Vrije Universiteit Brussel

Proposal bachelor thesis

Title: A Cache Simulator Language for DrRacket
Promotor: Theo D'Hondt
Includes preparation course: Yes

Context

Caches play an important role in todays hardware to bridge the gap between processor speed and memory speed. Just like any other program (hardware is just petrified software) caches come in a large variety of implementations. To learn about the different flavours of caches and their impact on software performance it would be interesting to have a cache simulator. From a pedagogical point of view, writing a generic cache simulator not only encourages one to study cache algorithms, it also assimilates this knowledge by providing first hand experience with the implementation of cache algorithms in all their flavours.

Proposal bachelor thesis

This bachelor thesis proposes the implementation of two Scheme dialects in DrRacket: one language is a small subset of Scheme in which simple programs can be written; the other language allows the implementation of a memory back-end (and thus any cache algorithm) for programs written in the first language. This effectively allows to execute the same program (written in the first language) on different memory back-ends, written in the second language. PLAI (see http://docs.racket-lang.org/plai/index.html) uses the same approach to experiment with different garbage collectors.
As an evaluation we want to execute a program of which the data access patterns are well known (e.g., classic matrix-matrix multiplication) and verify if a set of memory back-ends (cache algorithms) behave as expected. This requires to implement a set of cache algorithms in the new memory back-end language. Ideally this language is expressive enough to implement recurring cache properties, such as the associativity or the eviction strategy easily.

Preparatory course bachelor thesis

The preparatory work for this bachelor thesis is twofold. First, we need to establish a set of “memory primitives” which form the abstraction barrier between the two languages discussed above. I.e., how does a program written in the first language obtain memory, which is managed by a program in the second language. Again, here PLAI is a good starting point. Second, a survey of existing cache implementations needs to be created. This entails reading chapters on cached memory in books on computer (memory) architectures. This survey then forms the basis for the design of the memory back-end language.