Virtualization of HOL4 in Isabelle

Fabian Immler[1], Jonas Rädle[2], Makarius Wenzel

[1]Carnegie Mellon University

[2]Technische Universität München

ITP, 9 September 2019

Motivation

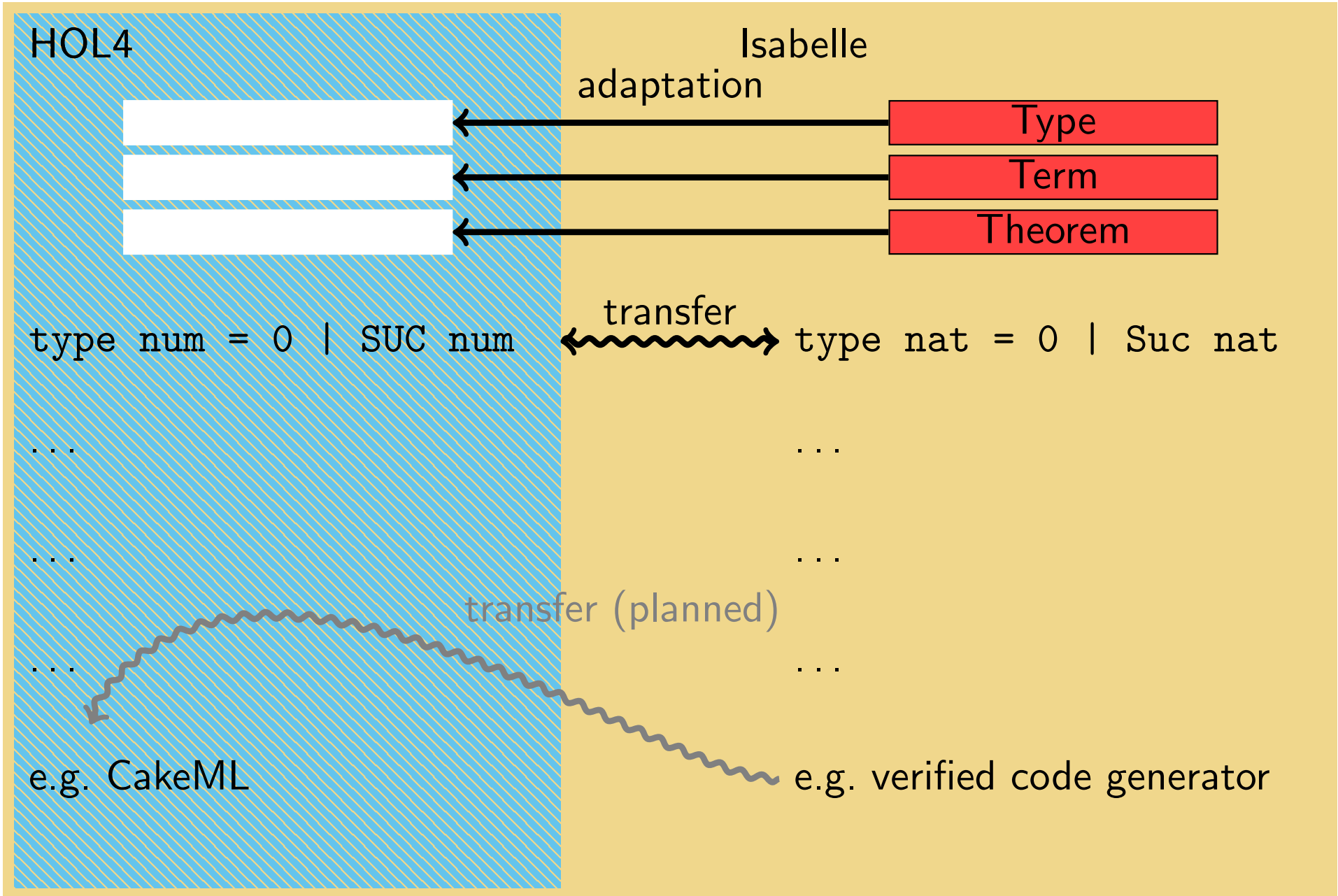
- Goal: Interoperability between HOL4 and Isabelle/HOL
- Reuse results proved in one system in the other
- Aim for high-level results
- Use case: Verified code generator for Isabelle/HOL [Hupel and Nipkow,2018]
  - Produces CakeML [Kumar et al.,2014] ASTs
  - Could be connected to compiler correctness proofs imported from HOL4
  - Eventually run CakeML compiler inside Isabelle to close verification gap

Related Work

- HOL Light importer for Isabelle/HOL [Kaliszyk and Krauss,2013]
- OpenTheory [Hurd,2009] aims for interoperability among the HOL family
- Based on kernel traces

Idea

- Isabelle and HOL4 implemented in PolyML
  - Run HOL4 inside Isabelle's run-time environment
- HOL4 has a small, modular logical kernel
  - Replace with a kernel that acts as a proxy to Isabelle


HOL4
Isabelle
adaptation
Type
Term
Theorem
transfer
type num = 0 | SUC num
type nat = 0 | Suc nat
...
...
...
...
transfer (planned)
...
...
e.g. CakeML
e.g. verified code generator

Virtualization – Idea

- Goal: Run HOL4 inside Isabelle's run-time environment
- Challenge: stateful excecution model of HOL4 vs. purely functional context management in Isabelle
- Solution: Provide an ML environment in which HOL4's global state is virtualized

Virtualization – Implementation

- Named ML environments to support ML applications within Isabelle
  - Separate namespace for HOL4
  - Mechanism to copy values to other environments
- Turn global mutable references into variables managed in Isabelle context
  - Unlike ML references these variables are not garbage collected
  - For performance, locally used references need to be marked manually

Kernel Adaptation

- Implement HOL4's type and term signatures

```
datatype preterm =
Const of string * typ
| Free  of string * typ
| Bound of int
| Abs   of string * typ * preterm
| $     of preterm * preterm
| Var   of indexname * typ
```

Preterms in Isabelle/Pure

```
datatype term =
Const of kernelid * hol_type
| Fv    of string * hol_type
| Bv    of int
| Abs   of term * term
| Comb  of term * term
| Clos  of term Subst.subs * term
```

Terms in HOL4 standard kernel

Adaptation – Types and Terms

- Crucial to precisely match behavior of HOL4 kernel (including e.g. naming)
- Isabelle uses abstract types `ctyp` and `cterm`
  - Avoid recertification where possible
- Delayed substitutions via closures not yet implemented

Adaptation – Theorems

- Defer inferences to primitives in Isabelle kernel
- Avoid unification by using manual instantiations
  - Predictable results and lower performance overhead
- Avoid increasing trusted code base
  - Map HOL4 axioms to theorems proved in Isabelle
  - Route type and constant definitions through the Isabelle kernel

Adaptation – Theorems

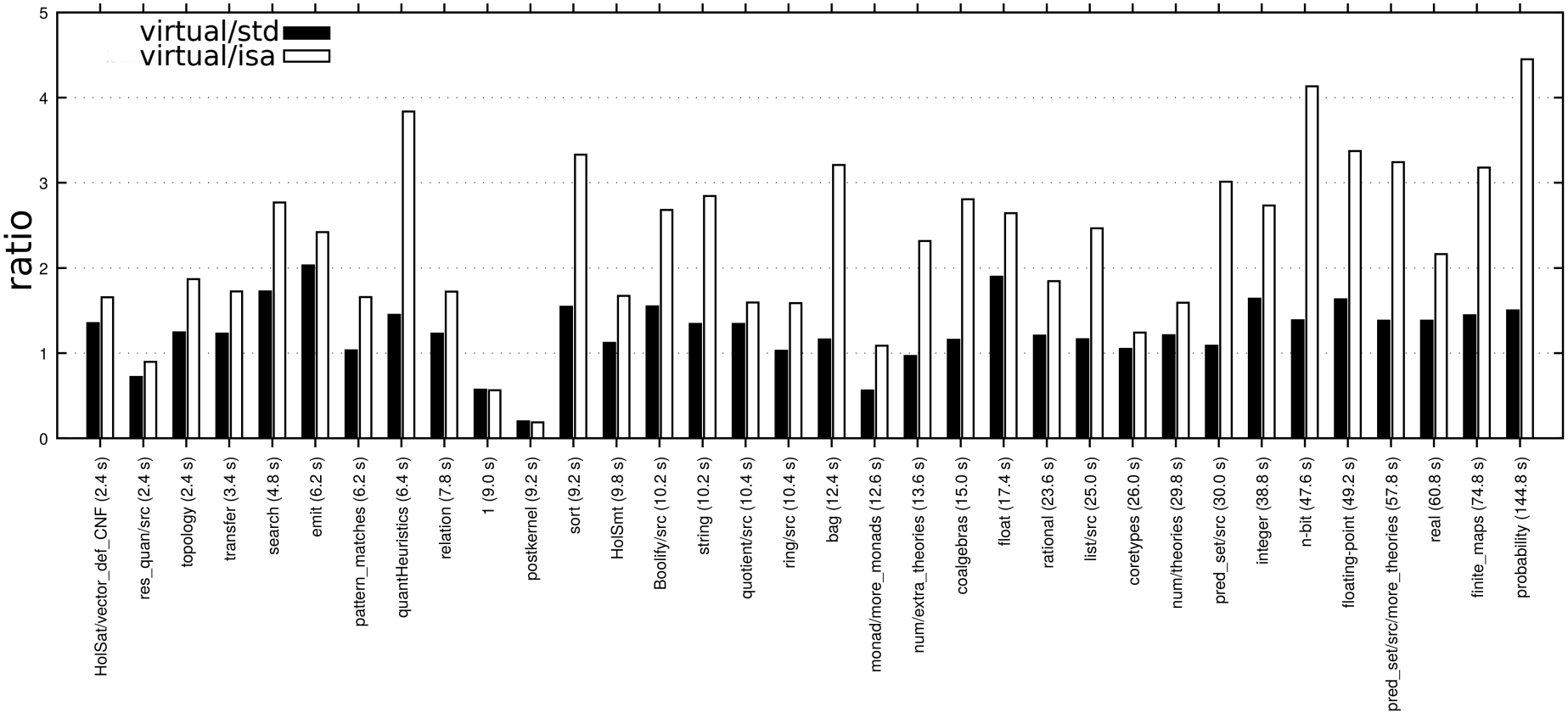
TRANS: $A_1 \vdash r = s \Longrightarrow A_2 \vdash s = t \Longrightarrow A_1 \cup A_2 \vdash r = t$

```
fun TRANS thm1 thm2 =
 let
   val (r, s) = dest_binop_thm thm1
   val (_, t) = dest_binop_thm thm2
   val cty = ctyp_of_cterm r
   val ty = typ_of cty
 in
  implies_elim
   (implies_elim
     (Thm.instantiate
       ([((("'a", 0), Isabelle.sort), cty)],
       [((("r",0),ty),r), ((("s",0),ty),s), ((("t",0),ty),t)]
     Isabelle.trans) thm1) thm2
 end
```

Build Process

- Emulation of Holmake
- "Does the right thing" most of the time
- Build unmodified HOL4 scripts
- Built so far
  - `core`, `more` and `large` build sequences from the HOL4 standard distribution
  - Some examples from the HOL4 standard distribution
  - CakeML semantics and associated proofs

Performance



Transfer – Idea

- Apart from axiomatization, HOL4 theories are imported as is
- Establish isomorphisms between definitions
- Transport theorems along these isomorphisms

$\Longrightarrow$ `transfer` package [Huffman and Kunčar, 2013]

Transfer – Example

- Fermat's last theorem in HOL4:

```
val FERMAT = boolLib.store_thm ("FERMAT",
  ``! a b c n. SUC (SUC 0) < n ==> ~(SUM (MAP (\x. SUC x ** n) [a; b]) = SUC c ** n)``
  fn g => (* Proof omitted *)
```

- Transfer to Isabelle:

```
lemma fermat_isabelle:
  "Suc (Suc 0) < n ⟶ (∑x←[a, b]. Suc x ^ n) ≠ Suc c ^ n"
  using [[hol4_thm ‹fermatTheory.FERMAT›, untransferred]]
  by simp
```

- Transfer in the other direction:

```
lemma fermat_hol4:
  "HOL4‹! a b c n. SUC (SUC 0) < n ==>
    ~(SUM (MAP (\\x. SUC x ** n) [a; b]) = SUC c ** n)›"
  by transfer (use fermat_isabelle in simp)
```

Conclusion

- Runs on recent mainline versions of HOL4 and Isabelle
- Same trusted code base as Isabelle/HOL
- Able to build large developments (e.g. CakeML semantics) with reasonable overhead
- Future goal: Use transfer methodology to connect developments

Thank You!