The ANR Project

Paral-ITP

FA: Relative Asynchronous Kernel Correctness: Model, Proof Architecture, and Progress.

Burkhart Wolff

Université Paris-Sud, LRI, Sept 2012

ANR-11-INSE-001

Motivation

Goals of “Taskgroup Formal Analysis”

- Accompany the development process by formalizing chosen aspects of the system architecture.

- Models and correctness proofs of key elements should help our development AND helping to sell the new architecture to the sceptics and LCF purists

…

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,

Parallel Nano-Kernel LCF - Architecture:

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

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

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

CertificateTable : CertId ß thycontext

status :: thm ß {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 ß thm,
...}

„Γ H$_\Theta$ ϕ“ thm = {context : CertId,
promises: name ß thm future,
hyps : term,
prop : term}

CertificateTable : CertId ß thycontext

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

...

“holes” in proofs to be filled in asyn-chronously later

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 ...

Wenzels Kernel Model[Damp 09]:

A) The synchronous part I ("LCF

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

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

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

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

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

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

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

Approaches to formalize Wenzels Kernel Model:

- **GTS like Syntactic Model of Lambda Calculus**
  Basis: Barendregt, Geuvers, et al.

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

- **Typed Syntactic Model Church Style**
  Basis: Berghofers AFP theory on $F_{\leq}$

- **Typed Syntactic Model Curry Style**
  Basis: Nipkow/Narrascheski 96 AFP theory on MiniML

The Syntactic Model

(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))"
```

The Syntactic Model

(AFP MiniML 96) ?

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

datatype type_scheme = FVar nat
                     | BVar nat
                     | SFun type_scheme type_scheme
                                                  (infixr "→" 70)

-- "expressions"
datatype
  expr = Var nat | Abs expr | App expr expr | LET expr expr

(*expr = Var nat | Abs expr | App expr expr | TApp expr
type_scheme | LET expr expr
*)
```

Problems of the Syntactic Model

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

- Consequences for dependent type systems ?
(Proof checking with promises of the form
$T \equiv T'$ **)?**

- Portability for Coq

- Model incomplete : binding,

Problems of the Syntactic Model

- Open problems: << ? id - ordering ?
  context subsumption tests ?
  [Makarius: transfer with empty K would
  suffice...]
- Open problems:
  - no context formation rules ...
    = critical pre-requisite for semantic
    interpretation of document model.
  - should we prove async provability