Representing and Reasoning with Preference Requirements Using Goals (revised)
No: CSRG-542

Sotirios Liaskos
School of Information Technology
York University
liaskos@yorku.ca

Sheila McIlraith
Dept. of Computer Science
University of Toronto
sheila@cs.toronto.edu

John Mylopoulos
Dept. of Computer Science
University of Toronto
jm@cs.toronto.edu

February 11, 2009

Contents

Abstract

We introduce a goal-based framework for representing and reasoning with preference and optional requirements. Temporally extended goal models are used for representing large numbers of alternative plans by which stakeholder goals can be fulfilled. Stakeholder preferences are then specified as weighted rankings over optional high-level characteristics of such plans. Well-studied algorithms and tools for preference-based planning are appropriately adapted and used to search the space of alternative plans and identify those that best fit the specified preferences. This way, priorities over the high-level desires of stakeholders can be used to explore solution configurations that are most suitable for them in given situations and contexts. We also explore ways by which preference formulae can be written using high-level structured English and discuss our experiences from our application at the health-care domain.

1. Introduction

Requirements are traditionally understood as statements that describe conditions over states and events in the world ([17]). These conditions are assumed to be desired by the stakeholders. Thus, posing a requirement statement is a way to imply that the stakeholder prefers states of the world in which the requirement is satisfied over states in which it is not satisfied.

Therefore, requirements can always be seen as *preferences*. For instance, assume we are analyzing the requirements for a meeting scheduler and, regarding the process of, for example, deciding the exact meeting time and place, we come up with the requirement *System to Choose Time and Place*. This means that a state of the world in which the system has decided the time and place of the meeting is preferred from a state of the world in which the system hasn't done that or a state in which the system does not even exist. From a point of view, this approach to requirements modeling assumes that stakeholders envision in an "all-or-nothing" manner in which there is nothing between something being appropriately fulfilled and the same thing being unfulfilled. Thus, in our example, either the system will choose the time and place or there is no other solution that will satisfy the stakeholder.

It is obvious, however, that stakeholder preferences are rarely so absolute. Stakeholders often prefer states of the world in which something is true over states of the world in which *something else* is true, the latter being also desired but to a lesser degree. In the previous example, the requirement for *System to Choose Time and Place* is, as we saw, preferred from its negation, but it may also be preferred from the requirement *Secretary to Choose Time and Place*. There may even be more options such as *Participants Collectively Pick a Time and Place over E-mail* or *Busiest Participant Decides Time and Place*. Each stakeholder may find each of these options attractive to a different degree, implying a preference of each one over each of the others.

Such preferences can also be posed over higher level desires of stakeholders. The goals *Ensure Scheduling Reliability* or *Keep Secretary Unburdened* are examples of such high-level desires. In different situations, different stakeholders may assume that one is more important than the other. Furthermore, the way by which stakeholders pose their preferences over such high-level desires influences the choice of the design that will satisfy these desires. Having the secretary choose the time and place of the meeting is more reliable than having the system to do so, but comes at a cost of burdening the secretary. Thus, alternative assertions about the relative importance of scheduling reliably over keeping an unburdened secretary imply alternative designs of the time and place selection process.

While prioritization and decision making in requirements engineering has been well studied, both the problem of *modeling* stakeholder priorities and preferences and the use of such models to *reason* about alternative designs has not been the focus so far. In this paper, we introduce a framework for both specifying requirements preferences and for using them for selecting behavioral designs that best fit the priorities of stakeholders. We begin by building on our previous work on goal-oriented variability modeling and propose a temporally extended goal-modeling language which allows representation of a great number of alternative system behaviors that can fulfill the same stakeholder goal. We then present a formal language for specifying preferences. The language is based on the construction of desire formulae in Linear Temporal Logic (LTL) and their subsequent use in weighted orderings depending on their relative importance. Then, we present a tool for selecting behavioral designs that best satisfy the specified preferences.

We organize our presentation as follows. In Section 2 we discuss related work. Section 3 provides a motivating example. Sections 4, 5 and 6 describe our modeling formalism and its semantics. Then, in Section 7, we describe the preferences language and its semantics and in Section 8 we show how reasoning about preferences is possible. In Section 9 we provide a modeling and reasoning alternative to behavioral analysis that is independent of time. We discuss how the tools for performing both types of analysis (Section 10) are implemented and how they perform. In Section 11 we show how preferences can be generated through higher level preference elicitation techniques. We provide our early feasibility evidence in Section 12 and conclude in Section 13.

2. Related Work and Background

The need for a view of requirements that explicitly takes attitudes, preferences and optionality into account has recently been illustrated by Jureta et al. in [19] through reference to the nature of the linguistic matter that serves the communication between stakeholders and analysts. The traditional notion of requirements prioritization originates exactly from the observation that not all requirements have the same importance for all stakeholders. An elementary requirements prioritization approach, for example, is to divide requirements into “must-haves” and “nice-to-haves”, whereby the former are understood as more important, urgent or otherwise of higher priority (e.g. [3]). In addition to this common qualitative approach, more elaborate quantitative prioritization techniques, such as the Analytic Hierarchy Process ([20, 2]) or multi-criteria preference analysis methods ([16]) have also been proposed and successfully used in practice. The use of multi-attribute decision theoretic approaches has also been explored, e.g. in [21].

The modeling and reasoning side of prioritization, however, has not received as much attention in requirements engineering. Instead, researchers have mostly been focusing on modeling requirements variability (e.g. [14, 8, 10, 32]), without including in their scope the problem of selecting requirements variants according to given stakeholder priorities. The limited number of efforts that do attempt reasoning about variability subject to given criteria focus, in most cases, on identifying combinations of coarse grained features of the system-to-be rather than behavioral details that derive from the stakeholder goals. In [36], for instance, the use of Bayesian belief networks for capturing the impact of low level feature-model based configuration decisions is proposed. This approach does not take into account behavioral properties of potential solutions nor does it introduce a method for searching for solutions given desired values in the Bayesian network. In [5], on the other hand, direct manipulation of feature models is proposed through staged variability binding. Closer to our proposal, a method for scenario generation from generic use-cases, proposed in [31], introduces a constraint language for selecting scenario instances. However, that language is a constraint language rather than a preference specification one and it is generally geared towards solution-oriented use-cases rather than stakeholder goals and partial satisfaction thereof.

The idea of specifying criteria for selecting among a large space of designs has also been studied in the *product configuration* community. In product configuration technologies ([28] for a survey) the typical solution is to construct a (generic) product model with a great number of degrees of freedom ([27]), accompany it by an infrastructure for describing individual requirements and constraints, and introduce an inference engine to search for configurations that satisfy both the generic and individual model (e.g. [9]). Preferences over predefined low-level decision points ([18]) and evaluation based on impact of decisions to high level qualities of the result ([1, 27]) have been proposed. Along the same lines, Zhang et al. propose the use of Bayesian belief networks for understanding the impact of low level feature-model based configuration decisions ([36]). None of these proposals supports the definition of constraints over temporal characteristics of admissible behavior; something that we introduce in this paper at a stakeholder goal level.

Goal models ([6, 34]) have been found to be effective in concisely capturing large numbers of alternative sets of low-level tasks, operations, and configurations that can fulfill stakeholder goals. The capture of a large space of such alternatives has been shown to be useful for exploring alternative designs during the analysis process ([25]), for customizing designs to fit individual user characteristics ([15]), or even for coping with the vast space of configurations that common desktop applications offer to users ([22]). An interesting feature of goal variability analysis is the ability to assess the impact of each concrete goal alternative to the satisfaction of more abstract goals that stakeholders pose. Thus, in [13], Giorgini et al. propose a formal approach that allows bottom-up assessment of satisfaction of high-level goals based on evidence about the fulfillment of low-level operational ones. We will discuss this proposal in more detail below. In addition to this bottom-up framework, top-down propagation of satisfaction values has also been introduced in [29]. In that work, an algorithm for deciding satisfaction configurations of low level goals given desired satisfaction values for high-level goals is given. Although our motivation is similar, our approach introduces some important possibilities, including that, again, we consider a language for specifying preferences versus hard constraints and that we focus on behaviors, i.e. sequences of goals and tasks, rather than plain sets thereof.

3. Motivating Example

To see how the need to model goal variability and user preferences over alternatives emerges during early requirements elicitation processes, we consider an example from the health-care domain, where we tried the ideas discussed in this paper. The context of the application is a geriatric assessment unit, where elderly with a variety of health issues are hospitalized for a period of time. The primary objective of the application is to increase the efficiency of the nursing activities by appropriately assisting nurses with their assigned tasks. In our example, we analyze the case where a patient needs to be attended to by a nurse due to an event. For instance, the patient may be trying to get up even though she is not allowed to due to her health condition, or she may have called the nurse herself to ask a question, or to request additional medicine. The nurse needs to be notified somehow, either through a broadcasted notification using the speakers of the unit, or through earphones he wears while on duty. Then, the nurse's reaction needs to be determined. Normally, he has to visit the patient's room, but if the patient only wants to ask a question or request permission for something, the visit may be replaced by establishing a voice link between patient and nurse. For example, the nurse may be carrying a mobile set with microphone and earphones, or there may be a device at the nursing station, which is conveniently located in the unit. The nurses think that this would increase unnecessary disturbance from some patients, but they acknowledge it would also increase their productivity, and save them from extra walking effort.

All these are alternative behavioral designs that need to be evaluated subject to criteria posed by individual stakeholder and context instances. Different geriatric assessment units, different stakeholders in the same unit or even the same stakeholders in different times and situations, may have different priorities over high-level characteristics of the desired solution. For example, in a particular unit the nurses may state that *"[they] don't like the idea of talking to the patient remotely, but if they had to, they would choose to do so at the nursing station.*". The managers of the unit, on the other hand, will use a more high-level language: *"we should definitely avoid anything that would make the patient unhappy, but it would also be nice to increase nurses' productivity somehow.*" How can we translate these statements into a selection of behavioral designs that best satisfy them? In this paper we attempt an approach to this problem. In the following section, we start by looking at the goal modeling language that can help us represent the various alternatives.

4. Goal Models

4.1. Overview

The goal modeling language we will use adopts the basics of existing goal modeling notations (particularly *i** - [34]) and is extended in order to accommodate quantitative analysis of goal satisfaction (adopting [13]), temporal constraints (similar to [12] and [33]) as well as variables describing the environment. More specifically, our goal model consists of:

1. a set of *hard-goals* H,
2. a set of *soft-goals* L,
3. a set of *tasks* T,
4. a set of *domain concepts* O.
5. a set of *domain predicates* R, which represent relations over domain concepts.

Goals are states of affairs or conditions that one or more actors of interest would like to achieve ([35]). Hard-goals are goals for which there is a clear-cut criterion to decide whether they are satisfied or not. For soft-goals, instead, such a criterion does not exist; soft-goals are satisfied to a "good enough" degree, depending on subjective judgment and based on relevant evidence. Thus, *Have Nurse Notified* is an example of a hard-goal, while *Happy Patient* is a typical soft-goal. Tasks, on the other hand, describe particular activity that the actors perform in order to fulfill their goals, e.g. *Send Audio Notification*. We use the *satisfaction predicates issat(g)* and *isperformed(t)* to denote that a hard-goal g or a task t has been satisfied or performed, respectively. For example *isperformed*('Send Audio Notification') means that the task *Send Audio Notification* has been performed. Domain facts express ways by which domain concepts, such as *nurse*, *nursingStation*, *english*, *printer*, relate to each other at a particular time instance and while actors are performing tasks to fulfill their goals. Examples of domain facts are *isAt(nurse, nursingStation)*, *isAvailable(nursingStation, printer)* and *speaks(patient, english)*. The truth value of domain facts may or may not change due to the performance of tasks.

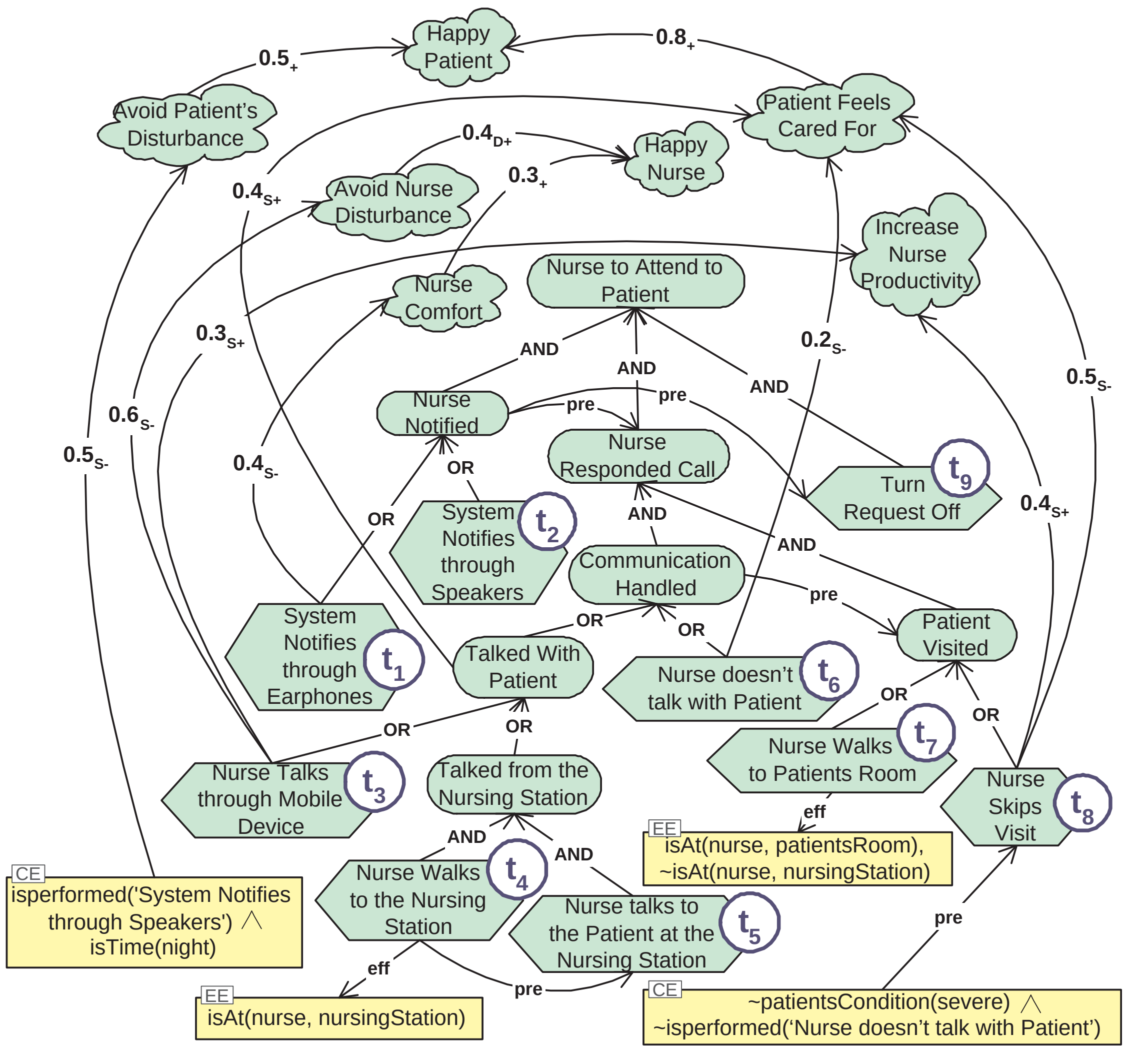
Using the domain facts together with 0-ary predicates that describe tasks and goals we can construct simple first-order formulae, which we will call *condition formulae*.

Definition 4.1 (Condition Formula - CF) A condition formula ϕ is drawn from a set K for which:
1. $R \subset K$
2. if $g \in H$ then *issat(g)*$\in K$
3. if $t \in T$ then *isperformed(t)*$\in K$
4. if $\phi, \phi_1, \phi_2 \in K$ then so do: $\neg\phi$, $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$.

A CF is understood in the context of a course of activities that aims at fulfilling a root goal. Predicates that represent tasks (respectively goals/domain facts) are true if and only if the respective task (goal/fact) has been performed (is satisfied/is true) at a given time instance, while the actor is active in order to fulfill the root goal. For example, *issat*('Nurse Notified') $\wedge$ *isAt(nurse, nursingStation)* is true at a given point in time if the goal *Nurse Notified* has been satisfied and the nurse is at the nursing station at that point.

Figure 1 shows how the above are represented diagrammatically. Each oval-shaped element represents a goal and each cloud-shaped element represents a soft-goal. Hexagonal elements represent tasks. There are also two types of rectangle-shaped elements: the condition elements (CE) and the effect elements (EE), each containing a CF and a list of effects, respectively. Further, to be concise in the rest of the paper, we have annotated each task in Figure 1 with a literal of the form t_i. In the rest of the paper, we will refer to each task using the corresponding literal. Thus, t_8 refers to the task *Nurse Skips Visit* and t_9 to *Turn Request Off*. Also, to ease our presentation, reference to goals in CFs (and all other types of formulas we will introduce) is done through quoting the exact informal title of the goal as seen in the model, instead of introducing special types of identifiers; for example, we use *isperformed*('System Notifies through Speakers') instead of e.g. *isperformed(systemNotifiesThroughSpeakers)*. Finally, in the Figure, annotations have been added to distinguish between different types of elements, although these are easily distinguished by the type of links by which they connect to the rest of the graph, as it will become apparent below. To further ease our presentation we will also use $\overrightarrow{p_e(\vec{o})}$ to denote the list of domain facts $p_e(\vec{o})$ contained in EE e.

Hard-goals and tasks form a decomposition tree and together with all EEs and some CEs form the *hard-goal subgraph*. Soft-goals, on the other hand, as well as the rest of the CEs, some goals and some tasks, form their own directed and *acyclic* subgraph, the *soft-goal subgraph*. The two subgraphs are connected through contribution links that originate from tasks and goals of the hard-goal graph and target soft-goals of the soft-goal graph. These tasks and goals are also considered to be parts of the soft-goal graph. The two sub-graphs have distinct functions in our framework. The hard-goal graph allows us to represent alternative ways by which a root hard-goal can be satisfied (e.g. different ways to have the *Nurse Notified*), whereas the soft-goal graph allows us to assess how each alternative affects high-level quality goals of the stakeholders (e.g. how different ways to have the *Nurse Notified* affect the soft-goal *Patient's Privacy*).



Happy Patient
0.5+
0.8+
Avoid Patient's Disturbance
Patient Feels Cared For
0.4D+
Happy Nurse
0.3+
0.4S+
Avoid Nurse Disturbance
Increase Nurse Productivity
Nurse Comfort
Nurse to Attend to Patient
0.3S+
0.2S-
AND
AND
AND
0.5S-
0.6S-
Nurse Notified
pre
pre
Nurse Responded Call
0.5S-
0.4S-
OR
Turn Request Off
t9
0.4S+
OR
System Notifies through Speakers
t2
AND
AND
Communication Handled
pre
System Notifies through Earphones
t1
OR
Talked With Patient
OR
Patient Visited
Nurse doesn't talk with Patient
t6
OR
OR
OR
Nurse Talks through Mobile Device
t3
Talked from the Nursing Station
Nurse Walks to Patients Room
t7
OR
Nurse Skips Visit
t8
AND
eff
EE
isAt(nurse, patientsRoom), ~isAt(nurse, nursingStation)
AND
Nurse Walks to the Nursing Station
t4
CE
isperformed('System Notifies through Speakers') ∧ isTime(night)
Nurse talks to the Patient at the Nursing Station
t5
pre
eff
pre
EE
isAt(nurse, nursingStation)
CE
~patientsCondition(severe) ∧ ~isperformed('Nurse doesn't talk with Patient')

Figure 1. A goal model

4.2. The hard-goal subgraph

We now present the hard-goal graph in more detail. Its backbone is an AND/OR decomposition tree which consists exclusively of hard-goals and tasks. Leaf level nodes are only tasks, and tasks can only be leaf level nodes of the hard-goal graph. The decomposition tree represents alternative ways by which its root g_r can be satisfied. When a goal g is AND-decomposed into goals or tasks $g_1, \ldots, g_n$ then g is satisfied iff g_i are satisfied (performed in the case of tasks) for all i. If g is OR-decomposed, then g is satisfied iff there exists an i such that g_i is satisfied (performed if it is a task). Thus, any AND/OR decomposition tree rooted at g_r implies a set of subsets of T that are capable of satisfying the root goal g_r. We call these *alternatives* for g_r.

Two additional types of links are associated with the hard-goals graph. The first one is the *precedence constraint link* that is applied in three ways:

At the leaf level, $\phi \xrightarrow{pre} t$, where ϕ a CF and t a task, means that t can be preformed only if ϕ is true at the time when performance of t is attempted.

Between AND-subgoals, $g_1 \xrightarrow{pre} g_2$, means that no task that is part of g_2's subtree can be performed unless a set of tasks that constitutes an alternative for g_1 has already been performed.

The second type of links, the *effect links* $t \xrightarrow{eff} e$ and, respectively $t \xrightarrow{nef} e$, are applied from a task t to an EE e and imply that completion of performance of the former instantly causes all facts $p_e(\vec{o})$ contained in the latter to become true (respectively, false).

4.3. The Soft-goal subgraph

Soft-goals, as well as some CFs, some hard-goals and some tasks form their own sub-graph, the *soft-goal subgraph*. Each such element *l* that participates in the soft-goal graph is assigned two variables: *valS(l)* which represents the degree by which we believe the element is satisfied and *valD(l)* which represents the degree by which we believe the element is denied. As we will see, the domain of values of each such variable depends on the type of element: CFs, hard-goals and tasks can not be partially satisfied and are therefore treated differently from soft-goals.

The elements of the soft-goal graph are exclusively connected through weighted contribution links. The contribution links are drawn between soft-goals, or from hard-goals, tasks or CFs to soft-goals. Thus, CFs, hard-goals and tasks can only be sources in the soft-goal graph. The links show how the satisfaction and denial evidence of their source can influence our knowledge of satisfaction or denial of its destination. In general, the domain of satisfaction and denial variables as well as the type of contribution links depend on the representation granularity we wish to achieve. In this thesis, we follow the Giorgini et al. modeling and evaluation framework ([13]) which offers two representation alternatives: a *qualitative* and a *quantitative* one. Below, we detail the specific modeling rules for each as well as their intuitive semantics.

4.3.1 Qualitative Modeling Framework

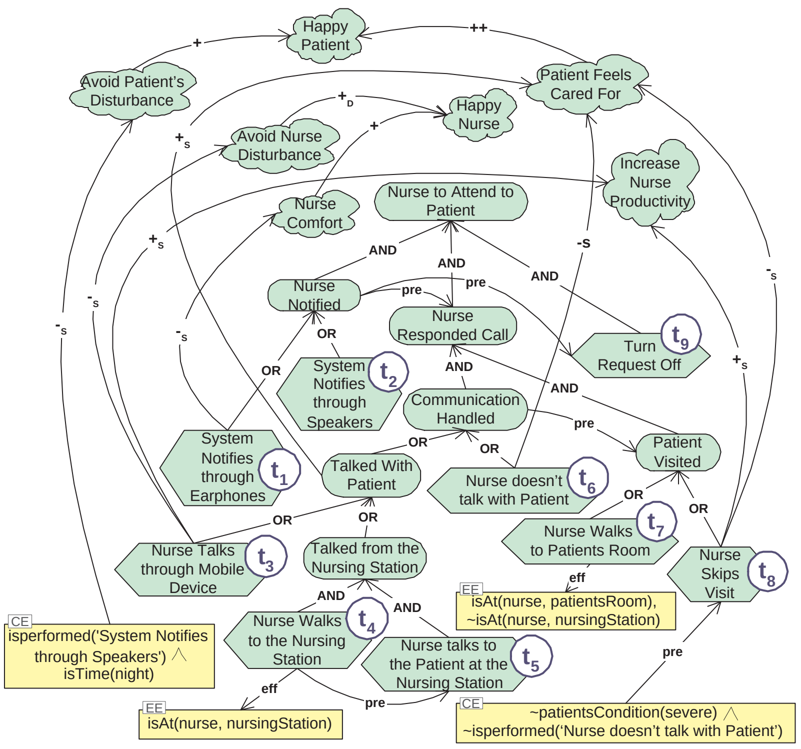
In qualitative modeling of soft-goal satisfaction propagation, the variable $valS(\cdot)$ (respectively $valD(\cdot)$) take values in the domain $\{F, P, N\}$, which mean **F**ull satisfaction (resp. denial), **P**artial satisfaction (resp. denial) or **N**o evidence of satisfaction (resp. denial) at all, respectively. It is assumed that these three values are totally ordered: $F > P > N$. For, CFs, hard-goals and tasks, for which partial satisfaction/performance is not defined, the domain is restricted to the values F and N. Thus, if *valS* ('Avoid Nurse Disturbance') equals P, this means that the goal *Avoid Nurse Disturbance* is partially satisfied. If *valD*('Happy Patient') equals F, this means that the respective soft-goal is known to be fully denied. However there cannot be such thing as *valS rm ('Nurse Notified')* equals P, as the goal *Nurse Notified* is either known to be fully satisfied or not; it cannot be "almost" satisfied or satisfied "to some extend".

Furthermore, there are eight types of contribution links between two elements l_1 and l_2 of the soft-goal graph, seen in Table 1. In the Table, the subscripts S and D, represent whether it is the satisfaction or the denial of l_1 that is influencing l_2, respectively. The sign of the propagation shows whether the link implies contribution to the satisfaction or the denial of l_2, depending on whether it is positive $+/++$ or negative $-/--$ respectively. The number of signs, one ($+/-$) versus two ($++/--$), show weak and strong influence, respectively. A qualitative version of the goal model of Figure 1 is given in 2. In the figure, *Talked with Patient*$\xrightarrow{+s}$*Patient Feels Cared For,* means that satisfaction of the goal to have h

Weak Contributions	Strong Contributions
$l_1 \xrightarrow{+_S} l_2$	$l_1 \xrightarrow{++_S} l_2$
$l_1 \xrightarrow{-_S} l_2$	$l_1 \xrightarrow{--_S} l_2$
$l_1 \xrightarrow{+_D} l_2$	$l_1 \xrightarrow{++_D} l_2$
$l_1 \xrightarrow{-_D} l_2$	$l_1 \xrightarrow{--_D} l_2$

Table 1. Qualitative Propagation Links

the nurse talk somehow with the patient partially helps to satisfy the goal to have patients feel cared for. *Avoid Nurse Disturbance*$\xrightarrow{-_D}$*Happy Nurse* means that denial of the goal to avoid the nurses being disturbed strongly hurts the goal to keep them happy. Absence of the subscript S or D implies that both possibilities are in effect. Thus, $l_1 \xrightarrow{+} l_2$ implies both $l_1 \xrightarrow{+_S} l_2$ and $l_1 \xrightarrow{+_D} l_2$.



Happy Patient
Avoid Patient's Disturbance
Patient Feels Cared For
Happy Nurse
Avoid Nurse Disturbance
Increase Nurse Productivity
Nurse Comfort
Nurse to Attend to Patient
Nurse Notified
Nurse Responded Call
Turn Request Off
System Notifies through Speakers
Communication Handled
Patient Visited
System Notifies through Earphones
Talked With Patient
Nurse doesn't talk with Patient
Nurse Walks to Patients Room
Nurse Talks through Mobile Device
Talked from the Nursing Station
Nurse Skips Visit
Nurse Walks to the Nursing Station
Nurse talks to the Patient at the Nursing Station
t1
t2
t3
t4
t5
t6
t7
t8
t9
+
++
+D
+S
-S
AND
OR
pre
eff
CE
EE
isperformed('System Notifies through Speakers') ∧ isTime(night)
isAt(nurse, patientsRoom), ~isAt(nurse, nursingStation)
isAt(nurse, nursingStation)
~patientsCondition(severe) ∧ ~isperformed('Nurse doesn't talk with Patient')

Figure 2. A goal model with qualitative labels

The satisfaction and denial value of a soft-goal depends on the satisfaction and denial values of all soft-goal graph elements that contribute to that goal through a link. More specifically, given a contribution link from element l_1 to soft-goal l_2, $valS(l_2)$ and $valD(l_2)$ are determined by the corresponding $valS(l_1)$ and $valD(l_1)$ values as well as the type of the contribution link, as shown in Table 2. We will later discuss the case of multiple contribution links targeting the same soft-goal.

Contribution	$valS(l_2)$	$valD(l_2)$
$l_1 \xrightarrow{+_S} l_2$	$min\{valS(l_1), P\}$	N
$l_1 \xrightarrow{++_S} l_2$	$valS(l_1)$	N
$l_1 \xrightarrow{-_S} l_2$	N	$min\{valS(l_1), P\}$
$l_1 \xrightarrow{--_S} l_2$	N	$valS(l_1)$
$l_1 \xrightarrow{+_D} l_2$	N	$min\{valD(l_1), P\}$
$l_1 \xrightarrow{++_D} l_2$	N	$valD(l_1)$
$l_1 \xrightarrow{-_D} l_2$	$min\{valD(l_1), P\}$	N
$l_1 \xrightarrow{--_D} l_2$	$valD(l_1)$	N

Table 2. Qualitative Contribution Links

4.3.2 Quantitative Modeling Framework

The quantitative framework allows more fine-grained analysis of satisfaction/denial propagation by using real numbers instead of labels N, P and F. Thus, the domain of the variables $valS(\cdot)$ and $valD(\cdot)$ is the set of real numbers in the interval $[0, 1]$. Again, however, specifically for CFs, hard-goals and tasks, the domain is restricted to the values 0 and 1. Thus if *valD*('Happy Patient') equals 0.4, the number implies the degree by which the respective soft-goal is known to be denied. Again, there cannot be such thing as e.g. *valS* ('Nurse Notified')=0.7, as the goal *Nurse Notified* is either known to be fully satisfied or not; therefore *valS* ('Nurse Notified') can be either 1 or 0.

	Contributions of Satisfaction	Contributions of Denial
Contributions to Satisfaction	$l_1 \xrightarrow{w_{S+}} l_2$	$l_1 \xrightarrow{w_{D-}} l_2$
Contributions to Denial	$l_1 \xrightarrow{w_{S-}} l_2$	$l_1 \xrightarrow{w_{D+}} l_2$

Table 3. Quantitative Propagation Links

The contribution links we use when modeling for quantitative analysis can be seen in Table 3. Intuitively, $l_1 \xrightarrow{w_{S+}} l_2$ (respectively, $l_1 \xrightarrow{w_{S-}} l_2$), denotes that the satisfaction (respectively, denial) of l_2 is understood to be equal to l_1's satisfaction factored by w. Similarly, $l_1 \xrightarrow{w_{D+}} l_2$ (respectively, $l_1 \xrightarrow{w_{D-}} l_2$), denotes that the denial (respectively, satisfaction) of l_2 is calculated as a proportion of l_1's denial. Again, the value of $valS(l_2)$ and $valD(l_2)$, depending on the respective values of l_1 and the type of contribution link from l_1 to l_2, are decided based on rules which are shown in Table 4. Note that, while [13] discusses several possibilities for interpreting quantitative propagation, Table 4 reflects the probabilistic approach. More details on other interpretations can be found in [13]. Again, as in the qualitative case, we omit the subscript S or D to denote coexistence of links of both satisfaction and denial. Thus, $l_1 \xrightarrow{w_+} l_2$ implies both $l_1 \xrightarrow{w_{S+}} l_2$ and $l_1 \xrightarrow{w_{D+}} l_2$.

While the quantitative framework is more expressive and allows fine-grained expression of how goals influence each other's satisfaction, it appears to be less popular than the qualitative one, in that it poses the difficulty of assessing the contribution weights. In Section 11 however we will see that we may use

Contribution	$valS(l_2)$	$valD(l_2)$
$l_1 \xrightarrow{w_{S_+}} l_2$	$w \times valS(l_1)$	
$l_1 \xrightarrow{w_{S_-}} l_2$		$w \times valS(l_1)$
$l_1 \xrightarrow{w_{D_+}} l_2$		$w \times valD(l_1)$
$l_1 \xrightarrow{w_{D_-}} l_2$	$w \times valD(l_1)$	

Table 4. Quantitative Contribution Links

the flexibility of the quantitative approach without compromising usability, through the construction of mappings from qualitative characterizations of satisfaction and contribution to numerical values. Thus, in our study, emphasis has been given to the quantitative propagation framework as it is more powerful and technically challenging.

5. Label Propagation

The purpose of introducing the propagation rules of Tables 2 and 4, apart from providing an intuition of what satisfaction/denial contribution means, is that it allows us to reason about satisfaction or denial of certain soft-goals in our soft-goal subgraph based on evidence that we have about the satisfaction or denial of the others. In [13], Giorgini et al. introduce such an algorithm, called the *label propagation (LP)* algorithm. Starting from initial satisfaction and denial values for goals that are sources to the graph, the LP algorithm iterates over the propagation rules until convergence for the satisfaction/denial degrees of all goals is reached. At each iteration, when a soft-goal is a target of many contribution links, from all potential satisfaction and denial values (including the existing ones), the maximum one (by absolute value) is selected to be the new value. We sketch this algorithm in Figure 4, which has been adapted from [13].

We introduce an adapted version of the label propagation algorithm presented in [13], which assumes that the soft-goals graphs do not contain directed cycles. We identify our algorithm as ALP (LP for Acyclic goal models). The additional acyclicity assumption allows us to change the original Label Propagation algorithm in a way that guarantees convergence within one iteration. To achieve this, for each goal node we calculate its *depth*, that is the maximum path length for reaching the node from any of the sources of the graphs (which are all hard-goals, tasks or CEs in our case). In Figure 3 the soft-goal subgraph of the goal graph of Figure 1 is shown, where each node is annotated with a number indicating its depth. Thus l_5 has a value of 3, which is the length of the path from t_1 or t_2.

Hence the ALP algorithm includes three changes, compared to the LP. Firstly the label updates are (partially) ordered by maximum path length ascending. Thus, in the Figure 3, l_1 and l_2 are evaluated first (in any order), l_3 next, and last l_4 and l_5, again the last two in any order. Secondly, the update does not (need to) take into account the current label of each node. Therefore, thirdly, only one iteration is needed.

In Figure 5, the pseudocode describing the algorithm is given, next to the original one presented in [13]. In the Figure, $withdepth(G, d)$ returns a set of nodes whose depth equals d or $NULL$ if no nodes of such depth exist. Also, $Label$ denotes a pair of satisfaction and denial values, a $LabelSet\ C$ is a set of such Labels, C_g denotes the Label in C that is associated with goal g, and $candS^i_g$ and $candD^j_g$, are

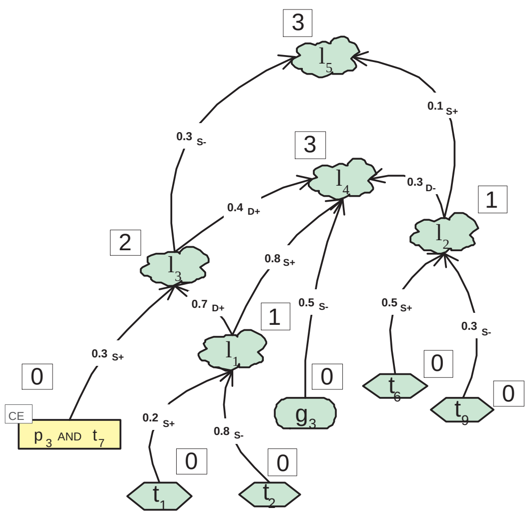


Figure 3. Maximum Path Lenght

arrays of candidate satisfaction and denial values, respectively, for goal g.

6. Goal Semantics in Situation Calculus

While we provided some rules for understanding and calculating propagation of satisfaction of soft-goals, we still need to provide semantics for the combined goal tree, which includes both the hard-goal subgraph where a family of sequences of possible leaf level tasks is modeled and the hard-goal subgraph which models how the performance of tasks influences the satisfaction of soft-goals. Thus, we appeal to the situation calculus to define the semantics of our goal language, which enables us to easily exploit existing algorithms and tools for preference-based planning for the purpose of evaluating goal-level preferences.

6.1. Situation Calculus

The situation calculus is a logical language for specifying and reasoning about dynamical systems [26]. In the situation calculus, the *state* of the world is expressed in terms of functions and relations (fluents) relativized to a particular *situation* s, e.g., $f(\vec{x}, s)$. A situation s is a *history* of the primitive actions, $a \in \mathcal{A}$, performed from a distinguished initial situation S_0. The function $do(a, s)$ maps a situation and an action into a new situation thus inducing a tree of situations rooted in S_0. The predicate Poss(a,s) is true if action a is possible in situation s.

A basic action theory, comprises the domain-independent foundational axioms of the situation calculus, successor state axioms, precondition axioms, axioms describing the intial state of the system, unique names axioms for actions and domain closure axioms for actions. $\mathcal{D}$ may also include some state constraints, such as ramification axioms or definitional axioms for fluents. Given a goal formula G, a *plan* in the situation calculus is a sequence of actions $\vec{\alpha} = \alpha_1, \alpha_2 \ldots, \alpha_n$ such that for the situation $s = do(\alpha_n, \ldots, do(\alpha_1, S_0))$, G holds in s and the precondition axioms are satisfied throughout $\vec{\alpha}$.

```
LabelSet Label_Graph(GoalGraph $G$, LabelSet $Initial$)
    $Current = Initial$;
    do
        $Old := Current$;
        for each $g$ in $G$
            $Current_g := UpdateLabel(g, Old)$;
    until $(Current == Old)$;
    return $Current$;

Label UpdateLabel(SoftGoal $g$, LabelSet $Old$)
    for each node $g_i$ that contributes to $g$
        $candS_g^i = Apply\_Rules\_Sat(g, g_i, Old)$;
        $candD_g^i = Apply\_Rules\_Den(g, g_i, Old)$;
    return $\langle max\{max_i(candS_g^i), Old_g.valS\},$
        $max\{max_i(candD_g^i), Old_g.valD\}\rangle$
```

Figure 4. Label Propagation

```
LabelSet Label_Propagation(GoalGraph $G$)
    $LabelSet\ C = NULL$;
    $LabelSet\ Res = NULL$;
    $int\ depth := 1$;
    $C := withdepth(G, depth)$;
    repeat
        for each $g$ in $C$
            $C_g = UpdateLabel(g)$;
        $Res := Res \cup C$;
        $depth := depth+1$;
        $C := withdepth(G, depth)$;
    until $(C == NULL)$
    return $Res$;

Label UpdateLabel(SoftGoal $g$)
    for each node $g_i$ that contributes to $g$
        $candS_g^i = Apply\_Rules\_Sat(g, g_i)$;
        $candD_g^i = Apply\_Rules\_Den(g, g_i)$;
    return $\langle max_i(candS_g^i), max_i(candD_g^i)\rangle$
```

Figure 5. Adapted Label Propagation

The details of $\mathcal{D}$ are described in [26]. In the section that follows, we show how to translate our goal model into a basic action theory, $\mathcal{D}$.

6.2. Translating the Goal Model

We now present the semantics of our visual goal language via a set of translation rules. Similar translation proposals are introduced in [12] and [33], but for different purposes; a distinguishing feature of our approach is the consideration of soft-goals as part of the translation. We first establish a mapping from the primitives of the goal based graphical language to those of the situation calculus:

6.2.1 Primitives

- For every task t that appears in the goal model introduce an action α_t and a relational fluent *performed(t,s)* in the situation calculus domain theory.
- For every goal g introduce an AND/OR formula $\varphi_g(s)$ of predicates of the type *performed(t,s)*. The formula is constructed as follows. Starting from g, each goal is recursively replaced by the conjunction or disjunction of its children, depending on whether g is AND or OR decomposed. If these subgoals are tasks, then the predicate *performed(t,s)* is used and the recursion terminates.
- For every domain predicate $p(\vec{o})$ introduce a relational fluent $f_p(\vec{x}, s)$, where $\vec{x}$ are individuals representing domain concepts $\vec{o}$.
- Use individuals (constants) r_t, r_g, r_l and r_e, to identify a task t, a goal g, a soft-goal l and a CE e that are part of the soft-goals graph. Also, let P_T, P_H, P_L and P_E respectively be the set of all

such individuals and P their union. Then define fluents $v_{\mathsf{s}}(r,s,w)$ and $v_d(r,s,w)$, where r is an individual in P, i.e. represents a node in the soft-goal graph. Thus, these fluents represent respectively the satisfaction and denial degree w of soft-goal graph node r in situation s. Obviously, the domain of w depends on the framework of use. Thus:

Quantitative:	[0,1]
Qualitative:	{N,P,F}

Notice that CEs, tasks and hard-goals also have a satisfaction value, albeit with a restriction, as we will see later.

- Introduce the fluent $link(r_1, r_2, y, w)$, where r_1 is in P, r_2 is in P_L and $y \in \{$"S+","S-","D+","D-"$\}$. The fluent represents the weight w of the contribution link originating from r_1 targeting r_2, while y denotes the type of the link. Again the domain of w depends on the framework we are using:

Quantitative:	(0,1]
Qualitative:	{some, full}

  For example the contribution link $l_1 \xrightarrow{0.3_{D+}} l_2$ produces $link(r_{l_1}, r_{l_2}, \text{D+}, 0.3)$. On the other hand the contribution link $l_1 \xrightarrow{++_D} l_2$ produces $link(r_{l_1}, r_{l_2}, \text{D+}, full)$ and $l_1 \xrightarrow{-_S} l_2$ gives $link(r_{l_1}, r_{l_2}, \text{S-}, some)$.

- For every CF ϕ appearing in a CE, produce its translation φ into the situation calculus ontology by translating each task predicate t, goal g and domain predicate $p(\vec{o})$ mentioned in the CF to the corresponding fluent *performed(t,s)*, formula $\varphi_g(s)$, and fluent $f_p(\vec{x}, s)$.

6.2.2 Successor State Axioms

We can now construct the successor state, precondition and initial situation axioms based on the following rules. Note that $\supset$ denotes the implication connective.

- Recall that effect links connect tasks with effect elements (EEs), the latter being lists of effects, i.e. sole domain predicates. For every such effect link $t \xrightarrow{eff} e$ from a task t to an EE e and every effect $p_e(\vec{o})$ contained in that e introduce a successor state axiom of the type:

$$Poss(\alpha, s) \wedge (\alpha = \alpha_t) \supset f_{p_e}(\vec{x}, do(\alpha, s)) \quad (1)$$

  Dually, for every negative effect link $t \xrightarrow{nef} e$ and every effect $p_e(\vec{o})$ contained in that e introduce a successor state axiom of the type:

$$Poss(\alpha, s) \wedge (\alpha = \alpha_t) \supset \neg f_{p_e}(\vec{x}, do(\alpha, s)) \quad (2)$$

  In both axioms, $f_{p_e}(\vec{x}, s)$ is the situation calculus formula that results from the translation of $p_e(\vec{o})$.

  Intuitively, the axioms ensure that relationships appearing in effect elements will be enabled (or disabled accordingly) when any of the tasks that points to these elements is performed, provided that the appropriate conditions are satisfied at that time.

- For each of the fluents *performed(t,s)* introduced above, construct a successor state axiom as follows:

  $$Poss(\alpha, s) \land (\alpha = \alpha_t) \supset \textit{performed(t,s)} \quad (3)$$

  Thus, the fluent *performed(t,s)* will become true once the action associated with the task t is performed.

6.2.3 Ramification Axioms

Ramification axioms describe consequences of direct effects. In our context, the existence of indirect effects in situation calculus reflects the effect of the performance of low level tasks to the satisfaction and denial of soft-goals, which may, in turn, influence the satisfaction or denial of other soft-goals. Thus, the axioms are written in accordance to the structure of the soft-goals graph, in a way that performance of our adapted label propagation algorithm is ensured. Thus:

- For every individual $r_t \in P_T$, $r_g \in P_G$ and $r_e \in P_E$ introduce a pair of axioms that associates the value of $v_{\mathtt{s}}(r_t, s, w)$, $v_{\mathtt{s}}(r_e, s, w)$, and $v_{\mathtt{s}}(r_g, s, w)$ with formulae grounded on fluents of type *performed*($\cdot, s$). Depending on whether we are working with the quantitative or qualitative framework we respectively have:

Quantitative	Qualitative	
performed(t,s) $\supset v_{\mathtt{s}}(r_t, s, 1)$	*performed(t,s)* $\supset v_{\mathtt{s}}(r_t, s, F)$	(4)
$\neg$*performed(t,s)* $\supset v_{\mathtt{s}}(r_t, s, 0)$	$\neg$*performed(t,s)* $\supset v_{\mathtt{s}}(r_t, s, N)$	(5)
$\varphi_g(s) \supset v_{\mathtt{s}}(r_g, s, 1)$	$\varphi_g(s) \supset v_{\mathtt{s}}(r_g, s, F)$	(6)
$\neg\varphi_g(s) \supset v_{\mathtt{s}}(r_g, s, 0)$	$\neg\varphi_g(s) \supset v_{\mathtt{s}}(r_g, s, N)$	(7)
$\varphi_c(s) \supset v_{\mathtt{s}}(r_e, s, 1)$	$\varphi_c(s) \supset v_{\mathtt{s}}(r_e, s, F)$	(8)
$\neg\varphi_c(s) \supset v_{\mathtt{s}}(r_e, s, 0)$	$\neg\varphi_c(s) \supset v_{\mathtt{s}}(r_e, s, N)$	(9)

  Where φ_c is the situation calculus translation of the CF contained in a condition element c. The above formulas set the satisfaction degree of soft-goal graph nodes which are tasks, goals or CFs. Observe that we prevent partial satisfaction to such nodes.

- For every soft-goal l in the goal model, let R_{S_+} and R_{D_-} be the sets of soft-goal graph nodes $k_i \in R_{S_+}$ and $m_j \in R_{D_-}$ for which $k_i \xrightarrow{w_{S_+}} l$ and $m_j \xrightarrow{w_{D_-}} l$, respectively, where w is the respective weight.

  Let $\mathbf{z_{S_+}}$ be an abbreviation for $z^{k_1}_{S_+}, z^{k_2}_{S_+}, \ldots$ for $k_1, k_2, \ldots, k_i, \ldots \in R_{S_+}$. Similarly, $\mathbf{z_{D_-}}$ is an abbreviation for $z^{m_1}_{D_-}, z^{m_2}_{D_-}, \ldots$ for $m_1, m_2, \ldots, m_j, \ldots \in R_{D_-}$. Then construct the successor state axiom:

  $$\begin{aligned} & \{\textstyle\bigwedge_{k_i \in R_{S_+}} link(r_{k_i}, r_l, \text{“S+”}, w_{k_i}) \land v_{\mathtt{s}}(r_{k_i}, s, y_{k_i}) \land rule(z^{k_i}_{S_+}, w_{k_i}, y_{k_i})\} \\ \land \quad & \{\textstyle\bigwedge_{m_j \in R_{D_-}} link(r_{m_j}, r_l, \text{“D-”}, w_{m_j}) \land v_d(r_{m_j}, s, y_{m_j}) \land rule(z^{m_j}_{D_-}, w_{m_j}, y_{m_j})\} \\ \land \quad & max(z_{max}, \mathbf{z_{S_+}}, \mathbf{z_{D_-}}) \supset v_{\mathtt{s}}(r_l, s, z_{max}) \qquad (10) \end{aligned}$$

Dually, for every soft-goal l in the goal model, let R_{S_-} and R_{D_+} be the sets of soft-goal graph nodes k_i and m_j for which $k_i \xrightarrow{w_{S_-}} l$ and $m_j \xrightarrow{w_{D_+}} l$, respectively, where w is the respective weight. Let $\mathbf{z_{S_-}}$ be an abbreviation for $z_{S_-}^{k_1}, z_{S_-}^{k_2}, \ldots$ for $k_1, k_2 \ldots, k_i, \ldots \in R_{S_-}$. Similarly $\mathbf{z_{D_+}}$, is an abbreviation for $z_{D_+}^{m_1}, z_{D_+}^{m_2}, \ldots$ for $m_1, m_2, \ldots, m_j, \ldots \in R_{D_+}$. Then construct the successor state axiom:

$$
\begin{array}{ll}
 & \{\bigwedge_{k_i \in R_{S_-}} link(r_{k_i}, r_l, \text{“S-”}, w_{k_i}) \wedge v_{\mathbf{s}}(r_{k_i}, s, y_{k_i}) \wedge rule(z_{S_-}^{k_i}, w_{k_i}, y_{k_i})\} \\
\wedge & \{\bigwedge_{m_j \in R_{D_+}} link(r_{m_j}, r_l, \text{“D+”}, w_{m_j}) \wedge v_d(r_{m_j}, s, y_{m_j}) \wedge rule(z_{D_+}^{m_j}, w_{m_j}, y_{m_j})\} \\
\wedge & max(z_{max}, \mathbf{z_{S_-}}, \mathbf{z_{D_+}}) \supset v_d(r_l, s, z_{max})
\end{array}
\qquad (10)
$$

Also, $max(y, x_1, x_2, \ldots, x_n)$ holds iff y equals the maximum of $x_1, x_2, \ldots, x_n$. Furthermore, $rule(z, w, y)$ is defined as follows depending on which framework we are considering:

Quantitative: $rule(z, w, y) \equiv (z = w \cdot y)$

Qualitative: The definition of $rule(z, w, y)$ is based on the following table:

w	z
some (+/-)	min(y,P)
full (++/--)	y

These axioms ensure that the satisfaction labels of the sources of the soft-goals graph are propagated according to the propagation rules we introduced earlier. Note, however, that the syntactic features of ramification axioms can cause an issue called the *ramification problem*, whereby unintended models of the ramification axioms are satisfied. We will discuss this problem below and show that the structure of our action theory is such, that makes it amenable to a syntactic manipulation that can lift the ramification problem.

6.2.4 Action Precondition Axioms

- For every task t in the goal model construct a precondition axiom as follows. First construct formula φ_{comp} as follows. Consider the path from t to the root goal. Let G_{OR} be the set of all nodes g_{OR} in the path which are OR-decomposed, including the root and t's parent. For each such g_{OR} consider its children that do not belong to the path from t to the root goal. Let G_{comp} be the set of all such children of all $g_{OR} \in G_{OR}$. Finally let T_g be the set of all leaf level tasks that are successors of a goal g. The formula φ_{comp} is constructed as follows:

$$\varphi_{comp} \equiv \bigwedge_{\forall g \in G_{comp}} \Big(\bigvee_{\forall t \in T_g} \textit{performed(t,s)} \Big)$$

Observe that t does not occur in any alternative together with any of the tasks in $T_g, \forall g \in G_{comp}$. Excluding consideration of t together with any of these tasks ensures that the plans are minimal with respect the goal tree, or, in other words, no subset of the tasks included in the plan satisfies the root goal. Thus, φ_{comp} will be true if some of the competing tasks has already been performed making t redundant.

Then consider the set G_{pre} of all hard-goals g_i such that $g_i \xrightarrow{pre} g_j$, where g_j is any ancestor of t in the hard-goals subgraph. The precondition axiom for t is the following:

$$Poss(\alpha_t, s) \equiv (\bigwedge_{\forall g_i \in G_{pre}} \varphi_{g_i}(s)) \wedge (\bigwedge_{\varphi(s) \in \Phi} \varphi(s)) \wedge (\neg \varphi_{comp}) \quad (12)$$

where Φ is the set of CFs φ, for which $\varphi \xrightarrow{pre} t$.

6.2.5 Initial Situation

For the initial situation D_{S_0}, every predicate of type *performed(*$\cdot, S_0$*)* is set to false and every fluent of type $f_p(\vec{x}, S_0)$ is set according to information given in the domain about $p(\vec{o})$. Moreover, every fluent of type $v_{\mathsf{s}}(r, s, y)$ and $v_d(r, s, y)$ is initialized depending on the framework of consideration:

Quantitative:	$v_{\mathsf{s}}(r, S_0, 0)$, $v_d(r, S_0, 0)$
Qualitative:	$v_{\mathsf{s}}(r, S_0, N)$, $v_d(r, S_0, N)$

6.2.6 Plans

If $\mathcal{D}$ is the action theory derived from the goal model and φ_g the formula representing the root goal g, then we will use the term *requirements plan* or simply *plan* to refer to a plan for $\mathcal{D}$ that achieves φ_g.

6.3. On the Ramification Problem

The ramification problem arises from the syntactic characteristics of ramification axioms when compared to their intended meaning. Consider the simple example of Figure 6. The ramification axioms associated with the satisfaction values of goal l_1 are:

$$\begin{array}{ll} & link(r_{t_2}, r_{l_1}, \text{“S+”}, 0.6) \wedge v_{\mathsf{s}}(r_{t_2}, s, y_{t_2}) \wedge rule(z_{t_2}, 0.6, y_{t_2}) \\ \wedge & max(z_{max}, z_{t_2}) \supset v_{\mathsf{s}}(r_{l_1}, s, z_{max}) \end{array}$$

For goal l_2 the corresponding axiom is the following:

$$\begin{array}{ll} & link(r_{t_1}, r_{l_2}, \text{“S+”}, 0.5) \wedge v_{\mathsf{s}}(r_{t_1}, s, y_{t_1}) \wedge rule(z_{t_1}, 0.5, y_{t_1}) \\ \wedge & link(r_{l_1}, r_{l_2}, \text{“S+”}, 0.9) \wedge v_{\mathsf{s}}(r_{l_1}, s, y_{l_1}) \wedge rule(z_{l_1}, 0.9, y_{l_1}) \\ \wedge & max(z_{max}, z_{t_1}, z_{l_1}) \supset v_{\mathsf{s}}(r_{l_2}, s, z_{max}) \end{array}$$

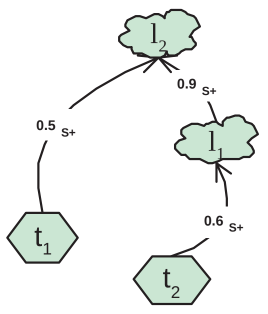


Figure 6. Simple Ramification Example

Assume now that $v_{\mathtt{s}}(r_{t_1}, s, 1)$ and $v_{\mathtt{s}}(r_{t_2}, s, 1)$, due to the performance of tasks t_1 and t_2. The obvious indirect effect to soft-goal satisfaction would be $v_{\mathtt{s}}(r_{l_1}, s, 0.5)$ and $v_{\mathtt{s}}(r_{l_2}, s, 0.54)$. But the way our ramification axioms are expressed may as well imply that $v_{\mathtt{s}}(r_{l_1}, s, 0.6)$, $\neg v_{\mathtt{s}}(r_{l_1}, s, 0.7)$ and $\neg v_{\mathtt{s}}(r_{l_2}, s, 0.56)$. Arguably the latter is not very useful for understanding the satisfaction value of l_2. Hence, we need to find a way to prevent our system of axioms from being satisfied by models (truth assignments) which do not completely calculate the satisfaction and denial values of all goals. For this to be true, the implication connective ($\supset$) of the ramification axioms needs to be treated as *definitional* [24], in a sense that the right-hand side of the implication connective is understood as defined in terms and only in terms of the left-hand side.

It has been shown in [24] that if the action theory in situation calculus is a *solitary stratified theory* then it can be re-written in a form that does not imply such unwanted models. A solitary stratified theory has the following characteristics:

Definition 6.1 Suppose D is a theory in the language of the situation calculus with domain fluents, L. Then D is a solitary stratified theory with stratification ($D_1, D_2, \ldots, D_n$) and partition $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$ if

- for $i = 1, \ldots, n$, $\mathcal{L}_i$ is the set of fluents F_i that are defined in stratum D_i and $\mathcal{L}_1 \cup \mathcal{L}_2 \cup, \ldots \cup \mathcal{L}_n = \mathcal{L}$
- D is the union $D_1 \cup D_2 \cup, \ldots \cup D_n$ of sets of axioms D_i where for each stratum, D_i is solitary with respect to $\mathcal{L}_i$, that is each T_i can be written as the union ($\mathcal{M}_i \leq \neg\mathcal{L}_i \cup \mathcal{E}_i \leq \mathcal{L}_i$), where:
  1. $\mathcal{L}_i$ is the set of fluents F_i, such that $[\neg]F_i$ is defined in D_i.
  2. $\mathcal{M}_i \leq \neg\mathcal{L}_i$ is a set of formulae of the form ($M_i \supset \neg F_i$), at most one for each fluent $F_i \in \mathcal{L}_i$, where each M_i is a formula containing no fluents drawn from $\mathcal{L}_i, \ldots \cup \mathcal{L}_n$.
  3. $\mathcal{E}_i \leq \mathcal{L}_i$ is a set of formulae of the form ($E_i \supset F_i$), at most one for each fluent $F_i \in \mathcal{L}_i$, where each E_i is a formula containing no fluents drawn from $\mathcal{L}_i, \ldots \cup \mathcal{L}_n$.

We will now show that our translation rules always provide a solitary stratified theory in situation calculus. Recall that in our goal graph, *depth* of a node is the length of the longest path from a source to that node. From the set of soft-goals L, let $L_j \subset L$ be the subset with depth i. Thus, L_0 is the set of the sources (tasks, hard goals, CEs). For $i \geq 1$, L_i are soft-goals. Then the strata are shown in Table 5 and the corresponding partitions in Table 6.

D_1	Successor state axioms of type (1) and (2)
D_2	Successor state axioms of type (3)
D_3	Ramification axioms of types (4)-(9)
D_4	Ramification axioms of types (10) and (11) for soft-goals in L_1
D_5	Ramification axioms of types (10) and (11) for soft-goals in L_2
$\ldots$	$\ldots$
D_i	Ramification axioms of types (10) and (11) for soft-goals in L_{i-3}

Table 5. The stratified theory.

Theorem 6.1 *The theory $D = D_1 \cup D_2 \cup \ldots \cup D_n$ of Table 5 is stratified with stratification $(D_1, D_2, \ldots, D_n)$:*

$\mathcal{L}_1$	$f_p(\vec{x}, s)$
$\mathcal{L}_2$	*performed*$(\cdot, s)$
$\mathcal{L}_3$	$v_{\mathsf{s}}(r, \cdot, s)$, where r is a task or hard-goal
$\mathcal{L}_4$	$v_{\mathsf{s}}(r, \cdot, s)$, $v_d(r, \cdot, s)$ where r is a soft-goal of depth 1
$\mathcal{L}_5$	$v_{\mathsf{s}}(r, \cdot, s)$, $v_d(r, \cdot, s)$ where r is a soft-goal of depth 2
$\ldots$	$\ldots$
$\mathcal{L}_i$	$v_{\mathsf{s}}(r, \cdot, s)$, $v_d(r, \cdot, s)$ where r is a soft-goal of depth $i-3$

Table 6. The partition.

Proof. To prove this we will show how the stratification of Table 5 complies with the definition.

- Each set of fluents $\mathcal{L}_i$ is defined in stratum D_i, respectively, as seen on Table 6.
- For $i = 1, 2, 3$ stratum D_i is trivially solitary with respect to $\mathcal{L}_i, i = 1, 2, 3$, respectively.
- For $i \geq 4$, the axioms in D_i are in the form $E_i \supset F_i$. To prove that D_i is solitary with respect to $\mathcal{L}_i$ we need to show that for every $j > i$, E_i does not contain fluents from $\mathcal{L}_j$.

  Indeed, for $i \geq 4$, F_i is a fluent of the form $v_{\mathsf{s}}(r_{l_i}, \cdot, \cdot)$ or $v_d(r_{l_i}, \cdot, \cdot)$ where r_{l_i} is a soft-goal with depth $depth(r_{l_i}) = i - 3$ in the soft-goals graph. On the other hand, the partition $\mathcal{L}_j$, $j > i$, contains (exclusively) fluents of the form $v_{\mathsf{s}}(r_{l_j}, \cdot, \cdot)$ or $v_d(r_{l_j}, \cdot, \cdot)$ where r_{l_j} is a soft-goal with depth $depth(r_{l_j}) = j - 3 > depth(r_{l_i})$. Since the depth of r_{l_i} is less than that of r_{l_j}, we infer that there is no path from r_{l_j} to r_{l_i}. Hence, nor is there a contribution link $r_{l_j} \longrightarrow r_{l_i}$. Since there is no such a contribution link, fluents of the from $v_{\mathsf{s}}(r_{l_j}, \cdot, \cdot)$ or $v_d(r_{l_j}, \cdot, \cdot)$ do not appear in E_i. But such fluents is all what $\mathcal{L}_j$ contains. Therefore, none of the fluents contained in E_i are in $\mathcal{L}_j$. $\Box$

7. The Preference Specification Language

Preference specification allows selection of behaviors that satisfy specific fitness criteria posed by stakeholders. Thus, instead of asking stakeholders to read and select from a vast set of alternatives, the stakeholders themselves describe what properties of the preferred behaviors are important for them. Alternatives that best satisfy those properties are then selected through automated search.

Our language for specifying stakeholder preferences is based on expressing priorities over temporal properties of behaviors implied by the goal model. Temporal properties are expressed through temporal logic based formulae, which we describe below.

7.1. Optional Condition Formule

We form *Optional Condition Formulae (OCFs)* to describe temporal characteristics of the behavior that emerges while goals are being fulfilled in a particular order and under certain circumstances. Linear Temporal Logic (LTL) is used to form OCFs. Thus:

Definition 7.1 (Optional Condition Formula - OCF) An Optional Condition Formula (OCF) is an LTL formula formed with atoms from $H \cup L \cup T \cup R$. It is drawn from the smallest set K for which:
1. $R \subset K$

2. if $g \in H$ then *issat(g)*$\in K$
3. if $t \in T$ then *isperformed(t)*$\in K$
4. If $l, l_1, l_2 \in L$, then $valS(l)\langle op\rangle c$, $valD(l)\langle op\rangle c$, $valS(l_1)\langle op\rangle valS(l_2)$ and $valD(l_1)\langle op\rangle valD(l_2)$ are in K, where $\langle op\rangle$ is one of $\leq, \geq$ and c is a real constant in [0..1].
5. If ϕ, ϕ_1, ϕ_2 are in K, then so do $\neg\phi$, $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\circ\phi$, $\Box\phi$, $\Diamond\phi$, $\phi_1 U \phi_2$ and $final(\phi)$.

The symbols $\Box$,$\Diamond$, $\circ$ and U, represent the temporal operators *always*, *eventually*, *next* and *until*, respectively. OCFs are similar to OCFs with two differences: the use of $valS()$ and $valD()$ predicates and the use of temporal operators. Thus, as opposed to CFs, which express a condition for a given time point, OCFs describe properties of a whole sequence of tasks. For example, given a course of tasks, the statement $\Box$*(isAt(nurse,patientsRoom))* is true if the nurse is at the patient's room at all times during that course.

Given an OCF and a plan for the root goal (therefore: a plan in the corresponding action theory), whether the plan satisfies the OCF can be evaluated by appealing to the situation calculus-based semantics of LTL given by Gabaldon ([11]). More specifically let us use the notation $\varphi[s, s']$ to denote that φ holds in all situations from s to $s' \equiv do(\vec{\alpha}, s)$. Also, $s \sqsubseteq s'$ means that either $s = s'$ or there is a sequence of actions $\vec{\alpha} = \alpha_1, \alpha_2, ...$ such that $s' = do(\vec{\alpha}, s)$. The semantics of OCFs in situation calculus terms are as follows.

$$
\begin{gathered}
p(\vec{o}) \in R \text{ then } p(\vec{o})[s, s'] \equiv f_{p(\vec{x})}[s] \\
g \in H \text{ then } \textit{issat(g)}[s, s'] \equiv \varphi_g[s] \\
t \in T \text{ then } \textit{isperformed(t)}\ [s, s'] \equiv performed(t)[s] \\
l \in L \text{ then } valS(l)\langle op\rangle c \equiv v_{\mathsf{s}}(r_l, y) \wedge (y\langle op\rangle c)[s] \\
l_1, l_2 \in L \text{ then } valS(l_1)\langle op\rangle valS(l_2)[s, s'] \equiv v_{\mathsf{s}}(r_{l_1}, y_1) \wedge v_{\mathsf{s}}(r_{l_2}, y_2) \wedge (y_1\langle op\rangle y_2)[s] \\
l \in L \text{ then } valD(l)\langle op\rangle c \equiv v_d(r_l, y) \wedge (y\langle op\rangle c)[s] \\
l_1, l_2 \in L \text{ then } valD(l_1)\langle op\rangle valD(l_2)[s, s'] \equiv v_d(r_{l_1}, y_1) \wedge v_d(r_{l_2}, y_2) \wedge (y_1\langle op\rangle y_2)[s] \\
\Diamond\phi[s, s'] \equiv (\exists s_1 : s \sqsubseteq s_1 \sqsubseteq s')\varphi[s_1] \\
\Box\phi[s, s'] \equiv (\forall s_1 : s \sqsubseteq s_1 \sqsubseteq s')\varphi[s_1] \\
\circ\phi[s, s'] \equiv (\exists\alpha.do(\alpha, s) \sqsubseteq s')\varphi[do(\alpha, s), s'] \\
final(f)[s, s'] \equiv f[s'] \\
\phi_1 U \phi_2[s, s'] \equiv (\exists s_1 : s \sqsubseteq s_1 \sqsubseteq s')\varphi_2[s_1, s'] \wedge (\forall s_2 : s \sqsubseteq s_2 \sqsubseteq s_1)\varphi_2[s_2, s']
\end{gathered}
$$

Returning to our example of Section 3 and Figure 1, consider the statement *"we should definitely avoid anything that would make the patient unhappy"*, expressed by the managers of the unit. In other words, while a sequence of tasks to attend to a patient's is executed, the patient should not be unhappy at any point. In our goal language this means that at any time (i.e. *always*, $\Box$), the denial value of the soft-goal *Happy Patient* (i.e. *valD*('Happy Patient')) must remain below a very small value (say 0.01 – we discuss how we come up with such numbers in later sections). Thus, we would write the OCF as follows:

$$\Box(valD(\text{'Happy Patient'}) \leq 0.01) \qquad (1)$$

Moreover, we can define more interesting time intervals in which a desire to satisfy (or deny) a high-level goal is relevant. Consider for example, the desire *"we should not avoid disturbing the nurse as long as the patient's condition is severe"*. This desire implies that the importance of the soft-goal *Avoid Nurse Disturbance* is relevant only when a certain condition is true and for as long as it is true. The OCF formula to express this is:

$$\Box(valD(\text{'Avoid Nurse Disturbance'}) \leq 0.1 \rightarrow \neg patientsCondition(severe)) \qquad (2)$$

Similarly, an operational detail may depend on the level of satisfaction or denial of a soft-goal. For example, *"if the patient is unhappy for any reason, then the nurse should not skip the visit"* would be again formalized as:

$$\Box(valD(\text{'Happy Patient'}) > 0 \rightarrow \Diamond(isAt(nurse, patientsRoom))) \qquad (3)$$

Observe how the use of soft-goals such as *Happy Patient* allows us to indirectly refer to desired operational level decisions without having to explicitly specify or even exactly know them at the time we construct the OCF.

The expressive power of LTL can be used to pose purely temporal constraints to preferred plans. These temporal constraints can be seen as optional counterparts of the mandatory constraints that are implemented in the goal graph through precedence links. For example *"the nurse should turn the request off only after she has responded to the patient's call"* can be formulated as follows:

$$\neg\, \textit{isperformed}(\text{'Turn Request Off'})\, U\, \textit{issat}(\text{'Nurse Responded Call'}) \qquad (3)$$

Given an OCF and a plan for the root goal, the plan will either satisfy or not satisfy the OCF. Thus, going back to Figure 1, OCF (1) above is satisfied by plan $[t_1, t_3, t_7, t_9]$ but not by plan $[t_1, t_3, t_8, t_9]$, due to the (indirect) negative contribution of t_8 to the soft-goal *Happy Patient*. Formally, whether a plan satisfies an OCF can be evaluated by appealing to the situation calculus-based semantics of LTL given by Gabaldon ([11]) and the corresponding semantics of goal plans; more details are again in [23].

7.2. Preferences over Conditions

Two types of preference formulas are used: *preference formulae* and *weighted preference formulae*.

Definition 7.2 (Preference Formula (PF)) , is a formula of the form $\phi_0[w_0] \succeq \phi_1[w_1] \succeq, \ldots, \succeq \phi_n[w_n]$, where $n \geq 0$, each ϕ_i is an OCF, $w_0 \geq 0$, $w_n \leq 1$ and $w_i < w_j$ for $i < j$. When n=0, preference formulae correspond to single OCFs.

The satisfaction of a PF is assessed as follows. Define $d(\phi)$ be the satisfaction degree of a OCF ϕ, for a given plan. If the plan satisfies ϕ, we set $d(\phi) = 0$ otherwise $d(\phi) = 1$. Given a whole preference formula $\Phi = \phi_0[w_0] \succeq \phi_1[w_1] \succeq, \ldots, \succeq, \phi_n[w_n]$ then $d(\Phi) = w_i$ where i is the minimum i for which ϕ_i is satisfied by the plan or $d(\Phi) = 0$ if no such i exists.

Returning to our example of Figure 1, the following is a PF consisting of two OCFs:

$\Diamond$ *isperformed*('Nurse doesn't talk with patient') $[0.2] \succeq \Diamond$ *isperformed*('Nurse Skips Visit') $[0.5]$

It means that the first OCF $\Diamond$ *isperformed*('Nurse doesn't talk with patient') is preferred from the second one $\Diamond$ *isperformed*('Nurse Skips Visit'). Given a plan, the PF is satisfied by a particular score, depending on which of its constituent desires is satisfied by the plan. Thus, if the first OCF is satisfied, then the PF is assigned a score 0.2 (as indicated inside the first pair of brackets), otherwise, if the second OCF is satisfied, then the PF is assigned a (worse) score of 0.5, as indicated inside the second pair of brackets. If neither of the constituent OCFs is satisfied, the PF is assigned the worst possible score: 1.0. Thus, plans $[t_1, t_6, t_8, t_9]$, $[t_1, t_3, t_8, t_9]$ and $[t_1, t_3, t_7, t_9]$ satisfy the above PF with score 0.2, 0.5 and 1.0 respectively.

Using PFs analysts can define priorities over desires posed by the same or different stakeholders. In our example, the nurse's statement that *"[they] don't like the idea of talking to the patient remotely, but if they had to, they would at least choose to do so at the nursing station"* can be formulated through this PF:

$\Box(\neg$ *issat*('Talked With Patient')$)[0.0] \succeq \Diamond$ *issat*('Talked from the Nursing Station') $[0.5]$

We use the PF when the satisfaction of an OCF at a higher priority implies that we are indifferent about the satisfaction of OCFs of a lower priority, which is the case in this example. Otherwise, we may use *weighted preference formulae* (WPFs), which are constructed from PFs as follows:

Definition 7.3 (Weighted Preference Formula - WPF) , is a formula of the form $\Sigma_i(w_i \times \{\phi_i\})$, where $0 \leq w_n \leq 1$, $\Sigma_i(w_i) = 1$, and ϕ_i a PF.

The weight of individual formulae ϕ_i in WPFs is also calculated as above. Note that PFs may consist of a single OCF. Returning to our nursing example, assume that the management provides a combination of desires: "*we should definitely avoid anything that would make the patient unhappy, but it would also be nice to increase nurses' productivity somehow.*" The statement implies a priority of the patient's happiness over the productivity of the nurses. The WPF to represent this can be:

$\{\Box(valD(\text{'Happy Patient'}) \leq 0.1)[0.0]\} \times 0.8 + \{final(valS(\text{'Increase Nurse Productivity'}) \geq 0.1)[0.0]\} \times 0.2$

The above WPF has a score of 0.0 if the OCFs of both of its constituent single-OCF PFs are satisfied, 0.2 if only the OCF of the first PF is satisfied, 0.8 if only the OCF of the second PF is satisfied and 1.0 if the OCF of neither PF is satisfied.

The ideal application of WPFs is them being the high-level result of combining PFs and WPFs of individual stakeholders, where the weights associated with each formula express the analyst's perception over the relative importance of each stakeholder and her overall desires. Thus, by giving the preferences of the management a weight of 0.9 and to the nurses 0.1, the formula of Figure 7 is the WPF resulting from combining the two individual formulas.

$$\begin{array}{c}
\{\Box(valD(\text{'Happy Patient'}) \leq 0.1)[0.0]\} \times 0.72+ \\
\{final(valS(\text{'Increase Nurse Productivity'}) \geq 0.1)[0.0]\} \times 0.18+ \\
\{\Box(\neg\ \textit{issat}(\text{'Talked With Patient'}))[0.0] \succeq \\
\Diamond\ \textit{issat}(\text{'Talked from the Nursing Station'})\ [0.5]\} \times 0.1
\end{array}$$

Figure 7. Preference Formula

The preferences language we propose is a simplification and adaptation of the one presented in [4] for the purposes of preference-based planning. In contrast to that proposal, we exclusively focus on quantitative aggregation of preferences and we also introduce WPFs which turned out to be very useful in practice. However, users of our goal-oriented framework that desire to adjust the expressive power of the preference specification language, can still use the same diagrammatic and evaluation infrastructure introduced in this paper, but formulate preferences following [4].

8. Behavioral Analysis

As we saw, the models constructed through the goal language we presented in Section 4 imply a great number of alternative plans for fulfilling the root goals. Given such a model and a preference specification, certain behaviors implied by the former become interesting in that they satisfy the latter with optimal score.

We extended a preference-based planner, called PPLan ([4]), to allow automatic search for plans and satisfy preference formulae of the type we discussed above. To perform this reasoning task, the planner takes as input a goal model, a preference formula and initial values for the domain facts, and returns a list of plans for the goal model prioritized by the degree by which they satisfy the preference formula.

In our example of Figure 1, assuming that we are given initial values for the domain predicates $\{isTime(afternoon), patientsCondition(moderate)\}$ and the preference formula of Figure 7, the resulting ranking can be seen in Figure 8.

Rank	Plan	Score
1.	$[t_1, t_3, t_7, t_9]$	.1
2.	$[t_2, t_3, t_7, t_9]$	.1
3.-6.	$[\ldots, t_3, \ldots, t_7, \ldots]$	.1
7.	$[t_1, t_4, t_5, t_7, t_9]$	.23
8.-14.	$\ldots$	.23
15.	$[t_1, t_4, t_5, t_8, t_9]$	.77
16.-22.	$\ldots$	.77
23.	$[t_1, t_3, t_8, t_9]$	.82
24.-28.	$\ldots$	.82
29.	$[t_1, t_6, t_7, t_9]$	.9
30.-34.	$\ldots$	.9

Figure 8. Preferred Plans

Thus, plans that include the nurse talking through a mobile device and eventually visiting the patient too end up having better score (0.1) due to the significant importance of patient satisfaction in the preference formula. None of the alternatives at the top half of the list seems to completely satisfy the nurses' desire not to establish any voice connection with the patient. However, if the nurses had been given the same weight as the management in constructing the WPF, that is 0.5 each, the top plans would involve at least partial satisfaction of the preferences of nurses, namely absence of carrying and talking through a mobile device, which is something that we know they dislike.

Thanks to the presence of CFs in the goal model, the resulting ranking is also sensitive to the original values of the domain predicates, which represent the state of the context. Consider the WPF:

$$\{\Box(valD(\text{'Avoid Patient's Disturbance'}) \leq 0.1)[0.0]\} \times 0.7+$$
$$\{\Box(valD(\text{'Nurse Comfort'}) \leq 0.1)[0.0]\} \times 0.3$$

In circumstances in which $isTime(night)$ does *not* hold, the score of the preference is minimized to 0.0 for any behavior of the form $[t_2 \ldots]$. The same behaviors, however, take a score 0.7 if the circumstances include $isTime(night)$. In the latter case, behaviors such as $[t_1 \ldots]$ are more preferred as they satisfy the WPF with 0.3.

The actual ranking of alternatives should be interpreted as an indication of groups of alternatives that have significant differences in their preference score. In the ranking above, for example, there is a clear distance between the first 14 alternatives and the remaining in the list. However, one should resist the temptation of comparing alternatives with similar scores, such as, for instance, the 6th and the 7th alternative in the ranking.

9. Static Analysis

9.1. Time-independent Goal and Preference Models

The use of the temporal extension of the goal modeling language and the LTL operators in the preference specification language are not mandatory for performing useful preference analysis. We can perform static (i.e. time-independent) analysis when we are not interested in the sequence of execution of tasks or satisfaction of goals. This implies the use of time-independent goal and preference models.

A time-independent version of the goal model of Figure 1, which we have been discussing so far, can be seen in Figure 9.

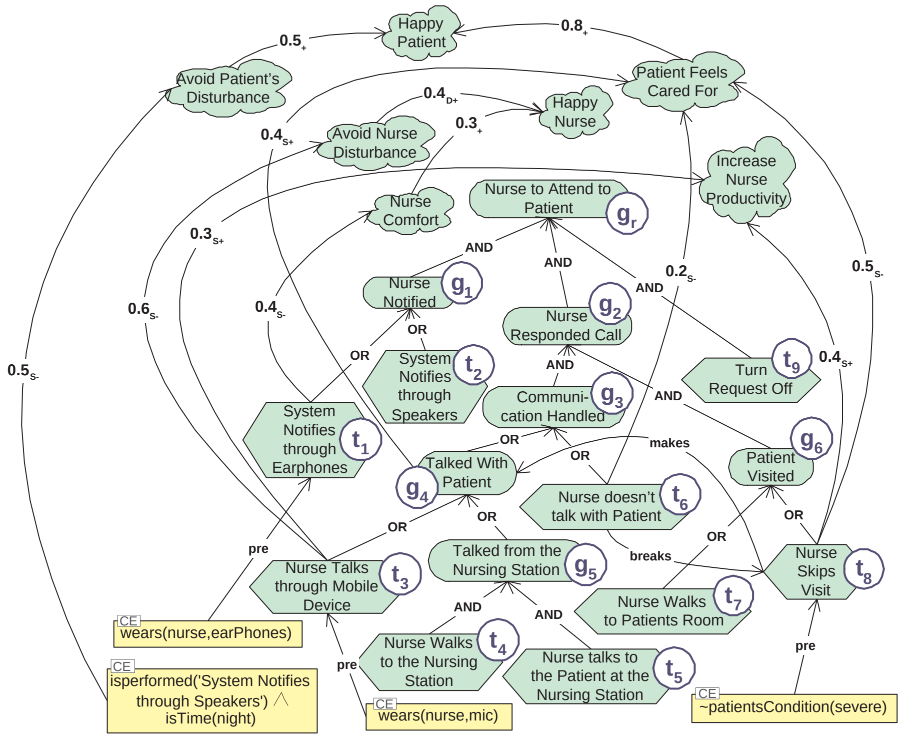

Happy Patient
Patient Feels Cared For
Avoid Patient's Disturbance
Happy Nurse
Avoid Nurse Disturbance
Increase Nurse Productivity
Nurse Comfort
Nurse to Attend to Patient
Nurse Notified
Nurse Responded Call
System Notifies through Speakers
Turn Request Off
Communi-cation Handled
System Notifies through Earphones
Talked With Patient
Patient Visited
Nurse doesn't talk with Patient
Talked from the Nursing Station
Nurse Talks through Mobile Device
Nurse Skips Visit
Nurse Walks to Patients Room
Nurse Walks to the Nursing Station
Nurse talks to the Patient at the Nursing Station
wears(nurse,earPhones)
isperformed('System Notifies through Speakers') ∧ isTime(night)
wears(nurse,mic)
~patientsCondition(severe)
makes
breaks
pre

Figure 9. A goal model for static analysis

The new time-independent model is different in several ways. Thus, effect links and EEs are not used. Instead, $\xrightarrow{makes}$ and $\xrightarrow{breaks}$ links, borrowed from *i** ([34]), are used to represent satisfaction constraints between hard-goals, tasks, CFs. Such links are drawn from hard-goals, tasks and CFs to goals and tasks. Thus, the link *Nurse Skips Visit*$\xrightarrow{make}$*Talked With Patient*, shows that if the task to *Nurse Skips Visit* is included in a solution, this necessarily implies that *Talked with Patient* must necessarily be satisfied in the same solution. Conversely, the link *Nurse does't talk to patient*$\xrightarrow{breaks}$*Nurse Skips Visit*, shows that if not talking to the patient is part of the solution, skipping the visit cannot be part of the same solution. Further, the $\xrightarrow{pre}$ link acquires a different interpretation: it represents a condition rather than a precedence. Thus *wears(nurse,mic)*$\xrightarrow{pre}$*Nurse Talks through Mobile Device*, means that for the task *Nurse Talks through Mobile Device* to be included in a solution, *wears(nurse,mic)* must hold true in the same solution.

Given these simplifications, the hard-goals sub-graph can now be formalized in propositional logic. We can associate each goal with a propositional literal and represent the satisfaction of the root goal in terms of a propositional formula $G \equiv S_g \wedge C_g$. S_g represents the AND/OR structure in terms of leaf level literals. Each non-leaf hard-goal node g is recursively replaced by the conjunction or XOR-disjunction of its children depending on whether the decomposition is AND or OR, respectively. We call the resulting formula *task-grounded formula* of the hard-goal g. In Figure 9, for example, g_4's task grounded formula is $t_3 \oplus (t4 \wedge t5)$. Notice the XOR treatment of OR-decomposition in constructing the task-grounded formula, for the interest of focusing on minimal solutions only. C_g represents the additional "makes", "breaks" and "pre" links. Each constraint link in the model results in a conjunct in the formula C_g as follows:

Link Type	Conjunct
$g_1 \xrightarrow{makes} g_2$	$g_1 \Rightarrow g_2$
$g_1 \xrightarrow{breaks} g_2$	$g_1 \Rightarrow \neg g_2$
$g_1 \xrightarrow{pre} g_2$	$g_2 \Rightarrow g_1$

In all cases, g_1 can be a condition formula, goal or task, while g_2 can be a goal or task. Whenever g_1 or g_2 represent goals, they are replaced with the corresponding task-grounded formula. This way, the entire formula G is grounded on literals representing leaf level tasks or domain predicates.

In the Figure, G is the conjunction of:

$S_g \equiv (t_1 \oplus t_2) \wedge (((t_3 \oplus (t_4 \wedge t_5)) \oplus t_6) \wedge (t_7 \oplus t_8)) \wedge t_9$

$C_g \equiv (t_1 \Rightarrow$ *wears(nurse,earphones)*$) \wedge (t_3 \Rightarrow$ *wears(nurse,mic)*$) \wedge (t_8 \Rightarrow \neg$*patientsCondition(severe)*$) \wedge (t_6 \Rightarrow \neg t_8) \wedge (t_8 \Rightarrow (t_3 \vee (t_4 \wedge t_5)))$

An alternative in the static hard-goal subgraph are defined in a way similar to the alternatives in the temporally extended version: an alternative for a goal is a solution to the AND/OR tree rooted to that goal. Further, in static graphs, an *admissible alternative* in the hard-goal subgraph is a solution of the AND/OR tree that also satisfies the "breaks", "makes" and "pre" constraints, given a truth assignment for the domain predicates. In propositional terms, an alternative is the part of a model of G (i.e. a truth assignment of its literals that satisfies it) that mentions only the leaf level tasks (i.e. without the domain predicates). In Figure 9, $\{t_1, t_3, t_7, t_9\}$ is an alternative given initial conditions e.g. $\{$wears(nurse, earphones),$\neg$ wears(nurse, earphones), $\neg$ patientsCondition(severe)$\}$. Should the domain predicates be $\{$ $\neg$ wears(nurse, earphones), $\neg$ wears(nurse, earphones), $\neg$ patientsCondition(severe)$\}$, then $\{t_1, t_3, t_7, t_9\}$ would not be an alternative as it would violate the constraint $(t_1 \Rightarrow$ *wears(nurse,earphones)*$)$.

Given an alternative of the hard-goal subgraph and a truth value for the domain predicates, the satisfaction and denial values of the soft-goal subgraph is found as follows. Let M be a propositional interpretation of the literals that comprise G (i.e. a truth assignment) such that $M \models G$. Then, each element of the soft-goals subgraph l is assigned two functions $valS_M(l)$ and $valD_M(l)$. The functions represent the result of the application of the label propagation algorithm (Figure 5) with initial values $valS^0_M(l)$ and $valD^0_M(l)$ set as follows:

- If l is a literal of formula representing a task then $valS^0_M(l) = 1$ iff $M \models l$ and $valS^0_M(l) = 0$ otherwise. If l represents a goal or a CF then let f_l be the corresponding a formula based on literals representing tasks or domain predicates. Then, set $valS^0_M(l) = 1$ iff $M \models f_l$ and $valS^0_M(l) = 0$ otherwise. We maintain $valD^0_M(l) = 0$ for all such literals.
- If l is a literal representing a soft-goal then $valS^0_M(l) = valD^0_M(l) = 0$.

Formulation of preferences, on the other hand, is exactly as described in Section 7, with one obvious adjustment: temporal operators are not used. Thus, we define the Static Optional Condition Formula (SOCF) as follows:

Definition 9.1 (Static Optional Condition Formula - SOCF) A Static Optional Condition Formula (SOCF) is an propositional formula formed with atoms from $H \cup L \cup T \cup R$. It is drawn from the smallest set K for which:

1. $R \subset K$
2. if $g \in H$ then *issat(g)*$\in K$
3. if $t \in T$ then *isperformed(t)*$\in K$
4. If $l, l_1, l_2 \in L$, then $valS(l)\langle op\rangle c$, $valD(l)\langle op\rangle c$, $valS(l_1)\langle op\rangle valS(l_2)$ and $valD(l_1)\langle op\rangle valD(l_2)$ are in K, where $\langle op\rangle$ is one of $\leq, \geq$ and c is a real constant in [0..1].
5. If ϕ, ϕ_1, ϕ_2 are in K, then so do $\neg\phi$, $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$.

The semantics of SOCF are again based on propositional calculus interpretation of the goal model. In particular let M be a satisfying interpretation of G and f_g be the task-grounded formula of a hard-goal g:

$$
\begin{gathered}
p(\vec{o}) \in R \text{ then } p(\vec{o}) \text{ holds iff } M \models p(\vec{o}) \\
g \in H \text{ then } \textit{issat(g)} \text{ holds iff } M \models f_g \\
t \in T \text{ then } \textit{isperformed(t)} \text{ holds iff } M \models t \\
l \in L \text{ then } valS(l)\langle op\rangle c \text{ holds iff } valS_M(l)\langle op\rangle c \\
l_1, l_2 \in L \text{ then } valS(l_1)\langle op\rangle valS(l_2) \text{ holds iff } valS_M(l_1)\langle op\rangle valS_M(l_2) \\
l \in L \text{ then } valD(l)\langle op\rangle c \text{ holds iff } valD_M(l)\langle op\rangle c \\
l_1, l_2 \in L \text{ then } valD(l_1)\langle op\rangle valD(l_2) \text{ holds iff } valD_M(l_1)\langle op\rangle valD_M(l_2)
\end{gathered}
$$

9.2. Time-independent Analysis

Given the definition of SOCF, Static PFs (SPFs) and Static WPFs (SWPFs) are defined exactly as in the behavioral, with the difference that the constituent formulae of SPFs are not OCFs but SOCFs. Returning to our example, assume we are interested in solutions reflecting the fact that on one hand the nurses *"prefer having to walk to the nursing station quite more than having to talk though a mobile device"* and on the other hand the management believes that *"having the patient satisfied somehow is strongly more important than having the nurse satisfied"*. The SWPF is then written as follows:

$$
\begin{gathered}
\{\ \textit{isperformed}(\text{'Nurse Talks Through Mobile Device'})\ [0.0] \\
\succeq \textit{isperformed}(\text{'Nurse Walks To Nursing Station'})\ [0.3]\ \} \times 0.5\ + \\
\{\ \textit{valS}(\text{'Happy Patient})\geq 0.1\ [0.0] \succeq \textit{valS}(\text{'Happy Nurse}) \geq 0.1\ [0.7]\ \} \times 0.5
\end{gathered}
$$

In order to reason about such time-independent preference formulae a separate reasoning component has been implemented. As opposed to behavioral analysis that makes use of a preference-based planner, the procedure we use to perform static analysis is significantly simpler and faster. The procedure reads as input the time-independent goal model, the preference formula and the value of the domain predicates and outputs a set of admissible alternatives (versus plans) ranked by the score by which they satisfy the preference. For the above SWPF and context (truth value of domain predicates) $\{$*wears(nurse,mic), wears(nurse,earPhones), patientsCondition(moderate), isTime(night*$\}$ the output is the ranking of Figure 10:

The above alternatives are not sequences but rather sets of tasks. Notice how the positive impact of talking to the patient somehow implies that alternatives that include t_6 have a very poor score.

Rank	Alternative	Score
1.	$\{t_1, t_4, t_5, t_8, t_9\}$	0.0
2.	$\{t_2, t_4, t_5, t_8, t_9\}$	0.0
3.	$\{t_1, t_4, t_5, t_7, t_9\}$	0.0
4.	$\{t_2, t_4, t_5, t_7, t_9\}$	0.0
5.	$\{t_1, t_3, t_8, t_9\}$	0.15
6.	$\{t_2, t_3, t_8, t_9\}$	0.15
7.	$\{t_1, t_3, t_7, t_9\}$	0.15
8.	$\{t_2, t_3, t_7, t_9\}$	0.15
9.	$\{t_1, t_6, t_7, t_9\}$	1.0
10.	$\{t_2, t_6, t_7, t_9\}$	1.0

Figure 10. Preferred Alternatives

10 Tool

Rankings of preferred plans can be produced using our prototype tool for evaluating preferences. The tool reads a goal model, a set of the domain predicates representing circumstances of interest, and a preference formula, all in the form of Prolog predicates, and returns a set of sequences of tasks ranked by the degree by which they satisfy the preference formula. As mentioned above, the tool is heavily based on PPlan ([4]), which employs a best-first search strategy to find preferred plans. In the following subsections, we provide a brief overview of PPlan, describe the extensions we developed and discuss its performance in a number of examples that we ran.

10.1 An overview of PPlan

PPlan employs an A* best-first search to identify plans from a specified initial situation to a situation that best satisfies a given preference formula. Beginning from the initial situation and the empty plan, the algorithm *progresses* through possible next situations that form through the performance of actions, aiming at reaching a situation in which the goal formula is satisfied. Hence, at every step, the algorithm first identifies which actions satisfy their precondition axioms and can be considered as the next action to be performed. Thus, a list of potential extensions to the current partial plan (*neighbors*) are constructed and then ordered subject to an *evaluation function*, forming the *frontier*. The candidate with the best score in the evaluation function is pursued, and the same procedure repeats from there.

The evaluation function is a prediction of the best and worst score the preference formula can possibly acquire in later stages, given the current situation. These are calculated by examining whether it is possible for the basic desires of the preference formulae (which we here call OCFs) to be true or false in subsequent situation, given the current situation and partial plan. For example, if in the current situation the fluent p holds, the desire $\Box\neg p$ that may appear as part of a preference formula can obviously never be true, independent of what further choices are going to be made. In other words, both optimistic and pessimistic estimations for $\Box\neg p$, if we continue on the current partial plan, are that it is false. On the other hand, again given that p holds, the desire $\Diamond p$ is true and will stay true independent of further choices. Nevertheless, if p has been true in all previous situations, a prediction of the truth value for formula $\Diamond\neg p$ can be either true (p continues to hold until a plan is found) or false ($\neg p$ holds in some future situation due to the performance of an action). Similar observations can be made with formulas based on other temporal operators as well as compound ones; we refer the reader to [4] for a more formal account.

Thus, given a partial plan and a preference formula, each constituent desire of the preference formula can be evaluated with its best and worst possible truth values, providing us with overall optimistic and pessimistic weights of the preference formula, respectively. The evaluation function for the A* search is exactly the optimistic score of the preference formula given a partial plan and the current situation. Thus, given a set of candidate partial plans for the next situation, the one with the best optimistic score is chosen. In case of a draw (equal optimistic scores), the pessimistic score is used. If there is a draw there, too, the shortest candidate plan is chosen.

The evaluation function is *admissible*, which means that the first solution that is found is guaranteed to be the optimal. This is because the actual weight calculated once the plan is found, cannot be better than the optimistic weight estimated for partial plan in the search process.

10.2 Extending PPlan

Two extensions were considered to serve our purposes. One was a Prolog implementation of the ALP algorithm we discussed earlier and its incorporation to PPlans search process. Thus, the ALP algorithm runs as part of PPlan's calculation of a progression. After the progression has been performed and a new set of fluents has emerged, the set is passed to the ALP execution routine which evaluates the impact of the new situation to the satisfaction and denial values of the softgoals. The fluents of type $v_{\mathtt{s}}(r,s)$ and $v_d(r,s)$ are updated according to the results of the ALP execution.

Our second extension to PPLan is an enhancement of the existing heuristic aiming at exploiting the structure of the goal tree. In particular, at a given situation, where a subset of leaf-level tasks of the goal model have already been performed, it is possible to calculate an estimation of the maximum and minimum number of tasks that need to be performed for the root goal to be satisfied. Let G be a set of nodes g comprising an AND/OR decomposition tree. For every such node $g \in G$, let g_{min} and g_{max} be the minimum and maximum, respectively, number of tasks that need to be performed for the satisfaction of g. Also let g^i be the $i-$th child of g and g^i_{min} and g^i_{max} its corresponding distances. Then the procedure for calculating the minimum and maximum distance from achieving the goal g, given the set of all nodes given the set T of leaf level nodes that have already been performed can be seen in Figure 11.

The distance of a candidate plan from satisfying the root goal is used together with the heuristics that are already employed in PPLan, but it is given lower priority. Thus, PPlan's frontier is set to sort partial plans with the following order: i) Optimistic Weight, ii) Pessimistic Weight, iii) Minimum Distance to Goal, and iv) Maximum Distance to Goal. In other words, when comparing two partial plans, their optimistic weight is first checked, and the plan with the lowest value is picked. In case of a draw (the weights are equal) the pessimistic weight is checked, and the lowest pessimistic weight is chosen. If there is a draw in the pessimistic weights too, we choose the plan with the smaller minimum distance to goal. If there is a draw there, too, we choose the one with the smaller maximum distance to goal. If all these are equal we choose non-deterministically. In Figure 12, the basic PPLan algorithm is sketched together with the function COMPAREVAL, which used by SORTNMERGEBYVAL for comparing partial plans.

Theorem 10.1 (Admissibility) *The score evaluation is admissible.*

Proof. Admissibility follows trivially by the fact the distance-to-goal criterion is given a lower priority than the optimistic and pessimistic weight criteria, which have been proven to constitute admissible

```
CalculateDistance($g$,$T$)
INPUT:
$T$: A set of task already been performed
$g$: The goal for which we calculate distance
RETURNS:
$g$ with its $g_{min}$, $g_{max}$ values updated
BEGIN
if $g$ is leaf then
    if $g$ in $T$ then
        $g_{min} := 0$; $g_{max} := 0$;
    else
        $g_{min} := 1$; $g_{max} := 1$;
    return $g$;
if g is AND decomposition then
    for every child $g^i$
        $g^i$ = CalculateDistance($g^i$, $T$);
    $g_{min} = \Sigma_i g^i_{min}$;
    $g_{max} = \Sigma_i g^i_{max}$;
    return $g$;
if g is OR decomposition then
    for every child $g^i$
        $g^i$ = CalculateDistance($g^i$, $T$);
    $g_{min} = min_i(g^i_{min})$;
    $g_{max} = max_i(g^i_{max})$;
    return $g$;
END
```

Figure 11. Distance-to-Goal Calculation

```
PPLAN($state, goal, preferences$)
    $frontier$ := INITFRONTIER($state, preferences$)
    while $frontier \neq \emptyset$
        $current$:= REMOVEFIRST($frontier$)
        $state$:= UPDATESTATE($current, state$)
        $preferences$:= UPDATEPREFERENCES($current, state, preferences$)
        if $goal \subset state$ and optW($current$) = pesW($current$)
            return $current$, optW($current$)
        end if
        $neighbours$:= EXPAND($current, state, preferences$)
        $frontier$:= SORTNMERGEBYVAL($neighbours, frontier$)
    end while

Partial Plan COMPAREVAL(Partial Plan $pl1$, Partial Plan $pl2$)
/* Comparisons between two partial plans $pl_1$, $pl_2$
are performed as follows: */
    if optW($pl1$) $\neq$ optW($pl2$)
        return argmin(optW($pl1$), optW($pl2$))
    if pesW($pl1$) $\neq$ pesW($pl2$)
        return argmin(optW($pl1$), optW($pl2$))
    $gRoot^1$ = CalculateDistance($gRoot, pl1$)
    $gRoot^2$ = CalculateDistance($gRoot, pl2$)
    if ($gRoot^1_{min} > gRoot^2_{min}$)
        return $pl2$
    else if ($gRoot^1_{min} < gRoot^2_{min}$)
        return $pl1$
    if ($gRoot^1_{max} > gRoot^2_{max}$)
        return $pl2$
    else if ($gRoot^1_{max} < gRoot^2_{max}$)
        return $pl1$
    return pickNonDet($pl1$, $pl2$)
```

Figure 12. Adapted PPlan

evaluation ([4]). □

The rationale behind using the distance-to-goal criterion is that, according to our experience, the minimum number of tasks that need to be performed for the root of a goal model to be satisfied is usually proportional to the size of the goal model. The original version of PPlan ignores that, and biases towards examining all plans of a certain length before it decides to examine longer ones, even when the goal model suggests that there does not exist a plan of that length.

10.3 Static Analysis

As we saw in Section 9 it is possible to do interesting preference analysis without using the temporal aspect of our goal and preference models. This greatly simplifies the reasoning task.

The algorithm for reasoning about time-independent preferences is seen in Figure 13. Intuitively, through simple traversal, the algorithm goes through all alternative solutions of the AND/OR tree. For each such alternative, it checks whether the constraints are satisfied, and if yes – which means it is an admissible alternative – it is used to evaluate the preference formula. The alternative is then assigned a number based on the score with which it satisfies the preference formulae. Based on that assigned score the alternative takes the appropriate place in the ranking of the so far visited alternatives. The algorithm returns when all alternatives of the AND/OR tree have been visited.

```
Traverse($\mathcal{G}$, $P_{init}$, $f_{pref}$)
INPUT:
$\mathcal{G}$: a graph representing the goal model
$P_{init}$: the set of domain predicates that hold true
$f_{pref}$: a preference formula (PF or WPF)
RETURNS:
A ranking *rank* of alternatives ordered by their score in satisfying $f_{pref}$
BEGIN
Construct $C_g$ by reading the "breaks", "makes" and "pre" relations in $\mathcal{G}$;
translate $P_{init}$ into a propositional interpretation $M_P$ for the domain predicates;
For each solution *soln* of the AND/OR tree of $\mathcal{G}$
    translate *soln* into the corresponding interpretation $M_S$ of the literals in $S_g$;
    set $M = M_S \cup M_P$;
    if $M \models C_g$ then
        for all soft-goals $l$ in $G$
            calculate $valS_M(l)$ and $valD_M(l)$;
        calculate the score *sc* of $f_{pref}$ based on $M$, $valS_M(l)$ and $valD_M(l)$;
        insert-sort *soln* in *rank* based on *sc*
    end-if
next solution
END
```

Figure 13. Algorithm For Static Analysis

Theorem 10.2 *The static analysis algorithm terminates after having evaluated all alternatives of the goal model.*

Proof. Recall that the hard-goal subgraph of static goal model can be translated into a formula $G \equiv S_g \wedge C_g$, where S_g is the subformula that is constructed by reading the AND/OR goal structure, and C_g the one that is formed by putting together all "pre", "makes" and "breaks" conjuncts. Trivially $G \models S_g$, that is every solution to G must necessarily be a solution to S_g. Thus, by generating all possible models (truth assignments) for S_g we are sure to have visited all possible solutions of G and perhaps more (e.g. those that do not satisfy C_g). But all possible models for S_g are all possible solutions to the AND/OR tree and vice versa. Termination trivially follows from the fact that the AND/OR tree has a finite number of alternatives and all functions called from within the loop terminate. $\Box$

Preferences are evaluated according to the propositional calculus semantics given in Section 9: every alternative of the AND/OR tree together with the truth values of all domain predicates constitutes a model M for G. This allows evaluation of SOCFs based on the rules of Section 9 and subsequent calculation of PF and WPF score based on the same rules that apply for temporally-extended preferences.

10.4 Performance Evaluation - Behavioral Analysis

We now take a look at the performance of our tool. While we expect it to be close to that of PPlan (discussed in [4]), we also expect that the distance-to-goal heuristic reduces the search time for certain types of preference formulae. In general, our experiments show that the distance-to-goal heuristic significantly boosts performance with simple preferences with mostly satisfiable constituent desires. However, when preferences are more complex and involve unsatisfiable desires, the performance of the heuristic may be less stable and occasionally worsen the performance of the original PPLan.

Our first experimental study is the sensitivity of the performance of the heuristic with respect to the minimum plan length that the goal model implies. We considered goal models with AND-decompositions only and varied the number of tasks and therefore the minimum plan length. We tried minimum plan lengths from 6 to 9. We removed any temporal constraints in order to maximize the search space. For each of these four models we constructed a similar set of preference formulae. The set contains a subset of satisfiable formulae (returning 0.0), a set of unsatisfiable formulae (returning 1.0) and a set of mixed ones (returning anything in between 0.0 and 1.0). Each set is tried in PPlan with or without the distance-to-goal heuristic. An AMD Phenom, with 2.5Ghz CPU, 4MB cache and 1Gb of available memory is used and stack and trail sizes in SWI Prolog are set to 128MB each.

Regarding the satisfiable set the comparison between the original PPlan and our extension is revealing of the effectiveness of our distance-to-goal heuristic. In Table 10.4 we simply average the running times over the satisfiable preferences. By increasing the minimum plan length, original PPlan's execution time increases exponentially, while the heuristic-enabled version remains low. The star (*) in the table means that the program run out of memory in all cases (within 8 hours of computation).

However, unsatisfiable and mixed preferences do not exhibit such encouraging results. Instead, when preferences are unsatisfiable or mixed, the performance of the heuristic may demonstrate remarkable fluctuations. Characteristic is the case of our model with minimum plan length 7, which we present through a boxplot in Figure 14. The graph summarizes experimental results with 13 different arbitrarily constructed unsatisfiable and mixed preference formulae. Observe that while the median of the heuristic-enabled version is lower than that of the original version, there is significant fluctuation as indicated by

Min. Plan Length	Original	Heuristic-Enabled
6	0.79 ± 0.02	0.04 ± 0.006
7	10.14 ± 0.05	0.05 ± 0.008
8	353 ± 3	0.07 ± 0.013
9	*	0.135 ± 0.013

Table 7. Performance for satisfiable preferences (behavioral analysis).

the great distance between the quintiles. Note that for minimum plan lengths of 10 and above, the running time would typically exceed our 8 hour limit..

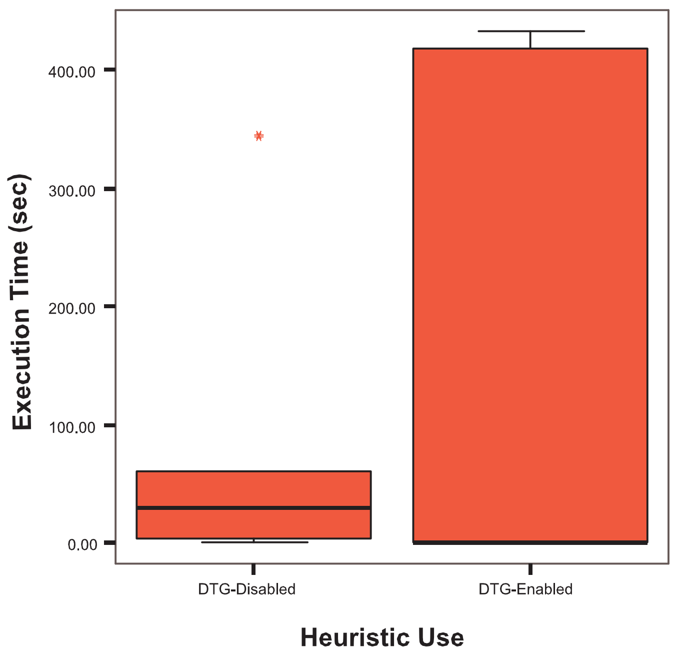


Figure 14. Performance for unsatisfiable and mixed preferences (behavioral analysis).

Our exploration reveals that the distance-to-goal heuristic exhibits such negative results when it builds a plan prefix (i.e. a partial plan) that leads to a situation with no solution or with a (“suddenly”) sub-optimal one. This can happen due to a choice that has been made early in the plan building process. The tenancy of our distance-to-goal heuristic to backtrack the progression towards plans that tend to be closer to the minimum distance-to-goal, may imply that the heuristic will take longer to correct the early choice. To confirm this we worked qualitatively. We developed the AND/OR tree of Figure 15. The “troubling” property of this tree is the precedence that connects t_1 with t_{13}. A preference specification of that includes a desire for both, for example, t_2 and t_{10} will obviously lead our search procedure in a “trap”: the planner will look for plans beginning from t_2 which however implies that t_{10} cannot be

Formula	**Original**	**Heuristic-Enabled**
$\Diamond t_2$	0.07	0.06
$\Diamond t_9$	2.26	0.24
$\Diamond t_{10}$	2.37	0.16
$\Diamond t_2 \times 0.6 + \Diamond t_{10} \times 0.4$	26.15	208.51
$\Diamond t_2 \times 0.7 + \Diamond t_4 \times 0.1 + \Diamond t_{10} \times 0.2$	26.08	211.09
$\Diamond t_2 \times 0.4 + \Diamond t_7 \times 0.6$	35.24	216.68

Table 8. Trapping the distance-to-goal heuristic (behavioral analysis).

performed. We are, then, questioning the ability of our distance-to-goal heuristic to escape the "trap" sooner than the original PPlan. In Table 10.4, the performance given particular preference formulae is given (times are in seconds). It is clear that while in simple formulae our heuristic performs better, formulae that lead the search routine in such traps can lead our heuristic to perform worse.

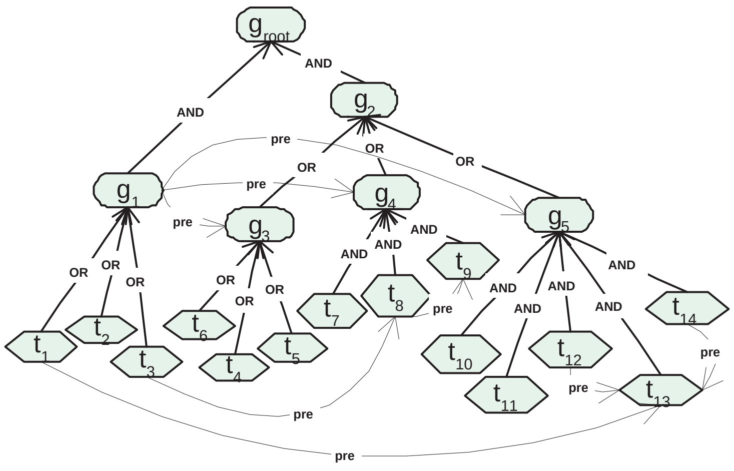


Figure 15. A goal model with non-local constraints.

As a last experimental step with the behavioral analysis component, we aimed at establishing the worst- and best- case performance boundaries of our heuristic-enabled tool. We experimented with a number of "artificial" goal models (i.e. models with dummy elements) but of several sizes, and arbitrarily constructed preference formulae. From the several randomly constructed goal structures, which vary in terms of the number of OR-decompositions and temporal constraints, we chose ones that appeared to generally worsen performance. We run the experiments on the same hardware infrastructure as above (a Pentium IV, with 2.5Ghz CPU and 1Gb memory). Using the number of leaf level tasks as an indication of the size of the goal model, we report in Table 9 the best and worse times we ever observed with any of the models we have tried, the symbol (*) meaning longer than 1 day of computation.

Thus, the evaluation of an arbitrary preference formula in an arbitrary goal tree is between these two extremes, without however necessarily reaching any of those extremes. For instance, the preference of Figure 7 over the 9-task model of Figure 1, is evaluated in about 1.5 seconds although it is unsatisfiable.

# tasks	6	7	8	9	10	20	30	40
Best	≤ 0.1	≤ 0.1	≤ 0.1	≤ 0.1	≤ 0.1	1	15	2m
Worse	11	24	10m	1.4h	*	*	*	*

Table 9. Worst and best case times (behavioral analysis)

Model (plan lgth / #alt)	Avg. (Behav.)	Avg. (Static)
Nursing (6/24)	0.1s	0.1s
ATM (16/32)	0.25s	0.4s
Bookseller (16/2)	0.2s	0.02s
Mtg. Sched. (12/4088)	4.3m	3.2m

Table 10. Realistic Model Results for Behavioral and Static Analyses

10.5 Performance Evaluation - Static Analysis

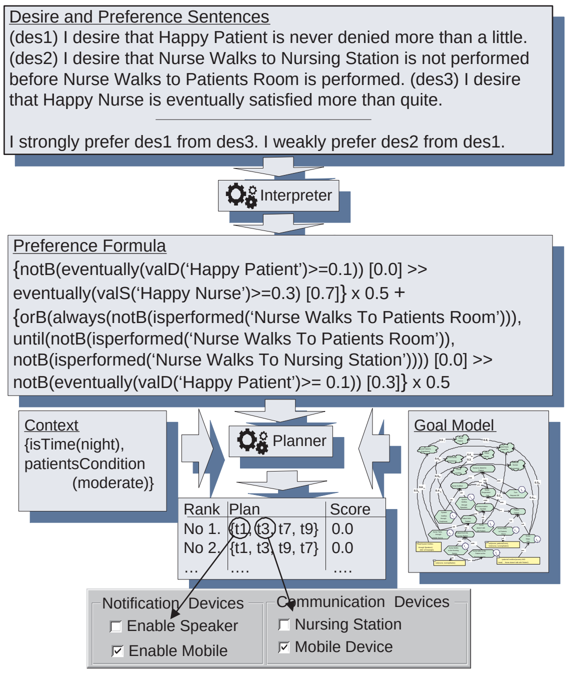
Performance of static analysis significantly deviates from the above results. Recall that for static analysis we perform brute-force exhaustive search of solutions of the AND/OR tree. Thus, performance is linear to the number of solutions of the AND/OR tree and thus exponential to the number of OR-decompositions the tree contains. We found that our tool requires about 50 sec to process 1,000 such alternatives, making it usable in all practical goal models we have worked with. It is interesting to observe that our brute-force approach allows rough estimation of the time that is required for a result to be returned, by simply multiplying the number of solutions of the AND/OR tree (which can be found by recursively adding the number of solutions of OR-rooted subtrees and multiplying the number of solutions of AND-rooted subtrees) by the time that the tool needs to process (check constraints, label propagate, calculate preference weights) each alternative. To compare performance of static analysis with best-case behavioral analysis, in Table 10, we provide the average running times for behavioral analysis over satisfiable preferences as well as the average time for static analysis observed while experimenting with our realistic models; the maximum plan length and number of alternatives of each model are given in parentheses.

11 Applications

11.1 Adding an Interaction Layer

The preference specification language we described has been designed to allow construction of arbitrarily complex temporal desires, which can be prioritized in several ways using real numbers as score values. In practice, however, users of the language may choose to use only part of its expressive power in return for simplicity and intuitiveness. Thus, the language can be seen as a low-level preference modeling infrastructure on top of which additional layers of elicitation mechanisms can be constructed. Probably the simplest example of implementing such a layer, is by constructing yet another preference specification language which is, however, much more abstract, less technical and therefore potentially useable by users without knowledge of LTL or of informed ways to come up with weights.

To show how this is possible we constructed such a language. It allows users to write sentences whereby they express their optional desires or preferences in structured English. To express desires, users can use elements of the goal model together with temporal prepositions, such as “before”, “until” or “after”. For soft-goals, in particular, reference to quantitative measure of satisfaction is replaced by qualitative labeling. The upper section of the top box of Figure 16, shows examples of three desires. Desire sentences are given unique identifiers (des1, des2 and des3 in the figure) to allow them be referred by preference sentences. The latter, seen at the bottom part of the top box of Figure 16, are used to rank desires subject to their relative importance. Again the quantitative measures of relative importance have been replaced by qualitative labels such as ‘strongly’ and ‘weakly’.



Desire and Preference Sentences

(des1) I desire that Happy Patient is never denied more than a little. (des2) I desire that Nurse Walks to Nursing Station is not performed before Nurse Walks to Patients Room is performed. (des3) I desire that Happy Nurse is eventually satisfied more than quite.

I strongly prefer des1 from des3. I weakly prefer des2 from des1.

Preference Formula

```
{notB(eventually(valD(‘Happy Patient’)>=0.1)) [0.0] >>
eventually(valS(‘Happy Nurse’)>=0.3) [0.7]} x 0.5 +
{orB(always(notB(isperformed(‘Nurse Walks To Patients Room’))),
until(notB(isperformed(‘Nurse Walks To Patients Room’)),
notB(isperformed(‘Nurse Walks To Nursing Station’)))) [0.0] >>
notB(eventually(valD(‘Happy Patient’)>= 0.1)) [0.3]} x 0.5
```

Context
{isTime(night),
patientsCondition
(moderate)}

Rank	Plan	Score
No 1.	{t1, t3, t7, t9}	0.0
No 2.	{t1, t3, t9, t7}	0.0
…	….	….

Notification Devices
☐ Enable Speaker
☑ Enable Mobile

Communication Devices
☐ Nursing Station
☑ Mobile Device

Figure 16. From Preference Sentences to Configurations

We constructed an interpreter that allows the translation of a set of such desire and preference sentences into the formal preference specification language. The interpreter deals with temporal prepositions (“before”, “until” etc) by considering the LTL pattern system, introduced in [7], which offers a mapping from informal expressions of high-level temporal properties to complex formulae in linear temporal logic. In addition, the interpreter is supplied with a mapping from qualitative labels to quantitative values for both soft-goal satisfaction and denial and for ranking weights. The construction of such a mapping depends on the intuition of the analysts. Thus, for our language, in terms of soft-goal satisfaction and denial value we assigned to expressions such as “a little”, “quite” and “a lot” numeric satisfaction or denial values such as 0.1, 0.3 and 0.7, respectively. In terms of weights of PFs, which can involve only two OCFs in our informal language, we associated terms such as “weakly” and “strongly” with numeric values chosen from the interval (0,1] for the least preferred OCF, and the value 0 for the most preferred one. Thus, “weakly prefer” implies that the least preferred OCF has a value of 0.3 while for “strongly” the least preferred OCF has a value of 0.7. Again, these choices for interpreting linguistic elements into numerical weights in preference formulae and for using PFs versus WPFs heavily rely on intuition. Our study of this semi-formal language, however, clearly illustrated that the complexity of our formal preference language can be hidden behind a custom-made preference communication mechanism.

11.2 Configuration

Apart from allowing exploration and understanding of the impact of stakeholder attitudes to design decisions, our framework also has a potential application to the configuration of the software artifact that results from the requirements and design process. In [22] we showed that it is possible under certain assumptions to associate alternative configuration options of a common software system with leaf-level elements of goal decomposition trees. When such a mapping is possible in the domain of interest, preference specification and analysis can be used for configuring the corresponding software system. From the output of the reasoning procedure, the top plan can be selected and, in turn, the configurations that map with the selection of the tasks that comprise the plan are constructed.

At the bottom of Figure 16, the configuration screen of a hypothetical nurse notification and communication system for our case study is presented. The high level preference sentences presented at the top box, have translated into a ranking of preferred plans, the first of which is used to define the current configuration. Thus, the absence of the task *System Notifies through Speakers*, which is due to the fact that it is night and there is a preference not to disturb the patients, is interpreted into a disabled speaker system of the unit at the configuration level. Notice that the same preference may yield a different preferred plan if it is not night-time, which would, in turn, imply enabled speakers.

12 Preliminary Evaluation

The main source of feasibility evidence comes from our case study on the nursing domain. The study is an exploratory one, as the domain investigation was unfolding during the initial development of our framework without a-priori hypotheses. The domain investigation process involved a series of interviews with health professionals. Acting as goal analysts we iteratively developed goal models and temporally extended them to a point where we thought that the result would reflect our understanding of the domain. We then attempted to construct simple preference formulae (using the formal low-level language) for each stakeholder, again based on our sense of what the desires and preferences of the stakeholders we interviewed were. Both the goal model and the preference formulae were iteratively revised by using the tool and testing the resulting plan rankings against our intuition of the domain.

We felt that the process converged to satisfactory goal and preference models. The use of the reasoning tool helped us explore the domain of possible solutions. It proved particularly useful for understanding the conflicts between stakeholder desires and how they affect the choice of the end solution. The desires of the nurses and the managers, for example, although they don't seem to be related when stated as high-level statements, they are found to be conflicting when interpreted into operational details, as we illustrated in Section 8.

We found, however, that more research needs to be done towards the development of instruments for systematic acquisition for: a) the weights and qualities (PFs versus WPFs) of the preference formulae, b) the weights of the contribution links in the goal model and the comparison constants for $valS()$ and $valD()$ in the preference formulae. Regarding the choice of weights in preference formulae, as we saw earlier, the software engineering community has shown that the elicitation of priorities amongst competing requirements is possible in a variety of ways. Although the concepts under comparison in existing requirements prioritization frameworks are coarse-grained software features, we believe that the same or similar techniques should also be effective for eliciting and quantifying priorities over behavioral and quality properties of the type we present here. The quality of the preference formula also depends on the elicitation method, PFs reflecting ranking-based approaches to preference, as discussed in economics (e.g. [30]) and WPFs being closer to weight-based formalizations of relative importance (e.g. AHP-style [20]). In terms of choosing weights for the contribution links and the satisfaction or denial of soft-goals, in our applications we often found convenient to limit ourselves to three or four equidistant values from the interval [0,1] which we would intuitively and uniformly throughout our analysis associate with terms such as “a little”, “a lot” etc. The usefulness of such a mapping between linguistic terms and weights became more evident in our experimentation with the semi-formal language we presented in Section 11. However, we believe that more research needs to be done towards a formal acquisition process of such mappings based on, for example, quantitative questionnaires.

In addition to the nursing domain (size: 31 goal elements) we also tried the same process on other generic domains, without however employing a realistic investigation process. Thus, we created models for the meeting scheduling problem (65 elements), for the ATM domain (34 elements), as well as for a (hypothetical) on-line bookstore (34 elements). Despite the lack of actual input from stakeholders, which would offer an increased sense of validity of the modeling result, our application on those examples offered further evidence that our modeling and analysis approach is feasible.

13 Conclusions

Evaluating alternative solutions subject to stakeholder priorities and preferences is an integral part of the requirements analysis process. The main contributions of this paper are the introduction of a goal-oriented approach for modeling temporally extended optional and preference requirements and a proposal for using such requirements for exploring and evaluating alternative solutions. We demonstrated possibilities for tool support both at the level of reasoning about alternatives, and at the level of acquiring preference specifications through structured English. Through our practical applications, we found that the overall framework is useful for understanding the impact of the attitudes of different stakeholders to the selection of a solution, and, potentially, for allowing automatic derivation of the appropriate low-level design and configuration choices.

For the future we need to better understand the merits and weaknesses of the proposed framework by applying it at larger scale case studies. Firstly, alternative preference elicitation processes need to be explored, by potentially adopting existing work from areas such as AI, Economics and Psychology. Secondly, we plan to develop and experiment on a concrete implementation of a preference-based software configuration tool. And, thirdly, we intend to explore ways to improve the performance of our reasoning tool. Recent advances in preference-based planning are only encouraging towards this direction.

References

[1] L. Ardissono, G. Friedrich, A. Goy, M. Holland, G. Petrone, C. Russ, and R. Schaefer. User-adaptive configuration of products and services. In *Proceedings of the IJCAI’03 workshop on Configuration*, Acapulco, Mexico, 2003.

[2] P. Avesani, C. Bazzanella, A. Perini, and A. Susi. Facing scalability issues in requirements prioritization with machine learning techniques. In *Proceedings of the 13th IEEE International Conference On Requirements Engineering (RE'05)*, 2005.

[3] K. Beck. *Extreme Programming Explained*. Addison Wesley, 1999.

[4] M. Bienvenu, C. Fritz, and S. McIlraith. Planning with qualitative temporal preferences. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR06)*, June 2006.

[5] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged configuration using feature models. In *Proceedings of the Third Software Product Line Conference*, pages 266–283, 2004.

[6] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993.

[7] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *ICSE '99: Proceedings of the 21st International Conference on Software Engineering*, pages 411–420, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[8] S. R. Faulk. Product-line requirements specification (PRS): An approach and case study. In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE'01)*, pages 48–55, 2001.

[9] A. Felfernig, G. Friedrich, and D. Jannach. Conceptual modeling for configuration of mass-customizable products. *Artificial Intelligence in Engineering*, 15(2):165–176, 2001.

[10] D. Fischbein, S. Uchitel, and V. Braberman. A foundation for behavioural conformance in software product line architectures. In *Proceedings of the ISSTA 2006 workshop on Role of software architecture for testing and analysis (ROSATEA'06)*, pages 39–48, New York, NY, USA, 2006.

[11] A. Gabaldon. Precondition control and the progression algorithm. In D. Dubois, C. Welty, and M. Williams, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pages 634–643. AAAI Press, 2004.

[12] G. Gans, M. Jarke, G. Lakemeyer, and T. Vits. Snet: A modeling and simulation environment for agent networks based on i* and ConGolog. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE'02), LNCS 2348*, Toronto, Canada, 2002.

[13] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Reasoning with goal models. In *Proceedings of the 21st International Conference on Conceptual Modeling (ER'02)*, pages 167–181, London, UK, 2002.

[14] G. Halmans and K. Pohl. Communicating the variability of a software-product family to customers. *Software and System Modeling*, 2(1):15–36, 2003.

[15] B. Hui, S. Liaskos, and J. Mylopoulos. Requirements analysis for customizable software: A goals-skills-preferences framework. In *Proceedings of the 11th IEEE International Requirements Engineering Conference (RE'03)*, 2003.

[16] H. P. In, D. Olson, and T. Rodgers. Multi-criteria preference analysis for systematic requirements negotiation. In *Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC'02)*, 2002.

[17] M. Jackson. The meaning of requirements. *Annals of Software Engineering*, 3:5–21, 1997.

[18] U. Junker. Preference programming for configuration. In *IJCAI-01 Workshop on Configuration*, 2001.

[19] I. Jureta, J. Mylopoulos, and S. Faulkner. Revisiting the core ontology and problem in requirements engineering. In *Proceedings of the 16th IEEE International Conference on Requirements Engineering (RE'08)*, pages 71–80, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[20] J. Karlsson and K. Ryan. A cost-value approach for prioritizing requirements. *IEEE Software*, 14(5), 1997.

[21] G. Klein and P. O. Beck. A decision aid for selecting among information system alternatives. *MIS Quarterly*, 11(2):177–185, June 1987.

[22] S. Liaskos, A. Lapouchnian, Y. Wang, Y. Yu, and S. Easterbrook. Configuring common personal software: a requirements-driven approach. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, Paris, France, 2005.

[23] S. Liaskos, S. McIlraith, and J. Mylopoulos. Representing and reasoning with preference requirements using goals (revision). Technical Report CSRG-542, Dept. of Computer Science, University of Toronto, ftp://ftp.cs.toronto.edu/pub/reports/csrg/542, 2008.

[24] S. McIlraith. Integrating actions and state constraints: A closed-form solution to the ramification problem (sometimes). *Artificial Intelligence*, 116(1-2):87–121, January 2000.

[25] J. Mylopoulos, L. Chung, S. Liao, H. Wang, and E. Yu. Exploring alternatives during requirements analysis. *IEEE Software*, 18(1):92–96, 2001.

[26] R. Reiter. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[27] V. Renneberg, R. Stegmann, and T. Leckner. Recommending personalized product configurations based on product scorings. In *Proceedings of the ECAI'04 Workshop on Configuration.*, August 2004.

[28] D. Sabin and R. Weigel. Product configuration frameworks-a survey. *IEEE Intelligent Systems*, 13(4):42–49, 1998.

[29] R. Sebastiani, P. Giorgini, and J. Mylopoulos. Simple and minimum-cost satisfiability for goal models. In *Proceedings of the 16th Conference On Advanced Information Systems Engineering (CAiSE'04)*, 2004.

[30] A. K. Sen. Behaviour and the concept of preference. *Economica*, 40(159):241–59, August 1973.

[31] A. G. Sutcliffe, N. A. M. Maiden, S. Minocha, and D. Manuel. Supporting scenario-based requirements engineering. *IEEE Transactions on Software Engineering*, 24(12):1072–1088, 1998.

[32] T. Ziadi and L. Hélouët and J.-M. Jézéquel. Modeling behaviors in product lines. In *International Workshop on Requirements Engineering for Product Lines (REPL)*, pages 33–38, September 2002.

[33] X. Wang and Y. Lesperance. Agent-oriented requirements engineering using ConGolog and i*. In *AOIS-2001, Bi-Conference Workshop at Agents 2001 and CAiSE'01.*, 2001.

[34] E. S. K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of the 3rd IEEE Int. Symposium on Requirements Engineering (RE'97)*, Washington D.C., USA, January 1997.

[35] E. S. K. Yu and J. Mylopoulos. Understanding "why" in software process modelling, analysis, and design. In *Proceedings of the Sixteenth International Conference on Software Engineering (ICSE'94)*, pages 159–168, 1994.

[36] H. Zhang, S. Jarzabek, and B. Yang. Quality prediction and assessment for product lines. In *15th International Conference on Advanced Information Systems Engineering (CAiSE)*, pages 681–695, 2003.