Introduction to PC-PATR program

PC-PATR is a unification-based syntactic parser. The program is free and can be downloaded from http://www.sil.org/pcpatr. A PC-PATR grammar consists of a set of rules and a lexicon. Each rule consists of a context-free phrase structure rule and a set of feature constraints. The lexicon provides the items that can replace the terminal symbols of the phrase structure rules, that is, the letters of the alphabet together with their relevant features.

The Grammar File

The grammar file includes all the rules, except for rules in which the right hand side is a terminal (those rules will be written in the lexicon file as will be explained later). The syntax of the grammar file is as follows:

- Each rule is written in a separate line with no punctuation mark at the end.
- Each rule is divided into two parts: context free backbone and feature constraints.
- The keyword `Rule` starts a context-free phrase structure rule with its set of feature constraints. Note that the keyword Rule is not case sensitive: `RULE` is the same as `rule`, and both are the same as `Rule`.
- A PC-PATR grammar rule has these parts, in the order listed:
  1. the keyword Rule
  2. the nonterminal symbol to be expanded
  3. an arrow ($- >$)
  4. zero or more terminal or nonterminal symbols, possibly marked for alternation or optionality
  5. zero or more feature constraints
- The terminal and nonterminal symbols in the rule have the following characteristics:
  - Upper and lower case letters used in symbols are considered different. For example, `NOUN` is not the same as `Noun`, and neither is the same as `noun`.
  - The symbol X may be used to stand for any terminal or nonterminal. For example, the following rule says that any category in the grammar rules can be replaced by two copies of the same category separated by a CJ.

```
Rule X -> X_1 CJ X_2
      <X cat>  = <X_1 cat>
      <X cat>  = <X_2 cat>
      <X arg1> = <X_1 arg1>
      <X arg1> = <X_2 arg1>
```

The symbol X can be useful for capturing generalities. Care must be taken, since it can be replaced by anything.

  - Index numbers are used to distinguish instances of a symbol that is used more than once in a rule. They are added to the end of a symbol following an underscore character (_). This is illustrated in the rule for X above.
  - The characters `(){}_<>=:/` cannot be used in terminal or nonterminal symbols since they are used for special purposes in the grammar file. The character _ can be used only for attaching an index number to a symbol.
  - By default, the left hand symbol of the first rule in the grammar file is the start symbol of the grammar.

- The symbols on the right hand side of a phrase structure rule may be marked or grouped in various ways:

  - Parentheses around an element of the expansion (right hand) part of a rule indicate that the element is optional. Parentheses may be placed around multiple elements. This makes an optional group of elements.
  - A forward slash (/) is used to separate alternative elements of the expansion (right hand) part of a rule.
  - Curly braces can be used for grouping elements. For example, the following says that an S consists of an NP followed by either a TVP or an IV:

    ```
    Rule S -> NP {TVP / IV}
    ```

    Alternatives are taken to be as long as possible. Thus if the curly braces were omitted from the rule above, as in the rule below, the TVP would be treated as part of the alternative containing the NP. It would not be allowed before the IV.

    ```
    Rule S -> NP TVP / IV
    ```

    Parentheses group enclosed elements the same as curly braces do. Alternatives and groups delimited by parentheses or curly braces may be nested to any depth.

- A rule can be followed by zero or more feature constraints that refer to symbols used in the rule. A feature constraint has these parts, in the order listed:

  1. a feature path that begins with one of the symbols from the phrase structure rule
  2. an equal sign
  3. either another path or a value

  For example, the rule

$$\begin{bmatrix}\text{CAT} : np \\ \text{NUM} : X \\ \text{CASE} : Y\end{bmatrix} \rightarrow \begin{bmatrix}\text{CAT} : d \\ \text{NUM} : X\end{bmatrix} \begin{bmatrix}\text{CAT} : n \\ \text{NUM} : X \\ \text{CASE} : Y\end{bmatrix}$$

  will be written in the grammar file in the following way:

```
rule   NP -> D N
             <NP NUM>=<D NUM>
             <NP NUM>=<N NUM>
             <NP CASE>=<N CASE>
```

The Lexicon File

The lexicon file includes all the rules of the form $A \rightarrow \sigma$ where σ is a terminal. The set of terminals is considered by PC-PATR to be the set of the symbols on the right hand sides of the rules in the lexicon file. For each such rule the lexicon file contains a record which is divided into fields, each of which begins with a standard format marker at the beginning of a line. The fields we will use are:

\w the lexical form of the word, spelled exactly as it will appear in any sentences or phrases input to PC-PATR (by default, \w also marks the initial field of each word's record).

\c word category - the nonterminal of the head of the rule

\f the feature constraints of the rule (i.e., of the feature structure on the left hand side).

For example, the rule

$$\begin{bmatrix} \text{CAT} : n \\ \text{NUM} : sg \\ \text{CASE} : Y \end{bmatrix} \rightarrow lamb$$

will be written in the grammar file in the following way:

```
\w lamb
\c N
\f <NUM>=sg
```

Parsing with PC-PATR

The first stage is to load the files into PC-PATR using the following commands:

`l l <filename>` loads the lexicon file. Note that this command erases any existing lexicon and reads a new lexicon from the specified file.

`l g <filename>` loads the grammar file. Note that this command erases any existing grammar and reads a new grammar from the specified file.

After loading the files, we can parse strings. This can be done using the following commands:

`parse [sentence]` attempts to parse the input sentence according to the loaded grammar. If a sentence is typed on the same line as the command, then that sentence is parsed. If the parse command is given by itself, then the user is prompted repeatedly for sentences to parse. This cycle of typing and parsing is terminated by typing an empty "sentence" (that is, nothing but the Enter or Return key). Both the grammar and the lexicon must be loaded before using this command.

`file parse input-file [output-file]` the same as the previous command only that the strings are read from an input file and the results are printed to an output file or to the screen, if no output file is given.

Example

Consider the grammar for $\{ww \mid w \in \{a,b\}^+\}$ we have seen in class. Then, the corresponding files for PC-PATR could be:

Grammar file:

```
Rule S -> T T_1
     <T FIRST>=<T_1 FIRST>
     <T REST>=<T_1 REST>
Rule T -> AT T_1
     <T FIRST>=ap
     <T REST FIRST>=<T_1 FIRST>
     <T REST REST>=<T_1 REST>
Rule T -> BT T_1
     <T FIRST>=bp
     <T REST FIRST>=<T_1 FIRST>
     <T REST REST>=<T_1 REST>
Rule T -> AT
     <T FIRST>=ap
     <T REST>=elist
Rule T -> BT
     <T FIRST>=bp
     <T REST>=elist
```

Lexicon file:

```
\w a
\c AT

\w b
\c BT
```

Exit

To exit the program use the command `exit`.