Concurrency Theory, August–November 2019

Assignment 2, 13 November, 2019
Due: 24 November, 2019

Note: Only electronic submissions accepted, via Moodle.

1 Trace theory and distributed automata

Notation

- For $w \in \Sigma^*$ and $X \subseteq \Sigma$, $w{\downarrow_X}$ denotes the projection of w with respect to X—that is, the word obtained by erasing all letters not in X from w. Formally, $\varepsilon{\downarrow_X} = \varepsilon$ and $wa{\downarrow_X} = w{\downarrow_X} \cdot a$, if $a \in X$ and $wa{\downarrow_X} = w{\downarrow_X}$ otherwise.
- A trace alphabet is a pair (Σ, I) where $I \subseteq (\Sigma \times \Sigma)$ is an irreflexive, symmetric independence relation. The complement of I, $D = (\Sigma \times \Sigma) \setminus I$, is called the dependence relation.
- Given a trace alphabet (Σ, I), $u \sim v$ denotes that u and v are trace equivalent.
- Given a distributed alphabet $(\Sigma_1, \Sigma_2, \ldots, \Sigma_k)$,
  - $\Sigma = \bigcup_{i \in \{1,2,\ldots,k\}} \Sigma_i$. For $a \in \Sigma$, $\mathrm{loc}(a) = \{i \mid a \in \Sigma_i\}$.
  - $I_{\mathrm{loc}} = \{(a,b) \mid \mathrm{loc}(a) \cap \mathrm{loc}(b) = \emptyset\}$ is the independence relation induced by loc and $D_{\mathrm{loc}} = (\Sigma \times \Sigma) \setminus I_{\mathrm{loc}}$ is the corresponding dependence relation.

Questions

1. Given a distributed alphabet $(\Sigma_1, \Sigma_2, \ldots, \Sigma_k)$ and a pair of words u, v, prove that $u \sim v$ if and only if $u{\downarrow_{\{a,b\}}} = v{\downarrow_{\{a,b\}}}$ for every pair of letters $(a,b) \in D$.

2. Let $(\Sigma_1, \Sigma_2, \ldots, \Sigma_k)$ and $(\Sigma'_1, \Sigma'_2, \ldots, \Sigma'_m)$ be two distributed alphabets with location functions loc and loc$'$, respectively, that induce the same independence relation—that is, $I_{\mathrm{loc}} = I_{\mathrm{loc}'}$.

   (a) Suppose $\mathcal{A}$ is an asynchronous automaton over $(\Sigma_1, \Sigma_2, \ldots, \Sigma_k)$. Show that there exists another asynchronous automaton $\mathcal{A}'$ over $(\Sigma'_1, \Sigma'_2, \ldots, \Sigma'_m)$ such that $L(\mathcal{A}) = L(\mathcal{A}')$.

   (b) Suppose $\mathcal{A}$ is a direct product automaton over $(\Sigma_1, \Sigma_2, \ldots, \Sigma_k)$. Will there always another direct product automaton $\mathcal{A}'$ over $(\Sigma'_1, \Sigma'_2, \ldots, \Sigma'_m)$ such that $L(\mathcal{A}) = L(\mathcal{A}')$? Prove the statement or construct a counterexample.

   (c) What happens in the case of synchronized product automata?

2 Equivalences on transition systems

3. Show that failure equivalence is decidable for finite-state transition systems. Think of representing a failure pair (w, X) as a word $w.X$ over $\Sigma \cup 2^\Sigma$.

4. Consider the following extension of failure equivalence.

   - Given a transition system $TS = (Q, \rightarrow, q_{in})$ over Σ and a state $q \in Q$, define $L(q)$ to be the language of TS with the initial state shifted to q — that is, $L(q)$ is the language of $TS_q = (Q, \rightarrow, q)$.

- A *language future* is a pair (w, L) such that $w \in \Sigma^*$ and $L \subseteq \Sigma^*$. Given a transition system $TS = (Q, \rightarrow, q_{in})$, we associate the set of language futures $LF(TS) = \{(w, L) \mid \exists q. q_{in} \xrightarrow{w} q, L = L(q)\}$.
  As usual, we say that TS_1 and TS_2 are language future equivalent if $LF(TS_1) = LF(TS_2)$.

(a) Compare the discriminating power of failure equivalence and language future equivalence.

(b) Compare the discriminating power of language future equivalence and strong bisimulation equivalence.
