WORKING NOTE AC59

Alan Creak
4 December 1986

NOTES ON RICHARD O'KEEFE'S MAORI PARSER

To make the programme run, you execute the predicate **go**, which

- reads a sentence using **read-sent**. That's designed for general sentences : it accepts a sequence of character strings up to the next full stop, exclamation mark, or question mark, and returns a list of the strings, each classified as an atom (containing only alphabetic characters), a string (between quotation marks), an integer (digits), an apostrophe, an apostrophe-s, or a punctuation character. I imagine Richard used **read-sent** mainly for the sake of convenience, as the Maori parser proper largely ignores the details returned by **read-sent**.

  NOTE : Are apostrophes used in Maori ? Are there any other special punctuation or other marks we should know about ?

  NOTE : It should be straightforward to handle numbers properly.

- uses **split** to reform the sentence. This removes the "atom" label from ordinary words so identified by **read-sent**, but simply passes an anything else - so quoted strings, numbers, etc. aren't handled. **Split** also checks for composite words, and replaces them with their separate parts (whence, presumably, the name); and it queries words not in its dictionary. It will accept in reply either the corrected word (assuming it was a spelling mistake) or the part of speech appropriate to the word, which it will then add to its dictionary for the session.

  NOTE : There's a peculiarity about the " biwords" and "triwords" which I suspect reflects a change of mind half way through writing the programme. Each word is tested by **biword** and **triword** within **split**; but, if those tests fail, it is tested by **word**, which repeats the **biword** and **triword** tests. Richard has noted that **word/1** is written "as a temporary measure" : I *guess* that he intended eventually to collect all the words together into a single dictionary, which would be much easier to administer than the present arrangement (or disarrangement) in which the vocabulary is thinly distributed throughout the whole programme in lots of different predicates for the various parts of speech.

- uses **sentence** to parse the sentence.
- uses **report** to display the parsed version.

Essentially all the grammatical work is done by **sentence**. A sentence is identified as a sequence of phrases, perhaps preceded by an interjection. An interjection is one or two words as defined in §37 of the book.

NOTE : the treatment of *anoo* as an optional addition seems to be faulty. I think that the grammar as written makes the *anoo* mandatory; we need an additional rule of the form **after-interjection (anoo, -) -->**

An interjection is always parsed as

interjection (Int, Mod), where
Int is the word of interjection itself (such as *kaatahi*), and
Mod is a modifier (such as *anoo*).
The two parts are always included; if there is no modifying word in the sentence, the modifier is given as "-". This practice seems to be continued throughout the programme. My instinctive reaction is that it's too rigid, but that's no more than a guess.

NOTE : Could a sentence be introduced by several interjections?

A phrase is parsed by the predicate **phrase** which looks for three components: a preposed periphery, and nucleus, and a postposed periphery. The same argument is used for all the corresponding predicates; the argument represents the parse tree (phrase marker) for the phrase. Its structure is :

phrase (Preposed, Head, Mods, Post)

Preposed is
verbal (Particle, If)
Particle is the verbal particle;
If is *e*, *me*, or -.

NOTE : it isn't clear whether the grammar will cope with an empty preposed periphery for a verbal phrase. (It's all right for nominal phrases.)

NOTE : I have no idea why If follows Particle; it's the other way round in the book. (§48.2)

or nominal (Prep, Sort, Def, Loc)
Prep is the preposition;
Sort says something about the preposition, and can be with, focus, of, at, - ;
Def is a definition:
def (T,P,N)
T is a possessive particle;
P and N are the person and number of the following pronoun;
or just the word;
or -
Loc is one of the positional particles *nei*, *na*, or *ra*, or -.

Head is a base. A preceding *a* is accepted under certain circumstances, but does not appear in the parse tree. The circumstances are :

- the preposed periphery is nominal with any preposition, Sort = at, and no definitive or positional particle, the base of Head is personal, and there are no modifiers; or
- the preposed periphery is nominal with any preposition, Sort = focus or -, and no definitive or positional particle, the base of Head is personal or locative, and there are no modifiers.

In any other circumstances, the base of Head must be consistent in type with the preposed periphery, with any modifiers which may follow, and with a possible terminating manner particle. Consistency for the preposed periphery is determined by the predicate :

Check (Prep, Type)
Prep is the preposed periphery
Type is universal, passive, stative, noun, locative, or personal.
The test succeeds if :
- the preposed periphery is verbal and the type is universal, passive, or stative;
- the preposed periphery is nominal with any of these combinations :
  Definitive not - ; type = noun, universal, or stative.
  Preposition not - ; definitive and locative both - ; type = locative.
  Sort of preposition not at; definitive and locative both - ; type = personal.

NOTE : There may be a flaw here somewhere. In parsing *teenaa koe* (converted by **split** into *te na koe*) it identifies *te na* as a nominal preposed periphery, and *koe* as a personal pronoun, but is then unable to make them agree.

Further: when the programme is given the sentence *haere mai ki te whare*, it determines that there is no preposed periphery (which appears in the parse tree as nominal (-,-,-,-)), then identifies *haere* as a universal base, but cannot match the preposed periphery with the base, and therefore fails. I suspect we need a special representation for an empty preposed periphery.

Mods is a sequence of bases possibly followed by a manner particle. Each has a type, which must agree with the type of the phrase. The type is checked by the predicate :

agree (Modtype, Headtype)
The acceptable combinations are :

Modtype	**Headtype**
passive	passive
derived	derived
anything but passive, derived, or personal	anything but passive or derived

NOTE :

1. There were two typing errors in the original, but I don't think they'll have affected the programme's performance. (I doubt if it ever got this far !)

2. A personal modifier won't agree with anything.

3. I don't know why he's attached the manner particle to the modifiers rather than to the postposed periphery. (§48.4)

Post is
post (D, P, A, H)
D is a directional particle or -
P is a positional particle or -
A is *anoo* or -
H is *hoki*, *anake*, *ana*, *ai*, or -
(*anake* is commented with "???" - I don't know why, but I notice that *anake* is missing from §48.4 of the book.)

To Summarise : the programme is running, but so far its score is 0/2. I stopped trying at that point and took away some traces to analyse so I'd have a better idea of what was going on. I conclude that all the bits are there, but that there's a lot of tidying up to be done.