Introducing Parallelism in an Educational Mathematics System Developed in a Functional Programming Language

Mathias Lehnfeld
Motivation

parallelism in Isabelle

Isac user requirements
Theoretical Background

functional programming
  · no side effects
  · immutable data

parallelism
  · futures

computer theorem proving
Standard ML

ML – command language for LCF

Standard ML (1990)

Poly/ML

· POSIX threads
Isabelle

logical framework
Isabelle/ML
  • guarded access
  • futures
Isabelle/PIDE
Isac

based on Isabelle mathematics-engine in Standard ML rule-based Java front-end
Isac Interaction Model

\[ \frac{6a+3b}{3b} = ? \]

**student**

\[ \frac{6a+3b}{3b} = 6a \]

**computer**

"You can’t simplify like that. Try to make products out of both, nominator and denominator.

Here is a link to look up."

\[ \frac{6a+3b}{3b} = \frac{6a}{3b} + \frac{3b}{3b} \]

\[ \frac{6a+3b}{3b} = \frac{2a+b}{b} \]

\[ \frac{6a+3b}{3b} = \frac{3(2a+b)}{b} \]
Isac Architecture

- Course author
- Students
- Engineer
- Web
- Moodle courses
- JVM
- GUI 1
- GUI 2
- ...(GUI n)
- Isac server (dialog)
- Knowledge
- Isac math-engine
- Programs
- Knowledge
- Isabelle
- Core 1
- Core 2
- ...Core n
- Mathematics author
- Scala bridge
Isabelle's Parallel Theory Evaluation

Isabelle theories in DAG structure

custom datastructures → Unsynchronized.ref

Theory_Data functor
Concurrent User Session Management

appendFormula & autoCalculate

futures

synchronized references (guarded access)
Performance Results

![Graph showing performance results with time in ms on the y-axis and number of calculations n on the x-axis. The graph compares sequential samples (seq. samples) and parallel samples (par. samples) with their respective means (seq. mean, par. mean).]
Future Work

more robust JVM / SML communication model

use more Isabelle features

- function package
- code generator
- floating point & complex numbers

Standard ML refactoring tools
Discussion

functional programming
parallelism
computer theorem proving
Isabelle
Isac
Fibonacci Numbers

fun fib 0 = 0
| fib 1 = 1
| fib n = fib (n - 1) + fib (n - 2);