Logic and Computation – CS 2800 Fall 2019

Lecture 8

Property-based testing

Stavros Tripakis

Northeastern University
Khoury College of Computer Sciences

Outline

- Answer question about fn-first-first
- Finish data definitions
- Controlling ACL2s
- Property-based testing
  - `test?` and `thm`

Quiz

- The two data definitions below are equivalent (i.e., they define the same type):

```
(defdata mytype (list int))
(defdata mytype (listof int))
```

A. Yes
B. No

Some ways to control ACL2s

- `:program` mode turns off theorem proving
  - No termination analysis attempted
  - No proving contracts etc
  - ACL2s will still test contracts and report any errors it finds
  - Useful for prototyping / experimenting

- `:logic` mode is the default mode
  - Use `:logic` to switch back into logic mode after going into program mode,
    - but note that `:logic` functions cannot depend on `:program` functions

- Other useful settings:
  - `(acl2s-defaults :set testing-enabled nil)`
  - `(set-defunc-termination-strictp nil)`
  - `(set-defunc-function-contract-strictp nil)`
  - `(set-defunc-body-contracts-strictp nil)`
  - To switch back: set to `t` instead of `nil`

- Documentation via `:doc`

- How to use these in homeworks

Property-based testing

What is a property?

- **Property** = a statement which either holds or not, i.e., the statement is either true or false

```
(definec even-natp (x :nat) :bool
  (natp (/ x 2)))

(definec even-intp (x :int) :bool
  (integerp (/ x 2)))
```

- Example property: *"the two functions above always return the same result when the input is a natural number"*

What is a property?

- We will write properties in logic – why?
  - using the language that ACL2s already provides!

```
(definec even-natp (x :nat) :bool
  (natp (/ x 2)))

(definec even-intp (x :int) :bool
  (integerp (/ x 2)))
```

```
(implies (natp n)
         (equal (even-natp n)
                (even-intp n)))
```

Property written in ACL2s

"the two functions always return the same result when the input is a natural number"

Property written in English

Testing and proving in ACL2s

```
(definec even-natp (x :nat) :bool
  (natp (/ x 2)))

(definec even-intp (x :int) :bool
  (integerp (/ x 2)))
```

- Property-based testing:

```
(test? (implies (natp n)
                (equal (even-natp n)
                       (even-intp n))))
```

- Theorem proving:

```
(thm   (implies (natp n)
                (equal (even-natp n)
                       (even-intp n))))
```

Testing vs proving

- Testing:

```
(test? (implies (natp n)
                (equal (even-natp n)
                       (even-intp n))))
```

  - We are asking the tool to generate many examples and test on each example whether property holds
    - In this case, an example is one specific **n**
  - The test passes if the tool cannot find an example violating the property: a counter-example
  - More powerful than `check=` which tests just one example

- Theorem proving:

```
(thm (implies (natp n)
              (equal (even-natp n)
                     (even-intp n))))
```

  - We are asking the tool to prove that the property holds for every `n`
  - More powerful than `test?` – why?
  - Also theorem proving techniques fundamentally different from testing

Next time

- Property-based testing continued