset$ *assigns a* domain *to each fluent, and* A *is a set of* action names.

Σ *is* propositional *iff* $D = \{0, 1\}$*, and it is* finite *iff* D*,* F*, and* A *are finite. Moreover, a fluent name* $f \in F$ *is* propositional *iff* $V(f) = \{0, 1\}$.

For simplicity we will assume throughout that action signatures are finite and that D is the set of non-negative integers.

Definition 2 *Let* $\Sigma = \langle D, V, F, A\rangle$ *be an action signature.*

A history*,* H*, over* Σ *is a sequence*

$$(s_0, a_1, s_1, a_2, s_2, \ldots, s_{n-1}, a_n, s_n),$$

where $n \geq 0$*, and*

- *each* s_i*,* $0 \leq i \leq n$*, is a mapping assigning each fluent* $f \in F$ *a value* $v \in V(f)$*, and*
- $a_1, \ldots, a_n \in A$.

The functions $s_0, \ldots, s_n$ *are called* states*, and* n *is the* length *of history* H*, symbolically* $|H|$.

The states of a history may be thought of as possible worlds, and the actions take one possible world into another.

We need to be able to refer to fluent and action names in a history. Since a fluent's value will vary depending on the time point under consideration, we also need to be able to refer to time points and their relations. To this end, we define a query language on histories of maximum length n over an action signature Σ, named $\mathcal{R}_{\Sigma,n}$.

Definition 3 *Let* $\Sigma = \langle D, F, V, A\rangle$ *be an action signature and* $n \geq 0$ *a natural number.*

The alphabet of $\mathcal{R}_{\Sigma,n}$ *consists of the following items:*

1. *a set* $\mathcal{V} = \{i, j, \ldots\}$ *of* time-stamp variables*, or simply* variables*,*
2. *the set of integers,*
3. *the sets* D*,* F*, and* A*,*
4. *the sentential connectives '*$\neg$*' and '*$\supset$*', and the quantifier symbol '*$\exists$*',*
5. *the arithmetic function symbols '*$+$*', '*$-$*', and '*$\cdot$*', the arithmetic relation symbol '*$<$*', and the equality symbol '*$=$*', and*
6. *the parentheses '(' and ')', and the symbols '?' and ':'.*

Terms in $\mathcal{R}_{\Sigma,n}$ are of two types: those denoting time points and those denoting fluent values.

Definition 4 *Let* $\Sigma = \langle D, F, V, A\rangle$ *be an action signature and* $n \geq 0$ *a natural number.*

The terms *of* $\mathcal{R}_{\Sigma,n}$ *are as follows:*

1. *A* time term *is an arithmetic term recursively built from* $\mathcal{V} \cup \{0, \ldots, n\}$*, employing* $+$ *and* $\cdot$ *(as well as parentheses) in the usual manner.*
2. *A* (fluent) value term *is either a member of* D*, an expression of the form* $f(t)$*, where* $f \in F$ *and* t *is a time term, or an arithmetic term recursively built from value terms, employing* $+$*,* $-$*, and* $\cdot$ *in the usual manner. Additionally, if* ϕ *is a formula (cf. Definition 5) and* e_1 *and* e_2 *are value terms, then* $\phi \ ? \ e_1 : e_2$ *is a value term.*

The intent here is that, in $\mathcal{R}_{\Sigma,n}$, time terms range over the numbers $0, \ldots, n$ only, while fluent value terms may denote arbitrarily large integers. So, our definition of the class of formulas of $\mathcal{R}_{\Sigma,n}$, given next, will accommodate that, for instance, $f(0) = 5$ is well formed, where f is a fluent name, with intended interpretation that f has value 5 at time point 0, whereas $\exists i f(0) = i$ is not well formed.

Definition 5 *Let* $\Sigma = \langle D, F, V, A\rangle$ *be an action signature and* $n \geq 0$ *a natural number.*

The formulas *of* $\mathcal{R}_{\Sigma,n}$ *are as follows:*

1. *A* time atom *is an expression of the form* $(t_1 < t_2)$ *or* $(t_1 = t_2)$*, where* t_1, t_2 *are time terms.*
   A value atom *is either an expression of form* $a(t)$*, where* $a \in A$ *and* t *is a time term, or an expression of the form* $(v_1 < v_2)$ *or* $(v_1 = v_2)$*, where* v_1, v_2 *are value terms.*
   An atom containing no variables is ground*.*
2. *A* literal *is an atom possibly preceded by the negation sign* $\neg$*.*
3. *A* formula *is a Boolean combination of atoms, along with quantifier expressions of form* $\exists v$*, for* $v \in \mathcal{V}$*, formed in the usual recursive fashion.*
4. *A* query *is a closed formula, i.e., with no free time-stamp variables.*

For a propositional fluent f and time term e, we write $f(e)$ for $f(e) = 1$ and $\neg f(e)$ for $f(e) = 0$; in such a case, $f(e)$ is said to be *true* or *false* (or equivalently *true at* e or *false at* e), respectively.

We define the operators $\wedge$, $\vee$, and $\leq$, and the universal quantifier $\forall$, in the usual way. Parentheses may be dropped in formulas if no ambiguity arises, and we may write quantified formulas like $\mathsf{Q}v_1\mathsf{Q}v_2\,\alpha$ as $\mathsf{Q}v_1, v_2\,\alpha$, for $\mathsf{Q} \in \{\forall, \exists\}$. For formula α, variables $v_1, \ldots, v_k$, and numbers $i_1, \ldots, i_k$, $\alpha[v_1/i_1, \ldots, v_k/i_k]$ is the result of uniformly substituting v_j by i_j in α, for each $j \in \{1, \ldots, k\}$. Thus, if $v_1, \ldots, v_k$ are the free variables in α, then $\alpha[v_1/i_1, \ldots, v_k/i_k]$ is a closed formula.

Variables range over time points, and so quantification applies to time points only. Atoms consist of actions or fluents indexed by a time point, or of a predicate on arithmetic (time point) expressions. Atoms are used to compose formulas in the standard fashion, and queries consist of closed formulas. This means that we remain within the realm of propositional logic, since quantified expressions $\forall v$ and $\exists v$ can be replaced by the conjunction or disjunction (respectively) of their instances.

Example 1 *Let* $pickup \in A$*,* $red \in F$*, and* $i, j \in \mathcal{V}$*. Then,*

$$pickup(4),\ red(i + j),\ i < j + 2$$

are atoms. As well,

$$red(j) \wedge (\forall k\ (k < j) \supset \neg red(k)),$$
$$ageJohn(t) > ageMary(t + 1)$$

are formulas but not queries, and

$$\exists i, j((i + 2 < j) \wedge pickup(i) \wedge \neg red(j)),$$
$$\exists i(ageJohn(i) > ageMary(i + 1)),$$
$$\exists i(\neg(ageJohn(i) = 35)),$$
$$(((0 = 1)?1 : 0) = 0)$$

are closed formulas and so queries (*assuming an appropriate action signature*).

The intent of the first formulas above is that it be true in a history in which $pickup$ is true at some time point and three or more time points later red is false. For the last formula above, we have that $0 = 1$ is false, therefore the value of $((0 = 1)?1 : 0)$ is 0, and since $0 = 0$ is true, the formula is true.

The following operators, which basically correspond to similar ones well-known from linear temporal logic (LTL), can be defined:

- $\Box b = \forall i\, b(i)$;
- $\Diamond b = \exists i\, b(i)$; and
- $(b \,\mathsf{U}\, g) = \exists i \left(g(i) \wedge \forall j((j < i) \supset b(j))\right)$.

Here, b and g are propositional fluent names or action names. Informally, $\Box b$ expresses that b *always* holds, $\Diamond b$ that b holds *eventually*, and $b \,\mathsf{U}\, g$ that b holds continually *until* g holds. Other LTL operators are likewise expressible.

Semantics of Queries

The definition of truth of a query with respect to a history is done in two parts. First, we define an interpretation function $\mathcal{I}$ that gives the denotation of terms; from this we define the notion of truth. To ease detail, we use the notation that for a ground term t, $val(t)$ is the value of t according to standard integer arithmetic. Note that although the next two definitions are interdependent, these definitions of denotation and truth are strictly compositional.

Definition 6 *Given query language $\mathcal{R}_{\Sigma,n}$ and a history $H = (s_0, a_1, s_1, \ldots, a_k, s_k)$ over Σ of length $k \leq n$, the* denotation $\mathcal{I}_H$ *is given by:*

1. *If t is a ground time term, then:*

$$\mathcal{I}_H(t) = \begin{cases} 0 & \text{if } val(t) < 0; \\ k & \text{if } val(t) > k; \\ val(t) & \text{otherwise}. \end{cases}$$

2. *If t is a ground fluent value term, then:*
   (a) *if t is $f(t')$, for $f \in F$, then $\mathcal{I}_H(t) = s_i(f)$, where $i = \mathcal{I}_H(t')$;*
   (b) *if t is $\phi\ ?\ e_1 : e_2$, then:*
   if $H \models_{\mathcal{R}_{\Sigma,n}} \phi$, then $\mathcal{I}_H(t) = \mathcal{I}_H(e_1)$, otherwise $\mathcal{I}_H(t) = \mathcal{I}_H(e_2)$;
   (c) *otherwise, $\mathcal{I}_H(t) = val(t)$.*

The rationale behind Part 1 in the definition above is to allow that, while a time term calculation may refer to a time point greater than the length of a history or less than time point 0, its denotation should not. Hence, if $val(t)$ is greater than the length k of history H, then a ground atomic query $\phi(t)$ will be satisfied by H if it is satisfied at the last state of H.

It can be observed for a ground time term t, that $0 \leq \mathcal{I}_H(t) \leq n$, while for a value term we do not necessarily obtain that $\mathcal{I}_H(t) \in D$. This conforms to intuitions: fluents have value only within a history, and time points need to refer to times within a history. On the other hand, value terms are used (among other things) to determine aggregate quantities. Thus, we could have a propositional domain with $D = \{0, 1\}$; however, if we wanted to count the number of times that a light was on, say, we would need other integer values.

Now we can define what it means for a history to satisfy a query expressed in $\mathcal{R}_{\Sigma,n}$.

Definition 7 *Let $H = (s_0, a_1, s_1, \ldots, a_k, s_k)$ be a history over Σ of length $k \leq n$, and let Q be a query of $\mathcal{R}_{\Sigma,n}$.*

We define $H \models_{\mathcal{R}_{\Sigma,n}} Q$ as follows:

1. *If $Q = a(t)$ is a ground action atom, then $H \models_{\mathcal{R}_{\Sigma,n}} Q$ iff $a = a_{\mathcal{I}_H(t)}$.*
2. *If $Q = (v_1 \circ v_2)$, for $\circ \in \{<, =\}$, is a ground (fluent or time) atom, then $H \models_{\mathcal{R}_{\Sigma,n}} Q$ iff $(\mathcal{I}_H(v_1) \circ \mathcal{I}_H(v_2))$.*
3. *If $Q = \neg\alpha$, then $H \models_{\mathcal{R}_{\Sigma,n}} Q$ iff $H \not\models_{\mathcal{R}_{\Sigma,n}} \alpha$.*
4. *If $Q = \alpha \supset \beta$, then $H \models_{\mathcal{R}_{\Sigma,n}} Q$ iff $H \not\models_{\mathcal{R}_{\Sigma,n}} \alpha$ or $H \models_{\mathcal{R}_{\Sigma,n}} \beta$.*
5. *If $Q = \exists v\alpha$, then $H \models_{\mathcal{R}_{\Sigma,n}} Q$ iff, for some $0 \leq i \leq n$, $H \models_{\mathcal{R}_{\Sigma,n}} \alpha[v/i]$.*

If $H \models_{\mathcal{R}_{\Sigma,n}} Q$ holds, then H *satisfies* Q. For simplicity, if $\mathcal{R}_{\Sigma,n}$ is unambiguously fixed, we also write $\models$ instead of $\models_{\mathcal{R}_{\Sigma,n}}$.

We have the following results concerning complexity in $\mathcal{R}_{\Sigma,n}$.

Theorem 1 *Let Σ be an action signature and $n \geq 0$ a natural number.*

1. *Given a history $H = (s_0, a_1, s_1, \ldots, a_k, s_n)$ over Σ of length n and a query*

$$Q = (\mathsf{Q}_1 i_1)(\mathsf{Q}_2 i_2) \ldots (\mathsf{Q}_m i_m)\ C$$

   of $\mathcal{R}_{\Sigma,n}$, where $\mathsf{Q}_i \in \{\forall, \exists\}$, $1 \leq i \leq m$, and C contains no quantifiers, then deciding whether $H \models_{\mathcal{R}_{\Sigma,n}} Q$ holds can be determined in $O(|C|^m)$ time.
2. *Given a query Q of $\mathcal{R}_{\Sigma,n}$, then deciding whether there is a history H over Σ of length n such that $H \models_{\mathcal{R}_{\Sigma,n}} Q$ holds is PSPACE-complete.*

The proof of the first part is straightforward, since for each quantifier expression $(\mathsf{Q}i)\alpha$, one needs to test the n substitution instances of α conjunctively (for universal quantification) or disjunctively (for existential quantification). For the second part, for showing that the

problem is at least in PSPACE, the reduction is from satisfiability of quantified Boolean formulas to formulas of $\mathcal{R}_{\Sigma,1}$; for showing that it is no worse than PSPACE, the reduction is from formulas of $\mathcal{R}_{\Sigma,n}$ to formulas of linear-time temporal logic (LTL).

The language $\mathcal{R}_{\Sigma,n}$ is quite expressive. Indeed, it can be shown that $\mathcal{R}_{\Sigma,n}$ subsumes the languages $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{R}$ due to Gelfond & Lifschitz (1998), as well as our earlier language $\mathcal{Q}_{\Sigma,n}$ (Delgrande, Schaub, & Tompits 2005), with respect to expressivity.

Macros

In realistic applications, one is often faced with non-propositional fluents, such as temperature, amount of rain, etc. Moreover, actions often have associated costs and other measures, such as duration. Frequently too, one is interested in aggregates of such quantities—for example, determining the total rainfall, total cost of actions, maximum value attained over an interval, and so on. In the next section, we will see how these types of quantities permit preference statements expressing certain optimisations or desiderata. In this section, we show how, using our conditional construct (_ ? _ : _) along with the addition of a macro capability, we can express aggregates of fluent values without adding to the overall complexity of the language.

Definition of Macros

Macros are defined as follows:

Definition 8 *Let $\Sigma = \langle D, F, V, A\rangle$ be an action signature and $n > 0$. Then, the alphabet of a macro language over Σ and n consists of the alphabet of $\mathcal{R}_{\Sigma,n}$, together with*

1. *a set of fluent names F', where $F \cap F' = \emptyset$, and*
2. *a set of parameter names $\mathbf{P} = \{\$1, \$2, \dots\}$.*

We refer to $\langle \Sigma, F'\rangle$ as the (macro) signature *for such a macro language.*

The fluent names in F' serve as macro names, while the elements of $\mathbf{P}$[1] serve as parameters for the macros, and for which terms will be substituted in the macro expansion.

Definition 9 *Given a macro signature $\langle \Sigma, F'\rangle$ and $n > 0$, a macro M is a sequence*

$$(\langle \nu_1, \mu_1\rangle, \dots, \langle \nu_m, \mu_m\rangle),$$

where

1. *each ν_i is of the form $f(t)$, where $f \in F'$, and either $t \in \{0, \dots, n\}$ or t is the parameter name $\$i$, and*
2. *each μ_i is a value term over $\langle D, F \cup F', V, A\rangle$, but with added primitive terms $\mathbf{P}$, formed in the expected recursive manner.*

[1] In the present paper we use only parameter $1, since we deal just with unary macros. The inclusion of $\mathbf{P}$ anticipates a possible generalization to many-placed macros (which in turn makes conceptual sense only if the language $\mathcal{R}_{\Sigma,n}$ is generalized to allow flunets of arity greater than 1).

In a formula involving macros, $\mu_i[x/t]$ will be the formula μ_i with every occurrence of parameter x replaced by term t.

Definition 10 *Let ϕ be a formula and*

$$M = (\langle \nu_1, \mu_1\rangle, \dots, \langle \nu_m, \mu_m\rangle)$$

a macro.

- ν_i matches *value term t in ϕ, if*
  - $\nu_i = f(a)$,
  - $t \in D$ *or* $t = f(b)$ *and if a is a constant, then $a = b$.*

  Matches will be understood to be symmetric.
- *$\phi[\nu_i/\mu_i]$ is defined by:*

  for every $t = f(b)$ in ϕ that matches $\nu_i = f(a)$, replace t in ϕ by $\mu_i[\$1/b]$.
- *ϕ^M, the macro expansion of ϕ by M, is defined by: $((\dots((\phi[\nu_1/\mu_1])[\nu_2/\mu_2])\dots)[\nu_k, \mu_k])^*$, where θ^* denotes iteration of the implicit operators to a fixed point.*

  The macro expansion of a set of macros is defined in the obvious manner.

Note that the definition of a macro expansion ensures, for a macro $(\langle \nu_1, \mu_1\rangle, \dots, \langle \nu_m, \mu_m\rangle)$, that for $i < j$, ν_i will be expanded before ν_j. Of course, the macro expansion is not always defined; for example, the macro $(\langle f(\$1), f(\$1)\rangle)$ can lead to problems. Rather, with appropriately defined macros, one now may express aggregate functions.

Example 2 *The following macro can be used to find the maximum value that fluent f takes on in a history of length n:*

$$\begin{array}{l}(\langle \mathit{maxf},\ \mathit{maxfh}(n)\rangle,\\ \ \langle \mathit{maxfh}(0),\ f(0)\rangle,\\ \ \langle \mathit{maxfh}(\$1),\ ((f(\$1) > \mathit{maxfh}(\$1-1))\ ?\\ \qquad f(\$1) : \mathit{maxfh}(\$1-1))\rangle).\end{array}$$

So, for example, in a history of length 2, the macro $maxf$ simply stands for the following expression (highlighting the structure by underlining):

$$(f(2)\ >\ \underline{(f(1) > \underline{f(0)}\ ?\ f(1) : \underline{f(0)})}\ ?\ \underline{f(2) : (f(1) > \underline{f(0)}\ ?\ f(1) : \underline{f(0)})}).$$

That is, neither $maxf$ nor its "helper macro" $maxfh$ appear in the actual expression. Last, given the associated mappings $s_0 : f \mapsto 2$, $s_1 : f \mapsto 3$, and $s_2 : f \mapsto 1$, the expression in Example 1 evaluates to 3.

It is important to note that aggregates like $\mathit{maxfh}(i)$ are merely macros representing nested value terms. Thus, in particular, they are not fluents nor are they terms in the language $\mathcal{R}_{\Sigma,n}$.

Example 3 *The following macro sums the values of f in a history:*

$$(\langle sumf,\ sumfh(n)\rangle,\\ \langle sumfh(0),\ f(0)\rangle,\\ \langle sumfh(\$1),\ f(\$1) + sumfh(\$1-1)\rangle).$$

Example 4 *A similar definition as in the previous example can be given for counting all occurrences of (propositional fluent) f being true:*

$$(\langle cntf,\ cntfh(n)\rangle,\\ \langle cntfh(0),\ f(0)\ ?\ 1 : 0\rangle,\\ \langle cntfh(\$1),\ cntfh(\$1-1) + (f(\$1)\ ?\ 1 : 0)\rangle).$$

Further refinements are easily specified, as illustrated next.

Example 5 *Preferring histories with the globally minimum number of days (states) on which it rained more than t litres can be modelled by an extension of the count macro:*

$$(\langle cntf,\ cntfh(n)\rangle,\\ \langle cntfh(0),\ (f(0) \leq t)\ ?\ 0 : 1\rangle,\\ \langle cntfh(\$1),\ cntfh(\$1-1) + ((f(\$1) \leq t)\ ?\ 0 : 1)\rangle).$$

For modelling action costs, we associate with each action a fluent yielding the cost of the corresponding action and then sum the fluent. This is simple, and moreover allows us to associate with an action other measures, such as duration.

Correctness of Macros

The correctness of a macro (that is to say, the macro does what it is supposed to do) can be determined informally by inspection, or by an inductive argument. Another approach is to define a term corresponding to a macro as an extension to the language $\mathcal{R}_{\Sigma,n}$, and then prove that this new term corresponds to the value computed by the macro. This as well gives a means for specifying the semantics of a macro (after a fashion) and also the semantics of aggregates. We illustrate with two examples.

To begin with, consider where we wish to define a fluent that will correspond to the maximum value of some other fluent obtained so far in a history. Specifically, for fluent f, we want to define a fluent *fmaxfh*, where

$$fmaxfh(i) = \max_{0 \leq j \leq i} f(j).$$

We can do this by extending Definition 6 as follows:

$$\mathcal{I}_H(fmaxfh(i)) = \\ \begin{cases} \mathcal{I}_H(f(0)), & \text{if } i = 0; \\ \mathcal{I}_H(f(i) > fmaxfh(i-1)\ ? \\ \quad f(i) : fmaxfh(i-1)), & \text{otherwise.} \end{cases}$$

We can now define fluent *fmaxf* to correspond to *fmaxfh*(n) – that is, extend the interpretation function so that $\mathcal{I}_H(fmaxf) = \mathcal{I}_H(fmaxfh(n))$. It is now a straightforward, albeit tedious, task to show that any formula ϕ that mentions fluent *fmaxf* has precisely the same truth value in any history as does the macro expansion of macro *maxf* in the formula ϕ with all occurrences of *fmaxf* replaced by *maxf*.

Similarly, we can define a fluent $fsumf$ that will correspond to the sum of the values of fluent f obtained so far in a history; i.e., we can define

$$fsumf(i) = \sum_{0 \leq j \leq i} f(j).$$

We do this by extending Definition 6 as follows:

$$\mathcal{I}_H(fsumf(i)) = \\ \begin{cases} \mathcal{I}_H(f(0)), & \text{if } i = 0; \\ \mathcal{I}_H(f(i) + fsumf(i-1)), & \text{otherwise.} \end{cases} \quad (1)$$

Again, we can show that the truth value of formulas mentioning *fsumf* will correspond to the macro expansion of corresponding formulas mentioning macro *sumf*.

The overall structure of these correspondences is clear. In the case of macros, we typically have a recursive expansion based on time points, and that terminates at time point 0 or n. In the case of the interpretation function $\mathcal{I}_H$ (relative to history H), this recursive expansion is mirrored in a recursive definition of the value of intermediate fluents. Thus, in this case, the value of a defined fluent (such as *fmaxf*) can be determined from the underlying fluent (viz. f) by a process akin to macro expansion.

The overall scheme can be obviously extended to more than one fluent, and more than a single time point for each step. For example, we can define a fluent ex that counts the number of times the value of fluent f exceeds that of g two time points ago, as follows:

$$\mathcal{I}_H(ex(i)) = \\ \begin{cases} 0, & \text{if } i = 0 \text{ or } i = 1 \\ \mathcal{I}_H(f(i) > g(i-2)\ ? \\ \quad ex(1-i) + 1 : ex(1-i))\ , & \text{otherwise.} \end{cases}$$

From this, a corresponding macro can straightforwardly be defined. This in turn suggests a methodology for constructing macros: First, give a mathematical definition for the desired mathematical concept, for example, the sum of fluent f is given by $\max_{0 \leq j \leq n} f(j)$. This can then be defined within our language $\mathcal{R}_{\Sigma,n}$, as done in (1), and then essentially discharged from the language by the macro definition in Example 3.

Application: Expressing Preferences on Histories

In this section, we sketch a major application of our language, and in particular of the conditional construct and macros, to expressing preferences between histories.

A Preference Language

We briefly summarise our earlier work on preferences and a preference language; for formal details see (Delgrande, Schaub, & Tompits 2005). The central notion of this approach is of a *preference frame*, consisting of a pair $(\boldsymbol{H}, \boldsymbol{P})$, where

- $\boldsymbol{H}$ is a set of *histories* and
- $\boldsymbol{P}$ is a set of *preferences* on histories.

Histories are as in the preceding sections. A preference specifies an individual criterion for distinguishing among histories. A preference defines a binary relation, consisting of pairs of histories where one history is preferred to the other, according to the preference. We define a preference among two histories directly in terms of a formula ϕ; this formula is in a language $\mathcal{P}_{\Sigma,n}$ that extends $\mathcal{Q}_{\Sigma,n}$ so that one can now also refer to histories, in the following sense. We define that H_h is not less preferred than H_l, written $H_l \preceq_\phi H_h$, just if $\langle H_l, H_h\rangle \models \phi$, where ϕ expresses a preference condition between these two histories in $\mathcal{P}_{\Sigma,n}$. $H_l \preceq_\phi H_h$ holds if ϕ is true by evaluating it with respect to H_l and H_h. This requires that we are able to refer to fluent and action names at time points and in histories. Preferences are expressed by means of a formula composed of

- Boolean combinations of fluents and actions indexed by time points and by a history, and
- quantifications over time points.

Indexing with respect to time points and histories is achieved via *labelled atoms* of form $\ell : b(i)$. Here, ℓ is a *label*, either $\boldsymbol{l}$ or $\boldsymbol{h}$, referring to a history which is considered to be lower or higher ranked, respectively, b is an action or fluent name, and i is a time point. Semantically, $\langle H_l, H_h\rangle \models \boldsymbol{l} : b(i)$ holds if b holds at time point i in history H_l; and analogously for $\langle H_l, H_h\rangle \models \boldsymbol{h} : b(i)$. This is extended to labelled formulas in the expected fashion.

For example, we can express that history H_h is preferred to history H_l if fluent f is true at some point in H_h but never true in H_l by the formula

$$\phi = (\boldsymbol{h} : \exists i f(i)) \ \wedge \ (\boldsymbol{l} : \forall i \neg f(i)), \tag{2}$$

providing $\langle H_l, H_h\rangle \models \phi$ holds.

Each preference $\phi \in \boldsymbol{P}$ induces a binary relation $\preceq_\phi$ on $\boldsymbol{H}$. Depending on the type of preference encoded in $\boldsymbol{P}$, one would supply a strategy from which a maximally preferred history is selected. Thus for preferences only of the form (2), indicating which fluents are desirable, the maximally preferred history might be the one which was ranked as "preferred" by the greatest number of preferences in $\boldsymbol{P}$.

Expressing Preferences among Histories

We define a preference among two histories, H_l and H_h, directly in terms of a formula ϕ:

$$H_l \preceq_\phi H_h \qquad \text{iff} \qquad \langle H_l, H_h\rangle \models \phi. \tag{3}$$

The intent with $\langle H_l, H_h\rangle \models \phi$ is that ϕ expresses a condition in which H_h is at least as preferred as H_l. This requires that we be able to talk about the truth values of fluents and actions in H_l and H_h. Using our query language on histories, $\mathcal{R}_{\Sigma,n}$, and the notion of truth in a history for a query, we can define a *preference-specification language*, enabling the definition of preference relations between histories, as in (3).

Definition 11 *Let $\Sigma = \langle D, F, V, A\rangle$ be an action signature and $n \geq 0$ a natural number.*

We define the preference-specification language $\mathcal{P}_{\Sigma,n}$ *over $\mathcal{R}_{\Sigma,n}$ as follows:*

1. *The alphabet of $\mathcal{P}_{\Sigma,n}$ consists of the alphabet of the query language $\mathcal{R}_{\Sigma,n}$, together with the symbols $\boldsymbol{l}$ and $\boldsymbol{h}$, called* history labels*, or simply* labels*.*
2. *Atoms of $\mathcal{P}_{\Sigma,n}$ are either time atoms of $\mathcal{R}_{\Sigma,n}$ or expressions of the form $\ell : q$, where $\ell \in \{\boldsymbol{l}, \boldsymbol{h}\}$ is a label and q is an action or value atom of $\mathcal{R}_{\Sigma,n}$.*
   Atoms of the form $\ell : p$ are also called labelled atoms*, with ℓ being the label* of $\ell : p$*. We call $\ell : p$* ground *iff p is ground.*
3. *Formulas of $\mathcal{P}_{\Sigma,n}$ are built from atoms, as introduced above, in a similar fashion as formulas of $\mathcal{R}_{\Sigma,n}$. We call formulas of $\mathcal{P}_{\Sigma,n}$ also* preference formulas*.*
4. *A* preference axiom*, or simply* axiom*, is a closed preference formula, i.e., containing no free time-stamp variables.*

For a formula α of $\mathcal{R}_{\Sigma,n}$ and a history label $\ell \in \{\boldsymbol{l}, \boldsymbol{h}\}$, by $\ell : \alpha$ we understand the formula resulting from α by replacing each action or value atom $b(t)$ of α by the labelled atom $\ell : b(t)$. Informally, a labelled atom $\ell : p$ expresses that p holds in a history associated with label ℓ. This is made precise as follows.

Definition 12 *Let Σ be an action signature and $n \geq 0$. Let ϕ be a preference axiom of $\mathcal{P}_{\Sigma,n}$ and H_l, H_h histories over Σ with $|H_i| \leq n$, for $i = l, h$.*

The relation $\langle H_l, H_h\rangle \models_{\mathcal{P}_{\Sigma,n}} \phi$ is recursively defined as follows:

1. *If $\phi = \ell : p$ is a ground labelled atom, for $\ell \in \{\boldsymbol{l}, \boldsymbol{h}\}$, then $\langle H_l, H_h\rangle \models_{\mathcal{P}_{\Sigma,n}} \phi$ iff*
   (a) *$H_l \models_{\mathcal{R}_{\Sigma,n}} p$, for $\ell = \boldsymbol{l}$, and*
   (b) *$H_h \models_{\mathcal{R}_{\Sigma,n}} p$, for $\ell = \boldsymbol{h}$.*
2. *Otherwise, $\langle H_l, H_h\rangle \models_{\mathcal{P}_{\Sigma,n}} \phi$ is defined analogously as the truth conditions for $\models_{\mathcal{R}_{\Sigma,n}}$.*

If $\langle H_l, H_h\rangle \models_{\mathcal{P}_{\Sigma,n}} \phi$ holds, then $\langle H_l, H_h\rangle$ is said to *satisfy* ϕ. If Σ and n are clear from the context, we may simply write $\models$ instead of $\models_{\mathcal{P}_{\Sigma,n}}$.

Definition 13 *Let ϕ be a preference axiom of $\mathcal{P}_{\Sigma,n}$. For histories H_l, H_h over Σ of maximum length n, we define*

$$H_l \preceq_\phi H_h \quad \textit{iff} \quad \langle H_l, H_h\rangle \models_{\mathcal{P}_{\Sigma,n}} \phi.$$

Note that the employment of the symbol $\preceq_\phi$ is purely suggestive at this point, since $\preceq_\phi$ may have none of the properties of an ordering.

We give some illustrations next.

Example 6 *The formula*

$$\big(\boldsymbol{h} : (\exists i f_1(i) \wedge \forall i \neg f_2(i))\big) \wedge \big(\boldsymbol{l} : (\exists i f_2(i) \wedge \forall i \neg f_1(i))\big)$$

expresses a preference of f_1 over f_2 in the sense that, for all histories H_l, H_h, we prefer H_h over H_l whenever it holds that H_h satisfies f_1 but not f_2, whilst H_l satisfies f_2 but not f_1.

Given this, we can now state a preference for histories where the fluent f is maximum (as defined in Example 2). In place of macro $maxf$, we employ two macros, $h{:}maxf$ and $l{:}maxf$.

In the former case, we have the macro definition:[2]

$$\begin{array}{l}(\langle h{:}maxf,\ h{:}maxfh(n)\rangle,\\ \ \langle h{:}maxfh(0),\ \boldsymbol{h} : f(0)\rangle,\\ \ \langle h{:}maxfh(\$1),\\ \quad ((\boldsymbol{h} : f(\$1) > h{:}maxfh(\$1-1))\ ?\\ \quad\quad (\boldsymbol{h} : f(\$1)) : (h{:}maxfh(\$1-1)))).\end{array}$$

The macro $l{:}maxf$ is defined analogously. This enables the expression of the preference:

$$h{:}maxf \leq l{:}maxf. \tag{4}$$

Similarly, we can state a preference for histories where the values for f are maximum over every subinterval $[0, i]$ where $0 \leq i \leq n$ by:

$$\forall i(h{:}maxfh(i) \leq l{:}maxfh(i)).$$

Since our designated fluent f could in fact be measuring a quantity such as action cost or action duration, the preceding examples show how one can express preferences for plans (assuming that the histories are produced by a planner) that maximize certain quality measures.

As a last example, we can express a preference for the range of values of f being maximal in one of (at least) three ways. This can be directly expressed, given our macro $maxf$ and a suitable definition of $minf$ by:

$$(h{:}maxf - h{:}minf) \ \leq \ (l{:}maxf - l{:}minf).$$

Otherwise, one could directly encode a macro expressing this notion, or, third, one could define a macro $maxdiff$ that expands to $maxf - minf$.

The preference frame with the preference (4) induces a total preorder on the set of histories $\boldsymbol{H}$. It is then a simple matter to select the most preferred history or histories. Of course, things may get much more complicated, particularly in the presence of different forms of preferences or multiple types of preferences. For details on such issues, see (Delgrande, Schaub, & Tompits 2005).

[2]To be sure, this isn't great notation. On the one hand, $h{:}maxf$ is a macro name (composed of 6 characters) while $\boldsymbol{h} : f(\$1)$ is a semantic entity in our language $\mathcal{P}_{\Sigma,n}$ once a term is substituted in for $\$1$ in the macro expansion.

Conclusion

We have addressed the problem of expressing general preferences that include aggregate quantities over histories. Thus, inter alia, we addressed adding preferences, including preferences on aggregates, in planning systems. We first defined a query language, $\mathcal{R}_{\Sigma,n}$, that extended our earlier query language $\mathcal{Q}_{\Sigma,n}$ (Delgrande, Schaub, & Tompits 2005) by the addition of a conditional term-forming operator. In addition, we defined the notion of macros for this language. Given this, we showed how our second language $\mathcal{P}_{\Sigma,n}$ could be employed for defining general preferences including preferences on aggregate quantities. As before, the framework allows the expression of conditional preferences, or preferences holding in a given context, as well as (trivially) absolute preferences, expressing a general desirability that a formula hold in a history.

We have argued previously that the overall approach is very general and flexible; specifically, we argued that previous approaches to preferences in planning are expressible in our formalism. With the added aggregate capability we can now express aggregate preferences like those discussed by Eiter *et al.* (2003), but notably within our broader preference framework. As well, while the approach is formulated within the framework of action languages, our results are applicable to general planning formalisms.

In a planning context, our approach would amount to generating plans and selecting the most preferred plan based on the preferences. As such, the approach is readily adaptable to an anytime algorithm, in which one may select the currently-best plan(s), but with the hope of a more-preferred plan being generated. An obvious topic for future work is to directly generate a preferred plan (rather than selecting from candidate plans); however, this appears to be a significantly difficult problem. Another topic for future work is to generalize our notion of macro, perhaps allowing for fluent arguments. Thus, if it can be carried out with no additional computational overhead, rather than having a macro $maxf$ for the maximum value of fluent f, it would be more convenient to express this as $max(f)$ where f is now an argument for a general max macro.

Acknowledgements The first author was partially supported by a Canadian NSERC Discovery Grant. The second author was partially supported by DFG under grant SCHA 550/6, TP C. The authors are grateful to Yannis Dimopoulos for getting them interested in preferences on aggregrates.

References

Bienvenu, M., and McIlraith, S. 2005. Specifying and generating preferred plans. In McIlraith, S.; Peppas, P.; and Thielscher, M., eds., *Seventh International Symposium on Logical Formalizations of Commonsense Reasoning*, 25–32.

Delgrande, J.; Schaub, T.; and Tompits, H. 2004. Domain-specific preferences for causal reasoning and planning. In Dubois, D.; Welty, C.; and Williams, M., eds., *Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning*, 673–682. Whistler, BC: The AAAI Press/The MIT Press.

Delgrande, J.; Schaub, T.; and Tompits, H. 2005. A general framework for expressing preferences in causal reasoning and planning. In McIlraith, S.; Peppas, P.; and Thielscher, M., eds., *Seventh International Symposium on Logical Formalizations of Commonsense Reasoning*, 46–54.

Eiter, T.; Faber, W.; Leone, N.; Pfeifer, G.; and Polleres, A. 2003. Answer set planning under action costs. *Journal of Artificial Intelligence Research* 19:25–71.

Fritz, C., and McIlraith, S. 2005. Compiling qualitative preferences into decision-theoretic Golog programs. In *IJCAI'05 Workshop on Nonmonotonic Reasoning, Action and Change*, 45–52.

Gelfond, M., and Lifschitz, V. 1998. Action languages. *Electronic Transactions on AI* 3. Available at `http://www.ep.liu.se/ej/etai/`.

Haddawy, P., and Hanks, S. 1992. Representations for decision-theoretic planning: Utility functions for deadline goals. In *Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning*, 71–82.

Myers, K., and Lee, T. 1999. Generating qualitatively different plans through metatheoretic biases. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, 570–576.

Son, T., and Pontelli, E. 2004. Planning with preferences in logic programming. In Lifschitz, V., and Niemelä, I., eds., *Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'04*), volume 2923 of *Lecture Notes in Artificial Intelligence*, 247–260. Springer Verlag.

Wellman, M., and Doyle, J. 1991. Preferential semantics for goals. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, 698–703.