Constraint Functional Multicore Programming

Petra Hofstedt and Florian Lorenzen
Department of Software Engineering and Theoretical Computer Sciene
Technische Universität Berlin, Germany
ph@cs.tu-berlin.de, florian.lorenzen@tu-berlin.de

We present the concurrent constraint functional programming language CCFL and the abstract machine ATAF for the evaluation of CCFL programs in a multicore environment.

Multicore architectures have become more and more important in recent years. Unfortunately, only truly parallel programs are able to benefit from their increase in computational power. There is, however, not yet an established method of programming these architectures, which is competitive to maintainability, stability, and performance of serial program development. Especially, many parallel programs do not automatically turn an increase in the number of cores into shorter run times like serial programs used to profit from higher clock rates. Regarding stability and maintainability, a declarative programming approach is desirable since side-effects, and explicit communication/synchronization of the imperative style are the root of many bugs hard to find or reproduce.

The multiparadigm programming language CCFL combines concepts from the constraint-based and functional paradigms and it supports the concurrent and parallel program development. CCFL's functional sub-language is a lazy language with a polymorphic type system. It can be considered as *computational core* of CCFL while the *coordination core* responsible for the coordination of concurrent processes is based on constraints. Ask- and tell-constraints and conjunctions allow to express concurrently working processes. We introduce the language and show how to elegantly and abstractly express typical data and task parallel execution patterns using CCFL.

The data and task distribution as well as the process coordination is controlled by the abstract machine ATAF. It implements a G-machine to evaluate functional expressions and provides facilities to run multiple cooperating processes on a fixed set of CPUs. Processes communicate via a shared constraint store realizing residuation semantics and committed choice. ATAF provides load-balancing mechanisms to maximize the utilization of each CPU and schedules processes accessing the shared constraint store with higher priority to keep locking times of atomic operations short.

CCFL programs compiled to ATAF are able to utilize several cores to gain performance in this way. Since CCFL is a declarative language, programs are written on a high level of abstraction by the virtue of a polymorphic type system, higher order functions, and recursive datatypes, as well as robust and understandable because of the absence of side-effects.

We show a few scaling results for parallel programs obtained with a prototypical implementation of ATAF on a quadcore machine.