Translating HOL4 Definitions into ACL2

Rump Session, ACL2 Workshop 2025

Matt Kaufmann;
joint work with Konrad Slind

May 13, 2025

Outline

OUTLINE

Introduction

This work is in the early stages.
How far we go with it may depend on demand.

It builds on work reported in a related rump session talk on integrating set theory with ACL2.

Like that work, no trust tags or ACL2 changes support this work.

ABSTRACT

Abstract. *We report on preliminary work on translating HOL4 definitions into ACL2. This work takes advantage of a recent integration of set theory with ACL2. This work may provide a key step towards being able to reason about HOL4 and ACL2 formalizations in a common ACL2 environment.*

Related Work

Various efforts have been made to provide ways to use more than one proof assistant in a common environment.

But the primary effort for combining ACL2 and HOL was an embedding of ACL2 into HOL4.
(Today: The other way around, **HOL4 into ACL2**!)

That tool has undergone bit rot, though relevant ACL2 code has been updated (in `books/projects/acl2-in-hol/`).

Michael J. C. Gordon, Warren A. Hunt, Jr., Matt Kaufmann, and James Reynolds. An Embedding of the ACL2 Logic in HOL. *Proceedings of ACL2 Workshop 2006*, August, 2006. ACM Digital Library URL http://portal.acm.org/toc.cfm?id=1217975.

Michael J. C. Gordon, James Reynolds, Warren A. Hunt, Jr., and Matt Kaufmann. An Integration of HOL and ACL2. *Proceedings of Formal Methods in Computer-Aided Design (FMCAD'06)* (A. Gupta and P. Manolios, editors). IEEE Computer Society Press, pp. 153-160, November, 2006.

Michael J. C. Gordon, Matt Kaufmann, and Sandip Ray. The Right Tools for the Job: Correctness of Cone of Influence Reduction Proved Using ACL2 and HOL4. *Journal of Automated Reasoning*, Volume 47, Number 1, Springer, 2011, pp. 1–16, DOI 10.1007/s10817-010-9169-y.

Outline

SUMMARY OF TRANSLATION APPROACH

See community books directory `projects/hol-in-acl2/` for details.

HOL values are value-type pairs with respect to a *HOL type alist*, as recognized by the predicate `hpp` (“hol pair p”). Example:

```
(include-book "projects/hol-in-acl2/acl2/hol"
              :dir :system)

(in-package "ZF")

(defun hta0 () ; redundant
  (acons :num (omega)
         (acons :bool (pair t nil)
                nil)))

(thmz (hpp '(3 . :num) (hta0))
      :hints
      (("Goal" :in-theory (enable hol-type-eval))))
```

SEMANTICS

Function `hol-type-eval` assigns a set to a type. Then:

```
(defun hol-valuep (x type hta)
  (declare (xargs :guard
                  (and (symbol-alistp hta)
                       (hol-typep type hta t))))
  (in x (hol-type-eval type hta)))
```

We can form the *set* of all HOL values because they are all contained in a set.

```
(defthmz hol-valuep-implies-in-v-omega*2
  (implies (and (in hta (v-omega*2))
                (hol-typep type hta t)
                (hol-valuep x type hta))
           (in x (v-omega*2)))
  :props ...)
```

Book layout

The book `examples/ex1.lisp` has translation examples and we'll look at one shortly. That file has the following shape.

```
(in-package "HOL")
(include-book "../acl2/theories")

; Define EX1$HTA and :HOL-THEORY table:
(open-theory ex1)

; Populate that table with automatic translations:
(defhol ...) (defhol ...) ... (defhol ...)

; Generate encapsulate with ex1$prop
; hypothesis function and translations:
(close-theory)

; Exhibit translations as theorems:
(set-enforce-redundancy t)
(defthm ...) (defthm ...) ... (defthm ...)
```

Outline

Here is a HOL definition of the function `FST`, which takes the first component of an ordered pair.

```
val FST = ⊢ ∀(x :α) (y :β). FST (x,y) = x: thm
```

Automatic translation produces:

```
(defhol
  :fns ((fst (:arrow* (:hash a b) a)))
  :defs ((:forall
          ((x a) (y b))
          (equal (hap* (fst (typ (:arrow* (:hash a b)
                                           a)))
                       (hp-comma x y))
                 x)))))
```

The next slide repeats that form and shows what is generated from it.

```
val FST = ⊢ ∀(x :α) (y :β). FST (x,y) = x: thm
(defhol
  :fns ((fst (:arrow* (:hash a b) a)))
  :defs ((:forall
          ((x a) (y b))
          (equal (hap* (fst (typ (:arrow* (:hash a b)
                                            a)))
                        (hp-comma x y))
                 x))))
(DEFTHM HOLFST
  (IMPLIES
   (AND (HPP X HTA) (EQUAL (HP-TYPE X) (TYP A))
        (HPP Y HTA) (EQUAL (HP-TYPE Y) (TYP B))
        (ALIST-SUBSETP (EX1$HTA) HTA)
        (FORCE (EX1$PROP)))
   (EQUAL (HAP* (FST (TYP (:ARROW* (:HASH A B)
                                    A)))
                (HP-COMMA X Y))
          X)))
```

OUTLINE

Conclusion

Again, this is **early-stage** work.
Future tasks may include the following.

- Translate HOL4 theorems and type definitions, not just function definitions.
- Extend translation to handle quantifiers and lambdas (by lifting).
- Do a small proof example, e.g., correspondence of trivial expression evaluators written in HOL and in ACL2.
- Improve robustness (better error messages, extend existing term checks, ...).
- Formalize and prove soundness of the translation approach.

We tend to be application-driven. So....
It will inspire us to have users of this work!