# T-(538|725)-MALV, Natural Language Processing
# lexc - **Lex**icon **C**ompiler

Hrafn Loftsson[1] Hannes Högni Vilhjálmsson[1]

[1]School of Computer Science, Reykjavik University

September 2008

# Outline

# Outline

# Two-level rules

## Kimmo Koskenniemi (1983)

| Rule | Description |
|---|---|
| a:b $\Rightarrow$ lc _ rc | *a* is transduced as *b* **only** when it has lc to the left and rc to the right |
| a:b $\Leftarrow$ lc _ rc | *a* is **always** transduced as *b* when it has lc to the left and rc to the right |
| a:b $\Leftrightarrow$ lc _ rc | *a* is transduced as *b* **always and only** when it has lc to the left and rc to the right |
| a:b $/\Leftarrow$ lc _ rc | *a* is **never** transduced as *b* when it has lc to the left and rc to the right |

More detail:

`http://www.sil.org/pckimmo/v2/doc/Rules_1.html#subsec:3.1.4`

# Two-level rules

## An example from English

| | | | |
|---|---|---|---|
| Examples | happy+er | party+s | marry+ed |
| | happi0er | parties | marri0ed |
| Rules | Cy+er | Cy+s | Cy+ed |
| | Ci0er | Cies | Ci0ed |

1. y:i $\Leftarrow$ C:C _ +:0 e:e r:r
2. y:i $\Leftarrow$ C:C _ +:e s:s
3. y:i $\Leftarrow$ C:C _ +:0 e:e d:d

All the rules are applied in **parallel**. Every rule must be successfully applied to the current pair of characters *lexical:surface* before moving to the next pair (see Fig. 5.10 page 139).

# Two-level rules

## Relation to finite-state transducers

- A two-level rules can be compiled into an equivalent finite-state transducer.
- A program which performs two-level morphological analysis:
  - The user writes a set of two-level rules.
  - The program compiles the rules into transducers.
- Example: **lexc** (Lexicon Compiler), a part of the **XFST** (Xerox Finite-State Tool). You can download the file *XFST.zip* under "Other material" in MySchool.
- Example: **PC-Kimmo**, `http://www.sil.org/pckimmo/`

# Outline

# lexc

## Lexicon Compiler

- A language (and a compiler) for defining automata and transducers.
- Specifically suitable for defining lexicons.
- Based on two-level morphology.

## The format of a lexc file

- Multichar_Symbols declaration
- Declarations section
- Lexicon Root
- Lexicon X
- Lexicon Y
- ...
- END

# lexc

## Lexicon Compiler

- A language (and a compiler) for defining automata and transducers.
- Specifically suitable for defining lexicons.
- Based on two-level morphology.

## The format of a lexc file

- Multichar_Symbols declaration
- Declarations section
- Lexicon Root
- Lexicon X
- Lexicon Y
- ...
- END

# lexc

## A simple example

```
! ex1-lex.txt (this line is a comment)
LEXICON Root
dog     # ;
cat     # ;
bird    # ;
END
```

## Running it

```
xfst          ; Starts up xfst, which waits for input
xfst[0]:    read lexc < ex1-lex.txt
xfst[1]:
```

# lexc

## A simple example

```
! ex1-lex.txt (this line is a comment)
LEXICON Root
dog     # ;
cat     # ;
bird    # ;
END
```

## Running it

```
xfst          ; Starts up xfst, which waits for input
xfst[0]:    read lexc < ex1-lex.txt
xfst[1]:
```

## Running it

```
xfst[1]: up dog      ; from surface to lexical
xfst[1]: down bird   ; from lexical to surface
xfst[1]: print words
xfst[1]: clear stack
xfst[0]
```

# lexc – Continuation classes

```
LEXICON Root
walk    # ;
walks   # ;
walked  # ;
walking # ;

talk    # ;
talks   # ;
talked  # ;
talking # ;

pack    # ;
packs   # ;
packed  # ;
packing # ;
```

# lexc – Continuation classes

```
LEXICON Root
walk    V ;
talk    V ;
pack    V ;

LEXICON V
s    # ;
ed   # ;
ing  # ;
     #;   ! <- an empty-string entry
```

# Lexical transducers

## Upper:Lower records

```
LEXICON Root
swim:swam       # ;
fi0ght:fought   # ;
```

## Running it

```
xfst[1]: up swam     ; from surface to lexical
xfst[1]: down fight  ; from lexical to surface
xfst[1]: print words
```

# Lexical transducers

## Upper:Lower records

```
LEXICON Root
swim:swam        # ;
fi0ght:fought    # ;
```

## Running it

```
xfst[1]: up swam      ; from surface to lexical
xfst[1]: down fight   ; from lexical to surface
xfst[1]: print words
```

# Lexical transducers

## An example for Icelandic

- See file *ice-lex.txt* in the XFST.xip file

```
xfst[0]: read lexc < ice-lex.txt
xfst[1]: up hestur
xfst[1]: down hest+NO+KK+FT+N
xfst[1]: down gest+NO+KK+FT+N
```