Concurrency Theory (WS 2016) Out: Thu, 19 Jan Due: Wed, 25 Jan

Exercise Sheet 11

D'Osualdo, Lederer, Schneider Technische Universität Kaiserslautern

Problem 1: Bisimulation of Petri nets

In the last lecture, we proved that bisimilarity between two Petri nets is decidable if one of the nets is bounded. In this exercise you are asked to give formal proof of some claims that we only informally justified in class.

Fix an ordinary labelled Petri net (N, λ_N) and a bounded labelled Petri net (A, λ_A). Recall that an ordinary net is one that has only weights 1 or 0. Let M be a marking of N, s_0 the initial marking of A, $s \in \mathcal{R}_A(s_0)$ a reachable marking of A and $n = |\mathcal{R}_A(s_0)|$.

a) Prove the following claim:

$$M \sim s \iff M \sim_n s \text{ and } \mathcal{R}_N(M) \subseteq B$$

where $B = \{M' \mid \exists s' \in \mathcal{R}_A(s_0)\colon M' \sim_n s'\}$

b) Prove that $M \sim_n M\lfloor_n$.

c) In the proof, we made the assumption that N is ordinary and we claimed this is without loss of generality. What needs to be changed in the proof if N is not ordinary?

[Hint: *Use the game characterization of bisimulation.*]

Problem 2: Problems in $\boldsymbol{\nu}$-free CCS

In this exercise, we want to establish the decidability status of two verification problems for CCS processes when the use of restrictions is forbidden. The $\boldsymbol{\nu}$-free fragment of CCS is the set of processes and definitions that do not contain any restriction $\boldsymbol{\nu}x.P$.

Technically, you can assume wlog that $\boldsymbol{\nu}$-free definitions take the form:

$$\mathsf{A}[\,\vec{x}\,] := \sum_{i \in I} \alpha_i.(\mathsf{A}_1[\vec{x_i}] \parallel \cdots \parallel \mathsf{A}_{n_i}[\vec{x_{n_i}}])$$

for some finite I.

Our aim is to prove the following claims for some finite set of $\boldsymbol{\nu}$-free definitions Δ:

Reachability
Given two $\boldsymbol{\nu}$-free CCS processes P and P', it is decidable whether P' is reachable from P.

Definition reachability
Given a $\boldsymbol{\nu}$-free CCS process P and some process identifier A, it is decidable whether a term containing $\mathsf{A}[\dots]$ is reachable from P.

These two claims can be proven by using a single encoding from $\boldsymbol{\nu}$-free CCS to some known model of computation which allows us to conclude the two claims. For the sake of simplicity, you can assume P and P' above do not contain sums.

a) Define and explain your encoding.

b) Test your encoding for $\mathsf{A}[\,x, y\,] := x.\mathsf{A}[\,y, x\,]$ and write down the resulting representation.

c) Use your encoding to conclude the two claims above.

Problem 3: Stratified Bisimulation

Let $(S_1, \rightarrow_1)$, $(S_2, \rightarrow_2)$ be LTS and $s_1 \in S_1$, $s_2 \in S_2$.

a) Prove that $s_1 \sim s_2$ iff $\forall m \in \mathbb{N}\colon s_1 \sim_m s_2$.

b) Prove that $s_1 \sim_m s_2$ iff Duplicator has a winning strategy for m turns.

Problem 4: Implementing a Queue

Let Γ be a finite alphabet. A FIFO queue over Γ can be specified by means of sequential processes as follows:

$$\begin{aligned} \mathsf{Queue}_\epsilon &:= \sum_{a \in \Gamma} enq_a.\mathsf{Queue}_a \\ \mathsf{Queue}_{bw} &:= \sum_{a \in \Gamma} enq_a.\mathsf{Queue}_{bwa} + deq_b.\mathsf{Queue}_w \qquad \text{for all } w \in \Gamma^* \end{aligned}$$

a) Implement the same behaviour in CCS using only finitely many definitions.

b) Use the algebraic rules seen in the lecture to show that your implementation from 4a is weakly bisimilar to the one given above.

   You will need the fact that solutions of the same set of weak-bisimilarity equations are weakly bisimilar to each other. In the context of this exercise, assuming your implementation represents a queue $w \in \Gamma^*$ with a process Q_w, your task reduces to proving that:

$$\begin{aligned} Q_\epsilon &\approx \sum_{a \in \Gamma} enq_a.Q_a \\ Q_{bw} &\approx \sum_{a \in \Gamma} enq_a.Q_{bwa} + deq_b.Q_w \qquad \text{for all } w \in \Gamma^* \end{aligned}$$

   Since our sequential specification trivially satisfies the same equations, you could then conclude that $\mathsf{Queue}_w \approx Q_w$ for all $w \in \Gamma^*$.