RULE-BASED FORMALISM AND PREFERENCE REPRESENTATION:
AN EXTENSION OF NEGOPLAN

Gregory E. Kersten & Stan Szpakowicz

a Decision Analysis Laboratory   b Artificial Intelligence Group
School of Business   Department of Computer Science
Carleton University   University of Ottawa
Ottawa, Ontario, Canada   Ottawa, Ontario, Canada

Abstract
NEGOPLAN is a prototype expert system shell for negotiation support and strategic decision support. It uses rule-based formalism for problem representation and modification, and a three-valued valuation function which gives the user strong control over the solution procedure. In this paper, extensions and modification of NEGOPLAN are discussed. Satisficing solutions can be obtained by applying a procedure which uses bounds and conditional bounds. The flexibility of a decision alternative reflects its ability to remain feasible when the environment changes. Two fact ranking methods are proposed to enable the choice of alternatives with a required flexibility. Preferences not accounted for by flexibility may be described by weighed ranks.

Keywords: Decision support, decision flexibility, rule-based formalisms, solution procedures, preferences, strategic decision making.

1. Introduction
NEGOPLAN is a prototype expert system shell for negotiation support. It uses a rule-based formalism for problem representation and modification, and it has been designed for the support of one party in two-party negotiations. NEGOPLAN, its previous version RUNE, and the experiments with it have been described elsewhere (see Kersten et al 1988; Matwin et al 1987; Michalowski et al 1987; Szpakowicz et al 1987). Here, we present generalizations and extensions of NEGOPLAN.

We consider first of all the "static" aspects of NEGOPLAN, deemphasizing one of the most powerful features of the system: its ability to support strategic decision processes (Kersten 1988). We discuss an extension of NEGOPLAN which enables the decision-maker to analyze the impact of the reactions of the environment (such as the opponent in negotiations) on the decision and on the decision problem itself. The decision-maker must often change his decision when the active environment responds to his actions (Young 1975). It is important to consider how much flexibility he has if an alternative is applied, this is to say, for what environment changes it remains feasible and satisficing.

Next, we show how preferences can be represented during the preparation of a compromise proposal - a negotiating position - by the party supported. This preparation is a decision process itself, in that one attempts to choose the most preferable position. The preparation process and the entire decision process, which may comprise more than one preparatory step, differ primarily with regard to the impact of the environment on the decision-maker and the problem. During a preparatory step the environment is considered static; its changes and responses are received and recorded by the decision-maker between the preparatory steps.

2. Problem representation
We assume that a decision problem is decomposable into smaller subproblems and that it can be expressed in terms of logically interconnected statements about the domain. Some statements are inferred from others, or explained by them. There is one statement which does not explain any other statement, called the principal goal. Every remaining statement explains at least one other statement. Statements which are not explained at all correspond to non-decomposable, directly solvable subproblems; we call such statements facts.

Statements correspond in our model to problem variables; in this paper we shall refer to them as variables. The principal goal is a variable, and so are facts. Variables other than facts are called goals. A variable which is used to explain another variable is an antecedent, and the variable explained is a consequent. This terminology will become clear in a while.

We denote by \( \alpha = [\alpha_0, \alpha_1, \ldots, \alpha_n] \) the sequence of all variables that occur in problem \( \alpha_0 \) is by convention the principal goal. Let us denote \[ \{0,...,n\} \] as \( I \). The index set of facts is denoted as \( I_f \) and the index set of goals as \( I_g \). The index set of facts and goals is \( I_f \cup I_g = I \).

The permissible values of a variable are taken from the set

\[ A = \{ \text{true, false, any} \} \]
The value *any* corresponds to the situation when a variable may be either *true* or *false* without influencing the values of its consequents. We assume that the required value of the principal goal \( \alpha_0 \) is *true*, and that there are at least two variables in \( I_g \), so that \( \alpha_0 \) is not a fact.

The value of a goal \( \alpha_i \) in \( I_g \), depends on the values of variables \( \{ \alpha_j \} \); \( j \), that explain \( \alpha_i \). This dependence can be expressed as a *rule* which intuitively says that \( \alpha_i \) is true (i.e. achieved) if some combination of \( \alpha_j \) is true.

**Definition 1.** A rule is an expression
\[
\alpha_i \rightarrow r_i [ \alpha_j ] \quad j,
\]
with antecedents \( \{ \alpha_j \} \); \( j \), and consequent \( \alpha_i \); \( r_i \) is an expression constructed from variables by operators \( OR, AND \) and \( NOT \) as follows:

- (i) \( r_i [ \alpha_j ] \) has one or more *disjuncts* separated by \( OR \); disjuncts correspond to different explanations (decompositions) of \( \alpha_i \).
- (ii) Each disjunct has one or more *conjuncts* separated by \( AND \).
- (iii) Each conjunct is some \( \alpha_j \) or a negation of \( \alpha_j \), written \( NOT \alpha_j \).

For example, the rule
\[
V1 \quad V2 \quad V3 \ OR \quad V4 \ AND \ NOT \ V5 \ AND \ V6
\]
has two disjuncts, \( V2 \) \& \( V3 \) and \( V4 \) \& \( NOT \ V5 \) \& \( V6 \), the second of which has three conjuncts, \( V4 \), \( NOT \ V5 \), \( V6 \).

**Definition 2.** A set of *flat* rules corresponding to rule
\[
\alpha_i \rightarrow r_i [ \alpha_j ] \quad j, \quad i \ IN \ I_g
\]
is
\[
R_i = [ \alpha_i \rightarrow r_i [ \alpha_j ] \quad j, \quad k \rightarrow K_i ]
\]
where \( K_i \) are the indices of all disjuncts, and \( J_{ik} \) are the indices of variables in the \( k \)-th disjunct.

For example, the set of flat rules for rule (1) is
\[
R_1 = [ V1 \quad V2 \ AND \ V3, \ V1 \quad V4 \ AND \ NOT \ V5 \ AND \ V6 ].
\]

**Definition 3.** Rules in a set of flat rules are *competitive* if \( | K_i | > 1 \).

For example, rules in (2) are competitive; it is sufficient to use only one of the two rules to determine the value of \( V1 \).

**Definition 4.** The representation of problem \( P \) is the pair
\[
P = < \alpha, R >
\]
where \( R = \{ R_i : i \ IN \ I_g \} \).

As an example, consider the representation of a problem with the principal goal \( A1 \), depicted graphically in Fig. 1a:
\[
R = \{ R_1, R_2, R_3, R_4, R_5 \} =
\]
\[
= \{ [ A1, A2, A1, NOT A3 ],
\]
\[
A2 NOT A4 AND A5 \},
\]
\[
A3 A8, A3 A9 AND NOT A10 \},
\]
\[
A4 A7 AND A6 \},
\]
\[
A5 NOT A6, A5 A8 \}.
\]

(We do not list \( \alpha \) because it is clear which variables are facts and which are goals. By convention, names of variables in problems illustrated by figures begin with \( A \), whereas variables discussed in other examples begin with \( V \).)

The problem representation \( P \) must have the following properties:

**Property 1.** Each goal other than the principal goal is explained by variables from the lower level and, in turn, explains one or more variables from the higher level. This means that for any fact \( \alpha_i \) in \( I_g \) there is at least one sequence of goals \( \alpha_i, \alpha_j, \ldots, \alpha_k \) such that \( \alpha_i \) is the antecedent of \( \alpha_j \), \( \alpha_j \) is the antecedent of \( \alpha_k \) etc, and finally \( \alpha_k \) is the antecedent of \( \alpha_0 \). Moreover, any goal \( \alpha_k \) in \( I_g \) belongs to at least one such a path from a fact to \( \alpha_0 \).

**Property 2.** Circular relations are not permitted; goal \( \alpha_i \ IN \ I_g \) may appear only once in any path \( \alpha_i, \alpha_j, \ldots, \alpha_k \).

Our considerations so far have been purely formal. We can write a rule or build a complete problem representation without thinking about the values of variables. Naturally, the notation strongly suggests an interpretation of rules in terms of truth values. For instance, the first rule in (2) can be interpreted as saying that \( V1 \) is true if \( V2 \) and \( V3 \) are true. The second rule says that \( V1 \) is true if \( V4 \) and \( V6 \) are true and \( V5 \) is false.

Notice that nothing certain can be said about \( V1 \)'s value if the value of \( V2 \) or \( V3 \) is *any*. In general, the value of a fact is obtained from the domain of decision-making, and we shall formalize this in our model as a valuation.

**Definition 5.** A *valuation* is a function from \( \alpha \) to \( A \):
\[
\forall(\alpha) \ A \ for \ i \ IN \ I_g.
\]
The values of facts are always imposed by circumstances independent of the structure of the problem, but the values of goals can be either similarly imposed or inferred from the rules. To infer the value of a goal α, it is necessary to check compatibility with the values of facts and with the rules, we take the rule

\[ \alpha \models r_i (α_j) \]

We then infer (by the same process) the values of all α_j and apply r_i to v(α_j).

For example, assume that the values of variables V1, V2, V3, V4, V5, V6 from rule (1) under valuation v’ are

<table>
<thead>
<tr>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
<th>V6</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

We treat AND, OR, NOT as the usual logical operations, we find that indeed

\[ v'(V1) = true \]

For valuation v” described by

<table>
<thead>
<tr>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
<th>V6</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>any</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

we have \[ v'(V1) = false \] while from the first competitive rule in (2) it follows that the value of V1 should be true rather than false.

**Definition 6.** A valuation is feasible if the inferred values of goals do not conflict with those defined in the valuation. The conflicting values are true and false, whereas any agrees with every value.

Valuation v’ in the previous example is feasible, but v” is not.

**Definition 7.** A solution of a problem represented by \( P = < α, R > \) is a pair \( s = < P, v, r > \), where v is a feasible valuation such that \( v(α_0) = true \).

With each problem solution, we associate a collection of values of facts

\[ \{ v(α_i) \}_i \]

Among \( v(α_i) \), some must be true or false in order for \( α_0 \) to be true. These facts do not allow flexibility, and consequently (although slightly imprecisely) we call them inflexible facts. The remaining facts are called flexible. They may have any of three values in A, but the value any is most general.

**Definition 8.** Given a valuation v, the set of facts \( [ α_i ]_i \) is partitioned into two sets: flexible and inflexible facts, denoted respectively as \( α^{flex} \) and \( α^{inflex} = [ α_i ]_i \) \( \alpha^{inflex} \)

(we shall use the notation \( α^{inflex} \) and \( α^{flex} \) if v will be important). \( α^{inflex} \) consists of all those facts which must have a fixed value (either true or false) for the principal goal to be true.

It is sufficient to find \( α^{inflex} \) to have solved the problem. Any two solutions which coincide on \( α^{inflex} \) can be treated as equivalent, in that both make the principal goal \( α_0 \) true and both represent decompositions of \( α_0 \) into the same relevant facts.

**Definition 9.** Solutions s1 = \( < P, v^1 > \) and s2 = \( < P, v^2 > \) are fact-equivalent if

\[ α^{inflex} \]

and

\[ v^1(α_i) = v^2(α_i) \] for each \( α_i \) \( α^{inflex} \).

Among the equivalence class of all mutually fact-equivalent solutions there is one most general, which subsumes all others. This follows from the generality of the value any: if the value of an antecedent is any, its consequent remains true when we change the antecedent’s value to true or to false.

**Definition 10.** A most general solution of a problem has a valuation v such that

\[ v(α_i) = any \] if and only if \( α_i \) \( α^{flex} \).

The notion of fact-equivalence allows us to reduce each problem to a minimal form, in which all flexible facts would be absent: by adding them back and giving them the value any we obtain the most general solution of .

**Definition 11.** Problem \( P_{min} = < α_{min}, R_{min} > \) is a minimal equivalent of problem \( P = < α, R > \) if

\[ (i) \quad α_0 = α_{min,0} \]

\[ (ii) \quad I_g \quad I_{min,g} \; \text{and} \; R \quad R_{min} \; \text{for} \; i \; I_{min,g} \]

\[ (iii) \quad \text{there exist solutions} \; s_{min} = < P_{min}, v_{min} > \; \text{and} \; s = < P, v > \; \text{such that} \]

\[ α_{min}^{inflex} = α^{inflex} \; \text{and} \; α_{min}^{flex} = . \]

In other words, \( P_{min} \) has only the facts necessary for solving P.

**Definition 12.** A solution for which \( α^{flex} = \) is called a canonical solution.

3. Standard representation and candidate problems

Decision problem of the class we consider here are likely to have many competitive rules, defining many different ways of achieving the principal goal. This is the main source of non-
determinism in the problem representation: choosing one competitive rule, one decomposition, 
excludes others that may be more interesting or more promising from the point of view of the 
domain.

It is convenient to introduce the notion of a standard form of problem representation, in which 
no goal is negated. The standard representation is logically equivalent to the original one but it 
facilitates considering the effects of choices.

**Definition 13.** Let problem \( P = < \alpha, R > \). The standard representation of \( P \) 
is the pair

\[ P_{\text{std}} = < \alpha_{\text{std}}, R_{\text{std}} > \]

obtained from \( P \) by Procedure 1 described below.

**Procedure 1.** The procedure consists in repeatedly finding negated goals in \( R \) and “pushing 
negation down” by applying DeMorgan’s law of propositional logic and the law of double 
negation. We shall not go into the details, but we shall discuss the procedure in general terms 
and give a large example.

If the negated goal has competitive rules, its negation will be a conjunction of subgoals. For 
instance, with the rules

\[ [ V1 \ V2, V1 \ \text{NOT} \ V3 ], [ V4 \ \text{NOT} \ V1 ] \]

the removal of the negated goal \( \text{NOT} \ V1 \) will give

\[ [ V4 \ \text{NOT} \ V2 \ \text{AND} \ V3 ] \].

If the goal has one non-competitive rule with two or more conjuncts, its negation will be a 
disjunction. For instance, with the rules

\[ [ V5 \ \text{NOT} \ V6 \ \text{AND} \ V7 ], [ V8 \ \text{NOT} \ V5 ] \]

we shall remove the negated goal \( \text{NOT} \ V5 \) to get

\[ [ V8 \ V6, V8 \ \text{NOT} \ V7 ] \].

Such rewriting may produce rules with nested right-hand sides, not allowed by Definition 1. 
Nested “rules” are rewritten by introducing new goals whenever necessary; we call such goals 
anonymous, because they are never shown to the user. For instance, with the rules

\[ [ V9 \ \text{V10 AND NOT} \ V11 ], [ V12 \ \text{NOT} \ V9 \ \text{AND} \ V13 ] \]

removing \( \text{NOT} \ V9 \) results in the nested “rule”

\[ [ V12 \ \text{NOT} \ V10 \ \text{OR} \ V11 ] \].

We must introduce the anonymous goal \( a14 \) to get

\[ [ V12 \ a14 \ \text{AND} \ V13 ], [ a14 \ \text{NOT} \ V10, a14 \ V11 ] \]

(notice that \( a14 \) can be semantically treated as \( \text{non-V9} \), the opposite of \( V9 \)).

The procedure terminates when facts remain the only negated variables in the current 
representation. **End of Procedure 1.**

We now show how Procedure 1 transforms problem (3). This is done in three steps, and the 
final result is illustrated in Fig. 1b.

The first negated goal to be processed may be \( \text{NOT} \ A3 \). We rewrite rule

\[ A1 \ \text{NOT} \ A3 \]

into \( A1 \ \text{NOT} (A8 \ \text{OR} (A9 \ \text{AND} \ A10)) \)

into \( A1 \ \text{NOT} A8 \ \text{AND} \ \text{NOT} (A9 \ \text{AND} \ A10), \)

and we introduce the anonymous goal \( a11 \) to represent the inner conjunction. The whole 
representation now is as follows:

\[ [ [ A1 \ A2, A1 \ \text{NOT} A8 \ \text{AND} \ a11 ], [ a11 \ A9 \ \text{AND} \ A10 ], [ A2 \ \text{NOT} A4 \ \text{AND} A5 ], (4) [ A4 \ A7 \ \text{AND} A6 ], [ A5 \ \text{NOT} A6, A5 \ A8 ] ] \].

In the second step, we remove the negated goal \( \text{NOT} \ A4 \) by rewriting

\[ A2 \ \text{NOT} A4 \ \text{AND} A5 \]

into \( A2 \ \text{NOT} (A7 \ \text{AND} A6) \) and A5

into \( A2 \ \text{NOT} (A7 \ \text{OR} \ A6) \) and A5, 

and we create goal \( a12 \), meaning \( \text{non-V4} \), to get the next version of the representation:

\[ [ [ A1 \ A2, A1 \ \text{NOT} A8 \ \text{AND} \ a11 ], [ a11 \ A9 \ \text{AND} \ \text{NOT} A10 ], [ A2 \ a12 \ \text{AND} A5 ], (5) [ a12 \ \text{NOT} A7, a12 \ \text{NOT} A6 ], [ A5 \ \text{NOT} A6, A5 \ A8 ] ] \].

Finally, we must get rid of the negation over \( a11 \). This requires us to introduce goal \( a13 \) to 
stand for \( \text{non-a11} \). The final result of the procedure is the following problem representation:

\[ [ [ A1 \ A2, A1 \ \text{NOT} A8 \ \text{AND} a13 ], [ a13 \ \text{NOT} A9, a13 \ A10 ], [ A2 \ a12 \ \text{AND} A5 ], (6) ] \].
The standard representation \( P_{std} \) of a decision problem can be directly used to find solutions of. By systematically performing all the relevant choices inherent in \( P_{std} \) we obtain all the possible minimal equivalents of \( P_{std} \) (Definition 11), which in turn determine all the possible most general solutions (Definition 10).

**Definition 14.** A candidate problem is any problem \( P_{cand} \) obtained from \( P_{std} \) by applying Procedure 2 described below. Procedure 2 reduces each set of competitive rules to exactly one, arbitrarily chosen rule.

**Procedure 2.** Let \( P_{std} = \langle \alpha_{std}, R_{std} \rangle \) where

\[
R_{std} = \{ \{ \alpha_{std} : i \} \mid k \}
\]

The procedure consists in constructing \( R_{cand} \) (and implicitly also \( \alpha_{cand} \)). We begin by choosing rule

\[
\alpha_0, R_{cand} = \alpha_0, r_{cand}(\alpha_j, J_{cand}) \cdot k \cdot K_t \] and removing all the other rules from the rule set for the principal goal \( \alpha_0 \). The resulting one-rule set goes into the initially empty \( R_{cand} \).

Now, we consider in turn all \( \alpha_m \) from the set \( \{ \alpha_m : m \cdot J_{cand} \leftrightarrow I_g \} \), this is to say, all goals in the rule we have chosen for \( \alpha_0 \). We choose exactly one rule from each set

\[
R_{std,m} = \{ \alpha_m \cdot r_{std}(\alpha_j, J_{std}) \cdot k \cdot K_m \}
\]

and put the resulting one-rule sets in \( R_{cand} \). This gives us the next level of \( P_{cand} \). We repeat the same operation with all goals from the right-hand sides until the only remaining variables are facts. *End of Procedure 2.*

Note that goals which occur only in a rule that has not been chosen will not go into \( \alpha_{cand} \). This means that \( I_{std,g} \) \( I_{cand,g} \) (see Definition 11).

As an example, here are two candidate problem obtained from problem (6), both illustrated in Fig. 1c:

\[
\{ \{ a_{12} \ \text{NOT} \ A_7, \ a_{12} \ \text{NOT} \ A_6 \}, \ \{ A_5 \ \text{NOT} \ A_6, \ A_5 \ A_8 \ \} \}.
\]

\[
\{ \{ A_1 \ A_2 \}, \ \{ A_{12} \ \text{AND} \ A_5 \}, \ \{ a_{12} \ \text{NOT} \ A_6 \}, \ \{ A_5 \ \text{NOT} \ A_6 \} \} ;
\]

\[
\{ \{ A_1 \ \text{NOT} \ A_8 \ \text{AND} \ a_{13} \}, \ \{ a_{13} \ A_{10} \} \}.
\]

**Observation 1.** A candidate problem has at most one solution. Indeed, its rules contain no negated goals and no disjunctions (OR), so that the value of the principal goal \( \alpha_0 \) is exactly the conjunction of fact values. This means that only one valuation for \( [\alpha_i]_i \) \( I_f \) makes \( \alpha_0 \) true: unnegated facts must be true, negated facts false.

**Observation 2.** A candidate problem \( P_{cand} = \langle \alpha_{cand}, R_{cand} \rangle \) has a feasible solution if no fact from \( \alpha_{cand} \) appears both negated and unnegated in rules from \( R_{cand} \). This follows from Definition 5.

**Observation 3.** A solvable candidate problem \( P_{cand} = \langle \alpha_{cand}, R_{cand} \rangle \) has the canonical solution \( s_{cand} = \langle P_{cand}, v_{cand} > \) such that \( \alpha_{cand}^{\text{flex}} = \) and for \( i \) \( I_f \)

\[
v_{cand}(\alpha_i) = \text{true} \quad \text{if and only if} \quad \alpha_i \text{ is never negated in } R_{cand};
\]

\[
v_{cand}(\alpha_i) = \text{false} \quad \text{if and only if} \quad \alpha_i \text{ is always negated in } R_{cand}.
\]

**Definition 15.** A solution of problem \( P_{std} \) induced by a canonical solution \( s_{cand} = \langle P_{cand}, v_{cand} > \) such that

\[
\alpha_{cand}^{\text{indflex}} = \alpha_{cand}^{\text{candflex}}
\]

\( \alpha_{cand}^{\text{indflex}} \) contains all remaining facts from \( \alpha_{std} \)

\( v_{cand}(\alpha_i) = v_{cand}(\alpha_i) \) for \( \alpha_i \), \( \alpha_{cand}^{\text{indflex}} \)

\( v_{cand}(\alpha_i) = \text{true} \) for \( \alpha_i \), \( \alpha_{cand}^{\text{indflex}} \).

From Procedure 2 it follows that \( \alpha_0 = \alpha_{cand,0} I_{std,g} I_{cand,g} \).

\( R_{std,i} \ R_{cand,i} \) \( i \) \( I_{cand,g} \). From Observation 3 we have \( \alpha_{cand}^{\text{flex}} = \). This gives us Corollary 1.

**Corollary 1.** A candidate problem \( P_{cand} \) obtained from \( P_{std} \) is a minimal equivalent of \( P_{std} \).

From Definitions 10 and 15, and from Corollary 1 we get Corollary 2.

**Corollary 2.** An induced solution of \( P_{std} \) is its most general solution.

Both these corollaries summarize our procedure for solving a decision problem.

**Procedure 3.**

1. Transform \( P \)’s representation \( P \) into \( P_{std} \) using Procedure 1.
2. Construct all \( P_{cand} \) using Procedure 2, and for each of them
   2a. Build the canonical solution \( s_{cand} \).
   2b. Extend \( s_{cand} \) onto the induced solution \( s_{ind} \).

**End of Procedure 3.**

The following lemma shows that Procedure 3 allows us to find all solutions of.
Lemma 1. The set of all most general solutions of \( \alpha_0 \) is identical with the set of solutions found from the standard representation. In other words, the transformation of \( P \) into \( P_{std} \) is a sound operation.

Outline of a proof. Procedure 1 preserves the logical properties of all rules (it only uses DeMorgan's laws and the law of double negation), and it does not change the set of facts. This means that any solution of \( P_{std} \) solves \( P \). Conversely, any solution of \( P \) can be reconstructed by considering \( \alpha_0 \)'s antecedents, their antecedents, and so on down to facts. Properties 1 and 2 guarantee that such a reconstruction will terminate; without going into the details, let us observe that each choice encountered will correspond to a choice involved in Procedure 2.

4. Preferences and rule-based formalisms

NEGOPLAN analyzes the decision problem to predict the values of facts or goals. It is unrealistic to assume that a valuation will be fully determined during the search for solutions, completely imposed by the rules. Typically, some fact values will already be known, and an acceptable solution will have to take these values into account. The solution method actually employed should be selected with regard to the proportion of facts known beforehand and facts whose values can be dictated by the method. Three cases are possible which require different algorithms.

Case 1. All facts have known values. We use the truth tables and reason forward to infer the value of the principal goal \( \alpha_0 \). If its valuation is \( \text{true} \), the given values form a solution. A modification of this situation is when not all facts are known, but those which are known all belong to \( \alpha_{\text{inflex}} \).

Case 2. No facts are known and we want to determine their values to get \( v(\alpha_0) = \text{true} \). In this situation we reason backward: assuming \( v(\alpha_0) = \text{true} \) we determine the required values of its antecedents, antecedents of these antecedents, until the values of facts are determined (see the outline of the proof at the end of the previous section).

Case 3. Some facts are known, but not enough to infer the value of the principal goal. In this case forward reasoning is applied to the known facts and continued until the value of some goal cannot be inferred because of the lack of a fact value. The goal whose values have been found are treated as if they were known facts, and we proceed as in Case 2.

Known fact values may be taken from the environment, or may be the result of the user's preferences. In the simplest situation, the user restricts some of the facts to be \( \text{true} \) or \( \text{false} \) regardless of the structure of the problem, and he wants to check whether these restrictions allow a solution. Needless to say, this is just one of many ways of expressing preferences.

There are two basic approaches to preference modelling (Jacquet-Lagreze 1982): (i) aggregation or direct assessment of outcomes (criteria, goals) yield by alternative decisions leading to overall analytical preference, and (ii) disaggregation or indirect assessment of wholistic intuitive preference leading to comparison of alternative decisions.

The first approach is often incorporated in a solution procedure. It changes or adapts the procedure so it is “progressive”, i.e. the alternative which is currently determined is at least not worse than the one determined previously. Simon (1969) introduced in General Problem Solver preferences based on the difference between the current problem representation and the goal - the final problem representation. GPS chooses an alternative so that the differences are reduced. This approach is also used in many of the multiple criteria decision models (MCDM) with known goal values and the distance function, or with a known utility function or lexicographic ordering of criteria.

The second approach is often used in conjunction with the first approach. In a number of MCDM interactive procedures determination of an efficient solution does not terminate the procedure. The user is then asked to compare the current solution with the one determined in the previous iteration. The effect of this comparison is used to determine the next solution. Here, we have modification of the search mechanism based on the wholistic assessment of alternatives.

The efficacy, flexibility and modularity of the rule-based formalism (Holland et al 1986) make it possible to introduce both conditional and unconditional preferences: those which act on the solution procedure and those which act on the problem representation. Research on logic of preferences dates, as Rescher (1966) has put it, back to Aristotle who discussed the concept of preferability as "the worthier of choice". Semantic foundations for the first-order logic of preference has been developed by Houthakker (1965), Rescher (1966) and von Wright (1963), among others. More recently, Roubens and Vincke (1985) discuss approaches to preference modelling using preference, indifference and incomparability relations.

In quantitative decision models the solution strategy is meant as the choice from the currently available alternatives or the choice of the direction of search for new alternatives. This strategy is well defined, for example, through the decision maker's evaluation of outcomes, or through an evaluation function (such as a utility function). In the interactive methods, the use of the evaluation function may be directed by the input from the user. In rule-based representations of decision problems there is no evaluation function. The problem is how to direct the search for solutions by choosing one competitive rule from among many. There are several possible solutions of this problem. For example, the bucket brigade algorithm (Holland et al 1986) determines the winner in the competition by revising the strength or appropriateness of the competitive rules.
This and similar approaches are suitable when, for the decision problem at hand, a number of alternatives have been analyzed. The outcome of this analysis is the revision (in systems with learning capabilities, also the modification) of the rules, and the choice of those best applicable. Problems modelled with NEGOPLAN are often unique and the user may not be willing to directly analyze and evaluate a large number of alternatives. It is important to introduce mechanisms which can reduce the number of alternatives presented to the user.

5. Bounds and conditional bounds

If the values of certain facts or goals are restricted, the overall number of possible solutions of problem may drop substantially. The user is given a complete list of facts in , and he defines bounds by setting the required values of facts. This set-option is now available in NEGOPLAN where any means "no presetting"; we are experimenting with an interpretation of any more consistent with the approach taken here. In Figure 2 we show how this feature is used during the analysis of negotiations with a hostage-taker. Conditional-set-option is a version of the set-option which requires introducing additional rules; we call these rules conditional bounds. Conditional-set-option makes the required value of a fact depend on the values of other facts.

Definition 16. A conditional bound is written as

\[ \alpha_i = \alpha_i^* \text{ AND } \ldots \text{ AND } \alpha_j = \alpha_j^* \]

where \( i_1, \ldots, i_s, j \in I_c \),

\( \alpha_i^* \) is the current value of \( \alpha_{im} \) for \( m = 1, \ldots, s \),

\( \alpha_j^* \) is the required value of \( \alpha_j \).

Defining fact values on the basis of other facts values may decrease the number of competitive rules by rendering useless those rules whose consequents evaluate to false. It may also cause conflicting or ambiguous situations. Four examples of such situations are shown in Fig. 3.

Figure 3a illustrates the impact of fact value restrictions. The user may not expect that by introducing the restricting rule

\[ A4 = true \text{ and } A7 = false \]  \hspace{1cm} (9)

the problem becomes not solvable (to simplify the notation we omit the valuation function \( v \), and write \( A4 = true \) instead of \( v(A4) = true \)).

The situation in Fig. 3b illustrates an ambiguity in the treatment of the value any. Assume that fact \( A4 \) is true at some point during the search for a solution. Now the value of \( A7 \) should be set; two values are possible, false as dictated by rule (9) or any which subsumes false. The question is: should we choose false and consider \( A7 \) inflexible, or is it better to make it flexible in spite of rule (9)? The choice rests with the user, since restrictions of fact values reflect his flexibility preferences. Figure 3c shows a related problem with any. If any is the value required, do we allow true or false as special cases? If the condition in a restricting rule mentions any, is true or false acceptable, too? The real problem here is that the mechanisms currently available in NEGOPLAN are not yet sensitive enough with respect to any.

The example in Fig. 3d illustrates reevaluation of facts. At some moment, the value of \( A7 \) is known to be true, and then we find that \( A4 = true \). Should we return to \( A7 \) and change its value using rule (9)? This may lead to infinite loops. This suggests a slightly finer version of conditional bounds.

Definition 17. A conditional bound in extended form is written as

\[ \alpha_i = \alpha_i^* \text{ AND } \ldots \text{ AND } \alpha_j = \alpha_j^* \alpha^* \hspace{1cm} \alpha_j = \alpha_j^* \]

where \( i_1, \ldots, i_s, j \in I_c \),

\( \alpha_j^* \) is the current value of \( \alpha_{jm} \) for \( m = 1, \ldots, s \),

\( \alpha_j^* \) is the required value of \( \alpha_j \).

We have discussed the set-option and the conditional-set-option for facts. Both options can be extended onto goals after slightly changing the procedures. Setting the value of goal \( \alpha_i \) requires solving the subproblem whose principal goal is \( \alpha_i \). If there is a solution which gives \( \alpha_i \) the requested value, we simply continue. Otherwise the user is notified that this value cannot be achieved; he must then change or disable the rule that restricts goal \( \alpha_i \).

6. Flexibility and the choice of competitive rules

6.1. Assigning ranks to competitive rules

The choice of a competitive rule from the set of flat rules

\[ R_j = \{ \{ \alpha_i, \{ \alpha_j, j \in J_k \} : k \in K_j \} \} \]

(10)

(where \( j \) is the index of antecedents in a rule defining \( \alpha_i \)) depends on the antecedents of \( \alpha_i \).

The choice may be accomplished in three ways:

(a) the importance of a problem variable to the principal goal is related to the number of participations as antecedents in different competitive rules;
(b) the importance is related to the number of intervening variables between the given problem variable and the principal goal;
(c) the user provides directly or indirectly preferences for all or some of problem variables, and these preferences are used to determine the importance of each competitive rule.
To choose a competitive rule $\alpha_i$, we assign ranks to all the elements of $R_i$. But it is not clear what could be the rationale for determining ranks. In quantitative models, ranks refer to the direct or indirect importance of decision variables or criteria. By assigning such ranks, we expect to achieve better (for example, higher) objective performance levels. When a variable has only three values (true, false, any), it should be easier to define preferences among so few values. The set-option can be profitably used if the preference is pre-emptive, this is to say, if the user strongly prefers one value. Otherwise, he must be able to indicate his more balanced preferences, which should be considered when constructing the problem $P_{\text{can}}$ and with a unique solution. This means that preferences are to be taken into account in the choice from competitive rules.

We have distinguished two types of facts: flexible and inflexible. We assume that the user prefers a solution with as many flexible variables as possible. This is because the more such values in an alternative, the more flexibility the user has in (i) selecting fact values, (ii) responding to challenges posed by the environment, and (iii) applying the alternative. As we said in Section 2, the value $\text{any}$ can be changed to true or false but the user knows that reversing it will not affect the value of the principal goal. The negotiator may present facts with values $\text{any}$ as, say, true. When the opponent demands changes in some of the facts, the negotiator may agree, fully aware that these changes will not influence adversely the feasibility of the proposed alternative.

This discussion shows that the user's preferences may be related to a number of flexible facts: the user prefers solution with more flexible facts over those with fewer flexible facts. This means that we should choose a competitive rule which requires, directly or indirectly, the minimal number of inflexible facts.

**Definition 18.** Let $S$ be the set of all most general solutions of problem $P_{\text{ad}}$. Let solution $s \in S$ introduce the set of flexible facts $\alpha_f^{\text{flex}}$. Solution $s'$ is most flexible if:

$$|\alpha_f^{\text{flex}}| = \max_s |\alpha_f^{\text{flex}}|.$$ 

Because the number of facts is given and equal to $|I_f|$, a most flexible solution has a minimal number of inflexible facts.

In the following discussion we assume the standard representation. We assign a weight to each variable, and then we take the rank of a rule to be the weight of its consequent. Each fact gets weight 1. The consequent of a flat rule has the weight equal to the sum of weights of its antecedents. To determine the weight of the common consequent of a set of competitive rules, choose a rule with the righthand side that gives the minimal number.

**Definition 19.** The weight $w$, given to all variables in $\alpha$, is a function defined recursively as follows:

$$w(\alpha_i) = \begin{cases} 1 & \text{for } i \in I_f, \\ \min_k \left\{ \sum_{j \in I_f} w(\alpha_j) \right\} & \text{for } i \notin I_f. \end{cases}$$

An example of rank assignment is given in Fig. 4a. All facts have weight 1. A7 has weight 2 because $w(A7) = w(A9) + w(A10) = 2$, and similarly $w(A8) = 2$, $w(A6) = 3$. The weight of A5 is 1 because A5 A8 or A5 A11, and $w(A5) = \min\{2, 1\}$. Continuing, we find that the weight of A1 is 2; see the figure for the values of other goals.

After determining the ranks of all rules, we create a candidate problem by always choosing a competitive rule with a minimal rank. The effect of ranks on Procedure 2 is to make each choice informed rather than random.

Consider once more the example in Fig. 4a. Beginning with the principal goal, we have a choice between A2 with rank 4 and A3 with rank 2, and we choose A3. Now, it is a choice between A6 and A8. We choose A8 because $w(A8) < w(A6)$. Goal A8 is true if its two antecedents are true, so to obtain a solution we must stipulate $A11 = \text{true}$ and $A12 = \text{true}$. The candidate problems are indicated by heavy lines in Fig. 4.

We had to set the values of two facts, and we left five facts flexible: A9, A10, A13, A14, A15. This happens to be the most flexible solution for the problem presented in Fig. 4, but in general, as we show in the next subsection, Procedure 2 with ranks does not always find a solution with the minimal number of inflexible facts.

The user may have preferences in determining flexible facts, it is more important that some facts are flexible than the others. If he can assign weights indicating the relative importance of facts, these weights may be used to modify ranks. A high weight indicates that the fact is meant to be flexible. We replace the first part of Definition 19 with

$$w(\alpha_i) = \omega_i \quad \text{for } i \in I_f,$$

where $\omega_i$ is the weight assigned to the fact $\alpha_i$.

An example is given in Fig. 4b. The user has said that facts A12 and A15 are the most important and he gave them weight 10; facts A10, A13 and A14 have been assigned weight 5, and facts A9 and A11 weight 1. We obtain a solution shown in Fig. 4b, with four flexible facts: A12, A13, A14, A15.

### 6.2. Lower and upper bounds for rank assignment

Rank assignment according to the formulae from Definition 19, together with the rule of choice of a competitive rule - see (10) - do not ensure that the obtained solution has a maximal number of flexible facts (or, equivalently, a minimal number of inflexible facts). An example of such a situation is given in Fig. 5a. Applying formulae from Definition 19, we obtained two
flexible facts A10 and A11. If, instead of antecedent A3, we choose antecedent A2 of A1 we shall obtain three flexible facts A13, A14 and A15 (see Fig. 5b).

When we assign ranks to competitive rules, we count facts for each rule separately. In case of multiple occurrences of a fact, it is counted more than once. In the example from Fig. 5a facts A8, A10 and A11 were counted respectively two, two and three times. Counting A8 twice does not make rank assignment incorrect, because A8 is an antecedent in two rules of which only one can be chosen. Multiple count of A10 and A11 does change the ranks of the competitive rules that explain goals A4, A5 and A7: the rule A1 A2 cannot outrank rule A1 A3, even though A2 depends actually on three facts A8, A10, A11, whereas A3 on facts A8, A13, A14, A15. We have w(A2) = 6 because six occurrences of facts determine the value of A2. Multiple count artificially increases the number of facts necessary to determine the value of α0.

Observation 4. Rank assignment given by Definition 19 gives the upper bound w(α0) for the minimum number of inflexible facts.

If we take into account the number of occurrences of facts and goals in the standard representation, the formulae of Definition 19 will be modified to give Definition 20.

Definition 20. The weight with occurrences u, given to all variables in α, is a function defined recursively as follows:

\[ u(α_i) = \begin{cases} 1 & \text{for } i \notin I_f \\ \min_k K_i (\mathcal{S}_j \cup I_k) u(α_j)/α_i & \text{for } i \in I_g \end{cases} \]

where \( α_i \) is the number of occurrences of \( α_i \) in \( R \).

Observation 5. Rank assignment by Definition 20 gives the lower bound \( u(α_0) \) for the minimum number of inflexible facts.

Note that we weigh both facts and goals by the number of their occurrences. Thus, in the problem illustrated by Fig. 5a and 5b, the minimal number of inflexible facts is 3, because the lower bound equals \( u(α_1) = 17/6 \). Note also that by Definition 20 rank assignment does not guarantee finding the most flexible solution. In Fig. 5c we present a situation when ranks weighed with occurrences indicate a solution with inflexible facts A11 and A10, but the solution with only one inflexible fact A8 is ignored.

From Observations 4 and 5 we find the bounds on the number \( |α_{S^F}^{\text{flex}}| \) of flexible facts in a most flexible solution:

\[ |I_f| - w(α_0) \leq |α_{S^F}^{\text{flex}}| \leq |I_f| - w(α_0). \]

It is possible to obtain a rank defining the exact number of inflexible facts by verifying fact occurrences in rank calculation. This requires a substantial computational effort. The calculation of w-ranks and a-ranks, given by Definitions 19 and 20, is efficient and it may be sufficient from the user's perspective. If the user wants to apply the set-option before rank assignment, we propose non-Archimedean (pre-emptive) weights: facts chosen in the set-option are assigned weight \( M_1 \), and the remaining facts weight \( M_2 << M_1 \).

7. Conclusion

Quantitative functions alone cannot adequately represent many decision problems. Strategic decision problems (such as the choice of a new technology, a merger of companies, diversification), negotiations and group decision-making are examples of decision processes where the qualitative aspects significantly influence the decision. Such decision problems depend both on time and context. They are ill-structured, and not all relevant information may be available. The structure of the problem, as well as the decision-maker's preferences are likely to change with new information.

Traditional approaches to problems of this kind have aimed at creating a closed model and aggregating variables (objectives) into an evaluation function. Although powerful, these approaches are not flexible enough. They impose high demand on information requirements, and they are difficult to use when information and preferences change. Changes introduced by the decision-maker are often difficult to explain. To quote Davis (1987: 19), "mathematical methods, as powerful as they are, and as appropriate as they are for a number of problems, simply are not and never were intended to be models of how people think".

Rule-based formalisms allow us to express the way we reason. They provide means to structure and solve context-dependent problems with incomplete information. They also allow rapid prototyping. By this we mean (i) quick development of the representation of a problem with available information, and (ii) changes in this representation dictated by new information. Such changes can be automated by the use of restructuring rules of the kind available in NEGOPLAN. Restructuring rules operate on rules from the set \( R \), which describe the static aspects of the problem: they can modify \( R \)-rules or create new ones.

The approach taken in NEGOPLAN has been expanded in two directions. First, the meaning of the value \( α_0 \) has been clarified and consistently introduced into all NEGOPLAN activities discussed here. This value is related to the flexibility of decisions which we consider as one of the important aspects of decision-making. The notion of flexibility discussed in this paper may be compared with postoptimal analysis in mathematical programming, but here it can be introduced into the solution procedure. Moreover, our approach gives information about flexible variables which, contrary to non-basic variables in a mathematical program, can assume any possible value and the solution still remains feasible.
The second extension is related to the user's control during the search for alternatives. Assuming the importance of flexibility of decisions, we provide means of determining solutions with a given number of flexible facts. The user can also be told the upper bound for the number of flexible facts; this makes it possible to compare the determined solution with a hypothetical most flexible solution. The user can also compute weighted ranks to introduce preferences both over flexible and inflexible facts.

The logic introduced here is not, strictly speaking, a classical three-valued logic. It would be interesting and practically important to expand some three-valued logic, such as Kleene's logic (1952) with the value \textit{any}. One can add to the set \( A = \{ \text{true}, \text{false}, \text{any} \} \) a new value \textit{unknown} which describes the situation when the user cannot specify the valuation of a fact.

\textbf{References}


