ECE 741/841

22 October 2002

Reservation System, Requirements

- The reservation system will store all pertinent data associated with the reservation.
  - type of fare
  - itinerary
  - seat selection
  - etc.
- Given a reservation or confirmation number (identifier), there shall be a way to retrieve the reservation information.
- It should be possible to add reservations to and delete reservations from the reservation system.

Mathematical Definition of Reservation System

We will define a function that has as its domain the set of all reservation numbers and as its range the set of all possible reservations.

Let

N = the set of reservation numbers (identifiers).
R = the set of reservations.
D = the set of reservation data bases, $N \rightarrow R$.

This is a very high level representation of the reservation system. All details have been abstracted away. An element of the set of reservations represent all the information described above.

Specification

It is possible that for a given reservation number, there is not a reservation associated with it.

We will define a constant *null* to represent that there is not a reservation associated with a reservation number.

$null \in R$

Using the constant *null*, we can define an empty data base:

let $n \in N$
$empty\_database : N \to R$

$empty\_database(n) = null$

Retrieving a Reservation

Let

N = the set of reservation numbers (identifiers).
R = the set of reservations.
D = the set of reservation data bases, $N \to R$.
$n \in N$ and $d \in D$.

retrieve_reserv : $D \times N \to R$

retrieve_reserv(d,n) = *d(n)*

Adding a Reservation

Let

N = the set of reservation numbers (identifiers).
R = the set of reservations.
D = the set of reservation data bases, $N \to R$.
$n1, n2 \in N$; $r \in R$; and $d \in D$.

add_reserv : $D \times N \times R \to D$

$$\textit{add\_reserv}(d, n1, r)(n2) = \begin{cases} d(n2) & \textit{if } n1 \neq n2 \\ r & \textit{if } n1 = n2 \end{cases}$$

Deleting a Reservation

Let

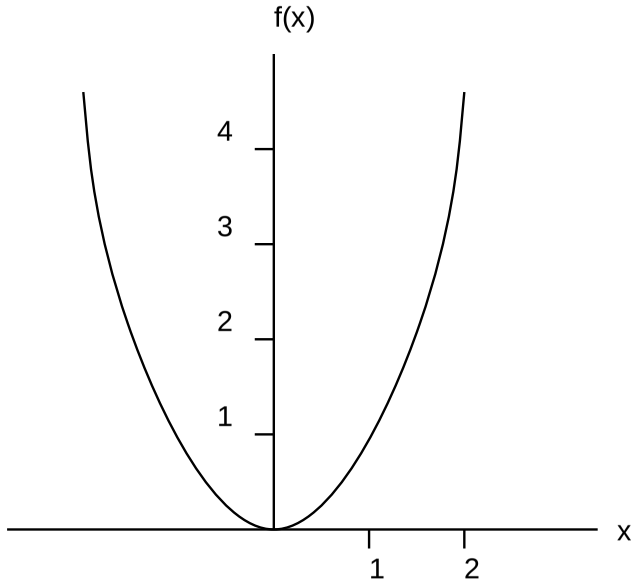
N = the set of reservation numbers (identifiers).
R = the set of reservations.
D = the set of reservation data bases, $N \to R$.
$n1, n2 \in N$; $r, null \in R$; and $d \in D$.

delete_reserv : $D \times N \to D$

$$\textit{delete\_reserv}(d, n1)(n2) = \begin{cases} d(n2) & if \;\; n1 \neq n2 \\ null & if \;\; n1 = n2 \end{cases}$$

Defining the Functions in PVS, the WITH Notation

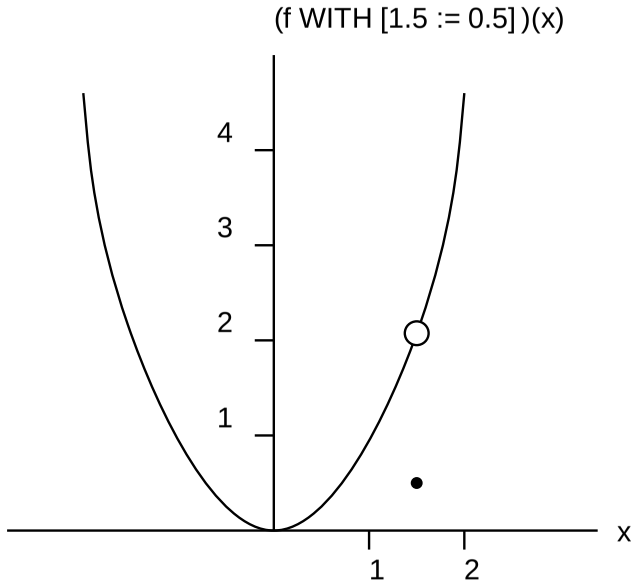
$f(x) = x^2$




```
f(x:real) :  real = x*x
```

Defining the Functions in PVS, the WITH Notation

$$f'(x) = f \ \ WITH \ \ [1.5 := 0.5]$$




```
f(x:real) :  real = x*x

fp(x) :  real = (f WITH [(15/10) := 5/10])(x)
```

Using the WITH Notation

```
test_with : theory
begin

f(x:real) : real = x*x

fp(x:real) : real = (f WITH [(15/10) := 5/10])(x)

check1 : lemma
f(2) = fp(2)

check2 : lemma
f(15/10) /= fp(15/10)

check3 : lemma
forall (x:real) : x /= 15/10 implies f(x) = fp(x)

end test_with
```

Reservation System Specification

Let

N = the set of reservation numbers (identifiers).
R = the set of reservations.
D = the set of reservation data bases, $N \to R$.
$n1, n2 \in N$; $r, null \in R$; and $d \in D$.

retrieve_reserv(d,n) = d(n)

add_reserv(d,n,r) = d WITH [n := r]

delete_reserv(d,n) = d WITH [n := null]

Putative Theorems

putative, adj. Commonly regarded as such; reputed; supposed.

putative1 : lemma
retrieve_reserv(add_reserv(d,n,r),n) = r

putative2 : lemma
delete_reserv(add_reserv(d,n,r),n) = d

Proof Attempt for Lemma Putative2

delete_reserv(add_reserv(d,n,r),n) = d

delete_reserv(d WITH [n := r],n) = d

(d WITH [n := r]) WITH [n := null] = d

d WITH [n := null] = d

This can only be equal if *d(n) = null*.

Specification Revision

The specification of the function *add_reserv* allows to overwrite a previous specification in the data base. This does not seem to be a good characteristic.

We define a predicate *scheduled* and redefine the function *add_reserv* to correct this deficiency.

$scheduled(d,n) : bool = d(n) \neq null$

add_reserv(d,n,r) =
if scheduled(d,n) then d else
d WITH [n := r]
endif

Proof Attempt for Lemma Putative1

retrieve_reserv(add_reserv(d,n,r),n) = r

retrieve_reserv(if scheduled(d,n) then d else
d WITH [n := r] endif,n) = r

(if scheduled(d,n) then d else
d WITH [n := r] endif)(n) = r

(if scheduled(d,n) then d(n) else
r endif) = r

This can only be equal if *scheduled(d,n) = false*.

Specification Revision, 2

The specification of the function *add_reserv* prevents overwriting but loses reservation if the reservation number is already in the data base.

We redefine the function *add_reserv* using a dependent type.

add_reserv(d,n:{n:N | d(n) = null},r) = d WITH [n := r]

When this function is used, a type correctness condition (TCC) is generated to show that the reservation number n is not already defined in d.

Using this definition, both putative lemmas can now be proven.

Change Reservation Definition

It is useful to have a function that allows to change reservations, in addition to creating and deleting reservations.

change_reserv(d,n,r) =
if scheduled(d,n) then d WITH [n := r] else
d
endif

We can also use dependent types to make sure that the function *change_reserv* can only be used when a reservation already exists.

change_reserv(d,n:{n:N | d(n) ≠ null},r) =
d WITH [n := r]

Extensionality

Syntax: `(extensionality "type")`

```
T1 : TYPE [nat -> nat]

FORALL (f,g : T1) :
(FORALL (i:nat) : f(i) = g(i)) implies f = g

T2 : TYPE [# a:nat, b:nat]

FORALL (r,s : T2) :
a(r) = a(s) AND b(r) = b(s) implies r = s
```

The Theory in PVS

```
reserv : theory
begin

N : TYPE+                 % Reservation numbers
R : TYPE+                 % Reservation information
D : TYPE = [N -> R]       % Reservation system data base

null : R                  % A null reservation

n : VAR N
r : VAR R
d : VAR D

emptydb(n) : R = null

retrieve_reserv(d,n) : R = d(n)
```

```
scheduled(d,n) : bool = d(n) /= null

add_reserv(d,(n:{n:N | d(n) = null}),r) : D = d WITH [n := r]

change_reserv(d,(n:{n:N | d(n) /= null}),r) : D = d WITH [n := r]

delete_reserv(d,n) : D = d WITH [n := null]

putative1 : lemma
retrieve_reserv(add_reserv(d,n,r),n) = r

putative2 : lemma
forall (d,(n:{n:N | d(n) = null})):
delete_reserv(add_reserv(d,n,r),n) = d

end reserv
```

Homework

due on 31 October 2002

What other putative theorems do you think this theory should have?