Towards a Declarative, Constraint-Oriented Semantics with a Generic Evaluation Algorithm for GRL

Hao Luo and Daniel Amyot

School of Computer Science and Electrical Engineering, University of Ottawa, Ottawa, Canada
haoluo.cn@gmail.com, damyot@eecs.uottawa.ca

Abstract. Goal models described with the Goal-oriented Requirement Language (GRL) are amenable to various kinds of analyses, including quantitative and qualitative propagations of satisfaction values. However, most approaches use bottom-up evaluations involving operational semantics that can only answer "what if" questions. This paper introduces a new declarative semantics for GRL based on a constraint-oriented interpretation of goal models. This semantics enables constraint solvers to evaluate and optimize goal models in a way that is more generic than bottom-up and top-down propagation techniques, hence enabling other questions to be answered. A prototype that combines the jUCMNav modeling tool and the JaCoP constraint solver to support quantitative evaluations is used to demonstrate the feasibility and potential of this new approach.

Keywords. Constraint-oriented goal evaluation, Goal-oriented Requirement Language, jUCMNav, semantics, User Requirements Notation.

1 Introduction

The User Requirements Notation (URN) is a language that combines scenario modeling (with Use Case Maps – UCM) and goal modeling (with the Goal-oriented Requirement Language – GRL) to support requirements engineering activities, especially for reactive systems and business processes. This standard language is defined with a metamodel, a concrete graphical syntax, and an XML-based interchange format [5].

In GRL, a model is composed of *intentional elements* (i.e., goals, softgoals, tasks, resources, and beliefs) connected by *links* (decomposition, contribution, and dependencies) and potentially allocated to *actors*. The URN standard includes a set of guidelines and requirements describing how to analyze GRL models based on a set of initial satisfaction values given to some of the intentional elements (i.e., an *evaluation strategy*), which are then propagated to the other intentional elements through the links connecting them. *Satisfactions* in GRL can be qualitative (satisfied, denied, etc.) or quantitative (integer in [–100..100]). Similarly, contributions can have a qualitative *weight* (make, break, etc.) or a quantitative one ([–100..100]). One particularity of GRL (which does not exist in *i**) is that intentional elements in an actor can have an *importance* factor that, when combined to satisfaction levels, helps measure the overall satisfaction of an actor (again, these measures can be qualitative or quantitative).

Although the standard does not impose specific propagation algorithms to evaluate a goal model against a given evaluation strategy, several algorithms (fully quantitative, fully qualitative, and hybrid) are suggested and are further explored in [1]. These algorithms are all automated, i.e., no interactivity is required to solve conflicts, and they are defined based on an operational semantics for GRL. However, these algorithms only support bottom-up propagation, which means that they can only answer "what-if" types of analysis questions. An evaluation strategy hence typically initializes some of the leaves in the goal graph, and the values are then propagated to the intentional elements higher in that graph.

Bottom-up analysis in limitative in practice; it is akin to testing in terms of analytical power, i.e., one often uses many strategies to find a good global trade-off or find an optimal satisfaction value for a given goal or actor (usually without any guarantee). There is a need to support top-down and inside-out analysis techniques for GRL in order to find solutions to more interesting questions such as "is there a way to reach this satisfaction level for this top-level goal?", or more generically "what is the maximum satisfaction of this goal given these constraints on other goals?".

This paper presents a new semantics and an algorithm for GRL model analysis that will enable modelers to answer all these types of questions. Tool support is also provided to demonstrate the feasibility of the approach.

2 Objectives of the Research

This research aims to support a generic and automatic propagation algorithm for GRL models that can support bottom-up, top-down, and inside-out analysis, as well as optimizations in the presence of constraints (i.e., intentional elements initialized anywhere in the model). To enable such an algorithm, we need to move away from the operational semantics often associated with GRL to a declarative semantics amenable to solving in the presence of constraints. Our approach is as follows: i) provide a declarative semantics for GRL (in this paper, we limit ourselves to quantitative evaluations only), ii) define a transformation of this semantics in terms of a constraint-oriented language for which automated solvers exist, and iii) prototype the approach.

The prototype algorithm is integrated to the jUCMNav tool as one of its GRL evaluation algorithms. jUCMNav is a free Eclipse-based URN modeling and analysis tool that already supports qualitative, quantitative, and hybrid evaluation mechanisms for GRL, but that only use bottom-up propagation [7]. jUCMNav offers an open architecture that makes it easy to add new GRL evaluation algorithms. The constraint-oriented technology we selected is JaCoP [6], a mature library that provides many modular constraint-solving and search mechanisms. JaCoP was selected because it is a Java open source project (like jUCMNav) and because it complies to the JSR-331 API for constraint programming, but others (e.g., Choco) could be used as well.

3 Scientific Contributions

This section briefly presents the three main contributions of our work.

3.1 Declarative Semantics for GRL

The proposed declarative semantics for GRL is expressed in mathematical terms. Intuitively, the syntactic domain is composed of intentional elements (with their types and importance factors), decomposition links (and their types: OR or AND), contribution links (and their weights), dependency links, actors, and a strategy, with the semantics and well-formedness rules expressed in the URN standard (e.g., an intentional element has only one type of decomposition). The semantic domain is one where each syntactically correct GRL diagram is interpreted as an evaluated set of intentional elements and actors, where evaluations are integers between –100 (denied) and 100 (satisfied), inclusively, as required by the URN standard [5]. The semantic function maps the syntactic constructs to evaluations in the semantic domains. $v(S)$ is used to denote the evaluation of intentional element S.

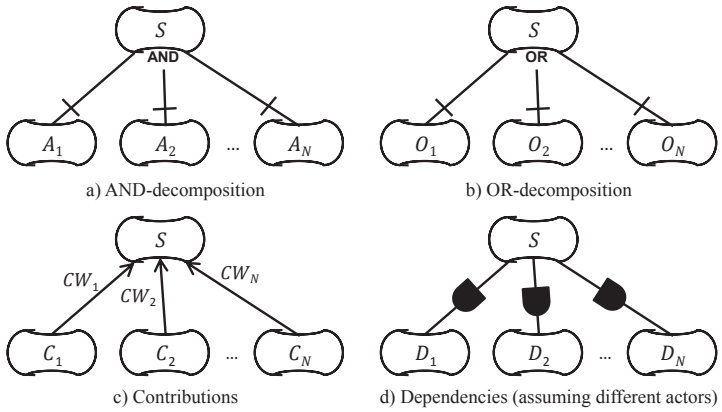

Fig. 1. GRL link types

Fig. 1 illustrates the four types of links we consider in this work. S, A_x, O_x, C_x, and D_x are all intentional elements whereas CW_x are contribution weights. In GRL, the relationships that exist between the sources and targets of these links are as follows.

- For the AND-decomposition (Fig. 1a), the satisfaction value of the parent S is the minimum satisfaction value of its children: $v(S) = \min_{1 \le x \le N} v(A_x)$.
- For the OR-decomposition (Fig. 1b), the satisfaction value of the parent S is the maximum satisfaction value of its children: $v(S) = \max_{1 \le x \le N} v(O_x)$.
- For contributions (Fig. 1c), the satisfaction value of the target S is the weighted sum of the satisfaction values of its sources, bounded to [–100..100] (we abstract from the notion of tolerance factor discussed in [1]): $v(S) = \max(-100, \min(100, \sum_{x=1}^{N} v(C_x) \times CW_x \,/\, 100))$.
- For dependencies (Fig. 1d), the depender S cannot be more satisfied than its dependees: $\bigwedge_{x=1}^{N} v(S) \le v(D_x)$.

When the same intentional element has multiple types of links, the semantics we provide is aligned with that of [1], i.e., decomposition values are considered as a basis to which contributions are added, and everything is constrained by dependencies. Hence, for AND-decompositions, we get the following general relationship:

$$v(S) = \max(-100, \min(100, \min_{1 \le x \le N} v(A_x) + \sum_{x=1}^{N} v(C_x) \times CW_x/100))$$

$$\wedge$$

$$\bigwedge_{x=1}^{N} v(S) \le v(D_x)$$

The relationship for the OR-decomposition is similar, but with max instead of min.

3.2 Transformation to a Constraint-Oriented Language

The JaCoP library comes with a set of functions and search mechanisms that can support an implementation of the semantics introduced in the previous section. Each intentional element in the GRL model becomes a unique variable in JaCoP, with [–100..100] as a domain. The equations and inequalities describing each of the intentional elements in terms of its links are converted to JaCoP constraints. For example, *B≤A* becomes `XlteqY(A,B)`, *C=min(A,B)* becomes `Min({A,B},C)`, and so on (some complex relationships need to be split into intermediate constraints along the way). Finally, the strategy definition provides constant values to the variables corresponding to the intentional elements initialized by that strategy.

We also experimented with the notion that GRL tasks that are leaf nodes in a model should only have 100 or 0 as possible satisfaction values, denoting the fact that these tasks are either performed/selected completely or not at all. It is trivial to detect such situations and to add corresponding JaCoP constraints.

JaCoP allows for different types of searches through the solution space to come up with a solution, when one exists. For example, one can maximize, minimize, or find median values for the variables. Heuristics are also available to accelerate the search. Multiple solutions can also be reported when more than one exists. However, refining the model while navigating through multiple solutions is left for future work.

3.3 Tool Prototype

Our proof-of-concept implementation integrates the JaCoP library to jUCMNav [9]. When the tool user selects a strategy to evaluate against the GRL model, the model and the strategy are converted on the fly to JaCoP constraints, and a search for a solution is started. We use by default a depth-first minimization search of the solution tree, and the first solution found by JaCoP is sent back to jUCMNav for display, with color coding. Fig. 2 illustrates interesting features of the algorithm and prototype tool. The intentional elements with a (*) and a dashed outline are initialized by the strategy.

Fig. 2a illustrates the traditional situation where we have the equivalent of a bottom-up propagation (leaves are initialized). Our algorithm can actually do everything that the quantitative algorithm in [1] can do (assuming a tolerance factor equal to 0). More interestingly, the tool will provide a solution to a situation where a root node is initialized, as shown in Fig. 2b. In this top-down-like situation, a minimal solution for B is found and returned by the JaCoP engine. In Fig. 2c, an additional constraint on C is added, and hence another minimal solution for B is returned. Note that in this generic case, any node can be initialized, not just roots or leaves (actually, the GRL model does not even need to be a tree; acyclic graphs are supported as well).

The absence of a solution can also be reported by JaCoP, and this is visualized in jUCMNav with a special value (–101, *conflict*) and blue color coding. For instance, in Fig. 2d, if C has a satisfaction level of –50, there is no solution for B such that A's satisfaction equals to 100. The impact of using tasks as leaf nodes (i.e., they can only have 0 or 100 as satisfaction values) is shown in Fig. 2e and Fig. 2f.

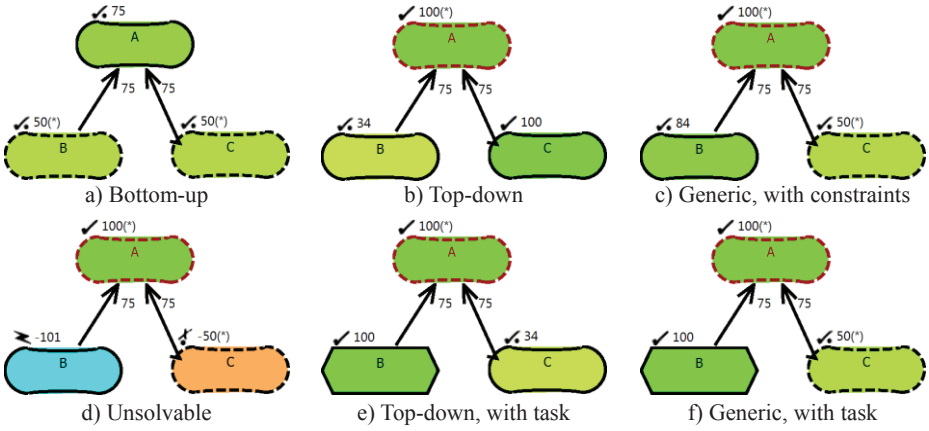

Fig. 2. Examples of GRL models solved with the generic algorithm

4 Conclusions and Related Work

This paper proposes a simple and yet powerful interpretation of GRL models through a declarative semantics amenable to transformations to constraint-oriented languages. The prototype implementation, which combines the jUCMNav modeling tool and the JaCoP constraint solver to support quantitative evaluations, demonstrates the feasibility and potential of the semantics and overall analysis approach.

Not only are the proposed semantics and quantitative propagation algorithm more generic than the existing ones for GRL [1,5], they also bring some benefits that could complement existing work in other goal languages. For example, Giorgini *et al.* [3] provide support for an automated top-down qualitative analysis for TROPOS models, with two values per intentional element (positive and negative evidence) but without conflict resolution nor quantitative values. Horkoff *et al.* [4] propose an interactive and iterative approach to find solutions in *i** models based on single evaluation values and a SAT solver. Their axiomatized qualitative propagation rules are useful for early requirements, conflict resolution, and the handling of multiple sources of weak evidence. However, a fully automated and quantitative approach like ours becomes quite beneficial when more precise and quantitative knowledge about the domain is available. Having the possibility to quickly set and evaluate constraints on intermediate nodes of a goal model is also useful during the exploration of the model (in a sense, this is another level of interactive analysis). In addition, running existing strategies when the model evolves (regression analysis) is more efficient in our context. Letier *et al.* [8] have extended KAOS goal models with probabilities for reasoning about

partial quantitative satisfaction in an automated way. However, we argue that the use of probabilities is another level of complexity and precision that few goal modelers can actually exploit in a pragmatic way, so our approach is likely more accessible.

5 Ongoing and Future Work

Extensions to this work include the support of remaining GRL concepts (e.g., XOR-decomposition, tolerance, and satisfaction at the actor level), but also GRL extensions like key performance indicators. We could also consider a global satisfaction for the entire model computed from actor satisfaction levels. Better formalization of the semantics, including an assessment of its soundness and completeness, are also required. There should also be a way to specify in a GRL strategy whether some goals should be maximized or minimized, and this could be taken into consideration by constraint solvers for an enhanced analysis experience. We also need to explore the usefulness of adapting this work to qualitative and hybrid evaluations. Although early performance results of our prototype are encouraging (most average-sized models we have played with can be solved within several milliseconds, hence enabling modelers to explore different values for a strategy on the fly), we often faced situations where a certain combination of model/strategy requires hours to solve. These situations need to be investigated, with a way to stop long evaluations. Finally, we believe that such declarative semantics will enable researchers to explore the usability of various semantics of goal modeling languages (in terms of complexity and customization) for different categories of users, application domains, and development phases.

Acknowledgements. We thank NSERC for financial support, and P. Heymans and F. Roucoux for useful discussions on the topics of GRL formalization and analysis.

References

1. Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., Yu, E.: Evaluating Goal Models within the Goal-oriented Requirement Language. International Journal of Intelligent Systems (IJIS), Vol. 25, Issue 8, 841-877 (August 2010)
2. Amyot, D., Mussbacher, G.: User Requirements Notation: The First Ten Years, The Next Ten Years. Invited paper, Journal of Software (JSW), Vol. 6, No. 5, 747-768 (May 2011)
3. Giorgini, P., Mylopoulos, J., Sebastiani, R.: Goal-oriented requirements analysis and reasoning in the Tropos methodology. Eng. Appl. of AI, 18(2):159-171 (2005)
4. Horkoff, J., Yu, E.: Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach. Conceptual Modeling – ER 2010. Springer, LNCS 6412:59-75 (2010)
5. ITU-T: User Requirements Notation (URN) – Language definition, ITU-T Recommendation Z.151 (11/08). Geneva, Switzerland (2008); http://www.itu.int/rec/T-REC-Z.151/en
6. JaCoP - Java Constraint Programming Solver, ver. 3.1 (2011); http://www.osolpro.com/
7. jUCMNav 4.4.0 (2011); http://softwareengineering.ca/jucmnav
8. Letier, E., van Lamsweerde, A.: Reasoning about partial goal satisfaction for requirements and design engineering. 12th Foundations of Software Eng. (FSE-12): 53-62, ACM (2004)
9. Luo, H.: Generic Propagation Algorithm for Goal Models. M.Sc. project, SITE, University of Ottawa, Canada (2011); http://www.UseCaseMaps.org/pub/