Group	: Computational Linguistics
Topic	: Rosetta3.doc.software

Title	: **The Rosetta Module Index**
Author	: Joep Rous and René Leermakers

Doc.Nr.	: 303
Date	: October 4, 2004
Status	: concept
Supersedes	: -
Distribution	: Project
Clearance	: Project
Keywords	:

Institute for Perception Research

Contents

1 Introduction

This document gives an overview of the modules which together make up the Rosetta software. The structure of the document reflects the structure of the Rosetta Translator Generator. The Translator Generator consists of three main parts

kernel The implementation of the Rosetta components (e.g. A-MORPH, M-PARSER).

compiler generator The compiler generator generates a compiler from a given PRosE dialect specification.

PRosE **dialect specification** Specifications of all PRosE dialects.

In order to generate a Translator for a specific natural language one has to define the various apsects of that language, e.g. morphology, syntax and lexical information, in the notation prescribed by the PRosE dialects. The collection of these natural language specifications, which is the Translator Generator input, is called *lingware*.

The Rosetta kernel will be described in Chapter 3, the compiler generator in chapter 4 and the PRosE dialect definitions in chapter 5. Chapter 6 contains the file names that make up the lingware of the system.

In Chapter 7 the command files are described which are used to execute the Rosetta system, the VMS compilers, the Rosetta compilers etc..

Chapters 3, 4, 5, 6 and 7 contain the description of all *genuine sources* of the Rosetta software. The subsequent chapters contain module descriptions of *targets*, that is, modules which are generated by compilers or linkers from other modules. Chapter 8 specifies the targets of the compiler generator. Chapter 9 contains a list of immediate targets of each Rosetta compiler and in Chapter 10 all Rosetta compiler executables are listed.

Finally, there is an index containing an alphabetically ordered list of all module names occurring in the document.

2 Naming conventions and terminology

In the document some naming conventions are used. All module names are labeled with one of the following strings:

(m) A normal PASCAL module consisting of an *environment* and *implementation* part which are in the same component.

(p) A PASCAL module which is the body of a program, that is, it only consists of an *implementation* part.

(oe) A PASCAL module which has no *implementation* part, only an *environment* part.

(em) The environment part of a normal PASCAL module.

(im) The implementation part of a normal PASCAL module.

(da) The file doesn't contain PASCAL code but only data in a fixed format.

(exe) The file is an executable

(com) The file is a DCL command file.

(pr) The file is a *lingware* file and contains a specification in a PRosE dialect.

(pd) The file is an inputfile for the compiler generator and contains a definition of a PRosE dialect.

(ls) Both the *environment* part and *implementation* part are *language specific* and have to be present in each language component.

(ld) The module is language dependent, that is, the module has both an *environment* and *implementation* part. The enviroment part is language independent and resides in the specified component. The implementation part, however, is language specific. Therefore, each language component (DUTCH, ENGLISH and SPANISH) will have its own version of the implementation part of the module.

(ldi) The module is language dependent, that is, the module has both an *environment* and *implementation* part. The enviroment part is language independent and resides in the specified component. The implementation part is language specific. The language specific parts for each language differ **only** with respect to the component names of the *inherited* modules. In particular, the DUTCH version of the implemention part inherits DUTCH modules, the ENGLISH implementation part inherits ENGLISH components, etc.. In stead of having three almost identical implementation part there is just one implementation part (residing in the same component as the environment part) in which the modules are inherited from the virtual component LANGUAGE.

Most of the Rosetta compilers have a similar structure. A compiler module with a specific task has been given a fixed name ending, e.g. the name of a compiler module in which a lexical scanner is implemented will end in '**scanner**'. Therefore the compiler modules which obey this naming convention are described only once (at their first cccurence).

3 Rosetta kernel

3.1 Components

The modules in this section are directly related to the definitions of the Rosetta components in the formalism. There is a 1-1 corespondence between these modules and the functions defined in the formalism. The only exception is the module **control** which has no counterpart in the formalism.

general:control (p) : Controls the execution of the analysis and generation process. It also passes information about debugging and interface printing to both processes.

general:analysis (p) :

- **general:alayout (m)** :
- **general:amorph (m)** :
  - **general:asegm (m)** :
  - **general:alex (m)** :
  - **general:alextree (m)** :
- **general:surfparser (m)** :
- **general:mparser (m)** :
- **general:atransfer (m)** :

general:generation (p) :

- **general:gtransfer (m)** :
- **general:mgenerator (m)** :
- **general:linearizer (m)** :
- **general:gmorph (m)** :
  - **general:glextree (m)** :
  - **general:glex (m)** :
  - **general:gsegm (m)** :
- **general:glayout (m)** :

3.2 Component interface modules

This section describes all modules in which the interface datastructures between **alayout**, **amorph**, **surfparser**, **mparser**, **atransfer**, **gtransfer**, **mgenerator**, **linearizer**, **gmorph** and **glayout** are defined,

general:interfaces (oe) : Defines the general interface datastructure.

general:interface0 (m) : Interface between **alayout** and **amorph**.

general:interface1 (m) : Interface between **amorph** and **surfparser**.

general:interface2 (oe) : Interface between **surfparser** and **mparser**.

general:interface3 (oe) : Interface between **mparser** and **atransfer**.

general:interface4 (oe) : Interface between **analysis** and **generation**.

general:interface5 (oe) : Interface between **gtransfer** and **mgenerator**.

general:interface6 (oe) : Interface between **mgenerator** and **linearizer**.

general:interface7 (m) : Interface between **linearizer** and **gmorph**.

general:interface8 (m) : Interface between **gmorph** and **glayout**.

3.3 Interface print modules

general:printerf (m) : Can print (on screen) the datastructure defined in **interfaces**.

general:printerf1 (m) : Prints datastructures defined in **interface1**.

general:printerf2 (m) : Prints datastructures defined in **interface2**.

general:printerf3 (m) : Prints datastructures defined in **interface3**.

general:printerf4 (m) : Prints datastructures defined in **interface4**.

general:printerf5 (m) : Prints datastructures defined in **interface5**.

general:printerf6 (m) : Prints datastructures defined in **interface6**.

general:printerf7 (m) : Prints datastructures defined in **interface7**.

3.4 Short-circuit modules

general:interface1to7 (m) : Converts output of **amorph** according to datastructure definition **interface1** to data according to **interface7** which can be used as input for **gmorph**.

general:interface3to5 (m) : Converts output of **mparser** according to datastructure definition **interface3** to data according to **interface5** which can be used as input for **mgenerator**.

3.5 O.S. interface

general:string (m) : Contains string manipulation routines (dynamic strings).

general:str (m) : Contains string manipulation routines (fixed length strings) .

general:files (m) : Provides basic operations on text files.

general:windows (m) : Provides routines for window handling on the terminal screen (window creation, deletion, text manipulation in windows).

general:pc (m) : Contains process handling routines (process creation, deletion).

general:mb (m) : Contains routines for mailbox handling (mailbox creation, deletion, connection to already created mailboxes, message handling). Mailboxes are not intended to transport large amounts of data.

vms:vmsrms (m) : A file I/O package. Provides sequential and indexed file access.

general:memory (m) : Implementation of the MARK and RELEASE function for memory management.

vms:globbuf (m) : Enables the user to define buffers which can be shared between several processes. The package is intended for the transport of bulk-data.

general:clock (m) : Functions which give consumed CPU time, date, etc..

vms:vmssmg (oe) : Low-level interface to the SMG package of VMS.

vms:vmsstr (oe) : Low-level interface to the STR package of VMS.

vms:libdef (oe) : Provides an interrupt generating function, which can be used to generate a run-time error.

3.6 Layout, phonetical and context-condition rules

general:ldmorfdef.env (ld) : Defines the interface to the layout rules, the phonetical rules and the context-condition rules.

3.7 Segmentation rule interface

general:asegmrules (m) : Takes care of the actual application of the analytical segmentation rules.

general:gsegmrules (m) : Takes care of the actual application of the generative segmentation rules.

general:segmrules (m) : Provides I/O routines for handling the segmentation rule files.

general:ldsucc.env (ld) : The interface to the control expression which defines the application order of the segmentation rules.

3.8 W-Rule interface

general:anlexif.env (ld) : Interface for the analytical lextree rules.

general:genlexif.env (ld) : Interface for the generative lextree rules.

3.9 Surface parser rule interface

general:surfrulesgraphs.env (ld) : Defines the interface to the control expression part of the surface rules.

general:surfrules.env (ld) : Defines the interface to the condition action part of the surface rules.

3.10 M-rule (interface) modules

general:ldmrules.env (ld) : Part of the M-rule interface that is the same both for analysis and generation.

general:ldsubgrammars.env (ld) : Contains functions that give information about the subgrammars which are defined in the M-rules.

general:ldanmrules.env (ld) : Analytical M-rule interface definition.

general:ldgenmrules.env (ld) : Generative M-rule interface definition.

general:limrules (m) : Converts the M-grammar control expressions into M-rule successor (predecessor) relations.

general:limatches (m) : Provides functions for M-rule model matching.

3.11 Transfer rule interface modules

general:ldanilrules.env (ld) : Analytical transfer-rule interface definition

general:ldgenilrules.env (ld) : Generative transfer-rule interface defintion

3.12 Lexicon interface modules

general:ldmdict (ldi) : Provides functions for accessing MDICT.

general:mdictdef (ldi) : Definition of the MDICT record structure.

general:lifixiddict (m) : Functions for accessing the fixed idiom dictionary FIXIDDICT.

general:ldblex (ldi) : BLEX access functions.

general:lisdict (m) : SDICT (defines the relation between fkeys and skeys) access functions.

general:lisiddict (m) : SIDDICT (the semi-idiom dictionary) access functions.

general:liiddict (m) : IDDICT (the idiom dictionary) access functions.

general:liildict (m) : ILDICT access functions. Furthermore, it includes a definition of the ILDICT record structure.

general:ldaffixlex (ldi) : Defines functions for accessing the affix lexicon.

general:ldgluelex (ldi) : Defines functions for accessing the glue lexicon.

general:strtokey (m) : A package access functions for the compiled skey, fkey and mkey definition files.

general:strkeyrecdef (oe) : Definition of the record structure of the compiled key definition files.

3.13 Tree definition modules

general:lidomaint (oe) : Defines the language independent counterparts of the types defined in module **language:lsdomaint**.

general:listree (m) : Defines the abstract datatype S-tree

general:superdtree (m) : Defines the abstract datatype Super D-tree

general:hyperdtree (m) : Defines the abstract datatype Hyper D-tree

general:hiltree (m) : Defines the abstract datatype Hyper IL-tree

3.14 Tree print modules

general:drawtree (m) : Implementation module of the Tree-print algorithm. The module is able to print various types of trees. Drawtree implements the tree type-independent part of the algorithm.

general:oldtree (m) : Module **oldtree** implements the tree-type dependent parts of the algorithm, e.g. GetSon, GetBrother, etc.

general:drawstree (m) : Prints an S-tree on the screen.

general:drawsuperdtree (m) : Prints a Super D-tree on the screen.

general:drawhyperdtree (m) : Prints a Hyper D-tree on the screen.

general:drawhiltree (m) : Prints a Hyper IL-tree on the screen.

general:lirectoscreen (m) : Prints the contents of a tree node on the screen.

general:rectoscreen (m) : Prints the contents of S-tree node record on the screen and enables the user to edit the record.

3.15 Debug support modules

general:error (m) : Prints a status line containing an error or status message on the screen.

general:log (m) : Writes a message to a log file. There is one log file for each process.

general:debug (m) : Writes debug information in a special window on the screen.

general:debugmparser (m) : Implementation of the M-parser debug mode.

general:debugmgenerator (m) : Implementation of the M-generator debug mode.

3.16 Conversion modules

general:ldconvrec.env (ld) : Contains conversion routines for string datastructure to record conversion and vice versa.

general:ldstrtotype.env (ld) : Conversion routines for string to category, relation, context condition etc.

general:ldtypetostr.env (ld) : Inverse of module **ldstrtotype**.

general:ldidpatterns.env (ld) : Conversion module from idiom pattern to string.

3.17 Miscellaneous

3.17.1 Surface parser modules

general:ldprims.env (ld) : Surface parser help routines.

general:items (oe) :

general:ldequal.env (ld) :

3.17.2 Mparser, Mgenerator modules

general:mpstatistics (m) : Puts a window on the terminal screen at the end of an M-parse with information about the number of M-rule applications etc..

general:globsubst (m) : Makes the substituent S-tree of the substitution rules globally available for other M-rules.

3.17.3 Miscellaneous

general:lsstree (ldi) : Definition of language specific S-tree.

general:globdef (m) : Module which has the function of a blackboard for the system. It is used to store and provide general information data.

general:ldgetkey.env (ld) : Returns the key value of basic S-trees. In case the S-tree is a variable it returns the index value of the variable.

general:ldcatsets.env (ld) : Provides characteristic functions of several sets which have been specified in the language domain.

general:ldsubsttovar.env (ld) : Gives for an S-tree a corresponding stree with a VAR category that corresponds to the category of the input S-tree.

general:ldstrtostr.env (ld) :

4 The compiler generator

4.1 Generical compiler modules

tools:gencomp (p) : Makes a copy of a generical module and changes the copy for usage in the compiler to be generated.

tools:gencom (p) : Generical program body of a compiler.

tools:gencomscanner (m) : Generical lexical scanner.

tools:gencomparser (m) : Generical parser module.

tools:gencomdecl (m) : Generical datastructure module.

tools:gencomgraph (m) : Generical module in which the datastructures are defined that are used in the implementation of EBNF expressions.

tools:gencomgraphdef (em) : Generical environment module. This module defines the interface of the ENBNF expression definitions.

4.2 Domain specification evaluation

tools:mrudomcom (p) :

tools:mrudomcomlangspec (m) :

tools:gencomscanner (em) :

tools:mrudomcomscanner (im) :

tools:gencomgraphdef (em) :

tools:mrudomcomgraphdef (im) : Definition of the EBNF expressions defining the global syntax of the language.

tools:gencomgraph (m) :

tools:gencomdecl (m) :

tools:gencomparser (m) :

tools:mrudomcomrules (m) : Definition of the actual attribute grammar.

4.3 Syntax and Code generation evaluation

tools:mrusurcom (p) :

tools:mrusurcomcode (m) : Definition of the code generationpart of the compiler.

tools:mrusurcomdecl (m) :

tools:mrusurcomscanner (m) :

tools:mrusurcomgraph (m) :
tools:mrusurcomgraphdef (m) :
tools:mrusurcomparser (m) :
tools:mrusurcomrules (m) :

5 PRosE dialect compilers

5.1 Domain dialects

5.1.1 Domain T dialect

tools:domcom (p) : The program body of the domain compiler

tools:domcomgraphdef (im) : Specification of the EBNF expressions defining the syntax of the Domain T language.

tools:domcomlangspec (m) : Definition of the atribute grammar *domain*.

tools:domcomrules (m) : Definition of the attribute grammar, including the code generation.

tools:domcomscanner (im) : The lexical scanner implementation part.

tools:gencomscanner (em) :

tools:gencomparser (m) :

tools:gencomdecl (m) :

tools:gencomgraph (m) :

tools:gencomgraphdef (em) :

5.1.2 Auxiliary domain language

tools:auxcom.gendom (pd) : Definition of the attribute grammar domain

tools:auxcom.gensur (pd) : Definition of the attribute grammar syntax and code generation.

5.2 Lexicon dialects

5.2.1 Rosetta lexicon language

tools:newdictgen (p) : The lexicon compiler program which defines the syntax of the Rosetta lexicon language.

5.2.2 Lexicon linker

tools:isfinit (p) : Initialization of all Rosetta lexicons.

tools:isfmerge (p) : Merges two versions of the lexicon files.

tools:fixidgen (p) : The compiler of the fixed idiom specification files which are generated by the lexicon compiler.

5.2.3 Lexicon constraint language

tools:constraintgen (p) : The compiler which defines the language in which the constraints of the Rosetta lexicon can be specified.

5.2.4 Key definition language

tools:strkey (p) : Defines the (very simple) language in which lexicon keys can be specified.

5.3 Affix rules

5.3.1 Affix rule language

tools:asegcom (p) : Body of the analytical affix rule compiler.

tools:gsegcom (p) : Body of the generative affix rule compiler.

tools:segcomdecl (m) : c.f. previously described compilers.

tools:segcomgraph (m) :

tools:segcomgraphdef (m) :

tools:segcomlangspec (m) :

tools:segcomparser (m) :

tools:segcomrules (m) :

tools:segcomscanner (m) :

5.3.2 Affix control expression language

tools:afxpr.gendom (pd) : Domain of the affix control expression attribute grammar.

tools:afxpr.gensur (pd) : Syntax and code generation part of the affix control expression grammar.

5.4 W-Rule language

tools:lexcom (p) : Body of the W-Rule compiler.

tools:lexcomgraphdef (m) : c.f. previously described compilers.

tools:lexcomgraph (m) :

tools:lexcomdecl (m) :

tools:lexcomparser (m) :

tools:lexcomscanner (m) :

tools:lexcomrules (m) :

tools:lexcomcode (m) :

5.4.1 W-Rule linker

tools:lexlink (p) : Generates a kind of switch over all W-Rules.

5.5 Surface Rule language

tools:surcom (p) : Body of the Surface Rule compiler.

tools:surcomcode (m) : c.f. previously described compilers.

tools:surcomdecl (m) :

tools:surcomscanner (m) :

tools:surcomparser (m) :

tools:surcomgraph (m) :

tools:surcomgraphdef (m) :

tools:surcomrules (m) :

5.6 M-Rule language

tools:mrucom.gendom (pd) : Domain of the M-rule attribute grammar.

tools:mrucom.gensur (pd) : Syntax and code generation part of the M-rule attribute grammar.

5.6.1 M-Rule linker

tools:mrulelink (p) : Collects global M-rule information, e.g all M-rule names, and generates functions which enable other modules to use this information.

5.7 Transfer Rule language

tools:tracom.gendom (pd) : Domain of the transfer rule attribute grammar.

tools:tracom.gensur (pd) : Syntax and code generation part of the transfer rule attribute grammar.

5.8 Interlingua Rule language

tools:ilacom.gendom (pd) : Domain of the interlingua attribute grammar.

tools:ilacom.gensur (pd) : Syntax and code generation part of the interlingua attribute grammar.

6 Lingware modules

6.1 Domains

language:lsdomaint.dom (pr) : Domain T specification.

language:lsauxdomain.auxdom (pr) : Auxiliary domain specification.

6.2 Lexicons

Specification of the Rosetta dictionaries.

language:auxverb.dict (pr) : Specification of AUXVERB entries.

language:badj.dict (pr) : Specification of BADJ entries

language:badv.dict (pr) : Specification of BADV entries

language:bnoun.dict (pr) : Specification of BNOUN entries

language:bverb.dict (pr) : Specification of BVERB entries

language:conj.dict (pr) : Specification of CONJ entries

language:exclam.dict (pr) : Specification of EXCLAM entries

language:misc.dict (pr) : Specification of entries for several categories.

language:particle.dict (pr) : Specification of PARTICLE entries

language:prep.dict (pr) : Specification of PREP entries

language:pronoun.dict (pr) : Specification of PRONOUN entries

language:skeydef.kdf (pr) : Definition of the 'skeys'.

language:constraints.constr (pr) : Constraint specifications for BLEX.

6.3 Morphology

6.3.1 String Phase

language:lglue.seg (pr) : Specification of left GLUE rules.

language:rglue.seg (pr) : Specification of right GLUE rules.

language:mglue.seg (pr) : Specification of middle GLUE rules.

language:suffix.seg (pr) : Specification of suffix rules.

language:prefix.seg (pr) : Specification of prefix rules

language:morphexpr.afxpr (pr) : Specification of the order in which prefix and suffix rules must be applied.

6.3.2 S-tree Phase

There are three files available in which the, so-called, W-rules can be specified:

language:lexrules1.lex (pr) :
language:lexrules2.lex (pr) :
language:lexrules3.lex (pr) :

6.4 Surface Grammar

There are three files in which the Surface rules can be specified:

language:surfrules1.sur (pr) :
language:surfrules2.sur (pr) :
language:surfrules3.sur (pr) :

6.5 M-Rules

The filenames of the files in which the M-rules have been specified have not been prescribed. The extension, however, must be '.mrule'. In order for the system to know which filenames have been used, they must be specified in the file 'mrules.mms'

language:*.mrule (pr) : M-Rule files.
language:mrules.mms (pr) : Specification of M-Rule filenames.

6.6 Transfer Rules

language:transferrules.trans (pr) : Specification of transfer rules which establish the link between M-Rules and IL-Rules.

6.7 PASCAL Lingware

A small part of the Lingware has still to be specified in PASCAL.

language:lsmruquo (ls) : PASCAL functions which are used in the M-rules
language:lsphondef (ls) : Specification of the phonetical information and the phonetical rules which are used in the morphology component.
language:ldidpatterns.pas (ld) : Conversion module from PASCAL idiom pattern value to string value.
language:ldmorfdef.pas (ld) : Specification of layout rules and context conditions.

6.8 Interlingua

interlingua:mkeydef.kdf (pr) : Definition of the names of the basic meanings.
interlingua:ildefinition.ilan (pr) : Definition of the IL rulenames

7 Command Files

7.1 The Rosetta system

general:runrosetta (com) : Starts the Rosetta system in interactive mode.

general:batchrosetta (com) : Asks the user about the system configuration he wants to run in batch-mode and starts the batch-mode job.

general:batchros (com) : Starts the actual batch mode job.

actions:tstb (com) : Starts a batch-mode job to run a test-bench.

actions:morftest (com) : Starts a version of the Rosetta system in which the morfological components have been shortcircuited.

7.2 The compiler generator

actions:gen (com) : Runs the compiler generator and creates a compiler.

actions:gencomp (com) : Command file used during generation of a compiler for preparing a generical module in order to be used in the compiler.

actions:gendom (com) : Command file used during compiler generation for domain evaluation.

actions:gensur (com) : Used for evaluation of the syntax specification and code generation.

actions:genobj (com) :

7.3 The compilers

7.3.1 VMS compilers

actions:pas (com) : Applies the PASCAL compiler to the implementation part of a module.

actions:env (com) : Applies the PASCAL compiler to the environment part of a module.

actions:obj (com) : Creates an .opt file.

actions:opt (com) : Links a number of object into an executable.

actions:merge_opt (com) : Adds a name of an object file to a .opt file.

actions:vmssysenv (com) : Copies the compiled system library routines from the system library.

7.3.2 Rosetta compilers

actions:dom (com) : Invokes the domain T compiler.

actions:auxdom (com) : Invokes the auxiliary domain compiler.

actions:dict (com) : Invokes the lexicon compiler.

actions:fixid (com) : Invokes the fixed idiom compiler.

actions:constraint (com) : Invokes the lexicon constraint compiler.

actions:kdf (com) : Invokes the key definition compiler.

actions:seg (com) : Invokes the segmentation rule compiler.

actions:afxpr (com) : Invokes the affix control expression compiler.

actions:lex (com) : Invokes the W-Rule compiler.

actions:sur (com) : Invokes the Surface Rule compiler.

actions:mru (com) : Invokes the M-rule compiler.

actions:mruall (com) : Invokes the M-rule compiler and compiles all immediate targets.

actions:idioms (com) : Invokes the M-rule compiler for the idiom mrule file(s).

actions:tra (com) : Invokes the transfer rule compiler.

actions:ila (com) : Invokes the interlingua compiler.

7.4 The Linkers

actions:isfinit (com) : Initializes the Rosetta lexicon.

actions:isfexit (com) : Completes the creation of the Rosetta lexicon.

actions:isfmerge (com) : Adds a sub-lexicon to the Rosetta lexicon.

actions:llk (com) : Calls the W-rule linker.

actions:mlk (com) : Calls the M-rule linker.

8 Compiler generator targets

tools/language:<*compilername*>**.exe (exe)** :

9 PRosE compiler targets

9.1 Domain dialects

9.1.1 Domain T targets

language:lsdomaint (oe) :
language:lstypetostr (ls) :
language:lsstrtotype (ls) :
language:lsconvrec (ls) :
language:lsconvattr (ls) :
language:maket (ls) :
language:copyt (ls) :
language:ldstrtotype.pas (ld) :
language:ldtypetostr.pas (ld) :
language:ldstrtostr.pas (ld) :
language:ldgetkey.pas (ld) :
language:ldconvrec.pas (ld) :
language:ldcatsets.pas (ld) :
language:ldequal.pas (ld) :

9.1.2 Auxiliary domain targets

language:lsauxdom (ls) :
language:ldsubsttovar.pas (ld) :

9.2 Lexicon dialects

9.2.1 Rosetta lexicon targets

For each of the lexicon sources (cf.) the following targets are generated:

language:blex<*source*>**.isf (da)** :
language:iddict<*source*>**.isf (da)** :
language:ildict<*source*>**.isf (da)** :
language:mdict<*source*>**.isf (da)** :

language:sdict<*source*>**.isf (da)** :
language:siddict<*source*>**.isf (da)** :
language:fixid<*source*>**.fixid (da)** :

9.2.2 Lexicon linker targets

language:blex.isf (da) :
language:iddict.isf (da) :
language:ildict.isf (da) :
language:mdict.isf (da) :
language:sdict.isf (da) :
language:siddict.isf (da) :
language:fixid.fixid (da) :
language:fixid.isf (da) :

9.2.3 Lexicon constraint targets

language:lsconstraints (ls) :

9.2.4 Key definition targets

language:skeydef.rmskdf (da) :
language:mkeydef.rmskdf (da) :

9.3 Affix rules

9.3.1 Affix rule targets

For each of the affix rule sources (lglue, rglue, mglue, prefix, suffix) the following targets are generated:

language:a<*source*>**.sro (da)** :
language:a<*source*>**.sso (da)** :
language:a<*source*>**.svo (da)** :
language:a<*source*>**.sco (da)** :
language:a<*source*>**.svs (da)** :
language:g<*source*>**.sro (da)** :

language:g<*source*>.sso (da) :
language:g<*source*>.svo (da) :
language:g<*source*>.sco (da) :
language:g<*source*>.svs (da) :

9.3.2 Affix control expression targets

language:ldsucc.pas (ld) :

9.4 W-Rule targets

For each of the three W-rule files the following targets are generated:

language:comlexrules<*i*> (ls) :
language:decomlexrules<*i*> (ls) :

9.4.1 W-Rule linker targets

language:anlexif.pas (ld) :
language:genlexif.pas (ld) :

9.5 Surface Rule targets

language:surfrules.pas (ld) :
language:surfrulesgraphs.pas (ld) :
language:ldprims.pas (ld) :

9.6 M-Rule targets

language:commrules<*i*> (ls) :
language:decommrules<*i*> (ls) :

9.6.1 M-Rule linker targets

language:ldmrules.pas (ld) :
language:ldsubgrammars.pas (ld) :
language:ldanmrules.pas (ld) :

language:ldgenmrules.pas (ld) :
language:helpsubgrammars (ls) :
language:lsparams (oe) :

9.7 Transfer Rule targets

language:ldanilrules.pas (ld) :
language:ldgenilrules.pas (ld) :

9.8 Interlingua rule targets

interlingua:liilrules (m) :

10 Executables

10.1 The Rosetta system

general:control (exe) : The control process.

language:analysis (exe) : The analysis process.

language:generation (exe) : The generation process.

10.2 the compiler generator generation

tools:gencomp (exe) : Generical module handling.

10.3 The compiler generator

tools:mrudomcom (exe) : Evaluates compiler domain definition.

tools:mrusurcom (exe) : Evaluates compiler syntax and code generation definition.

10.4 The compilers

tools:domcom (exe) : The domain compiler.

language:auxcom (exe) : The auxiliary domain compiler.

dutch:newdictgen (exe) : The lexicon compiler.

tools:fixidgen (exe) : The fixid idiom specification compiler.

tools:constraintgen (exe) : The lexicon constraint compiler.

tools:strkey (exe) : The key definition compiler.

language:asegcom (exe) : The analytical affix-rule compiler.

language:gsegcom (exe) : The generative affix-rule compiler.

tools:afxpr (exe) : The affix control expression compiler.

language:lexcom (exe) : The W-Rule compiler.

language:renesurcom (exe) : The Surface Rule compiler.

language:mrucom (exe) : The M-Rule compiler.

language:tracom (exe) : The Transfe Rule compiler.

language:ilacom (exe) : The Interlingua compiler.

10.5 The linkers

language:isfinit (exe) : The lexicon initializer.

language:isfmerge (exe) : The lexicon merger.

language:lexlink (exe) : The W-Rule linker.

language:mrulelink (exe) : The M-Rule linker.

Index