Behavioural Specifications

Aside: Throughout these notes, I will abbreviate functions by their graphs. **End aside.**

System boundaries and signatures

We take a "*black box*" point of view of systems

That is we

- describe the relationship between input and output quantities
- ignore internal quantities

We can describe such a relationship using a boolean expression.

For example

$$\langle V = 100 \times I \rangle$$

describes a 100 ohm resistor.

A **system boundary** consists of the inputs and outputs of a system

We name each input and output and specify its type with a **signature**

A **signature** is a partial function that maps names to nonempty sets of values.

Examples:

- $$\Sigma = \{\text{“}V\text{”} \mapsto \mathbb{R}, \text{“}I\text{”} \mapsto \mathbb{R}\}$$

  (I am abbrevating the function with its graph.)

- $$\Sigma_0 = \{\text{“}x\text{”} \mapsto \mathbb{Z}, \text{“}y\text{”} \mapsto \mathbb{Z}, \text{“}x'\text{”} \mapsto \mathbb{Z}, \text{“}y'\text{”} \mapsto \mathbb{Z}\}$$
  Here x, y are inputs while x' and y' are names of outputs. All are integers.

- $$\Sigma_1 = \left\{\text{“}d\text{”} \mapsto \left(\mathbb{N} \overset{\text{tot}}{\to} \mathbb{B}\right), \text{“}q'\text{”} \mapsto \left(\mathbb{N} \overset{\text{tot}}{\to} \mathbb{B}\right)\right\}$$
  Here the name “d” is the name of an input and “q'” is the name of an output. Both input and output are (modeled as) functions from the natural numbers to the booleans.

- $$\Sigma_2 = \left\{\text{“}x\text{”} \mapsto \left(\mathbb{R} \overset{\text{tot}}{\to} \mathbb{R}\right), \text{“}x'\text{”} \mapsto \left(\mathbb{R} \overset{\text{tot}}{\to} \mathbb{R}\right)\right\}$$

Convention: We use unprimed names like x, y, d for inputs and primed names like x', y', q' for outputs.

Behaviours

A **behaviour** is a partial function that maps names to values.

Examples:

- $$b_0 = \{\text{“}x\text{”} \mapsto -3, \text{“}y\text{”} \mapsto 5, \text{“}x'\text{”} \mapsto 5, \text{“}y'\text{”} \mapsto 5\}$$
- Let a be the function in $\mathbb{N} \stackrel{\text{tot}}{\to} \mathbb{B}$ such that $a(i) = (i \bmod 3 = 0)$ for all i

  $$a = (\mathbb{N}, \mathbb{B}, \{\, 0 \mapsto \mathfrak{true}, 1 \mapsto \mathfrak{false}, 2 \mapsto \mathfrak{false}, 3 \mapsto \mathfrak{true}, 4 \mapsto \mathfrak{false}, 5 \mapsto \mathfrak{false}, \cdots \})$$

  and b be the function in $\mathbb{N} \stackrel{\text{tot}}{\to} \mathbb{B}$ such that $b(i) = (i \bmod 3 = 1)$ for all i

  $$b = (\mathbb{N}, \mathbb{B}, \{\, 0 \mapsto \mathfrak{false}, 1 \mapsto \mathfrak{true}, 2 \mapsto \mathfrak{false}, 3 \mapsto \mathfrak{false}, 4 \mapsto \mathfrak{true}, 5 \mapsto \mathfrak{false}, \cdots \})$$

  then

  $$b_1 = \{\text{“}d\text{”} \mapsto a, \text{“}q'\text{”} \mapsto b\}$$

  is a behaviour
- $$b_2 = \{\text{“}x\text{”} \mapsto \sin, \text{“}x'\text{”} \mapsto 2 \times \sin\}$$

Notation: We write $b : \Sigma$ to mean that behaviour b **belongs to** signature Σ. This means that the same names are mapped and the behaviour obeys the type information provided by the signature.

Formally: $b : \Sigma$ iff $\operatorname{dom}(b) = \operatorname{dom}(\Sigma)$ and $\forall n \in \operatorname{dom}(\Sigma) \cdot b(n) \in \Sigma(n)$

Examples: $b_0 : \Sigma_0$, $b_1 : \Sigma_1$, and $b_2 : \Sigma_2$

Behavioural Specifications

Given a system, there are two kinds of behaviours:

- behaviours the system could engage in
- behaviours the system can not engage in

A **behavioural specification** distinguishes between these two kinds of behaviours.

We define a behavioural specification to be a pair

$$(\Sigma, f)$$

where Σ is a signature and f is a boolean function such that

$$b \in \operatorname{dom}(f) \qquad \text{, for all } b : \Sigma$$

Notation: I'll generally write (Σ, f) as f_Σ or (when Σ is clear from context) just as f.

If $f(b) = \mathfrak{true}$, we say that the specification f_Σ **accepts** behaviour b.

If $f(b) = \mathfrak{false}$, we say that the specification f_Σ **rejects** behaviour b.

Angle-Bracket Notation

I'll write boolean functions on behaviours as boolean expressions in angle brackets. For example

$$\langle x' = y \wedge y' = y \rangle$$

abbreviates the function f defined by

$$f(b) = (b(\text{“}x'\text{”}) = b(\text{“}y\text{”}) \wedge b(\text{“}y'\text{”}) = b(\text{“}y\text{”}))$$

Examples of specifications

An assignment Statement

Let x be the initial value of a program variable and x' be the final value of the same variable. Similarly with y.

Let

$$\begin{aligned} \Sigma \ &= \ \{\text{“}x\text{”} \mapsto \mathbb{Z}, \text{“}y\text{”} \mapsto \mathbb{Z}, \text{“}x'\text{”} \mapsto \mathbb{Z}, \text{“}y'\text{”} \mapsto \mathbb{Z}\} \\ e \ &= \ \langle x' = 0 \wedge y' = y \rangle \end{aligned}$$

then e_Σ is a specification that accepts behaviour

$$\{\text{“}x\text{”} \mapsto -3, \text{“}y\text{”} \mapsto 5, \text{“}x'\text{”} \mapsto 0, \text{“}y'\text{”} \mapsto 5\}$$

but rejects

$$\{\text{“}x\text{”} \mapsto -3, \text{“}y\text{”} \mapsto 5, \text{“}x'\text{”} \mapsto 5, \text{“}y'\text{”} \mapsto 5\}$$

Later we will write this specification as

$$x := 0$$

Examples of specifications (continued)

Another assignment Statement

Let

$$f = \langle x' = y \wedge y' = y \rangle$$

then f_Σ is a specification that accepts behaviour

$$\{\text{“}x\text{”} \mapsto -3, \text{“}y\text{”} \mapsto 5, \text{“}x'\text{”} \mapsto 5, \text{“}y'\text{”} \mapsto 5\}$$

since $f(\{\text{“}x\text{”} \mapsto -3, \text{“}y\text{”} \mapsto 5, \text{“}x'\text{”} \mapsto 5, \text{“}y'\text{”} \mapsto 5\}) = \mathfrak{true}$

but rejects

$$\{\text{“}x\text{”} \mapsto -3, \text{“}y\text{”} \mapsto 5, \text{“}x'\text{”} \mapsto 0, \text{“}y'\text{”} \mapsto -3\}$$

since $f(\{\text{“}x\text{”} \mapsto -3, \text{“}y\text{”} \mapsto 5, \text{“}x'\text{”} \mapsto 0, \text{“}y'\text{”} \mapsto -3\}) = \mathfrak{false}$.

Later we will write this specification as

$$x := y$$

Flip-flop

Let d represent the input to a d-flip-flop and q' represent the output to the same d-flip-flop. Let

$$\begin{aligned} \Sigma &= \left\{\text{“}d\text{”} \mapsto \left(\mathbb{N} \overset{\text{tot}}{\to} \mathbb{B}\right), \text{“}q'\text{”} \mapsto \left(\mathbb{N} \overset{\text{tot}}{\to} \mathbb{B}\right)\right\} \\ g &= \langle \forall t \in \mathbb{N} \cdot q'(t+1) = d(t) \rangle \end{aligned}$$

then g_Σ is a specification for a d-flip-flop. For example b_1 is accepted by this specification, whereas $\{\text{“}d\text{”} \mapsto b, \text{“}q'\text{”} \mapsto a\}$ is rejected.

Note that for each input value, there are 2 outputs values that make an acceptable behaviour.

Examples of specifications (continued)

Amplifier

Let x be an input signal as a function of time and x' be an output signal as a function of time

$$\Sigma = \left\{\text{“}x\text{”} \mapsto \left(\mathbb{R} \overset{\text{tot}}{\to} \mathbb{R}\right), \text{“}x'\text{”} \mapsto \left(\mathbb{R} \overset{\text{tot}}{\to} \mathbb{R}\right)\right\}$$

and define a function

$$h = \langle \forall t \in \mathbb{R} \cdot x'(t) = 2 \times x(t) \rangle$$

Then h_Σ represents an amplifier. At each point in time, the output signal is twice the input signal.

For example $\{\text{“}x\text{”} \mapsto \sin, \text{“}x'\text{”} \mapsto 2 \times \sin\}$ is accepted, whereas $\{\text{“}x\text{”} \mapsto \sin, \text{“}x'\text{”} \mapsto \sin\}$ is rejected, as is, $\{\text{“}x\text{”} \mapsto \sin, \text{“}x'\text{”} \mapsto 2 \times \cos\}$

Refinement

Suppose that f_Σ accepts every behaviour that g_Σ accepts, i.e.

$$\forall b : \Sigma \cdot g(b) \Rightarrow f(b)$$

Then we say that g_Σ **refines** f_Σ.

Notation: We write

$$f_\Sigma \sqsubseteq g_\Sigma$$

or (when Σ is clear from context)

$$f \sqsubseteq g$$

to say f_Σ **is refined by** g_Σ.

Uses of specifications and refinement

We can use formal specifications of systems for several different processes.

- **Documentation.** We can use a specification to describe the behaviour of a known system.
- **Requirements Specification.** We can use a specification to specify the required behaviour of a system to be built.
- **Testing.** Given a specification f_Σ and an observed behaviour b of a system, $\neg f(b)$ indicates an error.
- **Verification.** Suppose f_Σ represents the system desired (requirements) and g_Σ represents the system as designed. To verify that the design meets its requirements we need to check
  $$\forall b : \Sigma \cdot g(b) \Rightarrow f(b)$$
  I.e.
  $$f_\Sigma \sqsubseteq g_\Sigma$$
- **Design.** A design problem is one of the form "Given a specification f, find a specification g such that $f_\Sigma \sqsubseteq g_\Sigma$.
  - **Stepwise Derivation.** If we have a specification f_Σ . We can design a system by finding a sequence of specifications
    $$f_\Sigma \sqsubseteq f1_\Sigma \sqsubseteq f2_\Sigma \sqsubseteq f3_\Sigma \sqsubseteq g_\Sigma$$
    where g_Σ represents a design.

Examples of refinement

Consider the signature

$$\Sigma = \{\text{“}x\text{”} \mapsto \mathbb{Z}, \text{“}x'\text{”} \mapsto \mathbb{Z}\}$$

- Let

  $$\begin{aligned} f &= \langle x' > x \rangle \\ g &= \langle x' = x + 1 \rangle \end{aligned}$$

  Some example behaviours

  $\{\text{“}x\text{”} \mapsto 2, \text{“}x'\text{”} \mapsto 3\}$ Accepted by g and accepted by f

  $\{\text{“}x\text{”} \mapsto 2, \text{“}x'\text{”} \mapsto 1\}$ Rejected by g and rejected by f

  $\{\text{“}x\text{”} \mapsto 2, \text{“}x'\text{”} \mapsto 4\}$ Rejected by g and accepted by f

  However there is no behaviour that is accepted by g and rejected by f, therefore

  $$f \sqsubseteq g$$

  In this case, g is more restrictive about its output than f is.

- Let

  $$\begin{aligned} f &= \langle x' > x \rangle \\ g &= \langle x' \geq x \rangle \end{aligned}$$

  Some example behaviours

  $\{\text{“}x\text{”} \mapsto 2, \text{“}x'\text{”} \mapsto 3\}$ Accepted by g and accepted by f

  $\{\text{“}x\text{”} \mapsto 2, \text{“}x'\text{”} \mapsto 1\}$ Rejected by g and rejected by f

  $\{\text{“}x\text{”} \mapsto 2, \text{“}x'\text{”} \mapsto 2\}$ Accepted by g and rejected by f

  Therefore

  $$f \not\sqsubseteq g$$

- Let

$$f = \langle x > 0 \Rightarrow x' = x + 1 \rangle$$
$$g = \langle x \geq 0 \Rightarrow x' = x + 1 \rangle$$

We might say that f "cares about" inputs such that $x > 0$, whereas g "cares about" inputs such that $x \geq 1$.

- Some example behaviours

$\{$"x" $\mapsto -1,$ "x'" $\mapsto 3\}$ Accepted by g and accepted by f
$\{$"x" $\mapsto 2,$ "x'" $\mapsto 3\}$ Accepted by g and accepted by f
$\{$"x" $\mapsto 2,$ "x'" $\mapsto 4\}$ Rejected by g and rejected by f
$\{$"x" $\mapsto 0,$ "x'" $\mapsto 1\}$ Accepted by g and accepted by f
$\{$"x" $\mapsto 0,$ "x'" $\mapsto 2\}$ Rejected by g and accepted by f

However, we will not be able to find any behaviour such that is accepted by g and rejected by f. Therefore

$$f \sqsubseteq g$$

In this case, g cares about more input values.

Consider the problem of finding the sine of an angle to a limited degree of accuracy.

The requirements specification is

$$\Sigma = \{\text{“}x\text{”} \mapsto \mathbb{R}, \text{“}x'\text{”} \mapsto \mathbb{R}\}$$
$$f = \left\langle 0 \leq x \leq \frac{\pi}{4} \Rightarrow |x' - \sin(x)| < 0.001 \right\rangle$$

This says that if the input is between 0 and $\frac{\pi}{4}$, then the output should equal the sine to 3 decimal places.

- Suppose the actual system computes the sine to 4 decimal places

$$g = \left\langle 0 \le x \le \frac{\pi}{4} \Rightarrow |x' - \sin(x)| < 0.0001 \right\rangle$$

Then we have

$$f \sqsubseteq g$$

The requirements have been met!

- Suppose another system computes sines to 3 places for a larger range of inputs

  $$h = \left\langle \frac{-\pi}{2} \le x \le \frac{\pi}{2} \Rightarrow |x' - \sin(x)| < 0.001 \right\rangle$$

  This system also meets the requirements

  $$f \sqsubseteq h$$

- By the way,

  $$g \not\sqsubseteq h$$

  and

  $$h \not\sqsubseteq g$$

- We could also construct a system that combines the strengths of g and h:

  $$m = \left\langle \frac{-\pi}{2} \le x \le \frac{\pi}{2} \Rightarrow |x' - \sin(x)| < 0.0001 \right\rangle$$

  $$f \sqsubseteq g \sqsubseteq m, \qquad f \sqsubseteq h \sqsubseteq m$$

In a picture

$$\begin{tikzcd} & m & \\ g \arrow[ur, phantom, "\sqsubseteq" sloped] & & h \arrow[ul, phantom, "\sqsupseteq" sloped] \\ & f \arrow[ul, phantom, "\sqsupseteq" sloped] \arrow[ur, phantom, "\sqsubseteq" sloped] & \end{tikzcd}$$

We often use specifications of the form $\langle P \Rightarrow Q \rangle$ where P describes the inputs we care about and Q describes the relationship between the input and the output. In general

$\langle P_0 \Rightarrow Q_0 \rangle \sqsubseteq \langle P_1 \Rightarrow Q_1 \rangle$ if $\langle P_1 \rangle \sqsubseteq \langle P_0 \rangle$ and $\langle Q_0 \rangle \sqsubseteq \langle Q_1 \rangle$

That is $f \sqsubseteq g$ if g cares about at least the inputs that f cares about and is at least as restrictive on the inputs that f cares about.

Some properties of refinement

- Reflexivity

$$f \sqsubseteq f$$

- Transitivity

$$\text{if } f \sqsubseteq g \text{ and } g \sqsubseteq h \text{ then } f \sqsubseteq h$$

- Antisymmetry

$$\text{if } f \sqsubseteq g \text{ and } g \sqsubseteq f \text{ then } f = g$$

- A relation with these properties is called a **partial order**.