Finite state technology for natural languages processing

- Finite state technology is usually sufficient for describing natural languages phenomena such as morphology: in particular the rewrite rule $A \rightarrow B \mid L \_ R$ defines a regular relation.

- Finite state networks guarantee linear recognition time.

- The method: constructing finite state networks describing specific phenomena and combining them using the closure properties.

XFST and regular expressions calculus

- XFST is an interface giving access to finite state operations (algorithms such as union, concatenation, intersection, composition etc.)
- XFST includes a regular expression compiler
- The interface of XFST includes a look up operation (apply up) and a generation operation (apply down)
- The regular expression language employed by XFST is an extended version of standard regular expression

Basic regular expressions

(A)	Option	~A	Complement
\A	Term complement	$A	Contains
A*	Kleene star	A+	Kleene plus
A/B	Ignore	A B	Concatenation
A | B	Union	A & B	Intersection
A - B	Minus	A .x. B	Cross product
A .o. B	Composition	?	Any symbol
A.u	Upper language	A.l	Lower language
A.i	Inverse		

[ ] Empty language and precedence order

Extended regular expressions

The basic concept behind regular expressions calculus:

Create more complex regular expressions by using the basic ones.

For example:

$A $\equiv$?* A ?*

Unconditional replacement $A \rightarrow B$

Meaning: obligatory unconditional replacement of the language A by the language B.

- $a \rightarrow b$

bcd	abac	aaab
bcd	bbbc	bbbb

- $a^{+} \rightarrow b$

bcd	abac	aaab
bcd	bbbc	bbbb
		bbb
		bbb
		bb

Unconditional replacement $A \rightarrow B$

Construction:

$[\ noA\ \ [A\ .x.\ B]\ ]^*\ \ noA$ where noA abbreviates $\sim \$[\ A - [\ ]\ ]$.

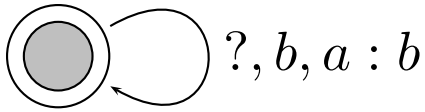
- This construction defines a regular relation were each member of this relation contains any number (which can be also zero) of iterations of $[A\ \ .x.\ \ B]$ combined with strings that do not contain A which are mapped to themselves.

- If $\epsilon \notin A$ then $\sim \$A$ and $\sim \$[\ A - [\ ]\ ]$ are equivalent. But if $\epsilon \in A$ then $\sim \$A$ is null whereas $\sim \$[\ A - [\ ]\ ]$ contains at least ϵ.

Unconditional replacement $A \leftarrow B$

- Meaning: obligatory unconditional inverse replacement.
- Definition and construction: $[B \rightarrow A]\,.i$
- What is the difference between $A \rightarrow B$ and $A \leftarrow B$?

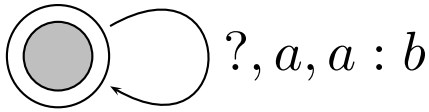
$a \rightarrow b$



bb, ab

bb

$a \leftarrow b$

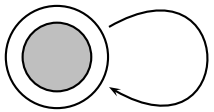


ab

ab, bb

Optional replacement $A(\rightarrow)B$

Example: $a\ (\rightarrow)\ b$



$?, a, b, a : b$ $\qquad ab$

ab, bb

The tempting (and incorrect) construction: $[A \rightarrow B]\ \mid\ [A]$

The correct way: $[?^*\ [A\ .x.\ B]\ ]^*\ ?^*$ or $[\$[A \rightarrow B]\ ]^*$

Conditional replacement

- $A \rightarrow B \;||\; L \_ R$ where L,R on the upper side.
- $A \rightarrow B \;//\; L \_ R$ where L on lower and R on the upper side.
- $A \rightarrow B \;\backslash\backslash\; L \_ R$ where L on upper and R on the lower side.
- $A \rightarrow B \;\backslash/\; L \_ R$ where L,R on the lower side.

Example:

$a\ b \rightarrow x \;||\; a\ b \_ a$

abababa

ab x x a

$a\ b \rightarrow x \;//a\ b \_ a$

abababa

ab x aba

Conditional replacement $A \rightarrow B \ || \ L \_ R$

The idea: make the conditional replacement behave exactly like unconditional replacement except that the operation does not take place unless the specified context is present.

The problem: the part being replaced can be at the same time the context of another replacement.

The solution: first, decompose the complex relation into a set of relatively simple components, define them independently of one another, and finally use composition to combine them.

Conditional replacement $A \rightarrow B \ || \ L \_ R$

1. InsertBrackets
2. ConstrainBrackets
3. LeftContext
4. RightContext
5. Replace
6. RemoveBrackets

Conditional replacement $A \rightarrow B \ || \ L \_ R$

- Two bracket symbols, $<$ and $>$, are introduced in (1) and (6).
- $<$ indicates the end of a complete left context. $>$ indicates the beginning of a complete right context.
- Their distribution is controlled by (2), (3) and (4). (2) constrains them with respect to each other, whereas (3) and (4) constrain them with respect to the left and right context.
- The replacement expression (5) includes the brackets on both sides of the relation.
- The final result of the composition does not contain any brackets. (1) removes them from the upper side, (6) from the lower side.

Conditional replacement $A \rightarrow B \ || \ L \_ R$

- Let $<$ and $>$ be two symbols not in Σ. The escape character $\%$ is used since $<$ and $>$ are saved symbols in XFST.

- $InsertBrackets = [\ [\ ] \leftarrow \ \% < \ | \ \% > \ ]$. InserBrackets eliminates from the upper side language all context markers that appear on the lower side.

- $ConstrainBrackets = [\ \sim \ \$[\% < \quad \% >] \ ]$. ConstrainBrackets denotes the language consisting of strings that do not contain $<>$ anywhere.

Conditional replacement $A \rightarrow B \ || \ L \_ R$

- $LeftContext =$

$$[\sim[\sim[...L] \ [<...]] \ \& \ \sim[[...L] \ \sim[<...]]]$$

LeftContext denotes the language in which any instance of $<$ is immediately preceded by L and every L is immediately followed by $<$, ignoring irrelevant brackets.
$[...L]$ denotes $[[?* \ L/[\% < \ | \ \% >]] - [?* \ \% <]]$, the language of all strings ending in L, ignoring all brackets except for a final $<$. $[< ...]$ denotes $[\% < /\% > \ ?*]$, the language of strings beginning with $<$, ignoring the other bracket.

Conditional replacement $A \rightarrow B \ || \ L \_ R$

- $RightContext =$

$$[\sim[\,[...>]\ \sim[R...]\,]\,\&\ \sim[\sim[...>]\ [R...]\,]\,]$$

RightContext denotes the language in which any instance of $>$ is immediately followed by R and any R is immediately preceded by $>$, ignoring irrelevant brackets.
$[R...]$ denotes $[\,[\,R/[\%< \ | \ \%>]\ ?*\,]-[\%<?*]\,]$, the language of all strings beginning with R, ignoring all brackets except for an initial $>$.
$[...>]$ denotes $[?*\ \%>/\%<]$, the language of strings ending with $>$, ignoring the other bracket.

Conditional replacement $A \rightarrow B \ || \ L \_ R$

- $$Replace = [\%< \ \ A/[\%< \ | \ \%>] \ \%> \rightarrow \ \%< \ \ B/[\%< \ | \ \%>] \ \%>]$$

  This is the unconditional replacement of $< A >$ by $< B >$, ignoring irrelevant brackets.

- $RemoveBrackets = [\ \%< \ | \ \%> \rightarrow \ [\ ] \ ]$. RemoveBrackets denotes the relation that maps the strings of the upper language to the same strings without any context markers.

Conditional replacement $A \rightarrow B \ || \ L \_ R$

Construction:

$$InsertBrackets$$

$$.o.$$

$$ConstrainBrackets$$

$$.o.$$

$$LeftContext$$

$$.o.$$

$$RightContext$$

$$.o.$$

$$Replace$$

$$.o.$$

$$RemoveBrackets$$

Conditional replacement $A \rightarrow B \;||\; L \_ R$

Special cases:

- $A = \{\epsilon\}$ or $\epsilon \in A$.
- Boundary symbol (.#.):

  $L \_ R$ actually means $?^* \; L \_ R \; ?^*$