INF4140 fall term 2012.
Week exercises 10 (Histories)

History functions

Using the techniques on the slides from lecture 10, define the following functions over histories:

1. a Boolean function **_endswith_** : *Hist* × *Set* → *Bool* such that *h* **endswith** *s* is true if *h* is nonempty and ends with an event in the set *s.* For instance, [a, b, c, d] **endswith** {b, c} is false, ε **endswith** {b, c} is false, and [a, b, c, d] **endswith** {b, d} is true.

2. a Boolean function **_beginswith_** : *Hist* × *Set* → *Bool* such that *h* **beginswith** *s* is true if *h* is nonempty and begins with an event in the set *s.* For instance, [a, b, c, d] **beginswith** {b, c} is false, ε **beginswith** {b, c} is false, and [a, b, c, d] **beginswith** {b, a} is true.

3. a Boolean function testing if one history is a subsequence of another history, _ # _ : *Hist* × *Hist* → *Bool*. For instance, [b, d, e] # [a, b, c, d, e], but not [b, e, d] # [a, b, c, d, e].

4. a function __ : *Hist* × *Set* → *Hist* such that h\s is the subsequence of *h* consisting of all events not in the set s.
For instance, [a, b, c, b, d, a]\{d, c} is [a, b, b, a].

5.a function *pending* : *Hist* → *Hist* such that *pending(h)* is the sequence of all *send* messages that not yet have been received. For instance, *pending*([A ↑ B : m1, A ↑ B : m2, A ↑ B : m1, A ↓ B : m1]) is [A ↑ B : m1, A ↑ B : m2]. (In case there are several identical *send* messages, and some but not all of these have been received, you may choose the order of the remaining ones as you wish. For instance, the example above could give the result [A ↑ B : m2, A ↑ B : m1].) Hint: here you need to distinguish between *send* and *receive* events in the definition, and you may need to introduce an additional function.

Coin Machine Users

Consider the coin machine in lecture 10, where the history invariant for the coin machine C is defined over the global history H by:

$$I_C(H/\alpha_C) = 0 \leq \text{sum}(H/\downarrow C) - \text{sum}(H/C\uparrow) < 15$$

In the lecture (page 25), a coin machine agent C was composed with a user agent U with exact change. We will here consider the com-position of C with two different uses, U1 and U2.

(a) User U1 inserts only “5 krone” coins, i.e., U1 only sends five messages. The user is specified by the following invariant:

$$\begin{aligned} I_{U1}(H/\alpha_{U1}) = \quad & H/\{U1\uparrow: \text{one}\} = \varepsilon \\ & \wedge\ \text{sum}(H/U1\uparrow) - \text{sum}(H/\downarrow U1) = \quad 0 \vee 5 \vee 10 \end{aligned}$$

where the formula $P = a \vee b \vee c$ is an abbreviation for
$P = a \vee P = b \vee P = c$.

Write down the global invariant for the system consisting of C and U1. Is it possible to use the legal function to simplify this invariant? For instance, may we say something more precise about the difference $\text{sum}(H/\downarrow C) - \text{sum}(H/C\uparrow)$ compared to what we know from I_C?

(b) User U2 sends both five and one messages to the coin machine, but U2 never cares about collecting the coins returned by the machine. User U2 is specified by the following invariant:

$$I_{U2}(H/\alpha_{U2}) = 0 \leq \text{sum}(H/U2\uparrow) \wedge \text{sum}(H/\downarrow U2) = 0$$

Write down the global invariant for the system consisting of C and U2. Is it possible to use the legal function to simplify this global invariant?