ANTWOORDMODEL Voorbeeldtentamen Voortgezette informatica (TVIT)

Ongeroosterd

Instructie

Dit voorbeeldtentamen is bedoeld om je een indruk te geven van het soort, en de hoeveelheid, vragen die je kunt verwachten op het tentamen. Je kunt deze maken tijdens de voorbereiding op de toets om te controleren of je de stof beheerst.

Er kunnen aan dit voorbeeldtentamen geen rechten worden ontleend. Er kunnen uiteraard ook vragen gesteld worden die onderdeel zijn de besproken stof, maar niet in dit voorbeeld aan bod komen.

Je hebt 3 uur de tijd voor het maken van de opgaven.

Succes!

Vraag 1: Eindige automaten (10)

(a) Wat is het nadeel van een non-deterministische eindige automaat tov een deterministische eindige automaat? (1)

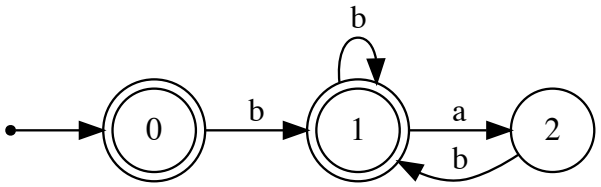
Antwoord:
Vanuit een toestand waar voor een input symbool meerdere overgangen zijn kan nu niet eenduidig de nieuwe toestand bepaald worden. Ze kunnen dus niet (direct) gebruikt worden voor een implementatie.

(b) Hoe zou je kort het belangrijkste verschil tussen een reguliere expressie en een grammatica kunnen omschrijven? (1)

Antwoord:
Grammatica's kunnen als recursieve reguliere expressies beschouwt worden. Iedere niet recursieve grammatica kan als een gewone reguliere beschreven worden, maar grammatica's met recursieve regels kunnen niet als gewone reguliere expressie uitgedrukt worden.

(c) Beschouw de taal van willekeurige strings bestaande uit de symbolen `a` en `b` waarin iedere `a` vooraf wordt gegaan en gevolgd wordt door minimaal 1 `b`. Bijvoorbeeld: `babab` of `bbabbabbb`. (2)
Maak voor deze taal een eindige automaat.

Antwoord:

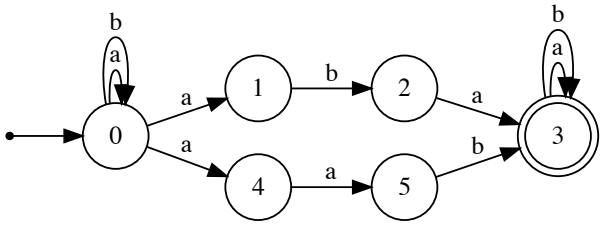


(d) Beschouw de taal van willekeurige strings bestaande uit de symbolen `a` en `b` waarin de substring `aba` of `aab` minimaal 1 keer voorkomt, bijvoorbeeld: `bbbaba`, `bbbaabbba`. (2)
Geef een reguliere expressie die deze taal beschrijft.

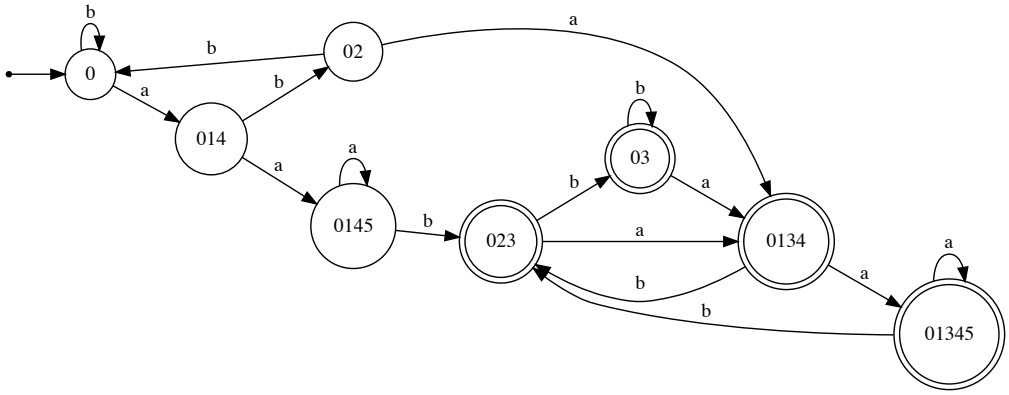
Antwoord:
`(a|b)* (aba|aab) (a|b)*`

(e) Maak een eindige automaat die de taal uit de voorgaande vraag accepteert. Doe dit door eerst een niet deterministische versie te maken aan de hand van de reguliere expressie en deze daarna deterministisch te maken. Laat beide versies zien. (4)

Antwoord:
Non-deterministische versie:



Deterministische versie:



- Opgave mag of helemaal systematisch met extra epsilon overgangen, of compacter zoals hier beschreven
- 2 punten voor non-deterministische oplossing
- 2 punten voor deterministische oplossing
- 1 punt aftrek per fout

Vraag 2: Grammatica's (8)

(a) Maak een grammatica voor de volgende taal bestaande uit de symbolen 0,1: (2)

Alle strings waarbij een string bestaande uit een aantal 0-en (1 of meer) gevolgd wordt door een oneven aantal 1-en en daarna net zoveel nullen als in het begin (bv 0011100 of 010)

Antwoord:

Bijvoorbeeld:

```
string = '0' string '0' | '0' enen '0'
enen = '1' | '1' enen '1'
```

Andere grammatica's die dezelfde taal beschrijven zijn ook goed.

- Helemaal correct 2 punten
- Kleine fout, maar correcte aanpak, 1 punt

(b) Maak een grammatica voor de volgende taal bestaande uit de symbolen 0,1: (2)

Alle strings die omgekeerd hetzelfde zijn (bv 00, 010, 1001). Ze mogen dus even of oneven lengte hebben.

Antwoord:

Bijvoorbeeld:

```
string = '0' | '1' | '11' | '00' | '0' string '0' | '1' string '1'
```

Andere grammatica's die dezelfde taal beschrijven zijn ook goed.

- Helemaal correct 2 punten
- Kleine fout, maar correcte aanpak, 1 punt

(c) Geef de grammatica voor rekenkundige infix expressies met getallen, haakjes en de operatoren +,−,∗,/ met hun gebruikelijk prioriteiten. (2)

Antwoord:

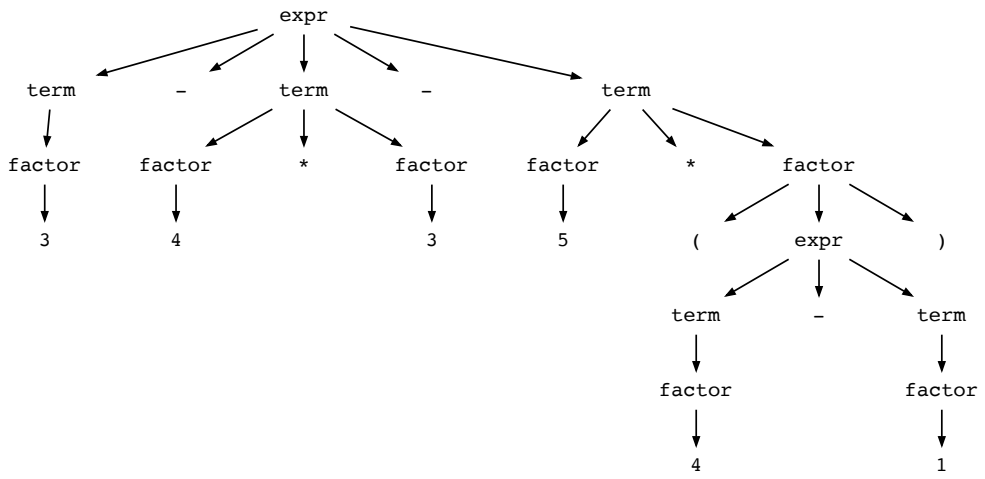
```
expr      = term (plusoper term)*
term      = factor (multoper factor)*
plusoper  = '+' | '-'
multoper  = '*' | '/'
factor    = getal | '(' expr ')'
```

Namen van de regels hoeven niet exact te kloppen. Extra tussenregels mag ook. Als de gelaagdheid van term en factor er maar in zit.

- 1 punt voor complete set regels
- 1 punt voor correcte prioriteiten

(d) Geef met behulp van de grammatica uit de vorige vraag de ontleedboom voor: `3 - 4 * 3 - 5 * ( 4 - 1)` (2)

Antwoord:



- 2 punten voor volledig correcte boom
- 1 punt aftrek per fout

Vraag 3: Turingmachines (10)

In de deelvragen mag je het volgende aannemen: de invoer bestaat uit 0, 1 en _ eventueel met hulpsymbolen. Je mag steeds zelf kiezen welke symbolen je schrijft. Rechts van de invoer is de tape gevuld met (oneindig veel) _.

(a) Een informaticus beweert een programmeertaal te hebben ontwikkeld die krachtiger is dan een Turing machine, dwz er kunnen berekeningen op worden gemaakt die niet met een Turing machine kunnen worden gedaan. Is dit mogelijk? (1)

Antwoord:
Nee, iedere mogelijke berekening kan met een Turing Machine gedaan worden (per definitie).

(b) Geef een voorbeeld van een niet berekenbaar probleem. (1)

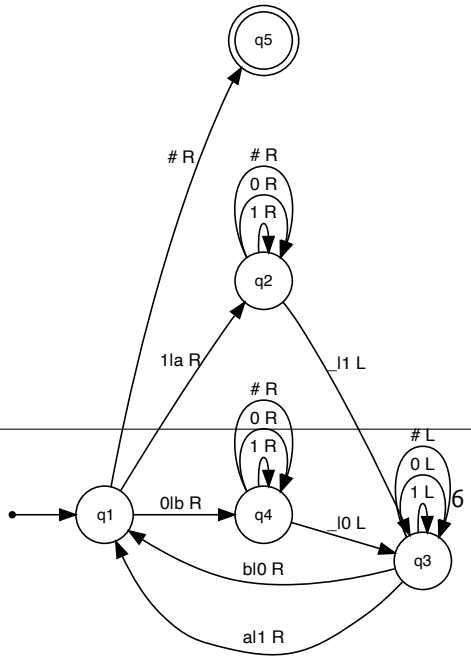
Antwoord:

- Het Halting probleem: gegeven een programma, stopt of loopt dit programma.
- Oplossen diophantische vergelijkingen
- Post correspondence probleem.
- Alle punten (1) voor een correct voorbeeld. Andere bekende voorbeelden buitenom de gegeven theorie mogen ook.

(c) De invoer bestaat uit een 0-1 string afgesloten met een #. Geef een Turing machine die de gehele string naar de tape achter # kopieert. Zorg er (zo nodig) ook voor dat de originele invoer weer wordt teruggezet. (4)

Antwoord:
Aanpak:

- Een voor een een 1 of 0 naar het eind verplaatsen.
- Markeren waar je geblevent bent met een a of b
- Einde kun je herkennen door _
- Als je bij het # is alles klaar




```
q1 1 a R q2

q2 1 1 R q2
q2 0 0 R q2
q2 # # R q2
q2 _ 1 L q3

q3 1 1 L q3
q3 0 0 L q3
q3 # # L q3
q3 a 1 R q1
q3 b 0 R q1

q1 0 b R q4
q4 1 1 R q4
q4 0 0 R q4
q4 # # R q4
q4 _ 0 L q3

q1 # # R q5

start q1
accept q5
```

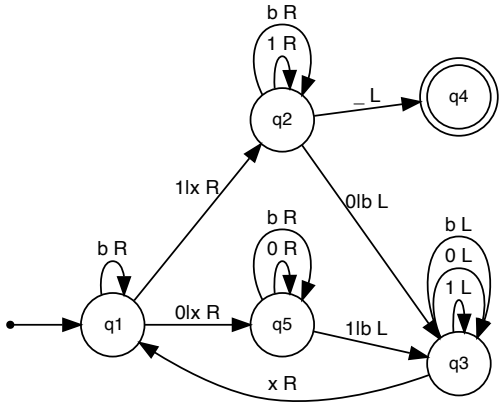
- Twee punten voor duidelijk maken aanpak
- Twee punten voor uitwerking

(d) De invoer bestaat uit enen gevolgd door nullen. Maak een Turing-machine die de invoer accepteert als het aantal enen groter is dan het aantal nullen. (4)

Antwoord:

Aanpak:

- Bekijk het eerste teken en streep het af. Voor een 0 zoek je een bijbehorende 1, voor een 1 een bijbehordende 0.
- Als je die kunt vinden ga je terug naar het begin en start je opnieuw
- Als je die niet kunt vinden weet je van welke er meer zijn
- Als je bij het eind komt, zijn het er evenveel en accepteer je dus niet.
- Bij het heen en weer spoelen wel steeds opletten dat je de afgestreepte tekens negeert.




```
q1 1 x R q2
q2 0 b L q3
q2 1 1 R q2
q2 b b R q2
q2 _ _ L q4

q3 1 1 L q3
q3 0 0 L q3
q3 b b L q3
q3 x x R q1

q1 0 x R q5
q5 1 b L q3
q5 0 0 R q5
q5 b b R q5

q1 b b R q1

start q1
accept q4
```

- Twee punten voor duidelijk maken aanpak
- Twee punten voor uitwerking

Vraag 4: IJMVM (14)

(a) Geef voor de volgende rekenkundige expressies IJVM code die de expressie uitrekent: (3)

- `4 * 7 -3 * 2 - 5`
- `3 * 8 - 5 + 3`

Antwoord:

`4 * 7 - 3 * 2 - 5`

```
bipush 4
bipush 7
imult
bipush 3
bipush 2
imult
isub
bipush 5
isub
print
```

`3 * 8 - 5 + 3`

```
bipush 3
bipush 8
imult
bipush 5
isub
bipush 3
iadd
print
```

De `print` instructie hoeft er niet bij te staan. Mag wel.

- Volledige punten bij alles goed
- 1 punt aftrek per fout(je)

(b) Beschouw de volgende Python functie en aanroep: (4)

```
def g(n,k):
    a = 0
    while a <= n:
        a = a + k
    return n + k - a

print(g(8,3))
```

Geef de IJVM code van de functie g inclusief de aanroep en `print`.

Antwoord:

```
bipush 8
bipush 3
call 2 1 1
```

```
  print
  stop
1 bipush 0
  istore 2
2 iload 0
  iload 2
  isub
  iflt 3
  iload 2
  iload 1
  iadd
  istore 2
  goto 2
3 iload 0
  iload 1
  iadd
  iload 2
  isub
  ireturn
```

- Volledige punten voor correcte letterlijke vertaling
- 1 punt aftrek per ontbrekende of foute vertaling van python statement

(c) Geef aan hoe de stack er uitziet op het moment dat je de eerste keer de while loop uit de vorige vraag ingaat. (2)

Antwoord:

`[8 3 0 3 0 ]`

De eerste twee getallen zijn de argumenten `n` en `k` van de aanroep `g(8,3)`. Het derde getal is de lokale variabele `a` die voor de loop de waarde `0` heeft. Het vierde getal is de het return-adres van de functie aanroep (de PC was `3` toen de functie was aangeroepen) Het vijfde getal is het begin van het huidige stackframe (`0` omdat het de eerste functie aanroep) zodat de argumenten en lokale variabelen weer vrijgegeven kunnen worden.

- 2 punten voor goede antwoord
- 1 punt bij fout antwoord correcte toelichting
- 1 punt bij correct antwoord en foute toelichting

(d) Beschouw de volgende IJVM code: (5)

```
   bipush 9
   call 1 1 11
   print
   stop
11 bipush 1
   istore 1
33 iload 0
   ifeq 22
   iload 0
```

```
   bipush 1
   isub
   istore 0
   bipush 1
   iload 1
   isub
   istore 1
   goto 33
22 iload 1
   ireturn
```

Geef hiervoor equivalente Python code.

Antwoord:

```python
def f(a):
    x = 1
    while a > 0:
        a = a - 1
        x = 1 - x
    return x

print(f(9))
```

- Volledige punten bij correcte letterlijke vertaling
- 1 punt aftrek per ontbrekend python statement

Vraag 5: Recursie (8)

(a) Beschouw (nogmaals) de volgende python functie: (4)

```
def g(n,k):
    a = 0
    while a <= n:
        a = a + k
    return n + k - a
```

Geef een recursieve versie van de functie g.

Antwoord:

```
def g(n,k,a = 0):
    if a <= n:
        return g(n,k,a + k)
    return n + k - a
```

- Volledige punten voor correcte functie met recursie
- Punten naar ratio volledigheid functie
- Minimaal 2 punten voor correcte inzet recursie

(b) Beschouw de volgende recursieve python functie: (4)

```
def r(s):
    if len(s) <= 1:
        return s
    else:
        return r(s[1:]) + s[0]
```

Geef een iteratieve versie van de functie `r`.

Antwoord:

```
def r(s):
    res = ""
    for c in s:
        res = c + res
    return res
```

Andere varianten van strings omdraaien met loops mogen ook. Bijv met `while` loop.

- Volledige punten voor correcte functie met een loopconstructie
- Punten naar ratio volledigheid functie
- Minimaal 2 punten voor correcte inzet iteratie

EINDE TOETS