Reasoning with Constrained Goal Models

Mai Chi Nguyen, Roberto Sebastiani, Paolo Giorgini, and John Mylopoulos
DISI, University of Trento, Italy

Abstract. Goal models have been widely used in Computer Science to represent software requirements, business objectives, and design qualities. Existing goal modeling techniques, however, have shown limitations of expressiveness and/or tractability in coping with complex real-world problems. In this work we exploit advances in automated reasoning technologies, notably SMT solvers, to propose and formalize: (i) an extended notion of goal model, namely Constrained Goal Models (CGMs), which makes explicit the notion of goal refinement and allows for constraints and penalties/rewards over goals and their refinements; (ii) a novel set of automated reasoning functionalities over CGMs, allowing for automatically generating suitable refinements of input CGMs, under user-specified assumptions and constraints, that also optimize given penalty/reward functions. We have implemented these modeling and reasoning functionalities in a prototype tool, using the Optimization Modulo Theory solver OptiMathSAT as automated reasoning backend.

1 Introduction

The concept of goal has long been used as useful abstraction in many areas of Computer Science, such as for example in Artificial Intelligence (AI) planning [17], agent-based systems [20], and knowledge management [12]. More recently, software engineering has also been using goal to model requirements for software systems, business objectives for enterprises, and design qualities [1, 6, 9].

Goal-oriented requirements engineering approaches have gain popularity in the last decade for a number of significant benefits in conceptualizing and analyzing requirements [24]. Goal models provide a wider system engineering perspective compared to the traditional requirements engineering methods, a precise criterion for completeness, and rationale for requirements specification, as well as automated support for early requirements analysis. Moreover, goal models are useful in explaining requirements to stakeholders, and goal refinements offer an accessible level of abstraction for decision makers in validating choices among alternative designs.

Current goal modeling and reasoning techniques, however, have limitations in coping with complex real-world problems, as recently highlighted by Horkoff and Yu in [10]. Leading approaches such as KAOS [6] and i* [25] are limited in expressing stakeholder preferences, but also in supporting scalable reasoning over goal models. More recent proposals, such as Techne [13] and [15] propose expressive extensions to goal models, but lack scalable reasoning facilities.

As an answer to the need for more expressiveness and more sophisticated reasoning support, we propose to exploit advances in automated reasoning technologies, notably
SMT solvers, to define an extended notion of goal model, namely Constrained Goal Model (CGM). CGMs offer an explicit notion of goal refinement, introduce the concept of domain assumption, and support the definition of arbitrary constraints. CGMs also allow stakeholders to express preferences by distinguishing between mandatory and optional requirements and assigning preference weights to goals and domain assumptions. Taking advantage of the formal semantics of CGMs and of the expressiveness and efficiency of Satisfiability Modulo Theories (SMT) [2] and Optimization Modulo Theories (OMT) [23] solvers, we also provide a set of automated reasoning functionalities on CGMs. Particularly, our approach allows for any given set of stakeholders’ assertions: the automatic check of the realizability of CGM; the interactive/automatic search for a refinement for a CGM; the automatic enumeration of possible realizations of a CGM; and, the automatic search for the “best” realization(s) in terms of penalties/rewards of a CGM under user assertions. Our approach is implemented as a prototype tool (CGM-Tool), a standalone java application based on the Eclipse RCP engine. The tool offers functionalities to create CGM models as graphical diagrams and to explore alternative scenarios running automated reasoning techniques. CGM-Tool uses the SMT/OMT solver OptiMathSAT [23] as automated reasoning backend.

The structure of the paper is as follows: §2 provides the necessary background on goal modeling and on SMT/OMT; §3 introduces the notion of CGM through an example; §4 introduces the syntax and semantics of CGMs; §5 presents the set of automated reasoning functionalities for CGMs; §6 gives a quick introduction of our prototype tool based on the presented approach; §7 gives some overview of related works; in §8 we draw some conclusions and present future research challenges.

2 Background

Our research baseline consists of our previous work on qualitative goal models and Satisfiability and Optimization Modulo Theories (SMT and OMT respectively). Our aim in this section is to introduce enough notions and results from this work so that the reader can follow the narrative in subsequent sections.

Goal Models. Qualitative goal models is introduced in [16], where the concept of goal is used to represent a desired state of affairs (aka a requirement) in terms of a proposition. A goal can be refined by means of AND/OR refinement relationships and qualitative evidence (strong and weak) for/against the fulfilment of a goal is provided by contribution links labelled +, − etc. In [9], the goal model is formalized by replacing each proposition g, standing for a goal, by four propositions (FSg, PSg, PDg, FDg) representing full (and partial) evidence for the satisfaction/denial of g. A traditional implication such as p ∧ q → r is then translated into a series of implications connecting these new symbols, including $FS_p \land FS_q \rightarrow FS_r$, $PS_p \land PS_q \rightarrow PS_r$, as well as $FD_p \rightarrow FD_r$, $FD_q \rightarrow FD_r$, etc. The conflict between a and b is captured by axioms of the form $FS_a \rightarrow FD_b$, and it is consistent to have both $FS_a$ and $FS_b$ evaluated to true at the same time. As a result, even though the semantics of a goal model is a classical propositional theory, inconsistency does not result in everything being true. In fact, a predicate g can be assigned a subset of truth values \{FS, PS, FD, PD\}.  

2
[21] extended the approach further by including axioms for avoiding conflicts of the form $FS_a \land FD_a$. The approach recognized the need to formalize goal models so as to automatically evaluate the satisfiability of goals. These goal models, however, do not incorporate the notion of conflict as inconsistency, they do not include concepts other than goals, cannot distinguish optional from mandatory requirements and have no notion of a robust solution, i.e. solution without "conflict", where a goal can not be (full or partial) denied and (respectively, full or partial) satisfied at the same time.

Satisfiability and Optimization Modulo Theories. Satisfiability Modulo Theories (SMT) is the problem of deciding the satisfiability of a quantifier-free first-order formula $\Phi$ with respect to some decidable theory $T$ (see [22, 2]). An Optimization Modulo Theories (OMT) problem $(\Phi, cost)$ is the problem of finding a solution to an SMT formula $\Phi$ which minimizes the cost function $cost$ (see [18, 23]).

In this paper we focus on the theory of linear arithmetic over the rationals, $LRA$: $SMT(LRA)$ is the problem of checking the satisfiability of a formula $\Phi$ consisting in atomic propositions $A_1, A_2, \ldots$ and constraints like "$(2.1 x_1 - 3.4 x_2 + 3.2 x_3 \leq 4.2)$", combined by means of Boolean operators $\neg, \land, \lor, \rightarrow, \leftrightarrow$; $OMT(LRA)$ is the OMT problem s.t. $\Phi$ and $cost$ are respectively a $SMT(LRA)$ formula and a $LRA$ cost function. Very efficient $SMT(LRA)$ and $OMT(LRA)$ solvers are available, which combine the power of SAT solvers with dedicated linear-programming decision and minimization procedures (see [22, 2, 18, 23]).

3 Constrained Goal Models

We introduce the main ideas of constrained goal models (CGMs) through a meeting schedule example (Figure 1).

Notationally, round-corner rectangles (e.g., ScheduleMeeting) are root goals, representing stakeholder requirements; ovals (e.g. CollectTimetables) are intermediate goals; hexagons (e.g. CharacteriseMeeting) are tasks, i.e. non-root leaf goals; rectangles (e.g., ParticipantsUseSystemCalendar) are domain assumptions. We call elements both goals and domain assumptions. Labeled bullets at the merging point of the edges connecting a group of source elements to a target element are refinements (e.g., $(GoodParticipation, MinimalConflict) \xrightarrow{R_{20}} GoodQualitySchedule$), while the $R_i$s denote their labels.\footnote{The label of a refinement can be omitted when there is no need to refer to it explicitly.}

Intuitively, requirements represent desired states of affairs we want the system-to-be to achieve (either mandatorily or possibly); they are progressively refined into intermediate goals, until the process produces actionable goals (tasks) that need no further decomposition and can be executed; domain assumptions are propositions about the domain that need to hold for a goal refinement to work. Refinements are used to represent the alternatives of how to achieve an element; a refinement of an element is a conjunction of the sub-elements that are necessary to achieve the element.

The main objective of the CGM is to achieve the requirement ScheduleMeeting, which is mandatory. ScheduleMeeting has only one candidate refinement $R_1$, consisting in five sub-goals: CharacteriseMeeting, CollectTimetables, FindASuitableRoom, FindASuitableSchedule, GoodQualitySchedule.
ChooseSchedule, and ManageMeeting. Since $R_1$ is the only refinement of the requirement, all these sub-goals must be satisfied in order to satisfy it. There may be more than one way to refine an element: e.g., CollectTimetables is further refined either by $R_{10}$ into the single goal ByPerson or by $R_2$ into the single goal BySystem. Similarly, FindASuitableRoom and ChooseSchedule have three and two possible refinements respectively. The subgoals are further refined until they reach the level of domain assumptions and tasks.

Some requirements can be optional, like LowCost, MinimalEffort, FastSchedule, and GoodQualitySchedule (in blue in Figure 1). They are requirements that we would like to include in our realizations, provided they do not cause too much effort to achieve and/or they are not in conflict with other requirements. To this extent, in order to analyze interactively the possible different realizations, one can mark [or unmark] requirements as satisfied, thus making them mandatory (if not so, they are optional). Similarly, one can interactively mark/unmark (effortful) tasks as denied, or mark/unmark some domain assumption as satisfied or denied. We call these marks user assertions.

**Constraints in a CGM.** Importantly, in a CGM elements and refinements are enriched by user-defined constraints, which can be expressed either graphically as relation edges or textually as logic formulas.

We have three kinds of relation edges. *Contribution edges* $\langle E_i \xrightarrow{+} - \rightarrow E_j \rangle$ between elements (in green), like “ScheduleAutomatically $\xrightarrow{+}$ MinimalConflicts”, mean that if the source element $E_i$ is satisfied, then also the target element $E_j$ must be satisfied (but not vice versa). *Conflict edges* $\langle E_i \leftarrow - \rightarrow E_j \rangle$ between elements (in red), like “ConfirmOccurrence $\leftarrow$ CancelMeeting”, mean that the two elements $E_i$ and $E_j$ cannot be both satisfied. *Refinement bindings* $\langle R_i \leftarrow - \rightarrow R_j \rangle$ between two refinements (in purple), like “$R_2 \leftarrow - \rightarrow R_7$”, are used to state that, if the target elements $E_i$ and $E_j$ of the two refinements $R_i$ and $R_j$ respectively are both satisfied, then $E_i$ is refined by $R_i$ if and only if $E_j$ is refined by $R_j$. Intuitively, this means that the two refinements are binded, as if they were two different instances of the same choice.

It is possible to enrich CGMs with logic formulas, representing arbitrary logic constraints on elements and refinements (plus possibly others). Such constraints can be global or local to elements and refinements, that is, each goal $G$ can be tagged with a pair of prerequisite formulas $\{\phi^+_G, \phi^-_G\}$, so that $\phi^+_G$ [resp. $\phi^-_G$] must be satisfied if we want $G$ to be satisfied [resp. denied]. (The same holds for each requirement $R$.)

For example, suppose that, in order to achieve the optional requirement LowCost we need a meeting to cost less than 100€, and that we know that CallParticipants costs 30€, that using a partner institution room costs 80€ (CallPartnerInstitutions), whilst using a hotel or convention center room costs 200€ (CallHotelsAndConventionCenters). Thus, to achieve LowCost, we cannot choose the task CallHotelsAndConventionCenters, and we cannot choose both tasks CallParticipants and CallPartnerInstitutions. To enforce this fact, one can tag LowCost with the prerequisite Boolean formula:

$$\phi^+_{\text{LowCost}} \overset{\text{def}}{=} \neg \text{CallHotelsAndConventionCenters} \wedge \neg (\text{CallPartnerInstitutions} \wedge \text{CallParticipants}).$$ (1)
Fig. 1. An example of a CCM
or, equivalently, add globally to the CGM the following Boolean formulas:

\[
\begin{align*}
\text{LowCost} & \rightarrow (\neg \text{CallHotelsAndConventionCenters}) \quad (2) \\
\text{LowCost} & \rightarrow (\neg (\text{CallPartnerInstitutions} \land \text{CallParticipants})). \quad (3)
\end{align*}
\]

(Notice that there is no way we can express (1) and (3) with the relation edges above.)

Analogous considerations based on time rather than on costs can be imposed to constrain the optional requirement FastSchedule, e.g., to require that ScheduleManually and ByPerson cannot be both satisfied:

\[
\phi^+_{\text{FastSchedule}} \overset{\text{def}}{=} \neg (\text{ScheduleManually} \land \text{ByPerson}). \quad (4)
\]

Alternatively, one can express the same facts much more straightforwardly and intuitively by using SMT(LRA) constraints: e.g., for (1) one can introduce three numerical variables \(\text{cost, cost calls, cost rooms} \geq 0\), add the global constraint \((\text{cost} = \text{cost calls} + \text{cost rooms})\) and the prerequisite SMT(LRA) constraints:

\[
\begin{align*}
\phi^+_{\text{LowCost}} & \overset{\text{def}}{=} (\text{cost} < 100) \\
\phi^+_{\text{CallParticipants}} & \overset{\text{def}}{=} (\text{cost calls} = 30) \\
\phi^+_{\text{CallPartnerInstitutions}} & \overset{\text{def}}{=} (\text{cost rooms} = 80) \\
\phi^+_{\text{CallHotelsAndConventionCenters}} & \overset{\text{def}}{=} (\text{cost rooms} = 200).
\end{align*}
\]

Notice that, unlike with (1), in order to conceive (5) one is not supposed to do any reasoning on Boolean relations among elements, leaving it to the automated reasoner.

**Realizations of a CGM.** We suppose now that ScheduleMeeting is marked satisfied (i.e. it is mandatory) and that no other element is marked. Then the CGM in Figure 1 has more than 20 possible realizations. The sub-graph which is highlighted in yellow describes one of them.

Intuitively, a realization of a CGM (if any) represents one of the alternative ways of refining the mandatory requirements (plus possibly some of the optional ones) in compliance with the user’s assertions. It is a sub-graph of the CGM including a set of satisfied elements and refinements: it includes all mandatory requirements, it includes [resp. does not include] all elements satisfied [resp. denied] in the user’s assertions; for each non-leaf element included, at least one of its refinement are included; for each refinement included, all its target elements are included; finally, a realization complies with all relation edges and formulas.

Apart from the mandatory requirement, the realization in Figure 1 allows to achieve also the optional requirements LowCost, GoodQualitySchedule, but not FastSchedule and MinimalEffort. In order to obtain these requirements, it requires accomplishing the tasks CharacteriseMeeting, EmailParticipants, ListAvailableRooms, UseAvailableRoom, ScheduleManually, ConfirmOccurrence, GoodParticipation, MinimalConflicts, and requires the domain assumption LocalRoomAvailable. The realization is optimal, in the sense that no requirement can be added without adding some more task, and no task can be dropped without dropping some requirement.
Weighted CGMs. In general, a CGM has many realizations. To distinguish among them, stakeholders may want to capture preferences on the optional requirements to achieve and on the tasks to accomplish.

To this end, the user can assign positive weights (penalties) to leaf elements and negative weights (rewards) to non-mandatory requirements (the numbers in Figure 1). If so, an OMT-based automated-reasoning tool can return a realization which minimizes its weight, that is, the total difference between the penalties and rewards. For instance, with the realization in figure, penalties $- rewards$ is $185 - 180 = 5$. Notice that the realization in Figure 1 is not a minimum-weight one. In fact, the minimal-weight realization of the example CGM, which was found by our automated tool (discussed in §6), achieves all the optional requirements with the weight of $-65$. Such realization requires accomplishing the tasks CharacteriseMeeting, CollectFromSystemCalendar, CallPartnerInstitutions, ScheduleAutomatically, ConfirmOccurrence, GoodParticipation, MinimalConflicts, MatchingEffort, and CollectionEffort, and requires the domain assumption ParticipantsUseSystemCalendar.

4 Abstract Syntax and Semantics

4.1 Constrained Goal Models

We call a goal graph \(D\) a directed acyclic graph (DAG) alternating element nodes (hereafter “elements”) and refinement nodes (“refinements”, collapsed into bullets), s.t.: 

(a) each element has from zero to many outcoming edges to distinct refinements and from zero to many incoming edges from distinct refinements; 

(b) each refinement node has exactly one outgoing edge to an element (target) and one or more incoming edges from distinct elements (sources).

Elements are either goals or domain assumptions, subject to the following constraints: a domain assumption cannot be a root element; if the target of a refinement \(R\) is a domain assumption, then it sources must be only domain assumptions; if the target of a refinement \(R\) is a goal, then at least one of its sources must be a goal. We call root goals and leaf goals requirements and tasks respectively. Notationally, we use the symbols \(R, R_j\) for labeling refinements, \(E, E_i\) for generic elements (without specifying if goals or domain assumptions), \(G, G_i\) for goals, \(A, A_i\) for domain assumptions.

Definition 1 (Constrained Goal Model). A Constrained Goal Model (CGM) is a tuple \(\mathcal{M} \equiv (B, \mathcal{N}, D, \Psi)\), s.t.

- \(B \equiv \mathcal{G} \cup \mathcal{R} \cup \mathcal{A}\) is a set of atomic propositions, where \(\mathcal{G} \equiv \{G_1, ..., G_N\}\), \(\mathcal{R} \equiv \{R_1, ..., R_K\}\), \(\mathcal{A} \equiv \{A_1, ..., A_M\}\) are respectively sets of goal labels, refinement labels, and domain assumption labels. We denote with \(\mathcal{E} \equiv \mathcal{G} \cup \mathcal{A}\); 

- \(\mathcal{N}\) is a set of numerical variables in the rationals; 

- \(D\) is a goal graph, s.t. all its goal nodes are univocally labeled by a goal label in \(\mathcal{G}\), all its refinements are univocally labelled by a refinement label in \(\mathcal{R}\), and all its domain assumption are univocally labeled by a assumption label in \(\mathcal{A}\); 

- \(\Psi\) is a SMT(\(\mathcal{LRA}\)) formula on \(B\) and \(\mathcal{N}\).
A CGM is thus an and-or directed acyclic graph (DAG) of elements, as nodes, and refinements, as (grouped) edges, which are labeled by atomic propositions and can be augmented with arbitrary constraints in form of $\text{SMT(\mathcal{LRA})}$ formulas – typically conjunctions of smaller global and local constraints – on the element and refinement labels and on the numerical variables. Intuitively, a CGM describes a (possibly complex) combination of alternative ways of realizing a set of requirements in terms of a set of beliefs and on the numerical variables. Notice that CGMs are more succinct than standard and-or decompositions: whilst an standard $n$-ary and-decomposition of a goal can be represented straightforwardly in a CGM by one refinement with $n$ sources [resp. $n$ one-source refinements], representing a CGM decomposition with $n$ non-ary refinements by means of standard and-or decomposition requires introducing $n$ new intermediate goals.

In general, the user might not be at ease in defining a possibly-complex global $\text{SMT(\mathcal{LRA})}$ formula $\Psi$ to encode constraints among elements and refinements, plus numerical variables. To this extent, as mentioned in §3, apart from the possibility of defining global formulas, CGMs provide some constructs allowing the user to encode easily and locally some desired constraints of frequent usage: relation edges, prerequisite constraints $\{\hat{\phi}_G^+, \hat{\phi}_G^−\}$ and $\{\hat{\phi}_R^+, \hat{\phi}_R^−\}$ and user’s assertions. Each is automatically converted into a simple $\text{SMT(\mathcal{LRA})}$ formula as follows, and then conjoined to $\Psi$.

- **Element-contribution edges**, $E_1 \xrightarrow{+} E_2$, are encoded into the formula $(E_1 \rightarrow E_2)$.
- **Element-conflict edges**, $E_1 \xleftarrow{−} E_2$, are encoded into the formula $\neg(E_1 \land E_2)$.
- **Refinement-binding edges**, $R_1 \xleftarrow{−} R_2$, s.t. $E_1, E_2$ are the target elements of $R_1, R_2$ respectively, are encoded into the formula $(E_1 \land E_2) \rightarrow (R_1 \leftrightarrow R_2)$.
- **Prerequisite constraints**, $\{\hat{\phi}_G^+, \hat{\phi}_G^−\}$ [resp. $\{\hat{\phi}_R^+, \hat{\phi}_R^−\}$] are encoded into the formulas $(G \rightarrow \phi_G^+)$ and $(\neg G \rightarrow \phi_G^−)$ [resp. $(R \rightarrow \phi_R^+)$ and $(\neg R \rightarrow \phi_R^−)$].
- **User’s assertions**, $E_i := \top$ and $E_i := \perp$, are encoded into the formulas $(E_i)$, $(\neg E_i)$.

Notice that, unlike refinements, relation edges are allowed to create loops, possibly involving refinements. It is easy to provide the user the possibility of defining, both globally and locally, more general and intuitive Boolean constraints (e.g., Requires $(E_1, E_2)$, AtMost $(N, \{E_1, ..., E_n\})$) with no need for the user to define the corresponding complicate propositional formulas.

The semantics of CGMs is formally defined as follows.

**Definition 2 (Realization of a CGM).** Let $\mathcal{M} \equiv (\mathcal{B}, \mathcal{D}, \Psi)$ be a CGM. A realization of $\mathcal{M}$ is a $\mathcal{LRA}$-interpretation $\mu$ over $\mathcal{B} \cup \mathcal{N}$ such that: $^2$

(a) $\mu \models ((\bigwedge_{i=1}^n E_i) \leftrightarrow R) \land (R \rightarrow E)$ for each refinement $(E_1, ..., E_n) \xrightarrow{R} E$;
(b) $\mu \models (E \rightarrow (\bigvee_{R_i \in \text{Ref}(E)} R_i))$, for each non-leaf element $E$;
(c) $\mu \models \Psi$.

We say that $\mathcal{M}$ is realizable if it has at least one realization, unrealizable otherwise.

$^2$ A $\mathcal{LRA}$-interpretation $\mu$ is a function which assigns truth values to Boolean atoms and rational values to numerical variables. “$\mu \models \Phi$” means that $\mu$ makes the formula $\Phi$ evaluate to true.
In a realization, each element $E$ or refinement $R$ can be either satisfied or denied (i.e., their label can be assigned to $\top$ or $\bot$ respectively by $\mu$). If an element $E$ is not a leaf, then it can be satisfied only by satisfying the set of source elements $E_1, \ldots, E_n$ of one of its refinements $(E_1, \ldots, E_n) \xrightarrow{R} E$. If $\mu$ satisfies a refinement $R$ of an element $E$, i.e., it satisfies all the source elements $E_1, \ldots, E_n$, then it satisfies the element $E$, but not vice versa (condition (a)). For a non-leaf element to be satisfied, at least one of its refinements must be satisfied (condition (b)). We call this fact closed world assumption (CWA). The satisfiability or deniability of each element or refinement can be further constrained by all the constraints defined inside the formula $\Psi$: every realization $\mu$ must satisfy such constraints (condition (c)). Notice that, by fulfilling condition (c), a realization must implicitly comply also with all the relation edges, with the local prerequisite constraints and with the user’s assertions, because the corresponding formulas are conjuncts of $\Psi$.

A realization $\mu$ for a CGM $\mathcal{M} \triangleq \langle B, N, D, \Psi \rangle$ is represented graphically as the sub-graph of $D$ where all the denied element and refinement nodes are eliminated.

### 4.2 Weighted Constrained Goal Models

**Definition 3 (Weighted CGM).** A Weighted Constrained Goal Model (WCGM) is a tuple $\langle B, N, D, \Psi, W \rangle$, s.t. $\langle B, N, D, \Psi \rangle$ is a CGM and $W : G \rightarrow \mathbb{Q}$ is a function which assigns to each element $E_i \in E$ a rational number, such that:

- $W(E_i) \geq 0$ if $E_i$ is a leaf element (task or assumption),
- $W(E_i) = 0$ if $E_i$ is an intermediate element,
- $W(E_i) \leq 0$ if $E_i$ is a root goal (requirement).

We call $W(E_i)$ the weight of $E_i$; if $W(E_i)$ is strictly positive, we also call it the penalty of the (leaf) element $E_i$; if it is strictly negative, we call $-W(E_i)$ the reward of the requirement $E_i$. When $W(E_i)$ is not specified, we assume by default $W(E_i) = 0$.

**Definition 4 (Realization, weight of a realization and minimum weight of a WCGM.).** Let $\mathcal{M} \triangleq \langle B, N, D, \Psi, W \rangle$ be a WCGM. A realization $\mu$ of $\mathcal{M}$ is a realization of $\mathcal{M}'$, s.t. $\mathcal{M}' \triangleq \langle B, N, D, \Psi \rangle$ is the un-weighted version of $\mathcal{M}$. We call the weight of $\mu$, denoted as $\text{WeightOf}(\mu)$, the value

$$\sum_{E_i \in E, \mu(E_i) = \top} W(E_i).$$

We call the minimum weight of $\mathcal{M}$, denoted as $\text{MinWeightOf}(\mathcal{M})$, the value:

$$\min_{\mu \text{ realization of } \mathcal{M}} \text{WeightOf}(\mu).$$

Notice that (6) is the difference between the total penalties of leaf elements and the total rewards of the requirements which are satisfied in the realization. Notice also that, if $E_i$ is a user assertion, then its truth value is forced in every realization by a unit clause inside $\Psi$, s.t. its weight plays no role in finding a minimum-weight realization.
Intuitively, the primary objective of introducing weight into CGMs is to effectively represent the preferences over goals. The optional requirements are given rewards proportional to their level of "preferences". Similarly, some leaf elements can be more expensive (in terms of effort, money, time, ...) of some others, so that we can give them a higher penalty in order to establish our preference.

Notice that one can obtain reasonable compromises between rewards and penalties. For instance, if the user thinks that achieving an optional requirement \( G_1 \) is worth the effort of performing one of tasks \( G_2 \) and \( G_3 \) but not both, then he/she can set the relative penalties and rewards so that \( \max(\text{penalty}(G_2), \text{penalty}(G_3)) \leq \text{reward}(G_1) < \text{penalty}(G_2) + \text{penalty}(G_3) \). Notice also that it is possible to prioritize rewards over penalties, or vice versa. For instance, one may want to find a maximum-reward realization and, if more than one can be found, choose the one of minimum penalty. This can be done by assigning to the rewards of requirements values which are much bigger than those assigned to the penalties of leaf elements. In general, it is possible to assign automatically weights to optional goals by using ad hoc algorithms.

5 Automated Reasoning Functionalities

5.1 Encodings of CGMs and WCGMs

Definition 5 (SMT(\( \mathcal{LRA} \)) Encoding of a CGM). Let \( \mathcal{M} \equiv \langle \mathcal{B}, \mathcal{N}, \mathcal{D}, \Psi \rangle \) be a CGM. The encoding of \( \mathcal{M} \) is the SMT(\( \mathcal{LRA} \)) formula \( \Psi_{\mathcal{M}} \equiv \Psi \land \Psi_E \land \Psi_R \), where:

\[
\Psi_E \equiv \bigwedge_{E \in \text{Roots(\( \mathcal{D} \))} \cup \text{Internals(\( \mathcal{D} \))}} (E \rightarrow (\bigvee_{R_i \in \text{Ref}(E)} R_i)) \tag{8}
\]

\[
\Psi_R \equiv \bigwedge_{(E_1, \ldots, E_n) \subseteq E, \ R \in \mathcal{R}} ((\bigwedge_{i=1}^n E_i \leftrightarrow R) \land (R \rightarrow E)) \tag{9}
\]

The following fact is a straightforward consequence of Definitions 2 and 5.

Proposition 1. Let \( \mathcal{M} \equiv \langle \mathcal{B}, \mathcal{N}, \mathcal{D}, \Psi \rangle \) be a CGM and \( \Psi_{\mathcal{M}} \) its encoding as in Definition 5; let \( \mu \) be \( \mathcal{LRA} \)-interpretation over \( \mathcal{B} \cup \mathcal{N} \). Then \( \mu \) is a realization of \( \mathcal{M} \) if and only if \( \mu \models \Psi_{\mathcal{M}} \).

Definition 6 (OMT(\( \mathcal{LRA} \)) Encoding of a WCGM). Let \( \mathcal{M} \equiv \langle \mathcal{B}, \mathcal{N}, \mathcal{D}, \Psi, \mathcal{M} \rangle \) be a WCGM. The OMT(\( \mathcal{LRA} \)) encoding of \( \mathcal{M} \) is given by the pair \( \langle \Psi_{(\mathcal{M}, \text{cost})}, \text{cost} \rangle \), where:

\[
\Psi_{(\mathcal{M}, \text{cost})} \equiv \Psi_{\mathcal{M}} \land (\text{cost} = \sum_{E_i \in \mathcal{E}, \mathcal{W}(E) \neq 0} t_i) \land \bigwedge_{E_i \in \mathcal{E}, \mathcal{W}(E) \neq 0} ((E_i \rightarrow (t_i = \mathcal{W}(E_i))) \land (\neg E_i \rightarrow (t_i = 0))) \tag{10}
\]

\(3\) E.g., it suffices to set the granularity of award values \( 10^k \) times bigger than that of penalty values, s.t. \( k > \lceil \log_{10}(N \cdot \delta) \rceil \), where \( N \) is the number of leaf elements and \( \delta \) is their maximum penalty value. This can be computed automatically.
where $\Psi_M$ is a SMT($\mathcal{LRA}$) formula as in Definition 5, cost and the $t_i$’s are fresh numerical variables, one for each root or leaf element whose weight is nonzero.

Intuitively, the cost to minimize is given by a sum of the terms $t_i$, one for each root or leaf element $E_i$ with nonzero weight, which are set to their respective weight $W(E_i)$ if $E_i$ is assigned to true, to zero otherwise.

The following fact is a straightforward consequence of Definitions 4 and 6.

**Proposition 2.** Let $M \equiv (B, N, D, \Psi_{(M,\text{cost})}, \mathcal{M})$ be a WCGM and $(\Psi_{(M,\text{cost})}, \text{cost})$ be its OMT($\mathcal{LRA}$) encoding, as in Definition 6; let $\mu$ be a $\mathcal{LRA}$-interpretation over $B \cup N$. Then $\mu$ is a minimum-cost realization of $M$ if and only if $\mu$ is a solution to the OMT($\mathcal{LRA}$) problem $(\Psi_{(M,\text{cost})}, \text{cost})$.

### 5.2 Automated Reasoning on Constrained Goal Models

Proposition 1 suggests that realizations of a CGM $M$ can be produced by applying SMT solving to the encoding $\Psi_M$. This allows for implementing straightforwardly the following reasoning functionality on CGMs.

**Search/enumereation of CGM realizations.** It is possible to check the (un)realizability of a CGM $M$ under a group of user assertions—or to enumerate one or more of its possible realizations— by invoking a SMT solver on the formulas in Definition 5. Notice that this can be done interactively by marking an unmarking (optional) requirements, tasks and domain assumptions, each time searching for a realization.

Importantly, when a CGM is found un-realizable under group of user’s assertions, it is possible to highlight the subparts of the CGM and the subset of assertions causing the problem. This can be done by asking the SMT solver to identify the unsatisfiable core of the input formula —i.e. the subset of sub-formulas which caused the inconsistency, see e.g. [4]— and mapping them back into the corresponding information.

Proposition 2 suggests that the search for minimum-weight realizations for a WCGM $M$ can be encoded into an OMT problem. This allows for implementing straightforwardly also the following reasoning functionalities on WCGMs.

**Search/enumereate minimum-weight WCGM realizations.** It is possible to find one (many, all) minimum-weight realization(s) which achieve the desired requirements while minimizing the total penalty of leaf elements. This is achieved by marking the requirements to be assigned to true in the user’s assertions, setting $W(G_i) = 0$ for all other requirements $G_i$ (if any), and setting the penalties of all leaf elements.

**Search/enumereate maximum-reward WCGM realizations.** When it is not possible to achieve all desired requirements, it is possible to find one (many, all) maximum-reward realization(s) which maximize the total reward while denying the (un)desired leaf elements. This is achieved by choosing the leaf elements to be assigned to false in the user’s assertions (if any), setting $W(E_i) = 0$ for all other leaf elements $E_i$, and setting the rewards of all interesting requirements.
Intermediate situations between the two latter cases can be obtained by setting rewards to interesting requirements and penalties to the priced leaf elements and setting zero-weight for all other elements, as described in §4.2. The realizations returned by OMT will be a compromise between the two needs.

6 Implementation

CGM-Tool is a computer-aided support for modeling and reasoning on constrained goal models. Technically, CGM-Tool is a standalone application written in Java and its core is based on Eclipse RCP engine. Under the hood, it encodes constraint goal models in OptiMathSAT \(^4\) [23] to support reasoning on goal models. It is freely distributed as a compressed archive file for multiple platforms \(^5\). CGM-Tool supports:

- **Specification of projects**: constrained goal models\(^6\) are created within the scope of project containers. A project contains a set of models that can be used to generate reasoning sessions with OptiMathSAT (i.e., scenarios);
- **Diagrammatic modeling**: the tool enables the creation (drawing) of models in terms of diagrams; furthermore it enhances the modeling process by providing real-

\(^4\) http://www.optimathsat.disi.unitn.it
\(^5\) https://trinity.disi.unitn.it/azura/cgm/
\(^6\) The tool currently allows users to define only propositional prerequisite constraints \(\\{\phi^+_G, \phi^-_G\}\) like (1); a parser for SMT(\(\mathcal{LRA}\)) prerequisite constraints like (5) is under development.
time check for refinement cycles and by reporting invalid refinement, contribution and binding links;

- **Consistency/well-formedness check**: CGM-Tool allows for the creation of diagrams conform with the semantics of the modeling language by providing the ability to run consistency analysis on the model;

- **Reasoning**: CGM-Tool provides automated reasoning functionalities by encoding the model into SMT2 formula. Results of OptiMathSAT are shown directly on the model as well as in a tabular form.

CGM-Tool extends the STS-Tool [19] as an RCP application by using the major frameworks shown in Figure 2: 

- **Rich Client Platform (RCP)**, a platform for building rich client applications, made up of a collection of low level frameworks such as OSGi, SWT, JFace and Equinox, which provide us a workbench where to get things like menus, editors and views;

- **Graphical Editing Framework (GEF)**, a framework used to create graphical editors for graphical modeling tools (e.g., tool palette and figures which can be used to graphically represent the underlying data model concepts);

- **Eclipse Modeling Framework (EMF)**, a modeling framework and a code generation facility for building tools and applications based on a structured data model.

7 **Related work**

We next offer a quick overview of, and comparison with some the state of the art goal-oriented modeling languages. [13] provide better and deeper comparisons on requirements modeling languages and the goal-oriented approach, including their advantages and limitations.

**KAOS.** KAOS [6] supports a rich ontology for requirements that goes well beyond goals, as well as an Linear Temporal Logic (LTL)-grounded formal language for constraints. This language is coupled with a concrete methodology for solving requirements problems. KAOS come with a number of analysis techniques, including obstacle, inconsistency and probabilistic goal analysis. However, unlike our proposal, KAOS does not support optional requirements and preferences, nor does it exploit SAT/SMT solver technologies for scalability.

**i∗ and Tropos.** i∗ [25] focuses on modelling actors for a requirements engineering problem (stakeholders, users, analysts, etc.), their goals and inter-dependencies. i∗ provides two complementary views of requirements: the Actor Strategic Dependency Model (SD model) and the Actor Strategic Rationale Model (SR model). Typically, SD models are used to analyze alternative networks of delegations among actors for fulfilling stakeholder goals, whilst SR models are used to explore alternative ways of fulfilling a single actor’s goals. i∗ is expressively lightweight, intended for early stages of requirements analysis, and did not support formal reasoning until recent thesis work by Horkoff [11]. Tropos [3] is a requirements-driven agent-oriented software development methodology founded on i∗. Goal models can be formalized in Tropos by using Formal Tropos [8], an extension of i∗ that supports LTL for formalizing constraints. Alternatively, qualitative goal models can be used, briefly reviewed in §2. The main deficiencies of this
work relative to our proposal is that Formal Tropos is expressive but not scalable, while qualitative goal models are variants of propositional logic, hence not too expressive.

**Techne and Liaskos.** Techne [13] is a recent proposal for a class of goal-modeling languages that supports optional goals and preferences, but is strictly propositional and has not been studied at all for reasoning and tractability. Liaskos [15, 14] has proposed extensions to qualitative goal models to support optional goals and preferences, as well as decision-theoretic concepts such as utility. This proposal is comparable to our proposal in this paper, but uses AI reasoners for reasoning (AI planners and GOLOG) and, consequently, does not scale very well relative to our proposal.

**Feature Models.** Feature models [5] share many similarities with goal models: they are hierarchically structured, with AND/OR refinements, constraints and attributes. However, each feature represents a bundle of functionality or quality and as such, feature models are models of software specification, not requirements. Moreover, reasoning techniques for feature models are limited relative to their goal model cousins.

### 8 Conclusions and Future Work

We have proposed a goal-based modeling language for requirements that supports the representation of optional requirements, preferences, constraints and more. Moreover, we have exploited automated reasoning solvers in order to develop a prototype tool that scales well as goal models grow to realistic sizes for real world requirements problems.

We are currently implementing a parser for full SMT($LRA$) prerequisite constraints \( \{ \phi^+_G, \phi^-_G \} \) in addition to propositional ones. After that, our next step is to implement syntactic-sugaring constraints such as \texttt{Requires} \((P_1, P_2)\), \texttt{AtMostN} \((N, P_1, \ldots, P_n)\), etc. to facilitate the modeling of some of the standard and intuitive constraints among assumptions, goals, and refinement labels without the need to define complex and less-than-intuitive propositional formulas.

We also plan to formalize the evolutionary versions of CGMs, which can handle the evolution requirements problems [7]. For such problems, the goal models can be changed even after a solution has been implemented. Thus, given a modified CGM and its previous solution, we have to find a new solution that minimize the effort of applying changes. Our idea is to define a notion of added penalty w.r.t. a given realization of the original CGM and optimize the added penalty while searching for new realizations of the evolved/modified CGM.

### References