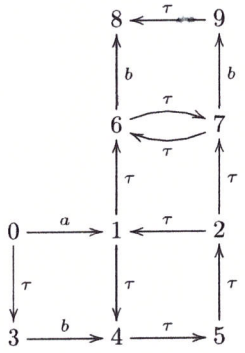
Home Assignment MACS 1 - part II - 23 Oct 2015

Remarks:

1. The ultimate date for returning this assignment is **Monday, November 9, 2015 (strict).**
2. All answers can be found by hand in principle, but the use of computer tools is advised. In that case, mention which tool you used, explain the input, commands, and output results.
3. Return your answers in two files, by E-mail to `J.C.vandePol@utwente.nl`:
   - A document **in pdf** with the answers and short explanation to the questions;
   - A **zip-file** containing files with all inputs and outputs for the tools that you used, plus a README.txt that describes exactly how I can reproduce your results.
4. You should work individually; it is not allowed to collaborate.

A. Confluence in LTS. Consider the following LTS, with initial state 0:



1. Determine the *maximal* confluent subset of τ-transitions; show which divergent pairs of transitions you have to check, and if (and how) they are joinable.
2. Choose one representation map based on the maximal set.
3. How many choices of valid representation maps are there in this case?
4. Draw the reduced LTS according to your chosen representation map.

B. Designing a Double-ended Task Queue

1. Next, we design a data type to represent and merge the contents of a queue of clients. For this we define a sort Queue over a further unspecified sort Client:

```
sort Task;
sort Queue = struct empty | addFirst(Task,Queue) ;
map append: Queue # Queue->Queue;
    addLast: Queue # Task->Queue;
    first: Queue -> Task;
    rest: Queue -> Queue;
var d:Task;
    k,l:Queue;
eqn first(addFirst(d,l)) = d;
    rest(addFirst(d,l)) = l;
    append(empty,l) = l;
    append(l,empty) = l;
    append(addFirst(d,l),k) = addFirst(d,append(l,k));
    addLast(l,d) = append(l,addFirst(d,empty));
```

   Question: Prove that the TRS consisting of these six equations (viewed as rewrite rules from left to right) is terminating. Do this by providing an monotone interpretation of the function symbols. Show for the last two rules in detail that they are decreasing according to your interpretation.

2. We now extend our original specification with a function that returns the last element of the list.

```
map getLast: Queue -> Task;
var d:Task;
    k,l:Queue;
eqn getLast(addFirst(d,empty)) = d;
    getLast(append(k,addFirst(d,l))) = getLast(addFirst(d,l));
```

   Question: Show that the resulting TRS (combining all rules so far) is not confluent, by providing a term (possibly containing variables) and two reductions from that term to distinct normal forms.

3. Write down all critical pairs, and check if they are joinable.

4. Apparently, this TRS is not complete. Add new equations (possibly inspired by Knuth-Bendix completion), to obtain a terminating and confluent TRS extending these ~~six~~ eight equations. Explain your approach and check your answer.

C. Confluence for Process Specifications. We consider a small bike shop with two bike repair men, sharing a common pump. The repair men require the pump twice in their repair sequence, as in the following specification. The pump-request and pump-acknowledgement are supposed to synchronize. We are only interested in the sharing of the pump, so we hide all other actions, unrelated to pumping.

```
sort Fiets = struct gazelle | sparta | batavus ;
sort Maker = struct jan | piet ;

act banderaf, banderop: ...;
act pomp_req, pomp_ack, pomp, lucht: ...;

proc FietsenMaker(m:Maker) =
   sum f:Fiets . banderaf(f,m) . pomp_req(f) . banderop(f,m) . pomp_req(f) . FietsenMaker(m);

proc FietsPomp = sum f:Fiets .  pomp_ack(f) . lucht(f) . FietsPomp ;

proc WerkPlaats =
  allow({...},
  comm({...},
    FietsenMaker(jan) || FietsenMaker(piet) || FietsPomp
  ));

init hide({...},WerkPlaats);
```

1. Finish the specification of the data actions, including their sorts and the allow, comm and hide sections.
2. Provide a linear specification of a repair man (FietsenMaker) and the Pump (Fiets-Pomp).
3. Provide a linear specification of the whole repair shop (WerkPlaats).
4. The hiding operator results in several tau-summands. Check which of these tau-summands can be marked confluent. Explain the result in terms of the original actions (banderop, banderaf, lucht).
5. Generate the state space with and without confluence reduction and report the size of the state spaces.
6. What equivalence should hold between the full and the reduced state space? Check that the equivalence holds indeed.