Ueda, Kazunori

Guarded Horn clauses. (English) Zbl 0771.68037

Cambridge, MA: MIT Press; Tokyo: Univ. Tokyo, Information Engineering Course, VI, 161 p. (1988).

Summary: The thesis introduces the programming language Guarded Horn Clauses which is abbreviated to GHC. Guarded Horn Clauses was born from the examination of existing logic programming languages and logic programming in general, with special attention paid to parallelism. The main feature of GHC is its extreme simplicity compared with the other parallel programming languages. GHC is a restriction of a resolution-based theorem prover for Horn-clause sentences. The restriction has two aspects: One is the restriction on data flow caused by unification, and the other is the introduction of choice nondeterminism. The former is essential for a general-purpose language and it also provides GHC with a synchronization primitive. The latter is required by the intended applications which include a system interacting with the outside world. What is characteristic with GHC is that all the restrictions have been imposed as the semantics given to the sole additional syntactic construct, guard. Although Guarded Horn Clauses can be classified into the family of logic programming languages, it has close relationship to other formalisms including dataflow languages, Communicating Sequential Processes, and functional languages for multiprocessing. Except for the lack of higher- order facilities, GHC can be viewed as a generalization of these frameworks. The simplicity and generality of GHC will make it suitable for a standard not only of parallel logic programming languages but of parallel programming languages. Moreover, it is simple enough to be regarded as a computation model as well as a programming language. Attention has always been paid to the possibility of efficient implementation during the design stage of GHC. We showed that stream merging and distribution which are expected to be heavily used can be implemented with the same time-complexity as the time-complexity of many- to-one communication in procedural languages. Furthermore, we made available an efficient compiler-based implementation of a subset of GHC on top of Prolog. GHC has lost the completeness as a theorem prover deliberately, not as a result of compromise. Nevertheless, it can be used for efficient implementation of exhaustive solution search for Horn- clause programs. We showed how to automatically compile a Horn-clause program for exhaustive search into a GHC program.

MSC:

68N17	Logic programming
68T99	Artificial intelligence
68N15	Theory of programming languages

Cited in **20** Documents

Keywords:

Guarded Horn Clauses; parallel logic programming languages; exhaustive solution search

Software:

GHC