HFST supports Replace Rules which replicate behavior of the xfst Replace Rules, described in *Kempe and Karttunen (1996)* and *Beesley and Karttunen (2003)* .

Although they share the same notation, Replace Rules in HFST were developed upon the concept of Generalized Lenient Composition, described in Yli-Jyrä (2008b). --> difference another paper?

Here, in this paper section will be described how to use Replace rules with HFST tools. For detailed description of each rule see *Beesley and Karttunen (2003)* .

**Examples:**

Parallel rules

The general parallel replace rule expression:

A1 -> B1 || L1 \_ R1 ,, ... ,, Ai -> Bi || Li \_ Ri

is a list of parallel replace rules separated with two commas (,,), where A, B, L and R are regular expressions that describe languages, states that Ai in upper language maps to Bi in lower language, if it occurs between contexts Li and Ri.

Depending on the type of the replacement, the arrow -> can be replaced with any from the table [1]. However, same as in xfst, in one parallel rule statement, all rules must have the same replace arrow.

|  |  |
| --- | --- |
| Replace arrow | Replace function |
| -> | Replace right |
| (->) | Replace right optional |
| <- | Replace left |
| (<-) | replace left optional |
| @-> | Longest match from left to right |
| ->@ | Longest match from right to left |
| @> | Shortest match from left to right |
| >@ | Shortest match from right to left |

Table 1- replace functions

For example, rule:

a -> b ,, b -> c is valid parallel rule, but a -> b ,, b (->) c will give back the result which will differ from expected one. Since the expression has different replace arrows in each parallel rule, parser will recognize that the arrows are not the same and will return the result as the arrow in the first rule was in all other rules as well. (TODO: warning as well)

Contexts:

Same as in XFST, in replace expressions, contexts are not obligatory. Also, there are four different combinations of contexts direction (weather they are taken from upper of lower language) :

|| - both contexts are taken from the upper language

// - left context is taken from the lower language, right from the upper

\\ - left context is taken from the upper language, right from the lower

\/ - both contexts are taken from the lower language.

In XFST, all rules in parallel rule expression have to have the same format. They need to have the same arrow and the same context layout. In our implementation, we keep the restraint that the replace a rules need to have the same arrow, but the context format for each rule can differ from each other.

Therefore, this rule would not be allowed in XFST, but is valid in HFST:

a -> b || m \_ n ,, b -> c || p \_

**hfst-regexp2fst**

All the rules can be used in HFST tool for parsing regular expressions hfst-regexp2fst.

Right arrow replace rule, could be used like this:

echo "a -> b ;" | ./hfst-regexp2fst -f sfst | ./hfst-fst2txt

The output from the hfst-regexp2fst tool is a transducer written in the transducer format defined with -f option. It can be stored in a file, or, like in this example, printed using HFST tool hfst-fst2txt.

Result transducer can be applied to a language by using composition. For example, if we want to change

A bit more complicated example

where A, B, Li are regular languages.

A -> B || L1 \_ R1 , L2 \_ R2 ;

A -> B || .#. L1 \_ R1 , L2 \_ R2 ;

Longest match

Can be parallel, but in that case they need to have the same arrow operator.

However, they don’t need to have the same context, like in xfst.

A –> B ,, B -> A ;

a -> b ,, b -> k || b \_ ;

References:

2008b Anssi Yli-Jyrä. 2008b. Transducers from Parallel Replace Rules and Modes with Generalized Lenient Composition. In Finite-state methods and natural language processing. Thomas Hanneforth and Kay-Michael Würtzner. 6th International Workshop, FSMNLP 2007. Potsdam, Germany, September 14--16. Revised Papers. Universitätsverlag Potsdam. 2008.

Parallel Replacement in Finite State Calculus (1996) by Andr Kempe , Lauri Karttunen

@INPROCEEDINGS{Kempe96parallelreplacement,  
    author = {Andr Kempe and Lauri Karttunen},  
    title = {Parallel Replacement in Finite State Calculus},  
    booktitle = {In the Proceedings of the 16th International Conference on Computational Linguistics},  
    year = {1996},  
    pages = {622--627}  
}

@book{Beesley\_Karttunen\_2003, title={Finite State Morphology}, volume={30}, url={http://acl.ldc.upenn.edu/J/J04/J04-2006.pdf}, number={2}, journal={Computational Linguistics}, publisher={CSLI Publications}, author={Beesley, Kenneth R and Karttunen, Lauri}, year={2003}, pages={597--607}}