The ANR Project

Paral-ITP

FA: First thoughts towards a Formal Parallel Kernel Model

Burkhart Wolff

Université Paris-Sud, LRI, Nov 2011

ANR-11-INSE-001

Motivation

Why Formal Analysis ?

- ITP's like Coq and Isabelle have due to their Kernel design a high reputation in trust-worthiness; sometimes they are even used as reference for ATP's !

- **But**: Parallelism and Asynchronicity increase complexity.

Motivation

Why Formal Analysis ?

- ITP's like Coq and Isabelle have due to their
  Kernel design a high reputation in trust-worthiness;
  sometimes they are even used as reference for ATP's !

- But: Parallelism and Asynchronicity
  increase complexity.

- Principle should be maintained:
  The prover may diverge or crash,
  but thm's should be thm's !!!

- Portability issues: Core's and GUI's should run
  on platforms Linux, MacOS, Windows (and better!)

Isabelle Kernel Implementation

- "LCF style" thm, protected in an ADT
- extended by thy-contexts that contain "managed" data from user-space
- extended by potential proof-objects
- extended by "promises", i.e. "futures" of thm's whose validations are still pending and can be effectuated by concurrent, asynchronous computations.

Parallel Nano-Kernel LCF - Architecture:

Putting the Classical Kernel actually into Plugins ... Isabelle2009 - 10 (!)

...

„Θ“ thycontexts = contexts + {
sign : Signature,
thm_db : name ⇀ thm,
...}

„$\Gamma \vdash_\Theta \varphi$“ thm = {context : CertId,
promises: name ⇀ thm future,
hyps : term,
prop : term}

CertificateTable : CertId ⇀ thycontext

status :: thm ⇀ { failed : bool, oracle: bool, unfinished: bool}

...

Parallel Nano-Kernel LCF - Architecture:

Putting the Classical Kernel actually into Plugins ... Isabelle2009 - 10 (!)

...

„Θ“ thycontexts = contexts + {
sign : Signature,
thm_db : name ⇀ thm
...}

„$\Gamma \vdash_\Theta \varphi$“ thm = {context : CertId,
promises: name ⇀ thm future,
hyps : term,
prop : term}

"holes" in proofs to be filled in asyn-chronously later

CertificateTable : CertId ⇀ thycontext

status :: thm ⇀ { failed : bool, oracle: bool, unfinished: bool}

...

Options for FA

- modeling implementation of Kernel directly ?

- lot of stuff:
  - type classes
  - proof objects
  - internal details of futures ...

  Would make formalization difficult to understand and too Isabelle-specific ...

Reminder: Methods and Goals

- We do not attempt pervasive verification ! (50000 lines of code . . .)

- Instead: We model key-aspects/mechanisms of the entire system architecture (we have the domain experts in the project !!!) and verify key properties.

- side-effect: we hope to learn better abstractions for interfaces ...

- where "unknown environment assumptions" have to be made, we use test-generation techniques!

Wenzels Kernel Model [Damp 09]:

A) The synchronous part I ("LCF Kernel")

$$\frac{\Theta, \Pi, \Gamma \vdash q : B}{\Theta, \Pi, \Gamma - \{p : A\} \vdash (\lambda p : A.\ q) : (A \Longrightarrow B)} \ (\textit{imp-intro})$$

$$\frac{\Theta_1, \Pi_1, \Gamma_1 \vdash p : (A \Longrightarrow B) \quad \Theta_2, \Pi_2, \Gamma_2 \vdash q : A}{\Theta_1 \cup \Theta_2, \Pi_1 \cup \Pi_2, \Gamma_1 \cup \Gamma_2 \vdash p\ q : B} \ (\textit{imp-elim})$$

$$\frac{\Theta, \Pi, \Gamma \vdash p : (\bigwedge x.\ B[x])}{\Theta, \Pi, \Gamma \vdash p\ a : B[a]} \ (\textit{all-elim})$$

$$\frac{\Theta, \Pi, \Gamma \vdash p[x] : B[x] \quad x \notin \mathrm{FV}\,\Gamma}{\Theta, \Pi, \Gamma \vdash (\lambda x.\ p[x]) : (\bigwedge x.\ B[x])} \ (\textit{all-intro})$$

Wenzels Kernel Model:

A) The synchronous part II ("LCF Kernel")

$$\frac{}{\Theta, \Pi, \{p : A\} \vdash p : A} \ (assm)$$

$$\frac{(c : A[?\overline{\alpha}]) \in \Theta}{\Theta, \emptyset, \emptyset \vdash c : A[\overline{\tau}]} \ (axiom)$$

Wenzels Kernel Model:

B) The asynchronous part

$$\frac{\mathrm{FV}\,A = \emptyset \quad \mathrm{TV}\,A = \{?\overline{\alpha}\}}{\Theta, \{a : A\}, \emptyset \vdash a[?\overline{\alpha}] : A[?\overline{\alpha}]}\ (\textit{promise})$$

$$\frac{\Theta_1, \Pi_1, \Gamma \vdash p : B \quad \Theta_2, \Pi_2, \emptyset \vdash q : A \quad \Theta_2 \subseteq \Theta_1 \quad \Pi_2 \ll a}{\Theta_1, (\Pi_1 - \{a : A\}) \cup \Pi_2, \Gamma \vdash p[a := q] : B}\ (\textit{fulfill})$$

Approaches to formalize Wenzels Kernel Model:

- **Untyped Syntactic Model of Lambda Calculus**
  - **Basis:** datatype lam = * | ⊞ | lam -> lam
    | Var nat
    | Lam lam lam
    | App lam lam

- **Untyped Semantic Model**
  Semantic domain D in HOLCF.

- **Typed Syntactic Model**
  - **Basis (?): Eg. Nipkow/Narrascheski 96 (AFP)**

The Syntactic Model

(Based on Nipkow Naraschewski AFP MiniML 96) ?

```
header "MiniML-types and type substitutions"

theory Type
imports Maybe
begin

-- "type expressions"
datatype "typ" = TVar nat | TCons nat "typ list"

fun free_tv :: "typ ⇒ nat set"
where  "free_tv (TVar x) = {x}"
       | "free_tv (TCons c C) = Union(set(map free_tv C))"
```

Problems of the Syntactic Model

+ Idea: The key is a let-elimination theorem
  à la Gentzen Hauptsatz.

- Extension to "parametric polymorphism"
  necessary ? Or just desirable ?
- Wenzel's Model assumes α/β congruence ?
- Curry Style vs. Church Style ?

Problems of the Syntactic Model

- Open problems: << ? id - ordering ?
  context subsumption tests ?
- Open problems:
  - no context formation ...
  - should we prove async provability in
    a world with VARIOUS global contexts?