Lösungsvorschlag zu Übungsblatt 4

Mit Dank an die Hilfe durch Herrn Georg Schneider.

Aufgabe 1

In einer Fabrik mit Tag- und Nachtschicht wird fortwährend ein Produkt durch Zusammenbauen zweier Teile A und B hergestellt. Die beiden Teile werden unabhängig voneinander produziert aber gleichzeitig bereit gestellt um anschließend zusammengebaut werden zu können. Während des Zusammenbauens können bereits die nächsten beiden Teile produziert werden.

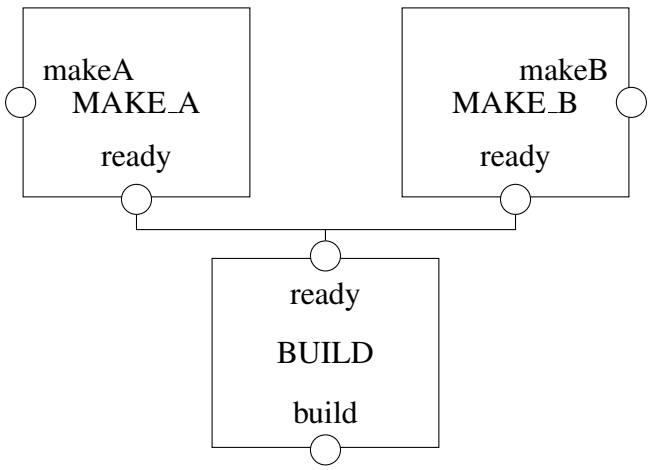
Modellieren Sie den Fertigungsprozess in FSP und geben Sie ein Strukturdiagrammmm für den Prozess an. Geben Sie auch das zugehörige LTS des Prozesses an.

Lösung

FSP-Prozesse

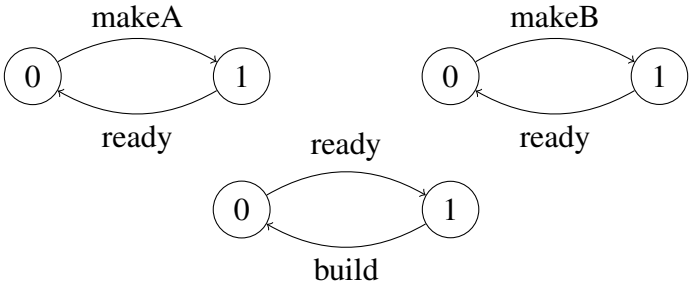
```
MAKE_A = (makeA -> ready -> MAKE_A).
MAKE_B = (makeB -> ready -> MAKE_B).
BUILD = (ready -> build -> BUILD).

||FACTORY = (MAKE_A||MAKE_B||BUILD).
```

Strukturdiagrammm

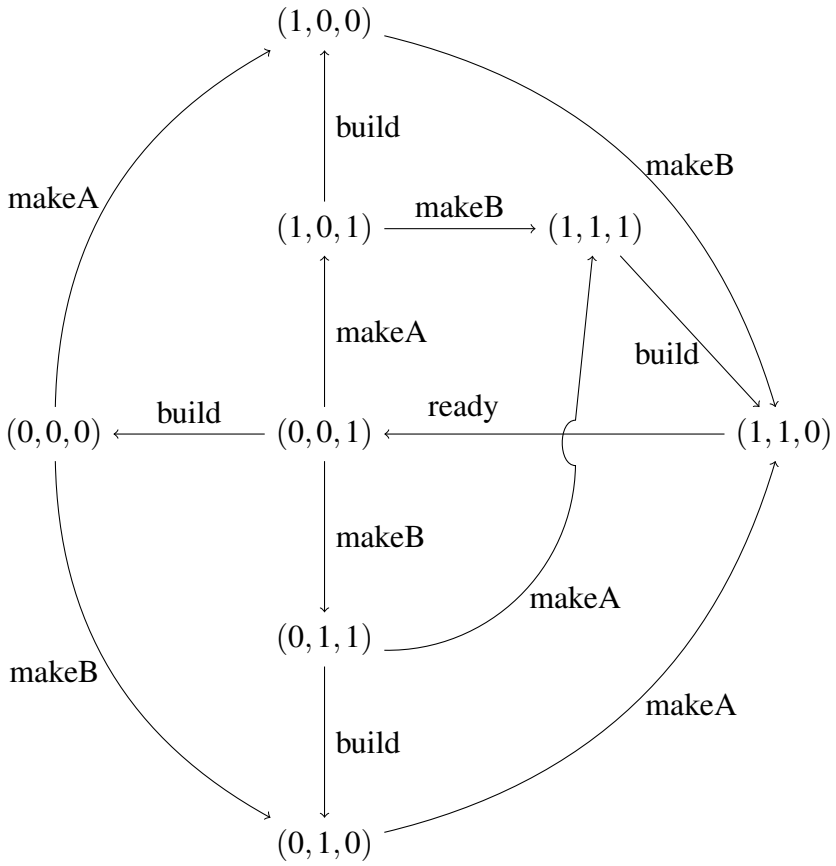


LTS

1. LTSe für MAKE_A, MAKE_B und BUILD.



2. Parallelkomposition ergibt das LTS von FACTORY:



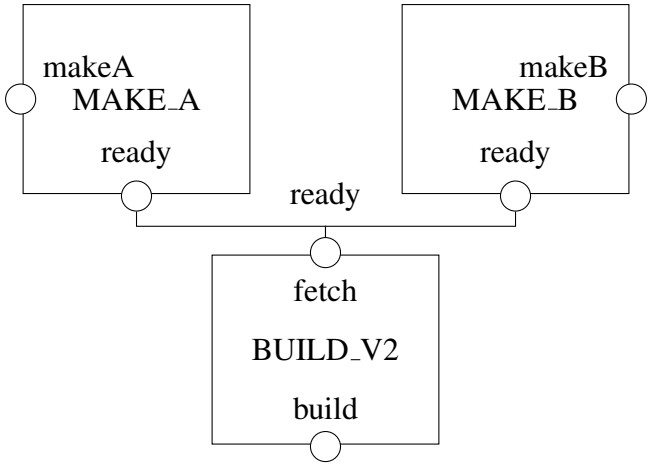
ACHTUNG: Die Umbenennung findet vor der Parallelkomosition statt!

- Variante mit Synchronisation durch Umbenennung.

  FSP-Prozesse

```
MAKE_A = (makeA -> ready -> MAKE_A).
MAKE_B = (makeB -> ready -> MAKE_B).
BUILD_V1 = (fetch -> build -> BUILD_V1).
||FACTORY_V1 = (MAKE_A||MAKE_B||BUILD_V1)/{ready/fetch}.
```

  Strukturdiagramm

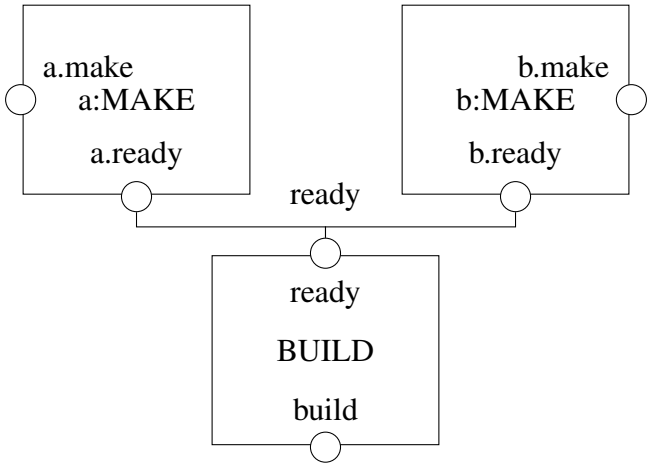


- Variante mit Prozesskopien und Synchronisation durch Umbenennung.

FSP-Prozesse

```
BUILD = (ready -> build -> BUILD).
MAKE = (make -> ready -> MAKE).
||FACTORY_V2 = (a:MAKE||b:MAKE||BUILD)/{ready/{a.ready,b.ready}}.
```

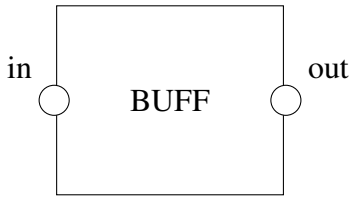
Strukturdiagramm



Aufgabe 2

Ein einelementiger Puffer werde durch folgenden Prozess beschrieben:
$BUFF = (in \rightarrow out \rightarrow BUFF)$.

für den einelementigen Buffer



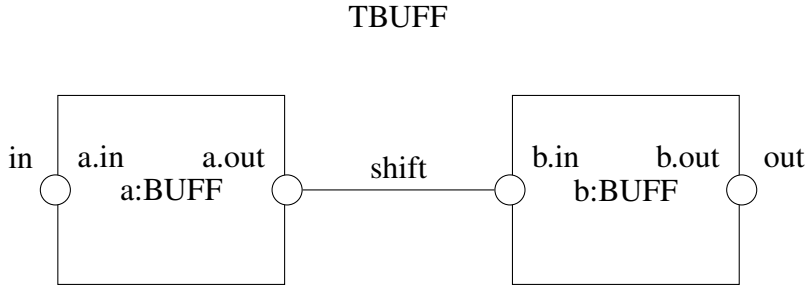
(a) Modellieren Sie einen zweielementigen Puffer TBUFF durch geeignete Hintereinanderschaltung zweier einelementiger Puffer und geben Sie das Strukturdiagrammm des Prozesses an. Geben Sie auch das LTS von TBUFF an.

Lösung

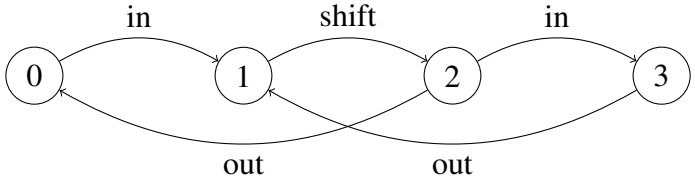
FSP-Prozesse

```
BUFF = (in->out->BUFF).
||TBUFF = (a:BUFF||b:BUFF)/{in/a.in, shift/a.out, shift/b.in, out/b.out}.
//oder kuerzer
///||TBUFF = (a:BUFF||b:BUFF)/{in/a.in, shift/{a.out,b.in}, out/b.out}.
```

Strukturdiagramm



LTS von TBUFF



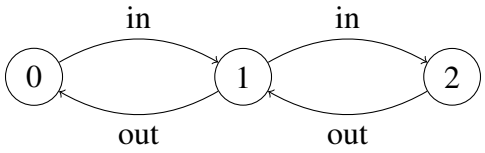
(b) Wie könnte ein zweielementiger Puffer auch direkt (ohne Zusammensetzen zweier einelementiger Puffer) durch einen sequentiellen FSP Prozess beschrieben werden?

Lösung

FSP-Prozesse

```
SBUFF = (in -> ONE),
ONE = (out -> SBUFF
      |in -> TWO),
TWO = (out -> ONE).
```

LTS von SBUFF



Kommentar: SBUFF entspricht einer Anforderungsspezifikation.

Durch beobachtbare Äquivalenz lässt sich prüfen, dass TWOBUFF die Anforderungen erfüllt.

Im LTSA-Tool lässt sich das durch Minimierung erreichen.

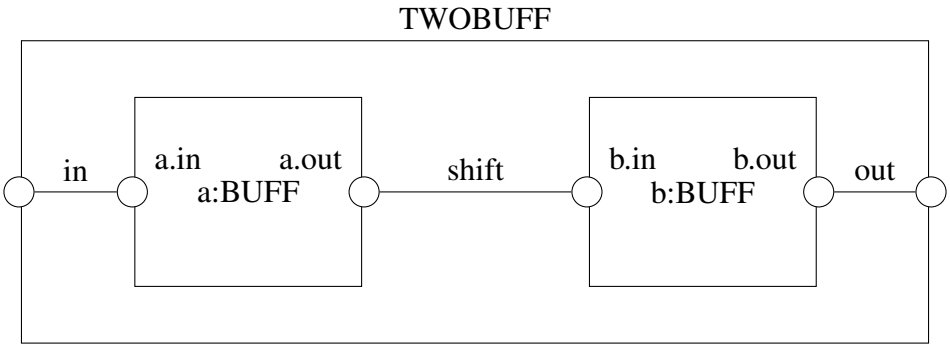
(c) Modellieren Sie das nach außen sichtbare Verhalten des zweielementigen Puffers aus Teil a) durch einen Prozess TWOBUFF und geben Sie das Strukturdiagrammm von TWOBUFF an. Geben Sie das LTS von TWOBUFF an. Geben Sie ein dazu beobachtbar äquivalentes LTS mit einer minimalen Menge von Zuständen an. Vergleichen sie dieses mit dem LTS Ihrer Lösung aus Teil b).

Lösung

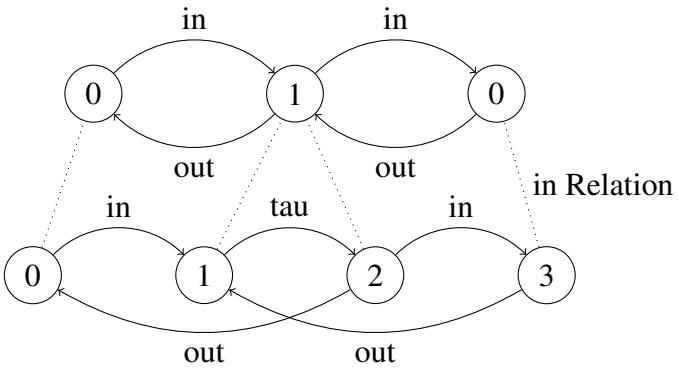
FSP-Prozesse

```
BUFF = (in -> out -> BUFF).

||TBUFF = (a:BUFF||b:BUFF)/{shift/a.out, shift/b.in, in/a.in, out/b.out}.

||TWOBUFF = TBUFF@{in, out}
```

Strukturdiagramm



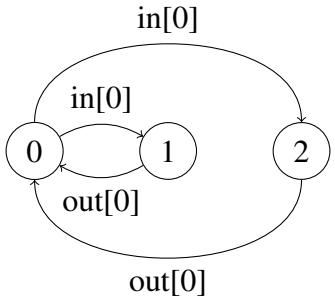
LTSe von SBUFF und TWOBUFF



(d) Berücksichtigen Sie in den Teilen a) - c) nun auch Daten im Bereich 0..1, die von dem Puffer gemäß des FIFO-Prinzips eingelesen und ausgegeben werden sollen.

Lösung

Vorüberlegung



FSP-Prozesse

```
1 //Teil d)
2
3 range T = 0..1
4 DBUFF = (in[i:T]->out[i]->DBUFF).
5
6 //zu a)
7 ||DTBUFF = (a:DBUFF||b:DBUFF)/{in/a.in, shift/{a.out,b.in}, out/b.out}.
```

```
1 //zu b)
2 //sequentieller Prozess mit Daten
3 DSBUFF = (in[i:T] -> ONE[i]),
4 ONE[i:T] = (out[i] -> DSBUFF
5             |in[j:T] -> TWO[j][i]),
6 TWO[j:T][i:T] = (out[i] -> ONE[j]).
```

```
1 //zu c)
2 ||DTWOBUFF = DTBUFF@{in, out}.
3 //Das minimalisierte LTS von DTWOBUFF stimmt mit dem LTS von DSBUFF ueberein
```

Aufgabe 3

Gegeben sei der Prozess ARADIO von Übungsblatt 3, Aufgabe 1; siehe Anhang.

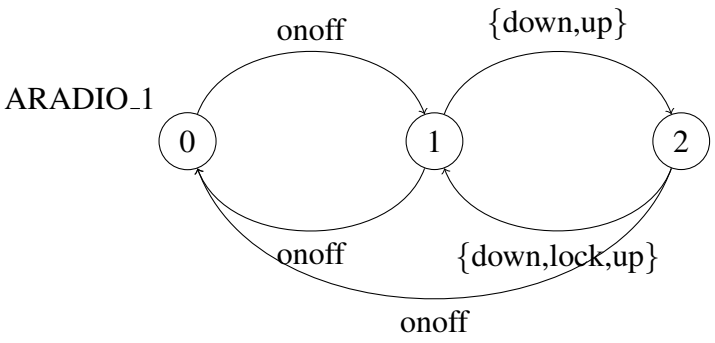
(a) Definieren Sie einen Prozess ARADIO_1, der die Aktionen godown, gomax, goup und gomin des RADIO Prozesses verbirgt und geben Sie das zugehörige, bzgl. beobachtbarer Äquivalenz minimalisierte LTS an.

Lösung

FSP-Prozess

```
1 ||ARADIO_1 = ARADIO\{godown, gomax, goup, gomin}.
```

LTS

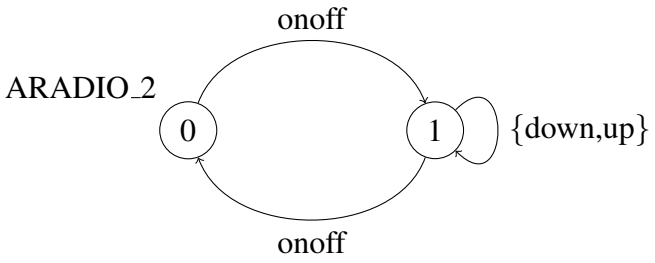


(b) Definieren Sie einen Prozess ARADIO_2, der in einer Schnittstelle die Aktionen onoff, down und up anbietet und geben Sie das zugehörige, bzgl. beobachtbarer Äquivalenz minimalisierte LTS an.

FSP-Prozess

```
||ARADIO_2 = ARADIO@{onoff,down,up}.
//oder
|||ARADIO_2 = ARADIO_1\{lock}.
```

LTS



Anhang

```
//Uebungsblatt 4
//Aufgabe 3


const MAX = 108
const MIN = 88
range F = MIN..MAX

ARADIO = (onoff -> LOCKED[MAX]),

LOCKED[f:F] = ( onoff -> ARADIO
              | down -> ( when (f  > MIN) godown -> DOWN[f-1]
                        | when (f == MIN) gomax  -> DOWN[MAX])
              | up   -> ( when (f  < MAX) goup   -> UP[f+1]
                        | when (f == MAX) gomin  -> UP[MIN])),

DOWN[f:F] = ( onoff -> ARADIO
            | when (f == MIN) gomax  -> DOWN[MAX]
            | when (f  > MIN) godown -> DOWN[f - 1]
            | {lock, up, down} -> LOCKED[f]),

UP[f:F] = ( onoff -> ARADIO
          | when (f == MAX) gomin -> UP[MIN]
          | when (f  < MAX) goup  -> UP[f+1]
          | {lock, up, down} -> LOCKED[f]).
```