Concurrent Programming via Access Permissions

Sven Stork

Carnegie Mellon University
svens@cs.cmu.edu
Supervisor: Jonathan Aldrich

University of Coimbra
stork@dei.uc.pt
Supervisor: Paulo Marques

July 13, 2009
Problem

Era of Concurrency is upon us!

- major paradigm shift in technology
- single core CPU reached physical and energetic limits
- Moore’s Law continues (double number of transistors every 18 months)
- add more cores to a single CPU
- “The free lunch is over” (Sutter, 2005)
- programs need to become concurrent to achieve higher performance
Problem

Why is concurrent programming hard?

- **explicit** and **low-level** primitives
- all the problem of sequential programming plus
  - deadlocks
  - race-conditions
- harder to reason about than sequential programming
- harder to debug than sequential programming

Why cannot the code automatically be parallelized?

- **implicit** or **unknown** side-effects of code
  - void foo() || void bar() ?!
Idea

A new programming paradigm

→ Concurrency by Default

Our Programming Language : Æminium

→ everything is concurrency by default

→ make side-effects explicit by using Access Permissions and Data Groups

→ use access permissions and data groups to infer data dependencies

→ dataflow system : concurrent execution solely limited by dependencies
Idea

Collection createRandomData() : \textbf{unit} \Rightarrow \textbf{unique}(\text{result})

\textbf{void} removeDuplicates(Collection c) : \textbf{unique}(c) \Rightarrow \textbf{unique}(c)

\textbf{void} printCollection(Collection c) : \textbf{immutable}(c) \Rightarrow \textbf{immutable}(c)

Dependencies compDeps(Connection c)
   : \textbf{immutable}(c) \Rightarrow \textbf{immutable}(c),\textbf{unique}(result)

Statistics compStats(Connection c)
   : \textbf{immutable}(c) \Rightarrow \textbf{immutable}(c),\textbf{unique}(result)

\textbf{void} main() {
   Collection c = createRandomData()
   printCollection(c)
   Statistics s = compStats(c)
   Dependencies d = compDeps(c)
   removeDuplicates(c)
   printCollection(c)
   ...
}

Work Plan

Current Work

- re-writting rules for simple language with unique/immutable permissions
- grammar for Æminium core language
  - unique, immutable and shared permissions
  - data groups

Future Work

- develop formal system
- implement compiler/runtime
- scalability evaluation
- user studies
Expected Results

Accepted Papers
IWACO’09 Workshop: ’Reducing STM Overhead with Access Permissions’ (co-author)

In Submission
Onward’09 Conference: ’Concurrency by Default: Using Access Permissions for Automatic Parallelisation’

Formal Work
POPL, ECOOP, OOPSLA, PLDI

Practical Work
SOSP, IDPDS, PPoPP, ISC, SC