The Module System of λProlog and its Realization

Gopalan Nadathur
Department of Computer Science
University of Chicago

The language λProlog incorporates a module notion that permits the space of type names, constant declarations and predicate definitions to be decomposed into smaller, named, units. Associated with each such unit is a signature that controls the visibility of names identified within that unit. Interactions between these units can take place through either an accumulation or an importation process. There are both static and dynamic effects to such interactions: the parsing of expressions may require names available through the signature of another unit and executable code may utilize procedures defined in other units.

In this talk, we will present the overall module system in some detail and will outline its realization in the *Teyjus* version of λProlog. A particular characteristic of the implementation is its support for the *separate* compilation of each module into an appropriate fragment of code. The dynamic semantics of module importation involves enhancing existing program contexts with the procedures defined in other modules. This effect is achieved through a run-time process for making available the compiler generated code for such procedures. The implementation method partitions the code space into distinct chunks corresponding to the module structure, with the management of the sub-parts being realized through a module table. Efficiency of execution requires a simple, but uniform, method for resolving names in the code for each sub-component. This requirement is realized through a suitable communication between the compilation and loading processes.