Applying Predicate Logic to Software Documentation

Subroutines are often documented by specifying the following information

- Precondition: A boolean expression describing what should be true when the subroutine begins execution
- May Change: A list of variables whose values may be changed.
- Postcondition: A boolean expression describing what will be true when the subroutine completes execution
  - In the postcondition, v' represents the final value of variable v,
  - while a plain v represents the initial value of the variable v

Example a subroutine that searches an array A for a particular value x may be described by

void find(**int** A[N], **int** x, **int** &i)
// Precondition: exists j : {0,1,...,N-1}, A[j]==x
// May change: i
// Postcondition: A[i']==x

- The precondition says that the subroutine should only be called if there is an x somewhere in A.
- The postcondition says that the final value of i should index an element of A equal to x.

Example a subroutine that sorts an array of integers

void sort(**int** A[N])
// Precondition: true
// May change: A
// Postcondition: (for all i : {1,2,...,N-1}, A'[i-1] $<$= A'[i])
//   and (for all x : Int, $|${i $|$ A[i]==x}$|$ == $|${i $|$ A'[i]==x}$|$)

- The first line of the postcondition says that the array A is sorted at completion
- The second line says that its contents have been permuted, but not otherwise changed.

Applying predicate logic to system specification

A “System” may be defined as

- an object that imposes a relationship on objects labeled as its inputs and outputs.

A “System model” is a boolean expression that describes the relationship the system imposes.

- The free variables of the model are the names of the inputs and outputs.
- Usually inputs and outputs are modelled as functions of time
- Often, but not always, a system model is a function (aka transform) from its inputs to its outputs.

Systems are often composed from subsystems

An example:

(We take time to range over the natural numbers counting clock cycles.)

Thus x, y and z range over functions from the natural numbers $\mathbb{N}$ to the set $\{T, F\}$

Consider a system consisting of a not-gate with input x and output y

$$\textit{Not}(x, y) \triangleq (\forall t, y(t) \leftrightarrow \neg x(t))$$

Consider a system consisting of a D-flip-flop with input x and output y.

$$\textit{DFF}(x, y) \triangleq (\forall t, y(t+1) \leftrightarrow x(t))$$

We can compose these two systems in various ways. Here are models of two

$$\textit{NotThenDFF}(x, y) \triangleq \exists z, \textit{Not}(x, z) \wedge \textit{DFF}(z, y)$$

and

$$\textit{DFFThenNot}(x, y) \triangleq \exists z, \textit{DFF}(x, z) \wedge \textit{Not}(z, y)$$

We can show (using predicate logic and other math) that these two composed systems have equivalent models.

Although this example deals with digital systems, exactly the same ideas apply to analog systems.