Group	: Computational Linguistics
Topic	: software

Title	: **New surface rule notation**
Author	: René Leermakers

Doc.Nr.	: 0434
Date	: 12-04-90
Status	: concept
Supersedes	: -
Distribution	: Project
Clearance	: Project
Keywords	: S-rules

Institute for Perception Research

Contents

1 Introduction

This document contains a few changes in the surface rule notation. They were necessary to make possible a more efficient implementation. The major new feature in surface rules is the distinction between *item* parameters and other parameters. Item parameters are parameters that appear in global conditions and in condition parts of those if-then-else statements in global actions that contain assignments to item parameters.

2 Item parameters

Item parameters are to be declared with three colons instead of two:

```
itempar ::: itemtype;
```

If some item parameter is not declared this way, a pascal compilation error will occur. If some parameter is declared an item parameter when it is not, the compiler will *not* notice this. Moreover, the surface parser will run correctly, but will use more space than necessary.

3 lssurfquo

Analogously to the module lsmruquo, some of the HELP functions of surface rules must now be written in the module 'language':*lssurfquo*. To be precise, the

1. functions called in global conditions,
2. the functions that appear in global actions in if-then-else statements that contain assignments to item parameters, and
3. procedures assigning values to item parameters

must be written in this module. In practise it is wise to await a compiler complaint before moving a function to lssurfquo. Of course lssurfquo functions and procedures must have a unique name. If a lssurfquo function or procedure refers to parameters, these parameters must of course in principle be supplied in the function heading. However, in the case of functions that are used only in global conditions, all parameter arguments may be *replaced* by one argument of the type 'rulename'rec. Example:

```
function ok:BOOLEAN; → function ok(rec:SENTENCESRULErec):BOOLEAN;
```

The body of each function in lssurfquo that refers to parameters via a *rec* parameter is to be contained in a with statement: WITH rec DO BEGIN body END; For examples, see lssurfquo. A call of such a function is to be changed too: ok → ok(rec^).

An *lssurfquo* function or procedure can (and should, for clarity) be removed from the surface rules. For now, I kept them as comments.

4 Compound statements in global actions

Compound statements in global actions that combine assignments to item parameters and non-item parameters are dangerous. The rule compiler has to split such compound statements into two new ones: one with item parameters only, and one without. In many cases, the compiler won't succeed, and will give an error message. Thus, the rule writer has to do the splitting himself. I did it in the present rules. Interestingly, the surface rules did not contain as many instances of this problem as I expected.

If the rulewriter is very reluctant to split a compound statement, he/she may convert every parameter that is assigned a value in the statement, into an item parameter, at the cost of run-time memory.

Example of splitting:

replace IF boolpar THEN BEGIN itempar:=val1; nonitempar:=val2 END ELSE itempar:=val3; nonitempar:=val4 END;

into IF boolpar THEN BEGIN itempar:=val1 END ELSE BEGIN itempar:=val3 END;
IF boolpar THEN BEGIN nonitempar:=val2 END ELSE BEGIN nonitempar:=val4 END;

In a compound statement in which item parameters are assigned, between the last assignment and the END no semicolon is expected. Violation will sometimes lead to pascal compiler error messages. If no message occurs, everything is ok.

If a procedure, that is called in a global action, changes item parameters as well as non-item parameters, this procedure must be split as well. The compiler will give an error if the procedure has no side-effects, in other words if the parameters to be changed are arguments of the procedure. If a procedure is such that none of its arguments are item parameters, and if it nevertheless contains a side effect on item parameters, this is accepted by the compiler. The result is a wrong implementation: the side effect is sometimes neglected! The present surface rules are free of this phenomenon, as I was able to check in half an hour, as only a few procedures had side effects at all (Compliment). I changed the headings of the few ones that did.

5 Small notational changes

In the surface rules the help functions must be put between question marks:

?

HELP

function ok:BOOLEAN; BEGIN ... END;

function ookok:BOOLEAN; BEGIN ... END;

?

Comments have the format !(* comment *). Note the exclamation mark.

6 Advices

- Never write a procedure that has side effects.
- Give (new) item parameters names that show that it is an item parameter, e.g. verbfounditmpar.
- Be careful with compound statements. Do not mix asignments to item parameters and non-item parameters in one if-then-else, in global actions. In final actions such mixes cause no harm.