# Simple and Minimum-Cost Satisfiability for Goal Models*

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

[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.** Goal models have been used in Computer Science in order to represent software requirements, business objectives and design qualities. In previous work we have presented a formal framework for reasoning with goal models, in a qualitative or quantitative way, and we have introduced an algorithm for forward propagating values through goal models. In this paper we focus on the qualitative framework and we propose a technique and an implemented tool for addressing two much more challenging problems: (1) find an initial assignment of labels to leaf goals which satisfies a desired final status of root goals by upward value propagation, while respecting some given constraints; and (2) find an minimum cost assignment of labels to leaf goals which satisfies root goals. The paper also presents preliminary experimental results on the performance of the tool using the goal graph generated by a case study involving the Public Transportation Service of Trentino (Italy).

## 1 Introduction

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 [9]. For example, a planning system might be given the goal `on(A,B) and on(B,C)`, which describes states where blocks `A`, `B`, `C` form a stack. The planning system can then analyze the goal (e.g., by decomposing it into two subgoals) and find suitable actions that will satisfy it. More recently, goals have been used in Software Engineering to model early requirements [2] and non-functional requirements [8] for a software system. For instance, an early requirement for a library information system might be "Every book request will eventually be fulfilled", while "The new system will be highly reliable" is an example of a non-functional requirement. Goals are also useful for knowledge management systems which focus on strategic knowledge, e.g., "Increase profits", or "Become the largest auto manufacturer in North America" [5]. Goal-oriented requirements engineering has received considerable attention recently, and is nicely motivated and surveyed in [12] and [11]. Given the

* We thank Maddalena Garzetti for sharing with us a version of her goal model for the Trentino Public Transportation Service, and Paolo Liberatore for helping us with the Minweight solver. The first author is sponsored by a MIUR COFIN02 project, code 2002097822_003.

criticality of requirements engineering for information system development, the formal representation and analysis of goals has become an important research problem to be addressed by the CAiSE research community.

Traditional goal analysis consists of decomposing goals into subgoals through an AND- or OR-decomposition. If goal G is AND-decomposed (respectively, OR-decomposed) into subgoals $G_1, G_2, \ldots, G_n$, then all (at least one) of the subgoals must be satisfied for the goal G to be satisfied. Given a goal model consisting of goals and AND/OR relationships among them, and a set of initial labels for some nodes of the graph (S for "satisfied", D for "denied") there is a simple label propagation algorithm that will generate labels for other nodes of the graph [10]. The propagation is carried out from subgoals towards an And/OR-decomposed. This algorithm can be used to determine if the root goals of a goal model are satisfied, given an assignment of labels for some of the leaf goals.

Unfortunately, this simple framework for modeling and analyzing goals won't 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, goals such as "Highly reliable system" have no formally defined predicate which prescribes their 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 "Thoroughly debugged system", "Thoroughly tested system" 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 don't guarantee it. The framework will also not work in situations where there are contradictory contributions to a goal. For instance, we may want to allow for multiple decompositions of a goal G into sets of subgoals, where some decompositions suggest satisfaction of G while others suggest denial.

The objective of this paper is to present further results on a formal model for goals that can cope with qualitative relationships and inconsistencies among goals. In [4] we presented an axiomatization of a qualitative and a quantitative model for goals and proposed sound and complete algorithms for forward reasoning with such models. In particular, given a goal model and labels for some of the goals, our algorithms propagate these labels forward, towards root goals. (If the graph contain loops, this is done until a fixpoint is reached.) Figure 1 illustrates a simple example of a goal graph. The figure shows a single root goal "Protect users" that might associated with a public transit system, AND/OR decomposed several times. The figure also includes some positive qualitative contributions, e.g., "Protect driver health" contributes positively ("+" label) to the goal "Ensure driver capabilities". The algorithm proposed in [4] takes as input labels for some of the lower goals of the model and infers other labels higher up. This is accomplished through propagations from the AND/OR subgoals of a goal to the goal itself, also propagations in the forward direction for qualitative relationships. It is important to note that this algorithm supports forward reasoning only and requires no search.

This paper uses the same formal setting as [4], but addresses a different set of problems. In particular, now we want to know if there is a label assignment for leaf nodes of a goal graph that satisfies/denies all root goals. Assuming that the satisfaction/deniability of every leaf goal may require some unit cost, we have also addressed the problem of

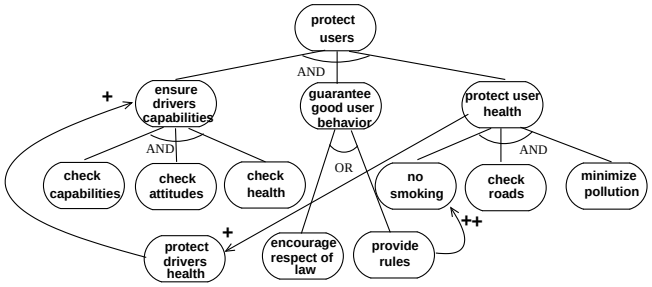


**Fig. 1.** A sample goal graph.

finding a minimum cost label assignment to leaf goals that satisfies/denies all root goals of a goal graph. Both problems are solved by reducing them to the satisfiability (SAT) and minimum-cost satisfiability (minimum-cost SAT) problems for Boolean formulas.

The rest of the paper is structured as follows. Section 2 introduces the formal framework of [4], including a definition of goal graphs and the axiomatization proposed for qualitative goal models. The section also reviews SAT and minimum-cost SAT. Section 3 defines the problem of (simple or minimum-cost) goal satisfiability for a goal graph, and shows how it can be reduced to a (simple and minimum-cost) SAT problem. Section 4 presents two tools named respectively GOALSOLVE and GOALMINSOLVE that solve either goal satisfiability problem. In section 5 we present experimental results on the performance of these tools using the goal graph generated by a case study involving the Public Transportation Service of Trentino (Italy).

## 2 Preliminaries

In this section we recall some preliminary notions which are necessary for a full comprehension of the paper. In Sections 2.1 and 2.2 we recall from [4] and extend a little the notions of goal graphs and the axiomatic representation of goal relations. In Section 2.3 we recall some basic notions about boolean satisfiability and minimum-weight boolean satisfiability.

### 2.1 Goal Graphs

As in [4], 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: $G_2 \stackrel{+_S}{\longmapsto} G_1$ [resp. $G_2 \stackrel{++_S}{\longmapsto} G_1$] 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 $G_1$; $G_2 \stackrel{-_S}{\longmapsto} G_1$ [resp. $G_2 \stackrel{--_S}{\longmapsto} G_1$] 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 $G_1$. The meaning of $+_D$, $-_D$, $++_D$, $--_D$ is dual w.r.t. $+_S$, $-_S$, $++_S$, $--_S$ respectively. (By "dual" we mean that we invert satisfiability with deniability.) The relations $+$, $-$, $++$, $--$ are such that each $G_2 \stackrel{r}{\longmapsto} G_1$ 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*.)

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 $G_1$,...,$G_n$. 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 *goal graph* a pair $\langle \mathcal{G}, \mathcal{R} \rangle$ where $\mathcal{G}$ is a set of goal nodes and $\mathcal{R}$ is a set of goal relations, subject to the following restrictions:

- each goal has at most one incoming boolean relation;
- every loop contains at least one non-boolean relation arc.

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

### 2.2 Axiomatization of Goal Relationships

Let $G_1, G_2, ...$ 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 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 the satisfaction/denial of a goal. We introduce a total order $FS(G) \geq PS(G) \geq \top$ and $FD(G) \geq PD(G) \geq \top$, 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 $\langle \mathcal{G}, \mathcal{R} \rangle$, we introduce the axioms described in Figure 2.

(1) state that full satisfiability and deniability imply partial satisfiability and deniability respectively. For an AND relation, (2) 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, (4) show that only the partial satisfiability (but not the full satisfiability) propagates through a "$+_S$" relation. Thus, an AND relation propagates the minimum satisfiability value (and the maximum deniability one), while a "$+_S$" relation propagates at most a partial satisfiability value.

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 $r$.

| **Goal** | **Invariant Axioms** | |
|---|---|---|
| $G:$ | $FS(G) \rightarrow PS(G), \quad FD(G) \rightarrow PD(G)$ | (1) |
| **Goal relation** | **Relation Axioms** | |
| $(G_1,...,G_i,...G_n) \stackrel{and}{\longmapsto} G:$ | $(\bigwedge_i FS(G_i)) \rightarrow FS(G), \ (\bigwedge_i PS(G_i)) \rightarrow PS(G)$ | (2) |
| | $\bigwedge_i (FD(G_i) \rightarrow FD(G)), \ \bigwedge_i (PD(G_i) \rightarrow PD(G))$ | (3) |
| $G_2 \stackrel{+_S}{\longmapsto} G_1:$ | $PS(G_2) \rightarrow PS(G_1)$ | (4) |
| $G_2 \stackrel{-_S}{\longmapsto} G_1:$ | $PS(G_2) \rightarrow PD(G_1)$ | (5) |
| $G_2 \stackrel{++_S}{\longmapsto} G_1:$ | $FS(G_2) \rightarrow FS(G_1), \ PS(G_2) \rightarrow PS(G_1)$ | (6) |
| $G_2 \stackrel{--_S}{\longmapsto} G_1:$ | $FS(G_2) \rightarrow FD(G_1), \ PS(G_2) \rightarrow PD(G_1)$ | (7) |

**Fig. 2.** Ground axioms for the invariants and the propagation rules in the qualitative reasoning framework. The (*or*), $(+_D)$, $(-_D)$, $(++_D)$, $(--_D)$ cases are dual w.r.t. (*and*), $(+_S)$, $(-_S)$, $(++_S)$, $(--_S)$ respectively.

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 2. We assume conventionally that $\top$ always holds. Notice that all the formulas in our framework 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.

We say that there is a *weak conflict* if $(PS(G) \wedge PD(G))$ holds, a *medium conflict* if either $(FS(G) \wedge PD(G))$ or $(PS(G) \wedge FD(G))$ hold, a *strong conflict* if $(FS(G) \wedge FD(G))$ holds, for some goal $G$.

### 2.3 SAT and Minimum-Cost SAT

Propositional satisfiability (SAT) is the problem of determining whether a boolean formula $\Phi$ admits at least one satisfying truth assignment $\mu$ to its variables $A_i$. In a broad sense, a SAT solver is any procedure that is able to decide such a problem. SAT is an NP-complete problem [1], so that we can reasonably assume that there does not exist any polynomial algorithm able to solve it.

In the last years we have witnessed an impressive advance in the efficiency of SAT techniques, which has brought large previously intractable problems at the reach of state-of-the-art solvers (see [13] for an overview).

The most popular SAT algorithm is DPLL [3], in its many variants, and CHAFF [7] is probably the most efficient DPLL implementation available. In its basic version, DPLL tries to find a satisfying assignment recursively by assigning, at each step, a value to a proposition. The input formula must be previously reduced in conjunctive normal form (CNF)[3]. At each step, if there exists a unit clause, DPLL assigns it to true; otherwise, it chooses a literal $l$ and it tries to find an assignment with $l$ set to true; if it doesn't success, it tries with $l$ set to false. In this way, DPLL performs the deterministic choices first while postponing, as far as possible the branching step, which is the main source of

[3] A boolean formula is in CNF if and only if it is in the form $\bigwedge_i \bigvee_{j_i} l_{j_i}$, $l_{j_i}$ being literals. A disjunction $\bigvee_{j_i} l_{j_i}$ is called *clause*. A one-literal clause is called *unit clause*.

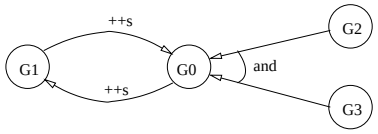


**Fig. 3.** The simple goal graph of Example 1.

exponential blow up. There are several techniques to improve the efficiency of DPLL such as, e.g., backjumping, learning, random restart (again, see [13] for an overview).

A noteworthy variant of SAT is Minimum-Weight Propositional Satisfiability (MW-SAT from now on) [6]. The boolean variables $A_i$ occurring in $\Phi$ are given a positive integer weight $w_i$, and MW-SAT is the problem of determining a truth assignment $\mu$ satisfying $\Phi$ which minimizes the value

$$W(\mu) := \sum_i \{w_i \mid A_i \textit{ is assigned } \top \textit{ by } \mu\}, \tag{8}$$

or stating there is none. In the general case MW-SAT is $\Delta_2^p$-complete problem[4][6], that is, it is much harder than simple SAT. The state-of-the-art solver for MW-SAT is MINWEIGHT [6], which is based on a variant of the DPLL procedure.

## 3 Goal Satisfiability for Goal Graphs

In [4] we focused on the problem of the *forward propagation* of goal values and of the *detection of conflicts*. Given a goal graph, the user assigns some initial values to some goals, *input goals* from now on (typically leaf goals), then these values are forward propagated to all other goals according to the rules described in Section 2. As the goal graph may be cyclic, the process stops when a fixpoint is reached. The user then can look the final values of the goals of interest, *target goals* from now on (typically root goals), and reveal possible conflicts. The whole algorithm is linear in time as it requires no form of search.

In this paper instead we focus on the *backward search* of the possible input values leading to some desired final value, under desired constraints. The user sets the desired final values of the target goals, and the system looks for possible initial assignments to the input goals which would cause the desired final values of the target goals by forward propagation. The user may also add some desired constraints, and decide to avoid strong/medium/weak conflicts.

### 3.1 Input and Target Goals

The notions of "input goal" and "target goal" deserve some more comment. Goal graphs may contain cycles, so that, in principle, it is not obvious a priori which goals are target goals and which are input ones. Although in our experience the boolean relations tend to have a dominant role, so that target goals are typically roots and input goals are typically leaves, the choice is typically left to the user. Nevertheless, the choice is not completely

[4] Broadly speaking, $\Delta_2^p$ is the class of problems requiring a polynomial amounts of calls to a procedure solving an NP problem.

$$FS(G) \rightarrow \begin{pmatrix} \bigwedge_i FS(G_i) & \vee & \text{If } (G_1,...,G_i,...G_n) \overset{and}{\longmapsto} G \\ \bigvee_i FS(G_i) & \vee & \text{If } (G_1,...,G_i,...G_n) \overset{or}{\longmapsto} G \\ FS(G_i) & \vee & \text{For every } R_i\text{: } G_i \overset{++_S}{\longmapsto} G \\ FD(G_i) & & \text{For every } R_i\text{: } G_i \overset{--_D}{\longmapsto} G \end{pmatrix} \quad FD(G) \rightarrow \begin{pmatrix} \bigvee_i FD(G_i) & \vee & \text{If } (G_1,...,G_i,...G_n) \overset{and}{\longmapsto} G \\ \bigwedge_i FD(G_i) & \vee & \text{If } (G_1,...,G_i,...G_n) \overset{or}{\longmapsto} G \\ FD(G_i) & \vee & \text{For every } R_i\text{: } G_i \overset{++_D}{\longmapsto} G \\ FS(G_i) & & \text{For every } R_i\text{: } G_i \overset{--_S}{\longmapsto} G \end{pmatrix} \tag{11}$$

$$PS(G) \rightarrow \begin{pmatrix} \bigwedge_i PS(G_i) & \vee & \text{If } (G_1,...,G_i,...G_n) \overset{and}{\longmapsto} G \\ \bigvee_i PS(G_i) & \vee & \text{If } (G_1,...,G_i,...G_n) \overset{or}{\longmapsto} G \\ PS(G_i) & \vee & \text{For every } R_i\text{: } G_i \overset{++_S}{\longmapsto} G \\ PD(G_i) & \vee & \text{For every } R_i\text{: } G_i \overset{--_D}{\longmapsto} G \\ PS(G_i) & \vee & \text{For every } R_i\text{: } G_i \overset{+_S}{\longmapsto} G \\ PD(G_i) & & \text{For every } R_i\text{: } G_i \overset{-_D}{\longmapsto} G \end{pmatrix} \quad PD(G) \rightarrow \begin{pmatrix} \bigvee_i PD(G_i) & \vee & \text{If } (G_1,...,G_i,...G_n) \overset{and}{\longmapsto} G \\ \bigwedge_i PD(G_i) & \vee & \text{If } (G_1,...,G_i,...G_n) \overset{or}{\longmapsto} G \\ PD(G_i) & \vee & \text{For every } R_i\text{: } G_i \overset{++_D}{\longmapsto} G \\ PS(G_i) & \vee & \text{For every } R_i\text{: } G_i \overset{--_S}{\longmapsto} G \\ PD(G_i) & \vee & \text{For every } R_i\text{: } G_i \overset{+_D}{\longmapsto} G \\ PS(G_i) & & \text{For every } R_i\text{: } G_i \overset{-_S}{\longmapsto} G \end{pmatrix} \tag{12}$$

**Fig. 4.** Axioms for backward propagation (G are non-input goals).

free, as we impose that every path incoming in a target goal must be originated in an input node, that is:

$$\begin{array}{l} \textit{for every target goal } G \textit{ there exists a direct acyclic subgraph} \\ \textit{(DAG) rooted in } G \textit{ whose leaves } G_{i_1},...,G_{i_k} \textit{ are input nodes,} \end{array} \tag{9}$$

so that the value of $G$ derives by forward propagation from those of $G_{i_1},...,G_{i_k}$. An easy-to-verify sufficient condition for (9) is that

$$\textit{all leaf goals are input goals.}^{5} \tag{10}$$

*Example 1.* Consider the simple goal graph of Figure 3, and suppose that $G_0$ is the target goal and $G_2$ and $G_3$ are the input goals. (Notice that $G_0$ and $G_1$ form a loop without input goals.) If we assigned a final value $FS(G_0)$, then by backward search we could have $FS(G_1)$ and then $FS(G_0)$ again. Thus, $FS(G_0)$ could be derived by forward propagation from itself without any input value, which is a nonsense. If instead $G_1$ is an input goal, then by backward search we obtain $FS(G_1)$ or $FS(G_2)$ and $FS(G_3)$, which are suitable initial assignments to the input goals. Notice that $(G_2, G_3) \overset{and}{\longmapsto} G_0$ and $G_1 \overset{++_S}{\longmapsto} G_0$ form a DAG rooted in $G_0$ whose leaves are input nodes. ◇

### 3.2 Basic Formalization

We want to reduce the problem of backward search for input values to that of the satisfiability (SAT) of a boolean formula $\Phi$. The boolean variables of $\Phi$ are all the values $FS(G)$, $PS(G)$, $FD(G)$, $PD(G)$ for every goal $G \in \mathcal{G}$, and $\Phi$ is written in the form:

$$\Phi := \Phi_{graph} \wedge \Phi_{outval} \wedge \Phi_{backward} \left[ \wedge \Phi_{constraints} \wedge \Phi_{conflict} \right], \tag{13}$$

where the conjuncts $\Phi_{graph}$, $\Phi_{outval}$, $\Phi_{backward}$ are explained below and the optional components $\Phi_{constraints}$, and $\Phi_{conflict}$ will be described in Section 3.4.

[5] Recall that, by definition of goal graph, every loop contains at least one leaf goal.

**Encoding the Goal Graph: $\Phi_{graph}$.** The first component $\Phi_{graph}$ is the representation of the goal graph $\langle \mathcal{G}, \mathcal{R} \rangle$, given in the form:

$$\Phi_{graph} := \bigwedge_{G \in \mathcal{G}} Invar\_Ax(G) \wedge \bigwedge_{r \in \mathcal{R}} Rel\_Ax(r), \tag{14}$$

$Invar\_Ax(G)$ being the conjunction of the invariant axioms (1) for the goal $G$ in Figure 2 and $Rel\_Ax(r)$ being the conjunction of the relation axioms in (2)-(7) and their dual ones corresponding to the relation $r$. These axioms encode the forward propagation of values through the relation arcs in the goal graph.

**Representing Desired Final Output Values: $\Phi_{outval}$.** The second component $\Phi_{outval}$ is a representation of the output values the user want to be assigned to the target goal. $\Phi_{outval}$ is written in the form:

$$\Phi_{outval} := \bigwedge_{G \in Target(\mathcal{G})} vs(G) \ \wedge \ \bigwedge_{G \in Target(\mathcal{G})} vd(G) \tag{15}$$

$Target(\mathcal{G})$ being the set of target goals in $\mathcal{G}$ and $vs(G) \in \{\top, PS(G), FS(G)\}$, $vd(G) \in \{\top, PD(G), FD(G)\}$ being the maximum satisfiability and deniability values assigned by the user to the target goal $G$. $\Phi_{outval}$ is a conjunction of unit clauses, which force the desired output values $vs(G)$ and $vd(G)$ to be assigned to $\top$.

**Encoding Backward Reasoning: $\Phi_{backward}$.** The third component $\Phi_{backward}$ encodes the backward search. $\Phi_{backward}$ is written in the form:

$$\Phi_{backward} \ := \ \bigwedge_{G \ \in \ \mathcal{G}/Input(\mathcal{G})} \ \bigwedge_{v(G)} Backward\_Ax(v(G)) \tag{16}$$

$$Backward\_Ax(v(G)) \ := \ v(G) \rightarrow \bigvee_{r \in Incoming(G)} Prereqs(v(G), r) \tag{17}$$

$Input(\mathcal{G})$ being the set of input goals in $\mathcal{G}$, $Incoming(G)$ being the set of relations incoming in $G$, $v(G) \in \{PS(G), FS(G), PD(G), FD(G)\}$, and $Prereqs(v(G), r)$ is a formula which is true if and only if the prerequisites of $v(G)$ through $r$ hold. The list of possible backward propagation axioms $Backward\_Ax(v(G))$ is reported in Figure 4, (11)-(12).

Suppose $G$ is not an input goal. If $v(G)$ holds, then this value must derive from the prerequisite values of some of the incoming relations of $G$. $Prereqs(v(G), r)$ are exactly the conditions which must be verified to apply the corresponding relation axioms (2)-(7) and their dual ones in Figure 2.

### 3.3 Correctness and Completeness

The following theorem states the correctness and completeness of the approach.

**Theorem 1.** *Let $\langle \mathcal{G}, \mathcal{R} \rangle$ be a goal graph. Let $G_{i1}, ..., G_{ik} \in \mathcal{G}$ be the input goals verifying condition (9). Let $G_{f1} ... G_{fn} \in \mathcal{G}$ be the target goals which are assigned the values $vs(G_{f1})$, $vd(G_{f1})$, ... $vs(G_{fn})$, $vd(G_{fn})$ respectively. Let $vs(G_{i1})$, $vd(G_{i1})$, ... $vs(G_{ik})$,$vd(G_{ik})$ be a set of values for the input goals.*

*Then $vs(G_{f1})$, $vd(G_{f1})$, ... $vs(G_{fn})$,$vd(G_{fn})$ can be inferred from $vs(G_{i1})$, $vd(G_{i1})$, ... $vs(G_{ik})$,$vd(G_{ik})$ by means of axioms (1)-(7) if and only if there exists a truth value assignment $\mu$ satisfying (1)-(7), (11)-(12) and the values $vs(G_{i1})$, $vd(G_{i1})$, ... $vs(G_{ik})$ and $vs(G_{f1})$, $vd(G_{f1})$, ..., $vs(G_{fn})$, $vd(G_{fn})$.*

*Proof.* **If:** Assume $\mu$ satisfies $vs(G_{i1})$, $vd(G_{i1})$, ... ,$vs(G_{ik})$ ,$vd(G_{ik})$ and $vs(G_{f1})$, $vd(G_{f1})$, ..., $vs(G_{fn})$, $vd(G_{fn})$ and all axioms (1)-(7) and (11)-(12). By condition (9), for every target goal $G$ there exists a DAG rooted in $G$ whose leaves are all input nodes. We reason on induction of the depth of this DAG. If $G$ is also an input goal, then $G = G_{ik}$ for some $k$, so that $v(G)$ is inferred from $v(G_{ik})$ by a zero-step inference. If $G$ is not an input goal, then there is one backward propagation axiom $A$ in (11)-(12) in the form (17) which is satisfied by $\mu$. As $\mu$ satisfies $v(G)$, $\mu$ satisfies $source(v(G),r)$ for some $r$. Thus $v(G)$ can be inferred from $source(v(G),r)$ by applying axioms (1)-(7). By induction, $source(v(G),r)$ can be inferred from $vs(G_{i1})$, $vd(G_{i1})$, ... $vs(G_{ik})$,$vd(G_{ik})$ by means of axioms (1)-(7). Therefore $v(G)$ can be inferred from $vs(G_{i1})$, $vd(G_{i1})$, ... $vs(G_{ik})$,$vd(G_{ik})$ by means of axioms (1)-(7).

**Only if:** from the hypothesis, $vs(G_{f1})$, $vd(G_{f1})$, ... $vs(G_{fn})$,$vd(G_{ik})$ are inferred from $vs(G_{i1})$, $vd(G_{i1})$, ... $vs(G_{ik})$, $vd(G_{ik})$ by means of axioms (1)-(7). Consider the assignment $\mu$ which assigns $\top$ to all the values which can be inferred from $vs(G_{i1})$, $vd(G_{i1})$, ... $vs(G_{ik})$, $vd(G_{ik})$ by means of axioms (1)-(7) and which assigns $\bot$ to all other values. By construction $\mu$ satisfies $vs(G_{i1})$, $vd(G_{i1})$, ... $vs(G_{ik})$, $vd(G_{ik})$ and $vs(G_{f1})$, $vd(G_{f1})$, ... $vs(G_{fn})$, $vd(G_{fn})$, and the axioms (1)-(7). Now let $A$ be a generic instance of a backward axiom in (11)-(12), and let $v(G)$ be the atom occurring on the left side of $A$, before the "$\rightarrow$". $G$ is not an input goal. If $\mu$ assigns $v(G)$ to $\bot$, then trivially $\mu$ satisfies $A$. If $\mu$ assigns $v(G)$ to $\top$, then, by construction of $\mu$, $v(G)$ is inferred by means of axioms (1)-(7) from some prerequisite values which are assigned $\top$ by $\mu$. Thus, at least one of the disjuncts on the left part of $A$ are satisfied by $\mu$, so that $\mu$ satisfies $A$. Therefore, $\mu$ satisfies all the backward propagation axioms (11)-(12). Q.E.D.

### 3.4 Optional components

We describe here the optional components $\Phi_{constraints}$ and $\Phi_{conflict}$ in (13). They allow the user to impose some constraints on the possible values of the goals or to force some desired value(s).

**Adding User's Constraints and Desiderata** The first optional component $\Phi_{constraints}$ allows the user to express constraint and desiderata on goal values. $\Phi_{constraints}$ is generically written in the form:

$$\Phi_{outval} := \bigwedge_i \bigvee_j lit_{ij}, \tag{18}$$

$lit_{ij} \in \{PS(G), FS(G), PD(G), FD(G), \neg PS(G), \neg FS(G), \neg PD(G), \neg FD(G)\}$, $G \in \mathcal{G}$. A positive unit clause value is used to impose a minimum value to a goal. (E.g., "$PS(G_1)$" means "$G_1$ is at least partially satisfiable", but it might be totally satisfiable.) A negative unit clause value is used to prevent a value to a goal. (E.g., "$\neg FD(G_1)$" means "$G_1$ cannot be fully deniable", but it might be partially deniable.) A disjunction of positive values is used to state an alternative desideratum. (E.g., "$FS(G_1) \vee FS(G_2)$" means "at least one between $G_1$ and $G_2$ must be fully satisfiable".) A disjunction of negative values is used to state a mutual exclusion constraint. (E.g., "$FD(G_1) \vee FD(G_2)$" means "$G_1$ and $G_2$ cannot be both fully deniable", but they can be partially deniable.)

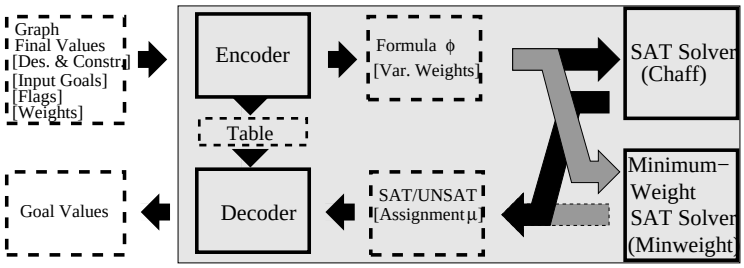


**Fig. 5.** Schema of GOALSOLVE (black arrows) and GOALMINSOLVE (gray arrows).

**Preventing conflicts** The second optional component $\Phi_{conflict}$ allows the user for looking for solutions which do not involve conflicts. Depending whether one wants to avoid (i) only the strong conflicts, (ii) the strong and medium conflicts or (iii) all conflicts, $\Phi_{conflict}$ is encoded respectively as follows:

$$\Phi_{conflict} := \bigwedge_{G \in \mathcal{G}} (\neg FS(G) \vee \neg FD(G)) \tag{19}$$

$$\Phi_{conflict} := \bigwedge_{G \in \mathcal{G}} ((\neg FS(G) \vee \neg PD(G)) \wedge (\neg PS(G) \vee \neg FD(G))) \tag{20}$$

$$\Phi_{conflict} := \bigwedge_{G \in \mathcal{G}} (\neg PS(G) \vee \neg PD(G)). \tag{21}$$

(19) states that $G$ cannot be fully satisfiable and fully deniable; (20) states that $G$ cannot be fully satisfiable and (fully or) partially deniable, and vice versa; (21) states that $G$ cannot be (fully or) partially satisfiable and (fully or) partially deniable. Notice that, by Axioms (1), (21) implies (20) and that (20) implies (19).

It is easy to see that Theorem 1 extends straightforwardly to the case when we have $\Phi_{constraint}$ and $\Phi_{conflict}$ components.

## 4 Solving Simple and Minimum-Cost Goal Satisfiability

Consider a goal graph $\langle \mathcal{G}, \mathcal{R} \rangle$ with input goals $G_{i1}, ..., G_{ik}$ and target goals $G_{f1}, ..., G_{fn}$, and a set of desired final values $vs(G_{f1})$, $vd(G_{f1})$, ... $vs(G_{fn})$, $vd(G_{fn})$ to the target goals (plus possibly a set of user constraints and desiderata), and let $\Phi$ be the formula encoding the problem, as in (13).

Theorem 1 states that (i) if $\Phi$ is unsatisfiable, then no value exists to the input goals from which the desired final values derive by forward propagation (verifying the desiderata and constraints) (ii) if an assignment $\mu$ satisfying $\Phi$ exists, then the maximum values $vs(G_{i1})$, $vd(G_{i1})$, ... $vs(G_{in})$, $vd(G_{ik})$ which $\mu$ assigns to $\top$ are such that the desired final values derive from them by forward propagation (verifying the desiderata and constraints). This allows for reducing the problem of backward search to that of propositional satisfiability.

### 4.1 GOALSOLVE

We have implemented a tool, called GOALSOLVE, for the backward search of the possible input values leading to some desired final value, under desired constraints. The

schema of GOALSOLVE is reported in Figure 5 (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 default choice is that indicated in condition (10), that is, all leaf goals are considered input goals.) 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 $\Phi$ as described in Section 3, plus a correspondence table `Table` between goal values and their correspondent boolean variable. $\Phi$ is given as input to the SAT solver CHAFF [7], which returns either "UNSAT" if $\Phi$ is unsatisfiable, or "SAT" plus a satisfying assignment $\mu$ if $\Phi$ is satisfiable. Then a decoder uses `Table` to decode back the resulting assignment into the set of goal values.

### 4.2 GOALMINSOLVE

In general, the satisfaction/deniability value of (input) goals may have different costs. Thus 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 5 (gray arrows).

Unlike GOALSOLVE, GOALMINSOLVE takes as input also a list of integer weights $W(val(G))$ for the goal values. (The default choice is $W(FS(G)) = (FD(G)) = 2$ and $W(PS(G)) = (PD(G)) = 1$ if $G$ is an input goal, $W(FS(G)) = (FD(G)) = W(PS(G)) = (PD(G)) = 0$ otherwise.) The encoder here encodes also the input weight list into a list of weights for the corresponding boolean variables of $\Phi$. Both $\Phi$ and the list of weights are given as input to the minimum-weight SAT solver MINWEIGHT [6], which returns either "UNSAT" if $\Phi$ is unsatisfiable, or "SAT' plus a minimum-weight satisfying assignment $\mu$ if $\Phi$ is satisfiable. The decoder then works as in GOALSOLVE.

Notice that, in general, there may be plenty many satisfying assignments —up to exponentially many— corresponding so 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 (21) down to the least restrictive (19), 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 —e.g. "$\neg FD(G_1)$", "$FS(G_2)$"— in the list of desiderata and constraints, until a satisfactory solution is found.

## 5 A Case Study

We consider as a case study the strategic objectives for the Public Transportation Service of the region of Trentino (Italy), we have recently analyzed for the CitiMan (Citizen Mobility Modeling and Management) project in collaboration with Centro Ricerche FIAT. As depicted in Figure 6, the main objective of the Trentino government is supply public transport service to the citizens, involving satisfy users, minimize costs per user, improve the quality of life, and protect users.[6]

[6] The goal names are translated from Italian.

The objectives are further analyzed and refined using AND/OR decompositions. For instance, supply public transport service to Trentino citizen is AND-decomposed in guarantee transportation, sell tickets and manage financial budget. In turn sell tickets is AND-decomposed into decide sale strategy, decide sales strategy and avoid frauds, while the goal manage financial budget is AND-decomposed into obtain founds, manage services' costs and manage profits. Likewise, improve transportation services is OR-decomposed into improve old services and supply new services, while protect users is AND-decomposed in guarantee drivers capabilities, guarantee the user behavior, and protect user health.

The graph shows also lateral relationships among goals. For example, the goal protect user health has positive $+$ contribution from the goal monitor pollution, while the goal minimize cost per user has two negative $-$ contributions from goals provide comfort and improve transportation services. Asymmetric relationships are also showed; for instance, the goal satisfy users has a $++_D$ contribution from protect user and guarantee transportation, and a $+_S$ contribution from improve quality of life and again guarantee transportation.

We run a series of experiments with GOALSOLVE and GOALMINSOLVE on the goal graph in Figure 6. Both GOALSOLVE and GOALMINSOLVE have been implemented in C++ and the tests were conducted on a Dell Inspiron 8100 laptop with a Pentium III CPU and 64 MB RAM (OS: GNU/Linux, kernel 2.4.7-10). The tests were intended to demonstrate the practical effectiveness and benefits our approach, and also to check the efficiency of the tools. As for the latter aspect, in all our experiments both GOALSOLVE and GOALMINSOLVE performed in less than one second. (Of course, such performances depend on the goal model and the desiderata we want to obtain.) To provide examples, we briefly describe a few of these experiments. (We report here only the results for GOALMINSOLVE, which are more interesting.)

In a first set of experiments we imposed supply public transport services as target goal and all the leaves nodes except satisfy users and minimize cost per user as input goals, having as a desideratum the full satisfiability of the target goal. We run GOALMINSOLVE with the default settings for the weights.

Imposing the strongest conflict avoidance conditions (21) or (20), GOALMINSOLVE discovered there were no solution, whilst imposing the weakest condition (19) GOALMINSOLVE found the results shown in Table 1 (Exp1). We notice the presence of conflicting values for some top and input goals. In fact, the *FS* value of supply public transport service is backward propagated down to all elements of the and tree rooted there, including invest on the service. This propagates a *PS* value to provide comfort and improve transport services, and hence a *PD* value to minimize costs per user, satisfy users, and hence to the target goal supply public transport service, generating thus a medium conflict. When we tried to eliminate only this conflict by imposing a $\neg PD$ constraint on the target goal, GOALMINSOLVE stated there was no solution anyway.

Thus, GOALMINSOLVE highlighted the fact that, with the goal graph of Figure 6, the full satisfiability of the main target goal cannot be accomplished without generating conflicts.

A second set of experiments showed that it is not possible to obtain the full satisfaction for the main goals of the Trentino Public Transportation System, namely supply public transport service, satisfy users, improve quality of life, and minimize cost per user,

| | Exp1 | | Exp3 | |
|---|---|---|---|---|
| **Top Goals** | **S** | **D** | **S** | **D** |
| supply public transport service | F | P | P | P |
| satisfy users | P | P | P | P |
| improve quality of life | P | | F | |
| minimize cost per user | P | P | P | P |
| protect users | | | F | |

| | Exp1 | | Exp3 | |
|---|---|---|---|---|
| **Input Goals** | **S** | **D** | **S** | **D** |
| guarantee transportation | F | | | |
| obtain funds | F | | P | |
| manage profits | F | | | |
| decide sale strategy | F | | | |
| cover services costs | F | P | P | P |
| invest on the services | F | P | P | P |
| use external funds | F | | P | |
| use profits | P | P | P | P |
| increase profits | P | P | P | P |
| estimate services costs | F | P | P | P |
| evaluate alternative resources | F | | | |
| define profits to reach | F | | | |
| verify tickets | F | | | |
| estimate sale places | F | | | |
| add sale point on the path | F | | | |
| facilitate tickets purchasing | P | P | P | P |
| avoid traffic | P | | P | |
| use special instruments | P | P | P | P |
| respect rules | | | F | |
| check driver health | | | F | |
| check drivers attitudes | | | F | |
| check drivers capabilities | | | F | |
| protect drivers health | | | P | |
| force to respect the law | | | P | |

**Table 1.** Results of the first and the third set of experiments. (We omit the values of non-input goals and of those input goals which were not assigned any value.)

even admitting all kinds of conflicts. This is easily explained with the fact that both satisfy users and minimize cost per user are non-input and have no boolean, $++_S$ and $--_D$ incoming relations, so that there is no way they can be assigned a $FS$ by forward propagation.

In a third set of experiments we have expressed as desiderata the full satisfaction for both goals improve quality of life and protect users, without saying anything about the other goal (in particular about the top goal supply public transpiration service). Also in this experiment it was not possible to have an assignment without conflicts, but we found a solution allowing medium conflicts (option (19)). As shown in Table 1 (Exp3), the assignment to the input goal produces conflicts to the top goals supply public transport services, satisfy users and minimize cost per user which, again, cannot be avoided.

A final set of experiments were carried out with graphs with a bigger number of nodes. The goal of the experiments was to evaluate the performance of our tool with respect to the growth of the dimension of the graphs. We generated such graphs randomly. Starting from the model of Figure 6, we generated new graphs adding randomly new goals and contribution links between goals, and thereby generating new cycles. Of course, since we were only interested on the structure of the graphs, we have not associated to them any semantics.

The results showed that also for bigger graphs (from hundred to two thousand nodes) GOALSOLVE performed in less than one second, whereas GOALMINSOLVE performed relatively well (less than five seconds) for graphs with a number of nodes less than three hundred. This suggests us that our approach can be applied in real life applications where goals models can count more than hundred goals.

## 6 Conclusions and future work

This paper introduces the problem of goal (plain/minimum-cost) satisfiability and proposed a solution by reducing goal satisfiability to the problem of (plain/minimum-cost) satisfiability for Boolean formulas. The solution has been implemented and evaluated with a case study. As illustrated by the case study, the solution makes it possible to answer questions such as "Is there a set of labels for input goals that satisfies all output (root) goals?", and if so, "Which solution is minimum cost?"

The proposed solution only works for qualitative goal models, where goals can be satisfied or denied, possibly partially. We plan to continue this research and extend these results so that they also apply to quantitative models, where one can also talk about goals being satisfied/denied with an attached probability/cost or other numerical measures.

## References

1. S. A. Cook. The complexity of theorem proving procedures. In *3rd Annual ACM Symposium on the Theory of Computation*, pages 151–158, 1971.
2. A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1–2):3–50, 1993.
3. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Journal of the ACM*, 5(7), 1962.
4. P. Giorgini, E. Nicchiarelli, J. Mylopoulos, and R.Sebastiani. Reasoning with Goal Models. In *Proc. Int. Conference of Conceptual Modeling – ER'02*, LNCS, Tampere, Finland, October 2002. Springer.
5. R. Jarvis, G. McArthur, J. Mylopoulos, P. Rodriguez-Gianolli, and S. Zhou. Semantic Models for Knowledge Management. In *Proc. of the Second International Conference on Web Information Systems Engineering (WISE'01)*, 2001.
6. 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/.
7. M. W. Moskewicz, C. F. Madigan, Y. Z., L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Design Automation Conference*, 2001.
8. 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.
9. A. Newell and H. Simon. GPS: A Program that Simulates Human Thought. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*. McGraw Hill.
10. N. Nilsson. *Problem Solving Methods in Artificial Intelligence*. McGraw Hill, 1971.
11. C. Rolland. Reasoning with Goals to Engineer Requirements. In *Proceedings 5th International Conference on Enterprise Information Systems*, 2003.
12. 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.
13. 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.

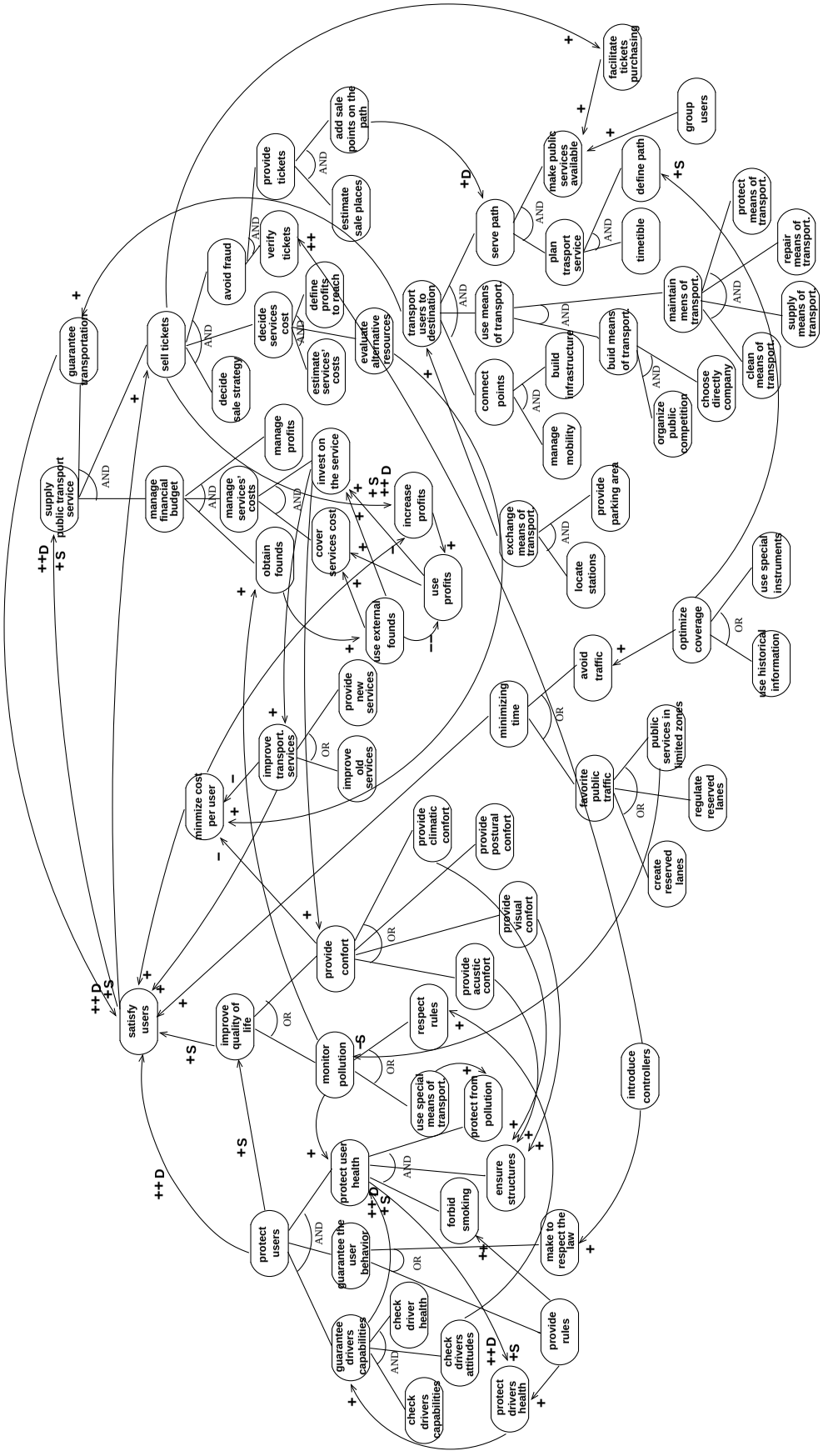


**Fig. 6.** The goal model for Trentino Public Transportation Service.