An Integration of Axiomatic Set Theory with ACL2

Rump Session, ACL2 Workshop 2025

Matt Kaufmann

UT Austin (retired)

May 13, 2025

Outline

Outline

Abstract

This talk will summarize work to date on enabling the use of ACL2 to `prove theorems of ZFC set theory`*, with applications to reasoning about notions typically considered 'higher-order". This work resides in the community books directory* `books/projects/set-theory/` *and is documented in topics ZFC and ZFC-MODEL.*

A Few Notes

For slides and video recordings of three 1.5-hour talks, see the ACL2 Seminar page.

This is **work in progress** (e.g., no comparison with Isabelle/ZF or others).

- Collaborators welcome!

The books use no trust tags and the work required **no ACL2 changes**.

Motivation

Zermelo Fraenkel (ZF) set theory is an established, intuitive foundation for mathematics.

Personal motivation: Combines my interest in set theory (including related research done more than 40 years ago!) with my current focus, ACL2.

Key new insight last Fall:
Define ACL2 primitives and data in pure set theory.

Additional motivation: Provides a way to embed higher-order logic (HOL) developments into ACL2.
(See next talk.)

Outline

SET THEORY AS FOUNDATION FOR ACL2 (1)

```
(encapsulate ; file base.lisp
  (((zfc) => *)          ; Hypothesis function
   ((in * *) => *)       ; (in a x) ⇔ a ∈ x
   ((pair * *) => *)     ; (pair x y) = {x,y}
   ((min-in *) => *)     ; (min-in x) is ∈-minimal in x
   ((union *) => *)      ; (union x) = ⋃{a: a ∈ x}
   ((omega) => *)        ; the set of natural numbers
   ((powerset *) => *) ; (powerset x) = {y: y ⊂ x}
   )
  (local (defun zfc () nil)) ; standard trick
  ...
  (defthm infinity
    (implies (zfc)
             (equal (in n (omega))
                    (natp n))))
  ...)
```

Set Theory As Foundation for ACL2 (2)

I lied, but only a little. We actually force the hypothesis.

```
(defthm infinity
  (implies (force (zfc))
           (equal (in n (omega))
                  (natp n))))
```

But here's the actual event, which expands to what's above.

```
(defthmz infinity
  (equal (in n (omega))
         (natp n)))
```

A metatheoretic argument provides a set-theoretic interpretation for which `(zfc)` is true...
so `defthmz` like `defthm` but for set theory.

Set Theory As Foundation for ACL2 (3)

Macros:

- `zsub` implements the Comprehension (Subset) Schema of ZF; and
- `zfn` implements the Replacement (Collection) Schema of ZF.

Example:

```
(zsub domain ...) ; introduces domain and domain$prop
(defthmz domain-union2
    (equal (domain (union2 r s))
           (union2 (domain r) (domain s)))
    :props (zfc domain$prop)
    :hints ...)
```

Note the hypothesis functions, which we view as true:
the metatheoretic argument extends for `zsub` and `zfn`.

Set Theory As Foundation for ACL2 (4)

ACL2 objects are defined as sets.

- `Cons` is represented using the Kuratowski ordered pair:
  `(cons x y)` $= \{\{x\}, \{x, y\}\}$.
  (This is actually a disabled rewrite rule exported from the initial `encapsulate` event.)

- Natural numbers are Zermelo (von Neumann) ordinals:
  0 is the empty set, $\{\}$;
  $1 = \{0\}$;
  and in general
  $n = \{0, 1, ..., n-1\}$.
- Other atoms are represented as tagged triples $\{0, n, val\}$
  - $-3 = \{0, 1, (3 \,.\, 0)\}$

Outline

Higher-Order Capabilities (1)

Function `zf::apply` is an alternative to `apply$`.
Example: if f is $\{\langle 1, 2\rangle, \langle 3, 4\rangle \langle 5, 6\rangle\}$, then `(apply f 3)` = 4.

Here is an example (in the "`ZF`" package).

```
(defun map (f lst)
  (declare (xargs :guard (true-listp lst)))
  (cond ((endp lst) nil)
        (t (cons (apply f (car lst))
                 (map f (cdr lst))))))

(defthm map-append
  (equal (map f (append x y))
         (append (map f x) (map f y))))
```

Higher-Order Capabilities (2)

An example involving computation:

```
(defun fib (n)
  (declare ...)
  (cond ((zp n) 0)
        ((= n 1) 1)
        (t (+ (fib (- n 1)) (fib (- n 2))))))

(zify zfib fib :dom (omega) :ran (omega))

(thmz (equal (map (zfib) '(0 1 2 3 4 5 6 7))
             '(0 1 1 2 3 5 8 13))
      :props (zify-prop zfib$prop))
```

For more higher-order programming and proving, see `foldr.lisp`.

Outline

Some Set Theory Developed in ACL2 So Far

- ordinals, including trichotomy
- de Morgan's laws and other theorems on set algebra
- transfinite induction
- transitive closure (closure under set membership)
- finite sequences
- iterated composition
- function spaces
- Cantor's Theorem (no surjection onto powerset)
- the Schröder-Bernstein Theorem (using Grant Jurgensen's ACL2 formalization and proof)

Outline

Future Work (Highly Incomplete List!)

- Transfinite recursion, e.g., V_α for all ordinals α
- Cardinals, cardinality
- Higher-order applications (e.g., temporal logics)
- More automation
  - ACL2 modification for parity-based rewriting (or maybe use *pick-a-point* clause-processor)
  - Quantifier instantiation (maybe Dave Greve's stuff?)
  - Automated functional instantiation (maybe Joosten et al.'s 2013 workshop paper on `instance-of-defspec`)
- More set theory
  - Uncountable cardinals (then cofinality, stationary sets, ...)
  - Addiitonal easy theorems (e.g., Mostowski collapse)
  - Additional difficult theorems (e.g., independence of CH)
  - ...
- Other math, e.g., point-set topology

Again:
Contact me if you're interested in collaborating on this work.

Thank you.