An ASP Based Reasoning Scheme for Contextual Requirements Modeling

Jasmine Yang, Sadie Clewien

Abstract

Requirements Engineering (RE) is meant to collect information from stakeholders and present it in a useful manner. This is an important step in software engineering and systems engineering. Goal oriented RE is a way to elicit needs and concerns of a system via goal based reasoning. Our goal was to create a ctool using Answer Set Programming to verify some properties of well formedness. Previous work has failed to create analysis systems using Answer Set Programming. There is also no other tool that does the same type of reasoning on contextual tropos model. Using a new model based on the contextual tropos model, we have created an analysis tool that verifies well formedness of a model. Using a library of violations we deemed important, our analysis tool checks seven parameters of well formedness to verify that a meaningful model was created. This can later be expanded to do other types of analysis, and provides an easy-to-use method of verification of user created models.

Introduction

Requirements engineering is a field of study under software engineering. Requirements engineering defines and processes requirements of a system, which is useful for eliciting requirements or analysis. An effective method of requirements engineering is through goal models. Goal models are an important step in understanding how actors in a system interact with each other and their motivations. These models can tell us what flaws exist in a system, how the user plans to execute goals, and if any conflicts exist. Existing goal models have common elements, including: actors, goals, tasks, context, and obstacles. Each model has its own set of requirements and analysis.

We looked at an existing modeling and analysis tool, STS-Tool, developed by the University of Trento in Italy. STS-Tool allows for users to build a system model with additional focus about the information transmitted in the system and the security of it, but the model itself did not allow for context. Without context, the model was lacking in adaptive properties.

The STS-Tool's modeling is based off the tropos model.The tropos model has the same elements, but does not contain the same security concerns. The tropos model can be extended to the contextual tropos model

The contextual tropos goal model allows for goals and tasks to be supplied context, allowing actors to independently decide which decisions are the best course of action. Context can have incredible impact on the goals and the ways in which these goals can be accomplished, therefore providing a way for context to be represented in our model was significantly important.

Texas Tech University, 2014 – 2016
Declarative Programming REU

Methodology

Our goal model is based off of the Tropos architecture. The elements of our goal models are agents, goals, tasks, and context. Agents can be systems or humans. These agents own goals which they are responsible for completing. Goals can be decomposed into subgoals, using AND or OR decomposition. AND decomposition specifies that all sub-goals must be completed for the parent goal to be considered complete. OR decomposition is satisfied when one subgoal is satisfied. Goals are satisfied when the tasks assigned to them are completed. Tasks must be actions the agent can take to complete the goal. These tasks are attached to the goal by task links. Tasks can also be decomposed into subtasks. Goal decomposition, task decomposition, and task links all can have context attached to the relationship. Using a JSON file, the model can be parsed into an ASP file to run the validation check on.

Our analysis checks for seven potential well-formedness violations. Well-formedness verification checks that the model is correctly built, so that it is usable in the real world or other analysis can be run on it later on. The analysis checks that the goals that are decomposed are decomposed into two or more goals. The same check is applied to tasks as well. The system also checks that there is not a decomposition cycle occurring. A decomposition cycle will occur if a child goal decomposes into one of the root goals. The system also checks for decomposition cycles in tasks as well. Next, the system checks that all goals have an agent assigned to them. The system also checks that goals are refined into either sub-goals or tasks, so that there is a documented method to achieve some goal. The program then checks that the tasks are justified by a goal. Tasks should not exist without a goal. These rules provide universal verification.

Program Architecture:

Our program consists of a parser that reads a JSON file that represents a goal model. The parser writes a SPARC file that includes the library of violations. The program then gets the output from the ASP solver, as shown in Figure 1. The output is currently a list of ASP predicates that gives the violation found and the goal or task that created the violation.

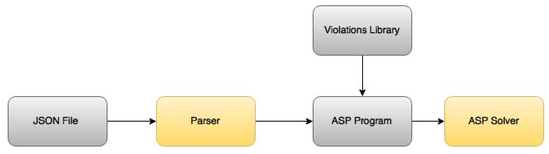


Figure 1. Program architecture

Case Study:

Imagine a simplified ATM. This ATM can only perform three types of transactions, as shown in figure 2. This graphic is a representation of a model with one agent (the ATM) with one root goal. The goals are represented by ovals and the tasks are represented with hexagons. Goal decomposition and task links are both represented with arrows. The type of decomposition is shown on the origin point of the arrows. If there is context attached, it is represented with an ID shown on the link. We did initial testing on the system with this model. We first converted it into a JSON file that the program could read. We also created the model in ASP manually, to compare the results. All of the agents, goals, tasks, context, and links were written into an ASP file along with the library of violation checks. This model generated no violations, as expected. When we injected conflict, by removing a task or goal, the system was able to successfully find the issue.

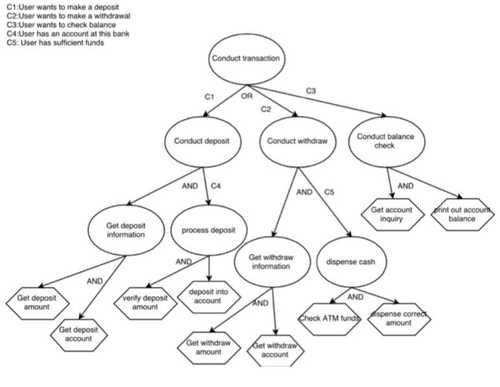
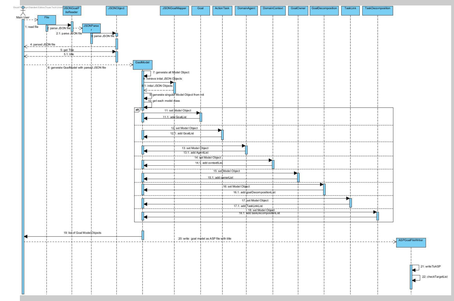


Figure 2. Graphical Representation of model

```
%%goal single decomposition
goalSingleDecomposition(X,Y) :- goalDec(D,X,Y,C), not hasMoreGoalDecomposition(X,Y).
hasMoreGoalDecomposition(X,Y) :- goalDec(D,X,Z,C), Y!=Z.

%%task single decomposition
taskSingleDecomposition(T,U) :- taskDec(D,T,U, C), not hasMoreTaskDecomposition(T,U).
hasMoreTaskDecomposition(T,U):- taskDec(D,T,V,C), U!=V.

%%goal decomposition cycle
inGoalCycle(G) :- reachableGoal(G,G).
reachableGoal(G1, G2) :- goalDec(D, G1,G2, C).
reachableGoal(G1, G2) :- goalDec(D, G1, G3, C), reachableGoal(G3, G2).

%%task decomposition cycle
inTaskCycle(T) :- reachableTask(T1, T1).
reachableTask(T1, T2) :- taskDec(D, T1, T2, C).
reachableTask(T1, T2) :- taskDec(D, T1, T3, C), reachableTask(T3, T2).

%%goal without ownership
noOwnership(G) :- not hasOwner(G).
hasOwner(G) :- ownsGoal(A, G).

%%goal is not refined
noRefinedGoal(G) :- goals(G, T), not isRefinedGoal(G).
isRefinedGoal(G) :- hasTaskDecomposition(G).
isRefinedGoal(G) :- hasGoalDecomposition(G).
hasTaskDecomposition(G) :- taskLink(G, T, C).
hasGoalDecomposition(G) :- goalDec(D, G, G1, C).

%%task is not justified
taskNotJustified(T) :- not taskJustified(T).
taskJustified(T) :- taskLink(G,T,C).
```

Figure 3. Sample of ASP violations library



This figure shows the sequence diagram of our program. The sequence diagram shows the objects in the program and the relationship between them.

Future Work

Future elements to be implemented in our project include, but are not limited to:

- A GUI that will allow users to easily interact with our software
- Soft goals so that qualitative goals can be substantiated
- AND/OR decomposition for task links
- AND/OR decomposition for task decomposition

Conclusion

Goal models are a useful tool for business analysis, software engineering, and other applications. We have created a goal model and a program to analyze the model's well formedness. This program has performed correctly with all current test cases. This program can be later expanded for more robust models.

References:

Ali, R., Dalpiaz, F., & Giorgini, P. (2010). A goal-based framework for contextual requirements modeling and analysis. *Requirements Engineering*, *15*(4), 439-458.

Gelfond, M., & Kahl, Y. (2014). *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. New York, NY: Cambridge University Press.

Dalpiaz, F., Giorgini, P., & Mylopoulos, J. (2013). Adaptive socio-technical systems: a requirements-based approach. *Requirements engineering*, *18*(1), 1-24.

Paja, E., Poggianella, M., Dalpiaz, F., Roberti, P., & Giorgini, P. (2014). Security Requirements Engineering with STS-Tool. In *Secure and Trustworthy Service Composition* (pp. 95-109). Springer International Publishing.

Acknowledgements:

Sara Sartoli
Oluwabukunmi (Lola) Oyedeji
Yuanlin Zhang, phD
Michael Gelfond, phD