Finite State Morphology

Alexander Fraser & Luisa Berlanda
fraser@cis.uni-muenchen.de

CIS, Ludwig-Maximilians-Universität München

Computational Morphology and Electronic Dictionaries
SoSe 2016
2016-13-06

Outline

- How to work with SFST?
- SFST Programming Exercises

SFST

- programming language for developing finite-state transducers
- compiler which translates programs to transducers
- tools for
  - applying transducers
  - printing transducers
  - comparing transducers

SFST Example Session

```
> echo "Hello\ World\!" > test.fst      storing a small test program
> fst-compiler test.fst test.a         calling the compiler
test.fst: 2

> fst-mor test.a                       interactive transducer usage
reading transducer...                  transducer is loaded
finished.
analyze> Hello World!                  input
Hello World!                           recognised
analyze> Hello World                   another input
no result for Hello World              not recognised
analyze> q                             terminate program
```

SFST Basics

- Write code in terminal:
  - echo " program code " > filename.fst
- Write code in a file:
  - write program code, save as filename.fst
- Compile:
  - fst-compiler filename.fst filename.a
- Execute:
  - fst-mor filename.a

SFST Programming Language

Colon operator a:b

empty string symbol <>

Example: m:m o:i u:<> s:c e:e

identity mapping a (an abbreviation for a:a)

Example: m o:i u:<> s:c e

{abc}:{AB} is expanded to a:A b:B c:<>

Example: {mouse}:{mice}

Is this expression equivalent to the previous two?

Exercise

Try to write these examples as code and try out, what they analyze and generate.

- m:m o:i u:<> s:c e:e
- m o:i u:<> s:c e
- {mouse}:{mice}

Exercise

Try to write these examples as code and try out, what they analyze and generate.

- John | Mary | James
- mouse | {mouse}:{mice}

Disjunction

John | Mary | James

accepts these three strings and maps them onto themselves

mouse | {mouse}:{mice}

analyses mouse and mice as mouse

Multi-Character Symbols

strings enclosed in <…> are treated as a single unit.

{mouse<N><pl>}:{mice}

analyses mice as mouse<N><pl>

Multi-Character Symbols

A more complex example:

```
schreib {<V><pres>}:{} (\
   {<1><sg>}:{e} |\
   {<2><sg>}:{st} |\
   {<3><sg>}:{t} |\
   {<1><pl>}:{en} |\
   {<2><pl>}:{t} |\
   {<3><pl>}:{en})
```

The backslashes (\) indicate that the expression continues in the next line

What is the analysis of schreibst and schreiben?

Exercise

Try to write this example as code and try out, what it analyzes and generates.

- schreib {<V><pres>}:{} (\
  {<1><sg>}:{e} |\
  {<2><sg>}:{st} |\
  {<3><sg>}:{t} |\
  {<1><pl>}:{en} |\
  {<2><pl>}:{t} |\
  {<3><pl>}:{en})

Multi-Character Symbols

- What is the analysis of schreibst and schreiben?
  - Schreibst: schreib<V><pres><2><sg>
  - Schreiben: schreib<V><pres><1><pl>
    schreib<V><pres><3><pl>

Character Ranges

[abc]:[AB] is expanded to a:A|b:B|c:B

Example: [a-z A-Z]:[b-za B-ZA]*

What do you get for the input IBM?

What does this transducer recognise?

[+-]? [0-9]* (\. [0-9]+)?

Note: Characters with a special meaning (! ? . & % | * : ( ) [ ] { } etc.) need to be quoted with a backslash. Blanks are ignored unless they are quoted with a backslash.

Exercise

Try to write these examples as code and try out, what they analyze and generate.

- [a-z A-Z]:[b-za B-ZA]*
- [+-]? [0-9]* (\. [0-9]+)?

Character Ranges

- What do you get for the input IBM?
  - HAL

- What does this transducer recognise?
  - Integers ans Floats, negative or positive

Composition

[a-z A-Z]:[b-za B-ZA]* || [a-z A-Z]:[b-za B-ZA]*

The output of the first transducer is the input of the next transducer (in generation mode).

The compiler produces a single transducer which directly produces the final output without an intermediate step.

Exercise

Try to write this examples as code and try out, what it analyzes and generates.

- [a-z A-Z]:[b-za B-ZA]* || [a-z A-Z]:[b-za B-ZA]*

Printing Transducers

> echo ‘[a-z A-Z]:[b-za B-ZA]* || [a-z A-Z]:[b-za B-ZA]*‘ > test.fst

> fst-compiler-utf8 test.fst test.a

> fst-print test.a

0 0 Y A

0 0 Z B

0 0 A C

0 0 B D

0 0 C E

…

0 0 x z

0

SFST Programs

- fst-compiler test.fst test.a
  reads an SFST program and translates it into a transducer.
  Use fst-compiler-utf8 it you use Unicode symbols.
- fst-print test.a
  prints the transitions of a compiled transducer in tabular form
- fst-generate test.a
  prints all the string-to-string mappings (with colon notation)
  and might not terminate.
- fst-mor test.a
  analyse and generate strings interactively
- fst-compact test.a test.ca
  convert the transducer to a more efficient format
- fst-infl2 test.ca input.txt
  analyses a sequence of strings with a compact transducer

Call the programs with option „-h“ to get information on available options.

Transducer Variables

```
$Vroot$ = walk | talk | bark          % list of verbs with regular inflection
$Vinfl$ = <V>:<> (\                   % regular verbal inflection
    [<inf><n3s>]:<> |\
    {<3s>}:{s} |\
    {<ger>}:{ing} |\
    {<past>}:{ed})

$Nroot$ = hat | head | trick          % list of nouns with regular inflection
$Ninfl$ = <N>:<> (\                   % regular nominal inflection
    {<sg>}:{} |\
    {<pl>}:{s})

$Vroot$ $Vinfl$ | $Nroot$ $Ninfl$    % combine stems and inflectional endings
```

Exercise

- Try to write this examples as code and try out, what it analyzes and generates.

```
$Vroot$ = walk | talk | bark          % list of verbs with regular inflection
$Vinfl$ = <V>:<> (\                   % regular verbal inflection
    [<inf><n3s>]:<> |\
    {<3s>}:{s} |\
    {<ger>}:{ing} |\
    {<past>}:{ed})

$Nroot$ = hat | head | trick          % list of nouns with regular inflection
$Ninfl$ = <N>:<> (\                   % regular nominal inflection
    {<sg>}:{} |\
    {<pl>}:{s})

$Vroot$ $Vinfl$ | $Nroot$ $Ninfl$
```

Exercise

- Write a program that maps all letters to a number
- Write a program that maps the word foot onto it´s plural form

Solution

- [0-9 0-9 0-9] :[a-z A-Z]*
- f e:o e:o t

Homework

- Write a pipeline that maps all letters to lowercase and orders them backwards

- Family Huber has three children. Their first child is called Mia, the next one Toni and the last one Pia.
- Family Band has three children as well. Michael, Paul and Pia.
- Write a program, that can tell us the following details about the children: which family does he/she belong to, is it a son or a daugter, was he/she the first, second or third child.
- Output format:
  - <family><family name><first, second child><gender>

- Thank you for your attention