An Electronic Phone Book Example in PVS

Néstor Cataño

ncatano@puj.edu.co

Faculty of Engineering

Pontificia Universidad Javeriana

An Electronic Phone Book: Requirements

- A phone book shall store the phone numbers of a city
- It shall be possible to retrieve a phone number given a name
- It shall be possible to add and delete entries from a phone book

An Electronic Phone Book: Requirements

- There are 3 entities mentioned :
  - Phone books
  - Phone numbers
  - Names

An Electronic Phone Book: Requirements

- We need 3 operations :
  - `findphone`: it takes a phone book and a name and returns the phone number associated with that name
  - `addphone`: it takes a phone book, a name, and a phone number, and adds the association between the number and the name to the phone book
  - `delphone`: it takes a phone book and a name and deletes the phone number associated with that name (if any)

Representing Entities in PVS

```
N: type  % names
P: type  % phone numbers
```

Representing a Phone Book

- A set of `(name, phone number)` pairs
- As a partial function from names to phone numbers where the domain of the function are the names currently stored
- As a total function from names to phone numbers where the domain of the function are all the possible names

Representing a Phone Book

PVS is a logic of total functions

```
B: type = [N -> P] % phone books
```

Representing a Phone Book

How can we indicate that a name has no phone number ?

- By using a `p0` which is different from any real phone number

An Empty Phone Book

```
p0: P
emptybook: B

empty_axiom: axiom
 forall(n: N): emptybook(n) = p0
```

A Phone Book Example

```
phone_1: theory
 begin
  N: type  % names
  P: type+ % phone numbers
  B: type = [N -> P]  % books

  p0: P
  emptybook: B

  empty_axiom: axiom forall(n: N): emptybook(n) = p0

  findphone: [B, N -> P]
  find_axiom: axiom forall(b: B), (n: N):  findphone(b, n) = b(n)

  addphone: [B,N,P -> B]
  add_axiom: axiom
    forall(b:B, n:N, p:P): addphone(b, n, p) = b with [(n) := p]

  delphone: [B, N -> B]
  del_axiom: axiom
    forall(b: B, n:N): delphone(b,n) = b with [(n):= p0]
end phone_1
```

Deficiencies in the Specification

1. The specification does not rule out the possibility of giving someone `p0` as a phone number
2. `addphone` changes the phone number when applied to someone who already has a phone number
3. The use of **`axiom`** can introduce inconsistencies

Deficiencies in the Specification

The specification does not rule out the possibility of giving someone `p0` as a phone number

```
GP: type = {p:P | p/=p0}
addphone:  [B,N,P -> B]
```

Deficiencies in the Specification

The specification does not rule out the possibility of giving someone `p0` as a phone number

```
GP: type = {p:P | p/=p0}
addphone:  [B,N,GP -> B]
```

Deficiencies in the Specification

The specification does not rule out the possibility of giving someone `p0` as a phone number

```
GP: type = {p:P | p/=p0}
addphone:  [B,N,GP -> B]
```

- The subtype `{p:P | a}` consists of those elements e satisfying `a[e/x]`
- Since elements in `{p:P | a}` satisfy the predicate `λ(x:P): a`, we call `{p:P | a}` predicate subtype

Deficiencies in the Specification

addphone changes the phone number when applied to someone who already has a phone number

Deficiencies in the Specification

addphone changes the phone number when applied to someone who already has a phone number

```
known?(b:B, n:N): bool = b(n)/=p0

addPhone(b:B, n:N, gp:GP): B =
  if known?(b,n) then b else b with [(n):= gp] endif

changePhone(b:B, n:N, gp:GP): B =
  if known?(b,n) then b with [(n):= gp] else b endif
```

Deficiencies in the Specification

The use of **axiom** can introduce inconsistencies

```
delphone: [B, N -> B]
del_axiom: axiom
  forall(b: B, n:N): delphone(b,n) = b with [(n):= p0]
```

Deficiencies in the Specification

The use of **axiom** can introduce inconsistencies

```
delphone: [B, N -> B]
del_axiom: axiom
  forall(b: B, n:N): delphone(b,n) = b with [(n):= p0]

delPhone(b:B, n:N): B = b with [(n):= p0]
```

A Phone Book Example

```
phone_2: theory
 begin
  N: type  % names
  P: type+ % phone numbers
  B: type = [N -> P] % books

  p0: P
  GP: type = {p:P | p/=p0}

  known?(b:B, n:N): bool = b(n)/=p0

  findPhone(b:B, n:N): P = b(n)

  addphone(b:B, n:N, gp:GP): B =
   if known?(b,n)then b else b with [(n):= gp] endif

  changephone(b:B, n:N, gp:GP): B =
   if known?(b,n) then b with [(n):= gp] else b endif

  delphone(b:B, n:N): B =  b with [(n):= p0]

  del_add: theorem
   forall(b: B, n:N, gp:GP):
    delphone(addphone(b,n,gp),n) = delphone(b,n)
end phone_2
```

Homework

Prove the following 3 theorems in PVS

```
find_add: theorem
 forall(b: B, n:N, gp:GP):
  not known?(b,n) implies findphone(addphone(b,n,gp),n) = gp

find_change: theorem
 forall(b: B, n:N, gp:GP):
  known?(b,n) implies findphone(changephone(b,n,gp),n) = gp

add_change: theorem
 forall(b: B, n:N, gp1:GP, gp2:GP):
  changephone(addphone(b,n,gp1), n, gp2) =
   addphone(changephone(b,n,gp2),n,gp2)
```