Group	: Computational Linguistics
Topic	: formalism

Title	: **Auxiliary-Domain Compiler**
Author	: Chris Hazenberg

Doc.Nr.	: 188
Date	: October 4, 2004
Status	: concept
Supersedes	: -
Distribution	: Linguists,René Leermakers
Clearance	: Project
Keywords	: auxiliary-domain,compiler

Institute for Perception Research

Contents

1 Introduction

This document describes an auxiliary-domain compiler i.e. a program that converts a standard grammar into Pascal.The standard grammar predefines some basic datatypes and may be a support for the linguists to write M-rules. To be more exact, it contains a description of predefined keys(i.e.integers), abbreviations and records for a certain language.

The output consists of a definition file and an implementation file.These modules may be used as "inherited" ones at the ultimate Pascal version of the M-rules.

In creating the auxiliary-domain compiler, use is made of the compiler generator decribed in documents R0167 and R0172.

Section 2 deals with the syntax of the input. Section 3 describes the output. And a small example will be given in section 4.

2 Input

2.1 Syntax

The syntax for the input grammar is as follows:

```
UTT          =  LANGVERSION . [KEYSECTION] . [RECSECTION] . [ABBRSECTION]

LANGVERSION  =  language .  colon . IDENTIFIER

KEYSECTION   =  keys . leftarrow . {ARGUMENTS} . rightarrow
ARGUMENTS    =  IDENTIFIER .  equivalent . ARGLIST
ARGLIST      =  leftarrow . key . equivalent . NUMBER .
                            term . equivalent . TERMARGUMENT .
                            category . equivalent . IDENTIFIER .
                rightarrow
NUMBER       =  IDENTIFIER
TERMARGUMENT =  IDENTIFIER | abstract | PUNCTUATION

RECSECTION   =  standard . records . leftarrow . {NUMRECORDS} . rightarrow
NUMRECORDS   =  IDENTIFIER . colon . TYPESECTION . FIELDLIST
TYPESECTION  =  IDENTIFIER
FIELDLIST    =  curlyopen . {IDENTIFIER . equivalent . FIELDVALUE} .
                curlyclose
FIELDVALUE   =  IDENTIFIER |(squareopen .(squareclose |
                                          (IDENTIFIER .{comma . IDENTIFIER}.
```

```
                                                squareclose)))
```

```
ABBRSECTION  =  abbrsets . leftarrow . {NUMABBRS} . rightarrow
NUMABBRS     =  IDENTIFIER . equivalent . ABBRLIST
ABBRLIST     =  squareopen . {IDENTIFIER . comma} . IDENTIFIER .
                squareclose
```

The CAPITAL-typed strings are the non-terminals and the others the terminal strings. An IDENTIFIER consists of characters and digits.A NUMBER may only consist of digits.A PUNCTUATION may only be one of the most often used punctuations, defined on the keyboard.

The next lists enumerate the terminal symbols of the grammar:

```
leftarrow   : '<'
rightarrow  : '>'
squareopen  : '['
squareclose : ']'
curlyopen   : '{'
curlyclose  : '}'
equivalent  : '='
comma       : ','
colon       : ':'
```

and the teminal strings of the grammar:

```
abbreviations : 'ABBREVIATIONS'
abstract      : 'ABSTRACT'
category      : 'CATEGORY'
keys          : 'KEYS'
key           : 'KEY'
language      : 'LANGUAGE'
records       : 'RECORDS'
abbrsets      : 'SETS'
standard      : 'STANDARD'
term          : 'WORD'
```

The allowed punctuations are: ‘ , ’ , ” , @ , # , $, % , ^ , & , * , (,) , _ , - , + , = , { ,} , [,] , > , < , : , ; , / , | , \ , ? , ¿ , ! , ¡ , , and . .

2.2 Explanation

The input file enables the linguist to increase his domain, used in the M-rules for a certain language.The input file consists of four separate sections.

In the section **LANGUAGE** you have to define the language. In the section **KEYS** it is possible to define keys that belong to words, abstract data and punctuations.When a range for the different kinds of categories is fixed, a small extension will create the possibility to check the **KEY** and its corresponding **TERM** and **CATEGORY**. The **STANDARD RECORDS** are implemented in such a way, that it is possible to compare a certain record with this Standard Record. In the section **SETS** abbreviations of sets are defined.

When defining this auxiliary domain, attention must be paid to the following facts:

- The input file has to be named: < *Language* >: *lsauxdomain.auxdom*,where '*Language*' is the same as defined in the input file.
- The chosen language has to be one for which the *lsdomaint*-file is accessable.
- The record-types have to be declared in this *lsdomaint*-file.
- A defined **KEY** has to be an integer.
- The abbreviations are implemented as sets ; identifiers, separated by commas,between square brackets.

3 Output

The output created by the auxiliary-domain compiler,consists of a definition file (*LSAUXDOM.ENV*) and an implementation-file (*LSAUXDOM.PAS*), both Pascal versions.To create these files one has to type the following command :

```
lbuild  < language >:LSAUXDOM.OPT
```

In these files, variable names, used in other modules,are concatenated with the modulename.

3.1 Definition-file

The definition file inherits some other files:

- The file STRING.ENV enables string-manipulations.
- The file FILES.ENV enables file-manipulations.
- The file language:LSDOMAINT.ENV declares the types,used in the standard records;'language' must be stated in the auxiliary domain.

The predefined keys are converted to Pascal constants.The names of the Record-Compare functions are a concatenation of de string 'LSAUXDOM_Compare' and the Standard Record identifier.Thus, comparing a Standard Record with an other record

is possible by calling the functionname derived from the Standard Record. An external procedure WriteAbbr enables the program to handle the abbreviations.

3.2 Implementation-file

The implementation file inherits the same files for the same reasons as the definition file.The implementation file of course inherits also its environment-file.

The implementation-file consists of a few procedures.

The RecordCompare functions compares an input record with a Standard Record.

WriteAbbr writes the complete string to a predefined outputfile (of1), when dealing with its abbreviation.Note that a Case-statement is not allowed here; the selector has to be an integer.

4 A small example

The following example may illustrate how to use the auxiliary-domain and its compiler.It shows the input(auxiliary-domain) and output(implementation and definition file).

4.1 Input

The input is written in the file named: DUTCH:lsauxdomain.auxdom.

```
LANGUAGE : DUTCH

KEYS
      <  DatKey = < KEY      = 1755126
                    WORD     = dat
                    CATEGORY = RelPro
                  >
         AbstrKey = < KEY      = 80675
                      WORD     = ABSTRACT
                      CATEGORY = Noun
                    >
         DotKey = < KEY      = 120000
                    WORD     = .
                    CATEGORY = punct
                   >
```

```
      >

STANDARD RECORDS
      <
        ADJPPROP1record: ADJPPROPrecord
                    { req             = omegapol
                      class           = omegaTimeAdvClass
                      deixis          = omegadeixis
                      aspect          = omegaAspect
                      retro           = false
                      aktionsart      = stative
                      superdeixis     = omegadeixis
                      actsubcefs      = [otheradj]
                      thetaadj        = omegathetaadj
                      adjpatternefs   = []
                      PROsubject      = false
                    }
        ASP1record : ASPrecord
                    { req         = omegapol
                      thanascompl = NPcompl
                    }
      >

SETS
      <
        NAW = [NAAM, ADRES ,WOONPLAATS]
        SENT= [VERB, NOUN]
        ONE = [onlyone]
      >
```

4.2 Output

The output is created by the following command:

```
lbuild : DUTCH:lsauxdom.opt
```

4.2.1 definition

```
[ENVIRONMENT('DUTCH:lsauxdom'),
     INHERIT('GENERAL:string',
```

```
            'GENERAL:files',
            'DUTCH:lsdomaint')]

MODULE LSAUXDOM;

CONST
  LSAUXDOM_DATKEY   = 1755126;
  LSAUXDOM_ABSTRKEY = 80675;
  LSAUXDOM_DOTKEY   = 120000;
```

```
[EXTERNAL] FUNCTION LSAUXDOM_CompareADJPPROP1RECORD
                          (rec:LSDOMAINT_ADJPPROPRECORD):BOOLEAN;
           EXTERN;

[EXTERNAL] FUNCTION LSAUXDOM_CompareASP1RECORD
                          (rec:LSDOMAINT_ASPRECORD):BOOLEAN;
           EXTERN;

[EXTERNAL] PROCEDURE LSAUXDOM_WriteAbbr(VAR of1:FILES_text;
                                abbrev:STRING_string);
           EXTERN;


END. {LSAUXDOM}
```

4.2.2 implementation

```
[INHERIT ('DUTCH:lsauxdom',
          'GENERAL:string',
          'GENERAL:files',
          'DUTCH:lsdomaint')]

MODULE LSAUXDOM(output);


[GLOBAL] FUNCTION LSAUXDOM_CompareADJPPROP1RECORD
                          (rec:LSDOMAINT_ADJPPROPRECORD):BOOLEAN;
VAR Bool : BOOLEAN;
BEGIN
  Bool := TRUE;
  WITH rec DO
  BEGIN
    IF Bool THEN
      Bool:= (REQ = OMEGAPOL )
    ELSE IF Bool THEN
      Bool:= (CLASS = OMEGATIMEADVCLASS )
    ELSE IF Bool THEN
      Bool:= (DEIXIS = OMEGADEIXIS )
    ELSE IF Bool THEN
      Bool:= (ASPECT = OMEGAASPECT )
```

```
    ELSE IF Bool THEN
      Bool:= (RETRO = FALSE )
    ELSE IF Bool THEN
      Bool:= (AKTIONSART = STATIVE )
    ELSE IF Bool THEN
      Bool:= (SUPERDEIXIS = OMEGADEIXIS )
    ELSE IF Bool THEN
      Bool:= (ACTSUBCEFS = [OTHERADJ] )
    ELSE IF Bool THEN
      Bool:= (THETAADJ = OMEGATHETAADJ )
    ELSE IF Bool THEN
      Bool:= (ADJPATTERNEFS = [] )
    ELSE IF Bool THEN
      Bool:= (PROSUBJECT = FALSE );
  END;{with}
  LSAUXDOM_CompareADJPPROP1RECORD:=Bool;
END;{function}



[GLOBAL] FUNCTION LSAUXDOM_CompareASP1RECORD
                          (rec:LSDOMAINT_ASPRECORD):BOOLEAN;
VAR Bool : BOOLEAN;
BEGIN
  Bool := TRUE;
  WITH rec DO
  BEGIN
    IF Bool THEN
      Bool:= (REQ = OMEGAPOL )
    ELSE IF Bool THEN
      Bool:= (THANASCOMPL = NPCOMPL );
  END;{with}
  LSAUXDOM_CompareASP1RECORD:=Bool;
END;{function}
```

```
[GLOBAL] PROCEDURE LSAUXDOM_WriteAbbr(VAR of1:FILES_text;
                            abbrev:STRING_string);

BEGIN
  Files_Open(of1,'un_specified',Files_MaxName,5);
  IF abbrev = 'NAW' THEN
     Files_WriteStr(of1,'[NAAM,ADRES,WOONPLAATS]',Files_MaxIO,Files_MaxIO,True)
  ELSE IF abbrev = 'SENT' THEN
     Files_WriteStr(of1,'[VERB,NOUN]',Files_MaxIO,Files_MaxIO,True)
  ELSE IF abbrev = 'ONE' THEN
     Files_WriteStr(of1,'[ONLYONE]',Files_MaxIO,Files_MaxIO,True);
  Files_Close(of1);
END;

END. {LSAUXDOM}
```