Introduction to XFST

Extended version of standard regular expressions

- The regular expression language employed by XFST is an extended version of standard regular expressions

- The interface of XFST includes a lookup operation

- XFST includes a regular expression compiler

- Intersection, composition, etc.

- Algorithms such as union, concatenation, iteration,
- XFST is an interface giving access to finite-state operations

Introduction to XFST
Introduction to XFS – example

- \( \text{\texttt{a} \cdot \text{\texttt{b}}} \) denotes every string in \( B \) to \( A \).

- \( \text{\texttt{a}} \) denotes every string in \( A \) to \( a \).

Introduction to XFS – denotative relations
Introduction to XFST - Variables

and Generation

Introduction to XFST - Example of Lookup
Introduction to XFSR - Replace Rules

XFSR define Rule % Morpheme boundary - 0:

"bookkeeper":

This can be used to clear unnecessary symbols introduced for

XFSR define Rule p => \w - !;
XFSR define Rule n => u - !;
XFSR define Rule N => \w - P - !;

Context can be omitted:

Introduction to XFSR - Replace Rules

XFSR read regex 0. Rule 0. Rule 2:
XFSR define Rule p => \w - !;
XFSR define Rule n => u - !;
XFSR define Rule N => \w - P - !;
XFSR clear stack !

to \w when the p is followed by \w.
The language also has an assimilation rule which changes \p elsewhere.

on its environment. \N is realized as \w before \p and as \w elsewhere.
The language has an underspecified nasal morphone.

Introduction to XFSR - Replace Rules

The simplest replace rule is of the form

regular expression metalinguage.

Replace rules are an extremely powerful extension of the
Introduction to XFST – Marking

**The rule:**

- The special symbol “…” in the right-hand side of a replace rule stands for whatever was matched in the left-hand side of the replace rule.

Introduction to XFST – Replace Rules

**Rules can apply in parallel:**

**Examples:**

```
abaa
```

```
xfst apply down abaa

xfst read regex a \<-q.b\<-q
xfst clear stack
```

```
bbba
```

```
xfst apply down bbba

xfst read regex a \<-q.b\<-q
```

```
xfst clear stack
```

```
qaa
```

```
xfst apply down qaa

xfst read regex q \<-o.a\<-q
```

```
xfst clear stack
```

```
qaaa
```

```
xfst apply down qaaa

xfst read regex q \<-o.a\<-q
```

```
xfst clear stack
```

```
 [ ] 4 \<-> B, B \<-> A | 1, 1, R, L, 2, 1, 2 - R, 2
```

or multiple replacements and multiple contexts:

```
 [ ] 4 \<-> B | 1, 1, R, L, 2 - R, 2
```

or multiple contexts:

```
 [ ] 4 \<-> B, B \<-> A | 1 - R
```

or multiple replacements that share the same context:

```
 [ ] 4 \<-> B, B \<-> A | 4
```

Rules can define multiple replacements:
Introduction to XFSI - Shallow Parsing

etc.

etc.

etc.

etc.

etc.

etc.
Introduction to FST - the coke machine
Introduction to XFS – the coke machine

...