Multi-Objective 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 Satisfiability and Optimization Modulo Theories (SMT/OMT), and we propose and formalize: (i) an extended notion of goal model, namely *Constrained Goal Models (CGMs)*, which makes explicit the notion of *goal refinement*, allows for associating *numerical attributes* (e.g., penalty/reward, cost, workload) to goals and refinements for defining *constraints* and multiple *objective functions* over goals, refinements and their numerical attributes; (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 the given objective functions. We have implemented these modeling and reasoning functionalities in a tool, named CGM-Tool, using the OMT 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, for example Artificial Intelligence (AI) planning, agent-based systems, and knowledge management. More recently, software engineering has also been using goals to model requirements for software systems, business objectives for enterprises, and design qualities [19].

Goal-oriented requirements engineering approaches have gained popularity in the last two decades for a number of significant benefits in conceptualizing and analyzing requirements [19]. Goal models provide a broader system engineering perspective compared to the traditional requirements engineering methods, a precise criterion for completeness of the requirements analysis process, 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 [8]. Leading approaches such as KAOS [4] and i^* [20] are limited in expressing stakeholder preferences, but also in supporting scalable reasoning over goal models. More recent approaches, such as Techne [10] and [12], propose expressive extensions to goal models, but still lack scalable reasoning facilities.

As an answer to the need for more expressiveness and more sophisticated reasoning support, we exploit advances in automated reasoning technologies, notably *Satisfiability Modulo Theories (SMT)* [1] and *Optimization Modulo Theories (OMT)* [17], to propose and formalize an extended notion of goal model, namely *Constrained Goal Model (CGM)*. CGMs provide many novelties w.r.t. previous definitions of goal models. In particular, they allow stakeholders for:

(i) defining *goal refinements*, which are explicitly labeled by Boolean propositions and can be interactively/automatically reasoned upon;
(ii) stating *domain assumptions* to represent preconditions to goals;
(iii) adding *Boolean constraints* over goals, domain assumptions and refinements;
(iv) expressing *preferences* over goals and their refinements, by distinguishing between mandatory and optional requirements and by assigning preference *weights* (i.e., penalties/rewards) to goals and domain assumptions;
(v) associating *numerical attributes* (e.g., resources like cost, worktime, room, fuel) to goals and refinements and for defining *constraints* and multiple *objective functions* over goals, refinements and their numerical attributes.

Taking advantage of the formal semantics of CGMs and of the expressiveness and efficiency of current SMT and OMT solvers, we also provide a set of automated reasoning functionalities on CGMs. Particularly, on a given CGM and for any given set of stakeholders' assertions and constraints, our approach allows for:

(a) the automatic check of the realizability of the CGM;
(b) the interactive/automatic search for realizations;
(c) the automatic search for the "best" realizations in terms of penalties/rewards;
(d) the automatic search for the realization(s) which optimize the objective functions defined by the stakeholder (point (v)).

Our approach is implemented as a 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 alternatives scenarios running automated reasoning techniques. CGM-Tool uses the SMT/OMT solver OptiMathSAT [17, 18] 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 overview of our prototype tool based on the presented approach; §7 gives an overview of related work; in §8 we draw conclusions and present future research challenges.

2 Background

Our research baseline consists of our previous work on qualitative goal models and of Satisfiability and Optimization Modulo Theories (SMT and OMT respectively).

Goal Models. Qualitative goal models are introduced in [13], where the concept of goal is used to represent respectively a functional and non-functional 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 [7], goal models are formalized by replacing each proposition g, standing for a goal, by four propositions (FS_g, PS_g, PD_g, FD_g) representing full (and partial) evidence for the satisfaction/denial of g. A traditional implication such as $p \wedge q \rightarrow r$ is then translated into a series of implications connecting these new symbols, including $FS_p \wedge FS_q \rightarrow FS_r$, $PS_p \wedge PS_q \rightarrow PS_r$, as well as $FD_p \rightarrow FD_r$, $FD_q \rightarrow FD_r$, etc. The conflict between goals 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_a 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 $\{FS, PS, FD, PD\}$.

[15] extended the approach further by including axioms for avoiding conflicts of the form $FS_a \wedge FD_a$. 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 (”nice to have”) from mandatory requirements and have no notion of a robust solution, i.e. solution without ”conflict”, where a goal can not be (fully or partially) denied and (respectively, fully or partially) 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 Φ with respect to some decidable theory $\mathcal{T}$ (see [16, 1]). In this paper we focus on the theory of *linear arithmetic over the rationals, $\mathcal{LRA}$*: SMT($\mathcal{LRA}$) is the problem of checking the satisfiability of a formula Φ consisting in atomic propositions $A_1, A_2, ...$ and linear-arithmetic constraints like “$(2.1x_1 - 3.4x_2 + 3.2x_3 \leq 4.2)$”, combined by means of Boolean operators $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$. An *Optimization Modulo Theories over $\mathcal{LRA}$ (OMT($\mathcal{LRA}$))* problem $\langle \Phi, \langle obj_1, ..., obj_k \rangle \rangle$ is the problem of finding solution(s) to an SMT($\mathcal{LRA}$) formula Φ which optimize the rational-valued objective functions $obj_1, ..., obj_k$, either singularly or lexicographically [17, 18]). [1]

Very efficient SMT($\mathcal{LRA}$) and OMT($\mathcal{LRA}$) solvers are available, which combine the power of modern SAT solvers with dedicated linear-programming decision and minimization procedures (see [16, 1, 17, 18]). For instance, in the empirical evaluation reported in [17] our OMT($\mathcal{LRA}$) solver OptiMathSAT was able to handle problems with up to thousands Boolean/rational variables in less than 10 minutes each.

3 Constrained Goal Models

We introduce the main ideas of CGMs and the main functionalities of our CGM-Tool through a meeting scheduling example (Figure 1) where we model the requirements for

[1] A solution optimizes lexicographically $\langle obj_1, ..., obj_k \rangle$ if it optimizes obj_1 and, if more than one such obj_1-optimal solutions exists, it also optimizes obj_2,..., and so on.

a meeting scheduling system, including the functional requirement ScheduleMeeting, as well as non-functional/quality requirements LowCost, FastSchedule, MinimalEffort and GoodQualitySchedule. They are represented as root goals.

Goals, Refinements, and Domain Assumptions. 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_is denote their labels. [2]

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 it.

The main objective of our 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, 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. The subgoals are further refined until they reach the level of domain assumptions and tasks.

User's Assertions. Some requirements can be *optional*, like LowCost, MinimalEffort, FastSchedule, and GoodQualitySchedule (in blue in Figure 1). They are requirements that we would like to fulfill with our solution, provided they do not conflict with other requirements. To this extent, in order to analyze interactively the possible different realizations, one can interactively mark [or unmark] requirements as satisfied, thus making them mandatory (if unmarked, they are optional). Similarly, one can interactively mark/unmark (effortful) tasks as denied, or mark/unmark some domain assumption as satisfied or denied. More generally, one can mark as satisfied or denied every goal or domain assumption. We call these marks *user's assertions*.

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 *Boolean or SMT*($\mathcal{LRA}$) *formulas*.

Relation Edges. We have three kinds of relation edges. *Contribution edges* "$E_i \xrightarrow{++} E_j$" 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* "$E_i \xleftrightarrow{--} E_j$" between

[2] The label of a refinement can be omitted when there is no need to refer to it explicitly.

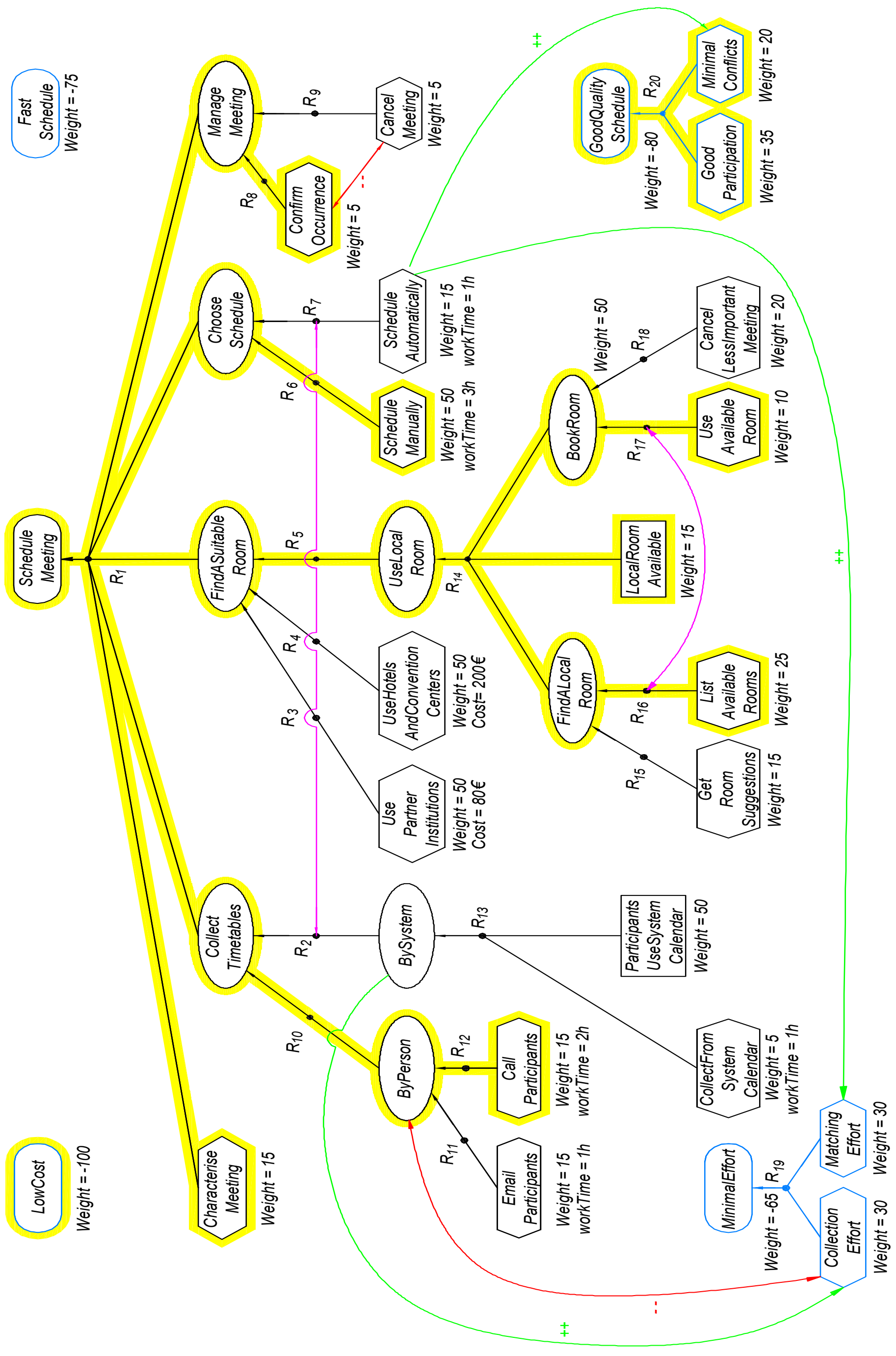

Fig. 1. An example of a CGM

elements (in red), like "ConfirmOccurrence $\overset{--}{\longleftrightarrow}$ CancelMeeting", mean that E_i and E_j cannot be both satisfied. *Refinement bindings* "$R_i \longleftrightarrow R_j$" between two refinements (in purple), like "$R_2 \longleftrightarrow R_7$", 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.

Boolean Constraints. 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 ϕ_G^+ [resp. ϕ_G^-] must be satisfied when G is satisfied [resp. denied]. (The same holds for each requirement R.)

For example, to require that, as a prerequisite for FastSchedule, ScheduleManually and ByPerson cannot be both satisfied, one can add a constraint to the positive prerequisite formula of FastSchedule:

$$\phi^+_{\mathsf{FastSchedule}} = \ ... \wedge \neg(\mathsf{ScheduleManually} \wedge \mathsf{ByPerson}), \tag{1}$$

or, equivalently, add globally to the CGM the following Boolean formula:

$$\mathsf{FastSchedule} \rightarrow \neg(\mathsf{ScheduleManually} \wedge \mathsf{ByPerson}). \tag{2}$$

Notice that there is no way we can express (1) or (2) with the relation edges above.

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 under given user's assertions (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 and user-defined constraints. 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 with all Boolean and SMT($\mathcal{LRA}$) constraints (see later).

Apart from the mandatory requirement, the realization in Figure 1 allows to achieve also the optional requirements LowCost, GoodQualitySchedule, but not FastSchedule and MinimalEffort; it requires accomplishing the tasks CharacteriseMeeting, CallParticipants, ListAvailableRooms, UseAvailableRoom, ScheduleManually, ConfirmOccurrence, GoodParticipation, MinimalConflicts, and requires the domain assumption LocalRoomAvailable.

Preferences via Penalties/Rewards. In general, a CGM has many possible realizations. To distinguish among them, stakeholders may want to capture *preferences* on the requirements to achieve and on the tasks to accomplish.

In order to state preferences directly, stakeholders can assign *positive weights* (*penalties*) to tasks and *negative weights* (*rewards*) to (non-mandatory) requirements (the numbers "Weight $= \ldots$" in Figure 1). This implies that requirements [resp. tasks] with higher rewards [resp. smaller penalties] are preferable. If so, thank to a call to an OMT solver, the system returns a realization which minimizes its global weight, that is, the total difference between the penalties and rewards.

For instance, the minimal-weight realization of the example CGM, achieves all the optional requirements except MinimalEffort, with a total weight of -65. Such realization requires accomplishing the tasks CharacteriseMeeting, EmailParticipants, UsePartnerInstitution, ScheduleManually, ConfirmOccurrence, GoodParticipation, and MinimalConflicts, and requires no domain assumption. (This was found automatically by our CGM-Tool of §6 in 0.008 seconds on an Apple MacBook Air laptop.)

In general, stakeholders might not always be at ease in assigning numerical values to state their preferences. This can be coped with in two possible ways. One simple way is to use more coarse-grained and user-friendly rankings (e.g. "critical", "important", "moderately important", "marginal") which are automatically translated into numbers. A more radical way is to allow stakeholders to set pairwise preferences or equivalences between tasks/requirements, and then to use an ad-hoc algorithm like *Analytic Hierarchy Process (AHP)* [11] to generate the numerical values automatically.

Numerical Attributes. In addition to Boolean constraints and penalties/rewards, it is also possible to use numerical variables to express different numerical attributes of elements (such as cost, worktime, space, fuel, etc.) For example, suppose we estimate that UsePartnerInstitutions costs 80€, whereas UseHotelsAndConventionCenters costs 200€. One can express these facts straightforwardly and intuitively by adding a global numerical variable cost to the model; then the system automatically generates a numerical attribute cost_E for each element E, whose value is set to the default value 0, and the default global constraint $\mathsf{cost} = \sum_{E \in \mathcal{E}} \mathsf{cost}_\mathsf{E}$; [3] then, for some element E of interest, one can set the value for cost_E in case E is satisfied (or denied): e.g., $\mathsf{cost}_{\mathsf{UsePartnerInstitutions}} := 80$ and $\mathsf{cost}_{\mathsf{UseHotelsAndConventionCenters}} := 200$. By doing so, the following prerequisite SMT($\mathcal{LRA}$) constraints are automatically added:

$$\begin{aligned} \phi^+_{\mathsf{UsePartnerInstitutions}} &= \ \ldots \wedge (\mathsf{cost}_{\mathsf{UsePartnerInstitutions}} = 80) \\ \phi^+_{\mathsf{UseHotelsAndConventionCenters}} &= \ \ldots \wedge (\mathsf{cost}_{\mathsf{UseHotelsAndConventionCenters}} = 200). \end{aligned} \tag{3}$$

and the corresponding negative prerequisite constraints (like $\phi^-_{\mathsf{UsePartnerInstitutions}} = \ldots \wedge (\mathsf{cost}_{\mathsf{UsePartnerInstitutions}} = 0)$), are also automatically added (if not specified otherwise).

SMT($\mathcal{LRA}$) Constraints. Suppose that, in order to achieve the optional requirements LowCost, we need to have a total cost smaller than 100€. This can be expressed by adding the prerequisite constraint: $\phi^+_{\mathsf{LowCost}} = \ldots \wedge (\mathsf{cost} < 100)$. Hence, e.g., due to (3), every realization that the tool generates automatically which satisfies LowCost will not involve the task UseHotelsAndConventionCenters.

Similarly to cost, one can introduce, e.g., another global numerical attribute workTime to reason on working time, and estimate, e.g., that the total working time for ScheduleManually, ScheduleAutomatically, EmailParticipants, CallParticipants, CollectFromSystemCalendar are 3, 1, 1, 2, 1 hour(s) respectively, and state that the optional requirement FastSchedule must require a global time smaller than 5 hours.

[3] Notice that this is only a *default* global constraint: the user is free to manipulate it.

Notice that one can build combinations of numerical attributes. For instance, if labor costs $35€/hour$, then one can redefine cost as ($\mathsf{cost} = \sum_{E\in\mathcal{E}} \mathsf{cost_E} + 35\cdot\mathsf{workTime}$), or introduce a new global variable totalCost as ($\mathsf{totalCost} = \mathsf{cost} + 35\cdot\mathsf{workTime}$).

Preferences via Multiple Objectives. Stakeholders may define rational-valued *objectives* $obj_1, ..., obj_k$ to optimize (i.e. maximize or minimize) as functions of Boolean and numerical variables —e.g., cost, workTime, totalCost can be suitable objectives, Weight is considered a pre-defined objective— and ask the tool to automatically generate realization(s) which optimize one objective, or some combination of more objectives (like totalCost), or which optimizes lexicographically an ordered list of objectives.

For example, the previously-mentioned optimum-weight realization of Figure 1 is such that $\mathsf{Weight} = -65$, $\mathsf{workTime} = 4$ and $\mathsf{cost} = 80$. Our CGM has many different minimum-weight realizations s.t. $\mathsf{Weight} = -65$, with different values of cost and workTime. Among them, it is possible to search, e.g., for the realizations with minimum workTime, and among these for those with minimum cost, by setting lexicographic minimization with order $\langle\mathsf{Weight}, \mathsf{workTime}, \mathsf{cost}\rangle$. This results into one realization with $\mathsf{Weight} = -65$, $\mathsf{workTime} = 2$ and $\mathsf{cost} = 0$ achieving all the optional requirements, requiring the tasks: CharacteriseMeeting, CollectFromSystemCalendar, GetRoomSuggestions, CancelLessImportantMeeting, ScheduleAutomatically, ConfirmOccurence, GoodParticipation, MinimalConflicts, CollectionEffort, MatchingEffort, and requiring the domain assumptions: ParticipantsUseSystemCalendar, LocalRoomAvailable. (This was found automatically by our CGM-Tool of §6 in 0.016 seconds on an Apple MacBook Air laptop.)

4 Abstract Syntax and Semantics

We call a *goal graph* $\mathcal{D}$ 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 outgoing 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* $\mathcal{M} \overset{\text{def}}{=} \langle\mathcal{B}, \mathcal{N}, \mathcal{D}, \Psi\rangle$*, s.t.*

– $\mathcal{B} \overset{\text{def}}{=} \mathcal{G} \cup \mathcal{R} \cup \mathcal{A}$ *is a set of atomic propositions, where* $\mathcal{G} \overset{\text{def}}{=} \{G_1, ..., G_N\}$*,* $\mathcal{R} \overset{\text{def}}{=} \{R_1, ..., R_K\}$*,* $\mathcal{A} \overset{\text{def}}{=} \{A_1, ..., A_M\}$ *are respectively sets of goal, refinement and domain-assumption labels. We denote with* $\mathcal{E}$ *the set of element labels:* $\mathcal{E} \overset{\text{def}}{=} \mathcal{G} \cup \mathcal{A}$*;*

- *$\mathcal{N}$ is a set of numerical variables in the rationals;*
- *$\mathcal{D}$ 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 $\mathcal{A}$;*
- *Ψ is a SMT($\mathcal{LRA}$) formula on $\mathcal{B}$ 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 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 tasks, under certain domain assumptions.

In general, the user might not be at ease in defining a possibly-complex *global* SMT($\mathcal{LRA}$) formula Ψ 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 constructs allowing the user to encode *graphically* and *locally* desired constraints of frequent usage: *relation edges*, *prerequisite constraints* $\{\phi_G^+, \phi_G^-\}$ and $\{\phi_R^+, \phi_R^-\}$ and *user's assertions*. Each is automatically converted into a simple SMT($\mathcal{LRA}$) formula as follows, and then conjoined to Ψ.

Element-contribution edges, $E_1 \xrightarrow{++} E_2$, are encoded into the formula $(E_1 \to E_2)$.
Element-conflict edges, $E_1 \xleftrightarrow{--} E_2$, are encoded into the formula $\neg(E_1 \wedge E_2)$.
Refinement-binding edges, $R_1 \longleftrightarrow R_2$, s.t. E_1, E_2 are the target elements of R_1, R_2 respectively, are encoded into the formula $(E_1 \wedge E_2) \to (R_1 \leftrightarrow R_2)$.
Prerequisite constraints, $\{\phi_G^+, \phi_G^-\}$ [resp. $\{\phi_R^+, \phi_R^-\}$] are encoded into the formulas $(G \to \phi_G^+)$ and $(\neg G \to \phi_G^-)$ [resp. $(R \to \phi_R^+)$ and $(\neg R \to \phi_R^-)$].
User's assertions, $E_i := \top$ and $E_j := \bot$, are encoded into the formulas (E_i), $(\neg E_j)$.

Notice that, unlike refinements, relation edges are allowed to create loops, possibly involving refinements.

The semantics of CGMs is formally defined as follows.

Definition 2 (Realization of a CGM). *Let $\mathcal{M} \overset{\text{def}}{=} \langle \mathcal{B}, \mathcal{N}, \mathcal{D}, \Psi \rangle$ be a CGM. A* realization *of $\mathcal{M}$ is a $\mathcal{LRA}$-interpretation μ over $\mathcal{B} \cup \mathcal{N}$ such that:* [4]

(a) $\mu \models ((\bigwedge_{i=1}^{n} E_i) \leftrightarrow R) \wedge (R \to E)$ *for each refinement* $(E_1, \dots, E_n) \xrightarrow{R} E$;
(b) $\mu \models \left(E \to (\bigvee_{R_i \in \mathrm{Ref(E)}} R_i)\right)$, *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.*

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 μ). If an element E is not a

[4] A $\mathcal{LRA}$-interpretation μ is a function which assigns truth values to Boolean atoms and rational values to numerical variables. "$\mu \models \Phi$" means that μ makes the formula Φ evaluate to true.

leaf, then it can be satisfied only by satisfying the set of source elements $E_1, \ldots, E_n$ of one of its refinements $\big(E_1, \ldots, E_n\big) \xrightarrow{R} E$. If μ 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 Ψ: every realization μ 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 user's assertions and with the local pre-requisite constraints $\{\phi_E^+, \phi_E^-\}$ and $\{\phi_R^+, \phi_R^-\}$, because the corresponding formulas are conjuncts of Ψ. Thus Ψ contains also the global and local SMT($\mathcal{LRA}$) constraints over global and local numerical attributes (e.g. $\mathsf{LowCost} \rightarrow (\mathsf{cost} \leq 100)$, $\mathsf{UsePartnerInstitutions} \rightarrow (\mathsf{cost}_{\mathsf{UsePartnerInstitutions}} = 80)$, and the definitions of objectives (e.g., $(\mathsf{totalCost} = \sum_{E \in \mathcal{E}} \mathsf{cost_E} + 35 \cdot \mathsf{workTime})$).

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

5 Automated Reasoning Functionalities

Definition 3 (SMT($\mathcal{LRA}$) Encoding of a CGM). *Let $\mathcal{M} \overset{\text{def}}{=} \langle \mathcal{B}, \mathcal{N}, \mathcal{D}, \Psi \rangle$ be a CGM. The* SMT($\mathcal{LRA}$) encoding *of $\mathcal{M}$ is the formula $\Psi_{\mathcal{M}} \overset{\text{def}}{=} \Psi \wedge \Psi_{\mathcal{E}} \wedge \Psi_{\mathcal{R}}$, where:*

$$\Psi_{\mathcal{E}} \overset{\text{def}}{=} \bigwedge_{E \in \mathsf{Roots}(\mathcal{D}) \cup \mathsf{Internals}(\mathcal{D})} \big(E \rightarrow (\bigvee_{R_i \in \mathsf{Refinements(E)}} R_i)\big) \quad (4)$$

$$\Psi_{\mathcal{R}} \overset{\text{def}}{=} \bigwedge_{\big(E_1, \ldots, E_n\big) \xrightarrow{R} E,\ R \in \mathcal{R}} \big((\bigwedge_{i=1}^{n} E_i \leftrightarrow R) \wedge (R \rightarrow E)\big), \quad (5)$$

$\mathsf{Roots}(\mathcal{D})$ *and* $\mathsf{Internals}(\mathcal{D})$ *being the root and internal elements of $\mathcal{D}$ respectively.*

The following facts are straightforward consequences of Definitions 2 and 3 and of the definition and OMT($\mathcal{LRA}$).

Proposition 1. *Let $\mathcal{M} \overset{\text{def}}{=} \langle \mathcal{B}, \mathcal{N}, \mathcal{D}, \Psi \rangle$ be a CGM, $\Psi_{\mathcal{M}}$ its SMT($\mathcal{LRA}$) encoding and μ a $\mathcal{LRA}$-interpretation over $\mathcal{B} \cup \mathcal{N}$. Then μ is a realization of $\mathcal{M}$ if and only if $\mu \models \Psi_{\mathcal{M}}$.*

Proposition 2. *Let $\mathcal{M}$ and $\Psi_{\mathcal{M}}$ be as in Proposition 1, and let μ be a realization of $\mathcal{M}$. Let $\{obj_1, ..., obj_k\}$ be $\mathcal{LRA}$-terms occurring in Ψ. Then we have that:*

(i) for every i in $1, ..., k$, μ minimizes [resp. maximizes] obj_i if and only if μ is a solution of the OMT($\mathcal{LRA}$) minimization [resp. maximization] problem $\langle \Psi_{\mathcal{M}}, \langle obj_i \rangle \rangle$;

(ii) μ lexicographically minimizes [resp. maximizes] $\langle obj_1, ..., obj_k \rangle$ if and only if μ is a solution of the OMT($\mathcal{LRA}$) lexicographic minimization [resp. maximization] problem $\langle \Psi_{\mathcal{M}}, \langle obj_1, ..., obj_k \rangle \rangle$.

Propositions 1 and 2 suggest that realizations of a CGM $\mathcal{M}$ can be produced by applying SMT($\mathcal{LRA}$) solving to the encoding $\Psi_{\mathcal{M}}$, and that *optimal* realizations can be produced by applying OMT($\mathcal{LRA}$) to $\Psi_{\mathcal{M}}$ and a list of defined objectives $obj_1, ..., obj_k$. This allowed us to implement straightforwardly the following reasoning functionalities on CGMs by interfacing with a SMT/OMT tool.

Search/enumerate realizations. Stakeholders can automatically check the realizability of a CGM $\mathcal{M}$ –or to enumerate one or more of its possible realizations– under a group of user assertions and of user-defined Boolean and SMT($\mathcal{LRA}$) constraints; the tool performs this task by invoking an SMT solver on the formula $\Psi_{\mathcal{M}}$ of Definition 3.

Search/enumerate minimum-penalty/maximum reward realizations. Stakeholders can assert the desired requirements and set the penalties of the tasks; then the tool finds automatically realizations achieving the former while minimizing the latter, by invoking the OMT solver on $\Psi_{\mathcal{M}}$ with the pre-defined Weight objective. The vice versa is obtained by negating undesired tasks and setting the rewards of optional requirements. Every intermediate situations can be also be obtained.

Search/enumerate optimal realizations wrt. user-defined objectives. Stakeholders can define their own objective functions $obj_1, ..., obj_k$ over goals, refinements and their numerical attributes; then the tool finds automatically realizations optimizing them, either independently or lexicographically, by invoking the OMT solver on $\Psi_{\mathcal{M}}$ and $obj_1, ..., obj_k$.

Notice that all these actions can be performed *interactively* by marking an unmarking (optional) requirements, tasks and domain assumptions, each time searching for a suitable or optimal realization.

6 Implementation

CGM-Tool provides support modeling and reasoning on CGMs. 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 [5] [17] to support reasoning on goal models. It is freely distributed as a compressed archive file for multiple platforms [6]. CGM-Tool supports:

Specification of projects: CGMs are created within the scope of project containers. A project contains a set of CGMs that can be used to generate reasoning sessions with OptiMathSAT (i.e., scenarios);

Diagrammatic modeling: the tool enables the creation (drawing) of CGMs in terms of diagrams; furthermore it enhances the modeling process by providing real-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;

Automated Reasoning: CGM-Tool provides the automated reasoning functionalities of §5 by encoding the model into an SMT formula. The results of OptiMathSAT are shown directly on the model as well as in a tabular form.

[5] `http://optimathsat.disi.unitn.it`

[6] `http://www.cgm-tool.eu/`

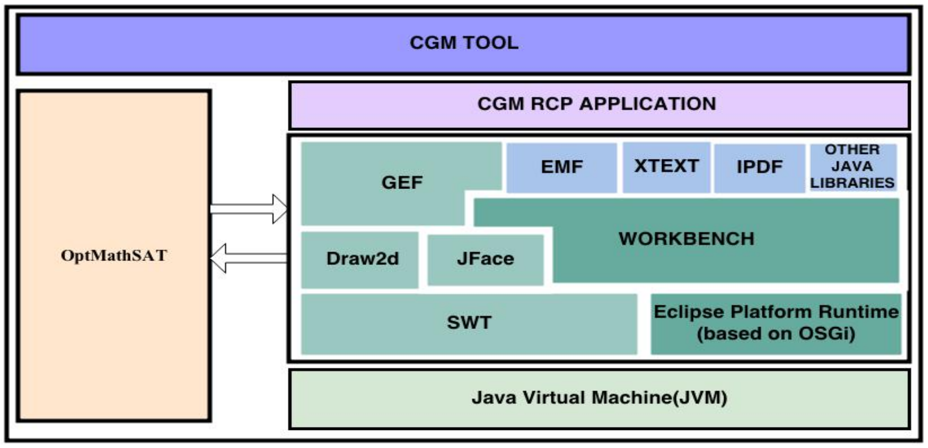

Fig. 2. CGM-Tool Component view

As mentioned in §2, OptiMathSAT successfully handles problems with up to thousands Boolean/rational variables [17]; this fact, combined with the negligible CPU times reported in §3, provides a strong support for the scalability of our approach.

CGM-Tool extends the STS-Tool [14] 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 Equnix, 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. [10] provide better and deeper comparisons on requirements modeling languages and the goal-oriented approach, including their advantages and limitations.

KAOS. KAOS [4] 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 supports 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^* [20] 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 [9]. Tropos [2] is a requirements-driven agent-oriented software development methodology founded on i^*. Goal models can be formalized in Tropos by using Formal Tropos [6], 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 [10] 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 [12, 11] 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 [3] 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 the encoding of syntactic-sugaring constraints, such as "$\mathsf{AtMostN}\,(N, P_1, \ldots, P_n)$", 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 *evolutionary* versions of CGMs, which can handle evolution requirements problems [5]. 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. Again, we plan to address this problem via OMT encodings.

References

1. C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability Modulo Theories. In *Handbook of Satisfiability*, chapter 26, pages 825–885. IOS Press, 2009.
2. J. Castro, M. Kolp, and J. Mylopoulos. Towards requirements-driven information systems engineering: The tropos project. *Inf. Syst.*, 27(6):365–389, Sept. 2002.
3. A. Classen, Q. Boucher, and P. Heymans. A text-based approach to feature modelling: Syntax and semantics of TVL. *Sci. Comput. Program.*, 76(12):1130–1143, 2011.
4. A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50, Apr. 1993.
5. N. A. Ernst, A. Borgida, and I. Jureta. Finding incremental solutions for evolving requirements. In *RE*, pages 15–24. IEEE, 2011.
6. A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, and P. Traverso. Specifying and analyzing early requirements in tropos. *Requir. Eng.*, 9(2):132–150, May 2004.
7. P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Formal reasoning techniques for goal models. *JOURNAL OF DATA SEMANTICS*, 1:1–20, 2004.
8. J. Horkoff and E. S. K. Yu. Comparison and evaluation of goal-oriented satisfaction analysis techniques. *Requir. Eng.*, 18(3):199–222, 2013.
9. J. M. Horkoff. *Iterative, Interactive Analysis of Agent-goal Models for Early Requirements Engineering*. PhD thesis, 2012. AAINR97565.
10. I. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos. Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In *RE*, pages 115–124. IEEE Computer Society, 2010.
11. S. Liaskos. On eliciting contribution measures in goal models. In *Proceedings of the 2012 IEEE 20th International Requirements Engineering Conference (RE)*, RE '12, pages 221–230. IEEE Computer Society, 2012.
12. S. Liaskos, S. A. McIlraith, S. Sohrabi, and J. Mylopoulos. Integrating preferences into goal models for requirements engineering. In *RE*, pages 135–144. IEEE Computer Society, 2010.
13. J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Trans. Softw. Eng.*, 18(6):483–497, June 1992.
14. E. Paja, F. Dalpiaz, M. Poggianella, P. Roberti, and P. Giorgini. STS-Tool: socio-technical security requirements through social commitments. In *Proceedings of the 20th IEEE International Conference on Requirements Engineering*, pages 331–332, 2012.
15. R.Sebastiani, P. Giorgini, and J. Mylopoulos. Simple and Minimum-Cost Satisfiability for Goal Models. In *Proc. 16th International Conference on Advanced Information Systems Engineering - CAISE'04*, LNCS, Riga, Latvia, May 2004. Springer.
16. R. Sebastiani. Lazy Satisfiability Modulo Theories. *Journal on Satisfiability, Boolean Modeling and Computation, JSAT*, 3(3-4):141–224, 2007.
17. R. Sebastiani and S. Tomasi. Optimization Modulo Theories with Linear Rational Costs. *ACM Transactions on Computational Logics*, 16(2), 2015. To appear.
18. R. Sebastiani and P. Trentin. Pushing the Envelope of Optimization Modulo Theories with Linear-Arithmetic Cost Functions. In *Proc. Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'15*, volume 9035 of *LNCS*. Springer, 2015.
19. A. Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, RE '01, pages 249–. IEEE Computer Society, 2001.
20. E. S. K. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *RE '97: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97)*, page 226. IEEE Computer Society, 1997.