ACL2 for Freshmen: First Experiences

Carl Eastlund    Dale Vaillancourt    Matthias Felleisen
’(cce dalev matthias)@ccs.neu.edu

Northeastern University
Boston, MA

ACL2 Workshop 2007
Outline

1. Background
2. Conjecture
3. Experiment
4. Evaluation
Outline

1. Background
2. Conjecture
3. Experiment
4. Evaluation
Freshman Year

Fall Semester

**Fundamentals I:**
Functional programming and the Design Recipe

**Discrete Structures:**
Discrete math, e.g. sets, functions, and induction

Spring Semester

**Fundamentals II:**
Object-oriented programming

**Symbolic Logic:**
Propositional and predicate logic
The Six-Step Design Recipe

;;;; A LoN is either:
;;;; - nil, or
;;;; - (cons Number LoN)

1. Data Definition
2. Contract & Purpose
3. Examples
4. Template
    Multiple clauses?
        Use cond.
    Compound data?
        Apply accessors.
    Inductive data?
        Recur.
5. Write Code
6. Run Tests
The Six-Step Design Recipe

;; A LoN is either:
;; - nil, or
;; - (cons Number LoN)

;; sum : LoN -> Number
;; Add all numbers in a list.

1. Data Definition
2. Contract & Purpose
3. Examples
4. Template
   - Multiple clauses?
     Use cond.
   - Compound data?
     Apply accessors.
   - Inductive data?
     Recur.
5. Write Code
6. Run Tests
The Six-Step Design Recipe

;; A LoN is either:
;; - nil, or
;; - (cons Number LoN)

;; sum : LoN -> Number
;; Add all numbers in a list.

(equal (sum nil) 0)
(equal (sum '(1 2)) 3)
The Six-Step Design Recipe

;; A LoN is either:
;; - nil, or
;; - (cons Number LoN)

;; sum : LoN -> Number
;; Add all numbers in a list.
(defun sum (ns)
  (cond
    ((endp ns) 0)
    (t (+ (car ns) (sum (cdr ns))))
  ))

(equal (sum nil) 0)
(equal (sum '(1 2)) 3)

1 Data Definition
2 Contract & Purpose
3 Examples
4 Template
   Multiple clauses?
     Use cond.
   Compound data?
     Apply accessors.
   Inductive data?
     Recur.
5 Write Code
6 Run Tests
The Six-Step Design Recipe

;; A LoN is either:
;; - nil, or
;; - (cons Number LoN)

;; sum : LoN -> Number
;; Add all numbers in a list.
(defun sum (ns)
  (cond
    ((endp ns) 0)
    (t
      (+ (car ns) (sum (cdr ns))))
  ))

(equal (sum nil) 0)
(equal (sum '(1 2)) 3)
The Six-Step Design Recipe

;; A LoN is either:
;; - nil, or
;; - (cons Number LoN)

;; sum : LoN -> Number
;; Add all numbers in a list.
(defun sum (ns)
  (cond
   ((endp ns) 0)
   (t (car ns)
     (cdr ns) )))

(equal (sum nil) 0)
(equal (sum '(1 2)) 3)

1 Data Definition
2 Contract & Purpose
3 Examples
4 Template
Multiple clauses?
  Use cond.
Compound data?
  Apply accessors.
Inductive data?
  Recur.
5 Write Code
6 Run Tests
The Six-Step Design Recipe

;; A LoN is either:
;; - nil, or
;; - (cons Number LoN)

;; sum : LoN -> Number
;; Add all numbers in a list.
(defun sum (ns)
  (cond
   ((endp ns) 0)
   (t (car ns)
      (sum (cdr ns))))

(equal (sum nil) 0)
(equal (sum '(1 2)) 3)

1. Data Definition
2. Contract & Purpose
3. Examples
4. Template
   - Multiple clauses? Use cond.
   - Compound data? Apply accessors.
   - Inductive data? Recur.
5. Write Code
6. Run Tests
The Six-Step Design Recipe

;; A LoN is either:
;; - nil, or
;; - (cons Number LoN)

;; sum : LoN -> Number
;; Add all numbers in a list.
(defun sum (ns)
  (cond
   ((endp ns) 0)
   (t (+ (car ns)
        (sum (cdr ns)))))))

(equal (sum nil) 0)
(equal (sum '(1 2)) 3)

1 Data Definition
2 Contract & Purpose
3 Examples
4 Template
  Multiple clauses?
  Use cond.
  Compound data?
  Apply accessors.
  Inductive data?
  Recur.
5 Write Code
6 Run Tests
The Six-Step Design Recipe

;; A LoN is either:
;; - nil, or
;; - (cons Number LoN)

;; sum : LoN -> Number
;; Add all numbers in a list.
(defun sum (ns)
  (cond
   ((endp ns) 0)
   (t (+ (car ns)
        (sum (cdr ns)))))))

(equal (sum nil) 0) ; => t
(equal (sum '(1 2)) 3) ; => t

1 Data Definition
2 Contract & Purpose
3 Examples
4 Template
   Multiple clauses?
   Use cond.
   Compound data?
   Apply accessors.
   Inductive data?
   Recur.
5 Write Code
6 Run Tests
Student Languages

ACL2 for Freshmen: First Experiences

Background

DrScheme

Welcome to DrScheme, version 371-svn18aug2007 [3m].
Language: Intermediate Student.
> (fact 3)
  6
> (g "3")
-: expects argument of type <number>; given "3"
>
ACL2 for Freshmen: First Experiences

Background

DrScheme

Teachpacks
ACL2 for Freshmen: First Experiences

Background

Dracula

Dracula Language

```lisp
(defun fact (n)
  (if (eq n 0)
      1
      (* n (fact (- n)))))

(defun g (x)
  (let ((y (fact (- x)))
      (+ y x)))
```

Welcome to DrScheme, version 360.

> (g 42)

2:7: top-level broke the contract (-> natural-number/c any) on zp; expected <natural-number/c>, given: -42

>
Dracula Proofs

---

(defthm firstn-preserves-consecutive
  (implies (consecutive-segments-adj
             (firstn n segs)))
  (consecutive-segments-adj
   (firstn n segs))))

(defthm worm-move-preserves-well-formedness ...
  Q.E.D.)

---

;; When the worm moves, we drop the...
;; This of course preserves that the...

;; Finally, we prove an interesting...

(:type-prescription consp-firstn)
(:type-prescription firstn)
(:type-prescription len)
(:type-prescription list-of-segments?)
(:type-prescription worm-well-formed?)

Warnings: Non-rec
ACL2 for Freshmen: First Experiences

Background

Dracula Teachpacks
Sample Logic Exercise

Prove the conclusion from the premises or provide an interpretation which establishes invalidity.

1. My shirt is under the bed. Your shirt is on the table. If your shirt is on the table, then it’s not under the bed. Therefore, my shirt is not your shirt.

2. If Tom votes, he will vote Democratic unless the party reverses its position on gun-control. The party will not reverse its position on gun-control. So, either Tom doesn’t vote or he will vote Democratic.

3. I will do well in this course and I will study the material. So, I will do well in this course if and only if I will study the material.
Outline

1 Background
2 Conjecture
3 Experiment
4 Evaluation
Remember the S.A.T.?

Logic : Computing :: Analysis : Physics
Preparing Freshmen for ACL2

(defun sum (ns)
  (cond
    ((endp ns) ...)
    (t ... (car ns)
      ... (sum (cdr ns)) ...)))

Multiple clauses?
Use cond.

Compound data?
Apply accessors.

Inductive data?
Recur.
Outline

1. Background
2. Conjecture
3. Experiment
4. Evaluation
ACL2 for Freshmen: First Experiences

Experiment

Overview

ACL2-based Logic Course

**Purpose:** Replacement for *Symbolic Logic*

**Target:** Students from *Fundamentals I*

**Curriculum:** Formal logic and ACL2 Verification

**Trial Run:** Spring 2007

**Format:** Half-credit class

**Size:** 6 freshmen, high A to mid B
Syllabus

1. Introduction
2. Structural Induction
3. Automated Theorem Proving
4. Expanding on Induction
5. Final Project
Syllabus

1. Introduction

   **Lecture Topic**  |  **Homework**  
   ACL2 Syntax       |  Simple programs  
   Propositional logic |  Validity checker  

2. Structural Induction

3. Automated Theorem Proving

4. Expanding on Induction

5. Final Project
Syllabus

1. Introduction

2. Structural Induction
   **Lecture Topic**
   - Structural induction principles
   - Inductive proofs
   **Homework**
   - Examples by hand
   - Examples by hand

3. Automated Theorem Proving

4. Expanding on Induction

5. Final Project
Structural Induction Principles

\[
\text{LoN} = \text{nil} \mid (\text{cons}\ Number\ \text{LoN})
\]

**Multiple kinds of data?** Add hypotheses.

**Structures with fields?** Add quantifiers.

**Inductive data definition?** Add inductive hypothesis.

\[\forall l \in \text{LoN}.\ P(l)\]
Structural Induction Principles

LoN = nil | (cons Number LoN)

Multiple kinds of data? Add hypotheses.

Structures with fields? Add quantifiers.

Inductive data definition? Add inductive hypothesis.

if \( P(\text{nil}) \)
and
\[ P((\text{cons } n \text{ l})) \]
then \( \forall l \in \text{LoN}. \ P(l) \)
Structural Induction Principles

LoN = nil | (cons Number LoN)

Multiple kinds of data? Add hypotheses.

Structures with fields? Add quantifiers.

Inductive data definition? Add inductive hypothesis.

\[
\begin{align*}
\text{if} & \quad P(\text{nil}) \\
\text{and} & \quad \forall l \in \text{LoN. } \forall n \in \text{Number. } \\
& \quad P((\text{cons } n \ l)) \\
\text{then} & \quad \forall l \in \text{LoN. } P(l)
\end{align*}
\]
Structural Induction Principles

\[ \text{LoN} = \text{nil} \mid (\text{cons Number LoN}) \]

**Multiple kinds of data?** Add hypotheses.

**Structures with fields?** Add quantifiers.

**Inductive data definition?** Add inductive hypothesis.

\[
\begin{align*}
\text{if} \quad & P(\text{nil}) \\
\text{and} \quad & \forall l \in \text{LoN. } \forall n \in \text{Number.} \\
& \quad P(l) \Rightarrow P((\text{cons } n \ l)) \\
\text{then} \quad & \forall l \in \text{LoN. } P(l)
\end{align*}
\]
Syllabus

1. Introduction

2. Structural Induction

3. Automated Theorem Proving
   - Lecture Topic: ACL2 strategies, Proof theory
   - Homework: Verify binary tree insert, Proof checker, ACL2 proofs

4. Expanding on Induction

5. Final Project
ACL2 Strategies

- Work out proofs by hand.
- Compare ACL2 output to hand proof.
- Read early checkpoints.
- Guide ACL2 with lemmas.
ACL2 for Freshmen: First Experiences

Experiment

Teaching

Fragile Solutions

We will induct according to a scheme suggested by (INSERT XS1 IT), but modified to accommodate (PERM IT XS2). These suggestions were produced using the :induction rules INSERT and PERM. If we let (:P IT XS1 XS2) denote *1.1 above then the induction scheme we'll use is

\[
\text{AND} \ (\text{IMPLIES} \ (\text{AND} \ (\text{NOT} \ (\text{ENDP} \ IT)) \ (\text{NOT} \ (\text{CONSP} \ IT)))) \\
\quad (:P \ IT \ XS1 \ XS2)) \\
\quad \quad (\text{IMPLIES} \ (\text{AND} \ (\text{NOT} \ (\text{ENDP} \ IT))) \\
\quad \quad \quad (\text{CONSP} \ IT) \\
\quad \quad \quad (\text{NOT} \ (\text{<=} \ XS1 \ (\text{CAR} \ IT)))) \\
\quad \quad \quad (:P \ (\text{CDR} \ IT) \ XS1 \ (\text{DEL} \ (\text{CAR} \ IT) \ XS2))) \\
\quad (:P \ IT \ XS1 \ XS2)) \\
\quad (\text{IMPLIES} \ (\text{AND} \ (\text{NOT} \ (\text{ENDP} \ IT))) \\
\quad \quad (\text{CONSP} \ IT) \\
\quad \quad (\text{<=} \ XS1 \ (\text{CAR} \ IT)))) \\
\quad (:P \ IT \ XS1 \ XS2)) \\
\quad (\text{IMPLIES} \ (\text{ENDP} \ IT) \ (:P \ IT \ XS1 \ XS2))).
\]
Syllabus

1. Introduction
2. Structural Induction
3. Automated Theorem Proving
4. Expanding on Induction
   - Lecture Topic: Generalized induction
   - Homework: Essay: quicksort
   - Lecture Topic: Proof about quicksort
   - Homework: Lemmas by hand
   - Lecture Topic: Proofs w/accumulators
   - Homework: Verify quicksort, accumulators
5. Final Project
Syllabus

1. Introduction
2. Structural Induction
3. Automated Theorem Proving
4. Expanding on Induction
5. Final Project

Lecture Topic | Homework
---|---
First-order logic | Tetris program
Final Project

Assignment: Tetris-like game

Given: One block, falling endlessly

In-class goal: Fix program; prove blocks hit bottom and stop falling

Final goal: 2-3 new, verified Tetris features
Student Performance

In Class: Contributed frequently, presented well

Logic: Proficient at induction with occasional prompting

Programming: “Forgot” the Design Recipe

ACL2: Could prove some theorems; gave up on the rest
Outline

1 Background
2 Conjecture
3 Experiment
4 Evaluation
Exit Interviews

- Fast-paced and challenging
- Overwhelmed by ACL2 output
- Underprepared for proof strategies, logic notation
- Liked the class enough to stay late on Friday afternoon.
Student Accomplishments

- Systematic structural induction
- Presentation skills
- Write, verify ACL2 programs
- All in half a regular course
Future Directions

- Stress the Design Recipe
- Begin ACL2 proofs earlier
- Unified proof strategy
- Simplified readout from ACL2
- Canon of robust proof exercises
- Fix, extend, document Dracula
Success

Northeastern adopted the course.
The End

Thank You!