A Proof of the Schröder-Bernstein Theorem in ACL2

Grant Jurgensen

Kestrel Institute

May, 2025

Outline

Introduction

Theorem 1 (Schröder-Bernstein)

If there exists an injection $f : P \rightarrow Q$ and an injection $g : Q \rightarrow P$, then there must exist a bijection $h : P \rightarrow Q$.

- Theorem #25 in Dr. Freek Wiedijk's "Formalizing 100 Theorems."
- It has been proved in many other theorem provers, but not in any of the Boyer-Moore family.
- The proof is interesting, requiring extensive use of quantifiers.
- Find it in the community books: `projects/schroeder-bernstein`.

.

The Informal Proof

A Theory of Chains

Let $f : P \to Q$ and $g : Q \to P$ be our two injections.

Definition 2

A **chain** $C \subseteq P \cup Q$ is a set of elements which are mutually reachable via repeated application of f and g, or their inverses.

For instance, the element $p \in P$ is a member of the chain:

$$\{\ldots,\ f^{-1}(g^{-1}(p)),\ g^{-1}(p),\ p,\ f(p),\ g(f(p)),\ \ldots\}$$

and $q \in Q$ belongs to the chain:

$$\{\ldots,\ g^{-1}(f^{-1}(q)),\ f^{-1}(q),\ q,\ g(q),\ f(g(q)),\ \ldots\}$$

Types of Chains

1. **Cyclic chains**: After some finite number of steps, the chain cycles back to a previous element.
2. **Infinite chains**: All acyclic chains are (countably) infinite. Infinite chains all extend infinitely in the "rightward" direction and may be further subdivided into two categories:
    1. **Non-stoppers**: Such chains extend infinitely in the leftward direction in addition to the rightward direction.
    2. **Stoppers**: Such chains do *not* extend infinitely leftward and may therefore be said to possess an **initial** element. On such an element, neither f^{-1} nor g^{-1} is defined (i.e., the element is not in the image of f or g).

We refer to chains with initial elements in P as "P-stoppers" and those with initial elements in Q as "Q-stoppers."

Ordering of Chain Elements

An ordering is implied from our previous example chains.

$$\frac{p \in P}{p \sqsubseteq f(p)} \qquad \frac{q \in Q}{q \sqsubseteq g(q)}$$

$$\text{Reflexivity}\,\frac{}{x \sqsubseteq x} \qquad \text{Transitivity}\,\frac{x \sqsubseteq y \quad y \sqsubseteq z}{x \sqsubseteq z}$$

- $\sqsubseteq$ forms a preorder.
- Initial elements are minimal w.r.t. $\sqsubseteq$.
- $chain(x) = chain(y)$ holds if and only if $x \sqsubseteq y$ or $y \sqsubseteq x$.

Definition of the Bijection

Let $stoppers_Q$ denote the set of Q-stoppers. Then we define our proposed bijection h:

$$h(p) = \begin{cases} g^{-1}(p) & \text{if } chain(p) \in stoppers_Q \\ f(p) & \text{otherwise} \end{cases}$$

- We had multiple options in our definition of h.
- When $chain(p)$ is cyclic or a non-stopper, either f or g^{-1} could be used. We opt to use f for convenience.

Lemmas

Lemma 3

Let $p \in P$ and $chain(p) \in stoppers_Q$. Then p is in the image of g.

Proof.

By the definition of a Q-stopper, the initial element of $chain(p)$ resides in Q. Since the initial element is unique and $p \notin Q$, p must not be initial. Therefore, it is by definition in the image of g. □

Lemma 4

Let $q \in Q$ and $chain(q) \notin stoppers_Q$. Then q is in the image of f.

Proof.

Similar to the above. □

Lemmas

Lemma 5

Let $p \in P$. Then $chain(h(p)) = chain(p)$.

Proof.

Either $h(p) = g^{-1}(p)$ or $h(p) = f(p)$. By definition, p is in the same chain as $f(p)$ as well as $g^{-1}(p)$, if it is defined. $\square$

Injectivity

Lemma 6 (Injectivity of h)

Let $p_0, p_1 \in P$, where $h(p_o) = h(p_1)$. Then $p_0 = p_1$.

Proof.

Case 1: $h(p_0)$ is in a Q-stopper.
By equality, $h(p_1)$ is also in a Q-stopper. By Lemma 5, so are p_0 and p_1. By definition, we have $h(p_0) = g^{-1}(p_0)$ and $h(p_1) = g^{-1}(p_1)$. From $h(p_0) = h(p_1)$, we get $g^{-1}(p_0) = g^{-1}(p_1)$. Applying g yields $p_0 = p_1$.

Case 2: $h(p_0)$ is not in a Q-stopper.
$h(p_1)$, p_0, and p_1 are also not in Q-stoppers. By definition, we then have $h(p_0) = f(p_0)$ and $h(p_1) = f(p_1)$. From $h(p_0) = h(p_1)$, we get $f(p_0) = f(p_1)$. By injectivity of f, we have $p_0 = p_1$. $\square$

Surjectivity

Lemma 7 (Surjectivity of h)

Let $q \in Q$. Then there exists $p \in P$ such that $h(p) = q$.

Proof.

Case 1: q is in a Q-stopper.
Then $g(q)$ is also in a Q-stopper by definition. Let $p = g(q)$. Then:
$h(p) = h(g(q)) = g^{-1}(g(q)) = q$.

Case 2: q is not in a Q-stopper.
By Lemma 4, $f^{-1}(q)$ is well-defined. Since q is not in a Q-stopper, neither is $f^{-1}(q)$. Let $p = f^{-1}(q)$. Then: $h(p) = h(f^{-1}(q)) = f(f^{-1}(q)) = q$. $\square$

ACL2 Formalization

Setup

Initial Definitions

```
(encapsulate (((f *) => *) ((g *) => *) ((p *) => *) ((q *) => *))
  ;; Definitions omitted

  (defrule q-of-f-when-p
    (implies (p x) (q (f x))))

  (defrule injectivity-of-f
    (implies (and (p x) (p y)
                  (equal (f x) (f y)))
             (equal x y)))

  (defrule p-of-g-when-q
    (implies (q x) (p (g x))))

  (defrule injectivity-of-g
    (implies (and (q x) (q y)
                  (equal (g x) (g y)))
             (equal x y))))
```

Function Inverses

We introduce the `definverse` macro to quickly introduce function inverses. The macro event `(definverse f :domain p :codomain q)` generates definitions:

```
(define is-f-inverse (inv x)
  (and (p inv)
       (q x)
       (equal (f inv) x)))

(defchoose f-inverse (inv) (x)
  (is-f-inverse inv x))

(define in-f-imagep (x)
  (is-f-inverse (f-inverse x) x))
```

Function Inverses

...and theorems:

```
(defrule in-f-imagep-of-f-when-p
  (implies (p x)
           (in-f-imagep (f x))))

(defrule p-of-f-inverse-when-in-f-imagep
  (implies (in-f-imagep x)
           (p (f-inverse x))))

(defrule f-inverse-of-f-when-p ;; Left inverse
  (implies (p x)
           (equal (f-inverse (f x)) x)))

(defrule f-of-f-inverse-when-in-f-imagep ;; Right inverse
  (implies (in-f-imagep x)
           (equal (f (f-inverse x)) x)))
```

Recognizer, constructor, and accessors

```
(define chain-elemp (x)
  (and (consp x)
       (booleanp (car x))
       (if (car x)
           (and (p (cdr x)) t)
         (and (q (cdr x)) t))))

(define chain-elem (polarity val) ;; Construct a chain element
  (cons (and polarity t) val))

(define polarity ((elem consp)) ;; Get the polarity of a chain element
  (and (car elem) t))

(define val ((elem consp)) ;; Get the value of a chain element
  (cdr elem))
```

Chain Ordering

chain<= corresponds to the previously introduced $\sqsubseteq$ order.

```
(define chain-step ((elem consp))
  (let ((polarity (polarity elem)))
    (chain-elem (not polarity)
                (if polarity
                    (f (val elem))
                  (g (val elem))))))

(define chain-steps ((elem consp) (steps natp))
  (if (zp steps)
      elem
    (chain-steps (chain-step elem) (- steps 1))))

(define-sk chain<= ((x consp) y)
  (exists n
    (equal (chain-steps x (nfix n))
           y)))
```

Chain Equivalence

Instead of $chain(x) = chain(y)$, we say `(chain= x y)`.

```
(define chain= ((x consp) (y consp))
  (if (and (chain-elemp x)
           (chain-elemp y))
      (or (chain<= x y)
          (chain<= y x))
    (equal x y)))

(defequiv chain=)
```

Initial Elements and Q-Stoppers

```
(define initialp ((elem consp))
  (if (polarity elem)
      (not (in-g-imagep (val elem)))
    (not (in-f-imagep (val elem)))))

(define initial-wrt ((initial consp) (elem consp))
  (and (chain-elemp initial)
       (initialp initial)
       (chain<= initial elem)))

(defchoose get-initial (initial) (elem)
  (initial-wrt initial elem))

(define exists-initial ((elem consp))
  (initial-wrt (get-initial elem) elem))

(define in-q-stopper ((elem consp))
  (and (exists-initial elem) (not (polarity (get-initial elem)))))
```

The Bijective Witness

- `sb-witness` corresponds to the function h in our informal proof.
- In this version we must explicitly tag `x` with its polarity.

```
(define sb-witness (x)
  (if (in-q-stopper (chain-elem t x))
      (g-inverse x)
    (f x)))
```

Supporting Lemmas

```
(defrule in-g-imagep-when-in-q-stopper
  (implies (and (in-q-stopper elem)
                (polarity elem))
           (in-g-imagep (val elem))))

(defrule in-f-imagep-when-not-in-q-stopper
  (implies (and (chain-elemp elem)
                (not (in-q-stopper elem))
                (not (polarity elem)))
           (in-f-imagep (val elem))))

(defrule chain=-of-sb-witness
  (implies (p x)
           (chain= (chain-elem t x)
                   (chain-elem nil (sb-witness x)))))
```

Final Theorems

```
(defrule q-of-sb-witness-when-p
  (implies (p x)
           (q (sb-witness x))))

(defrule injectivity-of-sb-witness
  (implies (and (p x) (p y)
                (equal (sb-witness x)
                       (sb-witness y)))
           (equal x y)))

(define-sk exists-sb-inverse (x)
  (exists inv
    (and (p inv)
         (equal (sb-witness inv) x))))

(defrule surjectivity-of-sb-witness
  (implies (q x)
           (exists-sb-inverse x)))
```

Conclusion

Conclusion

- We have formalized a well-known proof of the Schröder-Bernstein theorem in ACL2.
- See also Matt Kaufmann's adaptation to set theory: `projects/set-theory/schroeder-bernstein`.
- Thank you to the reviewers for their insightful comments.

Questions?