Goal-Oriented Requirements Analysis and Reasoning in the Tropos Methodology

Paolo Giorgini[1], John Mylopoulos[1,2], and Roberto Sebastiani[1]

[1] Department of Information and Communication Technology
University of Trento - Italy
{rseba,pgiorgio}@dit.unitn.it
[2] Department of Computer Science - University of Toronto - Canada
jm@cs.toronto.edu

Abstract. Tropos is an agent-oriented software methodology proposed in [1, 2]. The methodology is founded on the notions of *agent* and *goal*, and goal analysis is used extensively to support software development during different phases. This paper adopts a formal goal model defined and analyzed in [9, 15] to make the goal analysis process concrete through the use of forward and backward reasoning for goal models. The formal goal analysis is illustrated through examples, using an implemented goal reasoning tool.

Keywords: agent-oriented software development, goal-oriented requirements analysis, early requirements analysis, multi-agent systems.

1 Introduction

Tropos [1, 2] is an agent-oriented software development methodology founded on two novel features. First, the methodology is defined in terms of the concepts of *agent*, *goal*, and related mentalistic notions. These notions are used to support all software development phases, from early requirements analysis to implementation. Second, a crucial role is given to early requirements analysis that precedes prescriptive requirements specification for the system-to-be. As such, Tropos supports earlier phases of software development compared to other agent- and object-oriented methodologies.

The main motivation for supporting early requirements [5, 19] is to develop a rich conceptual framework for modeling and analyzing processes that involve multiple participants (both humans and software systems) and the intentions that these processes are supposed to fulfill. By so doing, one can relate the functional and non-functional requirements of the system-to-be to relevant stakeholders and their intentions. Tropos adopts Eric Yus *i** model [19] which offers *actors* (*agents*, *roles*, or *positions*), *goals*, and *actor dependencies* as primitive concepts for models used in different phases of software development. In particular, Tropos is intended to support four phases of software development: *early requirements analysis*, concerned with the understanding of a problem by studying its organizational setting; *late requirements analysis*, where the system-to-be is described within its operational environment, along with relevant functions and qualities; *architectural design*, where the systems global architecture is

defined in terms of subsystems, interconnected through data, control, and other dependencies; and *detailed design*, where behavior of each software component is defined in further detail.

The Tropos methodology has been presented in detail in [1, 2] using two case studies. In both papers, goal analysis is given a prominent role. In a nutshell, software development begins by identifying relevant stakeholders (represented as actors) and their goals. These root goals are analyzed, refined, and delegated to existing on new actors. The system-to-be and its components come about as new actors who are responsible for the fulfillment of some of the original or refined goals. The whole process ends when sufficient goals have been delegated so that if all actors fulfill their responsibilities, all root goals are fulfilled.

The objective of this paper is to make this goal analysis process concrete by using a formal goal model developed in [9, 15]. This goal model supports both qualitative and quantitative relationships between goals, and can be used to perform two types of analysis. The first type (forward reasoning) answers questions of the form: Given a goal model, and assuming that certain leaf goals are fulfilled, are all root goals fulfilled as well? The second type of analysis (backward reasoning) solves problems of the form: Given a goal model, find a set of leaf goals that together fulfill all root goals.

The rest of the paper is structured as follows. Section 2 introduces a case study, adopted from [2], and illustrates the Tropos methodology by showing how it supports requirements analysis phases. Section 3 introduces the formal goal model, while section 4 presents forward and backward reasoning with goal models, using examples from the case study. Section 5 presents a goal reasoning tool that can be used during software development to support goal analysis. Finally, section 6 summarizes the contributions of this paper and discusses possible directions for further research.

2 The Tropos methodology

This section describes and illustrate the requirements analysis phases of the Tropos methodology: *Early Requirements Analysis* and *Late Requirements Analysis*.

2.1 A Case Study

This case study is a revised version of the case study already presented in [2]. *Media Shop* is a store selling and shipping different kinds of media items such as books, newspapers, magazines, audio CDs, videotapes, and the like. *Media Shop* customers (on-site or remote) can use a periodically updated catalogue describing available media items to specify their order. *Media Shop* is supplied with the latest releases from *Media Producer* and in-catalogue items by *Media Supplier*. To increase market share, *Media Shop* has decided to open up a B2C retail sales front on the internet. With the new setup, a customer can order *Media Shop* items in person, by phone, or through the internet. The system has been named *Medi@* and is available on the world-wide-web using communication facilities provided by *Telecom Cpy*. It also uses financial services supplied by *Bank Cpy*, which specializes on on-line transactions. The basic objective

for the new system is to allow an on-line customer to examine the items in the *Medi@* internet catalogue, and place orders.

There are no registration restrictions, or identification procedures for *Medi@* users. Potential customers can search the on-line store by either browsing the catalogue or querying the item database. The catalogue groups media items of the same type into (sub)hierarchies and genres (e.g., audio CDs are classified into pop, rock, jazz, opera, world, classical music, soundtrack, ...) so that customers can browse only (sub)categories of interest. An on-line search engine allows customers with particular items in mind to search title, author/artist and description fields through keywords or full-text search. If the item is not available in the catalogue, the customer has the option of asking *Media Shop* to order it, provided the customer has editor/publisher references (e.g., ISBN, ISSN), and identifies herself (in terms of name and credit card number). Details about media items include title, media category (e.g., book) and genre (e.g., science-fiction), author/artist, short description, editor/publisher international references and information, date, cost, and sometimes pictures (when available).

2.2 Requirements Analysis in Tropos

Requirement analysis represents the initial phase in most software engineering methodologies. Requirements analysis in Tropos consists of two phases: *Early Requirements* and *Late Requirements* analysis. Early requirements is concerned with understanding the organizational context within which the system-to-be will eventually function. Late requirements analysis, on the other hand, is concerned with a definition of the functional and non-functional requirements of the system-to-be.

Tropos adopts the *i** [19] modeling framework for analyzing requirements. In *i** (which stands for "distributed intentionality"), stakeholders are represented as (social) actors who depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. The *i** framework includes the *strategic dependency model* (actor diagram in Tropos) for describing the network of inter-dependencies among actors, as well as the *strategic rationale model* (rationale diagram in Tropos) for describing and supporting the reasoning that each actor goes through concerning its relationships with other actors. These models have been formalized using intentional concepts from Artificial Intelligence, such as goal, belief, ability, and commitment (e.g., [3]). The framework has been presented in detail in [19] and has been related to different application areas, including requirements engineering [17], software processes [18], and business process reengineering [20].

Early Requirements Analysis
During early requirements analysis, the requirements engineer identifies the domain stakeholders and models them as social actors, who depend on one another for goals to be fulfilled, tasks to be performed, and resources to be furnished. Through these dependencies, one can answer *why* questions, besides *what* and *how*, regarding system functionality. Answers to *why* questions ultimately link system functionality to stakeholder needs, preferences and objectives. Actor diagrams and rationale diagrams are used in this phase.

An actor diagram is a graph involving *actors* who have *strategic dependencies* among each other. A dependency represents an "agreement" (called *dependum*) between two actors: the *depender* and the *dependee*. The *depender* depends on the *dependee,* to deliver on the dependum. The dependum can be a *goal* to be fulfilled, a *task* to be performed, or a *resource* to be delivered. In addition, the depender may depend on the dependee for a *softgoal* to be fulfilled. Softgoals represent vaguely defined goals, with no clear-cut criteria for their fulfillment. Graphically, actors are represented as circles; dependums – goals, softgoals, tasks and resources – are respectively represented as ovals, clouds, hexagons and rectangles; and dependencies have the form *depender* → *dependum* → *dependee*.

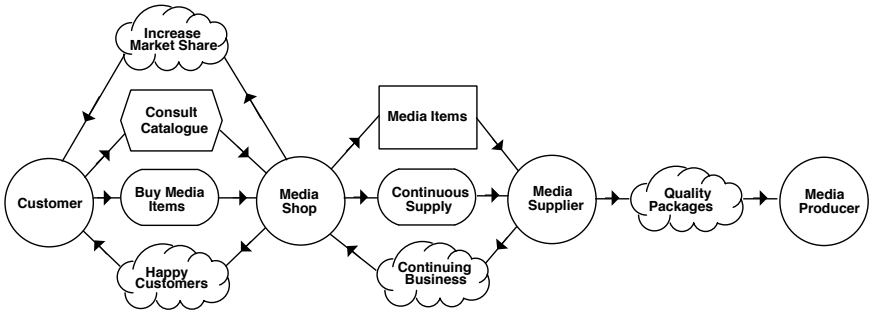

Fig. 1. Actor diagram for a Media Shop

Figure 1 depicts the actor diagram for our Medi@ example. The main actors are *Customer*, *Media Shop*, *Media Supplier* and *Media Producer*. *Customer* depends on *Media Shop* to fulfill her goal: *Buy Media Items*. Conversely, *Media Shop* depends on *Customer* to *increase market share* and make "*customers happy"*. Since the dependum *Happy Customers* cannot be defined precisely, it is represented as a softgoal. The *Customer* also depends on *Media Shop* to *consult the catalogue* (task dependency). Furthermore, *Media Shop* depends on *Media Supplier* to supply media items in a continuous way and get a *Media Item* (resource dependency). The items are expected to be of good quality because, otherwise, the *Continuing Business* dependency would not be fulfilled. Finally, *Media Producer* is expected to provide *Media Supplier* with *Quality Packages.*

Actor diagrams are extended during early requirements analysis by incrementally adding more specific actor dependencies which come out from a means-ends analysis of each goal. This analysis is specified using rationale diagrams.

A rationale diagram appears as a balloon within which goals of a specific actor are analyzed and dependencies with other actors are established. Goals are decomposed into subgoals and positive/negative contributions of subgoals to goals are specified. The intuitive meaning of the positive (+ and ++) and negative (– and – –) contributions, is that the satisfaction of a goal G contributes positively (negatively) to the satisfaction

(denial) of another goal G'. + and ++ (– and – –) specify the different strength of the contribution. In the next section such relationships are formally defined.

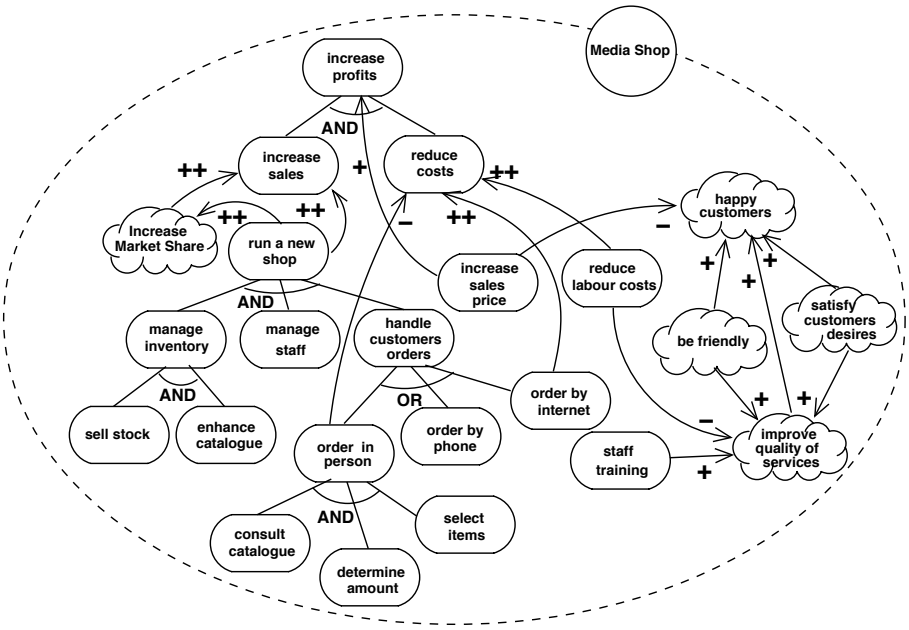

Fig. 2. Rationale diagram for the *Media Shop*

Figure 2 shows the rationale diagram for the *Media Shop* actor focusing on the goal *increase profits*, which is and-decomposed in *increase sales* and *reduce costs*. The *Media Shop* has also the softgoals *happy customers*, *increase market share*, and *improve quality of services*. The goal *run a new shop* gives a positive (++) contribution to the goal *increase sales* and to the softgoal *increase market share*. in order to satisfy to goal *run a new shop*, the *Media Shop* has to *manage inventory*, *manage staff*, and *handle customers orders*. these goals are further refined, so for instance *handle customers orders* can be achieved in three different ways: *order in person*, *order by phone*, or *order by Internet*. Each of these goals gives a different contribution to the goal *reduce costs*; namely, a positive contribution (++) for *order in person* and a negative contribution (–) for *order by internet*. The goal *reduce labour costs* contributes negatively to *improve the quality if services*, while it gives a positive contribution to the goal *reduce costs*. Finally, the softgoal *happy customers* receives positive contributions from the softgoals *be friendly*, *satisfy customers desires*, and *improve quality of service*.

Late Requirements Analysis

During *late requirements* analysis, the conceptual model developed during early requirements is extended to include the system-to-be as a new actor, along with dependencies between this actor and others in its environment. These dependencies define

functional (goals) and non-functional (softgoals) requirements for the system-to-be. Actor diagrams and rationale diagrams are used also in this phase.

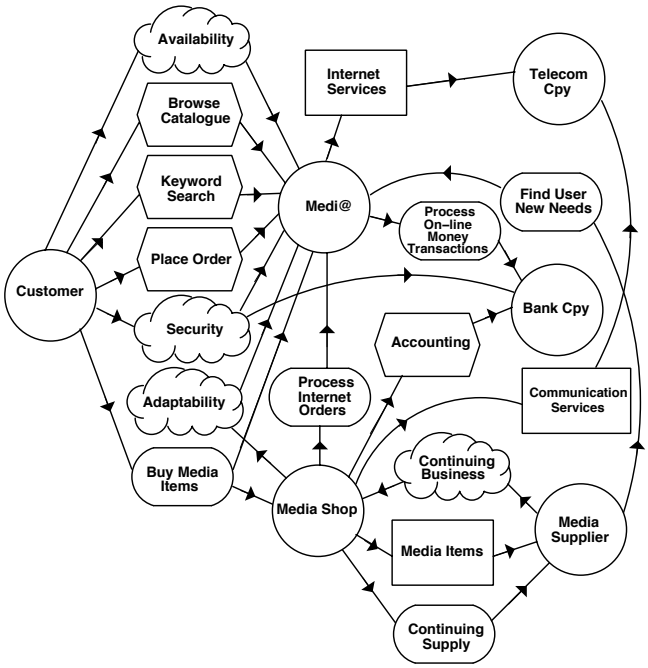

Fig. 3. Actor diagram for the Media Shop

For our example, the *Medi@* system is viewed as a full-fledge actor in the strategic dependency model depicted in Figure 3. With respect to the actors previously identified, *Customer* depends on *Media Shop* to buy media items while *Media Shop* depends on *Customer* to increase market share and make them happy (with *Media Shop* service). *Media Supplier* is expected to supply *Media Shop* with media items in a continuous way since depending on the latter for continuing business. It can also use *Medi@* to determine new needs from customers, such as media items not available in the catalogue while expecting *Media Producer* to provide her with *quality packages*. As indicated earlier, *Media Shop* depends on *Medi@* for processing internet orders and on *Bank Cpy* to process business transactions. *Customer*, in turn, depends on *Medi@* to place orders through the internet, to search the database for keywords, or simply to browse the on-line catalogue. With respect to relevant qualities, *Customer* requires that transaction services be secure and available, while *Media Shop* expects *Medi@* to be easily adaptable (e.g., catalogue enhancing, item database evolution, user interface update, . . .). Finally, *Medi@* relies on internet services provided by *Telecom Cpy* and on secure on-line financial transactions handled by *Bank Cpy*.

Although a strategic dependency model provides hints about why processes are structured in a certain way, it does not sufficiently support the process of suggesting, exploring, and evaluating alternative solutions. As late requirements analysis proceeds, *Medi@* is given additional responsibilities, and ends up as the depender of several dependencies. Moreover, the system is decomposed into several sub-actors which take on some of these responsibilities. This decomposition and responsibility assignment is realized using the same kind of means-ends analysis along with the strategic rationale analysis illustrated in Figure 2. Hence, the analysis in Figure 4 focuses on the system itself, instead of an external stakeholder.

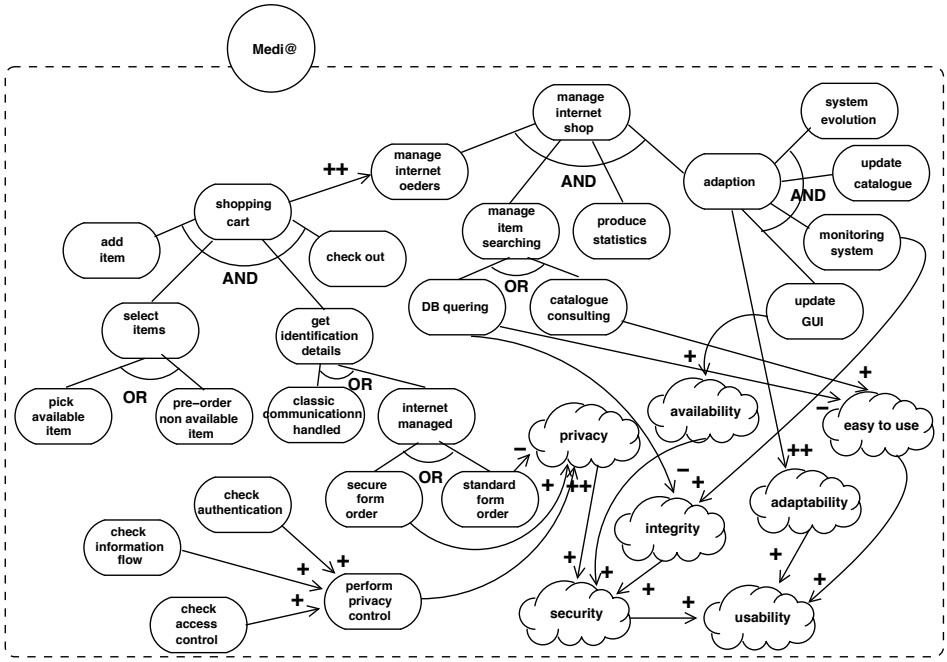

Fig. 4. Actor diagram for the Media Shop focusing on the goal *internet shop managed*

The figures shows the analysis for the goal *manage internet shop*, which solves partially the dependencies with the other actors reported in Figure 3. The goal is firstly refined into goals *manage internet order*, *manage item searching*, *produce statistics* and *adaptation*. To achieve *manage internet order* is used the goal *shopping cart* which is decomposed into subgoals *select item*, *add item*, *check out*, and *get identification details*. These are the main process activities required to design an operational on-line shopping cart [4]. The latter (goal) is achieved either through subgoal *classic communication handled* dealing with phone and fax orders or *internet handled* managing secure or standard form orderings. To allow for the ordering of new items not listed in the catalogue, *select item* is also further refined into two alternative subgoals, one dedicated to

select catalogued items, the other to pre-order unavailable products. To provide sufficient support (++) to the *adaptability* softgoal, *adaptation* is refined into four subgoals dealing with catalogue updates, system evolution, interface updates and system monitoring. The goal *manage item searching* might alternatively be fulfilled through goals *DB querying* or *catalogue consulting* with respect to customers' navigating desiderata, i.e., searching with particular items in mind by using search functions or simply browsing the catalogued products.

The figure reports also the analysis for the softgoals *security* and *usability*. *Security* receive positive contribution from the satisfaction of softgoals *privacy*, *availability*, and *integrity*, whereas *usability* from *adaptability* and *easy to use*. Notice, that *standard form order* gives a negative contribution to the *privacy*. Of course the analysis should include other non-function requirements, but for sake of simplicity we just focus on these two.

3 Goal Models

The concept of goal has been used in different areas of Computer Science since the early days of the discipline. In AI, problem solving and planning systems have used the notion of goal to describe desirable states of the world [13]. More recently, goals have been used in Software Engineering [16, 14] to model early requirements [6] and non-functional requirements [12] for a software system.

Unfortunately, the approaches presented in the literature for modeling and analyzing goals do not work for many domains where goals can't be formally defined, and the relationships among them can't be captured by semantically well-defined relations such as AND/OR ones. For example, in our example the goal "*happy customers*" has no formally defined predicate which prescribes its meaning, though one may want to define necessary conditions for such a goal to be satisfied. Moreover, such a goal may be related to other goals, such as "*improve quality of service*" and "*be friendly*", in the sense that the latter obviously contribute to the satisfaction of the former, but this contribution is partial and qualitative. In other words, if the latter goals are satisfied, they certainly contribute towards the satisfaction of the former goal, but certainly do not guarantee it.

In this section we present the formal model for goals adopted in Tropos, which allows the software engineer to cope with qualitative relationships and inconsistencies among goals. In particular, in following two sections we present the notions of goal graphs and the axiomatic representation of goal relations.

3.1 Goal Graphs

We consider sets of goal nodes G_i and of relations $(G_1,...,G_n) \stackrel{r}{\longmapsto} G$ over them, including the $(n+1)$-ary relations *and*, *or* and the binary relations $+_S$, $-_S$, $+_D$, $-_D$, $++_S$, $--_S$, $++_D$, $--_D$, $+$, $-$, $++$, $--$. We briefly recall the intuitive meaning of these relations:

- $(G_1,...,G_i,...G_n) \stackrel{and}{\longmapsto} G$ means that G is satisfied [resp denied] if all $G_1,...,G_n$ are satisfied [resp. if at least one G_i is denied];

- $(G_1,...,G_i,...G_n) \overset{or}{\longmapsto} G$ means that G is denied [resp satisfied] if all $G_1,...,G_n$ are denied [resp. if at least one G_i is satisfied];
- $G_2 \overset{+_S}{\longmapsto} G_1$ [resp. $G_2 \overset{++_S}{\longmapsto} G_1$] means that if G_2 is satisfied, then there is some [resp. a full] evidence that G_1 is satisfied, but if G_2 is denied, then nothing is said about the denial of G_1;
- $G_2 \overset{-_S}{\longmapsto} G_1$ [resp. $G_2 \overset{--_S}{\longmapsto} G_1$] means that if G_2 is satisfied, then there is some [resp. a full] evidence that G_1 is denied, but if G_2 is denied, then nothing is said about the satisfaction of G_1.
- $G_2 \overset{-_D}{\longmapsto} G_1$ [resp. $G_2 \overset{--_D}{\longmapsto} G_1$] means that if G_2 is denied, then there is some [resp. a full] evidence that G_1 is satisfied, but if G_2 is satisfied, then nothing is said about the denial of G_1;
- $G_2 \overset{+_D}{\longmapsto} G_1$ [resp. $G_2 \overset{++_D}{\longmapsto} G_1$] means that if G_2 is denied, then there is some [resp. a full] evidence that G_1 is denied, but if G_2 is satisfied, then nothing is said about the satisfaction of G_1.

The names $+_S$, $-_S$, $+_D$, $-_D$, $++_S$, $--_S$, $++_D$, $--_D$ have the following intuitive meaning: the "*S*" [resp. "*D*"] symbol denotes the fact that the satisfiability [resp. deniability] value of the source goal is propagated; the "+" [resp. "-"] symbol denotes the fact that the propagation is positive [resp. negative], in the sense that satisfiability propagates to satisfiability [resp. deniability] and deniability propagates to deniability [resp. satisfiability].
The meaning of *or*, $+_D$, $-_D$, $++_D$, $--_D$ is dual w.r.t. *and*, $+_S$, $-_S$, $++_S$, $--_S$ respectively. (By "dual" we mean that we invert satisfiability with deniability.) The relations $+$, $-$, $++$, $--$ are defined such that each $G_2 \overset{r}{\longmapsto} G_1$ is a shorthand for the combination of the two corresponding relationships $G_2 \overset{r_S}{\longmapsto} G_1$ and $G_2 \overset{r_D}{\longmapsto} G_1$. (We call the first kind of relations *symmetric* and the latter two *asymmetric*.) E.g., $G_2 \overset{+}{\longmapsto} G_1$ is a shorthand for the combination of $G_2 \overset{+_S}{\longmapsto} G_1$ and $G_2 \overset{+_D}{\longmapsto} G_1$.

If $(G_1,...,G_n) \overset{r}{\longmapsto} G$ is a goal relation we call $G_1...G_n$ the *source goals* and G the *target goal* of r, and we say that r is an *incoming relation* for G and an *outcoming relation* for G_1,...,G_n. Notice that all relations are *directional*, from the sources to the target goals. We call *boolean relations* the *and* and *or* relations, *partial contribution relations* the $+$ and $-$ relations and their asymmetric versions, *full contribution relations* $++$ and $--$ relations and their asymmetric versions. We call a *root goal* any goal with an incoming boolean relation and no outcoming ones, we call a *leaf goal* any goal with no incoming boolean relations.

We call a *path from G_1 to G_k* a sequence of goals $\pi := G_1, G_2, ..., G_k$ s.t., for every $i \in \{1,...,k-1\}$, G_i and G_{i+1} are respectively a source goal and the target goal of some relation r_i. We call a *loop* a path from a goal to itself. We call a *diamond* a pair of paths $\langle \pi_1, \pi_2 \rangle$ both from G_1 to G_k if π_1 and π_2 contain no common goal except G_1 and G_k.

We call a *goal graph* a pair $\langle G, R \rangle$ where G is a set of goal nodes and R is a set of goal relations, subject to the following restrictions:

each goal has at most one incoming boolean relation; (1)

every loop contains at least one non-boolean relation arc. (2)

In practice, a goal graph can be seen as a forest of *and*/*or* trees whose nodes are connected by contribution relations arcs. Root goals are roots of and/or trees, whilst leaf goals are either leaves or nodes which are not part of the trees.

The presence of contribution relations makes the tasks of formal reasoning on goal graphs much less straightforward than in the case of simple AND/OR graphs. The following factors contribute to complicate the picture.

Asymmetric value propagation. The way satisfiability and deniability values are propagated may be *asymmetric*. For instance, the relation $G_2 \overset{++_D}{\longmapsto} G_1$ suggests that the achievement of the goal G_2 is a necessary but not sufficient condition for achieving the goal G_1. In fact, if G_2 is denied, then there is a full evidence that G_1 is denied, but if G_2 is satisfied, then nothing is said about the satisfaction of G_1.

Partial evidence. The contribution relations described above may propagate only a *partial* evidence about the satisfiability/deniability of the target goals. This means that a formal semantics for goal graphs must provide *partial* satisfiability/deniability values for the goals, and provide rules for propagating both full and partial satisfiability/deniability values through the relations.

Conflicts. Different goals can provide contradictory contributions to the same goals. For instance, if the graph contains $G_1 \overset{+_S}{\longmapsto} G$ and $G_2 \overset{-_S}{\longmapsto} G$ and both G_1 and G_2 are satisfied, then the first relation induces some evidence that G is satisfied, whilst the second induces some evidence that G is denied. We call these situations, *conflicts*. To this extent, it is important to keep track of both satisfiability and deniability values for all goals.

Diamonds. The value of one goal alone can provide contradictory contributions to another goal due to the presence of diamonds. For instance, if the graph contains $(G_1, G_5) \overset{or}{\longmapsto} G_2$, $G_2 \overset{+_S}{\longmapsto} G_4$, $G_1 \overset{--_S}{\longmapsto} G_3$ and $G_3 \overset{+_D}{\longmapsto} G_4$, and both G_1 and G_5 are satisfied, then the satisfiability of G_1 propagates to G_4 through the diamond $\langle G_1G_2G_4, G_1G_3G_4\rangle$, providing both some evidence that G_4 is satisfied (path $G_1G_2G_4$) and some evidence that G_4 is denied (path $G_1G_3G_4$).

Loops. The satisfiability/deniability of one goal can provide a contribution contradicting itself due to the presence of loops. This is the typical situation in models containing negative feedback loops (see, e.g, the real-world example in [8]). For instance, if the graph contains $G_1 \overset{+}{\longmapsto} G_2$ and $G_2 \overset{-}{\longmapsto} G_1$, and if G_1 is satisfiable, then the fact that G_1 is satisfied propagates through G_2 providing some evidence that G_1 is denied.

3.2 Axiomatization of Goal Relationships

Let $G_1, G_2, \ldots$ denote goal labels. We introduce four distinct predicates over goals, FS(G), FD(G) and PS(G), PD(G), meaning respectively that there is (at least) *full* evidence that goal G is satisfied and that G is denied, and that there is at least *partial* evidence that G is satisfied and that G is denied. We also use the proposition $\top$ to represent the (trivially true) statement that there is at least a null evidence that the goal G is satisfied (or denied). Notice that the predicates state that there is *at least* a given level of evidence, because in a goal graph there may be multiple sources of evidence for

Goal	**Invariant Axioms**	
$G:$	$\mathsf{FS}(G) \rightarrow \mathsf{PS}(G), \quad \mathsf{FD}(G) \rightarrow \mathsf{PD}(G)$	(3)
Goal relation	**Relation Axioms**	
$(G_1,...,G_i,...G_n) \overset{and}{\longmapsto} G:$	$(\bigwedge_i \mathsf{FS}(G_i)) \rightarrow \mathsf{FS}(G), \ (\bigwedge_i \mathsf{PS}(G_i)) \rightarrow \mathsf{PS}(G)$	(4)
	$\bigwedge_i (\mathsf{FD}(G_i) \rightarrow \mathsf{FD}(G)), \ \bigwedge_i (\mathsf{PD}(G_i) \rightarrow \mathsf{PD}(G))$	(5)
$(G_1,...,G_i,...G_n) \overset{or}{\longmapsto} G:$	$(\bigwedge_i \mathsf{FD}(G_i)) \rightarrow \mathsf{FD}(G), \ (\bigwedge_i \mathsf{PD}(G_i)) \rightarrow \mathsf{PD}(G)$	(6)
	$\bigwedge_i (\mathsf{FS}(G_i) \rightarrow \mathsf{FS}(G)), \ \bigwedge_i (\mathsf{PS}(G_i) \rightarrow \mathsf{PS}(G))$	(7)
$G_2 \overset{+_S}{\longmapsto} G_1:$	$\mathsf{PS}(G_2) \rightarrow \mathsf{PS}(G_1)$	(8)
$G_2 \overset{-_S}{\longmapsto} G_1:$	$\mathsf{PS}(G_2) \rightarrow \mathsf{PD}(G_1)$	(9)
$G_2 \overset{++_S}{\longmapsto} G_1:$	$\mathsf{FS}(G_2) \rightarrow \mathsf{FS}(G_1), \ \mathsf{PS}(G_2) \rightarrow \mathsf{PS}(G_1)$	(10)
$G_2 \overset{--_S}{\longmapsto} G_1:$	$\mathsf{FS}(G_2) \rightarrow \mathsf{FD}(G_1), \ \mathsf{PS}(G_2) \rightarrow \mathsf{PD}(G_1)$	(11)
$G_2 \overset{+_D}{\longmapsto} G_1:$	$\mathsf{PD}(G_2) \rightarrow \mathsf{PD}(G_1)$	(12)
$G_2 \overset{-_D}{\longmapsto} G_1:$	$\mathsf{PD}(G_2) \rightarrow \mathsf{PS}(G_1)$	(13)
$G_2 \overset{++_D}{\longmapsto} G_1:$	$\mathsf{FD}(G_2) \rightarrow \mathsf{FD}(G_1), \ \mathsf{PD}(G_2) \rightarrow \mathsf{PD}(G_1)$	(14)
$G_2 \overset{--_D}{\longmapsto} G_1:$	$\mathsf{FD}(G_2) \rightarrow \mathsf{FS}(G_1), \ \mathsf{PD}(G_2) \rightarrow \mathsf{PS}(G_1)$	(15)

Fig. 5. Ground axioms for the invariants and the propagation rules.

the satisfaction/denial of a goal. We introduce a total order $\mathsf{FS}(G) \geq \mathsf{PS}(G) \geq \top$ and $\mathsf{FD}(G) \geq \mathsf{PD}(G) \geq \top$, with the intended meaning that $x \geq y$ if and only if $x \rightarrow y$. We call FS, PS, FD and PD the possible *values* for a goal.

We want to allow the deduction of *positive* ground assertions of type FS(G), FD(G), PS(G) and PD(G) over the goal constants of a goal graph. We refer to externally provided assertions as *initial conditions*. To formalize the propagation of satisfiability and deniability evidence through a goal graph $\langle G, R \rangle$, we introduce the axioms described in Figure 5. For instance, (3) state that full satisfiability and deniability imply partial satisfiability and deniability respectively; for an *and* relation, (4) show that the full and partial satisfiability of the target node require respectively the full and partial satisfiability of all the source nodes; for a "$+_S$" relation, (8) show that only the partial satisfiability (but not the full satisfiability) propagates through a "$+_S$" relation. Thus, e.g., an *and* relation propagates the minimum satisfiability value (and the maximum deniability one), while a "$+_S$" relation propagates at most a partial satisfiability value. To this extent, a "$+_S$" relation can be seen as an *and* relation with an unknown partially satisfiable goal. Similar considerations hold for the other relations.

Notice that, combining (3) with (4), and (3) with (8), we have, respectively,

$$(G_2, G_3) \overset{and}{\longmapsto} G_1: \ (\mathsf{FS}(G_2) \wedge \mathsf{PS}(G_3)) \rightarrow \mathsf{PS}(G_1) \tag{16}$$

$$G_2 \overset{+_S}{\longmapsto} G_1: \ \mathsf{FS}(G_2) \rightarrow \mathsf{PS}(G_1). \tag{17}$$

To this extent, henceforth we implicitly assume that axioms (3) are always implicitly applied whenever possible. Thus, e.g., we say that $\mathsf{PS}(G_1)$ is deduced from $\mathsf{FS}(G_2)$ and $\mathsf{PS}(G_3)$ by applying (4) —meaning "applying (3) and then (4)" — or that $\mathsf{PS}(G_1)$ is deduced from $\mathsf{FS}(G_2)$ and $\mathsf{FS}(G_3)$ by applying (4) —meaning "applying (4) and then (3)".

Let $A : (\bigwedge_{i=1}^{n} v_i) \rightarrow v$ be a generic relation axiom for the relation r. We call the values v_i the *prerequisites values* and v the *consequence value* of axiom A, and we say that the values v_i are the *prerequisites* for v through r and that v is the *consequence* of the values v_i through r.

We say that an atomic proposition of the form FS(G), FD(G), PS(G) and PD(G) *holds* if either it is an initial condition or it can be deduced via modus ponens from the initial conditions and the ground axioms of Figure 5. We assume conventionally that $\top$ always holds. Notice that all the formulas in the framework described so far are propositional Horn clauses, so that deciding if a ground assertion holds not only is decidable, but also it can be decided in polynomial time.

A *weak conflict* holds if $(\mathsf{PS}(G) \wedge \mathsf{PD}(G))$, a *medium conflict* holds if either $(\mathsf{FS}(G) \wedge \mathsf{PD}(G))$ or $(\mathsf{PS}(G) \wedge \mathsf{FD}(G))$, while a *strong conflict* holds if $(\mathsf{FS}(G) \wedge \mathsf{FD}(G))$, for some goal G.

4 Reasoning with Goal Models

In this section we present two forms of reasoning with goal models, *forward* and *backward reasoning*, and we show how they are applied in Tropos.

4.1 Forward Reasoning

Given a goal graph and an initial values assignment to some goals, *input goals* from now on (typically leaf goals), forward reasoning focuses on the forward propagation of these initial values to all other goals of the graph according to the rules described in Section 3. Initial values represent the evidence available about the satisfaction and the denial of a specific goal, namely evidence about the state of the goal. Usually such a evidence corresponds to qualitative values of satisfaction or denial of a goal. This is mainly because the evidence is usually provided very vaguely by the stakeholders, during the interviews with the analyst, or elaborated from documents or other available sources of information.

For each goal we consider three values representing the current evidence of satisfiability and deniability of goal: F (full), P (partial), N (none). We admit also conflicting situations in which we have both evidence for satisfaction and denial of a goal. So for instance, we may have that for goal G we have fully (F) evidence for the satisfaction and at the same time partial (P) evidence for denial. This could represent a situation in which we have two difference sources of information that provide conflicting evidence, or a multiple decompositions of goal G, where some decompositions suggest satisfaction of G while others suggest denial.

After the forward propagation of the initial values, the user can look the final values of the goals of interest, *target goals* from now on (typically root goals), and reveal

possible conflicts. In other words, the user observes the effects of the initial values over the goals of interests.

In [8, 9] we have presented the algorithm for the forward propagation and we have shown soundness and completeness with respect to the axiomatization. In the algorithm, to each node G of the graph G we associate two variables $\mathsf{Sat}(G), \mathsf{Den}(G)$ ranging in $\{F, P, N\}$ (full, partial, none) such that $F > P > N$, representing the current evidence of satisfiability and deniability of goal G. For example, $\mathsf{Sat}(G_i) \geq P$ states that there is at least partial evidence that G_i is satisfiable. (To this extent, e.g., $\mathsf{Sat}(G_i) \geq P$ is equivalent to say that $\mathsf{PS}(G_i)$ holds, and so on.) As the goal graph may be cyclic, the process stops when a fixpoint is reached. Starting from assigning an initial set of input values for $\mathsf{Sat}(G_i)$, $\mathsf{Den}(G_i)$ to (a subset of) the goals in G, we propagate the values through the goal relations.

Let us consider for instance the rationale diagram for the *Media Shop* presented in Figure 2. Let us suppose that we have full evidence for the satisfaction of goals *increase sales*, *increase sales price*, and *reduce labour costs*. The result of the forward propagation of these values is that we have full evidence for the satisfaction of the top goal *increase profits* and partial evidence for the denial of softgoals *happy customers* and *improve quality of services*.

As we have seen in Section 2, the Tropos methodology analyzes the requirements of the system-to-be in terms of goal models. Goals basically represent the functional requirements, while the softgoals represent the non-functional requirements of the system. In the goal models, OR relationships are used to model possible alternatives, and the adoption of each of them can have a different impact on the satisfaction of the softgoals. So for instance, in Figure 4 the two alternatives *secure form order* and *standard order form* contribute respectively positively and negatively to the satisfaction of the softgoal *privacy*.

Forward reasoning is adopted in Tropos for evaluating the impact of the adoption of the different alternatives with respect to the softgoals of the system-to-be. Table 1 reports the results of the forward reasoning in four different situations for the goal model presented in Figure 4. The table shows only the results for the goals involved in OR decompositions, the top goal *manage internet shop*, and all the softgoals of the model. For all the other (leaf) goals we assume they have full evidence for satisfaction as initial assignment. For each experiment, the table reports the initial (Init) and final (Fin) values assumed by each goal.

In the first experiment (Exp1) we adopt the goal *DB querying* as the choice to achieve *manage item searching*, the goal *pick available item* to achieve *select items*, and the goal *classic communication handled* to achieve *get identification details*. The result is that the top goal *manage internet shop* is fully satisfied (Sat(...)=F) and all the softgoals are at least partial satisfied (Sat(...)=P), except the softgoal *easy to use* that results partially denied (Den(...)=P). Notice also that this initial assignment produces a conflict for the *integrity* softgoal (Sat(...)=P and Den(...)=P). In the second experiment (Exp2) we adopt the goal *standard form order* instead of the goal *classic communication handled*. This mainly produce the result of moving the conflict from the softgoal *integrity* to the softgoal *privacy*. In the third experiment (Exp3) we decide to *manage item searching* using the *catalogue consulting* goal. The effect of this new assignment

is that softgoal *easy to use* is now partially satisfied, but we have conflicts for softgoals *integrity* and *privacy*. Finally, in the fourth experiment (Exp4) we adopt *secure form order* instead of the *standard form order* goal. This has the effect that now we do not have conflicts and all the softgoals are at least partially satisfied.

Goals	Exp 1				Exp 2				Exp 3				Exp 4			
	Init		Fin		Init		Fin		Init		Fin		Init		Fin	
	S	D	S	D	S	D	S	D	S	D	S	D	S	D	S	D
DB querying	F		F		F		F									
catalogue consulting									F		F		F		F	
pick available item	F		F		F		F		F		F		F		F	
pre-order non available item																
classic communication handled	F		F													
standard form order					F		F		F		F					
secure form order													F		F	
manage internet shop			F				F				F				F	
privacy			P				P	P			P	P			P	
availability			P				P				P				P	
integrity			P	P			P				P	P			P	
usability			P				P				P				P	
adaptability			F				F				F				F	
easy to use				P				P			P				P	
security			P				P				P				P	

Table 1. Evaluating alternatives in the goal model of Figure 4.

Table 1 reports just an example of analysis and it is limited to the simple model of Figure 4. Also in the model we have used only symmetric relationships and we have not distinguished between relations $+_S$ and $+_D$ or $-_S$ and $-_D$. In real-life case studies, the goal models to be analyzed are usually more complex. For instance, in [15], we have presented a goal model with more than hundred goals for the Trentino Public Transportation System, in which non symmetric relationships have been used.

The analysis presented above concerns only a goal model and do not consider the effects of a particular assignment to the goals of other goal models. This kind of analysis is called *intra actor* analysis since it does not involve goal models of other actors. Differently, the *inter actor* analysis extends the boundary of the analysis to the goal models of the other actors. So for instance, we could analyze the effects of an assignment of Table 1 to the softgoals of the goal model shown in Figure 2, such as *happy customers* and *improve quality of services*.

4.2 Backword Reasoning

Backword reasoning focuses on the *backward search* of the possible input values leading to some desired final value, under desired constraints. We set the desired final values of the target goals, and we want to find possible initial assignments to the input goals which would cause the desired final values of the target goals by forward propagation. We may also add some desired constraints, and decide to avoid strong/medium/weak conflicts.

So for instance, in the goal model of Figure 4 we may be interested in finding an assignment without any conflict such that the top goal *manage internet shop* and the softgoal *security* are both fully satisfied.

In [15] we have presented a solution to the backward reasoning reducing the problem to that of propositional satisfiability (SAT) [21]. The boolean variables of the formula Φ to be satisfied are all the values $\mathsf{FS}(G)$, $\mathsf{PS}(G)$, $\mathsf{FD}(G)$, $\mathsf{PD}(G)$ for every goal $G \in \mathcal{G}$, and Φ is written in the form:

$$\Phi := \Phi_{graph} \wedge \Phi_{outval} \wedge \Phi_{backward} \; [\, \wedge \Phi_{optional} \,], \tag{18}$$

where Φ_{graph} encodes the goal graph and the axioms presented in Section 3, Φ_{outval} represents the desired final output values and $\Phi_{backward}$ encodes the backward reasoning. (See [15] for details.) The optional formula $\Phi_{optional}$ allows the user to impose some constraints on the possible values of the goals and to force some desired value(s).

[15] also presents a variant to the approach that allows us to assign a cost value to the satisfaction (or deniability) to the goals and hence find a solution with the minimum overall cost. Thus, for instance, in the goal model of our *Medi@* shop, we may be interested in finding an assignment with the minimal cost able to guarantee the full satisfaction of the top goal *manage internet shop* and the softgoal *security*. This approach is based on a variant of SAT, namely Minimum-Weight Propositional Satisfiability (MW-SAT) [10]. (See again [15] for the details.)

In Tropos the backward reasoning is used to analyze goal models and find the set of goals at the minimum costs that if achieved they can guarantee the achievement of the desired top goals and softgoals. In other words, we find among the alternatives of the goal model those with the minimal cost that allow us to obtain our desired goals.

Goals	Exp 1				Exp 2				Exp 3				Exp 4			
	Init		Fin		Init		Fin		Init		Fin		Init		Fin	
	S	D	S	D	S	D	S	D	S	D	S	D	S	D	S	D
DB querying (3)															F	
catalogue consulting (6)							F				F					
pick available item (2)							F				F				F	
pre-order non available item (7)																
classic communication handled (4)																
standard form order (6)											F					
secure form order (8)							F								F	
manage internet shop	F				F		F		F		F		F		F	
privacy	F				P		P		P		P	P	P		P	
availability	F				P		P		P		P				P	
integrity	F				P		P		P		P	P			P	
usability	F				P		P		P		P				P	
adaptability	F				P		F		P		F				F	
easy to use	F				P		P		P		P					P
security	F				P		P		P		P				P	

Table 2. Backward reasoning with the goal model of Figure 4.

Table 2 presents the results of the backward reasoning in four different situations with the goal model of the *Medi@* shop presented in Figure 4. The cost of each alter-

native goal is reported near its label (e.g., the cost of the *DB querying* is 3.). In the first experiment we try to find an assignment at the minimal cost that allows to obtain the full satisfaction of the top goal *manage internet shop* and all the softgoals. Unfortunately, no solution exists and this is due to the fact that almost all the softgoals receive only (+) and no (++) contributions. In the second experiment we require the full satisfaction of the top goal *manage internet shop* and partial satisfaction of all the softgoals. The solution at the minimum cost results the full satisfaction for *catalogue consulting*, *pick available item* and *secure form order*. In third experiments, we relaxed the constraint of avoiding conflicts and we obtain that now the solution includes the full satisfaction of the goal *standard form order* instead of the goal *secure form order*. Of course, now in the final values of the target goals we have conflicts, and in particular a conflict for the goal *privacy* (Sat(...)=P and Den(...)=P) and the goal *integrity* (Sat(...)=P and Den(...)=P). In the final experiment we imposed only the full satisfaction for the goal *manage internet shop* and the softgoal *privacy*. The solution with no conflicts is reported in table.

Also for backward reasoning the analysis can be extended to the goal models outside the boundary of the single actor. In this case the desired values can be assigned to (soft)goals of different goal models and the final solution will include goals of one or more goal models.

5 Goal Reasoning Tool

Forward and backward reasoning is supported in Tropos by the Goal Reasoning Tool (GR-Tool). Basically, the GR-Tool (Figure 6) is graphical tool in which it is possible to draw the goal models and run the algorithms and tools for forward and backward reasoning.

The algorithms for the forward reasoning, already presented in [8, 9], have been fully developed in java and are embedded in the GR-Tool.

For the backward reasoning we have implemented a tool called GOALSOLVE. The schema of GOALSOLVE is reported in Figure 7 (black arrows). GOALSOLVE takes as input a representation the goal graph, a list of desired final values and, optionally, a list of user desiderata and constraint and a list of goals which have to be considered as input. The user may also activate some flags for switching on the various levels of "avoiding conflicts".

The first component of GOALSOLVE is an encoder that generates the boolean CNF formula Φ as briefly described in the previous section, plus a correspondence table `Table` between goal values and their correspondent boolean variable. Φ is given as input to the SAT solver CHAFF [11], which returns either "UNSAT" if Φ is unsatisfiable, or "SAT" plus a satisfying assignment μ if Φ is satisfiable. Then a decoder uses `Table` to decode back the resulting assignment into the set of goal values.

In order to deal the minimum cost solutions, we have implemented a variant of GOALSOLVE, called GOALMINSOLVE, for the search of the goal values of *minimum cost*. The schema of GOALMINSOLVE is reported in Figure 7 (gray arrows). Unlike GOALSOLVE, GOALMINSOLVE takes as input also a list of integer weights $W(val(G))$ for the goal values, with some default options. The encoder here encodes also the input weight list into a list of weights for the corresponding boolean variables of Φ. Both

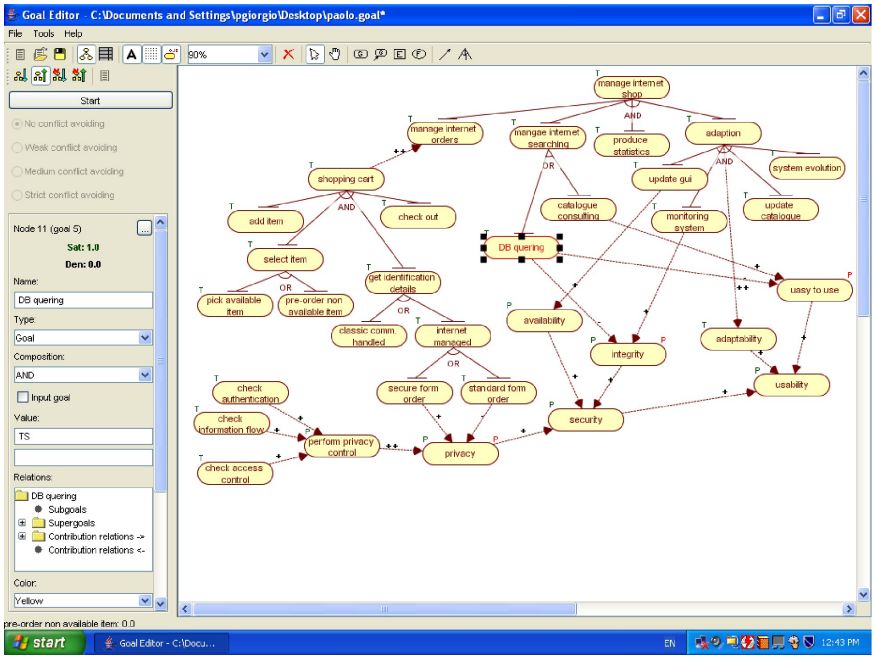

Fig. 6. A snapshot of the GR-Tool

Φ and the list of weights are given as input to the minimum-weight SAT solver MINWEIGHT [10], which returns either "UNSAT" if Φ is unsatisfiable, or "SAT' plus a minimum-weight satisfying assignment μ if Φ is satisfiable. The decoder then works as in GOALSOLVE.

Notice that, in general, there may be many satisfying assignments —up to exponentially many— corresponding to solutions for the problem. In a typical session with GOALSOLVE or GOALMINSOLVE, the user may want to work first with the "avoiding conflicts" flags, starting from the most restrictive down to the least restrictive, until the problem admits solution. (E.g., it often the case that no solution avoiding all conflicts exists, but if one allows for weak and/or medium conflicts a solution exists.) Then, once the level of conflict avoidance is fixed, the user may want to work on refining the solution obtained, by iteratively adding positive and negative values in the list of desiderata and constraints, until a satisfactory solution is found.

6 Conclusion

We have presented a formal framework for reasoning with goal models developed during agent-oriented software development using the Tropos methodology. Our work combines earlier work on an agent-oriented software methodology [1, 2], with a formal goal model defined and studied in [9, 15]. In a parallel effort [7] we have developed a tool, called the T-tool, which supports temporal reasoning of formal Tropos specifica-

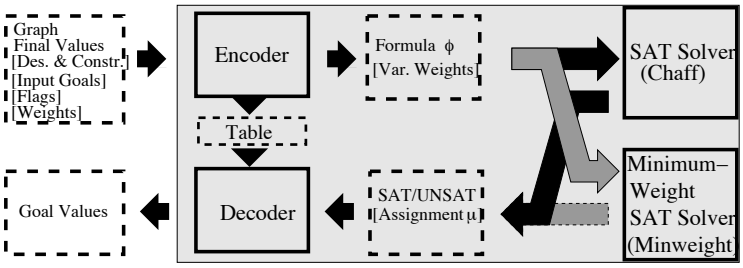

Fig. 7. Schema of GOALSOLVE (black arrows) and GOALMINSOLVE (gray arrows).

tions using a state-of-the-art model checker. Together, these tools can help the developer of Tropos models analyze them and make sure they are consistent with her intuitions. This feedback is essential, especially so when one is dealing with large models developed by a team of designers.

We envision other types of formal analysis for Tropos models. In particular, we propose to work on formal actor dependency models and develop scalable – and usable social analysis techniques that complement the temporal and intentional analysis techniques developed so far.

Acknowledgements

We are grateful to our colleagues working on the Tropos project for their intellectual contributions to this work. In particular, Jaelson Castro (Federal University of Pernambuco, Brazil) and Manuel Kolp (Catholic University of Louvain, Belgium) developed the initial Medi@ shop case study on which our examples are based. The goal reasoning tool was implemented with the contribution of Eleonora Nicchiarelli, Maddalena Garzetti, and Alberto Siena. Paolo Liberatore has provided help with the MINWEIGHT tool.

The Tropos project is partly funded by the Italian Government, the Provincial Authority of Trentino, and the Government of Canada through the Natural Sciences and Engineering Research Council (NSERC). The third author is sponsored in part by the CALCULEMUS! IHP-RTN EC project, code HPRN-CT-2000-00102, by a MIUR COFIN02 project, code 2002097822_003, and by a grant from the Intel Corporation.

References

1. P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An agent-oriented software development methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, May 2004.
2. Jaelson Castro, Manuel Kolp, and John Mylopoulos. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Information Systems*, 27(6):365–389, 2002.

3. P. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 32(3):213–261, 1990.
4. J. Conallen. *Building Web Applications with UML*. Addison-Wesley, 2000.
5. A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1–2):3–50, 1993.
6. A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1–2):3–50, 1993.
7. Ariel Fuxman, Lin Liu, Marco Pistore, Marco Roveri, and John Mylopoulos. Specifying and analyzing early requirements: Some experimental results. In *Proceedings of the 11th IEEE International Requirements Engineering Conference (RE'03)*, page 105. IEEE Computer Society Press, 2003.
8. P. Giorgini, E. Nicchiarelli, J. Mylopoulos, and R.Sebastiani. Reasoning with Goal Models. In *Proc. Int. Conference of Conceptual Modeling – ER'02*, volume 2503 of *LNCS*, Tampere, Finland, October 2002. Springer.
9. P. Giorgini, E. Nicchiarelli, J. Mylopoulos, and R.Sebastiani. Formal Reasoning Techniques for Goal Models. *Journal of Data Semantics*, 1, October 2003. Springer.
10. P. Liberatore. Algorithms and Experiments on Finding Minimal Models. Technical report, DIS, University of Rome "La Sapienza", December 2000. Available at http://www.dis.uniroma1.it/˜liberato/mindp/.
11. M. W. Moskewicz, C. F. Madigan, Y. Z., L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Design Automation Conference*, 2001.
12. J. Mylopoulos, L. Chung, and B. Nixon. Representing and Using Non-Functional Requirements: A Process-Oriented Approach. *IEEE Transactions on Software Engineering*, 6(18):483–497, June 1992.
13. A. Newell and H. Simon. GPS: A Program that Simulates Human Thought. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*. McGraw Hill.
14. C. Rolland. Reasoning with Goals to Engineer Requirements. In *Proceedings 5th International Conference on Enterprise Information Systems*, 2003.
15. R.Sebastiani, P. Giorgini, and J. Mylopoulos. Simple and Minimum-Cost Satisfiability for Goal Models. In *Proc. Int. conference on Advanced Information Systems Engineering, CAISE'04*, volume 3084 of *LNCS*, pages 20–33. Springer, June 2004.
16. A. v. Lamsweerde. Requirements engineering in the year 00: A research perspective. In *Proceedings 22nd International Conference on Software Engineering, Invited Paper, ACM Press*, 2000.
17. E. Yu. Modeling organizations for information systems requirements engineering. In *Proc. of the 1st Int. Symposium on Requirements Engineering, RE'93*, pages 34–41, San Jose, USA, January 1993.
18. E. Yu. Understanding 'why' in software process modeling, analysis and design. In *Proc. of the 16th Int. Conf. on Software Engineering, ICSE'94*, pages 159–168, Sorrento, Italy, May 1994.
19. E. Yu. *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, University of Toronto, Department of Computer Science, 1995.
20. E. Yu and J. Mylopoulos. Using goals, rules, and methods to support reasoning in business process reengineering. *International Journal of Intelligent Systems in Accounting, Finance and Management*, 5(1):1–13, 1996.
21. Lintao Zhang and Sharad Malik. The quest for efficient boolean satisfiability solvers. In *Proc. CAV'02*, number 2404 in LNCS, pages 17–36. Springer, 2002.