Software Foundations

Siddharth Bhat

Spring 2020

Contents

Chapter 1

Introduction

Let $S = \langle X_s, X_s^0, U_s, \xrightarrow{s}, Y_s, h_s \rangle$ be a transition system.

A *finite run* originating from $x_0 \in X^0$ is a finite sequence $(x_0, u_0, x_1, u_1, \ldots u_{n-1}, x_n)$ such that $\forall i \in \{0, \ldots, n-1\},\ x_i \xrightarrow{s}_{u_i} x_{i+1}$. This is sometimes denoted as $x_0 \xrightarrow{s}_{u_0} x_1 \xrightarrow{s}_{u_1} \ldots \xrightarrow{s}_{u_{n-1}} x_n$. In the notation of Tabuada, this is called as the *finite internal behaviour*. A run has information about both states and transitions.

$\langle x_0, x_1, \ldots, x_n \rangle$ is a *finite trajectory* iff $\exists u_0, \ldots u_{n-1}$ such that $x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} \ldots \xrightarrow{u_{n-1}} x_n$ is a finite run. The trajectory has information only about states.

$\langle y_0, y_1, \ldots, y_n \rangle$ is a *finite trace* iff there exists a finite trajectory $\langle x_0 \ldots x_n \rangle$ and $\forall i \in \{0, \ldots, n\},\ y_i = h_s(x_i)$. The finite trace has information only about projections of a state.

The *finite behaviour* of a system S is defined as the union of all finite traces of S. This is notated as $\mathcal{B}(S)$.

The infinite behaviour of a system S is the union of all infinite traces of S, notated as $\mathcal{B}^\omega(S)$.

The questions we are interested in answering are:

- What is the algebra of systems? How do we compose systems?
- Can we look at the definition of a system and predict its behaviour? (Simulation and Bisimulation). This is different from extensionally looking at traces of the system.

Chapter 2

System composition

- Run: sequences of internal states
- Trace: sequences of external states

Definition 1 *Let A and B be two transition systems. The interconnect $\mathcal{I}$ between A and B is any relation between x_a, u_a, x_b, u_b (u_a, u_b are the action spaces of A and B):*

$$\mathcal{I} \subseteq X_A \times U_A \times X_B \times U_B$$

If we want to feed both systems the same input, we can have $\mathcal{I} \equiv U_a = U_b$.

If we want to have both systems produce the same output, then we should have $\mathcal{I} \equiv h_a(x_a) = h_b(x_b)$

Intuitively, the relation $\mathcal{I}$ provides constraints on the direct product of the two systems that we want to enforce.

If we want to have feedback, $\mathcal{I} \equiv u_b = h_a(x_a) \wedge u_a = h_b(x_b)$

Definition 2 *If A, B are systems and $\mathcal{I}$ is the interconnect, then we have $A \times_{\mathcal{I}} B \equiv \langle X_c, X_c^0, U_c, \xrightarrow{c}, Y_c, h_c \rangle$:*

$$\begin{aligned}
X_c &\equiv \{(x_a, x_b) : x_a \in X_a, x_b \in X_b \wedge (\exists u_a \in U_A, u_b \in U_B, (x_a, u_a, x_b, u_b) \in \mathcal{I})\} \\
X_c^0 &\equiv \{(x_a, x_b) \in X_c : x_a \in X_a^0, x_b \in X_b^0\} \\
U_c &\equiv U_a \times U_b \\
(x_a, x_b) \xrightarrow{(u_a, u_b)} (x_a', x_b') &\equiv x_a \xrightarrow{u_a} x_a', x_b \xrightarrow{u_b} x_b', (x_a, u_a, x_b, u_b) \in \mathcal{I} \\
h_c((x_a, x_b)) &\equiv (h_a(x_a), h_b(x_b))
\end{aligned}$$

Question: why not define $U_c \equiv \{(u_a, u_b) : \exists x_a \in X_a, x_b \in X_b, (x_a, u_a, x_b, u_b) \in \mathcal{I}$

Chapter 3

Control

We have a classical plant-controller system, as in control theory. We can model this as:

$$x_c = f_c(x_p) \qquad x_p' = f_p(x_p, x_c)$$

That is, we have x_p^0. We find $x_c^0 = f_c(x_p^0)$. Then, $x_p^1 = f_p(x_p^0, x_c^0), x_c^1 = f_c(x_p^1)$ and the whole process continues.

3.1 Modelling loops

Note that we can model a `while` loop as a pure controller (the predicate) controlling the body of the function.

```
while (C(x)) { x = F(x) }
```

This becomes, as a feedback system:

$$F'(c, x) = \begin{cases} F(x) & \text{if } c = 1 \\ x & \text{otherwise} \end{cases} \qquad C(x) = \dots$$

3.2 Modelling loops with external feedback

```
while (C(x, e)) { x = F(x) }
```

$$x_c = f_c(x_p, e) \qquad x_p' = f_p(x_p, x_c)$$