Principles for software composition 2024/25

07 - Temporal and modal logics, GoogleGo and pi-calculus

[Ex. 1] Two processes p_1 and p_2 want to access a single shared resource r. Consider the atomic propositions:

req_i: holds when process p_i is requesting access to r;
use_i: holds when process p_i has had access to r;
rel_i: holds when process p_i has released r.

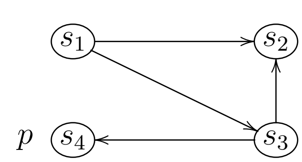
with $i \in [1, 2]$. Use LTL formulas to specify the following properties:

1. *mutual exclusion*: r is accessed by only one process at a time;
2. *release*: every time p_1 accesses r, it releases r after some time;
3. *priority*: whenever both p_1 and p_2 require r, p_1 is granted access first;
4. *no starvation*: whenever p_1 requires r, it is eventually granted access.

[Ex. 2] Three dogs live in a house with two couches and a front garden. Let $couch_{i,j}$ represent the predicate "the dog i sits on couch j" and $garden_i$ represent the predicate "the dog i plays in the front garden".

1. Write an LTL formula expressing the fact that whenever dog 1 plays in the garden then he keeps playing until he sits on some couch (but he may also play forever).
2. Write a CTL formula expressing the fact that dog 2 eventually plays in the garden whenever couch 1 is occupied by another dog.
3. Write a μ-calculus formula expressing the fact that no more than one couch is occupied at any time by dog 3.

[Ex. 3] Given the μ-calculus formula $\Phi = \mu x.((p \wedge \Box x) \vee (\neg p \wedge \Diamond x))$ write its denotational semantics $[\![\Phi]\!]\,\rho$ and evaluate it on the LTS below (where $V = \{s_1, s_2, s_3, s_4\}$ and $P = \{p\}$).



[**Ex. 4**] Write a GoogleGo function that takes one channel `ini` for receiving integers and one channel `ins` for receving strings and returns a channel `outp` where all the messages received on `ini` and `ins` will be paired.
Hint: define a `struct` *to form pairs*

[**Ex. 5**] Write a GoogleGo function that takes two channels `f` and `q` and tries to send the stream of Fibonacci numbers on `f` but quits when it receives `true` on channel `q`. Write a `main` program to test the function by printing the first 10 Fibonacci numbers.

[**Ex. 6**] The *asynchronous* π-calculus requires that outputs have no continuation:

$$p \ ::= \ \mathbf{nil} \ | \ \overline{x}\langle y\rangle \ | \ x(y).p \ | \ \tau.p \ | \ [x=y]p \ | \ p+p \ | \ p|p \ | \ (x)p \ | !p$$

Show that any process in the original π-calculus can be represented in the asynchronous π-calculus using an extra (fresh) channel to simulate explicit acknowledgement of name transmission.

[**Ex. 7**] The *polyadic* π-calculus allows communicating more than one name in a single action, i.e., its action prefixes are of the form:

$$\pi \ ::= \ \tau \ | \ \overline{x}\langle z_1, ...z_n\rangle \ | \ x(z_1, ...z_n)$$

The polyadic extension is useful especially when studying types for name passing processes. Show that the polyadic π-calculus can be encoded in the ordinary (monadic) π-calculus by passing the name of a private channel through which the multiple arguments are then passed in a sequence.