Taal- en spraaktechnologie 2011

Opdrachten Week 1

27 april 2011

1

Voor deze opdracht bouw je een automaat: een HERKENNER voor Nederlandse éénlettergrepige woorden.

Op de website vind je als startpunt een bestand met 5890 woorden die aan het volgende patroon voldoen: nul of meer medeklinkers, één of meer klinkers, nul of meer medeklinkers. Sommige van die woorden bestaan inderdaad uit één lettergreep (`aait`), andere niet (`hiaat`). Je taak is een automaat te bouwen die de éénlettergrepige woorden accepteert.

In het hoorcollege heb je gezien dat je een automaat voor een grote taak kan maken, door automaten voor kleinere taken met elkaar te combineren. Enkele operaties die je daarvoor kan gebruiken staan hieronder nog eens bij elkaar in §1.1. De syntaxis is die van de FSA Utilities, een handig programma om met Finite State Automata te werken. Dat programma is geïnstalleerd in de leerzalen. Opeenvolging, keuze, en herhaling zijn primitieve operaties; overige nuttige operaties laten zich makkelijk definiëren in termen van die primitieve operaties.

Om je lettergreepherkenner te bouwen, kan je je eigen kleine machienetjes een naam geven. Hieronder een paar voorbeelden om je op weg te helpen.

```
macro(letter,a..z).
macro(klinker,{a,e,i,o,u}).
macro(medeklinker,letter - klinker).
macro(syll,[onset^,rhyme]).
macro(rhyme,[nucleus,coda^]).
```

Hoe zou je `nucleus` willen definiëren? `klinker+` (één of meer klinkers) is niet nauwkeurig genoeg: dan zouden zowel *iaa* als *aai* goede lettergreepkernen zijn.

Probeer de deeltaken zo op te splitsen dat je goede generalisaties uit kan drukken. Misschien is het een goed idee om korte klinkers, lange, en medeklinkers te onderscheiden. Over de onset en de coda hoef je je niet teveel zorgen te maken — het gaat er bij deze opdracht alleen om de eenlettergrepige woorden goed te identificeren!

In de testset zitten ook 'vreemde' woorden, die niet aan het Nederlandse lettergreeppatroon voldoen. Het is verstandig die woorden als uitzonderingen te behandelen. Maar ook in die uitzonderingen kan een systematiek zitten.

Maak een tekening! Om greep op de materie te krijgen, kan het nuttig zijn een diagram te maken van de overgangen van je automaat (of van de componenten), zoals we die in het hoorcollege hebben gezien. Dan kan je vervolgens de reguliere expressie bedenken die bij het overgangsdiagram past. (Met de FSA Utilities kan je die diagrammen overigens automatisch produceren. Demo tijdens het werkcollege).

1.1 Reguliere expressies

Een alfabetsymbool staat voor zichzelf, `[]` is het lege rijtje.

EXPRESSIE	BETEKENIS
`[A,B,...]`	opeenvolging van rijtjes die aan patroon `A`, `B`, ... voldoen
`{A,B,...}`	*keuze* uit rijtjes die aan patroon `A`, `B`, ... voldoen
`A*`	*herhaling* van nul of meer rijtjes die aan patroon `A` voldoen
`?`	een willekeurig alfabetsymbool
`a..z`	een symbool uit het alfabetisch geordende rijtje `a` tot `z`
`A^`	misschien een rijtje uit taal `A`, optionaliteit: `{[],A}`
`A-B`	verschil: verwijder alle rijtjes van taal `B` uit taal `A`
`A+`	minstens één rijtje uit taal `A`: `A* - []` of `[A,A*]`
`$A`	rijtjes met een deelrijtje uit taal `A`: `[?*,A,?*]`
`~A`	complement van taal `A`: `?* - A`

1.2 Praktisch

Voor de meesten van jullie is unix een nieuwe omgeving. Haal een paar keer diep adem, en spendeer de eerste tien minuten met de kleine unix survival kit. Vervolgens hieronder een suggestie voor een stappenplan.

- ▷ Verdeel de taken zo, dat je in je team een egrep specialist hebt, en een FSA Utilities expert.
- ▷ Unix, heb je in de survival kit gezien, heeft ingebouwde ondersteuning voor reguliere patronen; op `http://www.regularexpression.info/` vind je een handig overzicht. Met `egrep` kan je snel door de testset zoeken. `rev` en `sort` zijn nuttig voor het omkeren en sorteren van je resultaten.
- ▷ Als je greep op de materie hebt aan de hand van `egrep` en vrienden, is het moment gekomen om op de FSA Utilities over te stappen. Open een editor, en werk de macro definities hierboven uit naar je eigen smaak. Save je definities als `mijnmacros.pl` of een andere bestandsnaam, met `pl` suffix. Je kan op verschillende manieren testen. Enkele voorbeelden (de dollar staat voor de unix prompt):

  $ `fsa -aux mijnmacros.pl -raa 'klinker*'`

  Als je dit commando geeft (afsluiten met een return), kan je aan het scherm interactief rijtjes invoeren, en kijken of die al dan niet herkend worden door de reguliere expressie `klinker*`. In plaats van `klinker*` kan je wat voor reguliere expressie dan ook invullen.

  $ `fsa -aux mijnmacros.pl -raa 'klinker*' < mijntestbestand`

  Zoals hierboven, maar nu test je `klinker*` voor de woorden in `mijntestbestand`
- ▷ Haal het `testfsa` testscript op. Activeer het met het commando `chmod a+x testfsa` Stel je uiteindelijke lettergreepmachine heb je `syll` genoemd. Je kan dan `syll` uitproberen op de hele testset `monosyll` met het commando hieronder.

  $ `./testfsa mijnmacros.pl syll monosyll`

1.3 Beoordeling

Oplossingen worden beoordeeld aan de hand van twee criteria:

- ▷ Precisie: hoe accuraat is de herkenner in het identificeren van de 368 éénlettergrepige woorden in de testset? Streefdoel: zero fouten!
- ▷ Economie: hoe compact is de beschrijving waarmee je het gewenste patroon karakteriseert?
- ▷ Herkennen versus genereren: maak een sample aan van 1000 rijtjes die je automaat produceert; zijn dit inderdaad *mogelijke* Nederlandse 1-lettergrepige woorden: woorden die niet in de testset zitten, maar die zouden kunnen bestaan ('prok': ok; 'rpok': fout)?

2

In het hoorcollege hebben we twee soorten eindige automaten vergeleken: een FSA herkenner herkent een verzameling rijtjes (reguliere taal); een transducer herkent een verzameling *paren* van rijtjes (reguliere relatie).

Bij de transducers horen een paar nieuwe bewerkingen: INVERSIE (omkeren van input en output) en COMPOSITIE: `A o B` zet rijtje w om in w'' als `A` rijtje w omzet in w', en `B` w' in w''.

2.1 Splitsen in lettergrepen

Deze opdracht is een vervolg op de lettergreepherkenner: je bouwt een transducer die de meerlettergrepige woorden uit de testset `monosyll` splitst in lettergrepen. Haal het bestand `negatief.no` met meerlettergrepige woorden op, en schrijf ze weg naar je werkfolder.

Bouw nu een omzetter die tussen twee nucleus rijtjes een afbreekteken zet.

```
macro(splits, ...).
```

Je kan stukken van je code voor de lettergreepherkenner gebruiken: wellicht heb je al een goede definitie voor `nucleus`, etc. Maar er zijn een paar dingen waar je op moet letten: *roeier* wil je graag in *roei-er* omzetten, niet in *roe-ier* of *ro-ei-er*, ...

TIPS. Om een teken in te voeren, kan je `[]` (het *lege rijtje*) omzetten in dat teken. Het afbreekteken is een speciaal teken voor de computer: zet er enkele aanhalingtekens omheen, als je het in je macro definities gebruikt: `'-'`.

2.2 T9

Voor de tweede transduceropdracht heb je een antiek mobieltje nodig. Je bouwt een omzetter die voor een invoerrijtje van de cijfertoetsen suggesties terruggeeft van woorden uit een woordenboek.

STAP EEN. Maak een omzetter `toets` die een rijtje van cijfers omzet in de mogelijke tekenrijtjes die erbij horen. Voorbeeld: bij de toets `2` horen de letters `a,b,c`, enzovoort.

STAP TWEE. Maak een woordenboek van de eenlettergrepige woorden uit de testset `monosyll` (je oplossing van de eerste opdracht). Een handige ingebouwde operatie is `words`: `words(Lijst)` maakt een FSA herkenner voor de woorden in `Lijst`. Bijvoorbeeld: `words([aap,noot,mies])` staat voor de herkenner `{[a,a,p],[n,o,o,t],[m,i,e,s]}`.

STAP DRIE. Combineer je machines `toets` en `lexicon` met compositie.

BONUS. Pas je `t9` omzetter aan, zo dat hij een cijferrijtje aanvult tot mogelijke woorden uit het lexicon. Bijvoorbeeld: voor invoer `66` krijg je `noot` (en `nood`, en `norm`, ...).

□