Philosophische Semantik

Manuel Bremer

Vorlesung 6

Semantic Algorithms

Examples of Algorithms of semantic justification

Just as a toy example I present some procedures in PSEUSO-CODE fashion (related to the programming language PASCAL):

```
function justify(statement): boolean;
var
    sT, gT: expression;
begin
    sT := parse(statement, singTerm);
    gT := parse(statement, genTerm);
    justify := apply(gT, identify(sT))
end.
function identify(singTerm): object;
function apply(genTerm, object): boolean;
function parse(statement, gramTyp): expression;
```

PSEUDO-CODE
JUSTIFYING A STATEMENT

Thus, the function `justify` should take us from a statement to a truth value. It does so by employing sub-functions to parse the statement for its constituent terms.
The parsing sub-function is merely syntactic and simple (as is well known from parsing natural languages).
The main step in `justify` is to apply the so-parsed general term to the result of the sub-function `identify`, which delivers the object referred to by the singular term.
For `apply` to work we need a more general function, which fetches the appropriate procedure for a general term from some lexical look-up table. This table might be thought of as our program's equivalent of a lexicon. So we need something like:

```
function lexlookup(expression): procedure;
```

`apply` has to get spelled out to a program of this type:

```
function apply(genTerm, object): boolean;
var
    p: procedure;
begin
    p := lexlookup(genTerm);
    apply := call p(object)
end.
```

PSEUDO-CODE

APPLYING A LEXICALIZED PROCEDURE

Correspondingly there has some such look-up procedure involved in `identify`.

Now, an example. Take the statement:

(α) The longest word of the hit list starts with a “b”.

`parse(`α`, singTerm)` will deliver: “the longest word of the hitlist”.

```
procedure longestWordHitList(list, OUT object);
var
    ob: object;
    int: integer;
begin
    ob := list[1];
    int := 2;
    while list < > [ ]    do
        begin
        if length(ob) < length(list[int]) then
            ob := list[int];
        int := int + 1;
        end;
    return ob
end.
```

PSEUDO-CODE

EXAMPLE OF IDENTIFICATION RULE LINKED TO A SINGULAR TERM

`parse(α, genTerm)` will deliver "( ) starts with a ‚b'".

The procedure linked to that could be:

```
procedure startsWithB(expression, OUT boolean);
var
    letter: char;
begin
    letter := expression[1];
    if  letter = 'b' then
        return true
    else
        return false
end.
```

PSEUDO-CODE

EXAMPLE OF CLASSIFICATION RULE LINKED TO A GENERAL TERM