Input and Output

Dividing behaviours into inputs and outputs

In this course we follow the convention that

- Names of inputs have no primes: x, y, z
- Names of outputs end with a prime: x', y', z'

Given a signature, e.g.:

$$\Sigma = \{\text{“}w\text{”} \mapsto A, \text{“}x\text{”} \mapsto B, \text{“}y'\text{”} \mapsto C, \text{“}z'\text{”} \mapsto D\}$$

its **input aspect** consists only of inputs

$$\overleftarrow{\Sigma} = \{\text{“}w\text{”} \mapsto A, \text{“}x\text{”} \mapsto B\}$$

and its **output aspect** consists only of outputs

$$\overrightarrow{\Sigma} = \{\text{“}y\text{”} \mapsto C, \text{“}z\text{”} \mapsto D\}$$

Note: No primes

Similarly for behaviours: if

$$b = \{\text{“}w\text{”} \mapsto m, \text{“}x\text{”} \mapsto n, \text{“}y'\text{”} \mapsto p, \text{“}z'\text{”} \mapsto q\}$$

then

$$\begin{aligned} \overleftarrow{b} &= \{\text{“}w\text{”} \mapsto m, \text{“}x\text{”} \mapsto n\} \\ \overrightarrow{b} &= \{\text{“}y\text{”} \mapsto p, \text{“}z\text{”} \mapsto q\} \end{aligned}$$

Note: The input and output aspects of the signature are rather like the source and target of a relation

We can put together signatures and behaviours using the † operator

$$\begin{aligned} \{\text{“}w\text{”} \mapsto A, \text{“}x\text{”} \mapsto B\} \dagger \{\text{“}y\text{”} \mapsto C, \text{“}z\text{”} \mapsto D\} &= \Sigma \\ \{\text{“}w\text{”} \mapsto m, \text{“}x\text{”} \mapsto n\} \dagger \{\text{“}y\text{”} \mapsto p, \text{“}z\text{”} \mapsto q\} &= b \end{aligned}$$

Response Set

For any particular input, $i : \overleftarrow{\Sigma}$, which outputs are acceptable? Define the **response set** of f_Σ for input i as

$$\mathrm{resp}(f_\Sigma, i) \triangleq \left\{ o : \overrightarrow{\Sigma} \mid f(i \dagger o) \right\}$$

Note that

$$f \sqsubseteq g \text{ iff } \forall i : \overleftarrow{\Sigma} \cdot \mathrm{resp}(f, i) \supseteq \mathrm{resp}(g, i)$$

So the direction of the $\sqsubseteq$ symbol might seem a little confusing at first.

The size of the response set is worth noting

- f_Σ is **determined**, for input i, iff $|\mathrm{resp}(f_\Sigma, i)| = 1.$
- f_Σ is **underdetermined**, for input i, iff $|\mathrm{resp}(f_\Sigma, i)| > 1$.
- f_Σ is **overdetermined**,for input i, iff $|\mathrm{resp}(f_\Sigma, i)| = 0$.

Nondeterminism

A specification is **deterministic** if it is determined for every input

$$\forall i : \overleftarrow{\Sigma} \cdot |\mathrm{resp}(f_\Sigma, i)| = 1$$

If a specification is not deterministic, it is **nondeterministic**

$$\exists i : \overleftarrow{\Sigma} \cdot |\mathrm{resp}(f_\Sigma, i)| \neq 1$$

Deterministic specifications are essentially total functions from an input space to an output space

We are interested in nondeterministic specifications because

- They allow us to not specify aspects that are not important.

- They allow us to model components that are not perfectly reliable.
- They allow us to omit quantities from the system boundary.
- They allow us to freely combine specifications with operators such as ‘and’ and ‘or’.

Implementability

While being underdetermined for one or more inputs is not a problem, there is a problem with specifications that are overdetermined for some inputs.

Such specifications are called **unimplementable**

$$\exists i : \overleftarrow{\Sigma} \cdot \mathrm{resp}(f_{\Sigma}, i) = \emptyset$$

Equivalently

$$\exists i : \overleftarrow{\Sigma} \cdot \forall o : \overrightarrow{\Sigma} \cdot \neg f(i \dagger o)$$

A specification that is not unimplementable is **implementable**

$$\forall i : \overleftarrow{\Sigma} \cdot \mathrm{resp}(f_{\Sigma}, i) \neq \emptyset$$

Equivalently:

$$\forall i : \overleftarrow{\Sigma} \cdot \exists o : \overrightarrow{\Sigma} \cdot f(i \dagger o)$$

The job of a system that meets a requirements specification f is to, for each input, i, select an output o from $\mathrm{resp}(f, i)$.

No physical system can select a behaviour from an empty set.

So no physical system will meet an unimplementable specification.

Example:

$$f = \langle |\sin(x) - x'| < 0.001 \wedge x' \geq 0 \rangle$$

This specification requires that the output is approximately the sine of the input, but also that it not be negative. This is a contradictory specification. For example for $x = \frac{-\pi}{4}$ there is no suitable value for x'.

Commandment: Thou shalt not write unimplementable requirements specifications.