LOSP SHORT DESCRIPTION

William Bricken
January 1987

We have developed a comprehensive computational approach to real-time parallel inference, called Losp.

Losp is based on boundary mathematics, a relatively unknown formalism that simplifies deduction via a representational transformation that decreases both the size of a logical language and the number of axioms or rules that are needed to implement the proof techniques. One logical operator/constant, the boundary, replaces the former minimum of three tokens required in traditional propositional calculus (or, not, and true). The primary computational technique that achieves deductive simplification is erasure, which replaces the traditional approach of rearrangement of tokens in the syntactic representation of the problem. This approach has been shown to be at least five times faster than the Boyer-Moore theorem-prover for problems in propositional calculus, without relying on parallelism. Memory is released as the computation proceeds toward a solution.

Boundary mathematics provides a natural form of parallelism. Logical structure is expressed by a single operator that is implicitly associative and commutative. As a consequence, all composition of boundary operators can be implemented in parallel. The parallel implementation of Losp is a network formalism in which local evaluation of connectivity determines the global result. This strong form of parallelism relies on a distributed representation of the problem in which no objects are represented more than once. We are able to parse Pure LISP code into Losp networks, permitting transparent conversion of linear code into a functionally invariant parallel model.

We have extended Losp to predicate calculus. The system combines the functional advantages of Scheme with the declarative advantages of Prolog, within a single boundary formalism. Knowledge about the function and behavior of defined predicates is expressed locally within each predicate, partitioning logical control information from control of predicates, such as transitivity rules, database instantiation, and partial evaluation. The key to this approach is a highly efficient implementation of terms as computational systems, each containing its own agenda mechanisms, memory, and context.