[2] D. S. Wile and D. G. Allard. Worlds: an organizing structure for object-bases. In P. Henderson, editor, *SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, pages 16–26, Palo Alto CA, December 1986. Special issue of *SIGPLAN Notices*, 22(1), January 1987.

[3] G. E. Kaiser, P. H. Feiler, and S. S. Popovich. Intelligent assistance for software development and maintenance. *IEEE Software*, 5(3):40–49, May 1988.

Box D: Production Systems

Production systems have enjoyed popularity as an implementation paradigm for expert systems. A production system typically consists of a collection of rules, where each rule is a pattern-action pair. The pattern defines the conditions under which the associated action should be applied (or triggered). Patterns are written in terms of the values of objects in a shared database, called working memory. When several patterns apply to a particular state of the database, the system automatically controls the sequencing of corresponding actions, as determined by rule-ordering policies, that vary depending on the kind of production system used. Production systems are based on tool abstraction, with working memory representing the shared data pool and the production rules representing the toolies. Interactions between the enhancement and existing functions are controlled by the production system itself.

The Formalized System Development (FSD) system [1, 2] and Marvel [3] are two rule-based software development environment architectures that are relatively close to pure tool abstraction. Each combines ideas from production systems and active data and each represents the software artifacts under development in an object-oriented database. Tool fragments are automatically invoked in response to chaining on the rules, which in turn can be triggered by changes to the data. In FSD, new rules and new subclasses of existing objects can be added at any time to accommodate additional Lisp tools. In Marvel, new rules, new classes of objects, and extensions to existing classes can be added at any time to integrate additional commercial-off-the-shelf tools into the environment.

References

[1] R. Balzer. A 15 year perspective on automatic programming. *IEEE Transactions on Software Engineering*, SE-11(11):1257–1268, November 1985.