TOPO: Quick Reference
Auxiliary Tools — version 3R6

José A. Maña
Tomás de Miguel
Tomás Robles
Joaquín Salvachua
Gabriel Huecas
Marcelino Veiga

Dpt. Ingeniería Telemática
E.T.S.I. de Telecomunicación
Univ. Politécnica de Madrid
E-28040 Madrid, SPAIN

<topo@dit.upm.es>

17 October, 1994

Abstract

TOPO is a set of interoperating LOTOS tools grouped in several packages. This one contains some auxiliary tools: a cross reference generator, an ADT cross reference generator, a pretty printer, a symbolic data type interpreter, and a test case generator for ADTs.

This paper is a quick reference guide for users.

1 Getting Started . . .

TOPO is a set of interoperating LOTOS tools grouped in several packages. This one includes some auxiliary tools

- `lxref`: a cross reference generator
- `adtxr`: an ADT cross reference generator
- `pl`: a pretty printer
- `ildi`: an interactive symbolic data type interpreter
- `rdtts`: A language to specify ADT test suites, and execute them

But for working properly, this package must be together with TOPO Front-End package.

2 Calling Tools

For calling the auxiliary tools of TOPO, it is used the same interface as TOPO Front-End package. Only some options are added.

`topo` spec[`.lot`]. . .	
-r . . .	Cross references. For every identifier in the spec, it generates a list of lines where it is referenced. Further options refine its behaviour: see below.
-a . . .	ADT cross references. By default it considers every ADT in the spec and prints all its evolution, that is the traces down to basic ADTs. The result is a tree of dependencies: union, renaming, actualization. Further options refine its behaviour: see below.
-pp . . .	Pretty printer. Produces in stdout a nice copy of the original specification. Further options refine its behaviour: see below.
-ildi	Interactive Symbolic ADT Interpreter. Allows to excersise the rewrite system interactively (or even batch). Useful to debug ADT specifications.
-rdtts . . .	Real Data Types Test Specification. Permits to specify in a compact form a test suite to excercise the ADT specification. It derives a test system that may be compiled and executed. Or a specification that may be into LOLA, or even into `ildi`. It pays attention to annotations, so it proves that the annotated specification works.

Options may be specified on line,

tool	help	example
adtxr	topo -ah	topo spec -a -iBoolean
lxref	topo -rh	topo spec -r -p
pl	topo -pph	topo spec -pp -x70
rdtts	topo null -rdtts -help	topo spec -rdtts -ildi

2.1 Cross Reference

`topo` spec`[.lot]` `-r ...`	
-h	On-line help. Sharp.
-p	Only process identifiers.
-g	Only gate identifiers.
-t	Only type identifiers.
-s	Only sort identifiers.
-S	Only formal sort identifiers.
-o	Only operation identifiers.
-O	Only formal operation identifiers.
-v	Only variable identifiers.
-P	Only the specification identifier.
-N	Only names (sort-names and operation-names).
-b	Brief listing: only line numbers.
-c	Listing is classified according to identifier class.
-u	Lists those identifiers that are never used.
-l	Work on syntax analysis (no semantics).

2.2 ADT Cross Reference

`topo` spec[`.lot`] `-a` ...	
-h	On-line help. Sharp.
-b	Only basic types are considered. Basic types are those self–contained.
-c	Only types that are build as union of other types.
-u	Lists those types that are not used by others.
-U	Lists those types that are further used by others: used in unions, renamings, actualizations.
-r	Lists those types that are derived as a renaming of another.
-a	Lists those types that are derived as an actualization of another.
-nT	Only reports on type T
-iT	Only reports on types that use T

2.3 Pretty Printer

`topo` spec[`.lot`] `-pp` ...	
-h	On-line help. Sharp.
-i #	indentation increment (default: -i2)
-s #	horizontal spacing between boxes (default: -s1)
-x #	max characters per line (default: -x80)

2.4 Symbolic Data Interpreter

Commands are usually provided interactively, or may be read from a batch file. Interactive access provides an emacs-like editing of the command line. It provides the following commands:

```
LOAD file
EXEC file
RENAME [ identifier IS identifier ]
REWRITE expression
CHECK expression '=' expression
VAR variable_declaration
LET identifier '=' expression
FREE identifier_free_list [ ':' sort_identifier ]
RESET identifier_reset_list [ ':' sort_identifier ]
IF '(' expression '=' expression ')' cmdl [ '::' cmdl ] ';'
DISPLAY [ item ]
SETDEB [deb]
TRACE [ON/OFF]
STAT stat
WRITE string
RESET
QUIT
HELP
```

2.5 Real Data Types Test Specification

`topo` spec[`.lot`] `-rdtts` ...	
-h	On-line help. Sharp.
-topo	generates code for topo compiler (default)
-lola	generates code for lola (or smile)
-ildi	generates code for ildi