A challenge problem:
Toward better ACL2 proof technique

Matt Kaufmann
The University of Texas at Austin
Dept. of Computer Science, GDC 7.804

ACL2 Workshop 2015

October 1, 2015

INTRODUCTION

I took a break this summer to return to my roots as a mathematical logician, hosted by Prof. Ali Enayat of the University of Gothenburg, Sweden.

Introduction

I took a break this summer to return to my roots as a mathematical logician, hosted by Prof. Ali Enayat of the University of Gothenburg, Sweden.

- Lots of fun chats!

INTRODUCTION

I took a break this summer to return to my roots as a mathematical logician, hosted by Prof. Ali Enayat of the University of Gothenburg, Sweden.

- Lots of fun chats!
- We are co-authoring a tutorial paper on *iterated ultrapowers*.

Introduction

I took a break this summer to return to my roots as a mathematical logician, hosted by Prof. Ali Enayat of the University of Gothenburg, Sweden.

- Lots of fun chats!
- We are co-authoring a tutorial paper on *iterated ultrapowers*.
- A key lemma in that paper can be abstracted to a lemma about finite sequences, with a pretty simple hand proof.

INTRODUCTION

I took a break this summer to return to my roots as a mathematical logician, hosted by Prof. Ali Enayat of the University of Gothenburg, Sweden.

- Lots of fun chats!
- We are co-authoring a tutorial paper on *iterated ultrapowers*.
- A key lemma in that paper can be abstracted to a lemma about finite sequences, with a pretty simple hand proof.
- Why not prove the abstracted lemma in ACL2?

INTRODUCTION

I took a break this summer to return to my roots as a mathematical logician, hosted by Prof. Ali Enayat of the University of Gothenburg, Sweden.

- Lots of fun chats!
- We are co-authoring a tutorial paper on *iterated ultrapowers*.
- A key lemma in that paper can be abstracted to a lemma about finite sequences, with a pretty simple hand proof.
- Why not prove the abstracted lemma in ACL2?

Horrors!
It took me about 16 hours to complete that exercise in ACL2.

Introduction (page 2)

Possible conclusions:

Introduction (page 2)

Possible conclusions:

- I suck at using ACL2?

Introduction (page 2)

Possible conclusions:

- I suck at using ACL2?
- ACL2 sucks?

Introduction (page 2)

Possible conclusions:

- I suck at using ACL2?
- ACL2 sucks?
- There are ways to use ACL2 more productively that I didn't use.

INTRODUCTION (PAGE 2)

Possible conclusions:

- I suck at using ACL2?
- ACL2 sucks?
- There are ways to use ACL2 more productively that I didn't use.
  - Structured development methodologies?

INTRODUCTION (PAGE 2)

Possible conclusions:

- I suck at using ACL2?
- ACL2 sucks?
- There are ways to use ACL2 more productively that I didn't use.
  - Structured development methodologies?
  - More help from existing libraries?

INTRODUCTION (PAGE 2)

Possible conclusions:

- I suck at using ACL2?
- ACL2 sucks?
- There are ways to use ACL2 more productively that I didn't use.
  - Structured development methodologies?
  - More help from existing libraries?
  - Nicer formalization of the problem?

Introduction (page 2)

Possible conclusions:

- I suck at using ACL2?
- ACL2 sucks?
- There are ways to use ACL2 more productively that I didn't use.
  - Structured development methodologies?
  - More help from existing libraries?
  - Nicer formalization of the problem?
  - ...

INTRODUCTION (PAGE 3)

Goal for today:

> *Present a challenge to construct an ACL2 proof more efficiently and to present lessons learned . . .*

INTRODUCTION (PAGE 3)

Goal for today:

> *Present a challenge to construct an ACL2 proof more efficiently and to present lessons learned …*
> perhaps in a future ACL2 Workshop.

Introduction (page 3)

Goal for today:

> *Present a challenge to construct an ACL2 proof more efficiently and to present lessons learned …*
> perhaps in a future ACL2 Workshop.

In this talk I'll point you to relevant books and I'll also present a very informal hand proof.

The Challenge(s)

The community book `books/demos/proofs/tightness-lemma.lisp` contains:

The Challenge(s)

The community book `books/demos/proofs/tightness-lemma.lisp` contains:

- a self-contained informal proof (as a Lisp comment) using standard mathematical notation;

The Challenge(s)

The community book
`books/demos/proofs/tightness-lemma.lisp` contains:

- a self-contained informal proof (as a Lisp comment) using standard mathematical notation;
- `encapsulate` and `defun` events introducing the requisite notions; and

The Challenge(s)

The community book `books/demos/proofs/tightness-lemma.lisp` contains:

- a self-contained informal proof (as a Lisp comment) using standard mathematical notation;
- `encapsulate` and `defun` events introducing the requisite notions; and
- a statement of the final theorem.

The Challenge(s)

The community book `books/demos/proofs/tightness-lemma.lisp` contains:

- a self-contained informal proof (as a Lisp comment) using standard mathematical notation;
- `encapsulate` and `defun` events introducing the requisite notions; and
- a statement of the final theorem.

I'm putting forth the following challenges.

THE CHALLENGE(S)

The community book `books/demos/proofs/tightness-lemma.lisp` contains:

- a self-contained informal proof (as a Lisp comment) using standard mathematical notation;
- `encapsulate` and `defun` events introducing the requisite notions; and
- a statement of the final theorem.

I'm putting forth the following challenges.

- **Preferred challenge**: Do a better, faster job than the proof given in community book `books/demos/proofs/tightness-lemma-proof.lisp`. **NOTE**: It's OK to change the formalization!

The Challenge(s)

The community book
`books/demos/proofs/tightness-lemma.lisp` contains:

- a self-contained informal proof (as a Lisp comment) using standard mathematical notation;
- `encapsulate` and `defun` events introducing the requisite notions; and
- a statement of the final theorem.

I'm putting forth the following challenges.

- **Preferred challenge**: Do a better, faster job than the proof given in community book `books/demos/proofs/tightness-lemma-proof.lisp`. **NOTE**: It's OK to change the formalization!
- **Alternate challenge**: "Reverse engineer" that proof into one that shows how to complete such proofs more efficiently.

Very Informal Theorem Statement

I'll be sloppy here and using pictures, just to give the idea. A more careful hand proof is in the aforementioned `tightness-lemma.lisp` book.

Very Informal Theorem Statement

I'll be sloppy here and using pictures, just to give the idea. A more careful hand proof is in the aforementioned `tightness-lemma.lisp` book.

Assume that we have:

- a set I and strict total ordering $\prec$ on I;
- functions $f(s)$ and $g(s)$, on $\prec$-increasing sequences from I of length n_f and n_g, respectively; and
- a unary predicate P.

Very Informal Theorem Statement

I'll be sloppy here and using pictures, just to give the idea. A more careful hand proof is in the aforementioned `tightness-lemma.lisp` book.

Assume that we have:

- a set I and strict total ordering $\prec$ on I;
- functions $f(s)$ and $g(s)$, on $\prec$-increasing sequences from I of length n_f and n_g, respectively; and
- a unary predicate P.

The next slide illustrates the remaining assumptions for $n_f = 4$ and $n_g = 3$.

Very Informal Theorem Statement (2)

Very Informal Theorem Statement (2)

ASSUMPTIONS

VERY INFORMAL THEOREM STATEMENT (2)

ASSUMPTIONS

(d) If $f(s_1) = f(s_2)$ and all of s_1 precedes all of s_2, then $P(f(s_1))$:

(s_1) `a a a a`

(s_2) `b b b b`

Very Informal Theorem Statement (2)

ASSUMPTIONS

(d) If $f(s_1) = f(s_2)$ and all of s_1 precedes all of s_2, then $P(f(s_1))$:

```
(s1)    a a a a
(s2)              b b b b
```

(e) For disjoint sequences s_1 and s_2, the truth of the equation $f(s_1) = g(s_2)$ depends only on how s_1 and s_2 are interleaved.

```
(s1)  x x     x   x
(s2)      y y   y
```

Very Informal Theorem Statement (2)

ASSUMPTIONS

(d) If $f(s_1) = f(s_2)$ and all of s_1 precedes all of s_2, then $P(f(s_1))$:

```
(s1)    a a a a
(s2)              b b b b
```

(e) For disjoint sequences s_1 and s_2, the truth of the equation $f(s_1) = g(s_2)$ depends only on how s_1 and s_2 are interleaved.

```
(s1)  x x     x   x
(s2)      y y   y
```

(g) For two specific disjoint sequences s_f and s_g, $f(s_f) = g(s_g)$.

VERY INFORMAL THEOREM STATEMENT (2)

ASSUMPTIONS

(d) If $f(s_1) = f(s_2)$ and all of s_1 precedes all of s_2, then $P(f(s_1))$:

```
(s1)   a a a a
(s2)                b b b b
```

(e) For disjoint sequences s_1 and s_2, the truth of the equation $f(s_1) = g(s_2)$ depends only on how s_1 and s_2 are interleaved.

```
(s1)  x x     x   x
(s2)      y y   y
```

(g) For two specific disjoint sequences s_f and s_g, $f(s_f) = g(s_g)$.

CONCLUSION: $P(f(s_f))$.

Very Informal Proof Sketch

(d) If $f(s_1) = f(s_2)$ and all of s_1 precedes all of s_2, then $P(f(s_1))$:

```
(s1)    a a a a
(s2)              b b b b
```

(e) For disjoint sequences s_1 and s_2, the truth of the equation $f(s_1) = g(s_2)$ depends only on how s_1 and s_2 are interleaved.

```
(s1)  x x     x   x
(s2)      y y   y
```

(g) For two specific disjoint sequences s_f and s_g, $f(s_f) = g(s_g)$.

Very Informal Proof Sketch

(d) If $f(s_1) = f(s_2)$ and all of s_1 precedes all of s_2, then $P(f(s_1))$:

```
(s1)   a a a a
(s2)           b b b b
```

(e) For disjoint sequences s_1 and s_2, the truth of the equation $f(s_1) = g(s_2)$ depends only on how s_1 and s_2 are interleaved.

```
(s1) x x     x   x
(s2)     y y   y
```

(g) For two specific disjoint sequences s_f and s_g, $f(s_f) = g(s_g)$.

Plan: We will see how to derive $P(f(s_f))$ from (g) by applying (e) repeatedly and then (d).

Very Informal Proof Sketch

(d) If $f(s_1) = f(s_2)$ and all of s_1 precedes all of s_2, then $P(f(s_1))$:

```
(s1)   a a a a
(s2)             b b b b
```

(e) For disjoint sequences s_1 and s_2, the truth of the equation $f(s_1) = g(s_2)$ depends only on how s_1 and s_2 are interleaved.

```
(s1) x x     x   x
(s2)     y y   y
```

(g) For two specific disjoint sequences s_f and s_g, $f(s_f) = g(s_g)$.

Plan: We will see how to derive $P(f(s_f))$ from (g) by applying (e) repeatedly and then (d).

```
x x y y x y x
```

We wish to show $P(f(s_f))$.

We wish to show $P(f(s_f))$.
Below, all $f(s_f)$ and $g(s_g)$ equal the first $f(s_f)$ and $g(s_g)$:

We wish to show $P(f(s_f))$.
Below, all $f(s_f)$ and $g(s_g)$ equal the first $f(s_f)$ and $g(s_g)$:

`x x y y x y x`

We wish to show $P(f(s_f))$.
Below, all $f(s_f)$ and $g(s_g)$ equal the first $f(s_f)$ and $g(s_g)$:

`x x y y x y x`

`x x y y x y                                        x`

We wish to show $P(f(s_f))$.
Below, all $f(s_f)$ and $g(s_g)$ equal the first $f(s_f)$ and $g(s_g)$:

```
x x y y x y x

x x y y x y                                                x

x x y y x                                                y x
```

We wish to show $P(f(s_f))$.
Below, all $f(s_f)$ and $g(s_g)$ equal the first $f(s_f)$ and $g(s_g)$:

```
x x y y x y x

x x y y x y                         x

x x y y x                         y x

x x y y                         x y x
```

We wish to show $P(f(s_f))$.
Below, all $f(s_f)$ and $g(s_g)$ equal the first $f(s_f)$ and $g(s_g)$:

x x y y x y x	
x x y y x y	x
x x y y x	y x
x x y y	x y x
x x y	y x y x

We wish to show $P(f(s_f))$.
Below, all $f(s_f)$ and $g(s_g)$ equal the first $f(s_f)$ and $g(s_g)$:

```
x x y y x y x

x x y y x y                     x

x x y y x                     y x

x x y y                     x y x

x x y                     y x y x

x x                     y y x y x
```

We wish to show $P(f(s_f))$.
Below, all $f(s_f)$ and $g(s_g)$ equal the first $f(s_f)$ and $g(s_g)$:

x x y y x y x	
x x y y x y	x
x x y y x	y x
x x y y	x y x
x x y	y x y x
x x	y y x y x
x	x y y x y x

We wish to show $P(f(s_f))$.
Below, all $f(s_f)$ and $g(s_g)$ equal the first $f(s_f)$ and $g(s_g)$:

x x y y x y x	
x x y y x y	x
x x y y x	y x
x x y y	x y x
x x y	y x y x
x x	y y x y x
x	x y y x y x
	x x y y x y x

We wish to show $P(f(s_f))$.
Below, all $f(s_f)$ and $g(s_g)$ equal the first $f(s_f)$ and $g(s_g)$:

```
x x y y x y x
x x y y x y                x
x x y y x                y x
x x y y                x y x
x x y                y x y x
x x                y y x y x
x                x y y x y x
               x x y y x y x
```

Now let's erase all but the first and last lines. . .

x x y y x y x

x x y y x y x

x x y y x y x

x x y y x y x

Now let's erase each y. . .

x x x x

x x x x

So, we have the same value of $f(s_f)$ for the first and final s_f:

x x x x

x x x x

So, we have the same value of $f(s_f)$ for the first and final s_f:

```
    x x     x   x
                                          x x     x   x
```

But recall:

(d) If $f(s_1) = f(s_2)$ and all of s_1 precedes all of s_2, then $P(f(s_1))$:

(s_1) `a a a a`

(s_2) `b b b b`

So, we have the same value of $f(s_f)$ for the first and final s_f:

```
x x   x  x
```

```
                                          x x   x  x
```

But recall:

(d) If $f(s_1) = f(s_2)$ and all of s_1 precedes all of s_2, then $P(f(s_1))$:

(s_1) a a a a

(s_2) b b b b

So $P(f(s_f))$, as was to be shown!

Conclusion

For a more complete informal proof, see community book `books/demos/proofs/tightness-lemma.lisp`.

Conclusion

For a more complete informal proof, see community book `books/demos/proofs/tightness-lemma.lisp`.

(E.g.: The ordered set I must have "room" to move to the right.)

CONCLUSION

For a more complete informal proof, see community book `books/demos/proofs/tightness-lemma.lisp`.

(E.g.: The ordered set *I* must have "room" to move to the right.)

I probably did do a few good things:

Conclusion

For a more complete informal proof, see community book `books/demos/proofs/tightness-lemma.lisp`.

(E.g.: The ordered set *I* must have "room" to move to the right.)

I probably did do a few good things:

- I left comments describing the next main goal.

CONCLUSION

For a more complete informal proof, see community book `books/demos/proofs/tightness-lemma.lisp`.

(E.g.: The ordered set *I* must have “room” to move to the right.)

I probably did do a few good things:

- I left comments describing the next main goal.
- I introduced a predicate for the inductive theorem I was trying to prove.

Conclusion

For a more complete informal proof, see community book `books/demos/proofs/tightness-lemma.lisp`.

(E.g.: The ordered set I must have “room” to move to the right.)

I probably did do a few good things:

- I left comments describing the next main goal.
- I introduced a predicate for the inductive theorem I was trying to prove.
- I put the proof in a separate book and used `SET-ENFORCE-REDUNDANCY`, to keep the problem statement clean.

Conclusion

For a more complete informal proof, see community book `books/demos/proofs/tightness-lemma.lisp`.

(E.g.: The ordered set *I* must have "room" to move to the right.)

I probably did do a few good things:

- I left comments describing the next main goal.
- I introduced a predicate for the inductive theorem I was trying to prove.
- I put the proof in a separate book and used `SET-ENFORCE-REDUNDANCY`, to keep the problem statement clean.

BUT DID IT REALLY NEED TO TAKE 16 HOURS?