Rete algorithm

The **Rete algorithm**[3] was the first to address this problem. The algorithm preprocesses the set of rules in the knowledge base to construct a dataflow network in which each node is a literal from a rule premise. Variable bindings flow through the network and are filtered out when they fail to match a literal. If two literals in a rule share a variable—for example, $Sells(x,y,z) \land Hostile(z)$ in the crime example—then the bindings from each literal are filtered through an equality node. A variable binding reaching a node for an n-ary literal such as $Sells(x,y,z)$ might have to wait for bindings for the other variables to be established before the process can continue. At any given point, the state of a Rete network captures all the partial matches of the rules, avoiding a great deal of recomputation.

Production system

Rete networks, and various improvements thereon, have been a key component of so-called **production systems**, which were among the earliest forward-chaining systems in widespread use.[4] The XCON system (originally called R1; McDermott, 1982) was built with a production-system architecture. XCON contained several thousand rules for designing configurations of computer components for customers of the Digital Equipment Corporation. It was one of the first clear commercial successes in the emerging field of expert systems. Many other similar systems have been built with the same underlying technology, which has been implemented in the general-purpose language OPS-5.

Cognitive architectures

Production systems are also popular in **cognitive architectures**—that is, models of human reasoning—such as ACT (Anderson, 1983) and SOAR (Laird *et al.*, 1987). In such systems, the "working memory" of the system models human short-term memory, and the productions are part of long-term memory. On each cycle of operation, productions are matched against the working memory of facts. A production whose conditions are satisfied can add or delete facts in working memory. In contrast to the typical situation in databases, production systems often have many rules and relatively few facts. With suitably optimized matching technology, systems can operate in real time with tens of millions of rules.

Irrelevant facts

Another source of inefficiency is that forward chaining makes all allowable inferences based on the known facts, *even if they are irrelevant to the goal*. In our crime example, there were no rules capable of drawing irrelevant conclusions. But if there had been many rules describing the eating habits of Americans or the components and prices of missiles, then FOL-FC-ASK would have generated irrelevant conclusions.

Deductive databases

Magic set

One way to avoid drawing irrelevant conclusions is to use backward chaining, as described in Section 9.4. Another way is to restrict forward chaining to a selected subset of rules, as in PL-FC-ENTAILS? (page 231). A third approach has emerged in the field of **deductive databases**, which are large-scale databases, like relational databases, but which use forward chaining as the standard inference tool rather than SQL queries. The idea is to rewrite the rule set, using information from the goal, so that only relevant variable bindings—those belonging to a so-called **magic set**—are considered during forward inference. For example, if the goal is $Criminal(West)$, the rule that concludes $Criminal(x)$ will be rewritten to include an extra conjunct that constrains the value of x:

$$Magic(x) \land American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \Rightarrow Criminal(x) .$$

[3] Rete is Latin for *net*. It rhymes with *treaty*.

[4] The word **production** in **production systems** denotes a condition–action rule.