The book about The Tranny Project

January 16, 2014

0.1 Tranny the programming language

This section of the book will show you how to write a simple program for mapping the simple language called Chinook Jargon in Tranny. If you are a linguist, you may find it useful to read up on this language before reading on.

Tranny is a programming language which lends itself to parsing (and generating) natural and synthetic language structures. In Tranny, the same program will both parse and generate the same texts, because there are various interpreters for Tranny. One interpreter may take some input and generate the meaning, plus data about conjugation and tense and such (which may be discarded), while another interpreter may, while interpreting the same program, be generating the sentence from the meaning.

This section of the book details the programming language called Tranny.

Tranny's instructions can be categorised in the following way:

1. Instructions for directing the recursive descent parser.
    - constituent
    - adjunct
    - into
    - flags
2. Instructions that scan (or generate) a particular sequence of characters
    - lit
3. Instructions that bind variables.
    - seme
    - rection
    - theta
    - sandhi
4. Miscellaneous directives for the compiler
    - include
    - sandhi-init, sandhi-fin

0.1.1 Directing the recursive descent parser

These combine to provide all the functionality needed to parse any input and provide some representation of the input's meaning. Here is an example rule in tranny:

```
(df sentence (constituent subject) (constituent predicate))
```

What does that mean? It defines a sentence as being a subject and a predicate, in that order. The computer does not need to know what "subject" and "predicate" are for this to work, since this only creates a definition for "sentence". But if you try to apply "sentence" to some input string, then this rule can not apply to the input string unless it can also apply "subject" to the input string, and then apply "predicate" to the same string at the offset where "subject" left off. Similarly, you can define "predicate" as being

```
(df predicate (constituent verb transitive) (constituent object))
```

which means that a predicate is (or, more precisely, a predicate *can be*) a verb followed by an object. And that verb must be labelled "transitive". Here are some example definitions for English verbs. Here is another example predicate:

```
(df predicate (constituent verb intransitive))
```

This definition says that a predicate may consist of an intransitive verb, and nothing else.

A single Tranny program may include arbitrarily many definitions, and it is common for many definitions to have the same name (this is not allowed in many programming languages, but this is the mechanism Tranny uses to describe many various subparsetrees). And when it comes to defining, say, a noun in your language, you will be able to put as many nouns in as you want, and call them all "noun", so that by saying (constituent noun), you are essentially saying, "any noun goes here". Then the computer will find a suitable noun and put it there.

0.1.2 Scanning (or generating) a particular sequence of characters

To be able to parse a sentence, you must also be able to scan a sequence of characters. In recursive descent parlance, this is called a "terminal" (the parser stops recursively descending and accepts some input). This is done by the instruction (lit), which takes one argument, which is appended to the output when generating, or scanned in the incoming string when parsing. There is also (space) which appends or scans a space.

To see an example noun:

```
(df noun (lit haus) (space))
```

In Chinook Jargon, "haus" is a noun, and it is followed by whitespace.

0.1.3 Binding variables

Parsing a sentence, or course, is only really useful if you can also say what it means. This is done with the namespace "seme". There are also a few other namespaces. The name of a namespace is also the name of the instruction used to bind a variable in the namespace. For example:

(df noun (lit haus) (space) (seme (head house)))

This definition is the same as the definition found above on p.2, but is also binds a variable called "head". This variable has the value "house". The notation for this is "(head house)".

Variables in the "seme" namespace give some information about the meaning of the input. Once a variable is bound, it appears in the list of variables together with its value. A variable cannot be bound to one value, and then to another. A variable can be bound negatively; the notation for this is "-example". That means "the variable called example does not have a value".

There are other namespaces beyond seme. This is an incomplete list of the namespaces:

- seme: used to describe semantic relations.
- rection: used to describe grammatical relations.
- theta: used to surface thematic roles (agent, patient, instrumental etc).
- sandhi: used to describe sandhi features of morphemes. (e.g. whether we say "a" or "an" in English depends on the next word, whatever it might be.)