Scientific Stay at Vrije Universiteit, Brussels

Petr Krajča

International Center for Information and Uncertainty
Palacky University Olomouc

August 15 – September 23, 2011

Vrije Universiteit, Brussels and Software Language Lab

- Flemish university (about 10.000 students)

Software Language Lab

- part of the CS department
- world leading institution in programming language research (publications at OOPSLA/SPLASH, ECOOP)
- 3 professors, 12 post-doc's, 20 members (Ph.D./master students)
- grants by IWT (Flemish Government agency for Innovation by Science and Technology, Intel, IMEC)

Research Activities

- Ambient (prof. Wolfgang De Meuter) – mobile devices, context-oriented programming
- CAMP (prof. Viviane Jonckers) – static analysis, DSL, Meta-Level architectures
- PPP (prof. Theo D'Hondt) – dynamic parallelization, programming models, virtual machines
- ...

Need for New Approaches to Parallel Programming

- increasing number of CPU cores (currently tenths, in the near future hundreds)
- $\Longrightarrow$ new issues
- CPU core failures (IBM BlueGene/Q chip 17 + 1 cores)
- new data structures (lockless $\Longrightarrow$ memory models)
- new algorithms (e.g., Map/Reduce for graphs)
- simplifying development (transactional memory, actor model)
- automatic memory management (stop-the-world problem)
- automatic parallelization
  - Lisp (Scheme, Clojure)
  - Smalltalk (Squeak, RoarVM)
  - JavaScript

Lisp/Scheme Related Project

- several projects (Beaver, SLIP, . . .) focused on automatic parallelization
- testing environment for new features and execution models
- interpretation (compilation as an optimization)
- AST transformation/partial compilation
- Finger-trees (Beaver)
- static vs. dynamic parallelization
- speculative execution
- work-stealing (cf. fork/join execution)
- CPS-based interpretation

Continuation Passing Style (1/3)

```
(define fact
  (lambda (n)
    (if (= n 0)
      1
      (* n (fact (- n 1))))))


(define fact2
  (lambda (n k)
    (if (= n 0)
      k
      (fact2 (- n 1) (* k n)))))

(fact2 6 1)
```

Continuation Passing Style (2/3)

```
(define fact3
  (lambda (n k)
    (if (= n 0)
      (k 1)
      (fact3 (- n 1)
             (lambda (v)
               (k (* n v)))))))

(fact3 10 (lambda (v) v))
```

Continuation Passing Style (3/3)

- unifies all types of calls and jumps
- transformation performed automatically
- control of tail-calls
- straightforward implementation of call/cc
- explicit stack manipulation (cf. Schemik)
- allows for distribution of computation among threads
- non-trivial implementation in C (setjmp, longjmp, makecontext, setcontext)

Memory Management

- no-stop-the-world garbage collector (complex, benefits?)
- cache access optimizations (object ,,recycling")
- allocation optimization
- lockless queues
- "worlds" $\Longrightarrow$ variant of a software transaction memory (Clojure, JavaScript)

Conclusions

- established connection between Palacky University and Software Langauge Lab, Vrije Universiteit, Brussels
- got familiar with new trends in programming language design
- found new opportunities for future colaboration