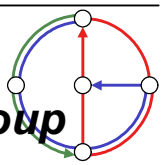
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing Group

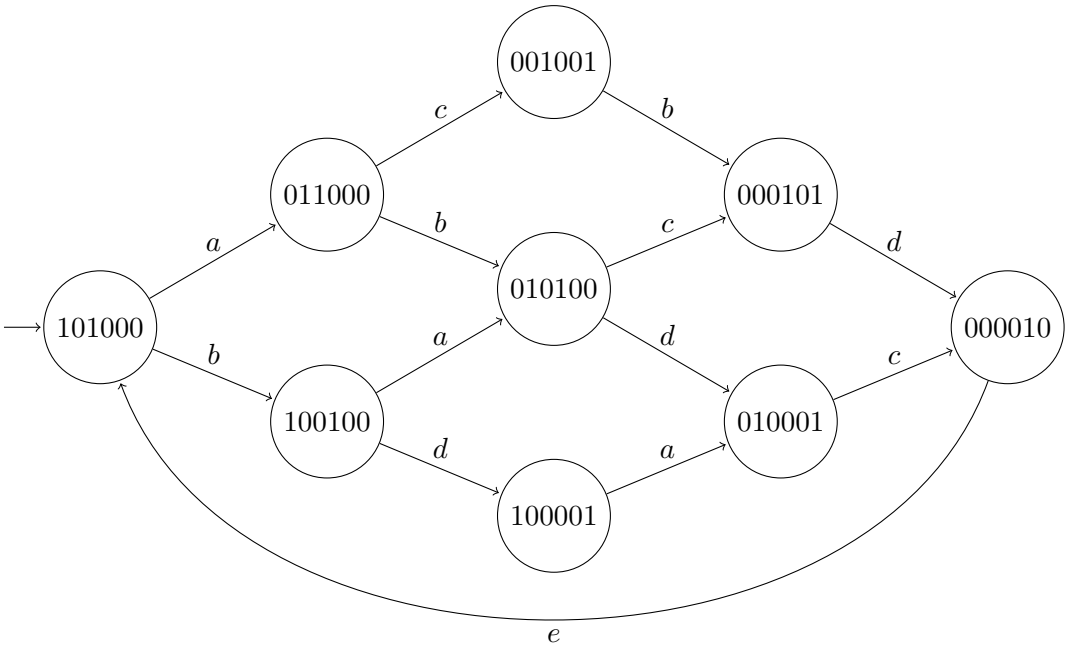


HS 2008 Dr. Kai Lampka / Raphael Eidenbenz / Roland Flury

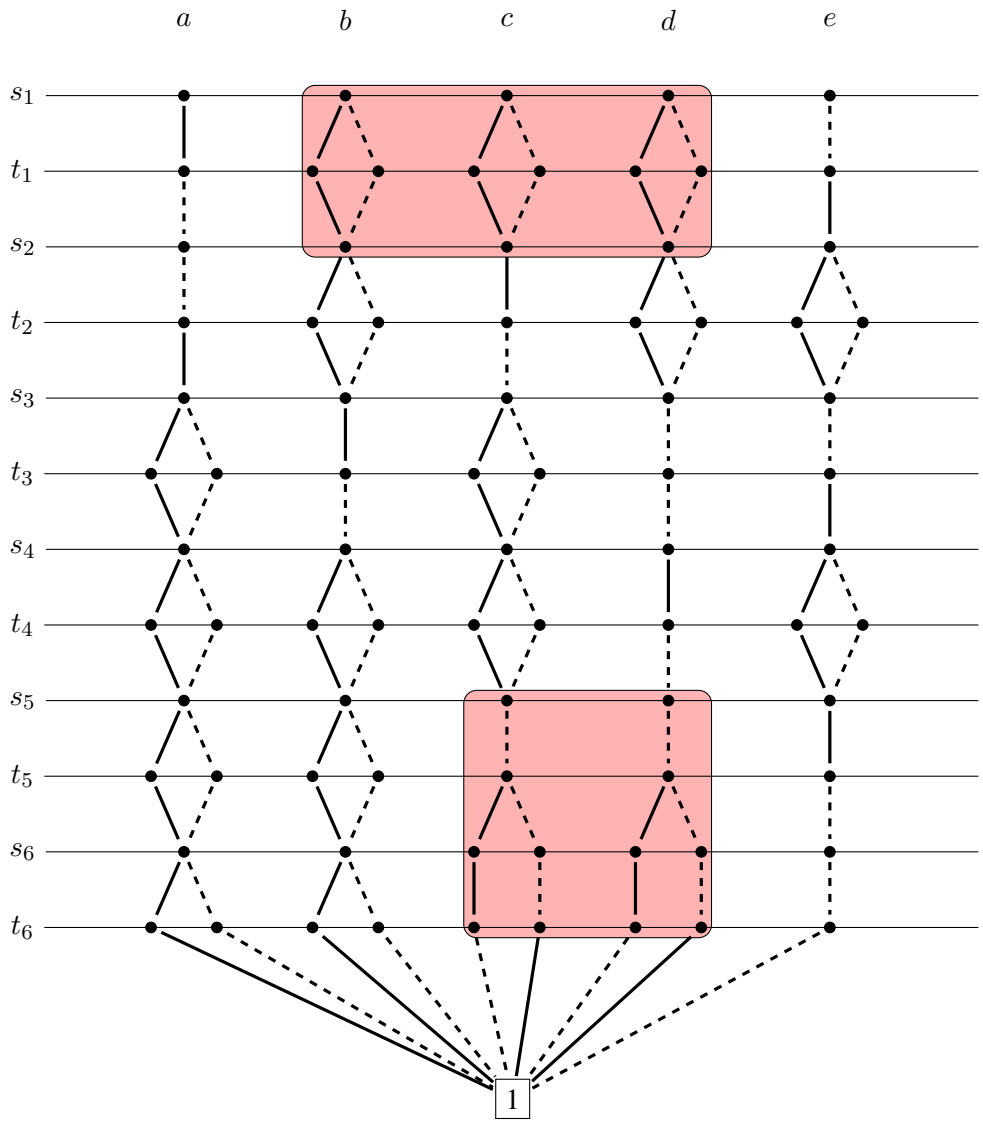
DES –Musterlösung zur Wiederholungsübung:
Petri Netze (PN), BDDs, Model Checking & Timed Automata

1 Symbolische Kodierung von LTS

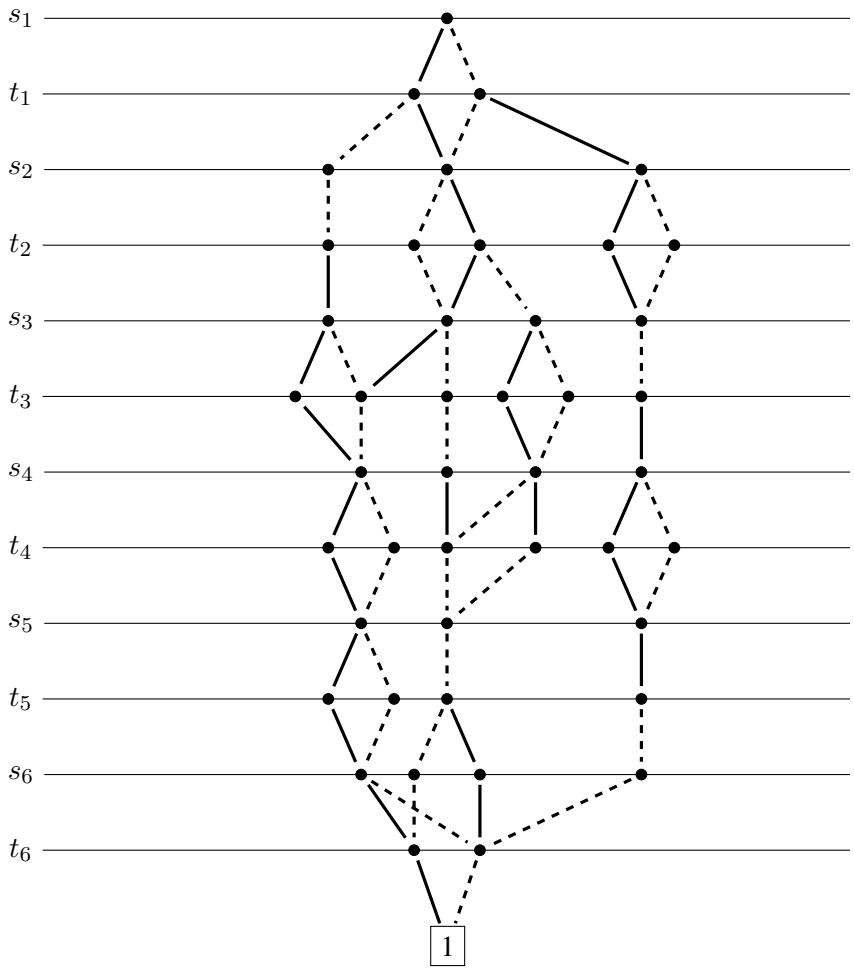
a) Folgendes LTS repräsentiert die Funktion f_{M}:



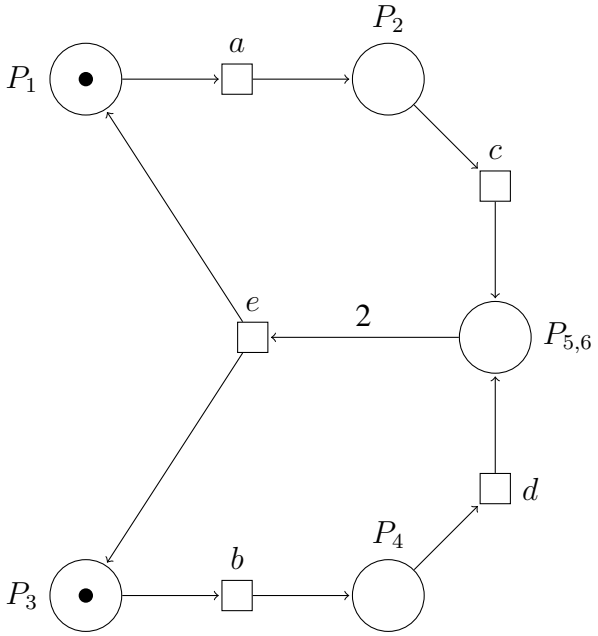
b) Wir zeichnen zuerst die verschiedenen Transitionsvorschriften a–e einzeln als BDD:



In einem zweiten Schritt minimieren wir unser BDD. Dazu können wir isomorphe Subgraphen von “unten” und von “oben” herkommend iterativ vereinigen. So verhindern wir, dass beim minimieren neue Pfade entstehen. Beispielsweise beginnen die Transitionen b,c,d gleich oder die Transitionen c,d hören gleich auf (Vgl. rote Kästen). Das fertig minimierte BDD sieht dann so aus:



c) Wir erhalten folgendes Petri-Netz.



2 Model Checking: Zustandsinvarianten

a) Man führt eine Erreichbarkeitsanalyse aus und überprüft jeden Zustand daraufhin, ob er die Zustandsinvariante erfüllt. (Vgl. Skript 3/100–3/114)

b) `S:` = set of reachable system states;
`P:` = invariant to be checked;

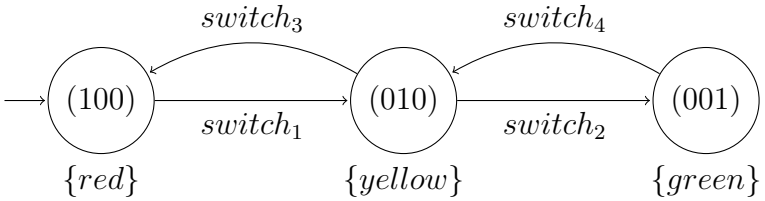
```
foreach(s in S){
 if (check(s,P)== false){
   return false;
 }
}
```

Dabei nehmen wir an, die Funktion `check(s,P)` gebe an, ob Invariante `P` in Zustand `s` erfüllt ist oder nicht.

c) Einfachste Safety-Properties wie bspw. Deadlock-Freiheit oder das Nicht-Erreichen eines unerwünschten Zustands.

3 Model Checking: Safety Properties

a) Das LTS zum gegebenen Petri Netz sieht wie folgt aus:

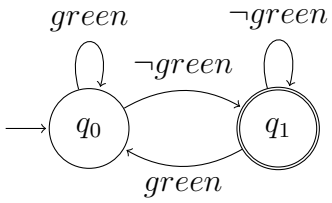


Die Eigenschaft ist erfüllt, da man Zustände mit dem Label red nur von Zuständen aus erreichen kann, die das Label $yellow$ tragen.

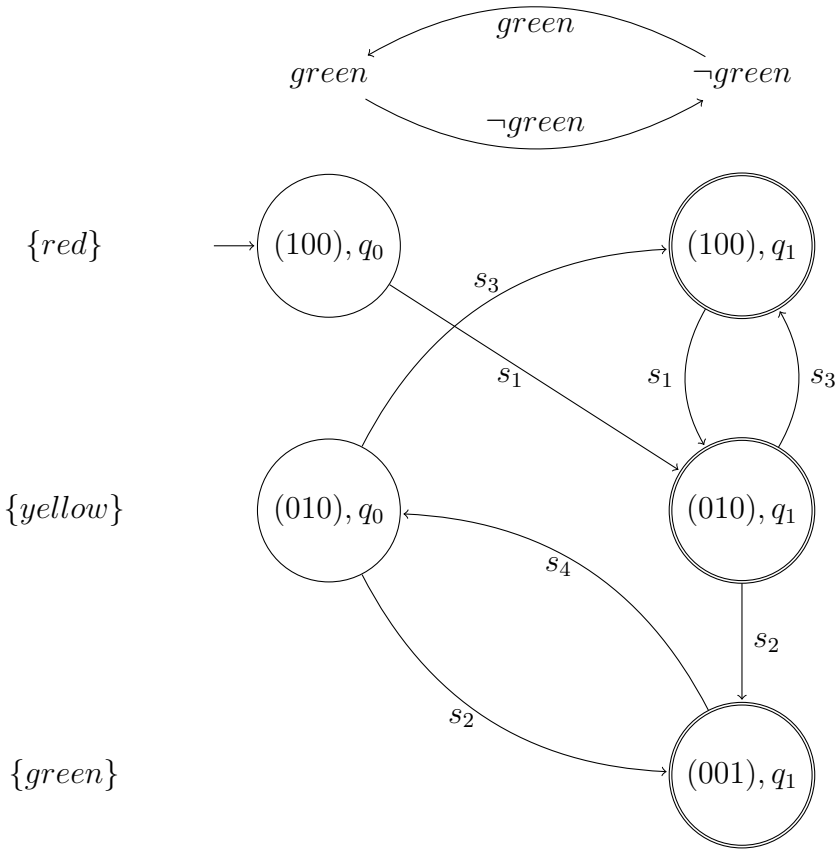
b) Hierbei handelt es sich um ein Safety Property, welches immer erfüllt sein muss.

c) Um die Eigenschaft $\Diamond\Box green$ zu überprüfen, konstruieren wir einen NBA für die Negation der Eigenschaft, $\neg(\Diamond\Box green) = \Box\Diamond\neg green$, und checken das Kreuzprodukt des NBAs und des LTS' auf Zyklen mit akzeptierenden Zuständen. Wenn wir solche Zyklen finden, heisst das, dass die negierte Eigenschaft zumindest manchmal erfüllt ist und somit die ursprüngliche Eigenschaft—welche immer gelten sollte—nicht erfüllt ist. Finden wir keine Zyklen mit akzeptierenden Zuständen, wissen wir, dass die Eigenschaft gilt. Für eine genauere Beschreibung dieses Vorgehens vergleichen Sie bitte die MuLö zur Übungsserie 8.

NBA of $\Box\Diamond\neg green$:



Das Kreuzprodukt sieht folgendermassen aus:



Wir stellen fest, dass es sogar mehrere Zyklen gibt, die akzeptierende Zustände durchlaufen. Folglich ist die LTL-Formel $\Diamond\Box green$ nicht korrekt. Dies ist auch intuitiv einleuchtend, da das System immer wieder auf $yellow$ zurück schalten kann, wenn es einmal $green$ ist. Um zu erzwingen, dass unser System garantiert irgendwann auf $green$ schaltet und schliesslich auch in $green$ bleibt, dürfen wir keine Loops erlauben bevor das endgültige $green$ erreicht ist. In diesem "Endstadium" befindet sich das System entweder in einem Loop, in dem $green$ immer gilt, oder in einem einzelnen Zustand, in dem $green$ gilt. Ein Petri-Netz, welches $\Diamond\Box green$ erfüllen würde, wäre bspw. folgendes: