Towards a Parallel Search for Solutions of Non-deterministic Computations

Fabian Reck and Sebastian Fischer

Department of Computer Science
Christian-Albrechts-University of Kiel
D-24118 Kiel
{fre,sebf}@informatik.uni-kiel.de

We present preliminary work on executing non-deterministic Haskell programs in parallel. Functional logic programming languages like Curry demonstrate conceptual benefits of implicit, i.e., built-in non-determinism and the purely functional language Haskell provides explicit, i.e., lightweight non-determinism by means of non-determinism monads. In principle, it should be possible to execute both explicitly non-deterministic Haskell programs as well as implicitly non-deterministic Curry programs such that different non-deterministic results are computed in parallel. However, currently there is no established framework for either of these tasks. We explore possibilities to employ parallel execution for Haskell programs that use non-determinism monads. In combination with recent advances in compiling Curry programs to pure Haskell, we hope that our developments can be directly used to improve the execution of Curry programs too.

We investigate three different approaches to multi-threaded execution of Haskell programs that use non-determinism monads, viz.,

1. explicitly using a fixed number of threads to execute different, statically known, parts of the search space in parallel,
2. explicitly using a fixed number of threads with dynamic load-balancing via a Bag-of-Tasks approach, and
3. using Haskell’s built-in support for semi-explicit parallelism to let the run-time system decide which parts to execute in parallel with how many threads.

The first approach works best if the search space is balanced such that no thread runs out of work early. It is not always easy to find a good work balance statically, however. Dynamic load balancing, as in our second approach, helps to distribute work evenly but causes more synchronization overhead because the individual tasks that are performed by the different threads are relatively small compared to our first approach. It seems reasonable to defer the choice of which tasks are worth being executed in parallel to the run-time system. Recently, its implementation has been improved considerably and first experiments indicate that semi-explicit parallelism is a good candidate for executing non-deterministic programs in parallel.