Reguliere Expressies

Theorie en praktijk – Leerboek voor het VO

Huub de Beer

Eindhoven, 31 mei 2011

Inhoudsopgave

1 Inleiding: patronen en tekst

1.1 Patronen in tekst zijn belangrijk

Patronen zijn overal. Elk telefoonnummer heeft tien cijfers; een e-mailadres bestaat uit een stuk aaneengesloten tekst voor het '@'-teken en een domeinnaam erna; een naam bestaat uit een voor- en een achternaam, soms gescheiden door een tussenvoegsel; enzovoorts. Je (her)kent deze patronen en dat vergemakkelijkt communicatie: je weet wat je kunt verwachten en hoe je deze patronen moet interpreteren.

Voorbeelden van patronen

- telefoonnummer: 06-22345465
- e-mailadres: konijn23@beestjes.nl
- tabelformaat: "naam", "titel", "jaar";
- getal in de wetenschappelijke notatie: $0.3466e$-23
- datum: 23-4-2009

Figuur 1.1: Enkele voorbeelden van patronen

Patronen zijn niet alleen belangrijk bij communicatie tussen mensen onderling. Patronen maken het namelijk mogelijk om met een computer te communiceren. Sterker nog, door middel van patronen kunnen verschillende computers met elkaar 'praten' en elkaar 'begrijpen': deze patronen zijn door programmeurs afgesproken en vervolgens geïmplementeerd.

Veel digitale informatie is gecodeerd in tekst. 'Menselijke informatie', zoals documenten, chatberichten, e-mails, enzovoorts, bestaan natuurlijk al uit tekst. Vaak zal die tekst vol zitten met codes om opmaak en metadata aan te geven; een computer herkent dergelijke codes, het zijn patronen, en kan daar relevante informatie uit destilleren.

Daarnaast zijn er genoeg digitale informatiebronnen die niet per sé direct voor menselijke consumptie bedoeld zijn, zoals een beschrijving van een 3D-avatar in een spel, een database, technische data van een printer of een thermostaat, enzovoorts. Maar ook dit soort bronnen bestaan over het algemeen uit tekst. Weliswaar tekst waar je geen wijs uit kunt zonder de precieze

specificatie te kennen, maar er zit structuur in de tekst. Die structuur kun je beschrijven met een of enkele patronen.

1.2 Tekstverwerken en patronen

Als een computer informatie verwerkt, verwerkt de computer tekst. Bijna elk niet triviaal programma zal dus in meer of mindere mate tekstverwerken: het valideren van invoer van een formulier, het schrijven naar of lezen van een bestand, het verwerken van een verzoek vanaf het internet, enzovoorts. Tekstverwerken is een belangrijk onderdeel van programmeren. Elk programma dat op een of andere manier met de buitenwereld communiceert, bijvoorbeeld via een bestand of een invoerveld, zal de invoer eerst moeten verwerken voordat het de informatie kan gebruiken.

In de informatica noemen we een tekst een **string**. Bijna elke programmeertaal kent eenvoudige stringoperaties zoals het bepalen van de lengte, het vinden van een deelstring, het weglaten van een gedeelte van de string, enzovoorts. Naast deze eenvoudige operaties bestaan er krachtige stringverwerkingsopdrachten gebaseerd op **reguliere expressies**.

Met behulp van een reguliere expressie kun je een patroon formuleren. Vervolgens kun je in strings zoeken naar deze patronen en een bepaalde actie ondernemen wanneer een bepaald patroon wel of niet voorkomt. Een belangrijke toepassing van reguliere expressies vind je in de compiler of interpreter van een programmeertaal. Zo'n compiler of interpreter gebruikt reguliere expressies om alle losse onderdelen van een programma, zoals keywords, getallen, namen, haakjes, enzovoorts, te herkennen en te controleren.

Zolang de te verwerken tekst voldoet aan een aantal patronen die met een reguliere expressie beschreven kunnen worden, kun je reguliere expressies gebruiken om die tekst te verwerken. Je kunt dan denken aan het valideren van invoer tot het inlezen van een tabel in het CSV-formaat[1]. Een goede programmeur beheerst daarom reguliere expressies.

Programmeurs hebben nog een tweede reden om reguliere expressies te kennen. Programmacode is tekst vol met patronen en een programma bestaat vaak uit tientallen zo niet honderden tekstbestanden. Je kunt reguliere expressies inzetten om bepaalde onderdelen in je programma te vinden of automatisch te vervangen. Stel je hebt je programma geschreven voor een MS Windows computer waar de padscheider een backslash is. Wil je je programma geschikt maken voor Linux of Mac OSX, die beide een slash als padscheider gebruiken, dan zul je alle paden in je programma moeten aanpassen. Met behulp van reguliere expressies kun je alle paden vinden, die hebben immers een vast patroon, en vervolgens alle backslashes vervangen door slashes.

In de volgende hoofdstukken ga je leren werken met reguliere expressies.

[1] CSV: comma separated values

We beginnen met het opstellen van reguliere expressies voor eenvoudige patronen. Daarna leer je welke automaten deze reguliere expressies herkennen. Deze eerste twee hoofdstukken behoren tot de theoretische achtergrond van reguliere expressies.

In het praktische deel leer je reguliere expressies schrijven in een speciale taal die je in veel programmeertalen en programma's kunt gebruiken. Je gebruikt deze reguliere expressies om naar patronen te zoeken en te vervangen.

Een goede uitgebreide website over reguliere expressies is: `http://www.regular-expressions.info/`.

1.3 Opgaven

Bij deze opgaven gaat het niet zozeer om het goede antwoord, als dat al bestaat, maar dat je eens rustig nadenkt over patronen, verschillen tussen patronen en de kwaliteit van patronen.

1.1 Hieronder zie je de datum 23-12-2009 op verschillende manieren weergegeven. Omschrijf bij elke manier het algemene patroon.

a) 12-9-09
b) 12-09-2009
c) zondag 12 september 2009
d) 20090912

1.2 Hieronder zie je verschillende soorten getallen. Geef bij elke soort het algemene patroon aan.

a) natuurlijke getallen: $0, 1, 2, 3, \ldots$
b) gehele getallen: $\ldots -3, -2, -1, 0, 1, 2, 3, \ldots$
c) reële getallen: $0.2354, 23.346, 0.004e-34, -12, \ldots$
d) rationale getallen: $\frac{1}{2}, \frac{23}{43554}, -\frac{23}{3}, \ldots$

1.3 Geef een omschrijving van het algemene patroon van een CSV-bestand zoals omschreven op http://nl.wikipedia.org/wiki/Kommagescheiden_bestand. Wat is het patroon van een tabel, een rij en een cel?

De antwoorden vind je in Bijlage A.1 op bladzijde 29.

2 Theorie I – Reguliere expressies

2.1 Definitie van alfabet, zin en taal

Je bent bekend met het Latijnse alfabet, dat gebruik je immers dagelijks. Er bestaan verschillende alfabetten, naast het Latijnse is er het Griekse en het Hebreeuwse. Een alfabet heeft iets met een taal te maken. Bepaalde combinaties van symbolen uit een Latijnse alfabet behoren bijvoorbeeld tot de Nederlandse taal. (Zoals elk woord uit deze paragraaf.)

Het Nederlands is een **natuurlijke taal**: deze taal is cultuur-historisch zo gegroeid. Naast natuurlijke talen bestaan er ook **formele talen**, dat zijn zogenaamde synthetische talen, oftewel 'bedachte' talen. Als we de theorie van reguliere expressies bespreken, gebruiken we formele talen. Enkele definities:

Definitie 2.1.1 *(Alfabet)*
Een verzameling unieke symbolen noemen we een ***alfabet***

Zo vormen de drie symbolen '+', '3' en 'g' een alfabet. Ook '$\star$', '$\bullet$', '$\oplus$', '$\clubsuit$' en '$\biguplus$' vormen samen een alfabet. Het ons welbekende Latijnse alfabet plus de cijfers 0 tot en met 9 geven we een aparte naam:

Definitie 2.1.2 *(Alfabet $\mathbb{G}$)*
Alfabet $\mathbb{G}$ = `a', `b', `c', `d', `e', `f', `g', `h', `i', `j', `k', `l', `m', `n', `o', `p', `q', `r', `s', `t', `u', `v', `w', `x', `y', `z', `0', `1', `2', `3', `4', `5', `6', `7', `8', `9'. Met andere woorden, alfabet $\mathbb{G}$ bestaat uit de kleine letters en de cijfers 0 tot en met 9. We noemen dit alfabet $\mathbb{G}$ omdat het lijkt op het gewone alfabet.

Vanaf nu gebruiken we alfabet $\mathbb{G}$ vaak in voorbeelden en opgaven. Let overigens op dat de ' ' (spatie) en andere leestekens niet in $\mathbb{G}$ zijn opgenomen.

Definitie 2.1.3 *(Zin)*
een reeks aaneengesloten symbolen uit een alfabet $\mathbb{A}$ noemen we een ***zin*** over alfabet $\mathbb{A}$.

Voorbeelden: gegeven $\mathbb{G}$, dan zijn 'aapnootmies', '06df34gt' zinnen; '06-11' en 'aap noot mies' zijn geen zinnen, '-' en ' ' zijn immers geen symbolen

in alfabet $\mathbb{G}$.

Definitie 2.1.4 *(Taal)*
Een verzameling zinnen gemaakt met symbolen uit een alfabet $\mathbb{A}$ noemen we een ***taal*** over alfabet $\mathbb{A}$. Vaak worden eisen gesteld waaraan zinnen in een taal moeten voldoen: niet alle combinaties van symbolen in $\mathbb{A}$ zijn dan zinnen in de taal.

Voorbeeld: Gegeven is alfabet $\mathbb{A}$ = 'a', 'b' en 'c'. De taal $\mathfrak{L}()$ over alfabet $\mathbb{A}$ bestaat uit alle zinnen (over $\mathbb{A}$) die met een 'a' beginnen. 'aab', 'a', 'aaaaaaa', 'acba' en 'acccc' zijn dus zinnen in de taal $\mathfrak{L}()$. 'a4', '23' en 'Aab' zijn geen zinnen in de taal $\mathfrak{L}()$: deze zinnen beginnen immers niet met 'a' of bevatten symbolen die niet in alfabet $\mathbb{A}$ voorkomen. Let op: 'A' zit niet in $\mathbb{A}$.

2.2 Definitie van een reguliere expressie

De taal $\mathfrak{L}()$ uit het vorige voorbeeld wordt gedefinieerd met behulp van een patroon: een 'a' gevolgd door een willekeurig aantal symbolen uit alfabet $\mathbb{A}$. We kunnen dit patroon opschrijven met behulp van de volgende reguliere expressie: $a \cdot (a|b|c)^*$.

Definitie 2.2.5 *(Reguliere expressie)*
Gegeven alfabet $\mathbb{A}$. De volgende expressies zijn reguliere expressies over alfabet $\mathbb{A}$:

1. Elk karakterer uit $\mathbb{A}$ is een reguliere expressie. We noemen dit soort reguliere expressies ook wel ***primaire reguliere expressies***.
2. Als r en s reguliere expressies zijn, dan zijn
   - $r \cdot s$
   - $r|s$,
   - r^* en
   - (r)

   dat ook.

Elke reguliere expressie specificeert nul of meer zinnen over alfabet $\mathbb{A}$

In de reguliere expressie $a \cdot (a|b|c)^*$ worden alle operatoren en meta-symbolen die in reguliere expressies voor kunnen komen, gebruikt. We bespreken de verschillende symbolen een voor een:

- $\cdot$ (**concatenatie**) Met behulp van de concatenatie-operator, de $\cdot$, knoop je twee aparte reguliere expressies aan elkaar.

Het meest eenvoudige voorbeeld van de concatenatie is het vormen van een eenvoudige zin, gegeven alfabet $\mathbb{G}$: de reguliere expressie $a \cdot a \cdot p$ geeft de zin 'aap' aan.

- $|$ (**keuze**) Met behulp van de keuze-operator, de $|$, geef je een keuze tussen twee verschillende reguliere expressies aan.

  Voorbeeld, gegeven alfabet $\mathbb{A}$: $a|b|c$ geeft de zinnen 'a', 'b' en 'c' aan.

- * (**herhaling**) Met behulp van de herhalings-operator, de *, geef je aan dat een bepaalde reguliere expressie *nul* of meer keer herhaald kan worden. De lege zin, '' behoort hier dus ook toe.

  Voorbeeld, gegeven alfabet $\mathbb{A}$: a^* geeft de zinnen '', 'a', 'aa', 'aaa', 'aaaa', ... aan, dat wil zeggen alle zinnen die uit alleen a's bestaan.

- (en) (**groepering**) Met behulp van haakjes kun je een (complexere) reguliere expressie opnemen in een grotere reguliere expressie. Deze haakjes werken precies zo als de haakjes in een wiskundige formule.

  Voorbeeld, gegeven alfabet $\mathbb{A}$: $(a|b)^*$ geeft de zinnen '', 'a', 'aa', ..., 'b', 'bb', ..., 'abbaabb', 'bbabbabb', 'bbba', ... aan, dat wil zeggen alle zinnen die uit enkel a's en/of b's bestaan.

Terug naar de bespreking van het voorbeeld $a \cdot (a|b|c)^*$. Deze reguliere expressie geeft dus alle zinnen aan die beginnen met een 'a' en gevolgd worden door een zin die uit enkel a's, b's, en/of c's bestaat.

Voorbeelden, gegeven alfabet $\mathbb{C}$ = '0', '1', '2', '3', '4', '5', '6', '7', '8' en '9':

- Het alarmnummer: $1 \cdot 1 \cdot 2$
- Zinnen die beginnen met '06': $0 \cdot 6 \cdot (0|1|2|3|4|5|6|7|8|9)^*$
- Twee willekeurige cijfers: $(0|1|2|3|4|5|6|7|8|9) \cdot (0|1|2|3|4|5|6|7|8|9)$

Met behulp van een reguliere expressie over een alfabet geef je dus een taal over dat alfabet aan. Zo'n taal noemen we een reguliere taal.

Definitie 2.2.6 *(Reguliere taal)*
Gegeven reguliere expressie r, dan is taal $\mathfrak{L}(r)$ de verzameling zinnen die voldoen aan reguliere expressie r.

Voorbeelden, gegeven alfabet $\mathbb{A}$ = 'a', 'b', '-':

- De taal $\mathfrak{L}((a|b) \cdot - \cdot (a|b))$ = 'a-a', 'a-b', 'b-a' en 'b-b'
- De taal $\mathfrak{L}(a^* \cdot - \cdot (a|b))$ = 'a-a', 'aa-b', 'aaaaa-a', ...
- De taal $\mathfrak{L}(- \cdot -^*)$ = '-', '- -', '- - -', '- - - -', ...

2.3 Opgaven

2.1 Gegeven alfabet $\mathbb{G}$. Geef, indien mogelijk, per reguliere expressie vijf verschillende zinnen die aan die expressie voldoen.

a) $q \cdot q \cdot q$
b) $w \cdot (x|y|z) \cdot w^*$
c) $(k|l|m)|(a|b|c)$
d) $(d|0)^* \cdot (e|1)^*$
e) $a^* \cdot (a|b) \cdot b^*$

2.2 Gegeven alfabet $\mathbb{G}$. Schrijf een reguliere expressie, per deelopgave, die onderstaande zinnen beschrijven:

a) 'aap', 'aak', 'aas', 'aal'
b) '9', '12', ..., '99', '0', '1'; zinnen als '05' komen *niet* voor
c) 'aha', 'ah', 'ahaa', ..., 'ahaaaa', ...

2.3 Gegeven alfabet $\mathbb{B}$ = '0', '1'. Schrijf een reguliere expressie, per deelopgave, die onderstaande zinnen beschrijven:

a) '0110', '001100', '110000', '0000011', '0011000', '000011000', ...
b) '111', '', '1111111', '1', '111111111', ...
c) '01010101', '00001', '10101', '111', '0', ...
d) '0', '1', '10', '11', '100', '101', '110', '111'

2.4 Geef het alfabet die bij onderstaande zinnen hoort en vervolgens de reguliere expressie die de zinnen beschrijft:

a) '10 + 111', '00 - 101', '10 + 000', ...
b) '06-11', '06-12', ... '06-99'
c) '1♣', '3♠', ..., '9♢', '10♠', 'K♡', 'Q♣', 'J♣', ..., '♡', '♢', ...

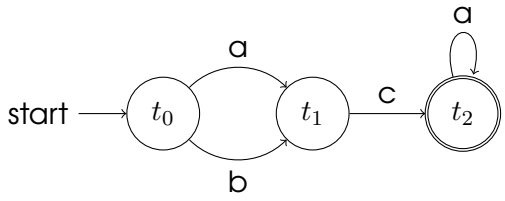
De antwoorden vind je in Bijlage A.2 op bladzijde 30.

3 Theorie II – Deterministische eindige automaten

3.1 Deterministische eindige automaten (DFA)

In het vorige hoofdstuk heb je geleerd hoe je met behulp van reguliere expressies formele talen kunt definiëren waarin alleen zinnen voorkomen die aan de reguliere expressie voldoen. We zeggen ook wel dat reguliere expressies generatief zijn: je kunt een reguliere expressie gebruiken om zinnen in de taal van die expressie te maken.

We zijn echter niet zozeer geïnteresseerd in het maken van zinnen die aan een bepaald patroon voldoen; we zijn geïnteresseerd in het *herkennen* van zinnen die aan een bepaald patroon voldoen. Voor het herkennen van zinnen die aan een reguliere expressie voldoen, bestaat een ander formalisme: **deterministische eindige automaten**.



Figuur 3.1: Een DFA dat zinnen accepteert die in de taal $\mathfrak{L}((a|b) \cdot c \cdot a^*)$.

In het Engels noemen we een deterministische eindige automaat een "deterministic finite automaton" of een "deterministic finite acceptor", we korten deterministische eindige automaten daarom af tot **DFA**. Het Engelse begrip "deterministic finite *acceptor*" geeft duidelijker aan dat een DFA een automaat is waarmee je kunt herkennen of een invoerzin aan een bepaalde reguliere expressie voldoet: als een invoerzin voldoet aan de reguliere expressie dan *accepteert* de automaat de invoerzin als een zin uit de taal van de reguliere expressie.

DFA's worden vaak visueel gerepresenteerd. In dit lesmateriaal doen we dat ook. Een voorbeeld: in Figuur 3.1 zie je de DFA die zinnen uit de taal $\mathfrak{L}((a|b) \cdot c \cdot a^*)$ accepteert.

Een DFA bestaat uit een of meer gelabelde **toestanden**, in Figuur 3.1 zijn dat t_0, t_1 en t_2. Er zijn drie verschillende soorten toestanden: precies

één **starttoestand**, aangegeven als een cirkel met een inkomende pijl "start" (t_0); een of meer **eindtoestanden**, aangegeven als een dubbele cirkel (t_2); en een aantal gewone toestanden, aangegeven met een cirkel (t_1).

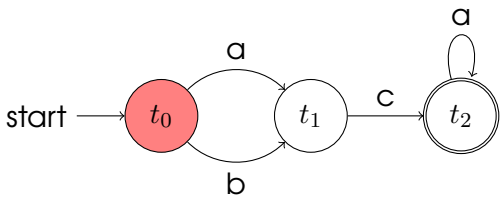
De automaat kan van de ene toestand in een andere overgaan met behulp van een **transitie**. Transities geven we aan als een gelabelde pijl die begint in de ene toestand en naar een andere toestand wijst. Zoals je ziet is het ook mogelijk dat een transitie naar zichzelf verwijst (met een 'a' van t_2 naar t_2). Het label is een symbool uit een alfabet.

DFA's zijn **deterministisch** omdat er in een toestand *nooit* meer dan een transitie met hetzelfde symbool mag bestaan. Zo mag er in het voorbeeld dus geen transitie van t_0 naar t_2 bestaan met het label 'a'. Hierdoor kun je dus altijd precies zeggen wat de automaat gaat doen: we noemen dat deterministisch.

3.2 Het herkennen van een zin

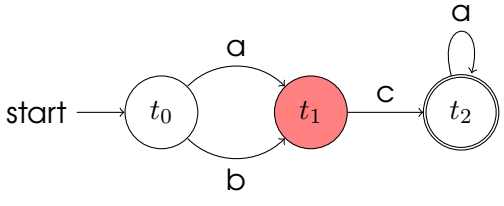
Hoe gaat het herkennen van een zin door een DFA in zijn werk? Gegeven een DFA en een invoer over een alfabet, de DFA accepteert de invoer als een zin als de automaat na het verwerken van het laatste symbool zich in een eindtoestand bevindt. Zodra de automaat zich in een toestand bevindt waar geen volgende stap mogelijk is, óf omdat de invoer op is, óf omdat er geen transities mogelijk zijn gegeven de overgebleven invoer, dan is de invoer géén zin in de taal.

We keren terug naar het voorbeeld in Figuur 3.1. Accepteert deze automaat de invoer 'acaa' als een zin in de taal $\mathfrak{L}((a|b) \cdot c \cdot a^*)$? De automaat begint natuurlijk in de starttoestand:



stap 1
Invoer: Acaa

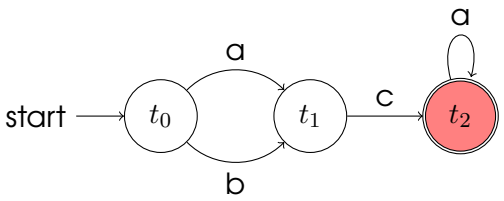
De automaat begint in de starttoestand en de het eerste symbool van de invoer wordt geïnspecteerd: een 'a'. Een 'a'-transitie is mogelijk: ga naar toestand t_1.



stap 2

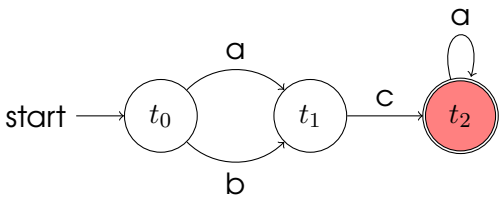
Invoer: aCaa

Het volgende symbool van de invoer wordt geïnspecteerd: een `c'. In toestand $t_1$ is er een `c'-transitie mogelijk: ga naar toestand t_2.



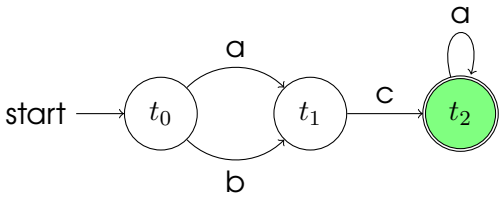
stap 3
Invoer: acCa

Toestand t_2 is een eindtoestand: `ac' is een zin in de taal. Maar er zijn nog meer symbolen in de invoer; het volgende symbool van de invoer wordt geïnspecteerd: een `a'. In toestand t_2 is er een `a'-transitie mogelijk: ga naar toestand t_2.



stap 4
Invoer: acaC

Toestand t_2 is een eindtoestand: `aca' is een zin in de taal. Maar er zijn nog meer symbolen in de invoer; het volgende symbool van de invoer wordt geïnspecteerd: een `a'. In toestand t_2 is er een `a'-transitie mogelijk: ga naar toestand t_2.



stap 5
Invoer: acaa

Toestand t_2 is een eindtoestand: `acaa' is een zin in de taal. Er is geen volgend symbool: deze automaat accepteert de invoer `acaa' als een zin in deze taal.

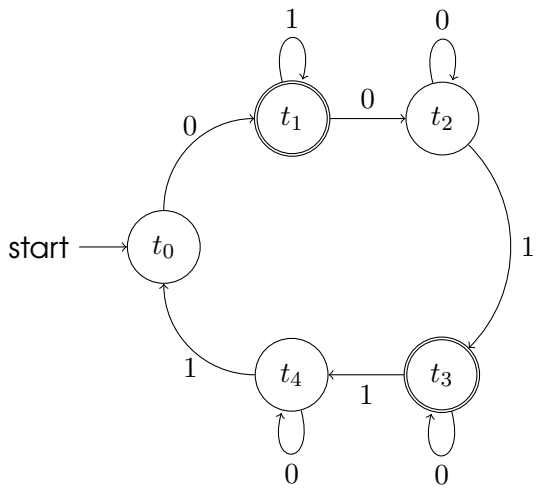
De automaat "eet" als het ware bij iedere stap het volgende symbool van de invoer en bekijkt vervolgens of het mogelijk is om een transitie te doen. Zo nee, dan is de invoer géén zin in de taal; zo ja, dan doet de automaat die transitie en gaat naar de volgende toestand.

Als alle symbolen van de invoer verwerkt zijn, dan bekijkt de automaat of hij zich in een accepterende toestand bevindt. Zo ja, dan is de invoer een zin; zo nee, dan was de invoer géén zin. Het kan voorkomen dat de automaat tijdens het verwerken in accepterende toestanden komt, dat zegt echter *niets* over het zin-zijn van de invoer: de rest van de symbolen van de invoer moeten nog verwerkt worden.

3.3 Opgaven

3.1 Gegeven alfabet $\mathbb{G}$ uit Hoofdstuk 2. Teken per deelopgave een DFA die enkel zinnen uit de aangegeven reguliere taal accepteert:

a) $\mathfrak{L}(q \cdot q \cdot q)$
b) $\mathfrak{L}(w \cdot (x|y|z) \cdot w^*)$
c) $\mathfrak{L}((k|l|m)|(a|b|c))$
d) $\mathfrak{L}(a^* \cdot b \cdot (a|b) \cdot b^*)$



Figuur 3.2: Een DFA over alfabet $\mathbb{B} = 0, 1$.

3.2 Gegeven alfabet $\mathbb{B} = 0, 1$ en de DFA uit Figuur 3.2 over dat alfabet. Accepteert deze DFA de volgende invoer als een zin? Zo ja, welke eindtoestand accepteert de invoer?

a) 010010010
b) 01001
c) 0110
d) 011
e) 010010
f) 0100
g) 0100101
h) 000000
i) 1111111111
j) 01101010101010010

3.3 Geef de reguliere expressie van de taal die DFA uit Figuur 3.2 herkent. Het alfabet is $\mathbb{B} = 0, 1$.

De antwoorden vind je in Bijlage A.3 op bladzijde 30.

4 Praktijk I – patronen

4.1 Reguliere expressies: van theorie naar praktijk

Tot nu toe hebben we de theorie van reguliere expressies besproken. In Hoofdstuk 2 zijn we begonnen met de definities van een alfabet, een zin en een taal. Vervolgens heb je gezien hoe je reguliere expressies kunt maken met behulp van de drie operatoren $\cdot$, * en $|$. Met behulp van deze reguliere expressies is het mogelijk om zinnen te genereren die aan die reguliere expressie voldoen.

In het vorige hoofdstuk zijn deterministische eindige automaten geïntroduceerd. Met behulp van de DFA's kunnen zinnen die aan een reguliere expressie voldoen, *herkend* worden. Computers gebruiken deze manier om reguliere expressies te herkennen. Dus elke keer als je reguliere expressies in de praktijk gebruikt, maak je gebruik van een DFA en de theorie achter reguliere expressies.

Reguliere expressies in de praktijk gebruiken altijd een vast alfabet, meestal de 128 ASCII tekens. Tegenwoordig wordt ook steeds vaker UTF-8 of UTF-16 ondersteund. In dit lesmateriaal zullen we alleen ASCII karakters gebruiken.

Praktische reguliere expressies kunnen opgebouwd zijn uit meer operatoren dan alleen concatenatie ($\cdot$), herhaling (*) herhaling en de keuze ($|$). Daarnaast zijn er voor de bekende operatoren vaak andere symbolen gekozen. In dit lesmateriaal gebruiken we zogenaamde PERL reguliere expressies: de syntaxis van reguliere expressies zoals (oorspronkelijk) gebruikt in de programmeertaal PERL. Tegenwoordig gebruiken vele andere programmeertalen en programma's dezelfde syntaxis.[1]

4.2 De syntax van reguliere expressies

Hieronder worden de meest gebruikte operatoren besproken met behulp van enkele voorbeelden:

. **Een willekeurig teken geef je aan met een punt.** Let op, concatenatie heeft nu *geen* symbool. Als twee reguliere expressies naast elkaar staan, betekent dat automatisch concatenatie.

[1] Waaronder de programmeertaal PHP en de tekstverwerker van Google Docs

Voorbeelden:

- `aa.` vindt alle zinnen die beginnen met twee a's en gevolgd worden door een derde teken. Let op, alle tekens uit het alfabet voldoen als dat derde teken, dus ook een spatie of een interpunctieteken.
- `.*` de hele invoer wordt hierdoor herkend.

`|` **De keuze, net zoals in de theorie**

Voorbeelden:

- `a|b` vindt alle zinnen van een a of een b.

`?` **Optionaliteit: reguliere expressie doet optioneel mee.**

Voorbeelden:

- `a?bb` vindt alle zinnen die eindigen op twee b's en wel of niet beginnen met een a; vindt zowel 'abb' als 'bb'.
- `(a.p)?en` vindt alle zinnen 'en', eventueel voorafgegaan door zinnen van de vorm `a.p`; vindt 'alpen', 'aapen', ..., 'en'.

`*` **Herhaling van nul of meer keer, net zoals in de theorie.**

Voorbeelden:

- `a*p` vindt alle zinnen beginnende met nul of meer a's en eindigend op een p; vindt 'p', 'ap', 'aap', 'aaap', ...

`+` **Herhaling van één of meer keer.**

Voorbeelden:

- `a+p` vindt alle zinnen beginnende met een of meer a's en eindigend op een p; vindt 'ap', 'aap', 'aaap', ...
- `hot(entot)+tentoonstelling` vindt 'hotentottentoonstelling', 'hotentotentottentoonstelling', 'hotentotentotentottentoonstelling', ...

`[ ]` **Definitie van een karakterklasse.** Een karakterklasse is een verzameling van karakters. Een voorbeeld: Als je gehele getallen wilt weergeven met behulp van een reguliere expressie zoals `( 1|2|3|4|5|6|7|8|9) (0|1|2|3|4|5|6|7|8|9)*`. Dit is veel werk, foutgevoelig en repetitief. Met behulp van karakterklassen kan het ook als volgt: `[1-9][0-9]*`. Dat is een stuk korter en overzichtelijker.

Een reeks geef je dus aan met een --teken. Naast een reeks kun je ook alle symbolen uit een karakterklasse opsommen, of een combinatie

van beide: alle kleine letters en de punt, komma en het vraagteken: `[a-z.,?]`.

Het is ook mogelijk om alle tekens in een karakterklasse op te nemen behalve degene die je opsomt tussen de rechte haken. Je gebruikt daarvoor het ^ -teken. Als je alle tekens wilt behalve de kleine, grote letters of het vraagteken, kun je dat aangeven met: `[^?a-zA-Z]`.

Voorbeelden:

- `(-|+)?[1-9][0-9]*` vindt alle gehele getallen, eventueel met teken.
- `[k-ma-e]+` vindt alle zinnen die de letters k, l, m, a, b, c, d of e bevatten.

`( )` **Je geeft een subexpressie of groep aan met behulp van gewone haakjes, net zoals in de theorie.**

`\` **Ontsnappingsteken, om de operatorsymbolen als karakter op te nemen**

Voorbeelden:

- `.+\.` vindt alle zinnen van minimaal twee tekens die eindigen op een punt.
- `[a-zA-Z]+\?` vindt alle laatste woorden van een vraag.

4.3 Reguliere expressies in de praktijk: regexp-websites

Met behulp van deze speciale symbolen kun je reguliere expressies opstellen van patronen die je dagelijks tegenkomt. In de opgaven ga je verschillende van dergelijke patronen beschrijven met een reguliere expressie. Je leert het best werken met reguliere expressies door ze ook uit te proberen: doen jouw reguliere expressie wel wat jij wilt? Worden er niet te veel zinnen, of juist te weinig zinnen gevonden?

Op het internet zijn verschillende websites te vinden waar je reguliere expressies uit kunt voeren, bijvoorbeeld:

- http://regexpal.com/,
- http://myregexp.com/ of
- http://www.myregextester.com/.

Controleer bij het maken van de vragen je antwoorden met behulp van een van deze regexp-websites.

4.4 Opgaven

4.1 Geef praktische reguliere expressies voor de volgende patronen. Controleer of je reguliere expressies ook werken en doen wat je denkt dat ze doen met behulp van de regexp-websites. Controleer ook of zinnen die niet geaccepteerd worden inderdaad niet geaccepteerd worden.

a) Gehele getallen met *verplicht* teken
b) Kommagetallen met cijfers voor en achter de komma
c) Wetenschappelijke getallen beschreven als een kommagetal met (optioneel) daarachter een 'e' of 'E' en de exponent, voorbeelden: $12.2342354e-12, 0.00023E44, 12.3$.
d) Nederlandse vaste telefoonnummers met een streepje tussen het netnummer en het abonneenummer.
e) Nederlandse vaste telefoonnummers met internationale annotatie, voorbeeld: +31(0)20-1234567.
f) Datums weergegeven met formaat "dd-mm-jjjj'
g) Een opentag van een HTML element, zoals <body> of
h) Een sluittag van een HTML element, zoals </body> of
i) Een opentag van een HTML element met attributen, zoals <body bgcolor="blackïd="nr2"> of
j) Een heel HTML-element, opentag met attributen, sluittag en inhoud.
k) Een URL, je mag "http://" weglaten uit je reguliere expressie.
l) Een e-mailadres. In een e-mailadres kunnen voor het @-teken kleine letters, grote letters, punten en plus-tekens voorkomen. Na het @-teken volgt een domeinnaam.
m) Een klasnaam op je school.
n) Een (Nederlandse) zin. Een zin begint met een woord met een hoofdletter en eindigt met een punt. Daartussen staan nul of meer losse woorden gescheiden door een spatie.
o) Een rij van een tabel: gehele getallen gescheiden door een komma.
p) Een rij van een tabel: waarden tussen dubbele aanhalingstekens gescheiden door een komma. Deze waarden kunnen naast gewone lettertekens ook een spatie bevatten.
q) Een rij van een tabel: waarden tussen dubbele aanhalingstekens of gehele getallen, gescheiden door een komma. Deze waarden kunnen naast gewone lettertekens ook een spatie bevatten.
r) Theoretische reguliere expressies over alfabet $\mathbb{B} = 0, 1$ (zoals in Hoofdstuk 2). Gebruik voor de concatenatieoperator een gewone punt.

De antwoorden vind je in Bijlage A.4 op bladzijde 31.

5 Praktijk II – patronen en vervangen

5.1 Patronen: een uitbreiding

In het vorige hoofdstuk heb je praktische reguliere expressies gemaakt en gebruikt. Je leerde verschillende extra operatoren kennen en je zag ook hoe de al bekende operatoren er in de praktijk uitzien. In deze paragraaf komen daar nog een aantal operatoren bij.

5.1.1 Voorgedefinieerde karakterklassen

Zoals je weet kun je een karakterreeks maken met behulp van vierkante haakjes. Zo geef je de kleine letters aan met `[a-z]`. Sommige karakterreeksen zul je vaker gebruiken dan andere reeksen. Sterker nog, sommige karakterreeksen worden zo vaak gebruikt dat er speciale afkortingen voor bestaan:

- `\d` De cijfers (0 tot en met 9) , dit is dus een afkorting voor: `[0-9]`
- `\D` Alles behalve de cijfers , dit is dus een afkorting voor: `[^0-9]`
- `\w` Alle lettertekens, de letters en de cijfers , dit is dus een afkorting voor: `[a-zA-Z0-9]`
- `\W` Alles behalve lettertekens, dus alles behalve letters en cijfers , dit is dus een afkorting voor: `[^a-zA-Z0-9]`
- `\s` Witruimte: de spatie, tab en nieuwe regel , dit is dus een afkorting voor: `[ \t\n\r\f]`
- `\S` Alle behalve witruimte , dit is dus een afkorting voor: `[^ \t\n\r\f]`

Voorbeelden: De gehele getallen kun je nu aanduiden met: `(\+|-)?[1-9]\d*`. Een zin kun je opschrijven als: `[A-Z]\w*( \w+)*\.`.

5.1.2 Begrensde herhaling

In het vorige hoofdstuk zijn drie vormen van herhaling besproken:

- nul of een keer, ook wel optionaliteit genoemd, geef je aan met `?`;
- een of meer keer geef je aan met `+` en
- nul of meer keer geef je aan met `*`

Soms wil je aangeven dat een bepaald subpatroon zich 3, 4, of 5 keer mag voordoen, of precies 12 keer. Met de operatoren die je nu kent kun je dat alleen maar aangeven met concatenatie en keuze; dergelijke reguliere expressies worden met de operatoren die je tot nu toe hebt gezien erg lang en onoverzichtelijk. Ook voor de zogenaamde begrensde herhaling bestaat een operator: `{o,b}`.

Met `{o,b}` geef je aan dat je een reguliere (deel)expressie minimaal *o* keer wilt herhalen en maximaal *b* keer. Met *o* en *b* geef je dus respectievelijk de *o*nder- en *b*ovengrens van de herhaling aan; *o* en *b* zijn positieve gehele getallen. In de opgaven van het vorige hoofdstuk werd je gevraagd naar een reguliere expressie voor een telefoonnummer. Telefoonnummers bestaan uit een drie-cijferig netnummer en een zeven-cijferig abonneenummer. De oplossing toen was: `[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9][0-9]`. Je kunt dat nu dus opschrijven als: `\d{3,3}-\d{7,7}`, een stuk korter en overzichtelijker.

Een ander voorbeeld uit het vorige hoofdstuk was een patroon voor een domeinnaam. Zoals je weet zijn er tweeletterige top-level domains, zoals *.nl* en drieletterige zoals *.com*. Met behulp van de begrensde herhaling kun je er een volgende reguliere expressie opschrijven domeinnamen: `(\w+)?\.\w+\.[a-zA-Z]{2,3}`.

5.1.3 Het begin en het einde van een regel

Als je een gestructureerd bestand bekijkt, bijvoorbeeld en log van een programma of een configuratiebestand, dan zie je vaak dat bepaalde patronen vooraan of juist achteraan de regel voorkomen. Stel je een eenvoudig TODO-lijstje voor: elke regel begint met de datum waarop het item klaar moet zijn. Hoe je een datum met een reguliere expressie kunt beschrijven, weet je, maar hoe geef je aan dat het *vooraan* de regel moet staan?

Daarvoor gebruik je de operator `^`. Een datum met patroon dd-mm-jjjj aan het begin van de regel geef je dan aan met de reguliere expressie: `^\d{2,2}-\d{2,2}-\d{4,4}`.

Ook voor het einde van een regel is er een operator: `$`. Een datum aan het einde van een regel kun je met de reguliere expressie `\d{2,2}-\d{2,2}-\d{4,4}$`

aan. Het hoedje staat dus vooraan de reguliere expressie, omdat het het *begin* van de regel aanduidt. Het dollarteken staat juist aan het *einde* van een reguliere expressie omdat het het einde van de regel aangeeft.

5.2 Vervangen

Tot nu toe heb je enkel patronen geschreven om ermee te zoeken. Reguliere expressies zijn ook krachtige hulpmiddelen bij het automatisch vervangen van gestructureerde tekst. Je kunt met behulp van de welbekende ronde haakjes (en) groepen maken in je reguliere expressies. De deelzinnen die door dergelijke groepen – dat zijn ook deelexpressies van de hele reguliere expressie – worden herkend, kun je later weer oproepen.

Een voorbeeld: stel je hebt een bestand met daarin datums in het patroon dd-mm-jjjj en je wilt juist datums in van de vorm jj/mm/dd hebben. Met behulp van reguliere expressies kun je de datums in dit bestand automatisch omzetten. Je begint met het schrijven van een reguliere expressie die de bestaande datums herkend:

```
\d{2,2}-\d{2,2}-\d{4,4}
```

We zijn geïnteresseerd in de dag, de maand en de laatste twee cijfers van het jaar. We maken van deze *interessante* deelpatronen groepen:

```
(\d{2,2})-(\d{2,2})-\d{2,2}(\d{2,2})
```

Deze groepen kun je vervolgens aanwijzen, van links naar rechts, met: `$1`, `$2` en `$3`. De hele zin die wordt herkend noemen we `$0`. Je kunt deze namen voor de groepen, de deelzinnen, gebruiken in het "vervangen"-vakje. In dit geval ziet de expressie in het vervangen-vakje er als volgt uit:

```
$3/$2/$1
```

Probeer zelf maar eens uit in Google Docs. Veel implementaties van reguliere expressies kennen niet meer dan tien groepsnamen ($0 tot en met $9). Let daarop bij het gebruik ervan.

5.3 Opgaven

5.1 Geef praktische reguliere expressies voor de volgende patronen. Controleer of je reguliere expressies ook werken met behulp van een van de regexp-websites opgesomd in paragraaf 4.3.

a) IP-adressen. IP-adressen bestaan uit vier getallen van 0 tot en met 255 gescheiden door een punt, bijvoorbeeld: 123.45.6.255.
b) Postcodes: twee letters, een spatie en vier cijfers.
c) Nieuwe Nederlandse kentekens: twee cijfers, een streepje, drie letters, een streepje en een cijfer. Met uitzonderingen hoef je geen rekening te houden.
d) Punten op je rapport: getallen tussen 1 en 10 met precies een cijfer achter de komma.
e) Spaties voor leestekens aan het einde van een regel.
f) LaTeX-commando's: Een commando in LaTeX ziet er als volgt uit: \commando{inhoud van het commando}, bijvoorbeeld: \textbf{vet af te drukken} om tekst **vet af te drukken**.
g) Een SQL-query moet altijd eindigen met een puntkomma.

5.2 Geef praktische reguliere expressies en vervangingsexpressies voor de volgende problemen. Controleer of je expressies ook werken met behulp van een van de regexp-websites opgesomd in paragraaf 4.3.

a) Je hebt een HTML bestand waarin je alle B-elementen automatisch wilt vervangen door EM-elementen.
b) Een log van een IRC (=chat) sessie zien er als volgt uit: op elke regel staat een bericht volgens het volgende patroon: `[23:14] <Jaap95> idd dat is l33t` Met andere woorden: de tijd tussen rechte haken, gevolgd door een spatie en een nickname tussen puntige haken, een spatie en vervolgens het gezonden bericht. Je wilt dit logboek "leesbaarder" maken door elke regel te vervangen door, voorbeeld, `Jaap95 sprak op 23:14 als volgt: "idd dat is l33t".` Met andere woorden: de nickname gevolgd door de tekst " sprak op ", de tijd, de tekst " als volgt: "', het bericht en afgesloten door "".".
c) Een stuk uit het logboek van mijn firewall:

```
Mar 13 19:17:25 IN=eth0 OUT= SRC=118.165.68.226 DST=192.168.1.70 ID=46093 PROTO=UDP SPT=22187 DPT=6881 LEN=273
Mar 13 19:17:42 IN=eth0 SRC=183.191.39.154 DST=192.168.1.70 LEN=293  TTL=111 ID=56453 PROTO=UDP DPT=6881 LEN=273
Mar 13 19:17:49 68:76:cc:89:08:00 SRC=114.47.88.149 DST=192.168.1.70 PREC=0x00 PROTO=UDP SPT=25377 DPT=6881 LEN=
```

Dit is ingewikkeld. Maak dit eenvoudiger leesbaar door enkel de datum, de tijd in uren en minuten, het protocol (PROTO=*protocol*), de bron (SRC=*bron*) en de bestemming (DST=*bestemming*) per regel weer te geven.

De antwoorden vind je in Bijlage A.5 op bladzijde 32.

6 Gemengde opgaven

6.1 Geef per deelopgave aan of het wel of geen alfabet is en, indien het geen alfabet betreft, leg uit waarom niet:

a) $\mathbb{A}$ = 'a', 'b', 'c', 'd', '1', '1', '3', 'e'
b) $\mathbb{B}$ = '3', '4', ' ', '⋆', '-', '5'
c) $\mathbb{C}$ = '+'

6.2 Gegeven alfabet $\mathbb{G}$ = 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'. Geef voor de volgende reguliere expressies, indien mogelijk, vier verschillende zinnen die door die reguliere expressie gegenereerd kunnen worden.

a) $a \cdot (b|c|d)^* \cdot a \cdot (b|c|d)^*$
b) $1 \cdot 2 \cdot 3 \cdot 4^* \cdot 0 \cdot 0^* \cdot 0^*$
c) $(k|w|0)^* \cdot (w|k^*|0) \cdot (k|w|0)$
d) $(u|u^*|u)^* \cdot u \cdot u \cdot (r|t^*) \cdot (t|r^*)^*$

6.3 Gegeven alfabet $\mathbb{G}$ = 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'. Schrijf per deelopgave een reguliere expressie die de zinnen in die deelopgave genereert.

Met '...' wordt aangegeven dat er nog veel meer zinnen met een vergelijkbaar patroon zijn; het eenvoudigweg opsommen van alle alternatieven in de deelopgave met behulp van de keuzeoperator is dan *geen* goede oplossing.

a) 'jas', 'jan', 'jaap'
b) 'a0001z', 'a9z', 'a902347z', 'a038402835707z', ...
c) 'ppq00q1', '2', 'qpp0q3aaab', '0002aaa', 'pppppp1abbaaab', 'qq3bbbba', ...

6.4 Gegeven is alfabet $\mathbb{Q}$ = '-', '+'. Schrijf per deelopgave een reguliere expressie die de zinnen in die deelopgave genereert.

Met '...' wordt aangegeven dat er nog veel meer zinnen met een vergelijkbaar patroon zijn; het eenvoudigweg opsommen van alle alternatieven in de deelopgave met behulp van de keuzeoperator is dan *geen* goede oplossing.

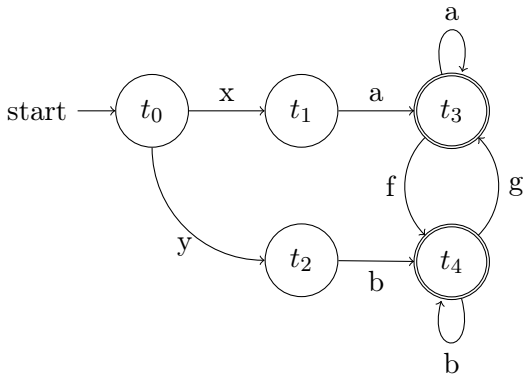
a) '', '-', '--', '---', ..., '+', '++', '+++', ...
b) '-+', '-+-+', '-+-+-+', ...

c) '++-', '+++-', '++++-', ...
d) '--', '----', '+--+', '+---', '-+----', '-+-+++--++---', ...

6.5 Gegeven alfabet $\mathbb{G}$ = 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'. Teken per deelopgave een DFA die zinnen uit de aangegeven reguliere taal accepteert

a) $\mathfrak{L}((a|b|c) \cdot 0 \cdot 0^* \cdot (p|q|r) \cdot r^*)$
b) $\mathfrak{L}(k^* \cdot l \cdot m^*)$
c) $\mathfrak{L}((1|2|3) \cdot (4 \cdot (1|2|3))^*)$
d) $\mathfrak{L}(x \cdot (y \cdot z^* \cdot y)^*)$

6.6 Gegeven is alfabet $\mathbb{Q}$ = '-', '+' en de DFA hieronder over dat alfabet.



accepteert deze DFA de volgende invoer als een zin? Zo ja, welke eindtoestand accepteert de invoer?

a) xafgaaa
b) ybbbg
c) xaafbyxa
d) ybgaafbbgafb
e) xafgfgfgfgfgfbbbbb
f) xafgfgfgfafgfbbbbb
g) ybgafgfbbga
h) xagfbgfaaaa

6.7 Geef een reguliere expressie van de taal die de DFA uit de vorige opgave herkent.

vanaf hier opgaven over praktische reguliere expressies!

6.8 Geef praktische reguliere expresies voor de volgende patronen.

a) Gehele getallen van -99 tot en met 99
b) Woorden met een koppelteken, zoals zwart-wit, cultureel-historisch of extreem-rechts. Let op, in een woord komen geen getallen voor.

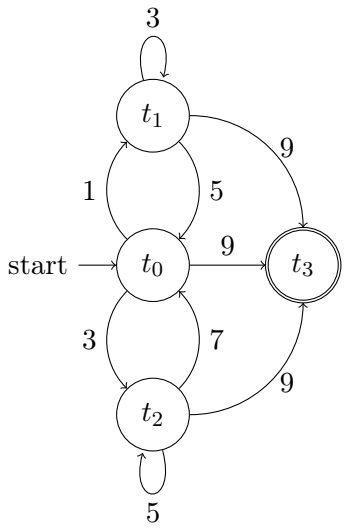
c) Een wachtwoord waarvoor geldt dat het bestaat uit letters, cijfers en/of de tekens -, #, &. Een wachtwoord is minimaal 7 tekens lang en maximaal 12.
d) ISB nummers. Een ISB nummer bestaat uit een cijfer, een streepje, twee cijfers, een streepje, zes cijfers, een streepje en tot slot weer een cijfer.
e) Een Microsoft DOS/Windows pad. Zo'n pad begint met een schijfnaam, een enkele hoofdletter, gevolgd door een dubbele punt en een backslash. Elke directory en de bestandsnaam zijn gescheiden door een backslash. Directorienamen en bestandsnamen bestaan uit kleine letters, grote letters, cijfers en punten. Voorbeeld:
`C:\pad\naar\bestand.extentie`.
f) Hexadecimale getallen, eventueel negatief. Een hexadecimaal getal bestaat uit de cijfers 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

De antwoorden vind je in Bijlage A.6 op bladzijde 33.

7 Oefentoets

naam _____ klas ___ datum _____ leerlingnummer _____

Oefentoets reguliere expressies voor vwo 6



7.1 4 punten Gegeven is alfabet $\mathbb{O}$ = '1', '3', '5', '7', '9' en de DFA hierboven over dat alfabet. Accepteert deze DFA de volgende invoer als een zin?

a) 13353579
b) 9
c) 37153715355739
d) 135579
e) 1357559
f) 1537133
g) 3715133519
h) 1515151539

7.2 4 punten Geef de reguliere expressie van de taal die de DFA hierboven herkent.

7.3 3 punten Gegeven alfabet $\mathbb{P}$ = '+', '=', '-'. Schrijf een reguliere expressie die per deelopgave de gegeven zinnen beschrijven. Het symbool '...' geeft aan dat er nog veel meer vergelijkbare zinnen zijn die ook door de reguliere expressie herkent moeten worden.

a) +--+, +--=-+, ++, +=====+, ...

b) =, -=+, --=+, ---=+++++++, ...
c) =+, -+, ++

7.4 9 punten Gegeven alfabet $\mathbb{Q}$ = '2', '4', '6', '8', 'T'. Geef, indien mogelijk, vier verschillende zinnen per taal en teken vervolgens een DFA die zinnen uit die taal herkent.

a) (3 punten) $\mathfrak{L}((2|4|6|8)^* \cdot T \cdot (2|4|6|8)^*)$
b) (6 punten) $\mathfrak{L}(((2(6 \cdot 8)^* \cdot T)|(4 \cdot (8 \cdot 6)^* \cdot T))^* \cdot ((2 \cdot 6)|(4 \cdot 8)|(2 \cdot (6 \cdot 8)^*)|(4 \cdot (8 \cdot 6)^*)))$

---

vanaf hier gaan de opgaven over praktische reguliere expressies!

7.5 15 punten Geef praktische reguliere expressies voor de volgende patronen:

a) Wetenschappelijke getallen beschreven als een kommagetal met (optioneel) daarachter een 'e' of 'E' en de exponent, voorbeelden: $12.2342354e-12, 0.00023E44, 12.3$.
b) Datums weergegeven met formaat "dd-mm-jjjj"
c) Punten op je rapport: getallen tussen 1 en 10 met precies een cijfer achter de komma.
d) Regels uit een chatlog die beginnen met de tijd in uren en minuten (zoals 12:34 of 02:54) en eindigen met het symbool '|'. Daartussen staan willekeurige tekens, waaronder ook het symbool '|' mag voorkomen.
e) Een heel HTML-element, opentag met attributen, sluittag en inhoud. Voorbeelden van een HTML elementen: ga naar H2 of Vet!.

7.6 5 punten Gegeven is de volgende chatlog:

```
   Dirja@hotmail.com (02:45) -- What do you mean?
marcus87@hotmail.com (02:45) -- What I said ealr, y'know
Dirja@hotmail.com (02:47) -- Oh, my fault
k34ad@gmail.com (02:48) -- FAIL!!!
```

Beantwoorde de volgende vragen:

a) Schrijf alle zinnen op die door de reguliere expressie `\w+\.\w+.?\(\d{1,2}` worden herkent in bovenstaande chatlog. Geef steeds een zo lang mogelijke zin. (Let op, we hebben het hier over een zin zoals gedefiniëerd in Hoofdstuk 2 van het lesmateriaal, niet over een Nederlandse zin ...)
b) Geef reguliere expressie en vervangingsexpressie om de regels uit bovenstaande chatlog om te zetten in het formaat: tijd (gebruiker): "bericht".

De antwoorden vind je in Bijlage A.7 op bladzijde 35.

A Antwoorden

A.1 Antwoorden Hoofdstuk 1

1. a) Het patroon bestaat uit drie gedeelten: een of twee cijfers, een '-', een of twee cijfers, een '-' en tot slot twee cijfers.

   b) Het patroon bestaat wederom uit drie gedeelten: twee cijfers, een '-', twee cijfers, een '-' en vier cijfers.

   c) De datum bestaat nu enkele aaneengesloten kleine lettertekens, een spatie, een een of twee cijfers, een spatie, enkele aaneengesloten kleine lettertekens, een spatie en vier cijfers.

   d) De datum bestaat uit precies acht cijfers.

2. a) Een cijfer van '1' tot '9' gevolgd door nul of meer cijfers van '0' tot '9'.

   b) Een cijfer van '1' tot '9' gevolgd door nul of meer cijfers van '0' tot '9', of enkel een '0'. Het geheel kan optioneel voorafgegaan worden door een '-'.

   c) Een of meer cijfers van '0' tot '9', optioneel gevolgd door een '.' en een of meer cijfers van '0' tot '9', optioneel gevolgd door een 'e' of 'E' gevolgd door een geheel getal (zie opgave 1.2.b). Het geheel kan optioneel voorafgegaan worden door een '-'.

   d) Twee gehele getallen (zie opgave 2.1.b) gescheiden door een streep. Eventueel voorafgegaan door een '-'.

3. Een volledige beschrijving van het patroon in tekst te geven is lastig: er bestaan nogal wat uitzonderingen en speciale gevallen. Een goede poging:

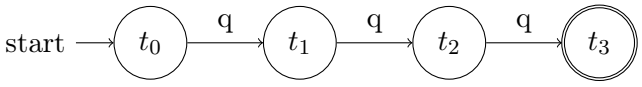
   - een cel is: nul of meer aaneengesloten letter- of cijfertekens óf een stukje tekst omsloten door '",
   - een rij is: nul of meer cellen gescheiden door een ','
   - een tabel is: nul of meer rijen gescheiden door een nieuwe-regel-teken

A.2 Antwoorden Hoofdstuk 2

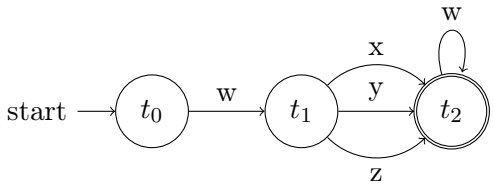
1. a) Er is er maar een : ‘qqq’
   b) ‘wxw;, ‘wyww’, ‘wz’, ‘wxwwwwwww’, ‘wywwwwwwwwwwww-wwwwwwwwwwwwwww’
   c) ‘k’, ‘b’, ‘c’, ‘a’, ‘m’
   d) ‘ddddddddddddddddddd’, ‘dd00d00000deee11’, ‘d0eeeeeeeeee111111111’, ‘00000000111111111111’, ‘1111111111111’
   e) ‘a’, ‘b’, ‘aaaaaaaaaaaaaaaabbbbbb’, ‘ab’, ‘aabbb’
2. a) $a \cdot a \cdot (p|k|s|l)$
   b) $(1|2|3|4|5|6|7|8|9) \cdot (0|1|2|3|4|5|6|7|8|9)$
   c) $a \cdot h \cdot a^*$
3. a) $0^* \cdot 1 \cdot 1 \cdot 0^*$
   b) 1^*
   c) $(0|1)^*$
   d) $0|1|(1 \cdot (0|1))|(1 \cdot (0|1) \cdot (0|1))$
4. a) alfabet: ’0’, ’1’, ’ ’, ’+’, ’-’
   reguliere expressie: $(0|1) \cdot (0|1) \cdot$ ’ ’ $\cdot (+|-) \cdot (0|1) \cdot (0|1) \cdot (0|1)$
   b) alfabet: ’0’, ’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’-’
   reguliere expressie: $0 \cdot 6 \cdot - \cdot (0|1|2|3|4|5|6|7|8|9) \cdot (0|1|2|3|4|5|6|7|8|9)$
   c) alfabet: ’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’K’, ’Q’, ’J’, ’♡’, ’♢’, ’♠’, ’♣’
   reguliere expressie: $((1|2|3|4|5|6|7|8|9|(1 \cdot 0)|K|Q|J) \cdot (♡|♢|♠|♣))|(♡|♢|♠|♣)$

A.3 Antwoorden Hoofdstuk 3

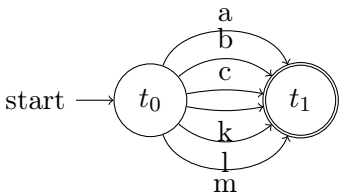
1. a

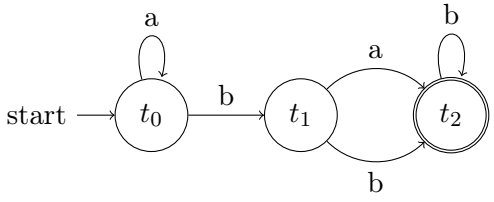


b



c





d

2. a nee

   b ja, t_3

   c nee

   d ja, t_1

   e ja, t_3

   f nee

   g nee

   h nee

   i ja, t_1

   j nee

3. Deze vraag lijkt vrij lastig, maar als je goed naar de automaat kijkt zie je twee verschillende accepterende toestanden. Vanaf t_0 kun je die beschrijven met de reguliere expressies: $t_1 = 0{\cdot}1^*$ en $t_3 = 0{\cdot}1^*{\cdot}0{\cdot}0^*{\cdot}1{\cdot}0^*$.

   Verder zie je dat de automaat cyclisch is, dat wil zeggen dat je oneindig vaak de automaat kunt doorlopen en steeds weer bij de starttoestand t_0 uit komt. Met andere woorden, de hele automaat kan 0 of meer keer worden doorlopen en ook dát kunnen we min of meer aangeven met een reguliere deelexpressie: $(0 \cdot 1^* \cdot 0 \cdot 0^* \cdot 1 \cdot 0^* \cdot 1 \cdot 0^* \cdot 1)^*$. Natuurlijk is t_0 hier geen eindtoestand.

   We kunnen deze drie reguliere deelexpressies combineren tot de reguliere expressie die heel de automaat beschrijft:

   $$(0 \cdot 1^* \cdot 0 \cdot 0^* \cdot 1 \cdot 0^* \cdot 1 \cdot 0^* \cdot 1)^* \cdot ((0 \cdot 1^*)|(0 \cdot 1^* \cdot 0 \cdot 0^* \cdot 1 \cdot 0^*))$$

A.4 Antwoorden Hoofdstuk 4

1. a) `(+|-)([1-9][0-9]*|0)`

   b) `([1-9][0-9]*|0),[0-9]+`

   c) `([1-9][0-9]*|0),[0-9]+((e|E)(+|-)[1-9][0-9]*)`

   d) `[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9][0-9]`

   e) `\+31\(0\)[0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9][0-9]`

f) `[0-9][0-9]-[0-9][0-9]-[1-9][0-9][0-9][0-9]`
(Let op, met dit patroon worden ook niet bestaande datums beschreven)

g) `<[a-zA-Z]+>`

h) `</[a-zA-Z]+>`

i) `<[a-zA-Z]+( [a-zA-Z]+=".*")*>`

j) `<([a-zA-Z]+( [a-zA-Z]+=".*")*>.*</[a-zA-Z]+)>`

k) `[a-zA-Z.+]+@[a-zA-Z]+\.[a-zA-Z][a-zA-Z][a-zA-Z]?`
(Let op, meer tekens zijn toegestaan in een URL, maar zonder nadere informatie voldoet een oplossing als deze)

l) `[a-zA-Z.+]+[a-zA-Z]+\.[a-zA-Z][a-zA-Z][a-zA-Z]?`

m) `(m|h|v)[1-6][a-z]`
(Let op, dit patroon levert ook niet bestaande klascodes op!)
beter: `m(1-4)[a-z]|h[1-5][a-z]|v[1-6][a-z]`
(maar ook dit patroon levert nog veel niet bestaande klascodes op)

n) `[A-Z][a-z]+( [a-z]+)*\.`
(Dit is een letterlijke uitwerking van de opgaven: het eerste woord bevat de enige hoofdletter in de zin, de rest bestaat uit spaties gescheiden woorden, en de punt volgt direct na het laatste woord.)

o) `([1-9][0-9]*|0)(,([1-9][0-9]*|0))*`

p) `("[a-zA-Z0-9 ]*")(,("[a-zA-Z0-9]*"))*`

q) `(("[a-zA-Z0-9]*")|([1-9][0-9]*|0))(,(("[a-zA-Z0-9]*")|([1-9][0-9]*|0)))*`

r) `[()*|.01]+`
(deze reguliere expressie herkent ook niet-theoretische reguliere expressies over B)

A.5 Antwoorden Hoofdstuk 5

1. a) `(0|[1-9]\d{0,2})(\.(0|[1-9]\d{0,2})){3,3}`

   b) `[a-zA-Z]{2,2} \d{4,4}`

   c) `\d{2,2}-[a-zA-Z]{3,3}-\d`

   d) `([1-9]|10),\d`

   e) `\\[a-zA-Z]+\.*\`

   f) `\\[a-zA-Z]+\{.*\}`

2. a) reguliere expressie: `<(b|B)>(.*)</(b|B)>`
   vervangingsexpressie: `<em>$1</em>`

b) reguliere expressie: `\[(\d{2,2}:\d{2,2})\] <(\w+)> (.+)$`
vervangingsexpressie: `$2 sprak op $1 als volgt:  "$3"`

c) reguliere expressie: `((Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec) \d{1,2}) (\d{2,2}:\d{2,2}) :\d{2,2}.*SRC=( (0|[1-9]\d{0,2}) (\.  (0|[1-9]\d{0,2}) ) {3,3} ).*DST=( (0|[1-9]\d{0,2}) (\. (0|[1-9]\d{0,2}) ) {3,3} ).*PROTO=([A-Z]+).*$`
vervangingsexpressie: `$1 $3 bron=$4 bestemming=$5 protocol=$6`
(Let op! de maand wordt ook in een groep vervangen, en dat is groep nummer 2. Die wil je overslaan bij het vervangen)

A.6 Antwoorden gemengde opgaven

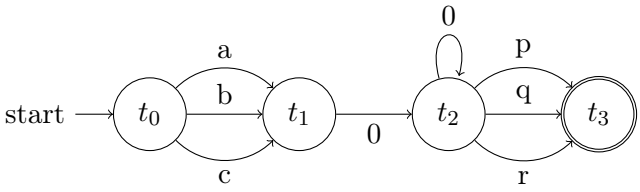
1. a Nee, het symbool '1' komt twee keer voor en een alfabet is een unieke verzameling symbolen.
   b Ja.
   c Ja.

2. a 'abbbbbbbbbab', 'aa', 'acbcbcbcdbdbcba', 'aacddddddddddddb'
   b '1230', '1123444444000000000000', '123000000000', '12344444440'
   c '0', 'kkk00', '000000000', '0wk'
   d 'uu', 'uuuuutttrrr', 'uuurtrtrtrt', 'uuutttrrr'

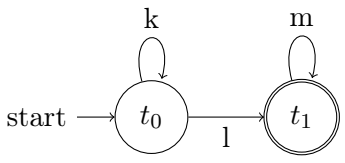
3. a $j \cdot a \cdot (s|n|(a \cdot p))$
   b $a \cdot (0|1|3|4|5|6|7|8|9) \cdot (0|1|3|4|5|6|7|8|9)^* \cdot z$
   c $(p|q|0)^* \cdot (1|2|3) \cdot (a|b)^*$

4. a $(-^*)|(+^*)$
   b $- \cdot + \cdot (- \cdot +)^*$
   c $+ \cdot + \cdot +^* \cdot -$
   d $(-|+)^* \cdot - \cdot - \cdot (-|+)^*$

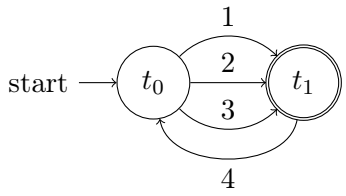
5. a



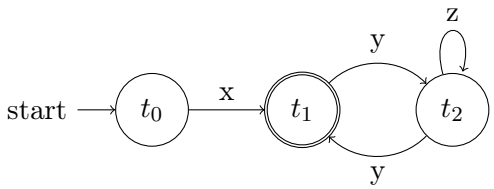
b



c



d



6. a Ja, t_3.

   b Ja, t_3.

   c Neen.

   d Ja, t_4.

   e Ja, t_4.

   f Neen.

   g Ja, t_3.

   h Neen.

7. $(x \cdot a \cdot (a^* \cdot f \cdot b^* \cdot g)^* \cdot (a^* | (a^* \cdot f \cdot b^*))) | (y \cdot b \cdot (b^* \cdot g \cdot a^* \cdot f)^* \cdot (b^* | (b^* \cdot g \cdot a^3)))$

   Als je naar de DFA van opgave 6.6 kijkt, zie je dat er twee mogelijke paden in de automaat zijn: beginnende met $x \cdot a$ of met $y \cdot b$. De weg van $x \cdot a$ verder uitwerkend, zien we dat er in t_3 een cycle is: die kan nul of meer keer doorlopen worden en dan zijn we weer terug bij t_3 $((a^* \cdot f \cdot b^* \cdot g)^*)$.

   Vanuit t_3 kunnen we naar de eindtoestand t_3 met a^* (inclusief geen enkele a!) en naar t_4 via $a^* \cdot f \cdot b^*$. Dus, in tekst geschreven: begin · cycle · einde. De weg via $y \cdot b$ is vergelijkbaar.

8. a `-?(0|[1-9]\d?)`

   b `[a-zA-Z]+-[a-zA-Z]+`

   c `[a-zA-Z0-9-#&]{7,12}`

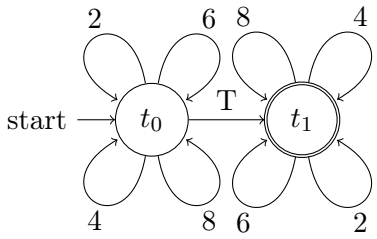
   d `\d-\d\d-\d\d\d\d\d\d−\d`

   e `[A-Z]:(\\[a-zA-Z0-9.]+)*\\[a-zA-Z0-9.]+`

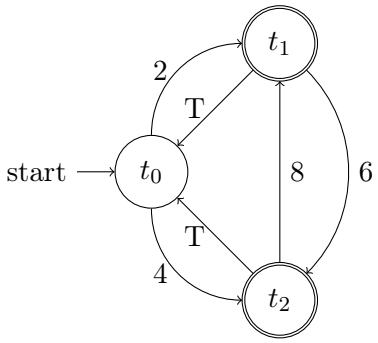
   f `-?([1-9A-F][0-9A-F]*|0)`

A.7 Antwoorden Oefentoets

1. a ja
   b ja
   c ja
   d nee
   e nee
   f nee
   g ja
   h ja
2. $((1 \cdot 3^* \cdot 5)|(3 \cdot 5^* \cdot 7))^* \cdot ((1 \cdot 3^* \cdot 9)|(3 \cdot 5^* \cdot 9)|9)$
3. a $+ \cdot -^* \cdot =^* \cdot -^* \cdot +$ óf $+ \cdot (-|=)^* \cdot +$
   b $-^* \cdot = \cdot +^*$
   c $(= | - |+) \cdot +$
4. a



   b



5. a `(-|\+)?(0|[1-9]\d*)\.\d+( (e|E)(-|\+)?[1-9]\d*)?`
   b `\d\d−\d\d−\d{4,4}`
   c `([1-9]|10)\.\d`
   d `\d\d: \d\d. * \|$`
   e `(<[a-zA-Z]+( [a-zA-Z]+=".*")*>)|(</[a-zA-Z]+>)`
6. (a) “`hotmail.com (02`” (drie keer) en “`gmail.com (02`”

(b) reguliere expressie:
`(\w+@\w+\.\w{2,3}) \((\d\d:\d\d)\) --(.*)$`
vervangingsexpressie:
`$2 ($1):"$3"`