

STEVIN-IRME WP6, deliverable 6.1

Report on results of incorporating idiomatic expressions into Rosetta

Nicole Grégoire
Uil-OTS, Utrecht University
Nicole.Gregoire@let.uu.nl

August 29, 2007

Contents

1	Introduction	3
2	Rosetta	4
2.1	Dictionary entries	4
2.2	Idiom formation rules	6
2.2.1	No parameterization	6
2.2.2	Parameterization	8
3	Conversion procedure	11
3.1	Without parameters	11
3.2	With parameters	14
	References	20
A	Rosetta code	21
A.1	RIDSUBNOUNTONOUN from <i>rc_idstartverbprules.mrule</i>	21
A.2	RIDALTNPFFORMATION1 from <i>rc_idstartverbprules.mrule</i>	22

1 Introduction

The original purpose of this document was to report on the results of incorporating idiomatic expressions into the Rosetta MT system (Rosetta, 1994). However, the Rosetta system was not working at the beginning of the IRME project and although one of the goals of the project was to build a running version of the Rosetta3 system, this goal has not been achieved within the set time period. This means that the purpose of this document has changed into giving an overview of the information needed to incorporate the standard lexical representation of idiomatic expressions into the Rosetta system.¹ It must be noted the information given in this document is solely based on a study of the available Rosetta code and documentation and that none of the steps described has been tested.

The treatment of idioms in the Rosetta system is discussed in section 2, whereas section 3 describes the basic procedure that needs to be followed when converting the standard representation to the representation required by Rosetta. It is assumed that readers of this document have knowledge of the description fields that are part of standard representation of MWEs and MWE patterns. The standard representation is described in detail in the *Encoding Protocol* (Grégoire, 2007).

¹The Rosetta system includes treatments for three types of multiword expressions, viz. fixed expressions, semi-idioms and idioms. This document solely focuses on the incorporation of idioms.

2 Rosetta

The Rosetta MT system is the result of seven years of research on machine translation started in 1985 at the Philips Research Laboratories in Eindhoven, and is meant to translate between English, Dutch and Spanish. The type of grammar used in Rosetta is called *M-grammar*, a computationally feasible variant of *Montague Grammar*. A general overview of M-grammars is given in Schenk (1994) and Rosetta (1994).

Incorporation of idiomatic expressions in Rosetta is possible, since the system contains a powerful method for dealing with idioms. Successful treatment of idioms comprises the presence of necessary lexical entries, addressed in section 2.1, and specific idiom formation rules, discussed in section 2.2. Illustrations are given using the example *de plaat poetsen* (lit. ‘to polish the plate’, id. ‘to clear off’)

2.1 Dictionary entries

The following Rosetta dictionaries require one or more entries for each idiom:

1. **BLEX**, the general lexicon, requires a lexical entry for each component of the idiom plus a lexical entry for the whole expression, which is represented as a copy of the head of the expression. The lexical entries for *plaat*, *poetsen*, and *plaatpoetsen* are respectively:

```
$PLAATBNOUNKEYBNOUN(  
  {req:} [ omegapol, negpol, pospol],  
  {env:} [ omegapol, negpol, pospol],  
  {dimforms:} [ jeDim],  
  {pluralforms:} [ enPlural],  
  {genders:} [ femgender, mascgender],  
  {class:} omegaTimeAdvClass,  
  {deixis:} omegadeixis,  
  {aspect:} omegaAspect,  
  {retro:} FALSE,  
  {sexes:} [ ],  
  {subcs:} [ othernoun],  
  {temporal:} TRUE,  
  {possgeni:} FALSE,  
  {animate:} noanimate,  
  {human:} NoHuman,  
  {posscomas:} [ count],  
  {thetanp:} omegathetanp,  
  {nounpatterns:} [ ],  
  {prepkey:} 0,  
  {personal:} TRUE  
);  
$S_AV_00_POETSBVERB(  
  {req:} [ omegapol, negpol, pospol],  
  {env:} [ omegapol, negpol, pospol],  
  {conjclasses:} [ 3],  
  {particle:} 0,
```

```

{possvoices:} [ DoorActive, Passive, Active],
{reflexivity:} notreflexive,
{synvps:} [ vpid1, synNP],
{thetavp:} vp120,
{adjuncts:} [ ResAP],
{CaseAssigner:} TRUE,
{subc:} Mainverb,
{perfauxs:} [ hebaux],
{prepkey1:} 0,
{prepkey2:} 0,
{controller:} none,
{verbraiser:} noVR,
{IPP:} NOIPP,
{classes:} [ durativeclass]
);

```

```

$PLAATPOETS_SKEYBVERB(
  {req:} [ omegapol, negpol, pospol],
  {env:} [ omegapol, negpol, pospol],
  {conjclasses:} [ 3],
  {particle:} 0,
  {possvoices:} [ DoorActive, Passive, Active],
  {reflexivity:} notreflexive,
  {synvps:} [ vpid1, synNP],
  {thetavp:} vp120,
  {adjuncts:} [ ResAP],
  {CaseAssigner:} TRUE,
  {subc:} Mainverb,
  {perfauxs:} [ hebaux],
  {prepkey1:} 0,
  {prepkey2:} 0,
  {controller:} none,
  {verbraiser:} noVR,
  {IPP:} NOIPP,
  {classes:} [ durativeclass]
);

```

2. **IDDICT**, the idiom dictionary, requires the idiom entry in the format **skey1 number_of_components skey2...skey_n skey_n+1 idpattern**, where *n* is the number of components. An example is:

```
S_AV_00_POETS 2 S_AV_00_POETS PLAATBNOUNKEY PLAATPOETS_SKEY vpid1
```

The syntactic key *skey1* is the lexical entry under which the idiom is listed in **IDDICT**. *number_of_components* refers to the number of lexical components that the idiom contains (disregarding articles). The syntactic keys *skey2...skey_n* refer to the **BLEX** entries of the individual components, whereas *skey_n+1* refers to the **BLEX** entry of the whole expression. The *idpattern* refers to the name of the idiom pattern, e.g.

vpid1 is used to identify idioms consisting of a singular, definite NP which is a fixed part of the idiom in object position, and a variable in subject position.

3. **ILDICT**, requires a mapping between the skey of the idiom and its mkey, which is the name of the meaning of the expression:

```
PLAATPOETS_SKEY PLAATPOETS_MKEY 0 0 BF
```

2.2 Idiom formation rules

Besides dictionary entries, Rosetta requires a so-called *M-rule* for each idiom type in order to define the idiom surface tree.² The *idpattern* links the entry in **IDDICT** to the corresponding M-rule.

An M-rule may contain alternation rules which take the *idpattern* as a parameter. Section 2.2.1 first describes an M-rule without parameterization. The working of parameterization is discussed in section 2.2.2.

2.2.1 No parameterization

Below the M-rule is listed for expressions with *idpattern vpid1*, such as *de plaat poetsen*:³

```
<m1: I1::SUBVERB{SUBVERBREC1}[head/BVERB(KEY1){BVERBREC1}]
  m2: T1
>

7: < m:  IDFORMATION/TIDCLAUSETOVPPROP
6:      [CLAUSETOSENTENCE/RSUBSTITUTION1{LEVEL:X1
          SUBST:2}
5:      [IDFORMATION/TIDVPPROPTOCLAUSE
4:      [VERBPPROPFORMATION/RVERBPATTERN1
3:      [VERBPPROPFORMATION/RSTARTVPPROP120
2:      [VERBDERIVATION/RBVERBTOSUB
1:      [BLEX/BVERB(KEY2)],
1a:     BLEX/I2:T1,
1b:     BLEX/NPVAR(X1){NPVARrec1}
      ]
      ]
      ],
      NPFORMATION/RNPFORMATION4
      [CNFORMATION/RCNFORMATION1
      [CNFORMATION/RSUBNOUNTONOUN1
      [NPDERIVATION/BNOUNTOSUBNOUN
      [BLEX/BNOUN(KEY3)]
      ]
      ]
```

²The M-rules for existing entries are listed in *rc_idstartverbprules.mrule* and *rc_idstartverbprules2.mrule*. For new types of idioms new M-rules must be defined.

³The original rule is rule **RIDDERIV1**, which works for two idiom types, viz. for idioms with *idpattern vpid1* and for idioms with *idpattern vpid25*. The example rule has been altered to serve as an example for a non-parameterized rule. For expository reasons, solely the generative part of the rule is shown.

```

    ]
  ]
]
>
MATCHCONDITIONS
<
  I1: SUBVERBREC1.thetavp = vp120
  I2: T1.CAT IN AUX_VARCATSET
  m2: T1.CAT IN AUX_VARCATSET
>
COMP
<
  C1: COMPINIDICT(KEY1, VPID1)
  A1: <KEY2, KEY3> := COMPGETIDICT (KEY1, VPID1);
>
&

```

The input model, $m1$: and $m2$:, consists of a subverb specifying the idiomatic key and the models of the free arguments of the idiom, KEY1 and T1, respectively. The output model, m :, consists of the syntactic derivation tree that specifies the rules which should be applied to derive the correct canonical S-tree of the idiom. Each line of the output model is represented as either:

- *SUBGRAMMAR/RULE*, e.g. NPFORMATION/RNPFORMATION4, or
- *DICTIONARY/BASIC_EXPRESSION*, e.g. BLEX/BVERB(KEY2)].

In the subgrammar IDFORMATION all rules and transformations that are specific to idioms have to be specified. The set of rules that form the NP is discussed in section 2.2.2. The rules that need to be applied to derive the correct canonical S-tree of the idiom are explained line by line below:

- 1:** describes a basic expressions of category BVERB with skey KEY2, which is determined using IDICT and which takes two arguments. The first argument is defined in *1a*:, and the second argument, which needs to be an NP, is defined in *1b*:.
- 2:** rule RBVERBTOSUB makes S-tree SUB out of BVERB.
- 3:** rule RSTARTVPPROP120 combines SUB with a subject and an unspecified argument.
- 4:** rule RVERBPATTERN1 categorizes the complement as the direct object.
- 5:** rule TIDVPPROPTOCLAUSE changes the category of the clause from VPPROP to CLAUSE.
- 6:** rule RSUBSTITUTION1 substitutes *de plaat* for NPVAR with index $x1$ in CLAUSE.
- 7:** rule TIDCLAUSETOVPPROP changes the category of the clause from CLAUSE to VPPROP resulting in *x de plaat poets*.

The rules TIDVPPROPTOCLAUSE and TIDCLAUSETOVPPROP are idiom specific, and introduced in order to apply existing rules in a normal way.

In the compositional conditions the function COMPINIDDICT checks whether the idiom is in the dictionary with KEY1 and VPID1. If this is the case, then in the compositional actions the function COMPGETIDDICT searches for the keys of the leaves of the idiom in the dictionary on the basis of KEY1 and VPID1. These keys are assigned to KEY2 and KEY3.

Having a group of expressions that meet the criteria of *vpid1*, i.e. being a verbal idiom containing a singular, definite NP in object position, and a variable in subject position, it takes a minimal amount of work to incorporate this group of expressions in the Rosetta system, see section 3.

In the *MWE Lexicon for Dutch* MWEs are also grouped according to their characteristics (or pattern). MWEs with the same pattern form so-called Equivalence Classes (Grégoire, 2007). The ECs in the standard lexicon, however, contain MWEs that may differ locally, i.e. whereas the description of a group of expressions with the same *idpattern* in Rosetta includes, inter alia, the number and definiteness of the NP, which makes the description rather specific, the description of an EC in the *MWE Lexicon for Dutch* solely comprises the category of the head of an expression, the category of the complements it takes, and the dependency between the constituents. For example, the description of the EC with the PATTERN_NAME EC1 is ‘Expressions headed by a verb, taking a direct object consisting of a fixed determiner and an unmodifiable noun.’

EC1 contains the MWE *de plaat poetsen*, but also *de benen nemen* (‘to escape’), which differs from *de plaat poetsen* in the form of the noun (singular vs. plural), and *bergen verzetten* (‘move mountains’) which contains a determinerless NP. The components of the expressions are represented in the CL (component list) in their non-inflected form. Parameters are used to specify the full form characteristics of each component.⁴

The parameters need to be taken into account in the conversion procedure, i.e. for each parameter alternative rules should be created so that the same M-rule can be used for expressions with the same PATTERN_NAME.

2.2.2 Parameterization

In this section, the creation of alternative rules is illustrated for various NP combinations. An NP that consists of a definite determiner and a singular noun, such as *de plaat*, can be derived using the following rules:

```

5: NPFORMATION/RNPFORMATION4
4:           [CNFORMATION/RCNFORMATION1
3:           [CNFORMATION/RSUBNOUNTONOUN1
2:           [NPDERIVATION/BNOUNTOSUBNOUN
1:           [BLEX/BNOUN(KEY3) ]
              ]
            ]
          ]

```

⁴The term *parameter* is a feature and can be defined as an occurrence of the pair <parameter category,parameter value>, where *parameter category* refers to the aspect we parameterize, and *parameter value* to the value a parameter category takes. Examples of parameters are <nnum,sg> for singular nouns, <afrm,sup> for superlative adjectives, <vfrm,part> for particle verbs (Grégoire, 2006).

- 1: describes a basic expression of category BNOUN with skey KEY3, which is determined using IDDICT. The features of KEY3 are extracted from BLEX and linked to BNOUN.
- 2: rule BNOUNTOSUBNOUN makes S-tree SUBNOUN out of BNOUN.
- 3: rule RSUBNOUNTONOUN1 makes NOUN out of singular count SUBNOUN.
- 4: rule RCNFORMATION1 creates CN out of NOUN.
- 5: rule RNPFORMATION4 is used for NP creation. In this example the rule makes an NP out of CN with head NOUN and introduces ‘de’ (or ‘het’).

In order to use the same M-rule for all possible NP combinations, e.g. plural nouns, determinerless NPS, etc., *alternative rules* should be created that depend on the idpattern, i.e. the description, of an idiom. An example of alternative rules covering both *de plaat poetsen* and *bergen verzetten* with the same M-rule is given below:

```

5: IDFORMATION/RIDALTNPFORMATION1{VPID}
4:           [CNFORMATION/RCNFORMATION1
3:           [IDFORMATION/RIDSUBNOUNTONOUN{VPID}
2:           [NPDERIVATION/BNOUNTOSUBNOUN
1:           [BLEX/BNOUN(KEY3) ]
           ]
           ]
           ]

```

The alternative rules are rule RIDSUBNOUNTONOUN{VPID} in line 3 and rule RIDALTNPFORMATION1{VPID} in line 5:

- Rule RIDSUBNOUNTONOUN, see appendix A.1, alternates between:
 1. rule RSUBNOUNTONOUN1, which sets the number SINGULAR to NOUN out of a singular count SUBNOUN, and
 2. rule RSUBNOUNTONOUN2, which sets the number PLURAL to NOUN out of a plural count SUBNOUN, and the number SINGULAR to NOUN out of a mass SUBNOUN.
- Rule RIDALTNPFORMATION1, see appendix A.2, alternates between:
 - rule RIDNPFORMATION2, which makes a bare NP out of a CN in idiomatic configurations where determinerless NPs are allowed even if the NOUN is not plural or mass, and
 - rule RNPFORMATION4, which makes an NP out of CN with head NOUN and introduces ‘de’ (or ‘het’).

Rule RIDNPFORMATION2 is a minor rule and created just for idiomatic expressions.

The choice for the rule that should be applied is determined in the COMP condition (in generation mode) of the alternative rule. For example, see the COMP conditions of the rule RIDALTNPFORMATION1 (copied from appendix A.2):

COMP

```
<
  C1: vpid * AUX_detlessNPvps <> []
  A1: @
>
```

COMP

```
<
  C1: vpid * AUX_detNPvps <> []
  A1: @
>
```

The condition *vpid * AUX_detlessNPvps <> []* checks whether the intersection (*) of the set *vpid* and the set *AUX_detlessNPvps* is not empty. The latter set contains the idpatterns that consists of a determinerless NP in direct object position. If this condition is TRUE then SUBRULE (* 1 *) is applied. The output model of SUBRULE (* 1 *) states T2: IDFORMATION/RIDNPFORFORMATION2[BLEX/T3], which means that rule RIDNPFORFORMATION2 will be applied.

The condition *vpid * AUX_detNPvps <> []* checks whether the intersection (*) of the set *vpid* and the set *AUX_detNPvps* is not empty. The set *AUX_detNPvps* contains the idpatterns that consists of an NP with determiner in direct object position. If this condition is TRUE then SUBRULE (* 2 *) is applied. The output model of SUBRULE (* 2 *) states T2: NPFORFORMATION/RNPFORFORMATION4[BLEX/T3], which means that rule RNPFORFORMATION4 will be applied.

The sets *AUX_detlessNPvps* and *AUX_detNPvps* are defined in LSAUXDOMAIN.AUXDOM:

```
SETTYPE synpatternSETtype
detlessNPvps          = [
vpid14, vpid20, vpid25
] (*idiom rules to make NP without determiner*)
```

```
SETTYPE synpatternSETtype
detNPvps              = [
vpid15, vpid19, vpid16, vpid1
] (*idiom rules to make NP with determiner*)
```

The next section shows how the idiom formation rules are used in the conversion procedure.

3 Conversion procedure

Although we cannot report on an actual conversion of the standard representation to the representation required by Rosetta, we briefly describe some steps that are needed for a conversion in this section.

For a successful treatment of idioms in the Rosetta systems the following adaptations need to be made:⁵

- A derivation tree has to be created by which the syntactic surface tree can be generated.
- The lexical components of the idiom and an entry of the idiom need to be present in BLEX.
- A lexical entry needs to be present in IDDICT.
- In the case of parameterization: the idpattern needs to be added to relevant sets in LSAUXDOMAIN.AUXDOM.

The description fields that are needed from the standard representation are:⁶

- PATTERN_NAME
- CL
- EXAMPLE

The proposed procedure is based on Odijk (2003). Since the presence of parameters in the standard representation complicates the conversion procedure, we will start with describing the procedure without taking into account parameters, i.e. converting solely expressions with exactly the same pattern, such as *de plaat poetsen*, *de boot missen* ('miss the boat'), etc.

3.1 Without parameters

The conversion procedure consists of two parts: a manual part and an automatic part.

Manual part The manual part has to be carried out once for each (non-parameterized) EC and consists of ten steps:

1. Set a unique identifier IDPAT for the idiom type.
2. Select one instance X with the chosen PATTERN_NAME.
3. Parse the example sentence of X , yielding a set of parses. To choose the correct parse, the pattern description of the EC should be consulted. The chosen parse is the *Reference Parse* (RP).

⁵We abstract away from the meaning of the expressions.

⁶The required description fields can be found in *mwes.tsv* in the package *MWE lexicon for Dutch.zip*

4. Create the M-rule that contains the derivation tree of the idiom structure on the basis of the RP. The IDPAT should be specified in the COMP and DECOMP conditions of the M-rule.
5. Determine the list of unique identifiers of the lexical items used in the idiom, using the derived idiom structure, yielding the *Component ID List* (CIL).
6. Select a unique identifier for the idiom, yielding the *Idiom Identifier* (IID).
7. Define a transformation to relate the CL of X and CIL.
8. Apply this transformation to the CL, yielding the *Transformed CL* (TCL) and check that the citation form of each lexical item equals the corresponding element on the CIL.
9. Determine the head (HD) of the expression and create a copy of its lexical entry with IID as identifier in BLEX.
10. Create a template for new IDICT entries, using the format **HD *number_of_components* CIL_1...CIL_n IID IDPAT**, where:
 - (a) *number_of_components* refers to the number of components in CIL, and
 - (b) CIL_1...CIL_n refer to the components of CIL, where n is the number of components.

The *number_of_components* and IDPAT should be filled in this step, whereas the other variables are filled in the automatic part.

Illustration To illustrate, we apply the manual part to expressions with PATTERN_NAME EC1:

1. IDPAT = *vpid1*
2. Select one instance X with PATTERN_NAME EC1 : $X = de\ plaat\ poetsen$
3. Parsing the example sentence *hij heeft de plaat gepoetst*, yields a set of candidate RPs. The correct RP is chosen.
4. The RP is changed into the M-rule that contains the derivation tree of the idiom structure. IDPAT is specified in the COMP part. See the M-rule specified in section 2.2.1.
5. We can use the keys of the resulting tree to determine the CIL:
<S_AV_00.POETS,PLAATBNOUNKEY>, in this order.
6. IID = *idiom1*
7. The citation forms listed in the CL (*de plaat poetsen*) can be brought in correspondence with the CIL by applying the transformation $1\ 2\ 3 \rightarrow 3\ 2$, i.e. delete the first element and reverse the remaining list.
8. Applying this transformation turns the CL *de plaat poetsen* into the TCL *poetsen plaat*. The citation forms of the CIL correspond to the elements on the TCL:

- citation form (S_AV_00_POETS) = *poetsen*
 - citation form (PLAATBNOUNKEY) = *plaat*
9. The head (HD) of the expression is the first component in the CIL: S_AV_00_POETS. A copy of its lexical entry with IID *idiom1* is created in BLEX.
 10. The template for new IDDICT entries is: HD 2 CIL_1 CIL_2 IID vpid1 → S_AV_00_POETS 2 S_AV_00_POETS PLAATBNOUNKEY idiom1 vpid1

In this way we have obtained a procedure to convert expressions with PATTERN_NAME *EC1* represented in the standard format into the structure required in the Rosetta system. After the manual work is done for one instance of an EC, the transformation of all other members of the EC can be done in a fully automatic manner.

Automatic part The automatic part of the conversion procedure consists of seven steps:

1. Parse the example sentence of the expression and check that the output is identical to the RP for the example sentence used in the manual step, except for the lexical items.
2. Select the component IDs from the parse tree, in order to obtain the CIL.
3. Set the IID.
4. Apply the idiom component transformation to the CL, in order to obtain the TCL.
5. Check that the citation form of each item in the CIL equals the corresponding element on the TCL.
6. Create a copy of the lexical entry of the head of the expression with IID as identifier.
7. Fill in the template to create a new IDDICT entry.

Illustration The automatic part is applied to each expression with PATTERN_NAME *EC1*. As illustration, we apply it to the idiom *de boot missen*. We follow the steps described above:

1. Parsing the example sentence *hij heeft de boot gemist* should lead to a syntactic D-tree that is identical to the one for *hij heeft plaat gepoetst*, except for the skeys.
2. CIL = <S_AV_00_MIS,S_AN_00_BOOT>, in this order.
3. IID = idiom2
4. The idiom component list transformation applied to the CL *de boot missen* yields *missen boot*.
5. The citation form of each item on the CIL equals the corresponding element on the TCL:
 - citation form (S_AV_00_MIS) = *missen*
 - citation form (S_AN_00_BOOT) = *boot*

6. HD = S_AV_00_MIS. A copy of its lexical entry with IID *idiom2* is created in BLEX.
7. New IDICT entry is:
S_AV_00_MIS 2 S_AV_00_MIS S_AN_00_BOOT idiom2 vpid1

3.2 With parameters

Although all parameters and determiners need to be taken into account in the conversion procedure, we will focus here on alternation in NP formation. As stated in section 2.2.2, alternative rules should be used to cover expressions within the same EC, but with different parameter combinations.⁷ Take for example the expressions *de plaat poetsen* and *wortel schieten*. The difference between the expressions is explicitly visible in the CL: *de plaat[sg] poetsen* vs. *EMP wortel[sg] schieten*. Both expressions contain a singular count noun, but the formation of the NP *de plaat* includes the determiner *de*, whereas the formation of the NP *wortel* includes the selection of no determiner. To cover these difference with one derivation tree, an alternative NP formation rule is substituted for the fixed NP formation rule. The alternative rule is RIDALTNPFORMATION1, see appendix A.2, and originally alternates between:

- rule RIDNPFORMATION2, which makes a bare NP out of a CN in idiomatic configurations where determinerless NPs are allowed even if the NOUN is not plural or mass, and
- rule RNPFORMATION4, which makes an NP out of CN with head NOUN and introduces ‘de’ (or ‘het’).

This means that instead of having a derivation tree for expressions such as *de plaat poetsen* and another derivation tree for expressions such as *wortel schieten*, we have one derivation tree taking the idpattern of the expression as a parameter to trigger the correct NP formation rule.

Before the manual conversion of an EC can start, each parameter combination needs to be linked to a Rosetta rule. Furthermore, each Rosetta rule should be specified in the alternative rule, in this case the alternative rule RIDALTNPFORMATION1, including defining the appropriate set in the corresponding COMP and DECOMP conditions. A parameter combination should contain a value for each component in the CL. Determiners must be represented as in the CL and if a component does not include a parameter a dash (-) should be used.

Table 1 shows the parameter combinations with the corresponding Rosetta rules and set types for expressions *de plaat poetsen* and *wortel schieten*, both instances of EC1.

parameter combi	Rosetta rule	set type
EMP [sg] -	RIDNPFORMATION2	AUX_detlessNPvps
de [sg] -	RNPFORMATION4	AUX_detNPvps
het [sg] -	RNPFORMATION4	AUX_detNPvps

Table 1: Parameter combinations with the corresponding Rosetta rules and set types.

⁷A *parameter combination* is the set of parameters present in the CL of an expression. For convenience we speak of *parameter combination*, but the combination also includes determiners.

Manual part Especially the manual step of the conversion procedure is more complicated with parameters:

1. Set a unique identifier IDPAT for each parameter combination defined in the pre-manual step, see table 1. Parameter combinations that correspond to the same rule should be assigned an identical IDPAT.
2. Add each IDPAT to the relevant sets in LSAUXDOMAIN.AUXDOM.
3. Select one instance X with the chosen PATTERN_NAME.
4. Parse the example sentence of X , yielding a set of parses. To choose the correct parse, the pattern description of the EC should be consulted. The chosen parse is the *Reference Parse* (RP).
5. Create the M-rule that contains the derivation tree of the idiom structure on the basis of the RP. Make sure relevant alternative idiom formation rules taking the parameter VPID are inserted in the tree. Each IDPAT should be specified in the COMP and DECOMP conditions of the M-rule.
6. Determine the list of unique identifiers of the lexical items used in the idiom, using the derived idiom structure, yielding the *Component ID List* (CIL).
7. Select a unique identifier for the idiom, yielding the *Idiom Identifier* (IID).
8. Define a transformation to relate CL of X and CIL.
9. Apply this transformation to the CL, yielding the *Transformed CL* (TCL) and check that the citation form of each lexical item equals the corresponding element on the CIL.
10. Determine the head (HD) of the expression and create a copy of its lexical entry with IID as identifier in BLEX.
11. Determine the parameter combination of X using the CL.
12. Use the parameter combination to select the correct IDPAT, yielding IDPATX.
13. Create a template for new IDDICT entries, using the format **HD *number_of_components* CIL_1...CIL_n IID IDPATX**, where:
 - (a) *number_of_components* refers to the number of components in CIL, and
 - (b) CIL_1...CIL_n refer to the components of CIL, where n is the number of components.

The *number_of_components* and IDPAT should be filled in this step, whereas the other variables are filled in the automatic part.

Illustration To illustrate, we apply the manual part to expressions with PATTERN_NAME EC1 and with one of the parameter combinations listed above:

	parameter combi	rule	set type	IDPAT
1.	EMP [sg] -	RIDNPFORMATION2	AUX_detlessNPvps	vpid2
	de [sg] -	RNPFORMATION4	AUX_detNPvps	vpid1
	het [sg] -	RNPFORMATION4	AUX_detNPvps	vpid1

2. Add each IDPAT to the relevant sets in LSAUXDOMAIN.AUXDOM:

```
SETTYPE synpatternSETtype
detlessNPvps      = [vpid2] (*idiom rules to make NP without determiner*)
```

```
SETTYPE synpatternSETtype
detNPvps          = [vpid1] (*idiom rules to make NP with determiner*)
```

3. Select one instance X with PATTERN_NAME EC1 : $X = de\ plaat\ poetsen$
4. Parsing the example sentence *hij heeft de plaat gepoetst*, yields a set of candidate RPs. The correct RP is chosen.
5. The RP is changed into the M-rule that contains the derivation tree of the idiom structure. To account for the parameter combinations, the alternative rule RIDALTNPFORMATION1 is substituted for the standard NP formation rule and each IDPAT is added to the COMP part, yielding:

```
<m1: I1::SUBVERB{SUBVERBREC1}[head/BVERB(KEY1){BVERBREC1}]
  m2: T1
>

< m:  IDFORMATION/TIDCLAUSETOVPPROP
      [CLAUSETOSENTENCE/RSUBSTITUTION1{LEVEL:X1
      SUBST:2}
      [IDFORMATION/TIDVPPROPTOCLAUSE
      [VERBPPROPFORMATION/RVERBPATTERN1
      [VERBPPROPFORMATION/RSTARTVPPROP120
      [VERBDERIVATION/RBVERBTOSUB
      [BLEX/BVERB(KEY2)],
      BLEX/I2:T1,
      BLEX/NPVAR(X1){NPVARrec1}
      ]
      ]
      ],
      IDFORMATION/RIDALTNPFORMATION1{VPID}  --> substitution
      [CNFORMATION/RCNFORMATION1
      [CNFORMATION/RSUBNOUNTONOUN1
      [NPDERIVATION/BNOUNTOSUBNOUN
      [BLEX/BNOUN(KEY3)]
      ]
      ]
```

```

    ]
  ]
]
>
MATCHCONDITIONS
<
  I1: SUBVERBREC1.thetavp = vp120
  I2: T1.CAT IN AUX_VARCATSET
  m2: T1.CAT IN AUX_VARCATSET
>
COMP
<
  C1: COMPINIDDICT(KEY1, VPID1)
  A1: <KEY2, KEY3> := COMPGETIDDICT (KEY1, VPID1);
  C2: COMPINIDDICT(KEY1, VPID2)    --> added
  A2: <KEY2, KEY3> := COMPGETIDDICT (KEY1, VPID2);    --> added
>
&

```

6. We can use the skeys of the resulting tree to determine the CIL:
 <S_AV_00_POETS,PLAATBNOUNKEY>, in this order.
7. IID = idiom1
8. The citation forms listed in the CL (*de plaat poetsen*) can be brought in correspondence with the CIL by applying the transformation $1\ 2\ 3 \rightarrow 3\ 2$, i.e. delete the first element and reverse the remaining list.
9. Applying this transformation turns the CL *de plaat poetsen* into the TCL *poetsen plaat*. The citation forms of the CIL correspond to the elements on the TCL:
 - citation form (S_AV_00_POETS) = *poetsen*
 - citation form (PLAATBNOUNKEY) = *plaat*
10. The head (HD) of the expression is the first component in the CIL: S_AV_00_POETS. A copy of its lexical entry with IID *idiom1* is created in BLEX.
11. The parameter combination of *X* is: de [sg] -
12. IDPATX = vpid1
13. The template for new IDDICT entries is: HD 2 CIL_1 CIL_2 IID IDPATX →
 S_AV_00_POETS 2 S_AV_00_POETS PLAATBNOUNKEY idiom1 vpid1

Automatic part The automatic part of the conversion procedure consists of seven steps:

1. Parse the example sentence of the expression, and check that the output is identical to the RP for the example sentence used in the manual step, except for the lexical items and NP formation rule, which should be identical to one of the rules in the alternation rule RIDALTNPFFORMATION1.

2. Select the component IDs from the parse tree, in order to obtain the CIL.
3. Set the IID.
4. Apply the idiom component transformation to the CL, in order to obtain the TCL.
5. Check that the citation form of each item in the CIL equals the corresponding element on the TCL.
6. Create a copy of the lexical entry of the head of the expression with IID as identifier.
7. Determine the parameter combination of the expression using the CL.
8. Use the parameter combination to select the correct IDPAT, yielding IDPATX.
9. Fill in the template to create a new IDDICT entry.

Illustration The automatic part is applied to each expression with PATTERN_NAME *EC1* and with a parameter combination as listed above. As illustration, we apply the automatic steps to the idiom *wortel schieten*:

1. Parsing the example sentence *hij heeft wortel geschoten* should lead to a syntactic D-tree that is identical to the one for *hij heeft de plaat gepoetst*, except for the skeys and except for the NP formation rule, which should be identical to one of the rules in the alternation rule RIDALTNPFFORMATION1.
2. CIL = <S_AV_00_SCHIET,S_AN_00_WORTEL>, in this order.
3. IID = idiom3
4. The idiom component list transformation applied to the CL *EMP wortel schieten* yields *schieten wortel*.
5. The citation form of each item on the CIL equals the corresponding element on the TCL:
 - citation form (S_AV_00_SCHIET) = *schieten*
 - citation form (S_AN_00_WORTEL) = *wortel*
6. HD = S_AV_00_SCHIET. A copy of its lexical entry with IID *idiom3* is created in BLEX.
7. The parameter combination is: EMP [sg] -
8. IDPATX = vpid2
9. New IDDICT entry is:


```
S_AV_00_SCHIET 2 S_AV_00_SCHIET S_AN_00_WORTEL idiom3 vpid2
```

The illustration given above includes solely three parameter combinations and two id-patterns yielding one alternation rule, and is a rather simple example of how parameters should be dealt with in the conversion procedure. Although we will not discuss the treatment of other parameters and the occurrence of multiple alternative rules in one derivation,

we briefly show how the alternative rule RIDALTNPFORMATION1 can be changed in order to deal with more parameter combinations included in EC1.

Table 2 gives an overview of various parameter combinations, the corresponding rule in Rosetta and a description.⁸ It should be noted that for these rules the parameters on the noun are often not relevant, hence the dash in the position of the noun.

parameter combi	rule	task (copied from Rosetta)
INDEF - -	RNPFORMATION1	making an NP out of DETP (simplex or partitive) and CN with head NOUN
de - -	RNPFORMATION4	making an NP out of ‘de’ or ‘het’ and CN with head NOUN
een - -	RNPFORMATION4a	making an NP out of ‘een’ and CN with head NOUN
zijn - -	RIDNPFORMATION	making an NP that contains a bound NPVAR
EMP [het][sg] -	RIDNPFORMATION2	making a bare NP out of a CN in idiomatic configurations where detless NPs are allowed even if the NOUN is not plural or mass.
PNP - -	RIDNPFORMATION3	making an NP that contains an NPVAR

Table 2: Various parameter combinations, the corresponding rule in Rosetta and a description

The rules in table 2 should be added as new subrules to rule RIDALTNPFORMATION1, see appendix A.2. The addition of each new subrule leads to the creation of a new idpattern in the conversion procedure.

The procedure described in this section assumes that the Rosetta lexicon contains a correct lexical entry for all the words that are part of the idioms in the standard lexicon, furthermore it assumes that all the example sentences can be parsed by the Rosetta system and that the set of parses at least contains the right parse. These assumptions may be idealistic, but whenever the Rosetta system cannot deal with an idiom, whether its lexicon does not contain the correct lexical items or the grammar cannot parse the example sentence, it is the responsibility of the system developer to extend the system’s lexicon and/or grammar so that it does yield a correct parse.

⁸In the standard representation determiners are represented in the form they take in the idiom, except for indefinites, empty determiners, and possessive NPs, which are represented with the labels INDEF, EMP and PNP respectively.

References

- Grégoire, N. (2006), Elaborating the parameterized equivalence class method for Dutch, in N. Calzolari (ed.), *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, ELRA, Genoa, Italy, pp. 1894–99.
- Grégoire, N. (2007), MWE lexicon for Dutch: Encoding protocol, *Technical report*, STEVIN IRME.
- Odiijk, J. (2003), Towards a standard for multi-word expressions. ISLE Project Report.
- Rosetta, M. T. (1994), *Compositional Translation*, Kluwer Academic Publishers, Dordrecht.
- Schenk, A. (1994), *Idioms and collocations in compositional grammars*, PhD thesis, University of Utrecht.

A Rosetta code

A.1 RIDSUBNOUNTONOUN from *rc_idstartverbprules.mrule*

```
%RULE RIDSUBNOUNTONOUN
```

```
<m1: T1
```

```
>
```

```
< m: T2
```

```
>
```

```
PARAMETERS
```

```
<vpid:synpatternSETtype
```

```
>
```

```
<
```

```
  SUBRULE (* 1 *)
```

```
    <T1: T3
```

```
    >
```

```
    <T2: CNFORMATION/RSUBNOUNTONOUN1 [BLEX/T3]
```

```
    >
```

```
COMP
```

```
<
```

```
  C1: vpid * AUX_countnounvps <> []
```

```
  A1: @
```

```
>
```

```
DECOMP
```

```
<
```

```
  C1: TRUE
```

```
  A1: vpid := AUX_countnounvps
```

```
>
```

```
  SUBRULE (* 2 *)
```

```
    <T1: T3
```

```
    >
```

```
    <T2: CNFORMATION/RSUBNOUNTONOUN2{numberpar:singular} [BLEX/T3]
```

```
    >
```

```
COMP
```

```
<
```

```
  C1: vpid * AUX_massnounvps <> []
```

```
  A1: @
```

```
>
```

```
DECOMP
```

```
<
```

```
  C1: TRUE
```

```
  A1: vpid := AUX_massnounvps
```

```
>
```

```
>
```

A.2 RIDALTNPFORMATION1 from *rc_idstartverbprules.mrule*

```
%RULE RIDALTNPFORMATION1
<m1: T1
>
< m: T2
>
PARAMETERS
<vpid:synpatternSETtype
>
<
  SUBRULE (* 1 *)
    <T1: T3
    >
    <T2: IDFORMATION/RIDNPFORMATION2[BLEX/T3]
    >
COMP
<
  C1: vpid * AUX_detlessNPvps <> []
  A1: @
>
DECOMP
<
  C1: TRUE
  A1: vpid := AUX_detlessNPvps
>

  SUBRULE (* 2 *)
    <T1: T3
    >
    <T2: NPFORMATION/RNPFORMATION4[BLEX/T3]
    >
COMP
<
  C1: vpid * AUX_detNPvps <> []
  A1: @
>
DECOMP
<
  C1: TRUE
  A1: vpid := AUX_detNPvps
>
>
```