Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach

Jennifer Horkoff[1] and Eric Yu[2]

[1] University of Toronto, Department of Computer Science
[2] University of Toronto, Faculty of Information
`jenhork@cs.utoronto.ca, yu@ischool.utoronto.ca`

Abstract. Modeling in the early stage of system analysis is critical for understanding stakeholders, their needs, problems, and different viewpoints. We advocate methods for early domain exploration which provoke iteration over captured knowledge, helping to guide elicitation, and facilitating early scoping and decision making. Specifically, we provide a framework to support interactive, iterative analysis over goal- and agent-oriented (agent-goal) models. Previous work has introduced an interactive evaluation procedure propagating forward from alternatives allowing users to ask "What if?" questions. In this work we introduce a backwards, iterative, interactive evaluation procedure propagating backward from high-level target goals, allowing users to ask "Is this possible?" questions. The approach is novel in that it axiomatizes propagation in the i* framework, including the role of human intervention to potentially resolve conflicting contributions or promote multiple sources of weak evidence.

Keywords: Goal- and Agent-Oriented Modeling, Early System Analysis, Model Analysis, Interactive Analysis, Iterative Analysis.

1 Introduction

Understanding gained during early stages of system analysis, including knowledge of stakeholders, their needs, and inherent domain problems, can be critical for the success of a socio-technical system. Early stages of analysis are characterized by incomplete and imprecise information. It is often hard to quantify or formalize critical success criteria such as privacy, security, employee happiness, or customer satisfaction in early stages. Ideally, early analysis should involve a high-degree of stakeholder participation, not only gathering information, but presenting information gathered thus far, allowing validation and improved understanding in an iterative process. Goal- and agent-oriented models (agent-goal models) have been widely advocated for early system analysis [1] [2], as such models allow even imprecise concepts to be reasoned about in terms of softgoals and contribution links, and have a relatively simple syntax, making them amenable to stakeholder participation.

We advocate methods for early domain exploration which provoke and support iterative inquiry over captured knowledge, prompting analysts and stakeholders

to review what is known, helping to guide elicitation, and facilitating early scoping and decision making. To this end we have created a framework for iterative, interactive analysis of agent-goal models in early system analysis. Previous work has introduced an interactive procedure which propagates evidence from means to ends, allowing users to ask "what if?" questions [3]. In this work we introduce an interactive "backward" procedure, propagating target values from ends to means, helping users to ask "Is this possible?", "If so how?" and "If not, why not?" questions.

The procedure introduced in this paper encodes forward and backward propagation rules in conjunctive normal form (CNF), iteratively applying a SAT solver and human intervention to search for an acceptable solution. In formulating such an interactive backward procedure we face some interesting questions and technical challenges. What types of questions could and should be posed to the user, and at what point in the procedure? How can the encoding be modified to reflect human judgment, what is added, what is removed? When a choice does not lead to an acceptable solution, to what state does the procedure backtrack? As information is lost in forward propagation when evidence is manually combined, what assumptions about this evidence can be made when propagating backward? How can the axiomization allow for explicit values of conflict and unknown, compared to approaches that only allow for positive and negative values [4]? How can we find a balance between constraining the problem sufficiently to avoid nonsensical values and allowing enough freedom to detect the need for human judgment? How can we use information about SAT failures to inform the user? Is there a computationally realistic approach? The procedure in this work represents one approach to answering these questions.

The paper is organized as follows: an overview of the framework for iterative, interactive analysis for agent-goal models is provided (Section 2), including a summary of the forward propagation procedure (2.1). We motivate the need for backward analysis (2.2), and provide an overview of the proposed backward analysis procedure (3). Background on SAT solvers are provided (3.1) along with a formalization of the i* Framework as an example agent-goal syntax (3.2), including axioms for forward and backward propagation. The iterative, backward algorithm is described in (3.5), including an example and a consideration of termination, run time, soundness, and completeness. Related work is described in Section 4, with discussion, conclusions, and future work in Section 5.

2   A Framework for Iterative, Interactive Analysis of Agent-Goal Models in Early System Analysis

We introduce a framework for iterative, interactive analysis of agent-goal models consisting of the following components [5]:

- An interactive, qualitative forward analysis procedure, facilitating "What if?" analysis.
- Management of results for each analyzed alternatives.

- An interactive, qualitative backward analysis procedure, facilitating "Is this possible?", "If so, how?", and "If not, why?" analysis.
- Management of human judgments provided by users.
- Integration with textual justifications for modeling and evaluation decisions.
- Reflecting model and judgment changes in alternative evaluation results.

Currently, the first component has been implemented, applied, and described in detail [6] [3] [7], with the first and second component implemented in the OpenOME tool [8]. In this work, we focus on the third component: interactive, qualitative backward analysis. Our analysis framework uses the i* notation as an example goal modeling framework [2], but could be applicable to any goal models using softgoals and/or contribution links.

2.1   Background: Forward Interactive Analysis

The forward analysis procedure starts with an analysis question of the form "How effective is an alternative with respect to goals in the model?" The procedure makes use of a set of qualitative evaluation labels assigned to intentions to express their degree of satisfaction or denial, shown in the left column of Table 2. Following [1], the *(Partially) Satisfied* label represents the presence of evidence which is (insufficient) sufficient to satisfy an intention. *Partially Denied* and *Denied* have the same definition with respect to negative evidence. *Conflict* indicates the presence of positive and negative evidence of roughly the same strength. *Unknown* represents the presence of evidence with an unknown effect. Although tasks, resources, and goals typically have a binary nature (true or false), the use of softgoal decompositions or dependencies on softgoals means that they often encompass quality attributes. We allow partial labels for tasks, resources, and goals for greater expressiveness.

The analysis starts by assigning labels to intentions related to the analysis question. These values are propagated through links using defined rules. See [2] or [9] for a review of i* syntax (legend in Fig. 2.2). The nature of a *Dependency* indicates that if the element depended upon (*dependee*) is satisfied then the element depended for (*dependum*) and element depending on (*depender*) will be satisfied. *Decomposition* links depict the elements necessary to accomplish a task, indicating the use of an AND relationship, selecting the "minimum" value amongst all of the values, using the ordering in (1). Similarly, *Means-Ends* links depicts the alternative tasks which are able to satisfy a goal, indicating an OR relationship, taking the maximum values of intentions in the relation. To increase flexibility, the OR is interpreted to be inclusive.

$$ \checkmark < \checkmark. < \overset{.}{?} < \mathbf{\times} < \overset{.}{\cancel{\times}} < \cancel{\times} \ . \tag{1} $$

We adopt the *Contribution* link propagation rules from [1], shown in Table 1. These rules reflect the intuitive semantics of contribution links.

The interactive nature of the procedure begins when human judgment is used to combine multiple incoming conflicting or partial values to determine the satisfaction or denial of a softgoal. In some cases, the labels can be automatically

Table 1. Propagation Rules Showing Resulting Labels for Contribution Links

Source Label		Contribution Link Type						
	Name	Make	Help	Some+	Break	Hurt	Some-	Unkn.
✓	Satisfied	✓	✓.	✓.	✗	✗.	✗.	?
✓.	Partially Satisfied	✓.	✓.	✓.	✗.	✗.	✗.	?
?	Unknown	?	?	?	?	?	?	?
➤	Conflict	➤	➤	➤	➤	➤	➤	?
✗.	Partially Denied	✗.	✗.	✗.	✓.	✓.	✓.	?
✗	Denied	✗	✗.	✗.	✓.	✓.	✓.	?

determined. For example, if all labels are positive and a satisfied label is present, the result is satisfied. In other cases the labels are determined by human judgment, prompting the evaluators for a resolution. Human judgment may be as simple as promoting partial values to a full value, or may involve combining many sources of conflicting evidence. When making judgments, domain knowledge related to the destination and source intentions should be used. Human judgment situations are areas where stakeholders involved in the modeling process can have discussions over the contents of the model and the related concepts in the underlying domain.

Once the procedure has finished interactive propagation, the final satisfaction and denial values for the intentions of each actor are analyzed in light of the original question. An assessment is made as to whether the design choice is satisficed ("good enough"), stimulating further analysis and potential model refinement.

Example: To illustrate forward and backward analysis we use a simple model of a generic application shown in Fig. 2.2. The application needs to Implement Password System with two options identified: Restrict Structure of Password, for example, must have a numeral and be more than five characters, and Ask for Secret Question, in case of password loss. These options are not mutually exclusive. The overall goal is to Attract Users, which is helped by both Security and Usability. Restrict Structure of Password makes Security (according to this model), but hurts Usability, while Ask for Secret Question helps Usability. In applying forward evaluation to this example, users would evaluate each feasible alternative. In this case there are three feasible evaluations, with one or the other alternative selected, or both. For example, in Fig. 1 we evaluate the alternative where Ask for Secret Question is satisfied but Restrict Structure of Password is denied. The first pass of the procedure propagates values to Implement Password System and Security automatically, but prompts the user for human judgment on Usability, with incoming values of partially satisfied from Ask for Secret Question and partially satisfied from Restrict Structure of Password. In this case the user decides Usability is partially satisfied. Next they are asked about Attract Users,
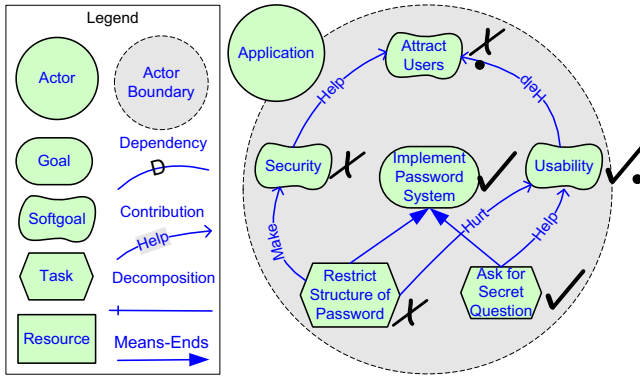
Fig. 1. Example i* model with Forward Analysis Results for an Alternative

receiving values of partially denied and partially satisfied. The user decides that with only partial usability and no security, Attract Users is partially denied.

2.2 The Need for Backward Interactive Analysis

In addition to "What if?"' questions, users also want to be able to answer questions such as "Is this goal achievable?", "If so, how?", and "If not, why?" For example, is it possible for Attract Users to be at least partially satisfied, and if so, how? To answer this type of question, we need a "backward" procedure which starts at the intention(s) of interest and, using the same propagation rules as the forward procedure when possible, works down the links in the model, looking for potential solutions. To be consistent with the results of the forwards procedure, this procedure must prompt for human judgment in situations where labels cannot be determined without human input.

One way to find answers for these questions would be to apply the forwards procedure repeatedly and exhaustively for all reasonable alternatives until either the desired values are produced, or not. However, this approach could be tedious and laborious, especially for larger models with many alternatives. In addition, if it is not possible to obtain a target value, it would be useful to know "why?", identifying the areas of the graph involved in the conflict. In the next section we describe a procedure aiming to answer these questions for agent-goal models.

3 Qualitative, Interactive, Iterative, Backward Analysis for Agent-Goal Models

The approach encodes the model in a SAT formula, then iteratively runs the SAT solver, prompting the user for input regarding intentions which required human judgment after each run. When human judgment is no longer needed and a satisfying assignment is found, the procedure ends, providing an answer. If a satisfying assignment is not found the procedure tries to backtrack over

human judgments. If a satisfying assignment is not found and no further human input can be given, the procedure ends, informing the user that the target is not possible. The choice of SAT as an implementation tool is discussed in Section 5. The procedure has been implemented in the OpenOME Tool [8].

Characterizing agent-goal model propagation in CNF requires a more formal definition for agent-goal model (in our case, i*) concepts, including evaluation values (intention labels). We develop axioms expressing i* propagation in both directions, including necessary constraints over evaluation values, and the encoding of human judgment. We describe the use of a SAT solver in an iterative procedure in more detail, using our simple example to illustrate. Finally, we consider run time, termination, soundness, and completeness for the procedure.

3.1 Background: SAT and Unsatisfiable Core

SAT solvers are algorithms which accept a Boolean formula in conjunctive normal form (CNF), composed of a conjunction of clauses. The algorithm searches for a truth assignment of the formula's clauses to make the formula true. It does so by making a series of decisions concerning the values of variables, backtracking if a decision proves to be not viable. Although the SAT problem is NP-Complete, algorithms and tools that can solve many SAT problems in a reasonable amount of time have been developed, for example, the zChaff tool [10], used in this work.

When a SAT solver fails to find a satisfying assignment, it is useful to know about the underlying conflict(s). Further improvements on SAT algorithms have resulted in the ability to find an unsatisfiable core, a list of clauses in the CNF which result in a conflict. These clauses can be used to form a resolution proof, showing how the clauses work together to produce a conflict ($a \wedge \neg a$). Finding a minimal unsat core is a computationally difficult problem [11], but many approaches exist for finding a small but not minimum core (for example [12]). We use the zMinimal application provided with zChaff to find a small but not minimal unsat core, showing the user a list of intentions included in the set of conflicting clauses when the SAT solver fails to find a solution.

3.2 Formally Expressing the i* Framework

We use the following notation:

- $\mapsto$ is used as a mapping from an intention or relation to a member of a set, so $i \mapsto \{a, b\}$ means that i maps to either a or b.
- $\rightarrow$ is used to represent relationships between elements, so if $(i_1, i_2) \in \mathcal{R}$ we write this as $\mathcal{R} : i_1 \rightarrow i_2$.

An Agent-Goal (i*) Model. In order to encode agent-goal propagation into a SAT formula, we express agent-goal model concepts such as actors and softgoals formally as follows.

Definition: agent-goal model. An i* model is a tuple $\mathcal{M} = < \mathcal{I}, \mathcal{R}, \mathcal{A} >$, where $\mathcal{I}$ is a set of intentions, $\mathcal{R}$ is a set of relations between intentions, and $\mathcal{A}$ is set of actors.

Definition: element type.   Each intention maps to one type in the
$IntentionType$ set, $\mathcal{I} \mapsto IntentionType$, where $IntentionType =$
$\{Softgoal, Goal, Task, Resource\}$.

Definition: relation type. Each relations maps to one type in the Relation-
Type set, $\mathcal{R} \mapsto RelationType$, where $RelationType = \{R^{me}, R^{dec}, R^{dep}, R^c\}$.
These relationships correspond to means-ends, decomposition, dependency,
and contribution links, respectively. R^c can be broken down into a further
set $ContributionType = \{R^m, R^{hlp}, R^u, R^{hrt}, R^b\}$ where if $r \in R \mapsto R^c$ then
$r \mapsto ContributionType$. The contribution link types correspond to make, help,
unknown, hurt, and break, respectively.

Definition: relation behavior.   The following relationships are binary
(one intention relates to one intention, $R : I \to I$): R^{dep}, R^c. The remaining
relationships, R^{me}, R^{dec}, are (n+1)-ary (one to many intentions relate to one
intention), $R : I \times \ldots \times I \to I$. When describing relations, the intentions on the
left hand side of a relation are referred to as sources, while the intention on the
right hand side is referred to as a destination.

The formalism could be supplemented with actor types and association links,
but as these types currently do not play a role in the automated portion of
the evaluation framework, we leave them out of our formalism. Some+ and
Some- links are included in the tool implementation, treated conservatively as
help or hurt, respectively. We exclude these links from $ContributionType$ in the
formalism for simplicity.
    We define other useful concepts such as leaf, root, positive and negative links.

Definition: leaf or root intention. An intention $i \in I$ is a leaf if there does
not exist any relation, $r \in R$ such that $r : I \to i$ or $r : I \times \ldots \times I \to I$, it is a root
if there does not exist any relation, $r \in R$ such that $r : i \to I$ or $r : i \times \ldots \times I \to I$.

Definition: positive or negative link. A relation $r \in R$ is positive if
$r \mapsto Pos = \{R^m, R^{hlp}\}$, it is negative if $r \mapsto Neg = \{R^b, R^{hrt}\}$.

Restrictions on an agent-goal model. In order to produce an agent-goal
model which can be more easily translated into CNF form and to ensure the
convergence and termination of the algorithm, we place the following restrictions
on the model:

 – Each element has at most one Decomposition, Dependency or Means-Ends re-
   lation which determines its level of satisfaction or denial, i.e., $\forall i \in I$, only one
   of $R^{dep} : I \to i$, $R^{dec} : I \times \ldots \times I \to i$, or $R^{me} : I \times \ldots \times I \to i$ holds for i.
 – The model must have no cycles, i.e., for every path in the model, $r_1, \ldots, r_n \in$
   $R, r_1 : i_1(\times \ldots \times I) \to i_2, r_2 : i_2(\times \ldots \times I) \to i_3, \ldots, r_{n-1} : i_{n-1}(\times \ldots \times I) \to$
   i_n, i_k must not equal i_j, for $1 < i, j < n$.

Analysis Predicates. In order to express evaluation labels, as in the forward procedure, we introduce analysis predicates, similar to those used in [4].

Definition: analysis predicates.   We express agent-goal model analysis labels using the following set of predicates, $\mathcal{V}$, over $i \in I$: $v(i) \in \mathcal{V} \mapsto AnalysisPredicates = \{S(i), PS(i), C(i), U(i), PD(i), D(i)\}$ where S(i)/PS(i) represents full/partial satisfaction, C(i) represents conflict, U(i) represents unknown, and D(i)/PD(i) represents full/partial denial.

For example, we express our target for Fig. 2.2, the partial satisfaction of Attract Users, as PS(Attract Users). If this predicate is true, Attract Users is partially satisfied, if it is false, Attract Users is not partially satisfied, (telling us nothing about the application of the other evaluation labels to this intention).

   Like the forward procedure, we choose to treat conflicts as a value to propagate, as opposed to something derived from other values. However, we still use the term "conflict" to indicate a situation where more than one analysis predicates hold for an intention, and those predicates are conflicting.

Definition: (conflict label vs. conflict.) A conflict label is the   ⬆ label originating when a user has selected conflict in human judgment. A conflict between labels for an intention $i \in I$ is when a predicate from more than one of the following four sets is true: $\{S(i), PS(i)\}, \{U(i)\}, \{C(i)\}, \{PD(i), D(i)\}$.

3.3   Expressing Qualitative, Interactive Propagation in CNF

To express the problem of assigning evaluation labels to an agent-goal model in terms of a CNF SAT formula, we follow the formalization in [4], adopting their classification of the components of the formula as follows:

- The target values for the procedure, $\phi Target$
- Axioms describing forward propagation, $\phi Forward$
- Axioms describing backward propagation, $\phi Backward$
- Axioms describing invariant properties of evaluation labels, $\phi Invariant$
- Any additional constraints on propagation, $\phi Constraints$

   The SAT formula is constructed as follows:

$$\phi = \phi Target \wedge \phi Forward \wedge \phi Backward \wedge \phi Invariant \wedge \phi Constraints \ . \quad (2)$$

Target. The target for an evaluation is simply a conjunction of the desired values for each target intention. We could constrain the target further by saying that the target should only have that value, for example if our target is PS(i), we add $\neg C(i)$ and $\neg U(i)$ and $\neg PD(i)$, but we want to allow for targets to have conflicting values, making them candidates for human intervention.

Invariant Axioms. Unlike the approach of [4], we are not able to define a total order over analysis predicates, such that for $v(i) \in AnalysisPredicates$,

$v_1 \geq v_2 \Leftrightarrow v_1 \to v_2$, as there are no implication relationships between satisfaction/denial values and unknown values. We chose not to add implication values from $\{S, PS, PD, D\}$ to Conflict labels (e.g., $PD(i) \wedge PS(i) \to C(i)$), due to our treatment of such labels as described in Section 3.2. We are, however, able to define and utilize the following partial orders.

$$\forall i \in I : S(i) \geq PS(i) \Leftrightarrow S(i) \to PS(i)$$
$$D(i) \geq PD(i) \Leftrightarrow D(i) \to PD(i) \ . \tag{3}$$

In addition, we can define a conceptually useful total order where $v_1 \geq v_2$ implies that v_1 is more desirable (or "higher") than v_2, similar to the order used in (1). This order is as follows:

$$S(i) \geq PS(i) \geq U(i) \geq C(i) \geq PD(i) \geq D(i) \ . \tag{4}$$

Here we chose an optimistic ordering between $U(i)$ and $C(i)$, with the idea that no information (unknown) is better than conflicting information.

Constraints. When using the analysis procedure, the user could add any additional constraints into the SAT formula, following the approach of [4]. In our example, we constrain leaf intentions such that these intentions must be assigned one of the six evaluation labels (5).

$$\forall i \in I \text{ , s.t. i is a leaf: } PS(i) \vee C(i) \vee U(i) \vee PD(i) \tag{5}$$

In our example, we would add these constraints for our two leaf intentions, Restrict Structure of Password and Ask for Secret Question.

3.4 Forward and Backward Propagation Axioms

In order to express the forward and backward propagation rules we develop axioms which express the results of each possible evaluation label when propagated through each type of relation in each direction. Generally, for an intention $i \in I$, $R : i_1 \times \ldots \times i_n \to i$ these predicates take on the form:

Forward Propagation:
(Some combination of $v(i_1) \ldots v(i_n)$, $v \in \mathcal{V}) \to v(i)$
Backward Propagation:
$v(i) \to$ (Some combination of $v(i_1) \ldots v(i_n)$, $v \in \mathcal{V})$

The forward propagation axioms can be derived from the propagation rules described in Section 2.1. For Dependency, Decomposition, and Means-Ends links, the backward propagation rules are identical to the forward, but in the opposite direction. For example, in a Means-Ends relationships with two sources b and c to destination a, either b or c must be satisfied for a to be satisfied in the forward direction, $(S(b) \vee S(c)) \to S(a)$. In the backward direction, if a is satisfied, then either b or c must be satisfied, $S(a) \to (S(b) \vee S(c))$. The SAT solver will try to find a satisfying assignment where either $S(b)$ or $S(c)$ or both are true.

The general form for forward and backward propagation of full satisfaction for Means-Ends links with n sources and destination i_J is $(\bigvee_{j=1}^{n} S(i_j)) \rightarrow S(i_d)$ and $S(i_d) \rightarrow (\bigvee_{j=1}^{n} S(i_j))$, respectively. The other axioms for Means-Ends or Decomposition use similar reasoning. We list only the forward axioms for these links in Table 2. Axioms in the table have been simplified when possible using the invariant clauses in Equation (3).

Propagation axioms for Contribution links are treated differently, as in the forward direction when an intention, i, is the recipient of mulitple contribution links (there exists an $r_1, \ldots, r_n \in \mathcal{R}$ such that $r_1^c : i_1 \rightarrow i \ldots r_n^c : i_n \rightarrow i$), each link from source to destination, r_j for j from $1 \ldots n$, contributes a label. These labels are combined into a single label using either automatic rules or human judgment. In the backward direction a single destination label for i, $v(i_d)$ is used to place constraints on the values of one or more sources, $v_j(i_j) \in \mathcal{V}$, for j from $1 \ldots n$. We can only make minimal reasonable assumptions concerning the labels of the source intentions given the label of the destination intention. For example, if $v(i_d) \mapsto PS$, we assume that at least one of the incoming values is PS, meaning that one of the positive links propagates at least a PS value (i.e. $\exists j, r_j \in Pos$, s.t. $v_j(i_j) \mapsto PS$) or one of the negative links propagates at least a PD value (i.e. $\exists k, r_k \in Neg$, s.t. $v_k(i_k) \mapsto PD$). The rest of the backward assumptions are similar.

3.5   Human Judgment

As the procedure requires input on intentions which require human judgment, we formally define what it means for an intention to require human judgment.

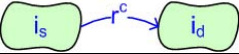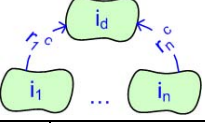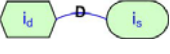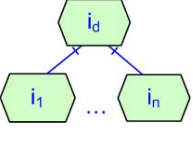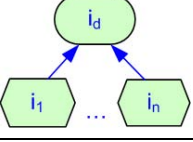**Definition: need for human judgment.** An intention, $i \in I$, needs human judgment if:

- i is the recipient of more than one incoming contribution link, i.e. there exists an r_1 and $r_2 \in \mathcal{R}$ such that $r_1^c : i_1 \rightarrow i$ and $r_2^c : i_2 \rightarrow i$, AND:
  - There is a conflict, as defined in Section 3.2.
  - Or, PS(i) or PD(i) holds and i has not received a human judgment in the current algorithm iteration

Conflicts can be optionally resolved by the user, and partial values can be optionally promoted to S or D.

When human judgment is required for an intention, given a target evaluation value for the recipient intention, target(i), the user is asked the following question:

"Results indicate that i must have a value of target(i).
Enter a combination of evaluation labels for intentions contributing to i which would
result in target(i) for i.
$(\forall j, j = 1 \ldots n, r_j : i_j \rightarrow i)$
I_j, r_j^c, (choice of one of S, PS, U, C, PD, D)
$\ldots$"

Table 2. Foward and Backward Propagation Axioms

Forward Contribution		$\mathbf{V(i_s)}$	$\mathbf{V(i_s)} \rightarrow \mathbf{V(i_d)}$	
		S	$c = m : S(i_s) \rightarrow S(i_d)$	$c = b : S(i_s) \rightarrow D(i_d)$
			$c = hlp : S(i_s) \rightarrow PS(i_d)$	$c = hrt : S(i_s) \rightarrow PD(i_d)$
		PS	$c = m, hlp : PS(i_s) \rightarrow PS(i_d)$	$c = b, hrt : PS(i_s) \rightarrow PD(i_d)$
$\mathbf{V(i_s)}$	$\mathbf{V(i_s)} \rightarrow \mathbf{V(i_d)}$	PD	$c = m, hlp : PD(i_s) \rightarrow PD(i_d)$	$c = b, hrt : PD(i_s) \rightarrow PS(i_d)$
U	$c = any : U(i_s) \rightarrow U(i_d)$	D	$c = m : D(i_s) \rightarrow D(i_d)$	$c = b, hrt : D(i_s) \rightarrow PS(i_d)$
C	$c = any : C(i_s) \rightarrow C(i_d)$		$c = hlp : D(i_s) \rightarrow PD(i_d)$	
		$v \in V$	$c = u : v(i_s) \rightarrow U(i_d)$	

Backward Contribution		$\mathbf{V(i_d)}$	$\mathbf{V(i_d)} \rightarrow \mathbf{V(i_1)} \dots \mathbf{V(i_n)}$
		S, PS	$PS(i_d) \rightarrow$ (for $r_j^c \in$ Pos, $\bigvee_{j=1}^n PS(i_j)$ $\vee$ for $r_j^c \in$ Neg, $\bigvee_{j=1}^n PD(i_j)$)
		C	$C(i_d) \rightarrow (\bigvee_{j=1}^n C(i_j) \vee$ (for $r_j^c \in$ Pos, $\bigvee_{j=1}^n PS(i_j) \wedge$ for $r_j^c \in$ Neg, $\bigvee_{j=1}^n PS(i_j)$)
			$\vee$(for $r_j^c \in$ Pos, $\bigvee_{j=1}^n PD(i_j) \wedge$ for $r_j^c \in$ Neg, $\bigvee_{j=1}^n PD(i_j)$))
$\mathbf{V(i_d)}$	$\mathbf{V(i_d)} \rightarrow \mathbf{V(i_1)} \dots \mathbf{V(i_n)}$	D, PD	$PD(i_d) \rightarrow$ (for $r_j^c \in$ Pos, $\bigvee_{j=1}^n PD(i_j) \vee$ for $r_j^c \in$ Neg, $\bigvee_{j=1}^n PS(i_j)$)
U	$U(i_d) \rightarrow \bigvee_{j=1}^n U(i_j)$		

Forward Dependency		$\mathbf{V(i_s)}$	$\mathbf{V(i_s)} \rightarrow \mathbf{V(i_d)}$
		$v \in V$	$v(i_s) \rightarrow v(i_d)$

Forward Decomposition		$\mathbf{V(i_d)}$	$\mathbf{V(i_1)} \dots \mathbf{V(i_n)} \rightarrow \mathbf{V(i_d)}$
		S	$(\bigwedge_{j=1}^n S(i_j)) \rightarrow S(i_d)$
		PS	$(\bigwedge_{j=1}^n PS(i_j)) \rightarrow PS(i_d)$
		U	$((\bigvee_{j=1}^n U(i_j)) \wedge (\bigwedge_{k=1}^j PS(i_k) \wedge \bigwedge_{p=j+1}^n PS(i_p)) \rightarrow U(i_d)$
		C	$((\bigvee_{j=1}^n C(i_j)) \wedge (\bigwedge_{k=1}^j \neg PD(i_k) \wedge \bigwedge_{p=j+1}^n \neg PD(i_p)) \rightarrow C(i_d)$
		PD	$(\bigvee_{j=1}^n PD(i_j)) \rightarrow PD(i_d)$
		D	$(\bigvee_{j=1}^n D(i_j)) \rightarrow D(i_d)$

Forward Means-Ends		$\mathbf{V(i_d)}$	$\mathbf{V(i_1)} \dots \mathbf{V(i_n)} \rightarrow \mathbf{V(i_d)}$
		S	$(\bigvee_{j=1}^n S(i_j)) \rightarrow S(i_d)$
		PS	$(\bigvee_{j=1}^n PS(i_j)) \rightarrow PS(i_d)$
		U	$((\bigvee_{j=1}^n U(i_j)) \wedge (\bigwedge_{k=1}^j \neg PS(i_k) \wedge \bigwedge_{p=j+1}^n \neg PS(i_p)) \rightarrow U(i_d)$
		C	$((\bigvee_{j=1}^n C(i_j)) \wedge (\bigwedge_{k=1}^j PD(i_k) \wedge \bigwedge_{p=j+1}^n PD(i_p)) \rightarrow C(i_d)$
		PD	$(\bigwedge_{j=1}^n PD(i_j)) \rightarrow PD(i_d)$
		D	$(\bigwedge_{j=1}^n D(i_j)) \rightarrow D(i_d)$

When a judgment is provided the SAT formula is adjusted as follows:

– Forward and backward axioms in the SAT formula which propagate to or from i are removed. These are axioms of the form:

(Any combination of $v(i_1) \dots v(i_n)$, $v \in \mathcal{V}) \rightarrow v(i)$
$v(i) \rightarrow$ (Any combination of $v(i_1) \dots v(i_n)$, $v \in \mathcal{V}$)

– New axioms representing the human judgment are added, for each r_j, r_j^c : $i_j \rightarrow i$, the value provided by the user for i_j, $v_j(i_j) \in \mathcal{V}$, is added to a forward and backward axiom as follows:

Forward: $(v_1(i_1) \wedge \dots \wedge v_n(i_n)) \rightarrow target(i)$
Backward: $target(i) \rightarrow (v_1(i_1) \wedge \dots \wedge v_n(i_n))$

In addition, we when encoding human judgment, we add the constraint that i must not have a conflict, to avoid situations where the SAT solver will assign extra values to i. For example if target = PS(i), then the following would be added to Φ:
$\neg U(i) \wedge \neg C(i) \wedge \neg PD(i)$

3.6   Backward Analysis Algorithm

Simplified Java code implementing the backward algorithm can be found in
Fig. 2. Generally, the algorithm converts the model to CNF form, using the
components of formula described in Section 3.3 (lines 7 and 8 in Fig. 2). Two
versions are converted, one using both the forward and backward propagation
axioms to try to find a solution, cnf, and one using only the backward axioms
in order to find targets for intentions, cnfBack. In a loop which terminates when
no more intentions require human judgment (lines 9, 14, 15), the algorithm calls
zChaff to find a solution for cnf (line 10). If a solution is found (line 11), the
algorithm displays the non-conflicting results (line 13) and finds the topmost
(closest to a root) intentions which need human judgment (line 13, 16). The
target for each of these intentions is found by running the solver on cnfBack
(line 17, 18) and taking the maximum label result for each intention, using the
ordering in 4.

For each topmost intention needing human judgment, the user is prompted
for judgment (line 21), and the judgment is added to the forward and backward
cnf as described in Section 3.3 (line 23, 24). If the user provided some judgments,
the list of topmost intentions needing human judgment is added to a stack (line
27). If, in the main loop, zChaff cannot find a solution (line 28), zMinimal is used
to find the minimum core, which is displayed to the users (line 29-31). In this
case, or when the user has no more judgments to add (line 25, 26), the algorithm
backtracks, popping the last set of intentions needing human judgment from the
stack (line 26) and backtracking over the cnf and cnfBack formula (removing
the judgment axioms and adding back in the default forward and backward
propagation axioms) (line 38, 39). Control is returned to the main loop (line 9)
where the process starts again by finding a solution for the cnf (line 10). Only
judgments over intentions in the minimal core are re-asked when backtracking
(not shown Fig. 2). If the procedure backtracks, but there are no more intentions
to backtrack over, the algorithm ends with no result (line 41-43).

3.7   Example

To illustrate the algorithm, we run an example over the model in Fig. 2.2.

Iteration 1: The SAT solver is run on the cnf SAT formula. A satisfying as-
signment is found; however, there are intentions which need human judgment:
Attract Users and Usability, of which Attract Users is the topmost. We prompt for
human judgment, asking the users what combination would produce a partially
satisfied value for Attract Users. The users indicate that Usability and Security
must be partially satisfied. cnf and cnfBack are modified accordingly and the
procedure loops.

Iteration 2: The SAT solver is called again on the new cnf. Human judgment is still
needed, and the procedure asks the user for input on the conflicted intention near-
est to the root, Usability. The user indicates that for Usability to be partially satisfied

```
0  Dimacs cnf; Dimacs cnfBack;
1  zChaffSolver solver = new zChaffSolver();
2  zMinimalSolver minSolver = new zMinimalSolver();
3  ModeltoAxiomsConverter converter = new ModeltoAxiomsConverter(model);
4  Stack<Vector<Intention>> hjStack = new Stack<Vector<Intention>>();
5
6  void reason() {
7     cnf = converter.convertBothDirections();
8     cnfBack = converter.convertBackward();
9     while( true )  {
10       int result = solver.solve(cnf);
11       if (result == 1) {
12          HashMap<Intention, int[]> results = solver.getResults();
13          Vector<Intention> needHJ = findHJAndDisplayResults(results);
14          if (needHJ.size() == 0) { //answer found, no judgments needed
15             showMessage("Success!");  return; }
16          Vector<Intention> topMostHumanJudgment = findTopMost(needHJ);
17          solver.solve(cnfBack);  //used for intermediate targets
18          Hashmap<Intention, int[]> backResults = solver.getResults();
19          int hjCount = 0;
20          for (Intention i: topMostHumanJudgment) {
21             if (promptForHumanJudgment(i, backResults.get(i))) {
22                hjCount++;  //count number of judgments given
23                cnf = converer.addHumanJudgment(cnf, i);
24                cnfBack = converter.addHumanJudgment(cnfBack, i); } }
25          if (hjCount == 0) {  //user has no more hj to add
26             if (backtrack() == -1) { return; }  }
27          else { hjStack.push(topMostHumanJudgment); } }
28       else if (result == 0) {  //solver found no solution
29          minSolver.solve(cnf);          //find unsat core
30          String minResults = minSolver.getResults();
31          showMessage ("Backtracking: " + minResults);
32          if (backtrack() == -1) { return; }  } } }
33
34 int backtrack() {
35    if (hjStack.size() > 0) {  //there are judgments to backtrack over
36       Vector<Intention> needHJ = hjStack.pop();
37       for (Intention i: needHJ) {  //backtrack over of the last judgments
38          cnf = converter.backtrackHumanJudgment(cnf, i);
39          cnfBack = converter.backtrackHumanJudgment(cnfBack, i);  }
40       return 1;
41    } else {  //there are no judgments to backtrack over
42       showMessage("Target(s) unsatisfiable.  Ending.");
43       return -1;  }  }
```

Fig. 2. Simplified Java Code for the Backward Analysis Algorithm

Ask for Secret Question should be satisfied and Restrict Structure of Password should
be denied. The SAT formulas are modified to reflect this information.

Iteration 3: The solver is run on the new cnf. In this case, the formula is unsatis-
fiable, if Restrict Structure of the Password is denied then Security is denied, when
the rule collected in the first iteration indicates it must be partially satisfied in
order for Attract Users to be partially satified. The procedure backtracks (mod-
ifying the cnf encodings) and the user is then asked for more possible viable
combinations for the last point of judgment, Usability. No more possibilities are
given which would make Usability partially satisfied. The procedure backtracks
again and asks the user if there are more combinations of source intentions that
would produce a partially satisfied value for Attract Users. This time the user indi-
cates that if Security were satisfied and Usability had a conflict value, Attract Users

would be partially satisfied. The axioms to and from Attract Users are again removed and the human judgment axioms are added.

Iteration 4: The solver is run again on the modified cnf. Usability requires human judgment. The user indicates that for Usability to have a conflict value, Restrict Structure of Password and Ask for Secret Question can be satisfied. The encodings are updated.

Iteration 5: The solver is run on the new cnf. This time, not only is satisfying assignment found, but all intentions in the model do not require human judgment. The procedure finishes, informing the user that in order for Attract Users to be partially satisfied, Restrict Structure of Password and Ask for Secret Question must be satisfied.

3.8   Run Time, Termination, Soundness, and Completeness

Run Time: In analyzing the runtime we exclude an exploration of the runtime complexity of zChaff or zMinimal, marking these values as $rt(zChaff)$ and $rt(zMinimal)$. The main loop reason() in Fig. 2 will loop until hjCount $== 0$. In the worst case each iteration involves a single new judgment for every intention. If a model has n intentions and each intention has a maximum of q sources, there is a maximum of $6^q \times n$ possible judgments, where $q < n$. The run time of the initial axiom conversion is $6l$, where l is number of links in the model. The cost of adding or backtracking human judgment on the converter is also l (finding the right axiom by links). In addition, the worst case runtime of findHJAndDisplayResults and is n, findTopMost is $2n$, and backtrack is $2nl$. If zChaff returns a result, the worst case runtime is either $2ln + 3n + rt(zChaff)$ or $2nl$, else it is $2nl + rt(zMinimal)$. Assuming $rt(zMiminal) \approx rt(zChaff)$, the worse case runtime for reason is then $6^q \times n(2ln + 3n + rt(zChaff)) + 6l$, or $O(6^q(ln^2 + nrt(zChaff)))$. Although this is an exponential value, q is usually a small number, less than 5 or 6. Also, although there is a worst case of 6^q possible combinations of human judgment for each intention, only a small subset of these judgments will be acceptable for the user, who will try to maximize positive contributions.

We have applied our implementation of the procedure to several medium sized example models, with the automated portion of the procedure completing within seconds. Future work should test the runtime on larger models, although, as the procedure is meant to be used over models created by hand, the maximum size of such models is reasonably constrained by human cognition. Potential procedure efficiency improvements are discussed in Section 5.

Termination: If the user continues to make the same judgments, the procedure will not terminate. However, the current implementation provides a list of previous judgments attempted which did not produce a solution. As there are a finite number of intentions each with a finite number of sources, there is a finite number of human judgments which can be provided (6^q). If the user does not continually reuse judgments, the procedure terminates.

Soundness: An examination of Table 2 will show that we have considered propagation rules for every combination of evaluation label and link type, given the restrictions on our model in Section 3.2.

Completeness: Our axiomatization would be complete if the propagation through backward axioms were shown to be equivalent to propagation through forward axioms given the same input and human judgment decisions. Currently the forward procedure takes into account additional agent-goal model structures such as mixes of link types and cycles, as a result, the results would only be equivalent if the models avoided these structures. We leave a formal completeness proof for future work.

4   Related Work

An early version of this procedure was briefly described in [13], without providing a detailed encoding or algorithm. In [4], the authors present a formal framework allowing for backward reasoning with goal models. The results for each goal are presented using two values, one for satisfaction and one for denial. Often results contain many goals which are both partially satisfied and denied, making it challenging to derive an overall analysis conclusion. The backwards procedure described in this work could be seen as an expansion or modification of this procedure, as we have borrowed our general CNF formulation (2) and part of our analysis predicates from this work. However, we make several expansions and modifications to [4], as follows:

- Incorporating user interaction through human judgment, allowing users to resolve conflicts and make tradeoffs.
- Accounting for additional agent-goal syntax (dependency, unknown, and some+/- links).
- Accounting for additional analysis values (conflict, unknown).
- Producing results which have only one value per intention.
- Providing information on model conflicts when a solution cannot be found.

Several other analysis procedures have been introduced for agent-goal models, employing methods such as automated quantitative propagation of the probability of goal satisfaction ([14], [15]). However, these procedures often require precise or specific domain information such as probabilities, costs, or quantitative estimates from "experts", difficult to acquire in early analysis stages. It may be difficult for stakeholders to trust results produced automatically over incomplete and imprecise information. We argue these approaches are less appropriate for early system analysis.

5   Discussion, Conclusions and Future Work

We have introduced an interactive, iterative procedure for backward analysis of agent-goal models in early system exploration. This procedure complements the existing forward procedure [3], thus greatly expanding the interactive analytical power of agent-goal models, encouraging stakeholder involvement in the early modeling and analysis process and increasing the likelihood of system success.

The procedure has addressed several of the questions and challenges listed in Section 1. It poses a specific type of question to the user ("What source values could produce a target value?") during iterations where conflicts or unaddressed partial values are detected, modifying the encoding by adding and removing axioms. We have defined assumptions concerning backward propagation over human judgment situations which include explicit conflict and unknown values and which avoid over constraining the model. The run time of the procedure has been analyzed, and although the worst case is exponential over the maximum number of children in the model, in practice this number is small. Limitations include not taking into account some i* constructs such as actor types or associations and restrictions to the structure of the model. We expect to remove these restrictions in future work.

Use of SAT: In the early stages of this work we considered encoding agent-goal model propagation as a Constraint Satisfaction Problem (CSP) or Satisfiability Modulo Theories (SMT) Problem. However, in order to capture the presence of conflicts and the need for human judgment, each intention would have to be assigned multiple variables, making the encoding roughly as complex as our SAT encoding. Consideration was also given to the use of an incremental SAT solver, reusing the state-space when clauses are added to the encoding. However, as our algorithm not only adds, but removes and re-adds clauses, these types of algorithms could not be applied.

Future Procedure Optimizations: We plan to optimize the backward algorithm in several ways. The algorithm in Fig. 2 backtracks by removing and adding clauses from the CNF encoding and recalling the SAT solver. Instead, it could store the zChaff solver results in another stack, popping those results when backtracking, reducing the number of times zChaff is called on average. The number of human judgment situations could be reduced in practice by optionally reusing judgments within a single evaluation, across both forward and backward evaluation, and by deriving judgments from existing judgments.

Use in Practice: We have applied backward analysis to ten individual case studies using graduate and undergraduate students evaluating models created by themselves and others. We have also conducted an action research case study applying agent-goal modeling with forward and backward evaluation to analyze the requirements for the Inflo "back of the envelope" calculation modeling tool. Results revealed that both the individual participants and the Inflo group were able to understand and apply backward analysis. In some cases participants could interpret the results of backward analysis in term of the domain, making interesting discoveries about the cause and effect captured in the model. More i* and evaluation training, or the participation of a facilitator, is needed to increase utility from model analysis. In the Inflo study, backward evaluation provoked interesting discussion concerning constructs in the model, such as Flexibility.

However, application of the procedure revealed several usability issues, some of which have led to improvements in the procedure. Results have shown that the completeness of the model affects the utility of backward analysis - analysis may be less useful on smaller models, or those without negative links. However,

application of judgment helped to reveal model incompleteness in some cases. One of the primary usability issues was understanding the reasons behind conflicts in the model, especially for large models. Future versions of the procedure will present the unsatisfiable core in a more user-friendly way, potentially highlighting the conflicting intentions in the model.

References

1. Chung, L., Nixon, B.A.: Dealing with non-functional requirements: three experimental studies of a process-oriented approach. In: ICSE 1995: Proceedings of the 17th International Conference on Software Engineering, pp. 25–37. ACM, New York (1995)
2. Yu, E.S.K.: Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. In: RE 1997: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering, Washington, DC, USA, p. 226. IEEE Computer Society, Los Alamitos (1997)
3. Horkoff, J., Yu, E.: Evaluating Goal Achievement in Enterprise Modeling An Interactive Procedure and Experiences. In: The Practice of Enterprise Modeling, pp. 145–160. Springer, Heidelberg (2009)
4. Giorgini, P., Mylopoulos, J., Sebastiani, R.: Simple and Minimum-Cost Satisfiability for Goal Models. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 20–35. Springer, Heidelberg (2004)
5. Horkoff, J., Yu, E.: A Framework for Iterative, Interactive Analysis of Agent-Goal Models in Early Requirements Engineering. In: 4th International i* Workshop (submitted, 2010)
6. Horkoff, J., Yu, E.: A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models. In: CAiSE 2009 Forum. CEUR-WS.org, vol. 453, pp. 19–24 (2009)
7. Horkoff, J., Yu, E.: Interactive Analysis of Agent-Goal Models in Enterprise Modeling. International Journal of Information System Modeling and Design (IJISMD) (in press)
8. OpenOME (2010), https://se.cs.toronto.edu/trac/ome/wiki
9. i* Wiki (2010), http://istar.rwth-aachen.de/
10. Mahajan, Y.S., Fu, Z., Malik, S.: Zchaff2004: An Efficient SAT solver. In: H. Hoos, H., Mitchell, D.G. (eds.) SAT 2004. LNCS, vol. 3542, pp. 360–375. Springer, Heidelberg (2005)
11. Zhang, J., Li, S., Shen, S.: Extracting Minimum Unsatisfiable Cores with a Greedy Genetic Algorithm. In: Australian Conference on Artificial Intelligence, pp. 847–856 (2006)
12. Bruni, R., Sassano, A.: Restoring Satisfiability or Maintaining Unsatisfiability by finding small Unsatisfiable Subformulae. Electronic Notes in Discrete Mathematics 9, 162–173 (2001)
13. Horkoff, J., Yu, E.: Qualitative, Interactive, Backward Analysis of i* Models. In: 3rd International i* Workshop, CEUR-WS.org, pp. 4–46 (2008)
14. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Reasoning with goal models. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 167–181. Springer, Heidelberg (2002)
15. Letier, E., van Lamsweerde, A.: Reasoning about partial goal satisfaction for requirements and design engineering. SIGSOFT Softw. Eng. Notes 29(6), 53–62 (2004)