Appendix 2

How Negfinder Detects Negation

An example of Negfinder's parsing process is given below for a simple sentence containing a negation. The relevant rules from the lexer and parser are listed, and their application is described.

Original sentence: "The patient had no other complaints."

After concept-matching, this becomes: "The patient had no other ~231216:846:10."

The lexer scans the text from left to right and returns single tokens as they match the regular expressions contained in its rules. The following are the relevant lexer rules for this example. Both the lexer and parser rules are shown here in a simplified form, of the type "output : input." For the lexer, the input is the concept-matched text, and the output is a series of tokens. These tokens ("terminal symbols" or "terminals") form the input of the parser, which transforms these into higher-level symbols known as "non-terminal symbols" or "non-terminals."

a) '*word*' / '*igword*' : [a-zA-Z''\-_0-9]+

b) '*prefixnegor*' : ([Nn]o) | ([Ww]ithout | ([Nn]either) | ([Nn]or))

c) '*concept*' : ~ [0-9]+:[0-9]+(:[0-9]+)

d) '.' : [\,\.\!\?;]

The six words and the final period of the sentence match rules a,a,a,b,a,c and d respectively, and the lexer returns the following seven tokens: '*word*' 'word' '*word*' '*prefixnegor*' '*igword*' '*concept*' '.'. (The token '*igword*' (ignored word) is returned for "other" instead of 'word', because after rule b, the lexer enters a different "state" using rule actions that are not detailed here. States add to the flexibility of the lexer by enabling the same regular expression to be translated to a different token depending on previously activated rules.)

The following are the relevant parser rules (terminal symbols are in italics):

a) clause : /*empty */

b) clause : clause '*word*'

c) prefixnegativeor : '*prefixnegor*'

d) prefixnegativeor : prefixnegativeor igword

e) prefixnegative: prefixnegativeor

f) conceptlist : '*concept*'

g) conceptset: conceptlist

h) negation : prefixnegative conceptset;

i) clause : clause negation

j) sentence : clause '.'

The parser traverses the seven tokens it receives from left to right, one at a time, converting ("reducing") them to non-terminal symbols by using any applicable rules, or going on to the next token ("shifting") if no rules can be applied. In this example the following rules are applied in succession, leaving the indicated non-terminal symbols on the parse stack after each rule.

First, rule a, generating the parser token 'clause' on the stack.

Next, rule b, once each for the next 3 tokens ('*word*' '*word*' '*word*') leaving 'clause' each time on the stack.

Next, rules c, d, and e are applied to the next 2 tokens ('*prefixnegor*' '*igword*'), finally leaving 'clause prefixnegative' on the stack

Next, rules f and g are applied to the next token ('*concept*'), leaving 'clause prefixnegative conceptset' on the stack

In the next crucial step, rule h is applied to the last two parser tokens on the stack (prefixnegative conceptset) leaving 'clause negation' on the stack. By mapping back to the original input, this indicates that the words "no other" form the negating phrase and "complaints" is the negated concept.

Next, rule i reduces 'clause negation' to 'clause'.

Finally, rule j is applied to the stack contents plus the final lexer token (clause '.'), allowing the parser to accept the input as a valid 'sentence' for this grammar.