Introduction
Oracle Policy Automation provides complex eligibility and policy determinations using a patented linear inferencing algorithm for forward-chain inferencing. This algorithm is particularly suited to

- Public sector organizations needing to make decisions based on legislation modeled in natural language
- Financial services companies that must model complex eligibility rules

In our experience, linear inferencing significantly outperforms the Rete algorithm, which is the standard approach used by the business rules industry.
The Basic Approach of the Linear Inferencing Algorithm

The linear inferencing algorithm exploits the way modern computer processors work. In particular, the algorithm maximizes the use of large, onboard processor memory caches by serializing the inferencing process. Essentially, the algorithm works by ordering rules so that they can be processed in a single left-to-right sweep for each forward inference cycle. Only one sweep of the rules is required, regardless of how much data is changed between inference cycles.

By minimizing the number of processor cache misses, this approach maximizes the efficiency of memory access to the rules. That is, rules are almost always accessed from the processor's high-speed onboard memory. This is the most important feature of the algorithm, because nontrivial rule sets such as those typically found in legislation and policy will not fit within such caches.

The sequential access of the rules also means more space-efficient data structures; such structures further increase the use of the onboard processor cache. Moreover, all rule access is read-only, which means the same cached copy can be shared across multiple inferencing sessions.

A Comparison with the Rete Algorithm

By comparison, the Rete algorithm—used by most rule engines—maintains a network of data flows with the aim of minimizing the discrete number of operations required for each inference. This, however, requires larger and more-complex data structures. A rule engine must essentially replicate the logical structure of the underlying rules for each inference session. Each individual change in data requires a “walk” of the network, because multiple changes in data cannot be handled at the same time. Each walk of the network requires nonsequential, mutable memory access. This results in substantially higher memory requirements and poor memory access locality, generating substantially more memory cache misses. As a result, it is less suitable to handling the complex logic expressed in legislation and organizational policy.
Extensions to the Basic Approach

The following are extensions to Oracle’s basic linear inferencing approach that extract further performance from Oracle Policy Automation.

Incremental Inferencing

The basic approach outlined earlier has been extended to provide a mechanism for incremental inferencing. The general strategy is to evaluate only those rules that are relevant for a given session, skipping past rules that cannot affect the current inference cycle. This extension requires a shared, read-only lookup table containing a map from data items to dependent rules and a per-session bitmap that stores the current relevancy of each rule.

Cyclic Dependencies

In practice nearly all rule sets (correctly) contain loops. For example,

- If the person is male, then they are not female
- If the person is female, then they are not male

Such cycles prevent an ordering of the rules that allow them to be executed strictly in sequence. These cycles can contain more than two rules, but, in practice, cyclical rule sets are usually small.

Oracle handles cyclic dependencies by treating each loop as a composite rule consisting of the minimal set of rules that fully contain the loop. These composite rules can then be sequentially ordered along with the remaining rule set.

Oracle handles cyclic dependencies by treating each loop as a composite rule consisting of the minimal set of rules that fully contain the loop. These composite rules can then be sequentially ordered along with the remaining rule set as described earlier. Composite rules are treated as indivisible during the ordering process, so the cycles within the sets do not matter; only the cycles’ composite dependencies on the remainder of the rule set are relevant.

Inferencing then proceeds as before, with a single left-to-right sweep of the rules. When a composite rule is encountered, a brute force approach is used to process the rules that comprise it. The brute force approach is not strictly sequential, but it is highly localized to protect the algorithm’s memory access performance.

Conditional Branching

Another important strategy employed by modern processors to boost performance is deep instruction pipelining. This involves overlapping the execution of instructions to keep every part of a processor as busy as possible.
One of the key hazards to instruction pipelining is conditional branching that can cause the pipeline to stall when the processor fails to predict the next instruction to execute. To ensure maximum performance is extracted from the processor, it is therefore necessary to minimize the frequency of unpredictable conditional branches.

In linear inferencing, avoidance of conditional branching through rule evaluation can pay large performance dividends. Oracle’s linear inferencing solution reduces the process of evaluating rules to a sequence of bitwise logical operations and table lookups. This helps eliminate conditional logic that, in turn, reduces the incidence of pipeline stalls and further increases the performance of Oracle Policy Automation.

Advantages of Linear Inferencing

Linear inferencing has several key advantages over the Rete algorithm used by other rule engines.

- **Easily handles multiple, simultaneous changes to input data.** This makes it better suited to real-world processing scenarios that include transactional processing and batch data upload from a database or interactive applications.

- **Minimizes per-session memory usage.** Less memory is required to service high-performance applications with heavy concurrent usage.

- **Better exploits modern processor architectures.** In particular, the linear inferencing algorithm’s space-efficient data structures increase the use of the onboard processor cache to provide much-higher performance.

Linear inferencing has several key advantages over the Rete algorithm used by other rule engines. In particular, linear inferencing can easily handle changes to input data, minimize memory usage, and optimize processor performance.

Conclusion

With its linear inferencing algorithm, Oracle Policy Automation can quickly and efficiently determine eligibility against even the most complex rule sets. Such capabilities make it valuable to both public sector organizations and financial services companies, both of which need to model complex eligibility rules in natural language.