LOGICAL TESTING FOR RULE-BASE MANAGEMENT

Fred GLOVER

Center for Applied Artificial Intelligence, Graduate School of Business, University of Colorado, Boulder, Colorado 80309-0419, USA

and

Harvey J. GREENBERG

Mathematics Department, University of Colorado at Denver, Denver, Colorado 80202, USA

Abstract

We present a procedure to logically reduce simple implications that comprise the rule-base of an expert system. Our method uses topological sorting on a digraph representation that detects logical inconsistency and circular reasoning in linear-time. Then, the sort order provides an efficient method to detect and eliminate forced values and redundant rules. We consider additional diagnostic aids for the rule-base manager, notably how to range the number of propositions that could be true and how to consolodate the rule-base. We than show how the simple case may be extended to logically test a general rule-base with a decomposition principle.

Keywords

Expert systems, rule-base management, logical testing, combinatorial optimization, precedence graphs, computational logic.

1. Introduction

We study the problems of logical inconsistency and circular reasoning, as cited by Bramer [4], associated with a rule-base. We also provide detection of redundant rules – that is, whose removal from the rule-base does not alter the set of feasible truth values – and propositions whose truth values are forced.

The first main result is a linear-time algorithm that detects inconsistency, if present, in a rule-base with only simple implications, and provides a diagnostic in the form of a minimal cycle for the expert to rectify the logical error. If the rule-base is consistent, the algorithm detects and eliminates circular reasoning during its first phase. Then, phase two detects and eliminates redundant rules and forced propositions, resulting in a logically irreducible rule-base, whereby there exist truth assignments for which each (remaining) proposition may be assigned either value (true or false).

Underlying the method is the use of topological sorting introduced by Kahn [12] and enhanced by Knuth [13]. Elimination of forced propositions is done efficiently due to the sort order obtained in phase one; elimination of redundant rules is done at the same time in a manner similar to that of Valdes et al. [17].

If the rule-base is consistent, optimization models are considered for the derived, logically irreducible rule-base. Our second main result then shows the applicability of max-flow/min-cut algorithms, for which there are efficient labelling methods (see [5] and [6]). This equivalence was shown in another context by Rhys [15] and Balinski [12], and the underlying combinatorial optimization problem was addressed more generally by Picard [14].

When the requisite structure for using a simple labelling algorithm is absent, special search techniques are still available with linear programming by augmenting the algebraic equivalent with special linear inequalities derived by Johnson and Padberg [11]. The derived inequalities have special significance in their own right, showing the expert implied rules that may otherwise not be easily discernable.

We next consider another problem of rule-base management: how to consolidate rules for efficient representation and processing. The third main result is a methodology, drawing from graph inversion [7,8].

Our fourth main result is to show how the logical testing method for simple rule-base extends to the general case by a decomposition principle. This is particularly well suited for a parallel computer architecture, such as the Intel Hypercube.

We shall use only propositional logic, but, as described by Jeroslow [10], the methods apply to predicate calculus under certain assumptions that generally apply to expert systems.

The rest of this paper is divided into six sections. Section 2 presents the basic concepts we shall use in deriving the main results. Section 3 presents the linear-time algorithm for the case of a simple rule-base. Section 4 presents a family of optimization models, aimed at helping the expert understand the scope of the rule-base, and presents the max-flow/min-cut equivalence. Section 5 introduces the "consolidation problem", which pertains to finding a logically equivalent rule-base with the fewest number of rules. Section 6 presents a decomposition principle for the general rule-base. Finally, section 7 presents avenues for further research.

2. Basic concepts

2.1. GRAPH-THEORETIC DEFINITIONS

This section reviews the standard graph-theoretic concepts we shall use (see Harary et al. [9]). A (finite) *digraph* $D = [V, A]$ consists of a finite set of vertices V

and a finite set of arcs A. Each arc is an ordered pair of vertices $\langle v, w\rangle$, which is oriented from v to w. We call v a *predecessor* of w, and w a *successor* of v. The sets of predecessors and successors of a vertex v are denoted, respectively, by:

$$P(v) = [u \in V : \langle u, v\rangle \in A] \quad \text{and} \quad S(v) = [w \in V : \langle v, w\rangle \in A].$$

Moreover, for $W \subseteq V$, we denote:

$$P(W) = U[P(w) : w \in W] \quad \text{and} \quad S(W) = U[S(w) : w \in W].$$

If S is any finite set, $\#S$ denotes its cardinality. A *source* is a vertex v for which $\#P(v) = 0$; a *sink* is a vertex v for which $\#S(v) = 0$.

A digraph $D' = [V', A']$ is a subgraph of digraph $D = [V, A]$, denoted $D' \subseteq D$, if $V' \subseteq V$ and $A' \subseteq A$.

A (directed) *path* of length p is a sequence of vertices $\langle v_0, v_1, \ldots, v_p\rangle$ such that $v_{j+1} \in S(v_j)$ for $j = 0, \ldots, p-1$. We say that the path contains the vertices $v_0, \ldots, v_p$ and the arcs $\langle v_0, v_1\rangle, \ldots, \langle v_{p-1}, v_p\rangle$; we also say that an arc $\langle v_j, v_{j+1}\rangle$ is traversed. A *semipath* is a sequence of vertices $\langle v_0, v_1, \ldots, v_p\rangle$ such that $v_{j+1} \in S(v_j) \cup P(v_j)$ for $j = 0, \ldots, p-1$. If $v_{j+1} \in S(v_j)$, we say the arc $\langle v_j, v_{j+1}\rangle$ is traversed in a forward direction; otherwise, if $v_{j+1} \in P(v_j)$, we say $\langle v_j, v_{j+1}\rangle$ is traversed in a backward direction. A path (or semipath) is *simple* if no vertex is contained twice. If $v_0 = v_p$ and $p > 1$, the path (semipath) is a *cycle* (respectively, *semicycle*). A digraph is acyclic if it contains no cycles. Vertex w is a descendant of vertex v, denoted $v \Rightarrow w$, if there exists a path with $v_0 = v$ and $v_p = w$.

An *implication graph* is a digraph with $2n$ vertices with the following properties:

(1) vertices are numbered so that for each vertex v there is a unique vertex $-v$, called its complement;

(2) if $\langle v, w\rangle$ is an arc, so is $\langle -w, -v\rangle$, called its contrapositive.

We also say v and $-v$ are a complementary pair and that $\langle v, w\rangle$ and $\langle -w, -v\rangle$ are contrapositive arcs.

For $W \subseteq V$, $\widetilde{W}$ denotes its set of complements: $\widetilde{W} = [-w : w \in W]$; and, for $B \subseteq A$, $\widetilde{B}$ denotes its set of contrapositive arcs: $\widetilde{B} = [\langle -w, -v\rangle : \langle v, w\rangle \in B]$. If G is an implication graph and $H = [W, B] \subseteq G$, the complement of H is $\widetilde{H} = [\widetilde{W}, \widetilde{B}]$. It follows that for any $H \subseteq G$, $H \cup \widetilde{H}$ is an implication graph.

If $\langle v_0, v_1, \ldots, v_p\rangle$ is a path, its contrapositive path is $\langle -v_p\ -v_{p-1}, \ldots, -v_0\rangle$. A cycle C is *contradictory* if it contains a complementary pair of vertices; otherwise, its vertices comprise an *equivalence class* E, and we say C generates E. (Note the contrapositive cycle $\widetilde{C}$ is also an equivalence class, composed of the complements, so $\widetilde{C}$ generates $\widetilde{E}$.) An implication graph is inconsistent if it contains a contradictory cycle; otherwise it is consistent. The condensation of an equivalence class $E \subseteq V$ is the replacement of all vertices in E by a single vertex e and all vertices in $\widetilde{E}$ with a complementary vertex $-e$ with their predecessor and successor sets given by:

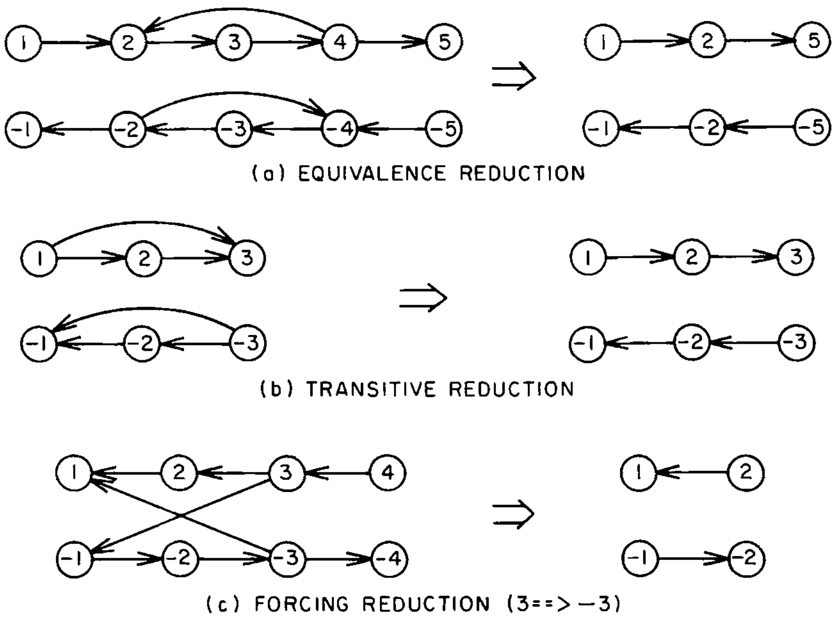

Fig. 1. Examples of logical reduction.

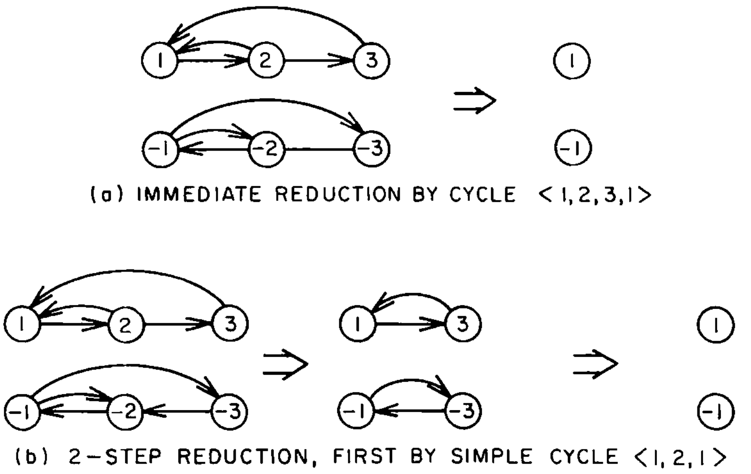
(a) IMMEDIATE REDUCTION BY CYCLE <1,2,3,1>
(b) 2−STEP REDUCTION, FIRST BY SIMPLE CYCLE <1,2,1>

Fig. 2. Application of Church–Rosser property to equivalence reduction.

$$P(e) = P(E) - \mathrm{E}, \qquad S(-e) = S(E) - E$$

$$P(-e) = P(\tilde{E}) - \tilde{E} \text{ and } S(-e) = S(\tilde{E}) - \tilde{E}.$$

(Note that E and $\tilde{E}$ are disjoint, since the generating cycle is presumed not to be contradictory.) The equivalence reduction of a consistent implication graph is the condensation of each equivalence class, which results in an acyclic implication graph. A contradictory cycle is minimal if it contains no proper subcycle that is contradictory.

A digraph is *transitively closed* if every descendant of a vertex v is also a successor of v. The transitive closure of a digraph is the digraph obtained by adding arcs to make it transitively closed. An arc is transitively redundant if its removal leaves the transitive closure unchanged. A digraph with no transitively redundant arcs is *transitively minimal.* The transitive reduction of a digraph is the unique transitively minimal digraph having the same transitive closure.

If, in an implication graph, $v \Rightarrow -v$ for some vertex v, then v is *forcing*; its forcing set $F(v)$ is $-v$ plus all descendants of $-v$. An implication graph is forcing if it contains a forcing vertex; otherwise, it is nonforcing. The forcing reduction of an implication graph is the removal of $F(v) \cup \tilde{F}(v)$ for all forcing vertices v.

An acyclic implication graph that is transitively minimal and has no forcing vertex is called *logically irreducible* or simply irreducible. Figure 1 illustrates each of the three reductions just defined: equivalence, transitive and forcing. When applied to a consistent implication graph, these three reduction operations can be shown to satisfy the Church–Rosser property [16]: if reductions are applied in any order until no reduction is possible, the result is a unique implication graph independent of the specific reductions applied. Figure 2 illustrates an example. By the Church–Rosser property, we arrive at the same irreducible implication graph whether we recognize the entire cycle $\langle 1, 2, 3, 1 \rangle$ at once or first the cycle $E = \langle 1, 2, 1 \rangle$.

The Church–Rosser property may fail for an inconsistent implication graph. Figure 3 shows an example where different minimal contradictory cycles are obtained, depending on whether we first apply transitive reduction.

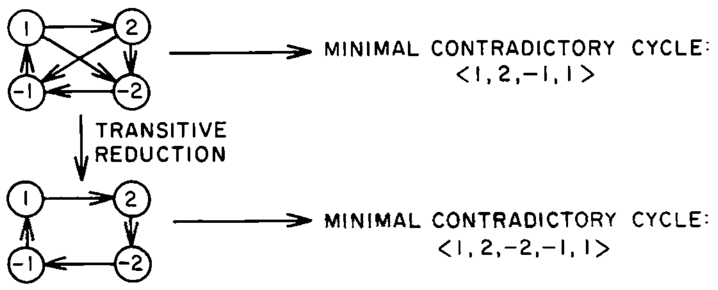

Fig. 3. Failure of Church – Rosser property for an inconsistent implication graph.

2.2. RULE-BASES

We are given atomic propositions $P_1, P_2, \dots, P_n$, and we let $\sim P_i$ denote the negation of P_i. A *literal* is an atomic proposition or its negation. A logical expression is formed by literals and logical connectives: the disjunction ($\vee$), conjunction ($\wedge$) and implication ($\rightarrow$).

A *rule-base* R is a collection of implications of the form $A \rightarrow C$, called a rule; A is called the antecedent and C is called the consequent. Both A and C are, in general, logical expressions. A rule is *simple* if its antecedent and its consequent are literals. The rule-base is simple if each of its rules is simple.

A *feasible solution* is a truth assignment to the atomic propositions such that all rules in R are logically true. The set of feasible solutions is denoted by $S(R)$.

A rule chain is an ordered sequence of rules $\langle A_1 \rightarrow C_1, \dots, A_k \rightarrow C_k \rangle$, such that $k > 1$ and $A_i = C_{i-1}$ for $i = 2, \dots, k$. This is sometimes abbreviated to $A_1 \rightarrow A_2 \rightarrow \dots \rightarrow A_k$.

The *rule-base validity problem* is to detect (immediate or deferred):

– *Inconsistency*: is $S(R)$ empty?
Equivalently, does R contain a rule chain,
$A \rightarrow \dots \rightarrow \sim A \rightarrow \dots \rightarrow A$?

– *Circularity*: does there exist a subset of the literals whose truth values must always be equal?
Equivalently, does R contain a rule chain,
$A \rightarrow \dots \rightarrow A$?

– *Redundancy*: does there exist a rule whose removal leaves $S(R)$ unchanged?
Equivalently, if $[A \rightarrow C] \in R$, does R contain a rule chain,
$A \rightarrow \dots \rightarrow C$?

– *Inflexibility*: does there exist a proposition whose truth value is the same in every feasible solution?
Equivalently, does R contain a rule chain,
$A \rightarrow \dots \rightarrow \sim A$?

We solve the simple rule-base validity problem with a 2-phase procedure in the next section. Then, we shall introduce optimization criteria and show how to go beyond these fundamental questions, giving insights to the expert for the scope of the rule-base. Afterwards, we shall consider the general case where rules may not be simple.

3. Logical reduction of simple rule-bases

Let R be a simple rule-base with m (simple) implications over n atomic propositions. Define the associated implication graph $G(R)$ by letting vertex i represent proposition P_i and $-i$ its negation $\sim P_i$. Then, define contrapositive arcs associated with each rule by the following productions.

$$P_i \rightarrow P_j \Rightarrow \langle i, j \rangle \text{ and } \langle -j, -i \rangle .$$

$$P_i \rightarrow \sim P_j \Rightarrow \langle i, -j \rangle \text{ and } \langle j, -i \rangle .$$

$$\sim P_i \rightarrow P_j \Rightarrow \langle -i, j \rangle \text{ and } \langle -j, i \rangle .$$

$$\sim P_i \rightarrow \sim P_j \Rightarrow \langle -i, -j \rangle \text{ and } \langle j, i \rangle .$$

The first phase of the logical reduction procedure applies topological sorting to the implication graph. If the implication graph is inconsistent, a contradictory cycle will be found, and it is simple to reduce this in linear-time to a minimal inconsistency to aid the expert in rectifying the rule-base. Otherwise, if the implication graph is consistent, phase 1 ends with an equivalence reduction, replacing each cycle with a literal representative. Phase 2 then proceeds to construct a logically irreducible implication graph by eliminating transitive redundancy and propositions whose truth values are forced.

Here is an overview of the 2-phase logical reduction procedure for simple rule-bases.

LOGICAL REDUCTION PROCEDURE

Phase 1: (Compute acyclic implication graph.)

For each $A \rightarrow C$ in R, enter arcs $\langle u, v \rangle$ and $\langle -v, -u \rangle$ into TOPSORT (where u and v are signed indexes associated with the literals that define the antecedent A and consequent C). If TOPSORT terminates successfully, end phase 1. Otherwise, if TOPSORT terminates with cycle C, test if C contains a complementary pair of vertices. If so, reduce C to a minimal inconsistency and exit INCONSISTENT; else, condense C and continue phase 1.

Phase 2: (Compute irreducible implication graph.)

Let $\langle v_1, v_2, \ldots, v_N \rangle$ denote the sort order obtained in phase 1. (Now N is the number of equivalence classes and v_i is a literal that represents the ith equivalence class.) For $i = 1, 2, \ldots, N$, do the following.

(1) Test if v_i is forcing. If so, remove $F(v_i) \cup \widetilde{F}(v_i)$ and advance to next i.

(2) For each $w \in S(v_i)$, test if $\langle v_i, w\rangle$ is transitively redundant. If so, remove $\langle v_i, w\rangle$.

Phase 1 of the logical reduction procedure executes Knuth's topological sorting algorithm TOPSORT, which is linear-time. If it is not successful, Knuth gives a linear-time algorithm to find a cycle C, and we add a step of recording the minimum index value:

$$k = \min\,[v : v \in C]\ .$$

Then, we assign the integer label $L(v) = k$ for all $v \in C$. If $L(v) = L(-v)$, v and $-v$ are in a cycle, so R is inconsistent. In that case, we merely traverse once more to eliminate subcycles, so the final inconsistency reported to the expert is in the form of a minimal cycle containing a proposition and its negation.

If the cycle does not contain a complementary pair, the implication graph is condensed, with k as its representative. This labelling gives automatic merger of cycles. For example, if we first find cycle $\langle 1, 2, 3, 1\rangle$, then $v = 1$ is its representative and, if the original graph also has the cycle $\langle 2, 3, 4, 2\rangle$, we will automatically assign labels of 1 to $v = 1, 2, 3, 4$, giving one equivalence class for all four propositions. Moreover, the Church–Rosser property holds: if we obtain cycle C, we also obtain $\widetilde{C}$, and the complementary pairs of cycles so obtained do not depend on the order of the condensations.

Thus, phase 1 has the following properties:

(1) It is linear-time.
(2) If R is inconsistent, phase 1 ends with a minimal inconsistent cycle.
(3) If R is consistent, phase 1 ends with an equivalence reduction.

Now consider phase 2. The topological sort order allows us to look forward (without backtracking) while eliminating all forced propositions and redundant rules. The worst time complexity is bilinear, $O(\#V\#A)$, but there are some time-saving tactics. During topological sorting, it is not difficult to define the *level function* [17] $L : V \to \{0, 1, \ldots, \#V\}$. If v is a source, $L(v) = 0$; otherwise, $L(v)$ is the length of a shortest path from some source.

When testing for whether v is forcing, we first ask: $L(v) < L(-v)$? If not, v cannot be forcing. If so, we need to test if there is a path from v to $-v$ (and we may confine lengths to be within $L(-v) - L(v)$). One way to do this is to define a subdigraph composed of v, $-v$ and all vertices w whose level is between: $L(v) < L(w) < L(-v)$. Vertex v is a source in this subdigraph, and vertex $-v$ is a sink. Putting unit capacities on all arcs, we compute a max-flow from v to $-v$. This flow will be 1 iff $v \Rightarrow -v$; otherwise, it will be zero. The worst time to do this is $O(\#A)$, giving an overall worst time of $O(\#V\#A)$.

When testing for whether $\langle v, w\rangle$ is transitively redundant, we may again filter candidates by using the min jump function [17], $M : V \to \{1, 2, \ldots, \#V\}$, where

$$M(v) = \min\,[L(w) - L(v) : w \in S(v)]\,.$$

Then, if $L(w) - L(v) = M(v)$, arc $\langle v, w\rangle$ is not transitively redundant. Otherwise, if $L(w) - L(v) > M(v)$, we must test further. This can be done with max-flow, as above, by defining the subdigraph with v as its source, w as its sink, and all between-level arcs, except $\langle v, w\rangle$, included. Then, this will determine if $v \Rightarrow w$ without arc $\langle v, w\rangle$.

Summarizing, phase 2 ends with the following properties:

(1) Its worst time complexity is bilinear, $O(\#V\#A)$.

(2) If any proposition is forced to have a particular truth value in every feasible solution, this is found and the proposition is eliminated (along with descendants).

(3) Each proposition not eliminated has value TRUE in some feasible solution and value FALSE in some (other) solution.

(4) If any rule is redundant, it is eliminated from R.

(5) Any rule not eliminated binds the solution set in that its removal admits at least one more feasible solution.

4. Optimization

Here, we suppose we have a logically irreducible simple rule-base, having applied the procedure of the previous section. Now, it may be desirable to infer additional properties, such as the maximum number of propositions that can have value TRUE.

We say R is *separable* if its implication graph $G(R)$ has the property that complementary vertices are in different weak components. In this case, we let $I(R)$ denote the *core* of the implication graph, where $G(R) = I(R) \cup \tilde{I}(R)$ and $I(R) \cap \tilde{I}(R) = \emptyset$. When this is the case, define the digraph $N(R)$ as follows. Let V^+ denote the vertices v of $I(R)$ for which $v > 0$, and let V^- denote those for which $v < 0$. Separability ensures that $V^+ \cup V^-$ is a partition of the vertices of $I(R)$, and that every proposition or its negation is represented (not both). Augment $I(R)$ with two vertices: a source s and a sink t. Then, $N(R)$ is this augmented network plus the arcs (s, V^+) and (V^-, t). Let V' denote the set of labelled vertices after determining the max-flow from s to t in $N(R)$, and let V'' be the unlabelled vertices. Finally, define

$$T(R) = \{v \in V' : v \neq s \text{ and } v > 0\} \cup \{-v : v \in V'' - \{t\} \text{ and } v < 0\}.$$

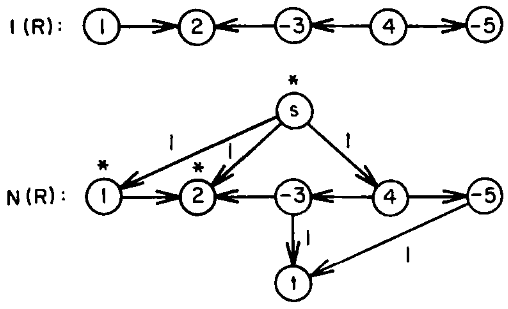

Fig. 4. Illustration of max-flow equivalence.

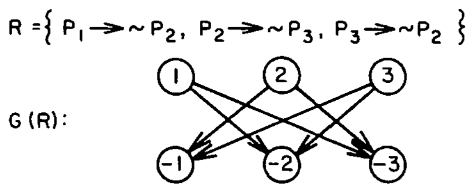

Fig. 5. Nonseparable rule-base.

MAX-FLOW EQUIVALENCE THEOREM

If R is a separable, simple rule-base, the maximum number of propositions that can be TRUE equals $\#T(R)$. Further, this is achieved by setting P_k = TRUE for all k in $T(R)$.

Figure 4 shows an example of a logically irreducible implication graph, with core $I(R)$. Figure 4 also shows labels ($\star$) on $N(R)$ obtained from the Ford–Fulkerson max-flow/min-cut algorithm, which yields the vertex partition $V' = \{s, 1, 2\}$ and $V'' = \{t, -3, 4, -5\}$. The maximum number of propositions that can be TRUE is 4, and one such assignment is $T(R) = \{1, 2, 3, 5\}$.

Figure 5 shows a logically irreducible implication graph that is not separable. In this case, max-flow equivalence fails. We next characterize an essential property of such failure.

ODD CYCLE THEOREM

If R is not separable, $G(R)$ contains an even semi-cycle that is an implication graph.

Proof

Suppose v and $-v$ are in the same weak component of $G(R)$. Let the semi-path connecting them be $(v_0 = v, v_1, \dots, v_{p-1}, v_p = -v)$. If this contains another complementary pair, reduce the semi-path until v_0 and v_p are the only complementary pair (i.e. discard $(v_0, \ldots, w)$ and $(-w, \ldots, v_p)$, then define $v = w$, if such w is an internal vertex in the semi-path). Now $G(R)$ must also contain the semi-path $(-v_p = v, -v_{p-1}, \ldots, -v_1, -v_0 = -v)$. The union of these two semi-paths is an implication graph and a semi-cycle. Further, there are $2p$ vertices (and arcs) in this union, so the semi-cycle is even.

COROLLARY

If $G(R)$ is a forest, R is separable.

Once R is logically reduced, it is easy to test whether it is separable. If so, the Max-Flow Equivalence Theorem gives an efficient algorithm for maximizing the number of propositions that can be true in a feasible solution. Moreover, it is not difficult to apply the same result to obtain the minimum number, giving a range on the total number of TRUE assignments. It is equally easy to assign weights to the propositions and obtain a range on total (additive) weight.

The next section presents another rule-base management aid, pertaining to another form of reduction, aimed at keeping as few rules as possible.

5. Rule-base consolidation

We suppose R is simple, logically irreducible and separable. Without loss of generality, we shall suppose each implication is of the form $P_i \rightarrow P_j$. (Negations may be replaced by simple transformation, owing to separability.) Two rule-bases, R and R', are said to be *logically equivalent*, written $R \Longleftrightarrow R'$, if they are defined over the same set of atomic propositions and their solution sets are equal:

$$S(R) = S(R').$$

We consider the following.

RULE-BASE CONSOLIDATION PROBLEM

Given a class of rule-bases $\boldsymbol{R}$ and a member R, find $R^* \in \boldsymbol{R}$ such that:

(1) $R^* \Longleftrightarrow R$, and

(2) $\#R^*$ is a minimum over $\boldsymbol{R}$, subject to (1).

To illustrate, consider the following example:

$$R = \{P_1 \rightarrow P_2,\ P_1 \rightarrow P_3,\ P_4 \rightarrow P_3,\ P_5 \rightarrow P_2\}.$$

Two equivalent rule-bases are:

$$R' = \{P_1 \to (P_2 \wedge P_3),\ P_4 \to P_3,\ P_5 \to P_2\},$$

$$R'' = \{(P_1 \vee P_5) \to P_2,\ (P_1 \vee P_4) \to P_3\}.$$

Each rule-base has fewer rules than R, but they are not simple. We have permitted disjunctive antecedents and conjunctive consequents.

The Rule-Base Consolidation Problem is a special case of digraph inversion [7,8], as follows. Let M denote a qualitative matrix (i.e. $M_{ij} \in \{0, -1, +1\}$ for all i, j). The fundamental digraph $D(M) = [V, A]$ is the following bipartite graph. Let $V = V_r \cup V_c$, where vertices in V_r correspond to rows of M, and vertices in V_c correspond to columns of M. For $M_{ij} = -1$, arc $\langle i, j\rangle$ corresponds with $i \in V_r$ and $j \in V_c$; for $M_{ij} = +1$, arc $\langle j, i\rangle$ corresponds. The row digraph $\mathrm{RD}(M) = [V_r, A_r]$ has vertex set V_r. Arc $\langle i, k\rangle \in A_r$ corresponds to a 2-path in $D(M)$; equivalently, there is a column of M, say j, for which $M_{ij} = -1$ and $M_{kj} = +1$. The digraph inversion problem is to characterize M for which $\mathrm{RD}(M) = D$, where D is a given digraph. Of particular interest are minimal inverses: no column may be removed without violating the defining structural equation $\mathrm{RD}(M) = D$. A minimum inverse is one with the fewest number of columns among all inverses.

The Rule-Base Consolition Problem may be formulated as a digraph inversion problem by associating V_r with propositions and V_c with rules. Then, for each column of M let $X_j \to Y_j$ denote an associated diclique:

$$X_j = \{i : M_{ij} = -1\} \quad \text{and} \quad Y_j = \{i : M_{ij} = +1\}.$$

In our case, X_j represents a disjunctive antecedent, and Y_j represents a conjunctive consequent. The set of all rule-bases logically equivalent to R is precisely the set of digraph inverses of $\mathrm{RD}(M)$ for M equal to the incidence matrix of $I(R)$.

For the above example, the digraph inverses corresponding to R' and R'' are M' and M'', as follows:

$$M' = \left[\begin{array}{ccc} -1 & & \\ 1 & & 1 \\ 1 & 1 & \\ & -1 & \\ & & -1 \end{array}\right]\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \qquad M'' = \left[\begin{array}{cc} -1 & -1 \\ 1 & \\ & 1 \\ & -1 \\ -1 & \end{array}\right]\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}$$

FIRST CONSOLIDATION THEOREM

Suppose R is a simple, separable rule-base with core graph $I(R) = [V, A]$. Denote sources by $S \subset V$ and sinks by $T \subset V$.

(1) If R is the class of rule-bases that has disjunctive antecedents and simple consequents, $\#R^\star$ equals the number of propositions that are not sinks in $I(R)$, given by the following (called the "inclaw inverse"):

$$R^\star = \{V(P_i : v_i \in P(v_j)) \rightarrow P_j : v_j \in V - T\}.$$

(2) If R is the class of rule-bases that has conjunctive consequents and simple antecedents, $\#R^\star$ equals the number of propositions that are not sources in $I(R)$, given by the following (called the "outclaw inverse"):

$$R^\star = \{P_i \rightarrow \wedge(P_j : v_j \in S(v_i)) : v_i \in V - S\}.$$

The proof of the First Consolidation Theorem follows from one of the results in [7]. The more complex case is when R is the class of rule-bases that has disjunctive antecedents and conjunctive consequents. Now a solution is not immediate, but we have the bound:

$$\#R^\star \geqslant \operatorname{diam}(I(R)).$$

(The diameter of a graph is the longest shortest path between two vertices.) This bound is useful when combined with a construction that yields a minimal rule-base. It allows us to bound how far the constructed rule-base is from a minimum.

We now show how to construct a minimal rule-base, using properties that every minimum rule-base must satisfy. Let M be the qualitative matrix associated with an irreducible rule-base, starting with M equal to the incidence matrix of $I(R)$, where R is the given simple rule-base. Then, the following *consolidation operation* may be performed iteratively until no further consolidation is possible.

For two columns, $X_j \rightarrow Y_j$ and $X_k \rightarrow Y_k$, for which either

$$X_j = X_k \text{ and } Y_j \cap Y_k = \emptyset, \text{ or } X_j \cap X_k = \emptyset \text{ and } Y_j = Y_k,$$

replace these with the one column: $X_j \cup X_k \rightarrow Y_j \cup Y_k$.

In our example, either of the two rule-bases could be generated by applying consolidation operations (differently).

SECOND CONSOLIDATION THEOREM

Suppose R is a simple, irreducible, separable rule-base. Then, for $\boldsymbol{R}$ equal to the set of rule-bases for which antecedents may be disjunctive and consequents may be conjunctive, the following hold:

(1) Consolidation preserves logical equivalence: $R' \Longleftrightarrow R$ if R' is the result of a consolidation applied to R.

(2) If consolidation operations are applied until no further consolidation is possible, the resulting rule-base is minimal.

(3) Every minimum rule-base of $\boldsymbol{R}$ can be reached by a sequence of consolidation operations.

6. Compound rule-bases

Here, we consider the general case of compound antecedents and consequents in any rule. Our strategy is to decompose the rule-base into a related family of simple rule-bases to which we can apply the efficient Logical Testing Procedure. First, let us partition the rule-base: $R = R_s \cup R_c$, where R_s is simple and R_c is compound.

Let $[A \to C] \in R_c$ be represented by two sets: a = set of literals that comprise the antecedent in elementary conjunctive form, and c = set of literals that comprise the consequent in elementary disjunctive form. The rule is thus presumed to be of the form:

$$(\wedge L_i : i \in a) \to (\vee L_j : j \in c).$$

(This loses no generality because any disjunction in the antecedent or conjunction in the consequent immediately decomposes: $(A \vee B) \to C$ becomes two rules, $A \to C$ and $B \to C$, and $A \to (B \wedge C)$ becomes $A \to B$ and $A \to C$.)

The *dimension* of a rule $\langle a, c\rangle$ is defined to be the product: DIM $(a, c) = \#a\#c$. Clearly, a rule is simple iff DIM $(a, c) = 1$. For a compound rule $\langle a, c\rangle \in R_c$, we define the *decomposition into simple rule-base* R_s, as the collection $\{R^1, R^2, \ldots, R^r\}$, where r = DIM (a, c) and

$$R^k = R_s \cup [L_i \to L_j] \quad \text{for} \quad k = (i-1)\#a + j, \ i \in a \ \text{ and } \ j \in c.$$

Here is an overview of the decomposition strategy, which we shall validate.

DECOMPOSITION PROCEDURE

Start with $r = 1$ and $R^1 = R_s$. Then, perform each of the following steps.

(1) Apply the Logical Testing Procedure to each simple rule-base $R^1, \ldots, R^r$.

(2) If R^i is inconsistent, permanently remove R^i, reducing r by 1 and , if this causes $r = 0$, exit INCONSISTENT.

(3) If P_j has the same forced value in all R^i, its value is forced in R. In that case, eliminate P_j and update all rules in R to reflect this elimination. If this causes a compound rule to become simple, transfer it from R_c to each R^i.

(4) If E is the same equivalence class in all R^i, E is an equivalence class in R. In that case, replace each literal in E by a representative for all rules in R. If this causes a compound rule to become simple, transfer it from R_c to each R^i.

(5) If there were any transfers from R_c, repeat steps (1)–(5).

(6) If $R_c = \emptyset$, terminate; otherwise, select a rule from R_c with minimum dimension, say $\langle a, c\rangle$.

(7) Decompose $\langle a, c\rangle$ into each simple rule-base R^i, and increase r accordingly. Then, go to step (1).

The Decomposition Procedure uses a strategy of "solve what we can" by a decomposition, which remains to be validated. Implicit in the tests are the following properties, which are immediate consequences of the definitions and elementary logic.

- Any feasible solution for R must be a solution for at least one of the rule-bases in the decomposition $\{R^1, \ldots, R^r\}$.
- If P_j has the same forced value in every rule-base in the decomposition, it must have the same forced value in every feasible solution of R.
- If E is the same equivalence class in every rule-base in the decomposition, it must be an equivalence class in R.

DECOMPOSITION THEOREM

The Decomposition Procedure terminates with the following properties:

(1) If exit is INCONSISTENT, $S(R) = \emptyset$.

(2) If exit is normal, the decomposition set $\{R^1, \ldots, R^r\}$ is composed of simple rule-bases and

$$S(R) = \cup\{S(R^i) : i = 1, \ldots, r\}.$$

That is, every feasible solution for R is a feasible solution for some R^i.

The decomposition Procedure did not mention redundancy because we must say more about its meaning for compound rules. In general, a rule $[A \to C] \in R$ is redundant if

$$S(R) = S(R - [A \to C]),$$

(i.e. removal of the rules does not alter the set of feasible solutions). In the case of a simple rule, redundancy is equivalent to transitive redundancy. For a compound rule, the notion of transitive is undefined. We can, however, infer redundancy in the course of executing the Decomposition Procedure by the following.

REDUNDANCY THEOREM

Suppose $\langle a, c\rangle$ is decomposed into R_s as $\{R', \ldots, R^r\}$, where $R^k = R_s \cup [L_i \to L_j]$. Then, $A \to C$ is redundant in $R = R_s \cup [A \to C]$ iff $L_i \to L_j$ is transitively redundant in R^k for all $k = (i-1)\#a + j$, $i \in a$ and $j \in c$.

Proof

$S(R_s \cup [A \to C]) = \cup\, S(R_s \cup [L_i \to L_j]) = S(R_s)$ iff $L_i \to L_j$ is transitively redundant in R^k for all k.

In closing, let us consider practical aspects of executing the Decomposition Procedure. Clearly, this is combinatorially explosive, so complete execution may not be practical. The procedure is well suited for a parallel computer architecture, such as the Intel Hypercube, so it is unclear how useful it may prove in practice. With related experience in other combinatorial problems, notably integer programming, we know that the search mechanism, perhaps with different selection criteria in step (6), can be very powerful. In general, it would seem that it is at least useful to apply the procedure one major iteration; that is, without executing step (6). This will at least test the simple rules and may cause some compound rules to become simple. The extent of branching [step (6)] is an avenue for further research, which brings us to the final section.

7. Avenues for further research

We have shown how a simple rule-base can be logically tested in linear-time to check consistency and circular reasoning. As a by-product, we can also eliminate forced values and redundant rules in bilinear time. In considering further testing, we showed how, when separability holds, an efficient labelling algorithm can determine a range of truth assignments. We have not settled the case of non-separable rule-bases, and we have not considered the complication of certainty factors (where additivity fails). These comprise two avenues for further research.

The consolidation problem needs additional research to examine what representation is most efficient for operation and maintenance. In particular, minimizing the number of rules need not be the best criterion.

The general case of compound rules was addressed, showing a way to exploit parallel architecture through a decomposition principle. The method is, however, combinatorially explosive, so more research is needed to examine how practicable this can be. The strategy of "solving what we can" is sometimes adequate, but clearly this is more likely to be a postponement of difficulties as we seek to manage large rule-bases.

Acknowledgement

The authors gratefully acknowledge stimulating discussions with Robert Jeroslow, which helped to organize these results.

References

[1] A.V. Aho, M.R. Garey and J.D. Ullman, The transitive reduction of a directed graph, SIAM J. Comput. 1(1972)131.

[2] M.L. Balinski, On a selection problem, Mgt. Sci. 17(1970)230.

[3] C.E. Blair, R.G. Jeroslow and J.K. Lowe, Some results and experiments in programming techniques for propositional logic, Comp. and O.R. (to appear).

[4] M.A. Bramer, Expert systems: The vision and the reality, in: *Research and Development in Expert Systems*, ed. M.A. Bramer (Cambridge University Press, 1985) pp. 1 – 12.

[5] L.R. Ford, Jr. and D.R. Fulkerson, *Flows in Networks* (Princeton University Press, 1962).

[6] F. Glover, D. Klingman, M. Mead and J. Mote, A note on specialized versus unspecialized methods for maximum-flow problems, NRLO 31(1984)63.

[7] H.J. Greenberg, J.R. Lundgren and J.S. Maybee, Inverting graphs of rectangular matrices, Discr. App. Math. 8(1984)255.

[8] H.J. Greenberg, J.R. Lundgren and J.S. Maybee, Digraph inversion, University of Colorado (1986).

[9] F. Harary, R.Z. Norman and D. Cartwright, *Structural Models: An Introduction to the Theory of Directed Graphs* (Wiley, 1965).

[10] R.G. Jeroslow, Computation-oriented reductions of predicate to propositional logic, Dec. Supp. Systems (to appear).

[11] E.L. Johnson and M.W. Padberg, Degree-two inequalities, clique facets, and biperfect graphs, Ann. Discr. Math. 16(1982)169.

[12] A.B. Kahn, Topological sorting of large networks, CACM 5(1962)558.

[13] D.E. Knuth, *The Art of Computer Programming, Vol. I: Fundamental Algorithms* (Addison-Wesley, Reading, MA, 1968).

[14] J.C. Picard, Maximal closure of a graph and applications to combinatorial problems, Mgt. Sci. 22(1976)1268.

[15] J.M.W. Rhys, Shared fixed cost and network flows, Mgt. Sci. 17(1970)200.

[16] R. Sethi, Testing for the Church – Rosser property, J. ACM 21(1974)671.

[17] J. Valdes, R.E. Tarjan and E.L. Lawler, The recognition of series parallel digraphs, SIAM J. Comput. 11(1982)298.