Principles Of Model Checking Exercises Solutions

Dario Bekic

December 5, 2025

1 Introduction

These are my solutions for the exercises of the book Principles Of Model Checking by Christel Baier and Joost-Pieter Katoen

2 3 Linear Time Properties

2.1 Exercise 3.1

This is like giving a regular grammar.

$$
\begin{aligned}
&S \to aS' | \emptyset S'' \\
&S' \to aS''' | b''' | \\
&S'' \to aS''' | b''' \\
&S''' \to aS'
\end{aligned}
$$

Exercise 3.2

a) $f : TS \mapsto TS^*$ (set TS of transitions systems with terminal states and TS^* is the one without). $f(S, Act, \to, I, AP, L, F) = (S \sqcup \{s_f\}, Act, \to \cup \{(t, \tau, s_f) | t \in F\}, I \setminus F, AP, L \cup \{(s_f, \emptyset)\})$ by viewing everything as a set.

b) Suppose $\sigma \in Traces(TS_1^*) \wedge \sigma \notin Traces(TS_2^*)$. Let π_1 be a TS_1^* path such that $Trace(\pi_1) = \sigma$. Either $\exists q.\pi_q = s_f$ or not. Assume the first case, and let j be the index of the first element in π_1 such that $\pi_1 \neq s_f$. The path fragment $\pi_1' = \pi_{1,1}, ..., \pi_{1,j}$ is a maximal path in TS_1 since $\pi_{1,j}$ is a terminal state by construction. By hypothesis there must exist a path $\pi_2' \in TS_2$ such that $Trace(\pi_1') = Trace(\pi_2')$ thus also π_2' must be a maximal initial finite fragment path i.e. must end in a terminal state of TS_2. Now we can extend the path π_2' with infinite loops and since s_{fin} i.e. the phantom state added to TS_2^* has no atomic proposition its trace will be equal to that of π_1. Namely, $\sigma = Trace(\pi_1) = Trace(\pi_1') = Trace(\pi_2') = Trace(\pi_2)$, contradicting the claim that $\sigma \notin Traces(TS_2^*)$.

Exercise 3.3

Algorithm 1 BFS for Invariant Checking (shortest counterexample)

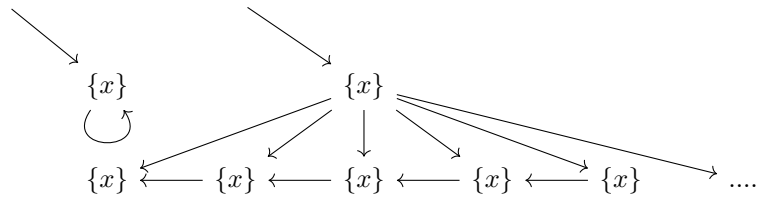
```
1: procedure BFSInvariant(TS(S, Act, →, I, AP, L), φ)
2:     Q ← emptyQueue()
3:     pred ← emptyMap() ▷ pred[s] is predecessor of s; absent means unseen
4:     for all s ∈ I do
5:         if L(s) ⊭ φ then
6:             return ⟨false, [s]⟩
7:         end if
8:         Q.enqueue(s)
9:         pred[s] ← ⊥                    ▷ mark as seen; ⊥ denotes start of path
10:    end for
11:    while ¬Q.empty() do
12:        u ← Q.dequeue()
13:        for all v ∈ post(u) do
14:            if v ∉ keys(pred) then              ▷ first time seen ⇒ shortest
15:                pred[v] ← u
16:                if L(v) ⊭ φ then
17:                    return ⟨false, Reconstruct(pred, v)⟩
18:                end if
19:                Q.enqueue(v)
20:            end if
21:        end for
22:    end while
23:    return ⟨true, emptyList⟩        ▷ invariant holds on all reachable states
24: end procedure
25: procedure Reconstruct(pred, t)
26:    path ← emptyList
27:    x ← t
28:    while x ≠ ⊥ do
29:        path.append(x)
30:        x ← pred[x]
31:    end while
32:    return reverse(path)
33: end procedure
```

Exercise 3.4

a) $\Longrightarrow$: pick a finite path fragment $\pi \in PathFragment_{fin}(TS)$ such that $Trace(\pi) \in Traces_{fin}(TS) \wedge Trace(\pi) \notin Traces_{fin}(TS')$. Call σ the size of π. I can extend π to a maximal path π' such that $Trace(\pi') \in Traces(TS) \wedge Trace(\pi') \in Traces(TS')$. By $Trace(\pi') \in Traces(TS')$

there must exist a path $\rho' \in Paths(TS')$ such that $Trace(\rho') = Trace(\pi')$. Thus, the prefix of length σ of ρ' is a finite path fragment ρ in TS' such that $Trace(\rho) = Trace(\pi)$, contradicting the hypothesis $Traces(\pi) \notin Traces_{fin}(TS')$. The same can be argued by swapping TS and TS' roles, completing the proof.
$\Longleftarrow$: pick a path $\xi \in Paths(TS)$ such that $Trace(\xi) \in Traces(TS) \wedge Trace(\xi) \notin Traces(TS')$. I argue that for any $n \in \mathbb{N}$ there exists a path fragment p^n in TS' such that its trace its equal to the prefix of length n of ξ and that it can be extended to path fragment of size $n+1$ which is equal to the trace of the prefix of length $n+1$ of ξ. The first part of the statement comes directly from the assumption that the finite traces sets equal each other. The second part comes from the fact any trace in both transition systems can be given by only one path (or path fragment), thus for the prefix of length $n+1$ of ξ i need necessarily to extend p^n because if there was another path different from p^n, call it p^*, then their prefix trace of size n would be the same, contradicting the claim. Suppose that in fact such p^* existed, then it may share a prefix of states with p^n, let $q < n$ be the last state they share, i.e. $p^*_q = p^n_q$ then there should be two states p^*_{q+1} and p^n_{q+1} such that $L(p^*_{q+1}) = L(p^n_{q+1})$ because p^*, p^n share the same trace up to the first n states, but this contradicts AP determinism.

b)



Exercise 3.5

a) `false` $\triangleq \emptyset$

b) `initEqualZero` $\triangleq \{A_0A_1, ... \in (2^{AP})^\omega | \exists i \in \mathbb{N}.A_i = \{x = 0\} \wedge \forall j < i.A_j = \emptyset\}$

c) `initDiffZero` $\triangleq (2^{AP})^\omega \setminus$ `initEqualZero`

d) This seems not to be safety nor liveness.

e) Liveness property

f) Liveness property

g) `initEqualZero` $\triangleq \{A_0A_1, ... \in (2^{AP})^\omega | (A_i = \{x = 0\} \wedge A_j = \{x > 0\} \wedge$ i even, j odd$) \vee (A_i = \{x = 0\} \wedge A_j = \{x > 0\} \wedge$ i odd, j even$)\}$

h) `true` $\triangleq (2^{AP})^\omega$

Exercise 3.6

a) Invariant: `Never` $\triangleq \{A_0A_1, ... \in (2^{AP})^\omega | A \notin A_i \forall i\}$

b) Safety: `Once` $\triangleq \{A_0A_1, ... \in (2^{AP})^\omega | \exists i.A \in A_i \wedge \forall j.(j < i \vee j > i) \wedge A \notin A_j\}$

c) Liveness: `AlternateInfinitely` $\triangleq \{A_0A_1, ... \in (2^{AP})^\omega | \exists i, j, q.j > i \wedge A_j = \{A\} \wedge q > i \wedge A_q = \{B\} \wedge (\forall t > i.A_t = \{A\} \implies \exists x.A_x = \{B\} \wedge \forall c.i < c < x.B \notin A_c) \wedge (\forall t > i.A_t = \{B\} \implies \exists x.A_x = \{A\} \wedge \forall c.i < c < x.A \notin A_c)\}$ (its a liveness property because the i existential witness can be large as one wishes).

d) Liveness `AlternateInfinitely` $\triangleq \{A_0A_1, ... \in (2^{AP})^\omega | \exists i.((\exists j < i.A \in A_j \wedge \forall c.j < c < i.B \notin A_c) \implies B \in i) \wedge (\forall q.q > i \wedge (A \in A_q \implies \exists t.t > q.B \in A_t))\}$

Exercise 3.7

Initial values: $r_1 = 0, r_2 = 1$. y value is given by the value of register r_2 and the formula is $((r_1 \oplus x) \vee (x \wedge r_1))$. r_1 value is given by $(r_1 \oplus x) \vee (r_2 \wedge x)$

a) Let α_{P_i} be a word in P_i and β_{P_i} be a word in $(2^{AP})^\omega \backslash P_i$. $\alpha_{P_1} \triangleq \{r_2, x, y\}(A)^\omega$ where $A \triangleq AP$, $\alpha_{P_2} \triangleq \{r_1, x\} \cdot (\emptyset)^\omega$, $\alpha_{P_3} \triangleq (\{y\}, \emptyset)^\omega$, $\alpha_{P_4} \triangleq \emptyset^\omega$, $\beta_{P_1} \triangleq \{x\}^\omega$, $\beta_{P_2} \triangleq \{r_2, r_1\}^\omega$, $\beta_{P_3} \triangleq \{y\}^\omega$, $\beta_{P_4} \triangleq \{x, r_1\}^\omega$

b) P1) If $x = 1$ is high will be high regardless of r_1: if $r_1 = 0$, $r_1 \oplus x$ will be 1, if $r_1 = 1$ then $x \wedge r_1$ will be 1. Thus it is satisfied.

P2) If $r_2 = 0, r_1 = 0$ in any state, in the next state we will $r_1 = 1$ if $x = 1$ so it is not satisfied.

P3) This is not satisfied because P_1 is satisfied.

P4) False, it is a possible initial state

c) P2, P3, P4 are safety properties and only P4 is an invariant.

i) Let X_i be equal to any set in $\{x, r_1, r_2, y\}$ of size i. S^* is the terminal state of the grammar: with that symbol we can expand to any set of properties.

P2)

$$S \to X_i \setminus \{r_2\}S' | X_i \cup \{r_2\}S$$
$$S' \to X_iS' | X_i \cup \{r_1\}S^*$$

P3)

$$S \to X_iS | X_i \cup \{y\}S'$$
$$S' \to X_i \cup \{y\}S*$$

P4)

$$S \to X_i S | X_i \cup \{x\} \setminus \{r_1\} S^*$$

ii) There is only $P4$: $\phi \triangleq \neg x \vee r_1$

Exercise 3.8

$\Longrightarrow$: this direction holds and directly comes from the definition of closure of a property: $closure(P) = \{\alpha \in (2^{AP})^\omega | pref(\alpha) \subseteq pref(P) = \{\alpha \in (2^{AP})^\omega | pref(\alpha) \subseteq pref(P') = closure(P')\}$.
$\Longleftarrow$: suppose $closure(P) = closure(P')$ but $\alpha \in pref(P) \wedge \alpha \notin pref(P')$. Since $\alpha \in pref(P)$ there exists $\sigma \in P$ such that α is a prefix of σ. By definition of closure, $\sigma \in closure(P)$ which implies $\sigma \in closure(P')$ by hypothesis. But $\sigma \in closure(P') \iff pref(\sigma) \subseteq pref(P')$ by definition of closure, and thus we have $\alpha \in pref(P')$.

Exercise 3.9

We need to show that i) the set $\Phi \triangleq closure(Traces(TS))$ is a LT safety property and ii) that TS satisfies it.

i) imagine there were $\alpha = A_0 A_1, ... \in (2^{AP})^\omega \setminus \Phi$ such that there exists $\pi \in pref(\alpha)$ such that $\Phi \cap \{\beta \in (2^{AP})^\omega | \pi \in pref(\beta)\} \neq \emptyset$. Since $\alpha \notin closure(Traces(TS))$ it means there is a finite prefix of α that is not a prefix of some trace in TS. Let this trace be our π. Then if $\sigma \in \Phi \cap \{\beta \in (2^{AP})^\omega | \pi \in pref(\beta)\}$ it means $pref(\sigma) \subseteq pref(Traces(TS))$ which means $\pi \in pref(Traces(TS))$.

ii) A word ξ is in Φ if and only if $pref(\xi) \subseteq pref(Traces(TS))$. Any path ρ in TS has $Trace(\rho) \in Traces(TS)$ thus $pref(Trace(\rho)) \subseteq pref(Traces(TS))$.

Exercise 3.10

Suppose $\phi \in pref(closure(P))$ then $\exists \rho \in closure(P)$ such that a finite prefix of ρ is ϕ. But any element in $closure(P)$ is just a word whose prefixes are elements of $pref(P)$ thus any prefix of ρ is a an element of $pref(P)$, even ϕ, thus $\phi \in pref(P)$.

Exercise 3.11

a) $\texttt{UNION\_SAFE} \triangleq P \cup P'$ is a safety property if P, P' are safety properties. Consider $w \in \texttt{UNION\_SAFE}$. For it to be a safety property, it is sufficient to prove that for any $\rho \in (2^{AP})^\omega \setminus (\texttt{UNION\_SAFE})$ and for any finite prefix π of it $(\texttt{UNION\_SAFE}) \cap (\underbrace{\{\alpha \in (2^{AP})^\omega | \pi \text{ is a finite prefix of } \alpha\}}_{\triangleq \texttt{PREFIX}}) = \emptyset$. This equals to say, by distributivity, $(P \cap \texttt{PREFIX}) \cup (P' \cap \texttt{PREFIX}) = \emptyset$. Since P, P' are safety properties we have $\emptyset \cup \emptyset = \emptyset$.

b) $\texttt{INTER\_SAFE} \triangleq P \cap P'$ is a safety property if P, P' are safety properties. Take any word $w \in (2^{AP})^\omega$. There are 2 cases to consider:

   i) $w \notin P \cup P'$: we wish to show that for any finite prefix $\pi \in pref(w)$, $P \cap P' \cap \underbrace{\{\alpha \in (2^{AP})^\omega | \pi \text{ is a finite prefix of } \alpha\}}_{\triangleq \texttt{PREFIX}}$. But since P' is a safety property $P' \cap \texttt{PREFIX} = \emptyset$, thus $P \cap \emptyset = \emptyset$.

   ii) $w \in P' \wedge w \notin P$: we wish to show again that for any finite prefix $\pi \in pref(w)$, $\texttt{INTER\_SAFE} \cap \texttt{PREFIX} = \emptyset$. By commutativity of intersection, $P \cap P' \cap \texttt{PREFIX} = P' \cap P \cap \texttt{PREFIX} = P' \cap \emptyset = \emptyset$. The case of $w \in P \wedge w \notin P'$ is identical.

c) $\texttt{UNION\_LIVE} \triangleq P \cup P'$ is a liveness property if P, P' are liveness properties. This is trivially arguable informally: if i can extend any finite prefix to a trace in $w_p \in P$ or to a trace $w'_p \in P'$ then i can extend any prefix to a trace in $P \cup P'$ (just pick either w_p or w'_p).

d) $\texttt{INTER\_LIVE} \triangleq P \cap P'$ is not necessarily a liveness property. To see a concrete counter-example consider $P \triangleq \{$ from one point there will be only 1s$\}$ and $P' \triangleq \{$ from one point there will be only 0s$\}$. Their intersection is empty, and the empty set is not a liveness property.

Exercise 3.12

1. $closure(P) \subseteq P_{safe}$. Assume there exists $\alpha \in closure(P)$ such that $\alpha \notin P_{safe}$. Since $\alpha \in closure(P)$, for any finite prefix β of α, $\beta \in pref(\alpha) \implies \beta \in pref(P)$. So pick any prefix β of α, since P_{safe} is a safety property we have $P_{safe} \cap \{\gamma | \beta \in pref(\gamma)\} = \emptyset$. But we also proved $\beta \in pref(P)$, and since P contains a subset of P_{safe} it means $\exists \lambda \in P_{safe}.\beta \in pref(\lambda)$. But this would imply $\lambda \in P_{safe} \cap \{\gamma | \beta \in pref(\gamma)\} \neq \emptyset$.

2. $P_{live} \subseteq \underbrace{P \cup ((2^{AP})^\omega \setminus closure(P))}_{\texttt{EXPR}}$. Suppose $w \in P_{live}$ but not in $\texttt{EXPR}$. This can only happen if $w \notin P$ which implies $w \notin P_{safe}$. We wish to prove $w \in \texttt{EXPR}$ if $w \notin P_{safe}$. This is equal to proving $w \notin P_{safe} \implies w \notin closure(P)$ by definition of $\texttt{EXPR}$. We can prove the contrapositive: $w \in closure(P) \implies w \in P_{safe}$ which we proved in the first point.

Exercise 3.13

One can of course use the Decomposition theorem, but there is a way of giving explicitly the two properties by using intuition.

One would of course first start from characterizing P: maybe the exercise is tricky and P is a safe or liveness property. Proving one such result would let us quickly end the exercise.

a. Is P a safety property? One could argue that the set of bad prefixes is $\{w_1, w_2, .. \in (2^{AP})^\omega | \exists n \in \mathbb{N}. w_n = \{a, b\} \wedge \exists j \in \mathbb{N}. j < n \wedge a \notin w_j\}$. However, the condition to have *infinitely* many $j \in \mathbb{N}. b \in w_j$ cannot be captured by a finite prefix. In conclusion, P is not a safety property.

b. Is P a liveness property? No, clearly $w' = \{\{b\}, \{a, b\}\}$ cannot be extended to a word satisfying P.

Even though we were unsuccessful in proving safety or liveness, we identified the parts that were problematic during our proof attempt.

In particular:

$$P_{safe} \triangleq (2^{AP})^\omega \setminus \{w_1, w_2, .. \in (2^{AP})^\omega | \exists n \in \mathbb{N}. (w_n = \{a, b\} \wedge \exists j \in \mathbb{N}. j < n \wedge a \notin w_j)\}$$

and

$$P_{live} \triangleq \{w_1, w_2, ... | \overset{\infty}{\exists} j. b \in w_j\}$$

The correctness is immediate.

Exercise 3.14

First, let's put here the transition graphs. Consider the TS for Peterson's algorithm in Figure 37 and for the Semaphore-based mutual exclusion in Figure 38. Note that we have req_1, req_2 labels even though they are not in AP just for ease of read. In general we can solve this exercise by reasoning on the LT properties that the transition systems satisfy or not. As we know, the semaphore transition systems does not guarantee starvation freedom i.e. it is possible that one process will never enter its critical region.

Thus, the trace:

$$w \triangleq \{\{req_1, req_2\}, \{wait_1, req_2, \}, \{crit_1, req_2\}\}^\omega$$

is in $Traces(TS_{sem})$ but not in $Traces(TS_{pet})$.

Exercise 3.15

(a) Regarding satisfaction of E': without *unconditional* fairness assumptions it is a matter of verifying if there exists a path which contains $\{\{a\}, \{a, b\}, \emptyset\}$ and this is true $s_1 \to s_2 \to s_3 \to s_1$, thus this property is never satisfied. For B_1 only E_a because we may loop on $s_1 \to s_2 \to s_1$. For B_2 both E_a and E_b as we need to exit the loop we mentioned because $s_1 \to s_3$ is infinitely enabled. For B_3 only E_b, because of the loops $s_1 \to s_2 \to s_1$ and $s_1 \to s_2 \to s_4 \to s_1$ we always pass by s_1 which has enabled the β transition to s_3.

(b) The same reasoning for E' applies. Now, recall that weaker assumptions restrict *the same* or more traces, thus we do not check again those we proved are not satisfied by strong assumptions. For example, B_1 now

looses E_a as α-transitions from s_1 are not continuously enabled, as we can loop on $s_1 \to s_3 \to s_1$. The same goes for B_3. And we also loose both E_a, E_b for B_2 as we can loop on $s_1 \to s_2 \to s_1$ continuously *alternating* between having α and β transitions.

Exercise 3.16

(a)
1. With no assumption this is false. Let $AP_1 = \{\alpha\}$ and $AP_2 = \{\beta\}$. Then let $\forall s \in S_2.\beta \in L(s), \forall s' \in S_1.\alpha \in L(s')$. Then $\forall A_1, A_2... \in Traces(TS_1||TS_2)$ we have $\beta \in A_i$ but no trace in TS_1 contains such label (as it is not part of TS_1 label set).
2. This is false. Consider Figure 1 where $\langle s_1, s_1' \rangle$ is the initial state and consider $Syn = \{\alpha\}$. It will be impossible to move out of the initial state as s_1' has no outgoing α transition to synchronize on. We have only have one trace: $Traces(TS_1||TS_2) = \{\{Trace(\langle s_1, s_1' \rangle)\}\} = \{\{L_1(s_1)\}\}$

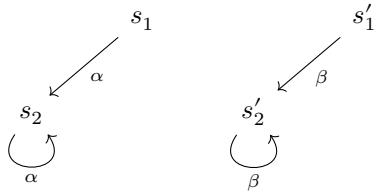


Figure 1: Exercise 3.16. point (a)

(b)
1. With no assumption it is the same as (a).
2. Assuming $AP_2 = \emptyset$ the result is true. Let us restrict to the subset of paths in $TS_1||TS_2$ that use only a transition from TS_1. We argue that for any path $\sigma_1 = s_0, s_1, ... \in Paths(TS_1)$ we have a path $\sigma_2 = \langle s_0', s_1', ... \rangle$ such that $\pi_1(s_i') = s_i$, where π_1 is the left projection. This can be proved easily by induction.

(c)
1. With no assumption it is the same as (a).
2. Assuming $AP = \emptyset$ it is still false. Consider Figure 2. The transition system $TS_1||TS_2$ allows for the path $P \triangleq (\langle s_1, s_1' \rangle \overset{\beta}{\to} \langle s_1, s_1' \rangle)^\omega$ with $Trace(P) = a^\omega$. However this trace is not available to TS_1 whose trace set is $\{\{a \cdot b^\omega\}\}$.

(d) False. Consider Figure 3 and the two transition systems, where we just write the Atomic Propositions that hold: and call the TS on the left TS_1 and the one on the right TS_2. It is obvious that $Traces(TS_1) \subseteq Traces(TS_2)$ as $Traces(TS_1) = Traces(TS_2)$. Clearly for $\mathcal{F}_{strong} = \{\{\alpha\}\}$ TS_2 has no infinite fair traces as it *has* to take the α transition while TS_1 does.

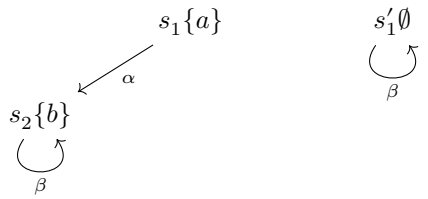


Figure 2: Exercise 3.16. point (c)

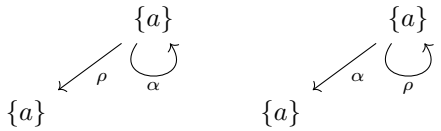


Figure 3: Exercise 3.16. point (d)

(e) We simply make a little change to the transitions systems of the point (d). As you can see in figure 4., we have clearly $Traces(TS_1) = \{a^\omega\} \cup \{a^n \cdot b | n \in \mathbb{N}\} = Traces(TS_2)$ thus $Traces(TS_1) \subseteq Traces(TS_2)$. Consider the liveness property:

$$\mathsf{EVENTUALLY\_B} \triangleq \exists n.b \in A_n$$

Under $\mathcal{F} = \{\emptyset, \{\{\alpha\}\}, \emptyset\}$ we have $TS_2 \models_{\mathcal{F}} \mathsf{EVENTUALLY\_B}$ but $TS_1 \not\models_{\mathcal{F}} \mathsf{EVENTUALLY\_B}$.

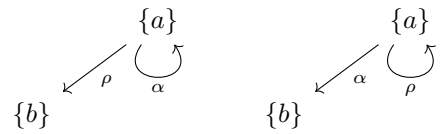


Figure 4: Exercise 3.16. point (e)

Exercise 3.17

It is enough to count all the cycles and see if one of those two non-mutually exclusive (but sufficient) conditions hold: *i*) to access the cycle by using any path we encounter a or *ii*) in at least one state of the cycle we have available either a α or β as outgoing transitions to states where a holds (or where we have proved that it eventually holds).
(Some) cycles are:

1 $s_1 \to s_1$: since in s_1 a holds we conclude by criterion *i*).

2 $s_2 \to s_3 \to s_2$: we have infinitely many times β transition available which sends us to s_1, thus by criterion ii) and point 1. we conclude.

3 $s_3 \to s_4 \to s_5 \to s_4 \to s_6 \to s_3$ both criterion do not hold, thus the property is not satisfied.

Exercise 3.18

(a) It is not realizable as there is no fair path starting from s_1 which is reachable with an α transition from s_0 because of the unconditional assumption $\{\alpha\}$.

(b) By having now the unconditional assumption $\{\delta, \alpha\}$ the loop $s_1 \to s_4 \to s_1$ is a fair path. Thus, $\mathcal{F}_2$ is realizable.

(c) This is too realizable. The loop $s_1 \to s_4 \to s_1$ will enable infinitely many times β and by strong fairness $\{\alpha, \beta\}$ we will be forced to take it infinitely many times thus satisfying the unconditional assumption $\{\beta\}$.

Exercise 3.19

(a) (i)

$$
\begin{aligned}
E &::= R_1(R_2^\omega) \\
R_1 &::= R_1' | \epsilon \\
R_1' &::= \emptyset R_1' | a \\
R_2 &::= \{a\}R_2' | \{b\}R_2 | \{a,b\}R_2 \\
R_2' &::= \{b\}R_2
\end{aligned}
$$

(ii) By the decomposition theorem we have

$$
\begin{aligned}
P_{safe} &= \{A_0 A_1 ... \in (2^{AP})^\omega | ...\} \\
... &= \exists n.(A_n = \{a\} \wedge \forall j.j < n \wedge A_j = \emptyset \wedge \forall k.k > n.A_k = \{a\} \implies A_{k+1} = \{b\}) \cup \emptyset^\omega \\
P_{live} &= P_1 \cup ((2^{AP})^\omega \setminus closure(P)) \\
&= (\{b\} + \{a,b\})^\omega \cup P_1 \cup \{A_0 A_1 ... \in (2^{AP})^\omega | ...\} \cup \{\{a\}A_1 ... | *\} \\
... &= \exists n.(n > 0 \wedge A_n = \{a\} \wedge \exists j < n.A_j \in \{\{a,b\},\{b\}\}) \\
* &= \exists i > 0.A_i = \{a\} \wedge A_{i+1} \neq \{b\}
\end{aligned}
$$

(iii) 1. Suppose $w \in (2^{AP})^\omega \setminus P_{safe}$. If $\nexists i.w_i = \{a\}$ then pick j the first non-empty set of w: it should be clear that any word with this prefix is not in P_{safe}. If $\exists i.w_i = \{a\}$ then it must be:
   i. Either $existsj.j < n \wedge A_j \neq \emptyset$. Then it is sufficient to pick the prefix $A_0 ... A_i$.
   ii. Or(they are not mutually exclusive) $\exists k > i.A_k = \{a\} \wedge A_{k+1} \neq b$. Pick the prefix $A_0, ... A_{k+1}$.

(b) First, let's re-draw the transition system which we can see in Figure 5, where each node is denoted with the pair $s\{L(s)\}$. we work separately for

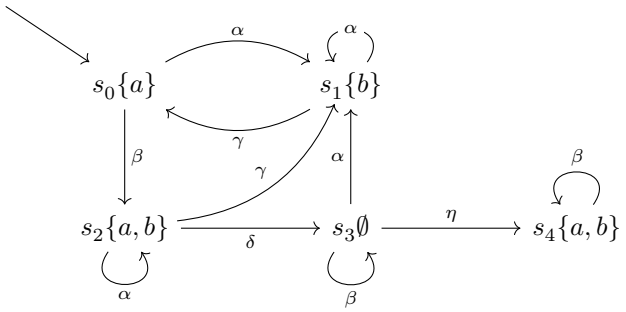


Figure 5: Exercise 3.19. point (b)

the two fairness assumptions. The first one is:

$$\mathcal{F}_1 = (\{\{\alpha\}\}, \{\{\beta\}, \{\gamma, \delta\}, \{\eta\}\}, \emptyset\})$$

First, state s_4 is ruled out as it will not allow for infinite α transitions that is a condition of the unconditional assumptions.

Notice that we will be forced to take the β transition from s_0 to s_2 as it is one of our strong assumptions, forcing us out from the hypothetical loop $s_0 \to s_1 \to s_0$.

Now the strong assumption $\{\gamma, \delta\}$ forces us only to eventually take either γ or δ an infinite number of times as we will pass s_2 an infinite number of times.

Thus, the first part property is satisfied as we will pass an infinite number of times to s_2 as we will re-enter the loop either through $s_2 \to s_1$ or $s_2 \to s_3 \to s_1$.

For the second part of the property, notice that if we pass to s_3 an infinite amount of time, we will need to take η thus invalidating that path. So any fair path passes trough s_3 a finite number of times, so pick as n the last time we are in s_3. Then any fair path passes only through s_0, s_1, s_2 and thus cannot invalidate the second part of the property as any outgoing transition from states s such that $a \in L(s)$ and for $\forall s'.s' \in out(s) \implies b \in L(s')$.

Now recall the other assumption:

$$\mathcal{F}_2 = (\{\{\alpha\}\}, \{\{\beta\}, \{\gamma\}\}, \{\{\eta\}\})$$

Now we are not forced to move with η even if we pass through s_3 an infinite number of times. So say such n existed, we can always loop back to s_2 and then move to s_3(as we can pass through to s_3 an infinite number of times).

Exercise 3.20

3 4 Regular Properties

Exercise 4.1

(a) yes, see Figure 6

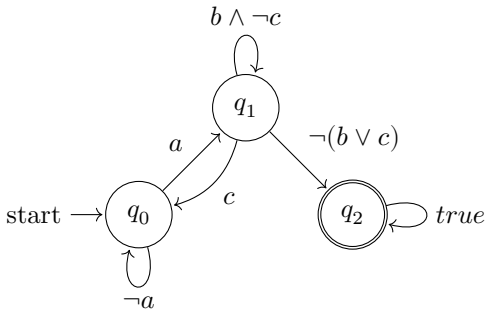


Figure 6: Exercise 4.1 point (a)

(b) yes, see Figure 7

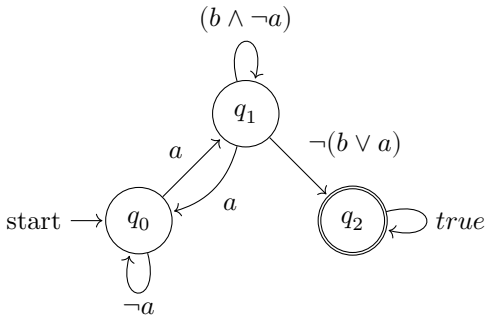


Figure 7: Exercise 4.1 point (b)

(c) No, as $L = c^n b^{n+1}, n \geq 0$ is a subset of the property but is not a regular language. To prove it one can use the Pumping Lemma: let P be the pumping length and let $w = c^L b^{L+1}$, clearly $c^{L+x} b^{L+1}$for $x > 0$ is not in the language.

(d) Yes, see Figure 8

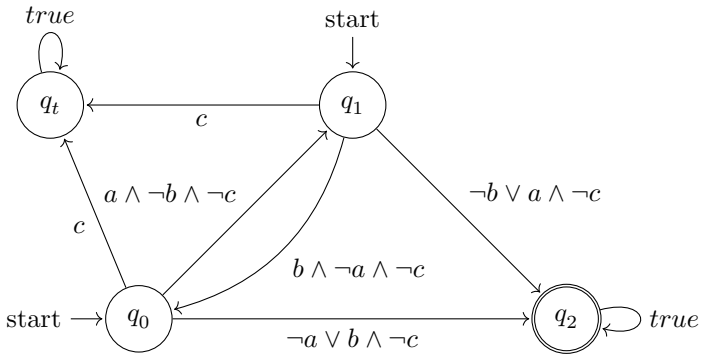


Figure 8: Exercise 4.1 point (b)

Exercise 4.2

1. Let $n = 3$ for simplicity.

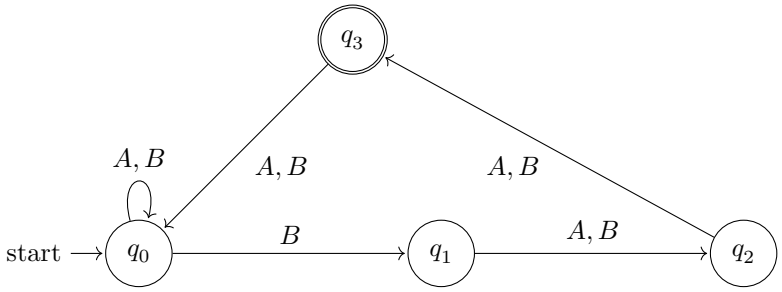


Figure 9: Exercise 4.2 point (a)

2. To determinize the NFA we start from q_0 and with a A transition we can only stay in q_0 while using B we can either move to q_0 or q_1, we call this state $q_{0,1}$.Then from this state by either A or B we go in the state $q_{0,2}$. Then, again by either A or B, we go to state $q_{0,3}$ and then again to state q_0. See Figure 10.

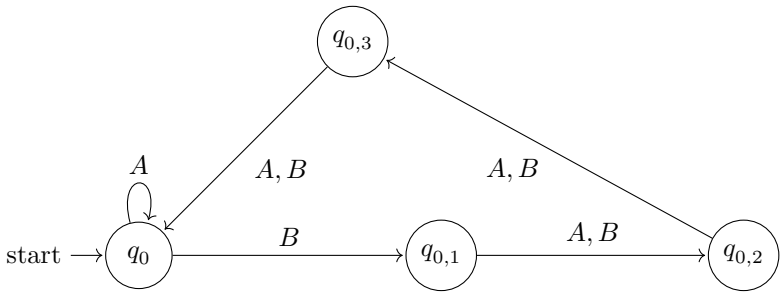


Figure 10: Exercise 4.2 point (b)

Exercise 4.3

(a) (i) See Figure 11

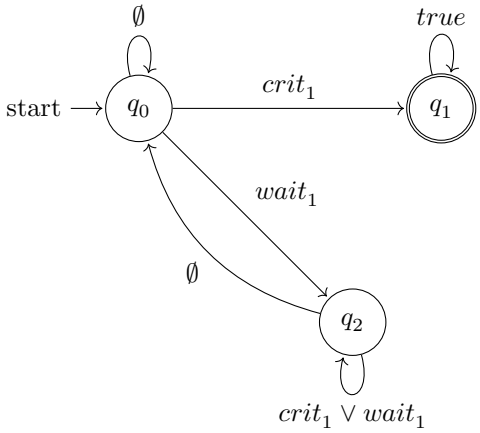


Figure 11: Exercise 4.2 point (b)

(ii) Recall the Semaphore based TS(Figure 38).The product with the previous point's NFA is given in Figure 12, as one can see no state has the right projection equal to q_1.

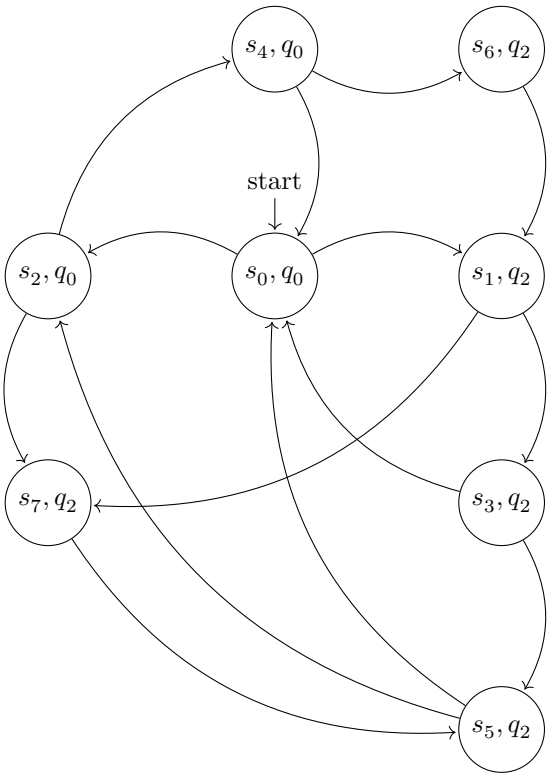


Figure 12: Exercise 4.3. (a) point (ii): Product Graph for Semaphore Algorithm

(b) (i) The NFA can be seen in Figure 13.

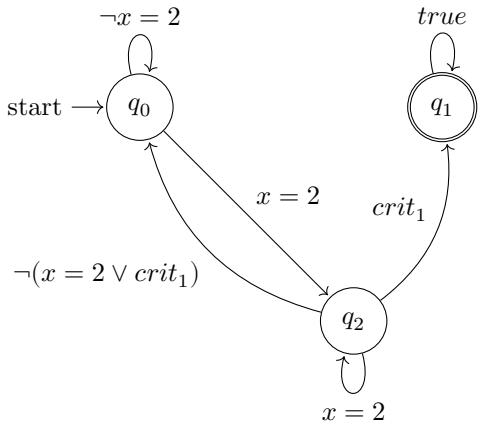


Figure 13: Exercise 4.3 point (b) the NFA of Bad Prefixes for "process 1 never enters its critical section from a state with x=2"

(ii) Recall Peterson's TS in Figure 37 and then we show (part of) the product graph in Figure 14

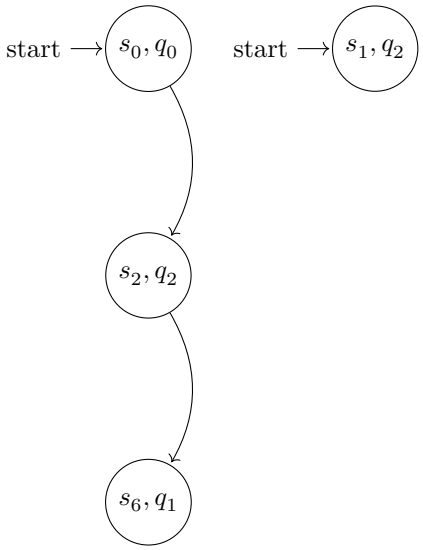


Figure 14: Exercise 4.3 point (b) (ii) part of the product automata for the property "process 1 never enters its critical section from a state with x=2". Clearly we can reach a state s where $L(s) = q_1$. So we conclude the TS does not satisfy the property.

Exercise 4.4

(a) It is true. Take the NFA $N = (Q, \Sigma, \delta, Q_0, F)$ corresponding to $\mathcal{L}$, then by hypothesis this NFA accepts the minimal bad prefixes of P_{safe} and some other bad prefixes. It is enough to remove any outgoing edge from end states in N, obtaining the NFA N', to get an NFA accepting the minimal bad prefixes of P_{safe}. If $w(|w| = n)$ is a minimal bad prefix then it is accepted by N by construction, which means there exists a run π such that $\exists i.\pi_i \in F$ and $\pi_1 \rightarrow^{w_1} \pi_2 \rightarrow^{w_2} ... \rightarrow^{w_n} \pi_i$. Then it is obvious that there cannot be another j such that $j < i$ and $\pi_j \in F$ because this would mean $w' = w_1...w_j$ is a bad prefix (since $L \subseteq BadPref(P_{safe})$) of a minimal bad prefix w. Then we know that N' can imitate this same path and thus accept any minimal bad prefix of P_{safe}. For the other direction, namely that N' accepts only the (minimal) bad prefixes: since N' is N after we remove some edges it cannot expand the language accepted, thus N' accepts only Bad Prefixes of L_{safe} and because no end state has outgoing transition it accepts only the minimal ones.

(b) False. Consider the regular safety property $\emptyset$, then $MinBadPref(P_{safe}) = \emptyset$ and $BadPref(P_{safe}) = (2^{\{0,1\}})^{\omega}$. Now consider $L = \{\{0\}^n\{1\}^n + 1 | \forall n \geq$

0} which is clearly not regular.

Exercise 4.5

First, some considerations on the NFA: it is described by the RE:

$$((bc)^*a.(a^*.(b+ab).a^*.(b+ab))^*.c)^*.(bc)^*a.c$$

. One possible way to describe the bad prefixes represented by this NFA could be: $c \in A_i \implies A_{i-1} = \{a\}$.

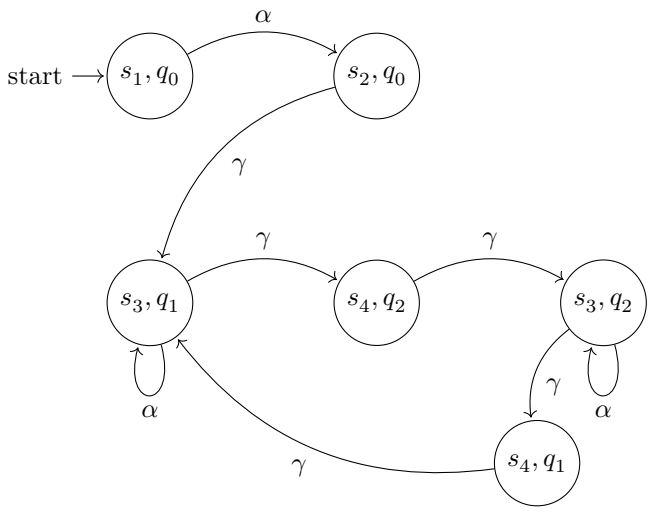


Figure 15: Exercise 4.5, the reachable fragment of the product $TS \otimes A$

Exercise 4.6

I read this property as "Always, if a becomes valid and $b \wedge \neg c$ holds in one of the states *before* then in every state *after* neither a nor b hold until a c holds". I consider the set $AP = \{a, b, c\}$.

(a) See Figure 6.

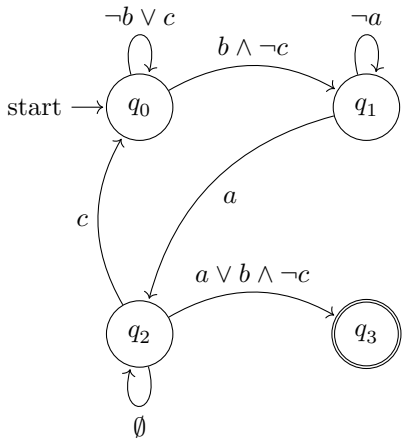


Figure 16: Exercise 4.6 part (a): The NFA A accepting MinBadPref(P_{safe})

(b) The TS satisfies the property as a state of the form x, q_3 is not reachable, see Figure 17.

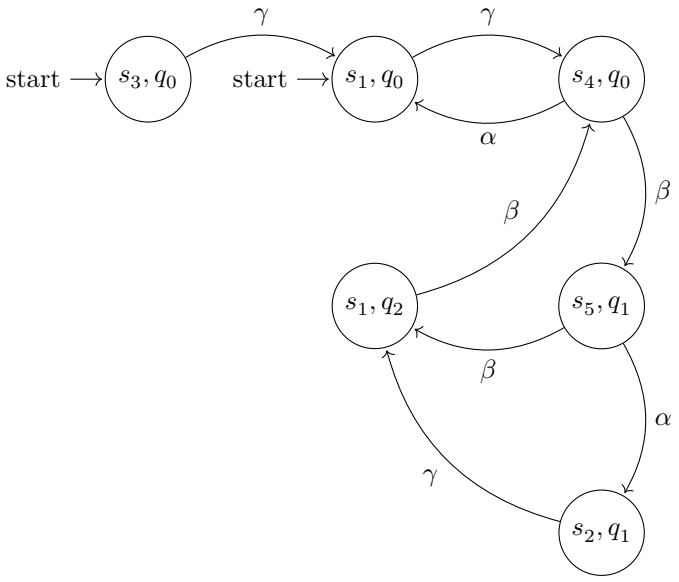


Figure 17: Exercise 4.6 part (b): The Product $TS \otimes A$

Exercise 4.7

(a) True.

$$\underbrace{L(E_1.F^\omega + E_2.F^\omega)}_{L1} = \underbrace{L(E_1).L(F)^\omega}_{L2} \cup \underbrace{L(E_2).L(F)^\omega}_{L3}$$

so any word $w \in L1$ is either a word in $L2$ or $L3$ (non exclusive or).

$$\underbrace{L((E_1 + E_2).F^\omega)}_{L4} = \underbrace{L(E_1 + E_2).L(F)^\omega}_{L5}$$

Suppose, without loss of generality, $w \in L2$, then it is made of a prefix p in $L(E_1)$ and an ω-suffix s in $L(F)^\omega$. Since $p \in L(E_1) \implies p \in L(E_1 + E_2)$ we conclude. For the other direction: if $w \in L4$ then it is made of a prefix p in $L(E_1 + E_2)$ and a suffix in $L(F^\omega)$, since $p \in L(E_1)$ or $p \in L(E_2)$ we conclude.

(b) False. Take $\Sigma = \{a, b, c\}$ as an alphabet, $E = a, F_1 = b, F_2 = c$. Then $a.(a.b)^\omega \in L(E).(F_1 + F_2)^\omega$ by definition of the ω operator. But $L(E.F_1^\omega) + L(E.F_2^\omega) = \{a.(b)^\omega, a.(c)^\omega\}$

(c) True. We can just show

$$\underbrace{L(F.F^*)^\omega}_{L1} = \underbrace{L(F)^\omega}_{L2}$$

Take any word $w \in L1$. w, is by definition, an infinite concatenation of words $w_i \in F.F^*$. We prove that there exists a word $w' \in L2$ such that $\forall i \in \mathbb{N}.w_1 ... w_i$ is a prefix of w'. This equals to say that $w' = w$. By induction:

- i=1: by definition w_1 is a concatenation of one or more (finite) words from F, i.e. $w_1 = w_{1,1}.w_{1,2} ... w_{1,n}$. Let $w'_1 = w_{1,1}, w'_2 = w_{1,2} ... w'_n = w_{1,n}$.
- i=k+1: suppose, by inductive hypothesis, that the prefix $w_1 ... w_k$ is long q. suppose $w_{k+1} = w_{k+1,1}.w_{k+1,2} ... w_{k+1,t}$ then let $w'_{q+1} = w_{k+1,1}, w'_{q+2} = w_{k+1,2} ... w'_{q+t} = w_{k+1,t}$.

For the other direction, namely that $w \in L2 \implies w \in L1$ we note that $L(F) \subseteq L(F.F^*) \implies L(F)^\omega \subseteq L(F.F^*)^\omega$.

(d) False. Take $\Sigma = \{a, b\}$ as an alphabet, $E = a, F = b$. Then $a.(a.b)^\omega \in L(E^*.F)^\omega$ but there is no word with infinite as in $E^*.F^\omega$ as the ω operator is applied only to the language $\{b\}$.

Exercise 4.8

We need to proceed by cases on the form of G, let F be an operator which takes in input a generalized ω-regular expression and returns one equivalent of the form $E + E_1.F_1^\omega + ... E_n.F_n^\omega$:

1. $F(\emptyset) = \emptyset$
2. $F(\epsilon) = \epsilon$
3. $F(A) = \{A\}$

4. $F(G_1 + G_2) = F(G_1) + F(G_2)$

5. $F(G_1.G_2)$: we know by inductive hypothesis that $F(G_1) = E + E_1.F_1^\omega + ... E_n.F_n^\omega$ and $F(G_2) = E' + E'_1.(F'_1)^\omega + ... E'_t.(F'_t)^\omega$ then we need to concatenate finite words of G_1 with finite and infinite words of G_2 and append the infinite words of G_1:

$$F(G_1.G_2) = E.E' + E_1.F_1^\omega + ... E_n.F_n^\omega + E.E'_1.(F'_1)^\omega + ... E.E'_t.(F'_t)^\omega$$

6. $F(G^*)$: by inductive hypothesis $F(G) = E + E_1.F_1^\omega + ... E_n.F_n^\omega$ thus

$$F(G^*) = E^* + E^*.E_1.F_1^\omega + ... E^*.E_n.F_n^\omega$$

7. $F(G^\omega)$: by inductive hypothesis $F(G) = E + E_1.F_1^\omega + ... E_n.F_n^\omega$. Following the same reasoning as for the Kleene closure, $F(G^\omega) = (E \setminus \epsilon)^\omega$.

Exercise 4.9

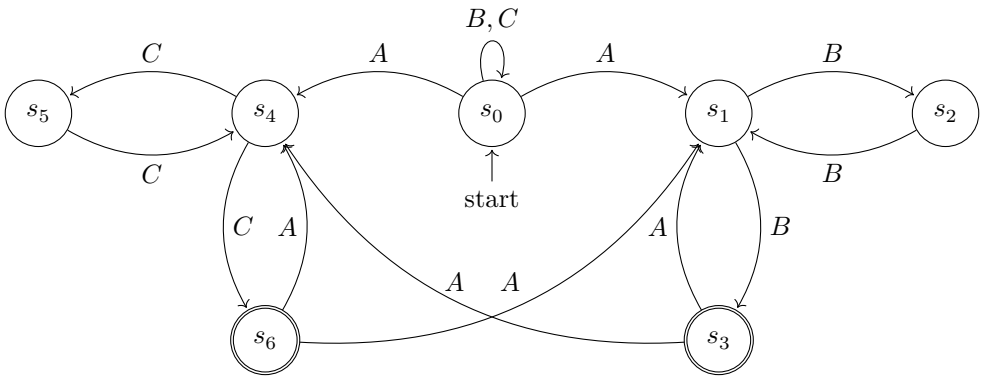


Figure 19: Exercise 4.10 NBA for point (c)

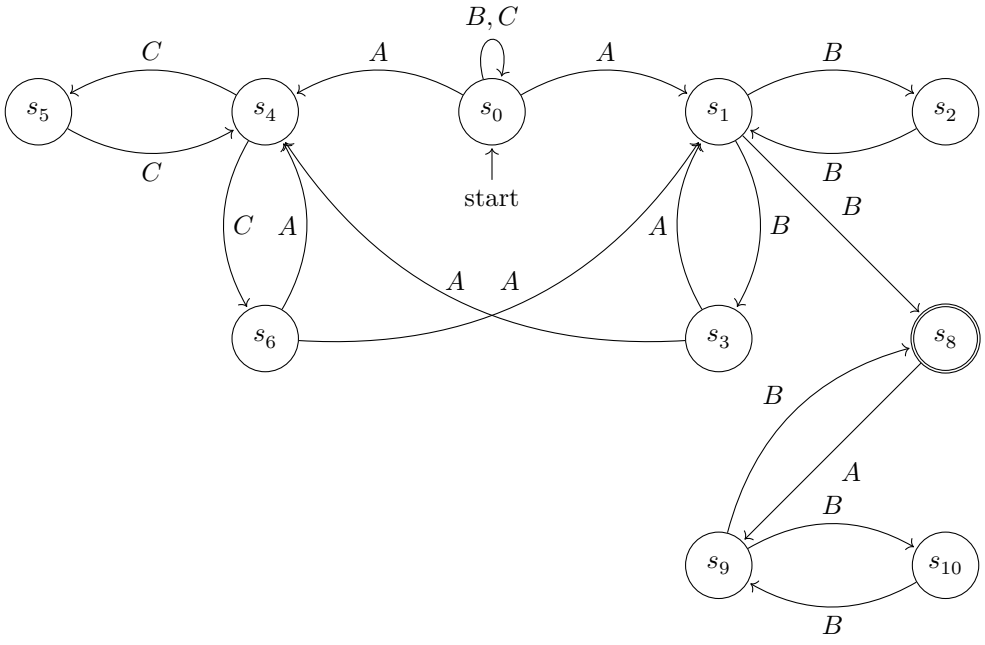


Figure 20: Exercise 4.10 NBA for point (b)

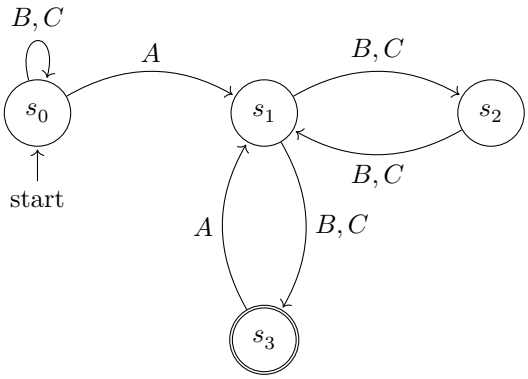


Figure 21: Exercise 4.10 NBA for point (c)

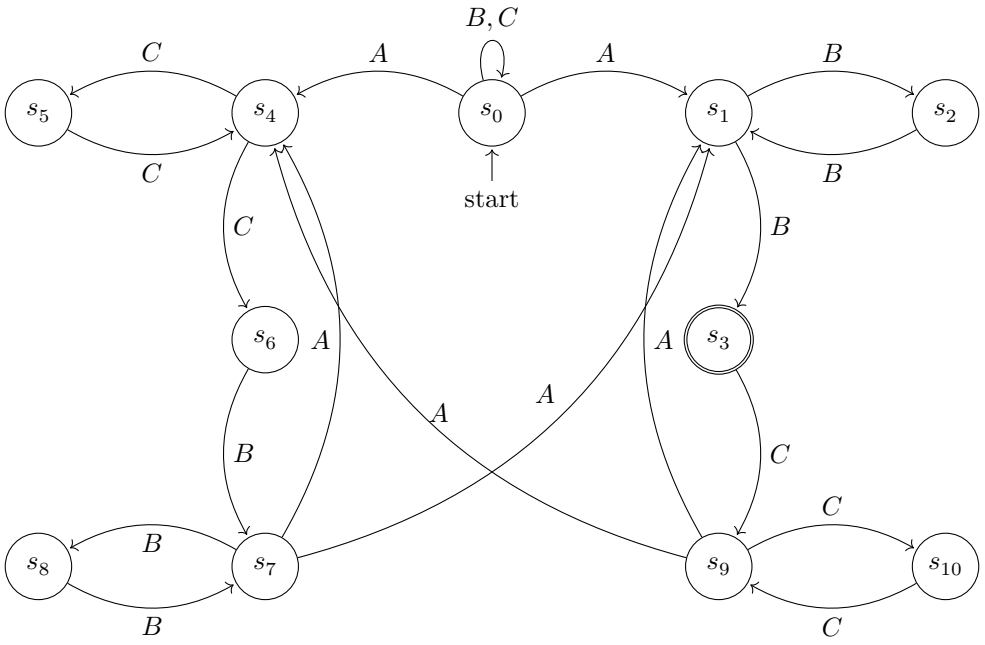


Figure 22: Exercise 4.10 NBA for point (d)

Exercise 4.11

The Automata has two initial states and two final states and it is depicted in Figure 23.

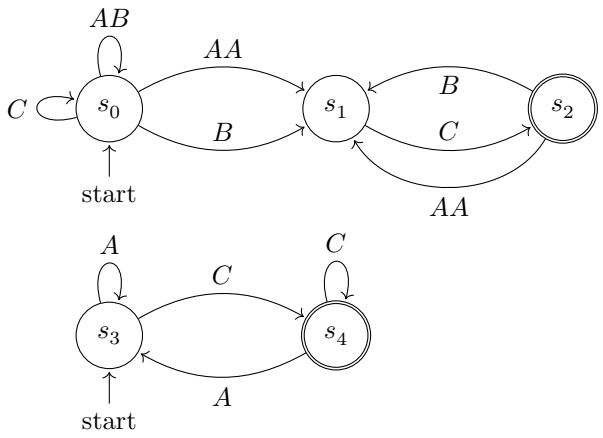


Figure 23: Exercise 4.11 NBA for the ω-regular expression $(AB + C)^*((AA + B)C)^\omega + (A^*C)^\omega$

Exercise 4.12

For $\mathcal{A}_1 : (A.(A+B+C)^*.A+C.(A+B+C)^*.C)^\omega$, for $\mathcal{A}_2 : (B+C)^*.A.(B.C)^\omega + (B + C)^*.A.(B.C)^*.B.A^\omega + (B + C)^*.B.A^\omega$

Exercise 4.13

First we apply ω operator to A_2. Intuitively, this means that from each end state we can operator as any initial state but keeping in mind that this holds only if the end state has no outgoing edges. Since q_2 has outgoing edges we need to duplicate him into a new node q_5 which takes all its outgoing edges, see Figure 24. Then to make the concatenation NBA, we need to make the end state of A_1 act as any initial state of A_2: in this case we just need to give the outgoing transitions of q_0 to p_0.

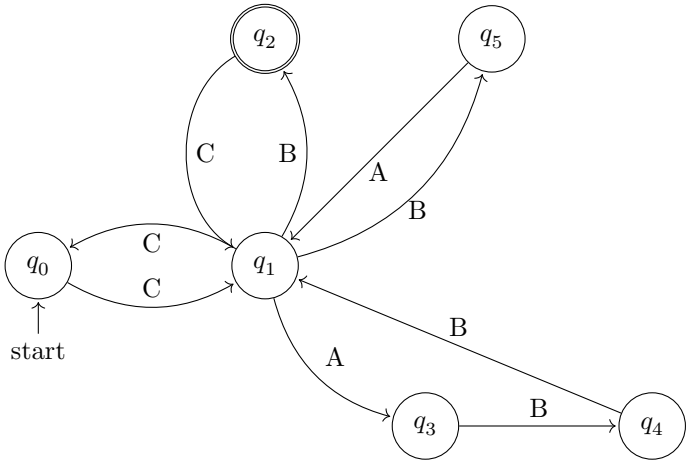


Figure 24: Exercise 4.13 NBA for A_2^ω

Exercise 4.14

This property means "infinitely often $a \wedge b$ but at least once $a \wedge \neg b$. Let $W = \{a\} + \{b\} + \{ab\} + \emptyset$. So $W^*.\{a\}.(W^*.\{a,b\})^\omega$.

Exercise 4.15

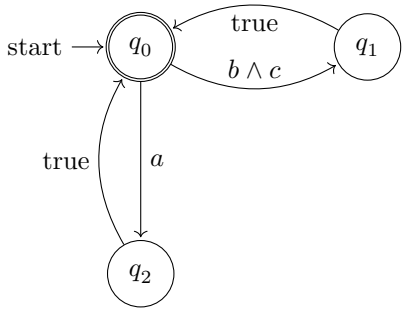


Figure 25: Exercise 4.15 NBA's

Exercise 4.16

$L_\omega(A_1) = \emptyset, L_\omega(A_2) = A^\omega$, however the powerset construction yields in both cases an automata with language accepted A^ω.

Exercise 4.17

(a) $(2^{AP}.a_1.(2^{AP\setminus\{a_1\}})^*.a_2.(2^{AP\setminus\{a_2\}})^*.a_3 \dots a_n)^\omega$ when i write 2^{AP} i mean the regular language of this finite set, i.e. the + of each of its members.

(b) Suppose there was one such automata A'. One word accepted by the language is $w = (a_1.a_2.a_3 \dots a_n)^\omega$ and so it must be accepted also by the automata A'. It can be shown that there exists a path π with $\mathit{Traces}(\pi) = w$ such that a final state A_f appears in the first n positions of the states traversed. The proof relies on the fact that we *need* to encounter an accepting state while consuming an $a_i.1 \leq i \leq n$. Say this first happens on the jth state we traverse, then it means we arrived in the $j-i+1$th state with a_1. So we can take the path $\pi_{j-i+1}\pi_{j-i} \dots \pi_j \cdot$ **as before onward** as from π_{j-i+1} we exit with a_1, from π_{j-i} with a_2 and so on. Remember that since w is accepted also $\Sigma^* \cdot w$ is accepted so we do not care about any finite prefix, we therefore always talk about acceptance of w for clarity. Let us reference the new path as $s_1, s_2, \dots s_n \dots$ and say the $k, k \leq n$-th state is the final one. By pigeon-hole principle there must be at least one state repeating in the prefix $s_1, \dots s_n$, say on indexes j and $j+i$. It is clear that once we exit the state with a_j label and the other time with one labeled $a_{j+i} \mod n$(here the modulo moves us back to 1 not 0). So we divide the problem by 3 cases:

- $i \leq k \leq i+j$: but wait...we can now create the path

$$(s_i \overset{a_i}{\to} \dots s_k \dots s_{i+j} \overset{a_{i+j}}{\to} \dots) = (s_i, \dots, s_{i+j-1})^\omega$$

  which is not in our language as a_{i+j} will not be repeated infinitely many times.

- $i + j < k$: but then I can do the path $s_1 \dots s_i \overset{a_{i+j}}{\to} \dots s_k \dots$ but the string $a_1 \dots a_{i-1}.a_{i+j} \dots a_n.(a_1 \dots a_n)^\omega$ is not in our language as it is not accepted by the automata.
- $i > k$: similar to the previous case.

Exercise 4.18

Pick any NFA such that its end states do not have outgoing edges, then the language accepted by the NBA is empty.

Exercise 4.19

TODO

Exercise 4.20

The NBA has to contain a strongly connected component S_1 where a final state of the NBA is reachable and for each state in this SCC the path gives a trace which is a string in $(AB+BA)^\omega$. It is not possible for S_1 to allow for transitions of the like $s \to^A s' \to^A$ as it would enable us to accept strings of the like "there are infinitely many BBs or AAs". Now from some state q we can consume A

and then we should be able to access S_1 (to express AB) but also to consume another A which cannot go in S_1 (because in S_1 we cannot express the language $(A+B)^\omega$). Thus, q has to have two transitions labeled A, hence it is not a DBA.

Exercise 4.21

TODO

Exercise 4.22

TODO

Exercise 4.23

Consider two generic non-blocking DBAs $A_1 = (Q, \Sigma, \delta, Q_0, F), A_2 = (Q', \Sigma, \delta', Q'_0, F')$ I claim that the DBA $A_{1\cup 2} = (Q \times Q', \Sigma, \delta^*, Q_0 \times Q'_0, F \times Q' \cup Q \cup F')$ recognizes $L_\omega(A_{1\cup 2}) = L_\omega(A_1) \cup L_\omega(A_2)$ where δ^* is defined as:

$$\delta^*(\langle q, s\rangle, a) = \langle \delta(q, a), \delta'(s, a)\rangle$$

Let $w \in L_\omega(A_1) \cup L_\omega(A_2)$, then assume $w \in L_\omega(A_1)$, the proof does not care whether $w \in L_\omega(A_2)$ and it is totally symmetric if we assumed $w \in L_\omega(A_2)$ instead. We have $w \in L_\omega(A_1) \implies \exists \pi.\pi \in Paths(A_1).Trace(\pi) = w$. Take the path π' in $A_{1\cup 2}$ such that $(a, b) = \pi'_i \xrightarrow{x} \pi'_{i+1} = (c, d)$ if and only if $a = \pi_i \wedge b = \delta'(\texttt{take\_right}(\pi'_{i-1}, x))$. Now it is evident that $\texttt{take\_left}(\pi'_i) = \pi_i$ and since π_i is an accepting path there must exists infinitely many indexes $j.\pi_j \in F$, which implies that those same indexes in π' are final as $F \times Q'$ is subset of the final states of $A_{1\cup 2}$. The observation on the size of the state space is immediate.

Exercise 4.24

The NBA is shown in Figure 26

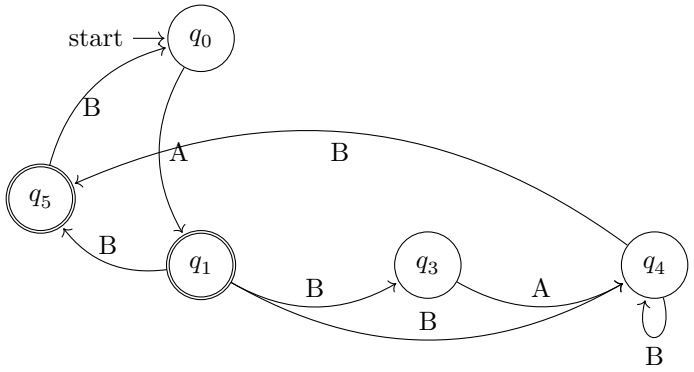


Figure 26: Exercise 4.24 NBA's

Exercise 4.25

TODO

Exercise 4.26

(a) $L = (BB + C(AA)^+C)^\omega + (BB + C(AA)^+C)^*C.(AA)^\omega$

(b) The GNBA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, \mathcal{F})$ accepts the same language as the Muller Automaton $(Q, \Sigma, \delta, Q_0, \mathcal{F}')$ where $\mathcal{F}' = \{A | \forall f \in \mathcal{F} \exists a. \in F \wedge a \in A\}$.

Exercise 4.27

(a) You can view the NBA for P_{live} in Figure 27 and the one for its complement in Figure 28.

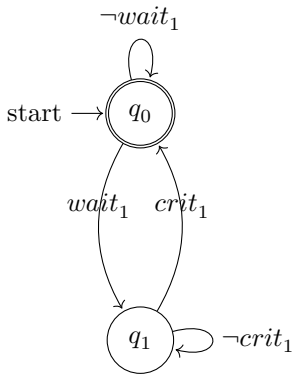


Figure 27: Exercise 4.27 NBA for P_{live}

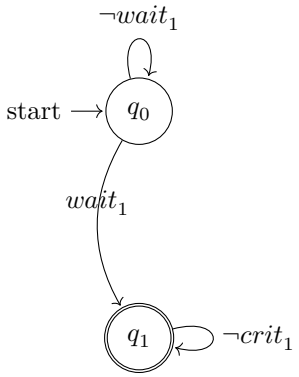


Figure 28: Exercise 4.27 NBA for the complement of P_{live}

(b) (i) Consider TS_{Sem} in Figure 38. The reachable fragment of $TS_{Sem} \otimes \hat{\mathcal{A}}$ is depicted in Figure 29. The nested DFS computes in which reachable states $\neg \bigwedge_{\forall f \in F} \neg f = q_1$. For example $\langle s_7, q_1 \rangle$. Then from this state runs a DFS and it first adds on the stack $\langle s_6, q_1 \rangle$ and checks if there is an edge $\langle s_6, q_1 \rangle \rightarrow \langle s_7, q_1 \rangle$ since it is negative, it checks $\langle s_1, q_1 \rangle$ which has an outgoing edge to $\langle s_7, q_1 \rangle$ and thus it returns $\langle s_7, q_1 \rangle, \langle s_6, q_1 \rangle, \langle s_1, q_1 \rangle, \langle s_7, q_1 \rangle$.

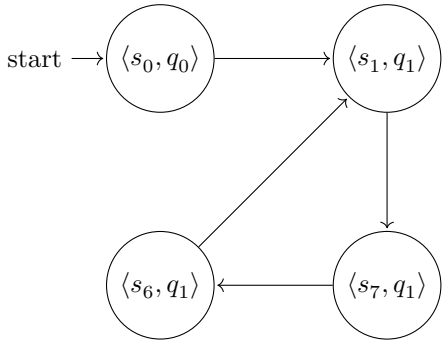


Figure 29: Exercise 4.27

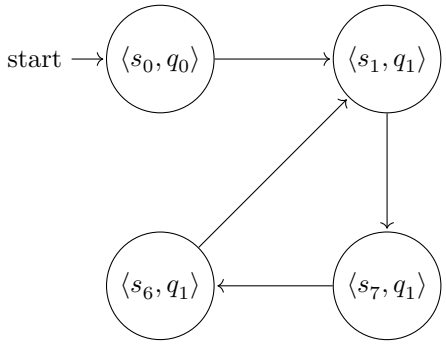


Figure 30: Exercise 4.27

4 Linear Time Logic

Exercise 5.1

Let $S = \{s_1, s_2, s_3, s_4\}$.

(1) ○a is satisfied by $S \setminus s_4$

(2) ○○○resolves in computing all paths of length 3 starting from a certain state and checking whether they end up in $S \setminus s_4$, clearly this happens

for any path starting from any state, so the whole state space satisfies the formula.

(3) This formula is satisfied by no state as eventually we enter in a state in $\{s_1, s_2\}$

(4) It is satisfied by S

(5) It is satisfied by S

(6) Any state satisfying $\Diamond b$ also satisfies $\Diamond(aUb)$ and all the states satisfy $\Diamond b$

Exercise 5.2

- $\neg TS \vDash \phi_1$: there is no state which satisfies $\Box c$.
- $TS \vDash \phi_2$: the only way not to satisfy this formula would be to loop through states which do not have c in their label set. The only two states which do not have c are s_1 (which is not reachable apart from when the path starts from it) and s_4 which, however, has no outgoing edges to $\{s_1, s_4\}$
- $\neg TS \vDash \phi_3$: $\pi \stackrel{\text{def}}{=} s_1 \to s_4 \to s_5 \to s_4 \in Paths(TS)$ but $\neg\pi \vDash \phi_3$.
- $\neg TS \vDash \phi_4$ because $c \notin L(s_2)$
- $TS \vDash \phi_5$ because all paths in the SCC $S' = S \setminus s_1$ satisfy $\Box b \vee c$ and for a path starting from s_1

Exercise 5.5

Let E_i denote the atomic proposition "the elevator is at the i-th floor", O_i denote the atomic proposition "the i-th floor's door is open", R_i denote "the elevator is requested at the i-th floor".

(a) $(\bigvee_{i=0})^{N-1} E_i \wedge O_i$

(b) $\Box \bigwedge_{i=0}^{N-1} (\Box\neg(E_i \wedge O_i) \wedge \neg R_i)$

(c) $\Box\Diamond E_0$

(d) $\Box((\neg \bigwedge_{i=0}^{N-2} (\neg E_i)\ \mathcal{U}\ E_{N-1}) \wedge \neg R_{N-1})$

Exercise 5.6

Consider $\phi = a$ and $\psi = b$.

(a) They are not equivalent. $\{a\}\emptyset\{b\}^\omega$ satisfies the formula on the left but not the one on the right.

(b) They are equivalent. If $S \vDash \Diamond\Box\phi \rightarrow \Box\Diamond\psi$ then either ϕ is not eventually always true or it is and this implies ψ is always eventually true. Let's put ourselves in the first case, then consider any S_i.

- If $S_i \vDash \phi$ then take the first S_k such that $S_k \vDash \neg\phi$, thing we can do because $\neg S_i \vDash \Diamond\Box\phi$ and moreover $\neg S_i \vDash \Box\phi$. Then $\forall i < j < k . S_j \vDash \phi$ because k was chosen minimal, and thus $S_i \vDash \phi\ \mathcal{U}\ (\psi \vee \neg\phi)$.
- If $\neg S_i \vDash \phi$ then we are immediately done because we chose as prefix where ϕ holds the empty one.

Now, consider the case where $S \vDash \Diamond\Box\phi$, that is, there exists minimal k such that $S_k \vDash \Box\phi$. Then it must also be true that $S \vDash \Box\Diamond\psi$. Consider any S_i:

- If $i < k$ we use the same reasoning as above as we are sure $\exists t . i \leq t < k$ such $S_t \vDash \neg\phi$ as, if that was not the case, $S_t \vDash \Box\phi$ but k was chosen to be minimal index such that $S_k \vDash \Box\phi$ and $t < k$.
- If $i \geq k$ the reasoning above does not apply as there is no next state where $\neg\phi$ holds. But we know $S_i \vDash \Diamond\psi$ so there must exist $t \geq i$ such that $S_t \vDash \psi$ so $S_t \vDash \psi \vee \neg\phi$ and since $\forall i \leq j < t S_j \vDash \phi$ we have $S_i \vDash \phi\ \mathcal{U}\ (\psi \vee \neg\phi)$.

Now lets prove the opposite direction, namely that $(S \vDash \Box(\phi\ \mathcal{U}\ (\neg\phi \vee \psi))) \implies (S \vDash \Diamond\Box\phi \rightarrow \Box\Diamond\psi)$. Let's work by contradiction, so we assume $S \vDash \Diamond\Box\phi \wedge \neg S \vDash \Box\Diamond\psi$. By the latter part of the conjugation we know there exists a S_j such that $\neg S_j \vDash \Diamond\psi$, i.e. $S_j \vDash \Box\neg\psi$. By the first part of the conjugation we know there exists k such that $S_k \vDash \Box\phi$ so let $p = max(k, j)$. Then, $S_p \vDash \Box(\phi \wedge \neg\psi)$ so $\neg S_p \vDash \Diamond(\neg\phi) \vee \Diamond\psi$ thus it cannot satisfy $\phi\ \mathcal{U}\ (\neg\phi \vee \psi)$ but we assumed $S \vDash \Box\phi\ \mathcal{U}\ (\neg\phi \vee \psi)$, contradiction.

(c) $\Box$ is idempotent, so $\Box\Box(\phi \vee \neg\psi) \equiv \Box(\phi \vee \neg\psi)$. The rest is DeMorgan, so they are equivalent.

(d) $\{a\}\{b\}\emptyset^\omega \vDash \Diamond\phi \wedge \Diamond\psi$ but $\neg(\{a\}\{b\}\emptyset^\omega \vDash \Diamond(\phi \wedge \psi))$

(e) They are equivalent. In particular any structure satisfying $\Box\phi$ also satisfies $\bigcirc\Diamond\phi$.

(f) $\{a\}\emptyset^\omega \vDash \Diamond\phi$ but $\neg\{a\}\emptyset^\omega \vDash \Diamond\phi \wedge \bigcirc\Box\phi$ as $\neg\{a\}\emptyset^\omega \vDash \bigcirc\Box\phi$ because $\neg\emptyset^\omega \vDash \Box\phi$.

(g) They are not equivalent. $\{a\}\emptyset^\omega \vDash (\Box\Diamond\phi) \rightarrow (\Box\Diamond\psi)$ vacuously but $\neg(\phi \rightarrow \Diamond\psi)$ as it requires at least an index k such that $S_k \vDash \psi$ but clearly no set contains b.

(h) They are equivalent, this is a basic duality notion.

(i) If $\phi_1 \not\equiv \phi_2$ this is easily not always true. If $\phi_1 \equiv \phi_2$ then they are equivalent: Suppose $k > 0$ and $S_k \vDash \phi_1$ then take $k' = k - 1$ and $S_{k'} \vDash \bigcirc\phi_1 \iff S_{k'} \vDash \bigcirc\phi_2$ that is $S \vDash \Diamond\bigcirc\phi_2$. Where we used equivalence in the $\iff$.

(j) They are equivalent. If $S \vDash (\Diamond\Box\phi_1) \wedge (\Diamond\Box\phi_2)$ then there are two indexes i, j such that $S_i \vDash \Box\phi_1$ and $S_j \vDash \Box\phi_2$. So $S_{\max(i,j)} \vDash (\Box(\phi_1 \wedge \phi_2))$. For the other direction by the same reasoning you obtain $i = j$.

(k) They are equivalent.

- $\Longrightarrow$: assume $S \vDash (\phi_1 \ \mathcal{U} \ \phi_2) \ \mathcal{U} \ \phi_2$ then $\exists k.S_k \vDash \phi_2 \wedge \forall i < kS_i \vDash (\phi_1 \ \mathcal{U} \ \phi_2)$, where such k is minimal. Then $S_i \vDash \neg\phi_2$ which implies $S_i \vDash \phi_1$, by until semantics. Thus $\forall i < k.S_i \vDash \phi_1$.
- $\Longleftarrow$: assume $S \vDash (\phi_1 \ \mathcal{U} \ \phi_2)$ then there $\exists k.S_k \vDash \phi_2 \wedge \forall i < k.S_i \vDash \phi_1 \wedge \neg\phi_2$. We need to prove $\forall i < k.S_i \vDash \phi_1 \ \mathcal{U} \ \phi_2$ i.e. that there exists $t \geq i$ such that $S_t \vDash \phi_2$ and $\forall i \leq j < t.S_j \vDash \phi_1$. Take $t = k$.

4.1 Exercise 5.7

(a) $\phi \ \mathrm{N} \ \psi \equiv \neg\psi \ \mathcal{W} \ (\phi \wedge \psi)$

(b) $\phi \ \mathrm{W} \ \psi \equiv \phi \ \mathcal{W} \ \neg\psi$

(c) $\phi \ \mathrm{B} \ \psi \equiv (\Diamond\psi \rightarrow (\phi \ \mathcal{U} \ \psi)$ if the *before* is to be intended as non-strictly before otherwise if ϕ needs to be satisfied in a moment strictly before I propose $\Diamond\psi \rightarrow (\phi \ \mathcal{U} \ (\psi \wedge \neg\phi))$

5 Computation Tree Logic

Exercise 6.8

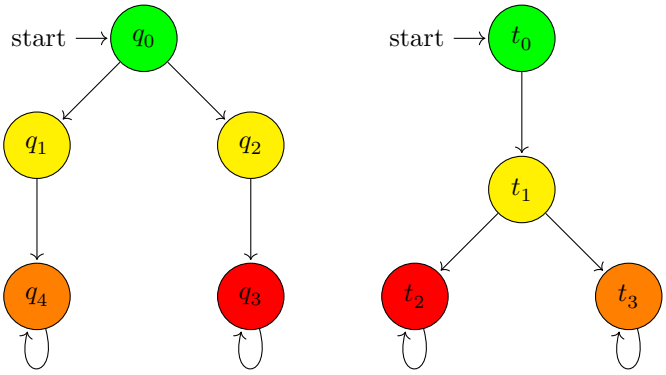


Figure 31: Exercise 6.8

The two transition systems are trace equivalent but the one on the left does not satisfy the CTL formula $\forall\bigcirc\exists\bigcirc \mathit{red}$

Exercise 6.13

The implication is false, as removing paths in TS' may invalidate state formulas previously true in TS. Suppose $\Phi = \mathit{green}$ and $\Psi = \exists\bigcirc \mathit{red}$. Then removing the transition from q_1 to q_2 makes q_1 not satisfy anymore Ψ and as a consequence no reachable state satisfies $\exists\bigcirc \mathit{red}$.

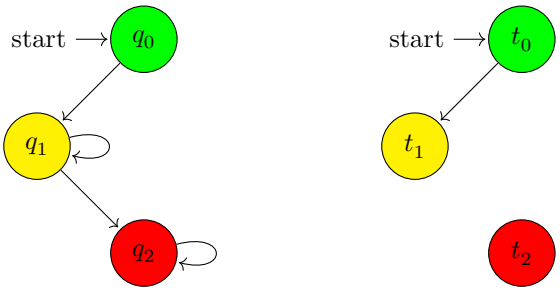


Figure 32: Exercise 6.13; implication counterexample

The other direction is true: suppose there is a path π that leads in *TS'* to a state γ where Ψ holds, then, all the states preceding γ in π satisfy $\Phi \wedge \neg\Psi$ because they have outgoing transitions (among which the ones we use to arrive at γ). This implies π satisfies $\Phi U \Psi$ and since π is a path of *TS'* it also a path in *TS*.

Exercise 6.15

(a) The LTL formula $\Phi_2 = \Diamond(a \wedge \bigcirc a)$ is not equivalent to Φ_1. Consider the transition system in 33, which satisfies Φ_1 and not Φ_2 (because the path $q_0(q_2)^\omega$ does not satisfy it).

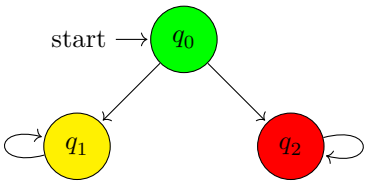


Figure 33: Exercise 6.15 a)

(b) We want to prove that $\Phi_2 \stackrel{\text{def}}{=} \forall\Diamond\exists\bigcirc\forall\Diamond\neg a$ has no LTL equivalent formula. We use the fact that if $Traces(TS) \subseteq Traces(TS')$ and TS' satisfies an LTL formula ϕ, also TS satisfies it ϕ. Suppose that an LTL counterpart existed, call it Ψ_2. Take the transition system on the left in 34, if we rename a as *green* then it satisfies Φ_2 as any of state in the two only possible paths $(q_0)^*(q_1)^\omega, (q_0)^\omega$ satisfy $\exists\bigcirc\forall\Diamond\neg a$: this path is exactly q_1^ω. Thus it also satisfies Ψ_2 by equivalence. However, if we take the right transition system, we see that its traces are only $green^\omega$ (the one obtained by the only admissible path $(q_0)^\omega$) and thus it does not satisfy Φ_2, contradiction.

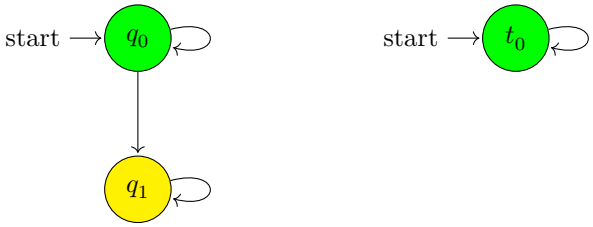


Figure 34: Exercise 6.15 b)

Exercise 6.17

See the pseudo code here 2

Algorithm 2 Computing $Sat(\forall(\Phi \,\mathsf{U}\, \Psi))$

```
1: procedure SatComputing(TS = (S, Act, →, I, AP, L), Φ, Ψ)
2:     E ← Sat(Φ)
3:     Q ← Sat(Ψ)
4:     V ← Q
5:     M ← HashMap[S, Int]
6:     for all x ∈ S do
7:         M[x] ← |succ(x)|
8:     end for
9:     while Q ≠ ∅ do
10:        q ← Extract(Q)
11:        for all x ∈ pred(q) do
12:            if x ∉ Q ∧ x ∈ E then
13:                M[x] ← M[x] − 1
14:                if M[x] = 0 then
15:                    Add(Q, x)
16:                    V ← V ∪ {x}
17:                end if
18:            end if
19:        end for
20:    end while
21:    return V
22: end procedure
```

Exercise 6.18

(a) We need to prove that for any state $x \in S$, $x \in Sat(\exists(\Phi W \Psi)) \iff x \in T^*$ where T^* is the greatest subset of S such that it satisfies the recurrence:

$$T \subseteq Sat(\Psi) \cup \{s \in Sat(\Phi) | post(s) \cap T \neq \emptyset\}$$

One can find T^* by finding the greatest fix point of a function $\mathcal{F} : \mathcal{P}(S) \mapsto \mathcal{P}(S)$ defined as $\mathcal{F}(x) = Sat(\Psi) \cup \{s \in S | post(s) \cap x \neq \emptyset\}$. The existence of a fixed point is guarantee because $\mathcal{P}(S)$ is a complete lattice and $\mathcal{F}$ is monotone on the order $\preceq$ defined as $x \preceq y \iff y \subseteq x$, using Knaster-Tarski.

- $\Longrightarrow$: assume x is in the satisfying set and suppose $x \notin T^*$. By hypothesis we know there exists a path $\pi \in Paths(x)$ such that $\pi \vDash \Phi W \Psi$, this means that either Ψ is true for a π_k and $\forall i < k . \pi \vDash \Phi \wedge \neg\Psi$ or $\forall j . \pi_j \vDash \neg\Psi \wedge \Phi$.

(a) In the case such a π_k exists it implies that $\pi_k \in T^*$ since $\pi_k \in Sat(\Psi)$ and T^* is maximal. Then by backward induction we can prove that all states $\pi_1, ..., \pi_k$ are part of T^*, where the base case is $i = k$ and the inductive case is $i-1$ assuming $\{\pi_i, ..., \pi_k\} \subseteq T^*$ and by using the right hand side of the equation, since π_{i-1} satisfies Φ by hypothesis, and therefore maximality forces $\pi_{i-1} \in T^*$.

(b) In the case there is no such state, all the states in π satisfy Φ. We need to show that adding all states in π, and thus adding $\pi_1 = x$, does not violate the recurrence, and hence all those states must already be present in T^* by maximality. Suppose $T' = T^* \cup \{\pi_1, ...\}$. Take any state π_t in π, since we added π_{t+1} and we assumed $\pi_t \in Sat(\Phi)$ by the right hand side condition $\pi_t \in Sat(\Psi) \cup \{s \in Sat(\Phi) | post(s) \cap T' \neq \emptyset\}$ and thus T' satisfies the condition. Since we can repeat this reasoning for all states traversed in π and T^* was the maximal set that satisfied the recurrence it implies $T' \subseteq T^*$ i.e. $x \in T^*$.

- $\Longleftarrow$: We will prove the contra-positive: $x \models \forall(\neg\Psi U \neg\Psi \wedge \neg\Phi) \implies x \notin T^*$ and we will do it by induction on the iteration of the operator, which we define as $\mathcal{F}^0(S) = S, \mathcal{F}^{i+1}(S) = \mathcal{F}(\mathcal{F}^i(S))$. Suppose k is the iteration number of the fixed point, i.e. $\mathcal{F}^{k+1}(S) = \mathcal{F}^k(S)$. Take $Paths(x) = \{\pi_1, ... \}$, by our assumption, in every such path it exists a minimal index such that ${\pi_i}^j \models \neg\Psi \wedge \neg\Phi$, call the set of those indexes "boundaries". Take the maximum one and call it ω. Then we want to prove that $x \notin \mathcal{F}^\omega(S)$. Notice that after this proof we know $x \notin T^*$ as our operator is monotone. We want to use induction but to do this we need to define a partial order: let X be the set of states $\{(\pi_i^j, i, j) | j \leq$ boundary index for ${\pi_i}\}$ and define a total order $(x, y, z) \preceq (x', y', z') \iff z < z' \vee (z = z' \wedge y < y')$. That is each state of each path is labeled by an unique integer (second component) and its position in its own path(third component). Now by induction on this partial order (where the base case is going to be the maximal element) we prove that each of those states is eventually removed. We use naturals to denote the elements of the partial order. Let Z be the maximal element.
  - Base case: $i = Z$. The maximal element of this total order is a state which has the third component maxed, i.e. it is the boundary index of some path. Thus it satisfies $\neg\Psi \wedge \neg\Phi$ and by the same reasoning it is eliminated after the first iteration of $\mathcal{F}$.
  - Inductive case: $i - 1$. Suppose all states represented by labels greater than $i - 1 = (x, y, z)$ have been eliminated. Then take any path such that is equal to ${\pi_y}$ up to the z state, then any path that extend from z have been eliminated by inductive hypothesis. Thus also x is removed.

The other point is similar.

Exercise 6.20

For $\exists(\Phi R \Psi) = \neg\forall((\neg\Phi)U(\neg\Psi))$:

$$\exists\Phi R\Psi \equiv \exists((\neg\Phi \wedge \Psi)U(\Phi \wedge \Psi)) \vee \exists(\Box(\Psi \wedge \neg\Phi))$$

so an expansion law for this operator is simply:

$$\exists\Phi R\Psi \equiv ((\neg\Phi \wedge \Psi) \wedge \exists\bigcirc\exists\Phi R\Psi) \vee (\Phi \wedge \Psi) \equiv \Psi \wedge (\Phi \vee (\exists\bigcirc\exists\Phi R\Psi)).$$

for $\forall(\Phi R\Psi) \equiv \Psi \wedge (\Phi \vee \forall\bigcirc\forall\Phi R\Psi)$.

Exercise 6.21

(a) $Sat(\Phi_1) = \{s_2, s_3\}$. To compute $Sat(\Psi_1)$ one needs to take only states satisfying $a \wedge \neg b = \{s_5\}$ and then take the preceding nodes satisfying b, i.e. $Sat(\Psi_1) = \{s_0, s_2\}$.

(b) To compute $Sat_{sfair}(\exists\Box true)$ one takes all the states in the transition system (the ones satisfying *true*), and then compute the SCCs: $SC_1 = \{s_0, s_1\}, SC_2 = \{s_2, s_5\}, SC_3 = \{s_3, s_4\}$. Then we check those against s_{fair}:

- Since $SC_1 = \{s_0, s_1\} \cap Sat(\Phi_1) = \emptyset$ we find that both of those states are in $Sat_{sfair}(\exists\Box true)$.
- Since $SC_2 = \{s_2, s_5\} \cap Sat(\Phi_1) = \{s_2\}$ and $SC_2 \cap Sat(\Psi_1) \neq \emptyset$ we have $SC_2 \subset Sat_{sfair}(\exists\Box true)$.
- Since $SC_3 = \{s_3, s_4\} \cap Sat(\Phi_1) \neq \emptyset$ and $SC_3 \cap Sat(\Psi_1) = \emptyset$ we need to remove s_3, however, $\{s_4\}$ is not anymore a connected component.

So in conclusion $Sat_{sfair}(\exists\Box true) = \{s_0, s_1, s_2, s_5\}$

(c) First we put the formula in **ENF**: $\Phi \equiv \exists\Diamond\exists\Box\neg a$. So we compute first $Sat(\neg a) = \{s_1, s_2, s_3, s_4\}$ and then remove all but $Sat(\neg a)$ states and then search for SCCs: SC_3 is the only SCC such that all of its states are present in $Sat(\neg a)$. However, no state in SC_3 has fair paths, so $Sat_{sfair}(\exists\forall\neg a) = \emptyset$. By the same reasoning no state satisfies $\exists\Diamond\exists\Box\neg a$ so we conclude $Sat_{sfair}(\Phi) = \emptyset$.

Exercise 6.22

(a) This is false: the right part is strictly stronger as it requires all paths exiting the start node to be fair. Assume the unconditional fair assumption $\Box\exists$*yellow*. A counter-example is shown in Figure 35 where the given transition systems satisfies that for any fair path *green U yellow* but it does not satisfy, for the path $q_0, (q_2)^\omega$ the path formula *green U (yellow* $\wedge a_{fair}$*)*.

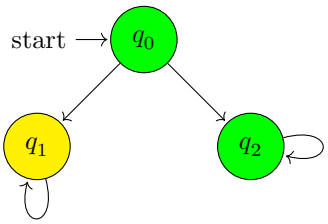


Figure 35: Exercise 6.15 b)

(b) Consider the fairness assumption $\Box\Diamond yellow$ and $a = green, b = yellow$. Consider the transition system shown in Figure 36: it satisfies $\exists(aW(b \wedge a_{fair}))$ since a is forever true and $b \wedge a_{fair}$ stays forever false (as a_{fair} is false). However, this transition system does not have any fair path satisfying $\exists(aWb)$.

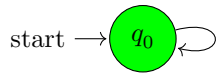


Figure 36: Exercise 6.15 b)

(c) We show the two direction separately:

- $\Longrightarrow$: we want to prove that $\forall \pi \in Paths(s)$ we have $\pi \vDash (a\mathtt{W}(a_{fair} \Longrightarrow b))$, meaning it must be either:
  - If π is a fair path then by assumption we know either:
    - there is a index k' such that $\pi_{k'} \vDash b$ and $\forall j < k'.\pi_j \vDash a$ then take $k = k'$, then $\pi_k \vDash a_{fair} \Longrightarrow b$ and $\forall i < k$ we have $\pi_i \vDash a$.
    - for all states it holds $\neg b \wedge a$ so $a_{fair} \Longrightarrow b$ will always be false (as a_{fair} is always true on a fair path) but this is accepted by the weak until semantics.
  - If π is not a fair then either:
    - There is a finite prefix $\pi_1, ..., \pi_k$ where a_{fair} holds, and, thus $\pi_1, ..., \pi_k$ is a prefix of some fair path and thus either b has been satisfied in $\pi_1, ..., \pi_k$ (and then the reasoning above applies) or it will be satisfied by following some other fair path (or it will never be satisfied). In both latter cases, we have $\pi_1, ..., \pi_k \vDash a$ and thus, since $\pi_{k+1} \vDash \neg a_{fair}$ we have $\pi_{k+1} \vDash (a_{fair} \Longrightarrow b)$.
    - If $\forall \pi_i.\pi_i \vDash a_{fair}$ but it is an unfair path, then each π_i is part of some fair path with prefix $\pi_1, \ldots, \pi_i$. Which means either $a\mathtt{W}b$ is satisfied by some prefix $\pi_1, \ldots, \pi_i$ and then we fall in the first case of the preceding point, or $\forall \pi_i.\pi_i \vDash a \wedge \neg b$. Which is ok for the formula $a\mathtt{W}(a_{fair} \Longrightarrow b)$.

- $\Longleftarrow$: take any fair path π then we want to prove $\pi \vDash a\mathtt{W}b$ assuming $\pi \vDash a\mathtt{W}(a_{fair} \implies b)$. By our assumption we know two cases can occur:
  - $\forall \pi_i.\pi_i \vDash a \wedge \pi_i \vDash (\neg b \wedge a_{fair})$. Which also means $\forall \pi_i.\pi_i \vDash a \wedge \neg b$ which implies $\pi \vDash a\mathtt{W}b$.
  - $\exists k.\pi_k \vDash (\neg a_{fair} \vee b)$. In this case if $\pi_k \vDash b$ it easy to finish the proof. But if $\pi_k \vDash \neg b$?. Then we just know $\pi_k \vDash \neg a_{fair}$, but remember π is a fair path, i.e. $\forall \pi_i.\pi_i \vDash a_{fair}$. So it can only be that $\pi_k \vDash b$.

5.1 Appendix

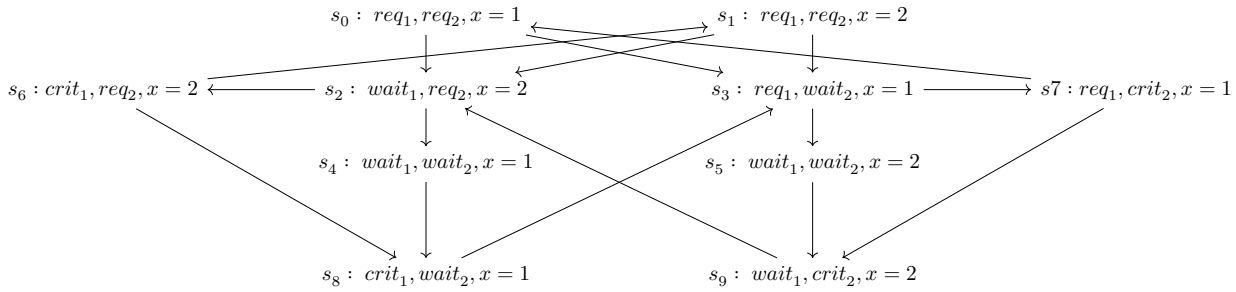


Figure 37: Transition Graph for Peterson Algorithm, TS_{pet}

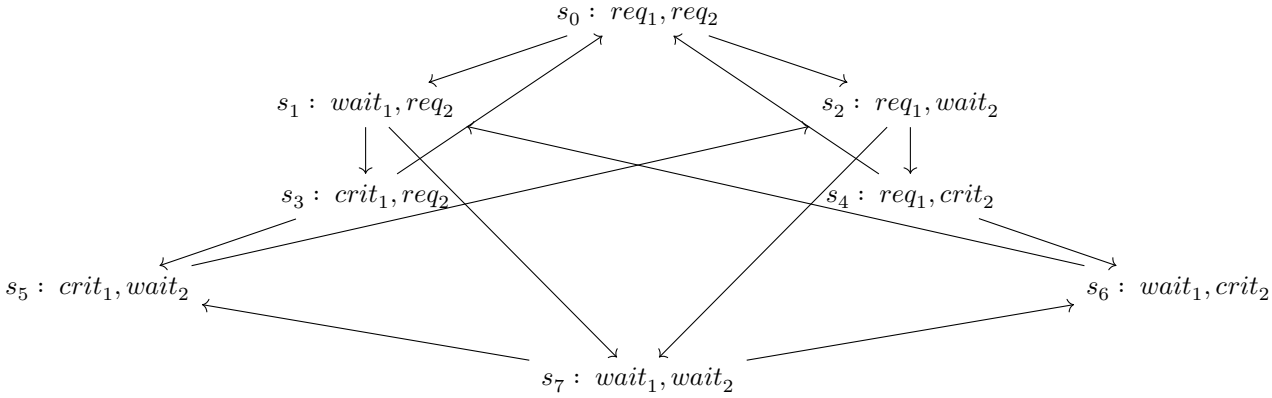


Figure 38: Transition Graph for Semaphore Algorithm, TS_{sem}