Propositional Logic Systems – Verification, Conflict Resolution, and Procedural Systems

Dr. Rick Hicks

Chief Scientist, A. I. Developers

Texas A&M International University

Talk Outline

- A Brief History of Verification of Rule-Based Systems
- Verification Criteria and How - To
- When to Verify - Verification and Solution Strategy
- Procedural Propositional Logic Systems
- Future Fun

Basic Definitions

- Verification – the science of proving that the specifications are implemented correctly.
  - “Did we build the system Right?”

- Validation – the art of determining if the system meets the firm’s needs.
  - “Did we build the Right system?”

Verification Research

- TEIRESIAS (Shortliffe) was used with MYCIN. Created knowledge map and explanations (How, Why, bugs).
- ONCOCIN (Suwa) partitioned the rule base, tested for conflicts, redundancy, subsumption, missing rules.

CHECK (Nyugen)

A. Subsumed rules
B. Redundancy
C. Conflicting rules
D. Unnecessary IF statements
E. Circular rules
F. Completeness
G. Unreferenced attribute values
H. Unreachable conclusions
I. Dead-end Ifs and goals

EVA (Stachowitz)

- Logic:
  - 1. Consistency
  - 2. Numeric completeness
- Extended Logic
  - 3. Inconsistency / 6. Conflict under generalization
  - 4. Inconsistency / 7. Conflict under incompatibility
  - 5. Inconsistency / 8. Conflict under synonymy

EVA Continued

- Structure
  - 9. Reachability
  - 10. Redundancy
  - 11. Relevance
  - 12. Cycles
- Extended Structure
  - 13. Duplication
  - 14. Subsumption
  - 15. Relevance
  - 16. Indirect cycle

EVA Continued

- Semantics
  - 17. Legal Range
  - 18. Legal values
  - 19. Data types
  - 20. Legal values for individual arguments
  - 21. Min and max occurrence
  - 22. Incompatible values
  - 23. Legal value combinations
  - 24. Subrelation argument and data type consistency

EVA conclusion ☺

- Future Testing
  - 25. Omission testing
  - 26. Rule proposer
  - 27. Behavior verifier
  - 28. Control checker

Two Tier Verification *

- Partitions both the rule base and the verification criteria.
- The rule base is partitioned into “rule clusters” in which every rule reaches the same conclusion.
- The verification criteria are partitioned into Local Criteria that are applicable to each rule cluster and Global Criteria applicable to the entire rule base.

*http://www.ez-xpert.com/whitepapers/verification.html

Local Verification Criteria

- Completeness – any combination of legal values will fire at least one rule (F, 2, 25, 26).
- Consistency – no two rules can return different conclusion values for the same set of fact values (C, 1, 2, 3, 4, 5, 6, 7, 8).
- Domain constraints – values must be legal values, ranges, and data types, no host language reserved words (H, 17, 18, 19, 20, 21, 22, 23)

Local Verification Criteria 2

- Conciseness – the knowledge in the rule base should be expressed in the most concise manner. Some problems are:
  - Subsumed rules (A, 11)
  - Redundancy (B, 10)
  - Unnecessary IF statements (D)
  - Local cycles (12)

Global Verification Criteria

- Reachability – every condition and conclusion will receive a value when needed.
  - Missing values (G)
  - Cycles (E, 16)
  - Unused conditions or rules (J, 9, 13, 14, 15)
- Global Domain Constraints – must hold between rule clusters as well as within the rule cluster (H, 1, 24)

Verification from an IDE Perspective

- Central Knowledge Repository
  - Definitions of conditions and actions
  - Definition of rule base structure
- Explicit definitions plus CWA
- Verifies for:
  - Domain constraints (Global and Local)
  - Reachability

Verification / IDE continued

- Rule builders – use Repository definitions and constrains rule during development.
  - Completeness
  - Consistency
- Conciseness is achieved with truth-preserving simplification algorithms.
  - ID3/4, C4.5, etc. (Ross Quinlan)
  - R5 Algorithms (chipping + range simplification)

- OK, that’s the easy part. We know what to verify.
- We know how to verify it.
- WHEN do we verify it???

Some Criteria are Always Verified

- Global Criteria (Domain constraints, reachability) are always enforced.
- Local Criteria:
  - Domain constraints are always enforced.
  - Conciseness is optional (but it eases maintenance and increases computational efficiency).
  - Completeness and Consistency are enforced… ***sometimes***, dependent on Rule Type.

The Need for Rule Types

- Each rule cluster in the rule base may have a different solution strategy.
- Each solution strategy dictates the verification necessary - there is a 1:1 correlation between the solution strategy used in a rule cluster and the verification criteria applicable to it.

Propositional Logic Rule Types

- We will discuss four types of rules used in propositional logic systems.
  - Deterministic
  - Incomplete
  - Exceptions
  - Belief-oriented

Deterministic Rules

- Deterministic rules are verified for Completeness and Consistency.
- Confidence is typically total (100).
- Easy to code - no need for defaults or unknowns.

Incomplete Rule Type

- Incomplete rules are defined as Incomplete but are verified for Consistency.
- A rule will not always fire. The implementation must accommodate the possibility of no value returned, such as Unknowns, Defaults, <Continue>, <Fail>, <Abort>.

Exception Rule Type

- Exception rules – like those about non-flying penguins - are expected to be Incomplete and Inconsistent, as they contain general rules and their exceptions.
- A rule will not always fire.

Belief-oriented Rules

- The desired solution may be the one with the highest belief in the conclusion, often measured with a Confidence Factor. These rules are expected to be Incomplete and Inconsistent, and are often multi-valued.
- A rule will not always fire.

From Declarative to Procedural

- Rule-based systems use an Inference Engine to perform consultations. The inference engine contains a methodology for solving rule-based problems.
- As we must know the Rule Type to perform verification, we know the solution strategy for each rule cluster during development.
- We can use the information we've gathered to create procedural propositional logic systems that *don't need an inference engine.*

Procedural Implementation of Propositional Logic

- The major function of the Inference Engine at run-time is Conflict Resolution (up to 95%!). In addition, the sequence of events that occur during a consultation impact on both accuracy and efficiency.
- To transpose declarative systems to procedural ones, we must perform conflict resolution during development.

When is Conflict Resolution Needed?

- Conflict resolution is not necessary for rule clusters with multi-valued conclusions. Any solution eventually becomes exhaustive.
- Conflict resolution is not necessary to solve for the highest belief (CNF) if all rules are tested.
- Conflict resolution is necessary for First Rule Satisfied (FRS) rule clusters – those that return only a single value.
- The remainder of this talk focuses on FRS strategies.

CLIPS Conflict Resolution Strategies

- Depth - newly activated rules are placed above all rules of the same salience.
- Breadth - newly activated rules are placed below all rules of the same salience.
- Simplicity - newly activated rules are placed above all activations of rules with equal or higher specificity.
- Complexity - places newly activated rules above all activations of rules with equal or lower specificity among rules of the same salience.
- LEX and MEA strategies orders rules of the same salience by the recency of the pattern activations in the rules.
- Random strategy uses a random number to determine the order among rules with the same salience.

Conflict Resolution in Propositional Logic

- In FOL systems, the timing of activations is quite significant, and account for four (Depth, Breadth, LEX. MEA) of the six serious conflict resolution strategies.
- In propositional logic systems, each variable has a single value, making timings of activations irrelevant. Only Simplicity and Complexity are relevant.

Re-examining Conflict Resolution

- Complexity is usually the default strategy, as it finds exceptions before more general rules, enhancing accuracy in Exception rule clusters.
- Simplicity is a good strategy for Deterministic rule clusters, as it enhances speed (simple rule covers large search space) without compromising accuracy.

Rigorous Enough for YOU?

- Simplicity and Complexity are implemented by counting the conditions in the rules.
  - This is unlikely to be sufficiently granular, as many rules may have the same number of conditions.
  - Conflict resolution often defaults to rule sequence.
- There is no conflict resolution strategy that considers the Belief in rules.
- Is this the best we can do? Two heuristics?

Other Factors in Rule Desirability

- CNF or other Belief in the rule as expressed by the rule builder.
- In addition to Simplicity, calculate the Cost of obtaining the values necessary to test a rule.
  - Assign cost to each condition.
  - Sum these costs, including subgoals.

NIET Conflict Resolution Factors

- Complexity / Simplicity – count of the conditions in the rule. Subgoals not considered.
- Cost – summation of the incremental costs necessary to get the values needed to test a rule, including cost of subgoals.
- CNF – Belief expressed in the rule by the rule base developer.

Deterministic Rules Conflict Resolution

- Deterministic rules are Complete and Consistent.
- The sequence of rules does not affect accuracy (Consistency) but does affect performance.
- Solution: solve efficiently.
  - Order by lowest cost.
  - If tied, order by the most general rule.
  - If tied, order by the highest CNF.

Incomplete Rules Conflict Resolution

- Incomplete rules are Incomplete and Consistent. A solution may not be found.
- The sequence of rules does not affect accuracy but does affect performance.
- Solution: solve efficiently.
  - Order by lowest cost.
  - If tied, order by the most general rule.
  - If tied, order by the highest CNF.

Exception Rules Conflict Resolution

- Exception rules are Incomplete and Inconsistent. A solution may not be found.
- The sequence of rules affects accuracy (Inconsistency).
- Solution: solve for exceptions first.
  - Order by the most specific rules.
  - If tied, order by the highest CNF.
  - If tied, order by the lowest cost.

Belief-Oriented Rules Conflict Resolution

- Belief-oriented rules are Incomplete and Inconsistent. A solution may not be found.
- The sequence of rules affects accuracy (Inconsistency).
- Solution: solve for highest CNF.
  - Order by the highest CNF.
  - If tied, order by the most specific rules.
  - If tied, order by the lowest cost.

FRS Belief-Oriented Rules

- Traditionally, the CNF was calculated using CNFs from the User and fired rules. All rules are fired and the solution returned the action with the highest CNF.
- In some implementations, all User CNFs may be 100 or User CNF inputs may be undesirable.
- *Belief-oriented rules can be solved FRS if only the highest rule developer's CNF (Rule CNF) is the solution criteria and User CNFs are not considered.*

Rule Base Optimization

- Optimize from the Bottom Up.
- Order each rule cluster by the three criteria.
- To determine subgoal cost, calculate the cost for the first rule in the subgoal cluster(s).
- Order the conditions in each rule cluster by lowest to highest cost.

Implementation Overview

- Conflict Resolution is performed during development, outputting a sequence of rules optimized for the rule types.
- The implementation will solve rules sequentially.
- FRS rule clusters return the first value obtained.
- Multi-valued conclusions and traditional CNF solutions are solved exhaustively.

Coding the System

- Language – choose whatever you’re comfortable with.
- Code the resulting rules as IF/THEN statements.
- Solve multi-valued conclusions and traditional CNF single-valued rule clusters exhaustively.
- Solve FRS rule clusters until one rule fires and return.
- Using the Repository, code user interface, system interfaces, and code to call other rule clusters.
- Obtain condition values by lowest cost first (BC).
- Test each value against the rule as it is obtained (BC).

Performance

- Solving for the last rule FRS (2.4Mhz PC) (CLIPS 6.21 w/90 rules < 2,000 rps)

Size	Time	Rules / Sec
648	.031	20,913
12,960	.515	25,165
19,440	.812	23,940

The Result?

- Better.
  - Higher Accuracy through exhaustive verification
  - Easy to understand – inference is transparent
- Faster.
  - Faster development using IDE
  - Higher Speed, lower computer requirements
  - Lower testing requirements
- Cheaper.
  - Inference engine costs
  - Inference engine support
  - Training for proprietary languages

The Next Step

- An IDE that integrates the three desirable solution strategies.
  - FOL
  - Propositional Logic
  - “Instant Inference” Rule Clusters

- But that’s another talk entirely…

Questions?

Dr. Rick Hicks
Chief Scientist
A. I. Developers, Inc.
Email: rick@ez-xpert.com
Voice: (210) 404-0888