Applying Predicates to Software Documentation

Subroutines are often documented by specifying the following information:

- **Precondition**: A predicate describing what should be true when the subroutine begins execution.
- **May Change**: A list of variables whose values may be changed.
- **Postcondition**: A predicate 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

```c
```

- 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

```c
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 predicates 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 predicate 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.
An example:

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

\[
\text{Not}(x, y) \triangleq (\forall t, y(t) = \neg x(t))
\]

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

\[
\text{DFF}(x, y) \triangleq (\forall t, y(t + 1) = x(t))
\]

We can combine these two systems in two ways

\[
\text{NotThenDFF}(x, y) \triangleq \exists z, \text{Not}(x, z) \wedge \text{DFF}(z, y)
\]

or

\[
\text{DFFThenNot}(x, y) \triangleq \exists z, \text{DFF}(x, z) \wedge \text{Not}(z, y)
\]

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

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