lish propositional inconsistency. This idea led John Alan Robinson (no relation) to develop resolution (Robinson, 1965).

Resolution was adopted for question-answering systems by Cordell Green and Bertram Raphael (1968). Early AI implementations put a good deal of effort into data structures that would allow efficient retrieval of facts; this work is covered in AI programming texts (Charniak *et al.*, 1987; Norvig, 1992; Forbus and de Kleer, 1993). By the early 1970s, **forward chaining** was well established in AI as an easily understandable alternative to resolution. AI applications typically involved large numbers of rules, so it was important to develop efficient rule-matching technology, particularly for incremental updates.

The technology for **production systems** was developed to support such applications. The production system language OPS-5 (Forgy, 1981; Brownston *et al.*, 1985), incorporating the efficient Rete match process (Forgy, 1982), was used for applications such as the R1 expert system for minicomputer configuration (McDermott, 1982). Kraska *et al.* (2017) describe how neural nets can learn an efficient indexing scheme for specific data sets.

The SOAR cognitive architecture (Laird *et al.*, 1987; Laird, 2008) was designed to handle very large rule sets—up to a million rules (Doorenbos, 1994). Example applications of SOAR include controlling simulated fighter aircraft (Jones *et al.*, 1998), airspace management (Taylor *et al.*, 2007), AI characters for computer games (Wintermute *et al.*, 2007), and training tools for soldiers (Wray and Jones, 2005).

The field of **deductive databases** began with a workshop in Toulouse in 1977 attended by experts in logical inference and databases (Gallaire and Minker, 1978). Influential work by Chandra and Harel (1980) and Ullman (1985) led to the adoption of **Datalog** as a standard language for deductive databases. The development of the **magic sets** technique for rule rewriting by Bancilhon *et al.* (1986) allowed forward chaining to borrow the advantage of goal-directedness from backward chaining.

The rise of the Internet led to increased availability of massive online databases. This drove increased interest in integrating multiple databases into a consistent dataspace (Halevy, 2007). Kraska *et al.* (2017) showed speedups of up to 70% by using machine learning to create **learned index structures** for efficient data lookup.

Backward chaining for logical inference originated in the PLANNER language (Hewitt, 1969). Meanwhile, in 1972, Alain Colmerauer had developed and implemented **Prolog** for the purpose of parsing natural language—Prolog's clauses were intended initially as context-free grammar rules (Roussel, 1975; Colmerauer *et al.*, 1973).

Much of the theoretical background for logic programming was developed by Robert Kowalski at Imperial College London, working with Colmerauer; see Kowalski (1988) and Colmerauer and Roussel (1993) for a historical overview. Efficient Prolog compilers are generally based on the Warren Abstract Machine (WAM) model of computation developed by David H. D. Warren (1983). Van Roy (1990) showed that Prolog programs can be competitive with C programs in terms of speed.

Methods for avoiding unnecessary looping in recursive logic programs were developed independently by Smith *et al.* (1986) and Tamaki and Sato (1986). The latter paper also included memoization for logic programs, a method developed extensively as **tabled logic programming** by David S. Warren. Swift and Warren (1994) show how to extend the WAM to handle tabling, enabling Datalog programs to execute an order of magnitude faster than forward-chaining deductive database systems.