Equational Reasoning

Pete Manolios
Northeastern

Logic and Computation, 2/7/2019

Example 4

- True or false?

```
(implies (consp x)
         (implies (and (tlp x)
                       (tlp y))
                  (implies (endp x)
                           (equal (aapp x y) (rrev y)))))
(definec rrev (x :tl) :tl
 (if (endp x)
     ()
   (aapp (rrev (rest x)) (list (first x)))))
```

- Prove/disprove it in groups.
  - A: we have a proof
  - B: we have a counterexample
  - C:not sure??

Propositional Skeleton

```
(implies (consp x)
         (implies (and (tlp x)
                       (tlp y))
                  (implies (endp x)
                           (equal (aapp x y) (rrev y)))))
```

A

B

C

D

E

$$A \Rightarrow (B \wedge C \Rightarrow (D \Rightarrow E))$$
$$= A \wedge B \wedge C \Rightarrow (D \Rightarrow E)$$
$$= A \wedge B \wedge C \wedge D \Rightarrow E$$

Exportation!

```
(implies (and (consp x)
              (tlp x)
              (tlp y)
              (endp x))
         (equal (aapp x y) (rev y)))
```

During exportation we only manipulate the propositional skeleton

Context

```
(implies (and (consp x)
              (tlp x)
              (tlp y)
              (endp x))
         (equal (aapp x y) (rev y)))
```

```
Context:
C1.(consp x)
C2.(tlp x)
C3.(tlp y)
C4.(endp x)
Derived Context:
D1. x=nil {Def tlp, C2, C4, cons axioms}
D2. nil   {C1, D1, cons axioms} (or {C1, C4, cons axioms})
```

ER Example 5

- Is this conjecture true? (A:yes, B:no)

```
(tlp x) ∧ (tlp y) ⇒
  [(endp x) ⇒
    [(not (in a (aapp x y))) ⇒
      (not (in a x))]]
```

```
(definec in (a :all X :tl) :bool
  (and (consp X)
       (or (equal a (first X))
           (in a (rest X)))))
```

- Notice that is equivalent to (exportation!):

```
(tlp x) ∧ (tlp y) ∧ (endp x) ∧ (not (in a (aapp x y))) ⇒ (not (in a x))
```

- Which is equivalent to:

```
(tlp x) ∧ (tlp y) ∧ (endp x) ∧ (in a x) ⇒ (in a (aapp x y))
```

- Because (negate & swap): $A \land B \Rightarrow C \equiv A \land \neg C \Rightarrow \neg B$
- During the exportation step, we allow any Boolean simplification

ER Example 5

Conjecture:

```
(tlp x) ∧ (tlp y) ∧ (endp x) ∧ (in a x) ⇒ (in a (aapp x y))
```

Context:

```
C1:(tlp x)
C2:(tlp y)
C3:(endp x)
C4.(in a x)
```

```
(definec in (a :all X :tl) :bool
  (and (consp X)
       (or (equal a (first X))
           (in a (rest X)))))
```

Derived Context:

```
D1.x=nil { Def tlp, C1, C3 }
D2.nil   { Def in, C4, D1 }
```

ER Example 6

- Prove:

```
(implies (tlp x)
         (implies (tlp y)
                  (implies (and (consp x)
                                (equal a (first x)))
                           (implies (not (in a (aapp x y)))
                                    (not (in a x) ))))))
```

- Folding exportation into context from now on for slides/hand proofs.
- Contract completion adds hypotheses: A. Yes B. No
- Context
  - `C1. (tlp x)`
  - `C2. (tlp y)`
  - `C3. (consp x)`
  - `C4. a = (first x)`
  - `C5. (in a x)`

ER Example 6

(tlp x) ∧ (tlp y) ∧ (consp x) ∧ a = (first x) ∧ (in a x) ⇒ (in a (aapp x y))

- C1. (tlp x)
- C2. (tlp y)
- C3. (consp x)
- C4. a = (first x)
- C5. (in a x)

```
(definec in (a :all X :tl) :bool
  (and (consp X)
       (or (equal a (first X))
           (in a (rest X)))))
```

```
  (in a (aapp x y))
= { Def aapp, C3 }
  (in a (cons (first x) (aapp (rest x) y)))
= { Def in, cons axioms }
  (or (equal a (first x)) (in a (aapp (rest x) y)))
= { C4, PL (prop logic) }
  true
```

ER Example 7

```
(implies (and (tlp x)
              (tlp y))
         (implies (and (consp x)
                       (not (equal a (first x)))
                       (implies (tlp (rest x))
                                (implies (in a (rest x))
                                         (in a (aapp (rest x) y)))))
                  (implies (in a x)
                           (in a (aapp x y)))))
```

Contract completion adds hypotheses: A. Yes B. No

ER Example 7

```
(implies (and (tlp x)
              (tlp y))
         (implies (and (consp x)
                       (not (equal a (first x)))
                       (implies (tlp (rest x))
                                (implies (in a (rest x))
                                         (in a (aapp (rest x) y)))))
                  (implies (in a x)
                           (in a (aapp x y)))))
```

Next: Prepare context

ER Example 7

```
(implies (and (tlp x)
              (tlp y))
         (implies (and (consp x)
                       (not (equal a (first x)))
                       (implies (tlp (rest x))
                                (implies (in a (rest x))
                                         (in a (aapp (rest x) y)))))
                  (implies (in a x)
                           (in a (aapp x y)))))
```

A

B

C

Exportation: $A \Rightarrow (B \Rightarrow C) \equiv (A \wedge B) \Rightarrow C$

ER Example 7

```
(implies (and (tlp x)                                    A
              (tlp y)
              (consp x)
              (not (equal a (first x)))
              (implies (tlp (rest x))                   B
                       (implies (in a (rest x))
                                (in a (aapp (rest x) y)))))
         (implies (in a x)
                  (in a (aapp x y)))))                   C
```

Exportation: $A \Rightarrow (B \Rightarrow C) \equiv (A \wedge B) \Rightarrow C$

ER Example 7

```
(implies (and (tlp x)
              (tlp y)
              (consp x)
              (not (equal a (first x)))
              (implies (tlp (rest x))
                       (implies (in a (rest x))
                                (in a (aapp (rest x) y)))))
         (implies (in a x)
                  (in a (aapp x y)))))
```

ER Example 7

```
(implies (and (tlp x)
              (tlp y)
              (consp x)
              (not (equal a (first x)))
              (implies (tlp (rest x))
                       (implies (in a (rest x))
                                (in a (aapp (rest x) y)))))
         (implies (in a x)
                  (in a (aapp x y)))))
```

A

B

C

Exportation again: $A \Rightarrow (B \Rightarrow C) \equiv (A \wedge B) \Rightarrow C$

ER Example 7

```
(implies (and (tlp x)
              (tlp y)
              (consp x)
              (not (equal a (first x)))
              (implies (tlp (rest x))
                       (implies (in a (rest x))
                                (in a (aapp (rest x) y))))      A
              (in a x))   B
         (in a (aapp x y)))))   C
```

Exportation again: $A \Rightarrow (B \Rightarrow C) \equiv (A \wedge B) \Rightarrow C$

ER Example 7

```
(implies (and (tlp x)
              (tlp y)
              (consp x)
              (not (equal a (first x)))
              (implies (tlp (rest x))
                       (implies (in a (rest x))
                                (in a (aapp (rest x) y))))
              (in a x))
         (in a (aapp x y))))
```

ER Example 7

```
(implies (and (tlp x)
              (tlp y)
              (consp x)
              (not (equal a (first x)))
              (implies (tlp (rest x))   A
                       (implies (in a (rest x))   B
                                (in a (aapp (rest x) y))))   C
              (in a x))
         (in a (aapp x y)))))
```

Exportation again: $A \Rightarrow (B \Rightarrow C) \equiv (A \wedge B) \Rightarrow C$

ER Example 7

```
(implies (and (tlp x)
              (tlp y)
              (consp x)
              (not (equal a (first x)))
              (implies (and (tlp (rest x))      A
                            (in a (rest x)))    B
                       (in a (aapp (rest x) y)))  C
              (in a x))
         (in a (aapp x y)))))
```

Exportation again: $A \Rightarrow (B \Rightarrow C) \equiv (A \wedge B) \Rightarrow C$

ER Example 7

```
(implies (and (tlp x)
              (tlp y)
              (consp x)
              (not (equal a (first x)))
              (implies (and (tlp (rest x))
                            (in a (rest x)))
                       (in a (aapp (rest x) y)))
              (in a x))
         (in a (aapp x y)))))
```

Notice that we cannot use exportation in the 5th hypothesis

We will apply exportation/PL simplification as much as possible and recursively!

ER Example 7

```
(definec in (a :all X :tl) :bool
  (and (consp X)
       (or (equal a (first X))
           (in a (rest X)))))
```

```
(implies
  (and (tlp x)
       (tlp y)
       (consp x)
       (not (equal a (first x)))
       (implies (and (tlp (rest x))
                     (in a (rest x)))
                (in a (aapp (rest x) y)))
       (in a x))
  (in a (aapp x y))))
```

```
C1. (tlp x)
C2. (tlp y)
C3. (consp x)
C4. a ≠ (first x)
C5. (tlp (rest x)) ∧ (in a (rest x))
      ⇒ (in a (aapp (rest x) y))
C6. (in a x)
-----------------------------------------------
D1. (tlp (rest x)) { Def tlp, C1, C3 }
D2. (in a (rest x))  { Def in, C6, C3, C4, PL }
D3. (in a (aapp (rest x) y)) { C5, D1, D2, MP }
```

```
(definec tlp (l :all) :bool
  (if (consp l)
      (tlp (rest l))
    (equal l () )))
```

ER Example 7

```
C1. (tlp x)
C2. (tlp y)
C3. (consp x)
C4. a ≠ (first x)
C5. (tlp (rest x)) ∧ (in a (rest x))
     ⇒ (in a (aapp (rest x) y))
C6. (in a x)
-----------------------------------------------
D1. (tlp (rest x)) { Def tlp, C1, C3 }
D2. (in a (rest x))  { Def in, C6, C3, C4, PL }
D3. (in a (aapp (rest x) y)) { C5, D1, D2, MP }
```

```
(definec in (a :all X :tl) :bool
  (and (consp X)
       (or (equal a (first X))
           (in a (rest X)))))
```

```
(definec tlp (l :all) :bool
  (if (consp l)
      (tlp (rest l))
    (equal l () )))
```

Goal: (in a (aapp x y))

```
  (in a (aapp x y))
=   { Def aapp, C3 }
  (in a (cons (first x) (aapp (rest x) y)))
=   { Def in, cons axioms }
  (or (equal a (first x)) (in a (aapp (rest x) y)))
=   { D3, PL }
  t
```

Fermat's Last Theorem

- For all positive integers x, y, z, and n, where n>2, $x^n + y^n \neq z^n$
- In 1637, Fermat wrote about the above:

  "I have a truly marvelous proof of this proposition which this margin is too narrow to contain."
- It took 357 years for a correct proof to be found (by Andrew Wiles in 1995)
- Can we express in ACL2s?

  Sure:

```
(defunc f (x y z n)
  :input-contract (and (posp x) (posp y) (posp z)
                       (natp n) (> n 2))
  :output-contract (booleanp (f x y z n))
  (not (equal (+ (expt x n) (expt y n))
              (expt z n))))

(thm (implies ic (f x y z n)))
```

Proving Theorems is Hard

- Notice also that if we change the output contract as follows

```
(defunc f (x y z n)
  :input-contract (and (posp x) (posp y) (posp z)
                       (natp n) (> n 2))
  :output-contract (equal (f x y z n) t)
  (not (equal (+ (expt x n) (expt y n))
              (expt z n))))
```

- then ACL2s would have to prove a theorem that eluded mankind for centuries in order to even admit f!

Admitting Functions is Hard

- Take any conjecture: phi over $x_1, \ldots, x_n$

```
(defdata true t)

(definec f (x1 :all … xn :all) :true
   phi)
```

- ACL2s has to prove phi to admit `f` (function contract)!
- How hard is this?
- This is undecidable. In fact, just using =, +, * and limiting the universe to numbers already results in an undecidable theory!
- But, it is easy to find a counterexample if a conjecture is false, right?
- No. There are many examples of conjectures that took a long time to resolve, and which turned out to be false