Abstract

Goal models have been found to be useful in Requirements Engineering for representing the relationship between stakeholder goals and system or human tasks needed to fulfill them. Often, such specifications constitute rather idealized plans for goal fulfillment, where task executions never fail and domain assumptions always hold. In reality, however, there is always uncertainty as to whether a particular execution of a specification will actually fulfill the stakeholder goals and preferences as it is supposed to. In this paper, we introduce the concept of decision-theoretic goals in order to represent and reason about both uncertainty and preferential utility in goal models. Goal models are extended to express probabilistic effects of actions and also capture the utility of each outcome with respect to the stakeholder preferences. They are then automatically translated into a formalism that allows a state-of-the-art AI tool, DT-Golog, to reason about optimal solutions. In this way, analysts can find possible courses of actions for fulfilling stakeholder goals while investigating the risks and expected value of those solutions. Applications in a real-world meeting scheduling problem and the London Ambulance Service (LAS) case are presented and performance experiments are conducted to illustrate the appropriateness of this toolset for Requirements Engineering problems.

Keywords: Information Systems Engineering, Goal Modeling, DT-Golog, Decision Theory
1 Introduction

Goal-Oriented Requirements Engineering is founded on the premise that functional requirements for information systems can be derived from stakeholder goals through a systematic process [1]. For example, the goal Schedule a Meeting in e.g. a University setting might be fulfilled by a system that supports a set of functions (gather constraints automatically, find free slots, send out reminders, etc.) as well as actions carried out by external actors (participants, a meeting initiator etc.). When the designer selects an alternative for fulfilling top-level stakeholder goals and generates a design, the implicit claim is that the design will fulfill every instance of the goal (e.g., successfully schedule and hold every requested meeting).

Unfortunately, the world is not that simple. The design – which implements a generic plan for fulfilling the goal – may actually fail for a number of reasons, including limited resources, bad scheduling, unexpected obstacles, and more. For instance, participants may provide inaccurate constraints or maintain incomplete on-line calendars. Or the email with the meeting invitation may include the wrong time or room. Even sending the email per se often does not guarantee its receipt, especially when mail servers or anti-spam filters are not appropriately maintained. Hence, actions carried out by the system or actors in its environment – produce effects that vary in a non-controllable way. In other words, a design may fail to fulfill instances of a goal due to violation of implicit domain assumptions and axioms [1], such as those pertaining to the expected effect of system or user actions.

At the same time, we wish to maintain the multi-objective nature of alternatives analysis. Thus, potentially conflicting goals such as Quick Scheduling vs. Maximize Attendance or Keep Secretary Unburdened vs. Minimize Reliance on Systems may each be served better by different designs. Stakeholders may be willing to exchange an increased probability of failure with an increased value in one or more of those objectives in case of success. In these circumstances, searching for a suitable design is a process of finding designs that offer the best combination of quality, based on stakeholder preferences, and likelihood of success (i.e. the best expected value).

In this paper, we introduce the concept of decision-theoretic goals, which combine the merits of their probabilistic [2] and their preferential [3] cousins in order to capture both probabilistic uncertainty and preferential utility in goal models. To achieve this, we extend the preference and priority-enabled goal modeling language we proposed in [3] to allow representation of probabilistic actions, that is, actions that do not have just one unique effect but a probability distribution over possible effects/outcomes. Utility functions, on the other hand, assign different desirability measures to different such action outcomes. The extended goal model is then translated into DT-Golog, a formal specification language that combines ideas from dynamic domain specification languages and Markov Decision Processes (MDPs) [4, 5]. A DT-Golog reasoning tool is then used to evaluate alternative designs by which the specified goals are fulfilled with optimal expected value. This way, both likelihood and value are considered when searching for good solutions to the given requirements problem.

We organize the paper as follows. In Section 2 we present our goal modeling notation and in Section 3 we show how we extend it to allow for decision-theoretic analysis. In Section 4 we show what kinds of automated reasoning the technique enables. Then,
in Section 6 we report on an application to a real-world meeting scheduling problem as well as the London Ambulance Service (LAS) case and discuss tool performance. Finally, we survey related work in Section 7 and conclude in Section 8.

## 2 Goal Models

Goal models ([1, 6]) have been found to be effective in concisely capturing large numbers of alternative sets of low-level tasks, operations, and configurations that can fulfill high-level stakeholder goals. In Figure 1, a (simplified) goal model for scheduling meetings is depicted. The model shows how the high-level goal of a meeting initiator to *Have a Meeting Scheduled* is analyzed into the particular subgoals and actions that are needed for the goal to be attained. The model primarily consists of *goals* and *tasks*. Goals – the ovals in the diagram – are generally defined as states of affairs or conditions that one or more actors of interest would like to achieve [6]. Tasks, on the other hand, – the hexagonal shapes – describe particular activities that the actors perform in order to fulfill their goals.

Goals and tasks are connected with each other via AND- and OR-decompositions. By AND-decomposing a goal into other subgoals or tasks, we indicate that the satisfaction of each of its children is necessary for the decomposed goal to be fulfilled. However, tasks and subgoals that are children of AND-decompositions can be desig-
nated as optional through a circular annotation added on their top side, such as Send Attendance Reminder in the figure. On the other hand, if the goal is OR-decomposed into other goals or tasks, then the satisfaction of one of these goals or tasks suffices for the satisfaction of the parent goal.

The order in which goals and tasks are satisfied and performed respectively is relevant. To express constraints over satisfaction ordering we use a precedence link (\( \text{pre} \rightarrow \)). A precedence link drawn from a goal/task to another goal/task, indicates that satisfaction/performance of the target of the link cannot begin unless the origin is satisfied or performed. Thus, the precedence link from Find Suitable Room to Meeting Announced indicates that unless the former is performed, none of the tasks below the latter can be performed. Furthermore, the negative precedence link (\( \text{npr} \rightarrow \)) indicates that performance of the link target cannot start if the element at the origin of the link has been satisfied.

Moreover, soft-goals (the cloud-shaped elements) represent goals whose fulfillment does not have a clear-cut satisfaction criterion. Since satisfaction of soft-goals cannot be established in a crisp manner, the degree by which they are satisfied is assessed through evidence of satisfaction of other goals. In the goal model, this is represented through positive helps (\( \downarrow \rightarrow \)) and negative hurts (\( \rightarrow \)) contribution links drawn from goals and tasks to soft-goals.

The AND/OR decomposition implies a number of sequences of leaf-level tasks that can satisfy the top level hard goal. We call such sequences plans. The variability of such plans emerges both due to the existence of OR-decompositions and optional sub-goals in the AND/OR tree, allowing for different subsets of tasks that can fulfill the root goal, and due to the fact that a given subset (i.e., a solution to the AND/OR tree) can be ordered in different ways subject to \( \text{pre} \rightarrow \) and \( \text{npr} \rightarrow \) constraints. Furthermore each plan has a different impact to high-level soft-goals. Back in Figure 1, a plan that includes calling everybody to acquire constraints has a negative impact to the soft-goal Reduce Labour and should be avoided if that soft-goal is important. It would be a good plan, however, if Quick Scheduling were a high priority goal.

3 Goals, Probabilities and Utilities

3.1 Decision-Theoretic Goals

In the standard notation we described above performance of tasks is assumed to bring about the desired result with certainty. In reality, however, tasks have multiple intended or unintended outcomes, each with different likelihood. As such, task performance does not guarantee goal achievement. To model and reason about this uncertainty, the traditional concept of a goal has been extended to include a probability of success to it [2, 7]. Hence, probabilistic goals describe a desired state of affairs as well as a minimum probability value for this state to be successfully reached. Thus:

“Ambulance has arrived within 15 mins”, prob 0.95

is a probabilistic goal that says that the desired state of affairs is actually also desired to be achieved 95% of the times. To this probabilistic extension of goals, however, we add here another dimension: that of utility as measured by the impact that solutions of the
goal have to high-level qualities and stakeholder preferences thereof. Thus, *decision-theoretic goals* require maximization of *expected utility*, which combines probability of success and utility. Thus:

> “Have Meeting Scheduled”, optimally

requires that a meeting is scheduled while maximizing expected utility. Nevertheless, since expected utility combines probability and utility, it is possible that a plan with a good expected utility score has a forbiddingly low success probability. Thus goal:

> “Have Meeting Scheduled”, optimally, prob 0.7

demands that we wish to schedule the meeting optimally but also ensure that the probability of success exceeds 0.7.

Decision-theoretic goals prescribe both the quality that plans to achieve them must meet, in terms of satisfying high-level preferences, and our risk tolerance with respect to those plans. To allow reasoning about such goals we need to extend the standard goal modeling formalism as we describe below.

### 3.2 Representing State and Probabilistic Effects

The first step in our extension is to explicate what is true in the environment or context before, while and after the tasks of the goal model are performed. To do so we use *domain predicates*. Domain predicates represent fixed facts about the domain – e.g. `available(secretary)` or `has(projector, meetingRoom)` – or facts that may vary due to the performance of tasks or exogenous reasons – e.g. `invitationsSent` or `requested(meetingRoom)`. The truth values of the domain predicates that have been defined determine the state in which the process for fulfilling the goals is.

Let us now focus on tasks. Ideally, performance of a task by an agent implies that certain facts in the domain change in a deterministic way, leading the system to a new state with certainty. In reality, as we claimed, this cannot always be assumed. Firstly, the outcomes of some tasks rely on chance due to their nature. For example the task `Find Suitable Time Slot` may or may not lead to a situation where `slotFound` holds, depending on the scheduling constraints at hand. Secondly, there are tasks that have one expected and/or desired outcome, but they always run a probability of failure. Thus, the task `Send Invitations` will most probably lead to the fact `invitationsReceived` being true, but there is always a probability of the same fact being false due to a number of factors, such as infrastructure error (server down, anti-spam false positives) or human error (accidental deletion or mishandling of email). Human actors in particular may have their own unidentified and conflicting goals that prevent them from acting as prescribed. Thus, we are interested in representing *probabilistic effects* of tasks – that is effects that, for a variety of reasons, may lead to different outcomes with different likelihood.

To do this we first associate each task with a number of effects that can potentially be brought about upon the task’s performance. These are actually domain predicates. Thus, performance of the task `Receive Responses` may or may not have an effect that `adequateResponsesReceived`, meaning that important participants responded but not many others, versus the competing (mutually-exclusive in our case) effects `tooFewResponsesReceived`, meaning that too few or none of the important participants responded and `excellentResponsesReceived`, meaning that a very satisfactory amount of responses
Table 1: Effect, Utility, and Preference Tables

<table>
<thead>
<tr>
<th>Domain Predicates</th>
<th>V</th>
<th>V</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>excellentResponsesReceived</td>
<td>0.0</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>adequateResponsesReceived</td>
<td>0.5</td>
<td>0.25</td>
<td>0.5</td>
</tr>
<tr>
<td>tooFewResponsesReceived</td>
<td>0.5</td>
<td>0.25</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Attainment Formula: (adequateResponsesReceived ∨ excellentResponsesReceived)

Table 1: Effect, Utility, and Preference Tables

has arrived, including the important participants. Each of those effects occurs with a certain probability given different conditions.

We can represent these probabilities using a decision table such as that of Table 1a; we call it the effect table for the task. The table actually represents the probability distribution over possible effects of the task Receive Responses. It contains one or more condition variables, which are the variables on which the probability of successful execution of the task at hand depends, as well as one or more decision variables, which represent possible configurations of effects that the task can bring about. Both condition and decision variables are drawn from the set of domain predicates. Each combination of condition and effect configurations occurs with a certain probability.

In the example of Table 1a, there are three decision variables (domain predicates: excellentResponsesReceived, adequateResponsesReceived and tooFewResponsesReceived) and the probability that a certain combination of truth values occurs depends on three condition variables that have to do with how long the initiator waited after the original invitation (domain predicates: waited1Day, waited3Days and waited1Week). Thus the probability that adequate responses will arrive within the first three days is 0.75. Note that, in the particular example, both decision and condition variables are mutually exclusive. In the general case, arbitrary combinations of values can be considered.

3.3 Redefining Goal Satisfaction

Hard-goals and Probabilistic Effects. The probabilistic interpretation of task effects necessitates certain refinements to the goal model of Figure 1. Firstly, satisfaction of hard-goals does not exactly reflect the AND/OR structure of the underlying subtree anymore, because it is now measured by the effect of the underlying tasks and not by the mere fact that the tasks are performed. Hence, we define satisfaction of the goal based on the desirable effects of the task.
In the case of multiple effects, as in Receive Responses of Table 1a, we construct the attainment formula of each task and hard-goal exclusively based on domain predicates, which signifies what effects must be brought about to consider a goal or task satisfied or performed. In our example, the attainment formula of task Receive Responses could be excellentResponsesReceived $\lor$ adequateResponsesReceived. Satisfaction of higher level hard-goals is defined via conjunctions or disjunctions of attainment formulae of tasks depending on the corresponding AND/OR structure. Thus, the attainment formula of Book Meeting is slotFound $\land$ roomBooked, each being, in turn, predicates describing probabilistic effects of tasks Find Suitable Slot and Find Suitable Room, respectively. Note that it is in the discretion of the modeller to re-define satisfaction conditions, by, for example, setting the attainment formula of Receive Responses to be just excellentResponsesReceived – hence stricter than the previous one.

Assessing Soft-goal Satisfaction. As with hard-goals, in light of probabilistic effects of tasks, a refined model of the satisfaction of soft-goals should also depend on the actual outcome of task performance, rather than the mere fact that a task was performed. For example, the claim that the task Send Attendance Reminder contributes negatively to the soft-goal Avoid Annoying the Participants can be supported only if the reminder actually went through – if not, no annoyance can reasonably be assumed. Thus, contribution links are refined into relationships between domain predicates and soft-goals. More specifically, similarly to the attainment formula we saw above, each soft-goal is assigned an attainment function that maps the set of states of the domain to the set of real numbers. Thus, different configurations of truth values for the domain predicates imply a potentially different value for the attainment function of the soft-goal at hand. The higher the attainment value, the more the soft-goal is believed to be satisfied. We consider the interval $[0,1]$ for the values, where 1.0 represents full satisfaction of the soft-goal and 0.0 its full denial. Thus, in effect, we quantify the originally qualitative contributions of the goal model – we show how below.

We found that representation of attainment functions is also possible using a tabular format such as that of Table 1b – we call it the utility table. The condition variables of the table represent the possible values of the domain predicates that influence the satisfaction of the soft-goal. Each configuration of truth values for the decision variables is associated with the actual satisfaction value (seen as a utility value) of the soft-goal at hand. Thus, these values express utility (with respect to the soft-goal at hand) of the situation that is described in each value configuration. In Table 1b, a possible attainment function for the soft-goal Avoid Annoying the Participants is shown. Attainment of that goal largely depends on whether the meeting organizer has called all participants on the phone to gather constraints, expressed through domain predicate calledEverybody as well as whether s/he has (successfully) sent them reminders to attend the meeting, modeled through the domain predicate reminderArrived. In the utility table, different combinations of truth values of these domain predicates imply a different attainment value for the soft-goal, shown in the last column.

Aggregating utilities through preferences. After we construct the tables that represent the attainment function for every soft-goal, each state of the domain (i.e., each combination of truth values of the domain predicates) implies a different attainment
value for each soft-goal. But how can we assign a universal “goodness” value of each state of the domain, in order to use it for comparing plans? Preferences [3] allow us to do exactly this. A preference profile is a representation of the relative importance of soft-goals, in form of a weighted numeric combination. Table 1c shows such a combination for three soft-goals. As we also show in [8] multiple such tables can be constructed in each level of a soft-goal hierarchy and aggregated through nested linear combinations. The ground terms of the resulting linear combination are satisfaction values of low-level soft-goals. These values, in turn, come directly from the utility tables. Thus, assuming the preference profile of Table 1c, in a state where soft-goal Avoid Annoying the Participants is satisfied by, e.g. 0.3 and Quick Scheduling and Reduce Labor are satisfied by 0.9 and 0.7 respectively, the overall value for that state will be

\[0.5 \times 0.6 + 0.9 \times 0.3 + 0.7 \times 0.1 = 0.52.\]

**Getting the numbers.** The quantitative measures we discuss above occur both in the form of probabilities and in the form of utility/priority values. Overall, while we focus in this paper on the technical representation and reasoning aspects, we believe that there are solid methods and experience in terms of eliciting probabilities and utility measures [9]. As we demonstrate below, probability numbers can come from either simple measurements in the domain or, in the absence of such, subjective judgement by the modellers. Further, there is a variety of ways by which utility and preference numbers can be found, including prominent requirements prioritization techniques such as AHP [10, 8]. Thus, both Tables 1b and 1c can be results of AHP’s pairwise comparisons – we describe how below. Even subjective ad-hoc assessment is a realistic possibility: it has been found that even if the numbers are not exact, they may be good enough to make correct informed decisions [11]. Otherwise, numerical attainment values are expressions of utility and as such can be obtained through a number of more systematic techniques such as reward elicitation [12].

## 4 Reasoning about Decision-Theoretic Goals

The above extensions are useful for performing automated reasoning about goal satisfaction under probabilistic effects, utilities and soft-goal preferences. To enable this, the extended goal model is translated into a formalism, called DT-Golog [4, 5]. The resulting specification allows us to use the DT-Golog interpreter in order to identify optimal plans which, as it can be shown, map to optimal plans in the goal model as well. Let us return to the example of Figure 1 and discuss different kinds of decision-theoretic goals we can reason about using DT-Golog with the generated specification.

**Optimizing expected utility.** Decision-theoretic goals of the form “Schedule Meeting”, optimally are satisfied by a plan \(p\) of goal model \(G\), iff \(p\) brings about the maximum accumulated expected utility in \(G\). The necessary probability and utility measures are drawn from the appropriately translated effect and utility tables we saw above. Thus, in Figure 1 and assuming we have introduced effect and utility tables for each of the involved tasks and soft-goals accordingly (which we do not present for the interest of space), by setting all soft-goals to be of equal preference we find a policy with total
accumulated expected utility 2.7, whose success plan is (referring to abbreviations in the parentheses inside the task symbols) \([\text{si}, \text{w7}, \text{rr}, \text{fs}, \text{fr}, \text{se}, \text{sar}, \text{pam}]\). DT-Golog will also inform us that the probability of successful termination of this plan is 0.4. This result is, of course sensitive to probability values as well as the structure of the utilities. Thus, if we assume that soft-goals follow the preference values of Table 1c, instead of having equal preference as we assumed above, the success plan would be \([\text{au}, \text{fs}, \text{fr}, \text{se}, \text{sar}, \text{pam}]\) with accumulated expected utility 3.1 and probability of success 0.34. Clearly, the increased importance of soft-goal Reduce Labour in the preference table favours the choice of automated constraint gathering \(\text{au}\).

**Testing Probability Thresholds.** The other kind of decision-theoretic goals that we saw has the form “Schedule Meeting”, optimally, prob \(c\), where \(c\) is a probability value. Such a decision-theoretic goal is satisfied by plan \(p\) of goal model \(G\) iff \(p\) has the maximum accumulated expected utility in \(G\) and \(p\) has a probability of success greater or equal to \(c\). Thus, DT-Golog simply tests if the optimal plan has a probability of success above \(c\). For example, the above optimal plan has a success probability of 0.34, meaning that, if we also had a probability threshold \(c\) of, say, 0.7, DT-Golog would report failure to find suitable plan. Note that optimality is defined in a global sense and independent of the probability threshold.

## 5 Translating to DT-Golog

Let us now turn our focus to the mechanics of the translation of our extended goal models into DT-Golog, which enables automated reasoning. We start by looking at DT-Golog in more depth and then present the translation details by example.

### 5.1 DT-Golog Basics

DT-Golog is a decision theoretic extension of Golog, a language for modeling and reasoning about dynamic domains [5, 4]. DT-Golog incorporates Markov Decision Processes (MDPs) in Golog’s reasoning infrastructure which allows for reasoning about probabilistic actions subject to optimization of expected utility. In the following we outline the aspects of DT-Golog that are essential for understanding the subsequent translation rules, referring the reader to the respective literature for more details [4].

The core of DT-Golog consists of constructs prescribed by situation calculus: *fluents, actions and situations*. Fluents, represented through n-ary predicates relativized to a particular situation, are state features whose value can vary from situation to situation due to the performance of actions. For example, the unary fluent `room_booked(largeTheater, s)` holds in situation \(s\) as a result of an action of sending a booking request. Actions are first order terms signifying specific activity performed by agents, e.g. `bookRoom(initiator, largeTheater)`. A situation is also a first-order term that denotes a sequence of actions. The function symbol `do(\alpha, s)` denotes the situation which results from performing action \(\alpha\) in situation \(s\). A special constant \(S_0\) denotes the initial situation.
A set of axioms over these basic constructs are then defined in order to describe the domain. From those the most important are action precondition axioms that tell us when actions are possible and successor state axioms that describe how fluent values change due to the performance of actions. The former are defined for each action $\alpha$, and are of the form $\text{Poss}(\alpha, s) \equiv \Pi_\alpha(s)$ signifying that performance of the action $\alpha$ is possible if and only if some condition $\Pi_\alpha$ is true in the particular situation. Successor state axioms, on the other hand, are defined for each fluent and are of the form $f(\vec{x}, \text{do}(\alpha, s)) \equiv \Phi_f(\vec{x}, \alpha, s)$ where $f$ is an $n$-ary fluent symbol, $\vec{x}$ represents its $n$ arguments, and $\Phi_f$ is a formula that says that $f$ will be true after performance of action $\alpha$, either if the action is an action that enables it or if the fluent was already true and $\alpha$ is not an action that turns it false.

Given an action theory, DT-Golog’s reasoning tool is able to find sets of plans (precisely: policies) that reach target situations of our choosing. This is possible through constructing programs which include, among other things, sequences of actions (denoted as $\alpha_1; \alpha_2$), if-then-else conditionals, while loops, (sub-)procedures with recursion as well as (and this is the important difference from common procedural languages) non-deterministic choices of actions (denoted as $\alpha_1|\alpha_2$), and non-deterministic choices of arguments.

The above are features found in all interpreters of the Golog family. DT-Golog extends these in different ways, two of which are the most interesting for our purposes. Firstly, to the agent actions of core Golog, called stochastic actions in DT-Golog, nature actions are added to denote exogenous events that may happen with a probability. We use a set of nature actions $\alpha_i$ to represent the actions that might have actually happened due to the influence of nature when $\alpha$ (e.g. sending an invitation) was attempted. The predicates $\text{prob}(\alpha_i, p_i, s)$ are used to assign probabilities $p_i$ to each such nature action. Note that the probabilities $p_i$ can be a simple numerical value or a complex numerical function of fluent values. Secondly, a reward symbol $\text{reward}(r, \text{do}(\alpha_i, s))$ assigns a reward value to situations, actions or both.

In the face of non-deterministic choices, DT-Golog’s reasoning engine searches for a policy that maximizes the total accumulated expected utility defined as a possibly discounted sum of the products of the reward and the probability that this reward occurs when following a certain action trajectory. This is similar to computing an optimal policy in the finite horizon Markov Decision Processes using forward decision tree search. This way, DT-Golog is able to evaluate plans that are not only feasible in the Golog action theory but also optimal in the Decision Theoretic extension. In addition, for each policy it returns, DT-Golog calculates the probability that the policy is executed successfully. Details on these calculations can be found in [4].

5.2 From Goal Models to DT-Golog

The translation of the refined goal model to DT-Golog is such that it can be automated allowing, thus, analysts to perform the reasoning task without having any knowledge of the DT-Golog formalisms. In the interest of simplicity and space, we sketch how the translation is possible based on the example of Figure 2 translated into DT-Golog specifications as (partially) described in Figures 3 and 4.
Figure 2: Translation by Example
Translating the base constructs. The translation of the basic goal model constructs is performed as follows. First, each leaf level task $t$ is translated into a stochastic agent action $a_t$. For each such stochastic action, we also introduce a set of nature actions $a_i$ denoting possible outcomes of the execution of $a_t$. These are directly derived by looking at the decision variables of the effect table of each task. Specifically, for each decision possibility (configuration of decision values) we introduce such a nature action. Thus in Table 1 of Figure 3, task $t_2$ of Figure 2 has been translated into four nature actions denoting the four combinations of truth values of effects $e_2$ and $e'_2$ (denoting their respective negations). Actions $a_{e_2,e'_2}$ and $a_{e_2,e'_2}$ are examples of how such nature actions are denoted based on the effects they bring about.

Precondition Axioms. For each nature action, we specify an action-precondition axiom. In particular, if there is an incoming $\preceq$ link to a task node $t$ from a hard-goal or task $h$, then the attainment formula of $h$ is added to the preconditions of all the associated nature actions $a_i$. Moreover, if there is an incoming $\npreceq$ link to the task node $t$, then the negation of the attainment formula for the source node of this link is added to the preconditions of $a_i$. In the absence of any such links, these actions are specified to be always executable. In Table 3 of Figure 3, we specify the preconditions of the tasks of Figure 2 using the special predicate $Poss(a,s)$, which, as we saw, denotes that action $a$ is executable in situation $s$.

Successor State Axioms. For each domain fluent $\phi_e$ associated with effect possibilities for task $t$, we need a successor-state axiom that succinctly encodes both direct effects and non-effects and specifies exactly when the fluent changes. Such axioms can be easily generated as in Table 4 of Figure 3. Thus, for $t_2$ there are two axioms: one stating which actions make $\phi_{e_2}$ true (these are nature actions $a_{e_2,e'_2}$ and $a_{e_2,e'_2}$) and one stating which actions make $\phi_{e'_2}$ true.

Procedures. For each goal $g$, we also introduce a DT-Golog procedure $proc_g$, which comprises of a test/wait action $\phi?$ that waits for the preconditions of the procedure to hold, followed by some program $\delta$. The precondition requires that conjunction of all incoming $\preceq$ links must be satisfied and the disjunction of all incoming $\npreceq$ links must not be satisfied. If $g$ is AND-decomposed, $\delta$ consists of the interleaving of its subtasks and subgoal procedures. On the other hand, if $g$ is OR-decomposed, the program $\delta$ consists of the non-deterministic choice between its various subgoals and subtasks. In Table 5 of Figure 3, the translation of the AND/OR structure of Figure 2 through applying these ideas can be seen. Note that $(a\|b\|c)$ denotes the non-deterministic
Table 1: Actions and Fluents

<table>
<thead>
<tr>
<th>Task</th>
<th>Action</th>
<th>Fluent</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>a₁, a₂, a₃</td>
<td>φ₁₁</td>
</tr>
<tr>
<td>t₂</td>
<td>a₂, a₂, e₂', a₂', e₂', a₂', e₂', a₂', e₂', φ₂₂, φ₂₂'</td>
<td></td>
</tr>
<tr>
<td>t₃</td>
<td>a₃, a₃, e₃</td>
<td>φ₃₃</td>
</tr>
</tbody>
</table>

Table 2: Attainment Formulae:

<table>
<thead>
<tr>
<th>Task/Goal</th>
<th>Axiom</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>φ₁₁</td>
</tr>
<tr>
<td>t₂</td>
<td>φ₁₂</td>
</tr>
<tr>
<td>t₃</td>
<td>φ₁₃</td>
</tr>
<tr>
<td>g₁</td>
<td>φ₁₁ ∨ φ₂₁</td>
</tr>
<tr>
<td>g₂</td>
<td>φ₁₂ ∨ φ₁₃ ∨ φ₁₄</td>
</tr>
</tbody>
</table>

Table 3: Action Precondition Axioms:

<table>
<thead>
<tr>
<th>Task</th>
<th>Precondition Axioms</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>Poss(a₁, s) ≡ true</td>
</tr>
<tr>
<td></td>
<td>Poss(a₂, s) ≡ true</td>
</tr>
<tr>
<td>t₂</td>
<td>Poss(a₁, s) ≡ true</td>
</tr>
<tr>
<td></td>
<td>Poss(a₂, s) ≡ true</td>
</tr>
<tr>
<td></td>
<td>Poss(a₂, e₂', s) ≡ true</td>
</tr>
<tr>
<td></td>
<td>Poss(a₂, e₂', s) ≡ true</td>
</tr>
<tr>
<td>t₃</td>
<td>Poss(a₁, s) ≡ φ₁₃(s) ∨ φ₂₁(s)</td>
</tr>
<tr>
<td></td>
<td>Poss(a₂, s) ≡ φ₁₃(s) ∨ φ₂₁(s)</td>
</tr>
</tbody>
</table>

Table 4: Successor State Axioms:

<table>
<thead>
<tr>
<th>Task</th>
<th>Successor State Axioms</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>φ₁₁(do(a, s)) ≡ φ₁₁(s) ∨ a = a₁</td>
</tr>
<tr>
<td>t₂</td>
<td>φ₁₂(do(a, s)) ≡ φ₁₂(s) ∨ a = a₂ ∨ e₂' ∨ a = a₂ ∨ e₂'</td>
</tr>
<tr>
<td></td>
<td>φ₁₂(do(a, s)) ≡ φ₁₂(s) ∨ a = a₂ ∨ e₂' ∨ a = a₂ ∨ e₂'</td>
</tr>
<tr>
<td>t₃</td>
<td>φ₁₃(do(a, s)) ≡ φ₁₃(s) ∨ a = a₃</td>
</tr>
</tbody>
</table>

Table 5: Procedures:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>g₁</td>
<td>proc₁ ≡ ([a₁]proc₁₂)</td>
</tr>
<tr>
<td>g₂</td>
<td>proc₂ ≡ ¬φ₁₁ ∨ (a₂ [a₁]a₄)</td>
</tr>
</tbody>
</table>

Figure 3: Examples of DT-Golog Specifications for Figure 2
choice between all possible interleaving of actions $a$, $b$, and $c$ – though some may not be feasible due to lower-level precedence constraints.

**Probabilities.** Returning to the effect table of a task $t$, we saw that for each configuration of values for the decision variables (the actual effects of $t$) a nature action $a_i^t$ is introduced. For each such nature action we simply introduce predicates of the form $\text{prob}(a_i^t, p_i, s)$, where $a_i^t$ is the nature action whose probability we define and $p_i$ is that probability taken from the effect table, by looking up the configuration of decision variables that correspond to $a_i^t$. In the presence of condition variables the formula depends on the configuration of those as well and is, hence, written as $\text{prob}(a_i^t, p_i, s) \text{ if } \Phi(s)$, where $\Phi(s)$ is a conjunction of all condition variables in the effect table or their negation depending on the value configuration at hand. Table 6 of Figure 4 shows the probability definitions for nature actions related to tasks $t_1$, $t_2$ and $t_3$ of Figure 2.

**Rewards.** The reward function is calculated in a very similar way, with the difference that, since reward is a unique value that characterizes an entire solution, values from individual reward tables are merged together based on the given soft-goal preference profile. Recall that a preference specification describes the relative importance of each of the top level soft-goals. At the same time, for each truth assignment for the effects, the satisfaction function of each soft-goal has a specific value. Gathering all those values, multiplying them by the weight of their corresponding soft-goal in the preference specification and adding them up gives us the overall reward value for the situation. For example in a situation in which $e_1$, $e_2$, $e_3$ are true and $e_4$ is false, by looking at the tables, $o_1$ is satisfied by 0.5 and $o_2$ by 0.75, and given their relative importance 0.8 and 0.2, the total reward is $0.5 \cdot 0.8 + 0.2 \cdot 0.75 = 0.55$. Table 7 of Figure 4 has reward examples for the model of Figure 2.
6 In Practice

6.1 A Meeting Scheduling Study

As a preliminary test of the feasibility of our technique, we applied it to a meeting scheduling problem that occurs in our workplace. Our SE@York seminars are events that we organize at York University and feature regular talks by visiting or resident software engineering scholars and PhD students. By examining this simple scheduling case we aim at: (a) assessing how feasible it is to model our own concerns pertaining to that problem using the extended goal modeling notation we introduced, (b) understanding how possible and easy it is to get the numbers necessary to perform the proposed reasoning exercise, (c) explore what strengths the automated reasoning task adds to the domain understanding process.

The first author is the meeting initiator of the SE@York meetings and has access to relevant data sources. Potential participants are professors and graduate students of the IT and CS departments. The standard request-based constraint acquisition is performed by the initiator as seen in the model of Figure 1. In terms of quality goals, the real concern of the organizers is to have good attendance. To a lesser extent they would like to have the meeting scheduled as quickly as possible, for varying reasons including that e.g. a visitor speaker is leaving the country or running out of patience. So far, we have been improvising and relying on our intuition to address certain dilemmas pertaining to scheduling our meetings. For example, what is a good enough response rate to the constraint invitation email before the initiator proceeds with deciding a slot? How long should the initiator wait for more constraints? Does sending a reminder for the meeting increase attendance?

Developing the Effect Tables. In our domain, probabilistic data comes from the initiator’s email archives (constraint requests and responses) as well as the paper-based room booking logs. The numbers presented in the effect table of task Receive Responses in Table 1a are actual values coming out of our data. The email archive data also allow us to calculate the probability that a slot will eventually be found (0.75, in our case). The room booking logs, on the other hand, allow us to calculate the probabilities that the meeting room will be available. For our study, we simply looked at the probability that the room is available at any workday from 9am to 5pm in January. Unfortunately, no reliable data is available on attendance and the effect of reminders to it. To address the lack of data, we assign unconditional 1.0 to be the probability of adequate attendance. Alternatively we could attempt a subjective judgement of the probability based on intuition. In either case, we would chunk the space of numbers of people who show up into a small number of discrete labels similar to the ones we use to characterise response rate, e.g. “adequate”, “marginal”, “inadequate” etc.

Developing the Utility and Preference Tables. To elicit utilities we make use of the Analytic Hierarchy Process (AHP) [8]. AHP is based on forming a criteria hierarchy and comparing solutions subject to the leaves of the hierarchy. In AHP, both the relative “goodness” of solutions with respect to leaf level criteria and the relative importance of criteria at each level of the hierarchy is assessed separately and, to increase
validity, through pairwise comparisons. The results are aggregated into global values that characterize the overall preferability of each solution. To use AHP for assessing contribution measures in the goal model, the soft-goal hierarchy is treated as a small AHP criteria hierarchy [8]. Thus, in Figure 5 softgoals Maximize Attendance and Quick Scheduling are the only criteria of the AHP problem with elicited importance values 0.75 and 0.25 respectively. Such values constitute preference weights for representations such as the one of Table 1c. At the bottom of the hierarchy, two decision problems of concern are: (a) how important it is to have excellent responses when scheduling a meeting (versus just adequate) and (b) what the effect of waiting for responses (1 day vs. 3 days vs. 1 week) is to quick scheduling. Each decision problem is judged with respect to a different criterion and is treated as independent of the other.

**Reasoning.** While the problem is simplified to fit our narrow interests and data, DT-Golog allows a quantification of the value and likelihood of various possibilities. For the utilities of Figure 5, the optimal solution (value = 1.49) is to wait for seven days before deciding on a slot. The probability of success in that case is 0.46. Lack of success means for example that a slot or room is not found immediately or that most invitees don’t respond within a week – i.e. failure does not mean that the meeting is not possible but that the optimal plan may fail as such and a different plan is needed. Should Quick Scheduling be more important than Maximize Attendance – and in our SE@York meetings there have been such cases – after swapping the preference weights, the optimal solution with utility 1.28 is to try waiting for 1 day but much lower probability of success, 0.23, since within 1 day adequate responses may have not been received preventing the initiator from proceeding as the optimal plan suggests. Thus, DT-Golog gives us useful hints as to what practice is better under given preferences and what the relative likelihood is that complications will arise and that the meeting initiator will need to change plans or improvise.
Overall, we find that this preliminary exploration indicates that (a) even in an informal and unstructured environment where data collection is not a priority, probabilities that allow for useful analysis can be found, (b) AHP offers an applicable approach to identification of the necessary utility measures, and (c) even with rough data DT-Golog allows exploration of practices for best fulfilling goals and comparisons with respect to how likely it is that these practices will succeed. Of course, more thorough empirical work would allow us to assess the reliability of DT-Golog’s output in absolute terms with respect also to the intensity of the data collection effort that makes this possible. A study with encouraging results towards this direction is described below.

6.2 Adding Detail

Our use of DT-Golog with the specification that is generated from the semi-formal goal model, exploits only a subset of DT-Golog’s expressive power. To further study how DT-Golog’s expressive capabilities are applicable to the requirements analysis problem, an application to the well known London Ambulance Service (LAS) [14] case was also performed. The application is described in detail elsewhere [15] – here we focus on key features. The particular case concerns the problem of managing a fleet of ambulances to respond to emergency incidents in the city of London, UK. What makes the case particularly interesting for our purposes is the explicit performance requirements that can be imposed in the form of an exact probability distribution of allowable ambulance response times. More specifically, concrete performance and reliability requirements can be set for candidate dispatch strategies. Thus, we can demand that a request is responded to within 14 minutes of the time a call is placed. We can even require that activation time (call receipt and decision) should always be made in less than 3 minutes, while travel time to the incident should be 11 minutes 95% of the time and 8 minutes 50% of the time.

To search for designs that meet these performance objectives, extension of the initial DT-Golog specification needs to be performed by adding detail in a number of ways. Firstly, domain information is added in the form of particular instances of objects, agents and contexts that are involved in the LAS operations. Thus, the geography of three city regions is modeled using 10x10 grids. Each hospital, ambulance, incident etc. is represented as a DT-Golog fact and occupies at a given point in time a particular cell in the grid, representing its geographical position. Actions and fluents are relativised to particular objects through parameters. Thus, a fluent of the type \texttt{carLocation}(c,l,t,s) is used to represent that an ambulance \texttt{c} is at location \texttt{l} at time \texttt{t} in situation \texttt{s} – the location is represented through a term \texttt{loc(x, y)}, where \texttt{x} and \texttt{y} are co-ordinates in the grid. Actions also have a temporal argument, with which their duration is encoded.

To allow analysis of different dispatch strategies, each expected to have different performance characteristics, Golog procedures describing those strategies are written. These are more complex than simple sequences and choices that the translation framework we described above produces. Furthermore, the utility functions are an essential part of each strategy, as they describe the chosen optimization approach. Thus, aspects such as the familiarity of an ambulance driver in an area or the effect of personnel fatigue are modeled through appropriately structured utility tables.

Moreover, \textit{simulation} is necessary when there is a need to model random variables
Table 2: Time (in sec) to find optimal solution.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Bound</th>
<th>Time</th>
<th>Nodes</th>
<th>Bound</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>0.0</td>
<td>45</td>
<td>19</td>
<td>95.3</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>0.08</td>
<td>50</td>
<td>14</td>
<td>75.6</td>
</tr>
<tr>
<td>30</td>
<td>8</td>
<td>0.07</td>
<td>60</td>
<td>16</td>
<td>4395</td>
</tr>
<tr>
<td>40</td>
<td>12</td>
<td>3.7</td>
<td>65</td>
<td>21</td>
<td>(*)</td>
</tr>
</tbody>
</table>

representing exogenous events. In the LAS case, these are the occurrence of emergency incidents. A Poisson distribution of incidents is assumed with various arrival frequency scenarios. Different dispatch strategies are then repeatedly tried for a large number of requests. The response times are counted/averaged and compared with the set requirements, allowing better understanding of the behaviour of different response strategies.

Overall, the LAS case study shows that a great deal of detailed modeling work can be added to the basic model in order to obtain more realistic data and perform more precise analysis. Thus, it allows us to appreciate how the proposed toolset enables analysis at various granularity levels, from exploration of simple models, as in the SE@York case, to detailed analysis, as in the case of LAS.

6.3 Tool Performance

DT-Golog has been found to perform reasonably well compared to plain MDP solving. But how does it perform with our goal models? To explore this we tried it with different sizes of goal models, which we constructed by randomly combining smaller models we have developed for real domains (meeting scheduler, automatic teller machine, on-line bookstore and nursing). This way, the resulting artificial models preserved some degree of structural naturalness. Random numbers were entered for the probability values.

We had DT-Golog compute optimal policies for each root goal. The search horizon was set to the maximum plan length the goal model can yield. We used an Intel(R) Core(TM)2 CPU T5500 1.67 GHz with 4.00 GB RAM under Windows 7 to perform the experiments. In Table 2, the time to get the result is given in seconds with respect to the size of the goal model, (*) signifying non termination within an hour – the bound also indicates the maximum plan length the model can yield. For design time analysis, the tool seems to perform adequately well for sizes up to about fifty nodes. Note also the dependency of the performance on the maximum plan length. We are optimistic that these times will improve in the future as more research is already taking place on the matter of reasoning performance (e.g. [16]). Furthermore, as shown in [17], dealing with heavy computation times that solving hard problems entails can be effectively dealt with by breaking the large problem into smaller independent sub-problems. It is important to point that the presence of a DT-Golog program effectively restricts the state space to a subset that is meaningful for the domain at hand. This allows DT-Golog to reason much more efficiently than e.g. a plain MDP based approach would. In the LAS case we described above, for instance, the overwhelming space of $30^{300} \cdot 2^{300}$
possible states did not prevent DT-Golog from doing useful analysis.

7 Related Work

Probabilistic analysis of requirements has been a subject for some investigation the past few years. Notable is the work by Letier and van Lamsweerde [2], in which goal structures offer the basis for structuring probability density functions that constitute a measure of achievement of certain non-functional objectives. Genetic-algorithm based reasoning was further proposed to allow for selecting static solutions that optimize such measures [18]. Recently, these ideas were applied for supporting obstacle analysis [7]. Our framework is different in a number of ways including that it focuses on agent action and dynamic aspects of the solutions (plans) in addition to choices in the goal hierarchy and that it systematically integrates separate measures of preference, priority, utility and probability in a semi-formal manner.

Probabilistic model checking with MDPs has been proposed in PRISM [19] and successfully used in a variety of applications – albeit not yet in the context of goal modeling. One fundamental difference between the model checker and DT-Golog that makes the later more suitable for our particular purpose is the fact that DT-Golog readily allows us to specify complex actions as programs and evaluate alternative designs, which is crucial for requirements analysis. Thus, DT-Golog goes beyond the classic MDP approach, where only primitive stochastic actions are allowed and not programs composed from such actions. Other approaches for dealing with uncertainty in requirements engineering have focussed on self-adaptive systems and follow a fuzzy logic based approach [20, 21]. In comparison, we model probability and utility as separate measures, and focus on automated reasoning about optimal behaviours, in terms of either or both those measures. In addition, a wealth of proposals exist for reasoning about goal models [22]. In that line of work, however, whenever dynamic aspects of the domain are considered, analysis is deterministic and does not take uncertainty of action into account.

8 Concluding Remarks

We presented a combined action- and decision-theoretic framework for reasoning about alternative designs within requirements goal models. The framework is based on the recognition that optimal solutions for fulfilling stakeholder goals will not necessarily be executed as planned, but may fail due to human or system error or other unknown factors. Therefore, to allow for pragmatic design-time analysis, we must take uncertainty into account. This calls for rethinking the semantics of standard goal models that is used for reasoning about alternatives. The main contributions of this paper towards those directions are an approach to probabilistically extend goal models to allow for modeling agent actions with uncertain effects, as well as a way to translate them into a formal specification language that allows for evaluating alternative designs based on utility optimization. We also show how detailed analysis can be performed using this toolset. Differences of our proposal from the work already done in the area include
a strong focus on dynamic/behavioural aspects of solutions (i.e. plans, sequences of tasks) and allowing exploration of the interplay between preference, utility and probability.

For the future, we wish to work on the core of the DT-Golog reasoner to also allow searching for local optima with respect to probability thresholds, effectively allowing trade-offs between probability and expected utility. Further, empirical assessment of the reliability and accuracy of precise DT-Golog analysis (and the effort investment it takes) seems to be a priority. Scalability is an issue to be investigated in such a context. In terms of scalability of the modeling process, our current sense is that, due to the modularity of the probability and utility specification process (each task and soft-goal has its own table), larger goal models should easily accommodate definition of effects and utilities. In terms of scalability of the automated reasoning, our early results are encouraging for small-to-medium practical models. Nevertheless, we still need to explore solutions with larger models, such as breaking the problem into subproblems and reasoning about each separately.

References


21
