FSP
finite state processes

9 december 2013

David N. Jansen

eerder college

Interleaving

- acties van meerdere processen door elkaar
- niet altijd precies afwisselend
  - processen zijn soms niet even snel

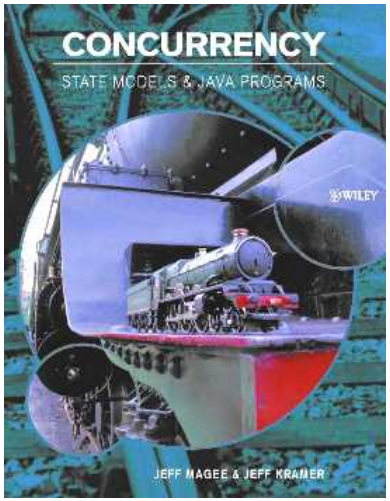
- voorbeeld in FSP-notatie:
  P = (a → b → STOP) Q = (c → d → STOP)
  - a → b → c → d
  - a → c → d → b
  - c → a → d → b
  - a → c → b → d
  - c → a → b → d
  - c → d → a → b

Leerdoel voor vandaag

- begrippen:
  - proces en procesterm
  - LTS = labelled transition system
- vaardigheden:
  - procestermen in FSP schrijven
  - LTSA (LTS analyser) gebruiken

Literatuur

- Concurrency: state models & Java programs / Jeff Magee & Jeff Kramer.



- http://www.doc.ic.ac.uk/ltsa/

- http://www.doc.ic.ac.uk/~jnm/LTSdocumention/FSP-notation.html

Waarom FSP?

- eenvoudigste procesalgebra
- procesalgebra :=
  een taal om processen te beschrijven,
  met nadruk op parallelisme en communicatie

- doel van procesalgebra:
  - specificatie
  - deadlocks en andere fouten vinden

Procesterm

- processen doen atomaire acties
  - idealisering: acties hebben geen tijd nodig
- operatoren om acties samen te stellen
  - STOP geen actie
  - a −> P sequentie eerst a, dan P
  - (P | Q) keuze óf P óf Q
  - (P || Q) parallelisme P en Q interleaven
    (in FSP met beperkingen)

Voorbeelden van eenvoudige procestermen

- LAMP =
  (aan −> uit −> LAMP).
- VERKEERSLICHT =
  (rood −> groen −> geel −> VERKEERSLICHT).
- DRANK = (rood −> koffie −> DRANK
  |groen −> thee −> DRANK
  |leeg −> STOP).
- MUNT = (worp −> kop −> MUNT
  |worp −> munt −> MUNT).

Communicatie tussen processen

- parallele processen
  moeten soms synchroniseren
- in FSP: acties met dezelfde naam synchroon

- voorbeelden
  - (a −> STOP) || (b −> STOP). onafhankelijk
  - (a −> c −> STOP) || (b −> c −> STOP). c tegelijk
  - (a −> b −> STOP) || (b −> a −> STOP). deadlock

Voorbeelden van samengestelde procestermen

- KLOK = (tik −> KLOK).
  RADIO = (aan −> uit −> RADIO).

  ||RADIOWEKKER = (KLOK || RADIO).

- KOK = (kook −> serveer −> KOK)
  GAST = (serveer −> eet −> GAST).

  ||RESTAURANT = (KOK || GAST).

beperking: −> en | niet in samengestelde processen

|| niet in eenvoudige processen

Betekenis van procesterm

- Procesterm beschrijft (mogelijke) acties
- toestand van proces: structuur van de toekomstige mogelijke acties

- procesterm beschrijft toestand van proces – proces is „implementatie” van procesterm

Labelled Transition System

- wiskundig model voor proces: toestanden en transities
  - transities hebben namen: acties
- idee:
  - het proces is altijd in een toestand
  - van tijd tot tijd doet het proces een actie en springt het in een andere toestand
  - idealisering: transitie heeft geen tijd nodig

LTSA

- tool om LTS te visualiseren
  - toestand = bolletje, transitie = pijltje (met actie)
- meerdere LTSen synchroniseren:
  acties met dezelfde naam synchroon

- demonstratie/screenshots

LTSA als concurrency lab

- beschrijf proces met procesterm
- laat LTSA het toestandsdiagram tekenen

- Zijn er deadlock-toestanden?
- Is een ongewenste toestand bereikbaar?
- Gebeurt er überhaupt iets goeds?

LTSA features

Namen van acties veranderen

- indien synchronisatie tussen verschillende acties gewenst:
  - acties herbenoemen (relabelling)
  - alfabet uitbreiden (fictieve synchronisatie)
- indien synchronisatie tussen dezelfde acties ongewenst:
  - actie-prefix:
    - één prefix
    - een verzameling prefixes
  - interne acties (hiding)

Actie-prefix

- BUFFER = (in –> out –> BUFFER).
- buffer voor twee elementen?
  ||BUFF2 = (BUFFER || BUFFER).

- buffer voor twee elementen:
  ||BUFF2 = (a:BUFFER || b:BUFFER).

Acties herbenoemen

- BUFFER = (in –> out –> BUFFER).

- ||QUEUE =
  (BUFFER/{transfer/out}
  ||BUFFER/{transfer/in}).

BUFFER = (in –> transfer –> BUFFER).

BUFFER = (transfer –> out –> BUFFER).

Groter voorbeeld

- Magee / Kramer, ornamental garden (p. 68)
  - een tuin met een oost- en een west-ingang
  - teller moet bijhouden
    hoeveel mensen in de tuin zijn gekomen.
- bevat ook parameters en lokale processen.
  - een lokaal proces is onderdeel van de definitie van een eenvoudige procesterm.
  - een lokaal proces kan parameters hebben.

Ornamental Garden

de eerste keer dat je een parameter gebruikt moet je het bereik aangeven

- VAR = VAL[0],
  VAL[u:0..4] = (read[u] −> VAL[u]
  |write[v:0..4] −> VAL[v]).
- TURNSTILE = (arrive −> INCREMENT),
  INCREMENT = (read[x:0..4] −> write[x+1]
  −> TURNSTILE).
- ||GARDEN = (east:TURNSTILE
  ||west:TURNSTILE
  ||{east,west}::VAR).

Interne acties

- Sommige acties zijn niet relevant voor buitenstaanders
  - voor andere processen (communicatie)
  - voor de gebruiker (LTSA-diagram)

- interne actie heet “tau”
  bijzondere behandeling:
  - geen synchronisatie
  - valt bij minimalisatie in LTSA-diagram weg

Voorbeeld

- QUEUE met interne acties

- operator:
  \ {verzameling van interne acties}

- ||QUEUE =
  (BUFFER/{transfer/out}
  ||BUFFER/{transfer/in}) \ {transfer}.

Requirements als procestermen

- speciaal soort procesterm
- test of de rest van het systeem aan requirement voldoet

- speciale toestand “ERROR” met nummer –1.
  - deadlock-toestand
  - acties die niet uitdrukkelijk worden toegestaan leiden naar –1.

Voorbeelden van requirements

- Het is verplicht te kloppen voor intreden
  (en het is verplicht in te treden na kloppen):

  property POLITE = (knock −> enter −> POLITE).

- property MUTEX =
  (p[i:1..3].enter −> p[i].exit −> MUTEX).

Liveness-requirements

- progress property := geen starvation
- voldoende voor veel liveness requirements

- syntax: progress *Naam* = {*acties*}
- betekenis:
  In elke toestand kan het proces
  minstens één van de acties uitvoeren,
  nu of later.

Voorbeeld: oneerlijke munt

- MUNT = (kies −> KMUNT
  |kies −> MMUNT),
  KMUNT = (worp −> kop −> KMUNT),
  MMUNT = (worp −> munt −> MMUNT).

- progress P1 = {kop}
- progress P2 = {munt}

Oefenopgave

Een drankautomaat verkoopt suikerola voor EUR 0,30 per flesje. Hij accepteert munten van EUR 0,10, EUR 0,20 en EUR 0,50 en geeft wisselgeld.

Modelleer de automaat als FSP-procesterm.

(Het alfabet van de automaat is dus {inworp10, inworp20, inworp50, suikerola, wissel10, wissel20, wissel50}.)

voorbeelduitwerking

```
AUTOMAAT = AUTOMAAT[0],
  AUTOMAAT [i:0..70] = (when (i<30)     inworp10 -> AUTOMAAT[i+10]
                       |when (i<30)     inworp20 -> AUTOMAAT[i+20]
                       |when (i<30)     inworp50 -> AUTOMAAT[i+50]
                       |when (i>=30)    suikerola -> WISSEL[i-30]),
  WISSEL [i:0..40] =  (when (i>=10)    intern -> wissel10 -> WISSEL[i-10]
                      |when (i>=20)    intern -> wissel20 -> WISSEL[i-20]
                      |when (i>=50)    intern -> wissel50 -> WISSEL[i-50]
                      |when (i<10)     intern -> AUTOMAAT) \ {intern}.
```

voorbeelduitwerking

```
AUTOMAAT = AUTOMAAT[0],
  AUTOMAAT [i:0..7  ] = (when (i<3  )     inworp10 −> AUTOMAAT[i+1  ]
                        |when (i<3  )     inworp20 −> AUTOMAAT[i+2  ]
                        |when (i<3  )     inworp50 −> AUTOMAAT[i+5  ]
                        |when (i>=3  )  suikerola −> WISSEL[i−3  ]),
  WISSEL [i:0..4  ] =  (when (i>=1  )   intern −> wissel10 −> WISSEL[i−1  ]
                       |when (i>=2  )   intern −> wissel20 −> WISSEL[i−2  ]
                       |when (i>=5  )   intern −> wissel50 −> WISSEL[i−5  ]
                       |when (i<1  )    intern −> AUTOMAAT) \ {intern}.
```

Basisidee van opdracht 2

- Beschrijf een eenvoudig systeem in FSP
- Gebruik LTSA om te onderzoeken
  of het systeem fouten of deadlocks heeft.
- Verbeter het systeem tot de fouten
  verdwenen zijn...

- ik maak de opdracht morgen af.