Natuurlijke Taalverwerking
Natural Language Processing

Gosse Bouma and Begoña Villada

3e trimester 2002/2003

Overview

1. Recognizers vs Transducers,
2. Applications of Transducers,
3. Example Automata and Reg Ex Notation,
4. (Non-)determinism,
5. Composition of Transducers,
6. Input/Output Reversal,
7. Finite State Part of Speech Tagging

Recognizers vs Transducers

- A finite state recognizer is an automaton which recognizes strings:

1. De jongen herkende alleen zijn vrienden (YES)

- A finite state transducer is an automaton which produces output for the strings it recognizes:
  - ⋆ Recognize (1),
  - ⋆ Output: The boy recognized only his friends

Stemming

- Translate a word into its base form,
- Useful for text classification and information retrieval tasks:
- If you are interested in *klooster*, you are probably also interested in texts aboute *kloosters*.
  - ⋆ Kloosters hebben in Amsterdam twee eeuwen bestaan
  - ⋆ `klooster heb in amsterdam twee eeuw besta`

Part of Speech Tagging

- Translate a sequence of words into a sequence of Part of Speech Tags
- Useful as a first step towards full parsing or to support searching for linguistic patterns,

Grapheme to Phoneme Conversion

- Translate a sequence of letters into a sequence of phonemes
- Required for Text to Speech applications
- Each letter or sequence of letters is translated into a phoneme

a	a	l	s	c	h	o	l	v	e	r
$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$
ϵ	a	l	s	ϵ	x	O	l	v	@	ϵ

Part of Speech Tagging

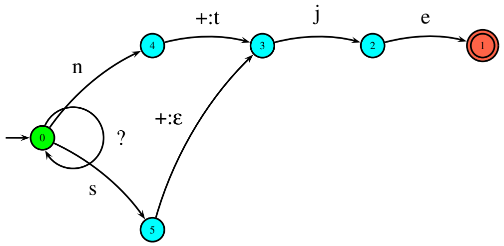
```
Op      Prep(voor)
de      Art(bep,zijd_of_mv,neut)
laatste Adj(attr,overtr,verv_neut)
dag     N(soort,ev,neut)
voor    Prep(voor)
het     Art(bep,onzijd,neut)
begin   N(soort,ev,neut)
van     Prep(voor)
het     Art(bep,onzijd,neut)
Olympisch       Adj(attr,stell,onverv)
jaar    N(soort,ev,neut)
keerde  V(intrans,ovt,1_of_2_of_3,ev)
Kamiel  N(eigen,ev,neut)
```

Dimunitives

huis	huisje	ring	ringetje
haan	haantje	koning	koninkje
lam	lammetje	bloem	bloempje
raam	raampje	bloem	bloemetje
bom	bommetje	pop	poppetje
boom	boompje	pop	popje

Finite State Transducer

huis+je $\longrightarrow$ huisje
haan+je $\longrightarrow$ haantje

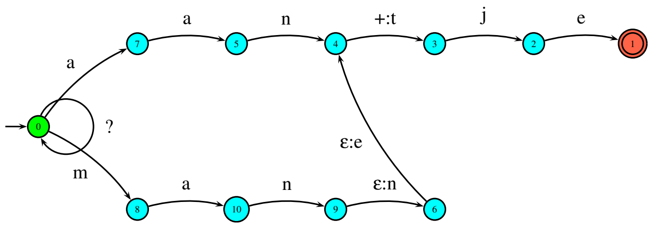


Regex Notation for Transducers

- `[a:b, c*]` translates, among others, accc in bccc.
- `:` is the 'pair'-operator: it translates a symbol A in a symbol B.

Finite State Transducer 2

maan+je $\longrightarrow$ maantje
man+je $\longrightarrow$ mannetje



Regex Notation for Transducers

- `[a:b, c*]` is short for `[a:b, (c:c)*]`
- By default, a regular expression without ':' is read as the identity-transducer: every symbol in the input is mapped onto itself.

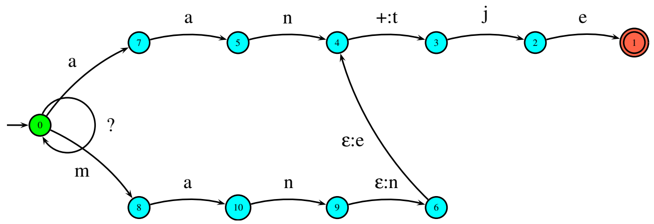
Example

huis+je	$\longrightarrow$	huisje
haan+je	$\longrightarrow$	haantje
maan+je	$\longrightarrow$	maantje
man+je	$\longrightarrow$	mannetje

```
[? *,{[s,+ :[]],
      [a,a,n,+ :t],
      [~a,a,n,[]:n,[]:e,+ :t]
     },
 j,e
]
```

Non-Determinism: Example

maan+je	$\longrightarrow$	maantje
man+je	$\longrightarrow$	mannetje



(Non-)determinism

- An transducer is deterministic if for every state and inputsymbol, at most a single transduction to a new state is possible.
- Non-deterministic transducers can sometimes be made deterministic, but not always.
- Non-deterministic recognizers can always be made deterministic.

Two Sources of Non-determinism

- Unbounded Look-ahead

```
{[a:b, c*, b], [a:d, c*, d]}
```

acccb $\longrightarrow$ bcccb acccd $\longrightarrow$ dcccd

- Multiple outputs

```
[?*, o, e, m, {+:p, +:[e,t,]}, j, e]
```

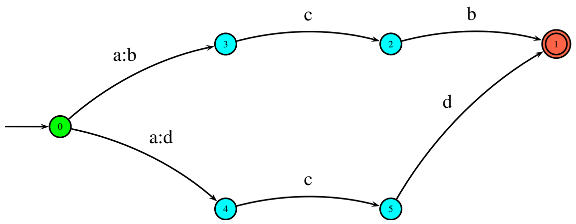
bloem+je $\longrightarrow$ bloempje
bloem+je $\longrightarrow$ bloemetje

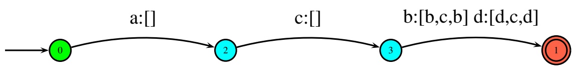
Making a Transducer Deterministic

- Deterministic transducers are more efficient than non-deterministic transducers (because no choice-points/backtracking/search is required).
- But deterministic transducers can be much larger than corresponding non-deterministic transducer.
- (`t_determinize` option in FSA).

Making a Transducer Determinisic

maan+je $\longrightarrow$ maantje
man+je $\longrightarrow$ mannetje





From Number Words to Numbers

drieentwintig	eenendertig
23	31

```
macro(eentallen,
   {een:1, twee:2, drie:3 } ).
macro(twintig,
   [ []:2, eentallen, entwintig:[] ]).
macro(dertig,
   [ []:3, eentallen, endertig:[] ]).
```

From Dutch to English Numbers

- Automatic translation of (spoken) Dutch into English requires translation of number words,
- eenentwintig $\longrightarrow$ twentyone,
- eenentwintig $\longrightarrow$ 21 $\longrightarrow$ twentyone

From Dutch to English Numbers

- Transducer T1 for translating Dutch Number Words into Numbers,
- Transducer T2 for translating Numbers into English Number Words
- The output of T1 is used as input by T2.

Number Translation by Composition

```
macro(dutch2num,
  {een:1, twee:2, drie:3, ....}).
macro(num2eng,
  {1:one, 2:two, 3:three, ....}).
macro(dutch2eng,
  dutch2num o num2eng).
```

Composition

- The composition of transducers T1 and T2 is a new transducer T3, which is equivalent to passing the input through T1, taking the output of T1 as input for T2, and taking the output of T2 as output.
- T1 o T2 denotes the composition of T1 and T2.

Input/Output reversal

- The inverse of a transducer T is a transducer which takes as input the output of T, and produces as output the input of T.
- In FSA `inverse(T)` produces the inverse of T.
- Translating English to Dutch:

```
macro(eng2dutch,
  inverse(num2eng) o inverse(dutch2num) ).
```

Finite State POS Tagging

- Assign Part of Speech tags to words,
- but many words have more than one POS:
  - ⋆ De/det fiets/n staat/v in/p de/det schuur/n
  - ⋆ Ik/pro fietsv naar/p school/n

Finite State POS Tagging

```
macro(lexicon,
  { de:det, fiets:n, fiets:v, naar:p.
    in:p, school:n, schuur:n, staat:v }
  ).
macro(no_det_v,
 ~ [ det, v ] ).
macro(tagger,
  lexicon o no_det_v ).
```

Finite State POS Tagging

- A Solution:
  - ⋆ A non-deterministic T which assigns a word all possible POS tags,
  - ⋆ Recognizers R which filter the output of T,
  - ⋆ Compose T and (the identity transducer for) R.
- Requires linguist to develop filters.

Finite State POS Tagging

- Using Transformation-based (Error-driven) learning:
  - ⋆ Assign each word its most frequent tag initially,
  - ⋆ Learn rules which correct frequent mistakes
- Tagger is composition of the transducer for the initial system and the error-correction rules,
- Requires a corpus annotated with POS tags.