XSB as a Deductive Database

Konstantinos Sagonas Terrance Swift David S. Warren
Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400
{kostis, tswift, warren}@cs.sunysb.edu

XSB is a logic-based programming system that can serve as an efficient in-memory Deductive Database (DDB) engine. XSB fundamentally extends the standard functionality of Prolog to include implementations of SLG resolution (tabling or memoing), and of HiLog.

Perhaps the most significant difference between the XSB implementation and conventional DDB implementations lies in XSB's use of SLG to extend tuple-at-a-time evaluation with bottom-up declarativity. This strategy can give XSB an advantage over other DDBs in object-oriented applications, where information can be kept in complex terms in addition to being distributed across many relations.

SLG resolution is useful for recursive query evaluation, allowing programs to terminate correctly in many cases where Prolog's SLD resolution does not. In particular, SLG computes all Datalog programs finitely with polynomial data complexity. SLG also provides a solution to the practical problem of Prolog's tendency to redundantly recompute subcomputations. For these reasons, users interested in memory-resident DDBs, Parsing, and Program Analysis applications may benefit from XSB. XSB's SLG implementation:

- Is incorporated at the emulator level for maximal efficiency. The speed improvement over meta-interpreters written by the XSB group is 2-3 orders of magnitude, with considerable improvements in space as well.
- Allows programs with (modularly) stratified negation and aggregation to be evaluated, according to the well-founded semantics, at the engine level. (The current version evaluates programs with general negation at the interpreter level.)
- Allows for declaration of tabled predicates either automatically by the system or manually by the user.
- Provides standard tabling predicates which can be used to program a number of applications like meta-interpreters for the well-founded semantics, etc.
- Allows full Prolog functionality in tabled code, including cuts, meta-logical and second-order predicates.

HiLog supports a type of higher-order programming in which predicate symbols can be variable or structured. This allows unification to be performed on the predicate symbols, in addition to the arguments of the predicates. Also, the flexibility of HiLog's syntax offers a useful means of knowledge representation for object-based schemas. XSB's HiLog implementation:

- Includes a fully integrated HiLog preprocessor. HiLog terms can be used anywhere in XSB.
- Includes full compilation of HiLog. Compiled higher-order predicates execute at a minimal overhead over comparable first-order predicates.
- Provides a number of meta-logical standard predicates for manipulating HiLog terms.
- Allows sets to be constructed and manipulated in the same way as terms.

In addition, version 1.4 of XSB includes indexing capabilities greatly improved over those of standard Prolog systems. Users are offered the choice of Prolog-style hash-based indexing, or *transformational indexing*. In hash-based indexing, users can index on any argument, or on multiple arguments, removing a limitation of Prolog for data-oriented queries. Furthermore, users can index clauses with transformational indexing which builds a non-deterministic discrimination net, reducing unnecessary backtracking.

XSB has been tested on most common 32-bit hardware platforms running many versions of Unix and is available through anonymous ftp from `cs.sunysb.edu`.