

Reference Manual for CRESS

(Communication Representation Employing Systematic Specification)



© Kenneth J. Turner
Computing Science and Mathematics, University of Stirling,
Stirling FK9 4LA, Scotland
(www.cs.stir.ac.uk/~kjt)

Version 4.6 (27th October 2010)

1.	Introduction	5
CRESS Tools.....		6
2.	Overview.....	6
3.	Tools.....	6
4.	Diagrams	8
5.	Installation.....	9
6.	Directory Structure	10
7.	Cress Tools	11
7.1	Tool Options	11
7.2	cadp_annotate	12
7.3	chive_font	14
7.4	cress_bpel.....	14
7.5	cress_check	15
7.6	cress_cpl.....	16
7.7	cress_create.....	16
7.8	cress_deploy.....	18
7.9	cress_expand.....	19
7.10	cress_lotos.....	20
7.11	cress_realise	21
7.12	cress_sdl.....	21
7.13	cress_test.....	22
7.14	cress_validate.....	22
7.15	cress_verify	22
7.16	cress_vxml	23
8.	Diagram Editors	23
CRESS Syntax		25
9.	Diagrams	25
9.1	Diagram Structure.....	25
9.2	Rule Boxes	26
9.3	Node Labels	26
9.4	Diagram Parsing	27

9.5	Diagram Checking	28
10.	Expressions.....	28
10.1	General Rules.....	28
10.2	Translation to BPEL	28
10.3	Translation to CPL.....	29
10.4	Translation to Lotos	29
10.5	Translation to SDL.....	30
10.6	Translation to VoiceXML.....	31
CRESS Application Domains.....		33
11.	Overview.....	33
12.	DS (Device Services).....	33
12.1	Configuration Diagram.....	34
12.2	Services/Features	34
12.3	Signals.....	34
12.4	Actions	35
12.5	Events	35
12.6	Dynamic (Run-Time) Variables	35
12.7	Variable Types.....	35
12.8	Specification Validation	35
13.	GS (Grid Services).....	35
13.1	Configuration Diagram.....	35
13.2	Services/Features	36
13.3	Signals.....	36
13.4	Actions	36
13.5	Events	36
13.6	Dynamic (Run-Time) Variables	36
13.7	Variable Types.....	36
13.8	Specification Validation	36
13.9	Specification Verification	37
13.10	Implementation Validation	38
14.	IN (Intelligent Network).....	39
14.1	Configuration Diagram.....	39
14.2	Services/Features	40
14.3	Signals.....	40
14.4	Static (Profile) Variables.....	42
14.5	Dynamic (Run-Time) Variables	42
14.6	Announcement Messages.....	43
14.7	Variable Types.....	43
14.8	Specification Validation	43
15.	IVR (Interactive Voice Response).....	45
15.1	Configuration Diagram.....	45
15.2	Services/Features	45
15.3	Signals.....	46
15.4	Actions	46
15.5	Events	47
15.6	Variable Interpolation	47
15.7	Platform Variables	47
15.8	Variable Types.....	48
15.9	Specification Validation	48
15.10	Specification Verification	50
16.	SIP (Session Initiation Protocol)	51
16.1	Configuration Diagram.....	51
16.2	Services/Features	51

16.3	Signals.....	52
16.4	Static (Profile) Variables.....	53
16.5	Dynamic (Run-Time) Variables	53
16.6	Announcement Messages.....	53
16.7	Variable Types.....	54
16.8	Specification Validation	54
17.	VoIP (Voice over Internet Protocol)	55
17.1	Configuration Diagram.....	55
17.2	Services/Features	55
17.3	Actions.....	56
17.4	Guards.....	57
18.	WS (Web Services)	57
18.1	Configuration Diagram.....	57
18.2	Services/Features	57
18.3	Signals.....	57
18.4	Actions.....	58
18.5	Events	58
18.6	Dynamic (Run-Time) Variables	58
18.7	Variable Types.....	59
18.8	Specification Validation	60
18.9	Specification Verification.....	60
18.10	Implementation Validation	61
CRESS Target Languages.....		63
19.	Overview.....	63
20.	Working with BPEL.....	63
21.	Working with CPL	65
22.	Working with Lotos.....	65
23.	Working with SDL	67
24.	Working with VoiceXML	67
CRESS File Manifest		68
25.	General Files	68
25.1	Naming Conventions	68
25.2	Binary Files (directory <i>bin</i>).....	68
25.3	Documentation Files (directory <i>doc</i>)	69
25.4	Support Files (directory <i>supp</i>)	69
25.5	Test Files (directory <i>test</i>).....	70
26.	Application Domain Files.....	70
26.1	Chisel Files (directory <i>chisel</i>).....	70
26.2	Device Service Files (directory <i>ds</i>).....	71
26.3	Grid Service Files (directory <i>gs</i>).....	71
26.4	Intelligent Network Files (directory <i>in</i>)	71
26.5	Interactive Voice Response Files (directory <i>ivr</i>)	72
26.6	Session Initiation Protocol Files (directory <i>sip</i>).....	72
26.7	Voice over Internet Protocol Files (directory <i>voip</i>)	73
26.8	WS Files (directory <i>ws</i>)	73
27.	Target Language Files.....	73
27.1	BPEL/WSDL Files (directory <i>bpel</i>).....	73
27.2	CPL Files (directory <i>cpl</i>)	74
27.3	Lotos Files (directory <i>lotos</i>).....	74
27.4	SDL Files (directory <i>sdl</i>)	74
27.5	VoiceXML Files (directory <i>vxml</i>).....	76
28.	CRESS Licence.....	76

29.	History	76
29.1	Versions 1.0 – 2.0	76
29.2	Version 2.1	76
29.3	Version 2.2	76
29.4	Version 2.3	76
29.5	Version 2.4	76
29.6	Version 2.5	77
29.7	Version 2.6	77
29.8	Version 2.7	77
29.9	Version 2.8	77
29.10	Version 2.9	77
29.11	Version 3.0	77
29.12	Version 3.1	77
29.13	Version 3.2	77
29.14	Version 3.3	78
29.15	Version 3.4	78
29.16	Version 3.5	78
29.17	Version 3.6	78
29.18	Version 4.0	78
29.19	Version 4.1	79
29.20	Version 4.2	79
29.21	Version 4.3	80
29.22	Version 4.4	81
29.23	Version 4.5	82
29.24	Version 4.6	83

1. Introduction

See the [Cress home page](#) for an overview of Cress and some references. Cress (Communication Representation Employing Systematic Specification) supports:

- diagrams for the following kinds of services, though the approach is extensible for other domains:
 - DS (Device Services, i.e. BPEL/WSDL used with OSGi)
 - GS (Grid Services, i.e. BPEL/WSDL)
 - IN (Intelligent Network)
 - IVR (Interactive Voice Response, i.e. VoiceXML)
 - SIP (Session Initiation Protocol, i.e. Internet Telephony)
 - VoIP (Voice over Internet Protocol, i.e. Internet Telephony with SIP/CPL)
 - WS (Web Services, i.e. BPEL/WSDL)
- parsing of service diagrams prepared with one of the following editors, though the approach is extensible for other graphical editors:
 - *Chive*
 - *Diagram!*
 - *yEd*
- checking of service diagrams for syntactic and static semantic correctness
- translation of service diagrams into the following target languages, though the approach is extensible for other languages:
 - BPEL/WSDL
 - CPL
 - Lotos
 - SDL
 - VoiceXML
- miscellaneous utilities

From the user point of view, Cress simplifies service creation to the point of drawing diagrams, clicking buttons and issuing simple commands.

From the developer point of view, Cress is a very complex toolset. It is roughly equivalent to five compilers rolled into one, as it supports three diagram editors, five target languages, and seven application domains. A relevant quote for Cress might be:

“Υδραν τέμνεις
(‘You are cutting into a Hydra’, Plato, Republic 426)

CRESS Tools

2. Overview

See the [Cress home page](#) for an overview of Cress and some references. Cress (Communication Representation Employing Systematic Specification) supports:

- parsing of service diagrams prepared with various graph editors
- checking of service diagrams for syntactic and static semantic correctness
- translation of service diagrams into various target languages
- miscellaneous utilities

The current relationship between application domains and target languages is as follows:

Domain	BPEL	CPL	Lotos	SDL	VXML
DS	✓		✓		
GS	✓		✓		
IN			✓	✓	
IVR			✓	✓	✓
SIP			✓	✓	
VoIP		✓			
WS	✓		✓		

3. Tools

The following main tools are provided in the Cress toolset:

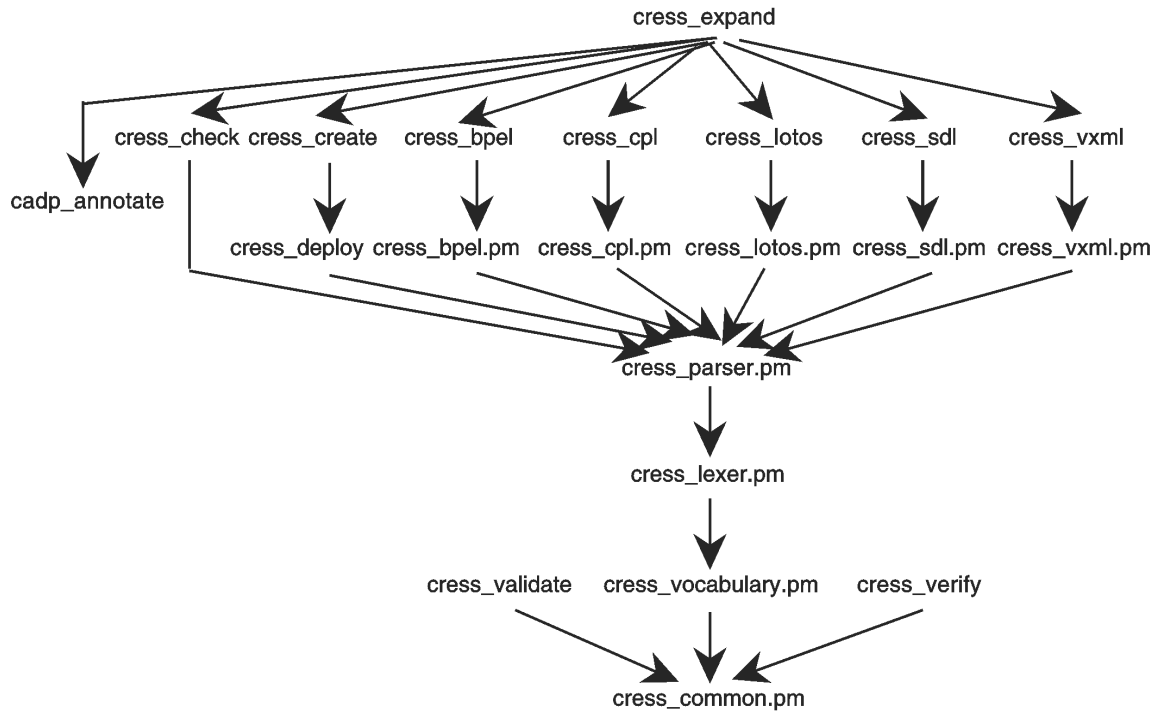
Tool	Purpose
<i>cress_bpel</i>	translate Cress diagrams to BPEL
<i>cress_check</i>	check Cress diagrams
<i>cress_cpl</i>	translate Cress diagrams to CPL
<i>cress_create</i>	create BPEL service archives
<i>cress_deploy</i>	deploy BPEL service archives
<i>cress_expand</i>	expand macros in Cress diagrams
<i>cress_lola</i>	clean up Lola (Lotos) simulation traces
<i>cress_lotos</i>	translate Cress diagrams to Lotos

<i>cress_sdl</i>	translate Cress diagrams to SDL
<i>cress_sdt</i>	turn a Lola (Lotos) test process into an MSC/PR file for Tau SDT (SDL)
<i>cress_test</i>	run JUnit tests on BPEL services
<i>cress_tidy</i>	delete temporary Lola (Lotos) and Tau SDT (SDL) files
<i>cress_validate</i>	validate Cress diagrams
<i>cress_verify</i>	verify Cress diagrams
<i>cress_vxml</i>	translate Cress diagrams to VoiceXML
<i>sdt_in</i> <i>sdt_sip</i> <i>sdt_vxml</i>	these are Tau Analyzer filters for IN, SIP and VoiceXML; set them as a filter in the Tau SDT Analyzer dialogue

These in turn rely on various Perl modules:

Module	Purpose
<i>cress_bpel.pm</i>	Cress diagram to BPEL/WSDL translator; variants apply for each vocabulary
<i>cress_common.pm</i>	Cress common definitions
<i>cress_cpl.pm</i>	Cress diagram to CPL translator
<i>cress_lexer.pm</i>	Cress lexical analyser (diagram analyser)
<i>cress_lotos.pm</i>	Cress diagram to LOTOS translator; variant code applies for each vocabulary
<i>cress_parser.pm</i>	Cress diagram parser (syntax analyser)
<i>cress_sdl.pm</i>	Cress diagram to SDL translator; variant code applies for each vocabulary
<i>cress_vxml.pm</i>	Cress diagram to VoiceXML translator
<i>cress_vocab.pm</i>	Cress vocabularies

The relationship among the main scripts is as follows:



4. Diagrams

Dependent diagrams are included automatically if not already given. Repeated diagram names are ignored. Diagram names must consist of letters or underscores, finishing with a letter. Either case of letter may be used. The diagram names *A*, *FEATURES*, *GATES*, *HOME* (for *DS*), *PROFILES* and *TYPES* are reserved.

The name of a diagram may be followed by priority, e.g. '*ACCOUNT:50*'. This is used to override the predefined priority of a known service/feature, or to define the priority of an unknown one.

Each diagram directory contains various files resulting from translation into the target languages. In addition, there is typically a **.mustard* file that contains scenario-based tests of the diagram as a Mustard file. This can be used to validate the diagram specification using either Lotos (Lola) or SDL (Tau). There may also be a **.clove* file that contains properties to be verified as a Clove file. This can be used to verify the diagram specification using CADP.

The special diagrams *FEATURES*, *GATES*, *PROFILES*, *TYPES* may be given on their own as parameters to trigger the translation of features or profiles (from the configuration diagram), Lotos gates, or types (from the *TYPES* diagram).

Files for each domain normally live in a directory whose name corresponds to the domain vocabulary (e.g. *sip*). If files are not processed in this directory, it is necessary to define the vocabulary to the tools (e.g. *-v sip*). Each domain directory has its own configuration diagram whose suffix is the vocabulary name (e.g. *CONFIG_SIP*). In general, a configuration diagram begins with a line of the form:

```
Deploys tool_options / diagram ...
```

for example:

```
Deploys -b 2 -c -n 1 -r / FALL FALL_MOVEMENT
```

The tool options are optional, and are those for the domain compiler.

A blank line is required after the **Deploys** line. The configuration options that follow are domain-dependent; see the relevant domain section later for more details.

5. Installation

To run these tools requires a Unix-like environment and Perl 5. The tools have been run on Unix (Fedora Core 4, NextStep 3.3/OpenStep 4.2, Solaris 7/8) and Windows (XP, under [CygWin](#) 1.5). The following tools may be required, depending on which domains and languages you intend to use:

- to use BPEL needs [ActiveBPEL](#) 5.0 or later
- to use device services requires [Knopflerfish](#) 2.3 or later
- to use grid services requires [Globus WS Core](#) 4.2 or later
- to test device, grid or web services requires [JUnit](#) 4.0 or later or [Mint](#), as well as [MySQL](#) 5.0 or later
- to use CPL needs a SIP server that supports it (e.g. [SER](#), [Vocal](#))
- to use Lotos needs Lola/Topo version 3.6 or later and CADP version 2009-b or later
- to use SDL needs [Tau](#) 4.6 or later
- to edit diagrams needs Diagram! from Lighthouse Design (available for a variety of NextStep/OpenStep systems), [yEd](#) from yWorks, or the Cress-specific editor [Chive](#).

A Unix-like installation is assumed in the following, though it should be possible to install and run on other platforms where Perl runs (e.g. Windows). In the following, it is assumed that the files are extracted to \$HOME/bin/cress.

For Lotos, the Stirling library files in the *supp/topo* directory needs to be made available to Topo. Do this by copying *supp/topo/stir.** to the Cress *lotos* directory or to the *topo/stdlib* installation directory.

For grid services, modified address handling and RPC handling needs to be made available to Globus WS Core. Do this by copying *supp/gt4/cress.jar* to *globus/common/lib*. In addition, copy *server-config.wsdd* to *globus/etc/globus_wsrf_core*, replacing the current version of this file.

In a few places in the code, a Unix-like environment is assumed. Macintosh and Windows end-of-line may not be correctly handled. Paths and filenames are assumed to have '/' separators. Search paths are assumed to have ':' separators. On a Windows system, it is suggested that the CygWin version of Perl be used (set to use Unix end-of-line); ActiveState Perl *may* be suitable but has not been tried.

The following environment variables must be set up. For *bash* and the like, typically place *set/export* commands in *~/.bashrc*. For *csh* and the like, typically place *setenv* commands in *~/.cshrc*. For Windows XP, set environment variables using *Control Panel/System/Advanced*.

Variable	Purpose
<i>ANT_HOME</i> (used by <i>cress_create</i> and <i>cress_deploy</i>)	location of the Ant installation, e.g. <i>C:/usr/local/ant</i> (set a dummy value if you are not using grid services)
<i>BPEL_LOG</i> (used by <i>cress_deploy</i>)	Root directory in which ActiveBPEL logs are stored, e.g. <i>C:/Docume~1/user/AeBpelEngine</i> (optional)
<i>CADP_CC</i> (used by CADP)	specification of the C compiler and the location of the Cress libraries for CADP, e.g. <i>gcc \$HOME/bin/cress/supp</i>
<i>CATALINA_HOME</i> (used by <i>cress_create</i> and <i>cress_deploy</i>)	location of the Tomcat installation, e.g. <i>C:/Program Files/Tomcat</i> (set a dummy value if you are not using web or grid services)
<i>CRESSPATH</i> (used by <i>cress_common.pm</i> and hence most Cress tools)	a colon-separated, Unix-like directory path used to locate Cress diagrams, e.g. <i>.: \$HOME/bin/cress/supp: \$HOME/bin/cress/in</i> ; note that the <i>supp</i> directory must always be included in the path if you are using Windows, you will not be able to cite drive letters unless you use CygWin references such as <i>/C/home/me/bin/cress/in</i>
<i>CRESSTEMP</i> (used by	temporary directory (default /tmp)

<i>cress_verify</i>)	
<i>GLOBUS_LOCATION</i> (used by <i>cress_create</i> and <i>cress_deploy</i>)	location of the Globus installation, e.g. <i>C:/usr/globus</i> (set a dummy value if you are not using grid services)
<i>JAVA_HOME</i> (used by <i>cress_create</i>)	location of the Sun JDK installation, e.g. <i>/usr/local/jdk</i> (optional)
<i>PATH</i> (used by command-line)	a directory path used by a shell to locate executables, e.g. include <i>\$HOME/bin/cress/bin</i>
<i>PERLLIB</i> (used by most tools)	a directory path used by Perl to locate modules, e.g. include <i>\$HOME/bin/cress/bin</i>

6. Directory Structure

Cress uses the following directory structure:

Directory	Purpose
<i>.archive</i>	This top-level directory (normally hidden in Unix) contains archived translations for each service/feature.
<i>bin</i>	This contains Perl and shell executables.
<i>ds</i> <i>gs</i> <i>in</i> <i>ivr</i> <i>sip</i> <i>voip</i> <i>ws</i> ...	Each application domain has its own top-level directory. These contain language-independent descriptions of services/features in the domain (e.g. <i>in/rc</i> describes the IN Return Call feature, <i>ws/lender</i> describes a Lender web service). Each service/feature has a subdirectory with its name. In turn, these subdirectories contain files with the same base name (e.g. <i>ws/lender/lender.chx</i> contains a Chive diagram, <i>ws/lender/lender.pdf</i> a PDF version of this, <i>ws/lender/lender.mustard</i> a Mustard scenario file, <i>ws/lender/lender.clove</i> a Clove verification file). Files generated for a service/feature may also be accumulated in this subdirectory. In addition to service/feature directories, there must be a configuration directory named after the application domain (e.g. <i>ws/config_ws</i>). This contains a configuration diagram that must be edited to change the selection or parameters of services/features.
<i>bpel</i> <i>cpl</i> <i>lotos</i> <i>sdl</i> <i>vxml</i> ...	Each target language has its own top-level directory. Each typically contains the fixed specification framework for the language and each application domain (e.g. <i>lotos/ivr.base</i> is for Lotos specification of IVR, <i>sdl/SIP*</i> is for SDL specification of SIP). These files should not be changed – unless you really know what you are doing. The <i>.archive</i> subdirectory (normally hidden in Unix) contains archived translations for major service/feature combinations (e.g. <i>POTS</i> plus all IN services, <i>BOOKING</i> plus all relevant features for IVR services). The specification framework is automatically combined with translations of the service/feature diagrams listed in the configuration diagram. This results in complete specifications for the language and the application domain (e.g. <i>lotos/ivr.lot</i> specifies IVR in Lotos including whatever features were configured, <i>sdl/SIPSystem.pr</i> specifies SIP in SDL including whatever features were configured).
<i>supp</i>	This contains supporting files: a modified WS-Addressing schema, a modified WS-Resource interface definition, the <i>stir</i> library for Topo, and the <i>types</i> directory used to accumulate the results of data type generation.
<i>test</i>	This contains test files for developer use.

7. Cress Tools

The main tools are used as follows. The translators accept all options, but ignore those that are not relevant. See the code for some examples and for how to use the minor utilities.

Most tools can be invoked without parameters to get a brief summary of how to call them. Most tools have a *customise* subroutine where local variations can be set.

7.1 Tool Options

The options used by the main tools are summarised in the following table.

Option	Purpose
<i>-a limits</i>	annotate for CADP (<i>cadp_annotate</i> , <i>cress_lotos</i>)
<i>-b</i>	blank removal (<i>chive_font</i>)
	break specification down compositionally (<i>cress_verify</i>)
<i>-b memory</i>	bit state hash memory size in MB (<i>cress_validate</i>)
<i>-b version</i>	BPEL version (<i>cress_bpel</i>)
<i>-c</i>	comment generation (most)
	compile tests (<i>cress_test</i>)
<i>-d</i>	deploy (<i>cress_deploy</i> , <i>cress_expand</i>)
	suppress deadlock freedom check (<i>cress_verify</i>)
<i>-d N</i>	maximum depth of exploration (<i>cress_validate</i>)
<i>-e level</i>	error report level (most)
<i>-f partners</i>	foreign partner names (<i>cress_create</i> , <i>cress_lotos</i>)
<i>-f family</i>	font family (<i>chive_font</i>)
<i>-h</i>	help (all)
<i>-i</i>	interleaving (<i>cress_lotos</i> , <i>cress_sdl</i> , <i>cress_vxml</i>)
	suppress initials safety check (<i>cress_verify</i>)
<i>-k key</i>	authorisation key (<i>cress_deploy</i> , <i>cress_expand</i> , <i>cress_realise</i>)
<i>-l</i>	level indenting (<i>cress_bpel</i> , <i>cress_cpl</i> , <i>cress_lotos</i> , <i>cress_sdl</i> , <i>cress_vxml</i>)
	suppress livelock freedom check (<i>cress_verify</i>)
<i>-l library</i>	Lotos library (<i>cadp_annotate</i>)
<i>-m</i>	manual validation/verification (<i>cress_validate</i> , <i>cress_verify</i>)
<i>-m partners</i>	merge partners (<i>cress_bpel</i> , <i>cress_lotos</i>)
<i>-n number</i>	number of instances (<i>cress_lotos</i> , <i>cress_sdl</i>)
<i>-o timeout</i>	operational timeout (<i>cress_bpel</i>)
<i>-p</i>	suppress prompt count (<i>cress_lotos</i>)
	parameter check (<i>cress_sdl</i>)
	show BCG progress (<i>cress_verify</i>)
<i>-p mode[runs]</i>	performance test (<i>cress_validate</i>)
<i>-q</i>	qualified (per-scope) dispatchers (<i>cress_lotos</i>)
<i>-q qualifier</i>	qualifier for Clove macro names (<i>cress_validate</i> , <i>cress_verify</i>)
<i>-r</i>	redeploy (<i>cress_deploy</i>)
	repeat behaviour (<i>cress_lotos</i> , <i>cress_sdl</i>)
<i>-r relation</i>	reduction relation (<i>cress_verify</i>)
<i>-s</i>	swap diagram labels (<i>cress_check</i> , <i>cress_lotos</i>)
<i>-s file</i>	signature definitions (<i>cadp_annotate</i>)

Option	Purpose
-s size	font size (<i>chive_font</i>)
-t target	target language (<i>cress_check</i> , <i>cress_realise</i> , <i>cress_validate</i> , <i>cress_verify</i>)
-u	undeploy (<i>cress_deploy</i>)
	user-defined iteration (<i>cress_verify</i>)
-v vocabulary	vocabulary (most)
-w width	width of line (<i>cress_bpel</i> , <i>cress_check</i> , <i>cress_cpl</i> , <i>cress_lotos</i> , <i>cress_vxml</i>)

7.2 cadp_annotate

This script takes LOTOS files *<file>.lot* given on the command line, and converts them into a form suitable for CADP *<file>_cadp.lot*. It requires Lola/Topo to be available. Temporary files (**.asf*, **.lfe*, **.lsf*) are deleted after calling Lola/Topo.

The original specification must be valid Lotos, and must define sorts, operations and equations – through library import or through type definitions. The header and behaviour of the original specification are copied to the new specification.

The specification must conform to certain conventions (which Cress respects). Certain Lotos keywords must have an initial capital letters (**Behavior**, **Behaviour**, **Library**, **Type**, **Where**) and must not occur in comments with an initial upper-case letter. The top-level behaviour must end in **Where**, and each process instantiation in this must be on one line.

Sorts and operations are annotated according to the following flags which may be combined with logical "and" to get a composite result:

Flag	Meaning	Operation Name
1	constructor	
2	external	
4	implemented by	<i><RANGE>_<OPERATION></i> (operation) or <i><RANGE>_SORT</i> (sort)
8	compared by	<i><RANGE>_COMP</i>
16	iterated by	<i><RANGE>_FIRST</i> and <i><RANGE>_NEXT</i>
32	printed by	<i><RANGE>_PRINT</i>
128	domain name	used when two or more operations have the same range and name; this flag appends the domain sorts to qualify the name, e.g. <i>check : Bool,Nat -> Value</i> is mapped to <i>VALUE_CHECK_BOOL_NAT</i>

Common combinations of these are:

- sorts: 60 (printed iterated compared implemented)
- operations: 1 (constructor), 5 (implemented constructor), 6 (implemented external), 7 (implemented external constructor), 133 (internal implemented constructor).

Symbols in operation names are mapped to text as follows:

Symbol	Operation Name
+	PLUS
-	MINUS
*	STAR
/	SLASH
\	BSLASH

=	EQ
@	AT
~	TILDE
%	PCENT
^	CARET
&	AMP
#	HASH
.	DOT
<	LT
>	GT
{	LBRACE
}	RBRACE

The *set_signatures* subroutine defines hashes named *%signature_<library>*. The library name is what may be specified with the ‘-l’ option. The entries in such a hash have three forms:

- *<sort> => <flag>*
sort name, flag value (typically 60)
- ** : <domain1>,... -> <range> => <flag>*
any operation, operation domain, range, flag value (typically 5)
- *<operation> : <domain1>,... -> <range> => <flag>*
operation domain, range, flag value (typically 5)
- *<operation> : <domain1>,... -> <range> => <flag> <function>*
operation domain, range, flag value (typically 5), implementation function

A signature file in similar format can be specified on the command line with ‘-s’. This can assign new or replacement values to signature hashes, e.g.:

```
$signature_stir{"_eq_ : signal,signal -> bool"} = "6 SIGNAL_EQ";
```

There is a problem with the type conversion by Lola/Topo: nested types are taken to the top level. In the process, operation signatures can be duplicated. Although these are removed, expressions using alternative nested types can become ambiguous. These may need to be disambiguated with ‘**Of** <sort>’.

Types generated by Cress will be properly annotated through use of the *types.pl* file fed into *cadp_annotate* by *cress_expand*. Types defined in partner specification may need some additional information. Typically this applies to types that define their own sorts. The constructors of such a sort need to be stated explicitly. This can be achieved by annotating the partner type specifications as follows:

```
table (*! 1 *) : Nat,Text,Jobs -> Table
vehicle (*! $CON *) : Text,Number,Nat -> Vehicle
```

These are CADP-like annotations, but their contents are values understood by *cadp_annotate* (i.e. flags as might be used in the *set_signatures* subroutine). They are not true CADP annotations (though they are translated into these). The annotations are either numeric or symbolic and are evaluated as numeric expressions (e.g. ‘\$CON + \$IMP + \$EXT’).

By default, arrays, sets and strings can contain unlimited values. This will cause problems for verification with CADP. Their maximum size may therefore be set with the ‘-a’ option (e.g. ‘-a Deals’ to allow the default three values in this string, ‘-a Deals=1’ to allow only one value). If adding an element to such a type would exceed this maximum, the addition is ignored. Such an addition may be made through prefixing (‘~’), appending (‘~’) or concatenation (‘~~’) for arrays and strings, or through insertion (‘insert’) for sets.

Arrays and strings normally have ‘~’ as the main (prefix) constructor. When these types are limited, the constructor becomes ‘~-’. The main constructor for sets is ‘insert’ in both unlimited and limited cases. This aspect is handled automatically, substituting ‘~-’ for ‘~’ in equations, whether Cress-generated or partner-defined. To distinguish an instance of ‘~’ that may be substituted, it must be followed by the CADP-like annotation ‘(*! ~ *)’.

A side-effect of the type annotations is that ordinary Lotos comments are removed from the generated code.

Command-line options are as follows:

Option	Purpose
<i>-a limits</i>	generate <i><service>.lotos</i> with annotations for CADP; ‘.’ means no type limits, while a comma-separated list has <i>type=integer</i> for a specific limit or <i>type</i> for the default limit of 3 (default no annotations)
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-h</i>	print usage help
<i>-l library</i>	set Lotos library (default ‘stir’)
<i>-s file</i>	file of signature definitions

7.3 chive_font

This script normalises text in a Chive diagram by setting a standard font family, size and use of blanks.

Option	Purpose
<i>-b</i>	remove repeated blanks (default is not to remove)
<i>-f family</i>	specify font family (default ‘Arial Narrow’)
<i>-h</i>	print usage help
<i>-s size</i>	specify font size (default 10 point)
<i>file[.chx] ...</i>	Chive diagram files to update

7.4 cress_bpel

This script translates Cress diagram files given on the command line to BPEL and WSDL. These may optionally be given with a diagram suffix. Output of BPEL, PDD and WSDL files is to the directory containing the root diagram.

A file *<service>.extra* is treated as extra definitions to be included in *<service>_defs.wsdl*, and a file *<partner>.extra* is treated as extra definitions to be included in *<partner>.wsdl* (disregarding any *<parent>* suffix). Such a file can add *<binding>*, *<message>* and *<portType>* entries. However, it can add *<types>* only if these do not already exist in the WSDL being modified (since only one such section is permitted).

If the directory *<partner>* is found in the directory of its owning service, this will be treated as the implementation of the partner. Otherwise, for a web service only, a dummy implementation will be created. For a normal partner, the owning diagram is the last (highest) one in the hierarchy that refers to the diagram. For a phantom partner, the owning diagram is the first one in the hierarchy.

The following files are generated in the directory of the main diagram:

Service	Files
<i>main service</i>	<ul style="list-style-type: none"> • a BPEL file <i><service>.bpel</i> • a PDD (Process Deployment Descriptor) file <i><service>.pdd</i> • a WSDL common definitions file <i><service>_defs.wsdl</i> • a meta-information directory <i>META-INF</i>, used for a service deployment descriptor <i>deploy.wsdd</i> and for a WSDL catalogue <i>wsdlCatalog.xml</i> • a directory <i><Service>Defs</i> named after the service URN, containing common definitions in Java • a WSDL file <i><service>.wsdl</i> • a BPR (Business Process Archive) file <i><service>.bpr</i>
<i>partner services</i>	<ul style="list-style-type: none"> • a WSDL file <i><service>.wsdl</i>; merge (shared) partners are named <i><service>_<parent>.wsdl</i> • a WSR (Web Service Archive) file <i><service>.wsr</i> • a directory named after the partner URI, containing service skeleton and stub files in Java

Command-line options are as follows (*-a, -f, -i, -n, -p, -q, -r, -s* are unused and ignored):

Option	Purpose
<i>-b version</i>	BPEL version to generate (1 – BPEL4WS (default), 2 – WS-BPEL 2.0)
<i>-c</i>	provide comments
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-h</i>	print usage help
<i>-l</i>	levels of code shown by indenting (default is no level indenting)
<i>-m partners</i>	merge partners as a comma-separated list; this is needed only if one partner is shared by several business processes, the merged WSDL being stored in the last such diagram
<i>-o timeout</i>	operational test properties with server timeout in seconds generated for Mint
<i>-v vocab</i>	use the named vocabulary (<i>ds, gs, ws</i> ; default is current directory, e.g. <i>ws</i>)
<i>-w</i>	width of BPEL/WSDL output line in characters (default 80)
<i>file ...</i>	diagram files to translate into BPEL and WSDL (<i>FEATURES, PROFILES, TYPES</i> are special cases)

7.5 cress_check

This script parses and checks Cress diagram files given on the command line. These may optionally be given with a diagram suffix.

Command-line options are as follows:

Option	Purpose
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-h</i>	print usage help
<i>-s</i>	swap uses new diagram label (default is old diagram label)

<i>-t language</i>	target language (<i>bpel, cpl, lotos, sdl, vxml</i> ; needed for some vocabularies)
<i>-v vocab</i>	use the named vocabulary (<i>ds, gs, in, ivr, sip, voip, ws</i> ; default is current directory, e.g. <i>in</i>)
<i>file ...</i>	diagram files to check

7.6 cress_cpl

This script translates Cress a diagram file given on the command line to CPL. (Only one service diagram may be translated at a time.) The file may optionally be given with a diagram suffix. A single CPL file **.cpl* is generated in the directory of this diagram. In normal use, the configuration diagram deploys just one service diagram.

Command-line options are as follows (*-a, -b, -f, -i, -m, -n, -o, -p, -q, -r, -s* are unused and ignored):

Option	Purpose
<i>-c</i>	provide comments
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-h</i>	print usage help
<i>-l</i>	levels of code shown by indenting (default is no level indenting)
<i>-v vocab</i>	use the named vocabulary (<i>voip</i> ; default is current directory, e.g. <i>voip</i>)
<i>-w</i>	width of CPL output line in characters (default 80)
<i>file ...</i>	diagram file to translate into CPL (<i>FEATURES, PROFILES, TYPES</i> are special cases)

7.7 cress_create

Convert command-line base names to service archives. If *<base>.bpel* exists then conversion to a process BPR is done, else to a partner GAR/WSR. A BPEL process is bundled as BPEL, PDD and WSDL files into *<base>.bpr*. A partner grid service is converted from WSDL to a Java service, compiled with its implementation (if any), then turned into *<base>.gar*. A partner web service is converted from WSDL to a Java service, compiled with its implementation (if any), then turned into *<base>.wsr*.

If a partner name matches the regular expression given by the ‘-f’ option (default ‘_\$', i.e. a name ending with ‘_’), it is treated as an external implementation that is not managed by Cress. No attempt is then made to create a GAR or WSR file from it.

The following JAR files must exist in the Globus library directory (*\$GLOBUS_LOCATION/lib*): *addressing-1.0.jar, axis.jar, cog-jglobus.jar, commons-discovery.jar, jaxrpc.jar, saaj.jar, wsdl4j.jar, wsr_core.jar, wsr_core_stubs.jar, wss4j.jar*.

The following JAR files must exist in the Tomcat library directory (*\$CATALINA_HOME/shared/lib*): *activation.jar, axis.jar, commons-discovery.jar, commons-logging.jar, jaxrpc.jar, mail.jar, saaj.jar, wsdl4j.jar*.

For a BPEL process, the files used are *<base>.bpel, <base>.pdd* and all WSDL files in the current directory. These are copied to a subdirectory of the same name and built there. Test programs can use the classes defined in this subdirectory.

For a grid service partner, the implementation is created in a subdirectory named after the base file. For example, *converter_splitter.wsdl* causes files to be created in sub-directory *converter*. This directory must exist, and must contain implementation code in the form of Java files. These may exist in a package (e.g. *uk/ac/stir/cs/converter/*.java*). Any package and file names must be consistent with

what is created from the WSDL. The partner's WSDL file (plus any WSDL files that it imports) are copied to the `<partner>/schema` subdirectory and used in the build process.

For a partner service the following definition files are generated by Axis in a sub-directory of the current directory using the namespace (e.g. `urn:LoanStarDefs` implies directory `LoanStarDefs`, `http://my_approval.com/services` implies `com/my_approval/services`):

File	Purpose
<code><Type>.java</code>	class for each complex type
<code><Fault>Message.java</code>	class for each fault message

The implementation directory has the following structure:

File	Purpose
<code>jndi-config-deploy.xml</code> (supplied)	JNDI deployment file
<code>server-deploy.wsdd</code> (supplied)	web service deployment file
<code><directory>*.java</code> (supplied)	Java implementation
<code><service></code> (generated)	a sub-directory determine by the service namespace (e.g. <code>urn:AlterEgo</code> implies sub-directory <code>AlterEgo</code> , <code>http://conversions.com/services</code> implies <code>com/conversions/services</code>)
<code>lib</code> (generated/supplied)	JAR file containing all the compiled classes for the implementation and its data types; existing JAR files may be placed there prior to the build
<code>schema</code> (generated)	WSDL for the partner service and the definitions it relies on

The `<service>` sub-directory has the following files generated by Globus:

File	Purpose
<code><Service>Service.java</code>	service
<code><Service>ServiceAddressing.java</code>	service addressing interface
<code><Service>ServiceAddressingLocator.java</code>	service addressing
<code><Service>ServiceLocator.java</code>	service locator
<code><Port>Port.java</code>	port
<code><Port >BindingStub.java</code>	port binding client stub

For a web service partner, the implementation is created in a subdirectory named after the base file. For example, `approver.wsdl` causes files to be created in sub-directory `approver`. This directory must exist, and must contain implementation code in the form of Java files. These must exist in the relevant namespace directory (e.g. `BigDeal` must hold the implementation Java files such as `dealer1.java`). The package and file names must be consistent with the JNDI and service deployment files. The partner's WSDL file (plus any WSDL files that it imports) are copied to the `<partner>` subdirectory and used in the build process.

For a web service partner, the following service files are generated by AXIS in a sub-directory of the current directory using the service namespace URI (e.g. `urn:FirstRate` implies directory `FirstRate`, `http://my_approval.com/services` implies `com/my_approval/services`):

File	Purpose
<i>deploy.wsdd</i>	web service deployment descriptor
<i>undeploy.wsdd</i>	web service undeployment descriptor (unused by Cress)
<Service> <i>Service.java</i>	interface for each service
<Service> <i>ServiceLocator.java</i>	locator for each service
<Port> <i>Port.java</i>	interface to operations for each port
<Port> <i>BindingImpl.java</i>	service implementation template for each binding
<Port> <i>BindingSkeleton.java</i>	service skeleton for each binding
<Port> <i>BindingStub.java</i>	client stub for each binding

If files <base>.java exists, they are assumed to contain implementations for the service ports. All files <Name>BindingImpl.java are then deleted so that *.java are used. These must exist in the appropriate package directory for each port. For example if the ports are *Check* and *Loan*, and the service URN is *FirstRate*, the Java file(s) must contain classes *CheckBindingImpl* and *LoanBindingImpl* in package *FirstRate*. Note that these classes must not be public classes – omit any qualifier.

The Java compiler is currently called with the following options:

- warnings are ignored
- the target version of Java is set to 1.5 since ActiveBPEL 5.0.2 runs only under JRE 1.5.

Command-line options are as follows:

Option	Purpose
<i>-d</i>	deploy grid/web services
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-f partners</i>	foreign partners as a comma-separated list, not be to be created or deployed; for DS, partners whose names end with ‘_’ are always considered to be foreign
<i>-h</i>	print usage help
<i>-v vocab</i>	use the named vocabulary (<i>ds</i> , <i>gs</i> , <i>ws</i> ; default is current directory, e.g. <i>ws</i>)
<i>file[.*] ...</i>	files to be processed

7.8 cress_deploy

Deploy/undeploy/redeploy a business process/web service file using ActiveBPEL or a grid service using GT4. Files are tried in the order <base>.bpr, <base>.gar, <base>.wsr.

Note that it may take up 20 to 40 seconds for ActiveBPEL to notice the (re/un)deployment. If the service file is in use when it is re/undeployed, re/undeployment will be unsuccessful. It is then necessary to restart Tomcat. The environment variable BPEL_LOG can optionally be defined if deployment success should be checked.

Command-line options are as follows:

Option	Purpose
<i>-d</i>	deploy grid/web services (default)

Option	Purpose
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-h</i>	print usage help
<i>-k key</i>	key for CPL server authorisation <i>user:password@host</i>
<i>-r</i>	redeploy grid/web services (will be undeployed first if it exists)
<i>-u</i>	undeploy grid/web services (must exist)
<i>file[.*] ...</i>	files to be processed

7.9 cress_expand

This script updates the BPEL, Lotos, SDL or VoiceXML file given on the command line by expanding Cress macro calls. If a file does not have a recognised extension, it is silently ignored.

In general, the output is created from a file of the same name but with the extension *.base*. In the case of Lotos, the ‘-a’ option will also create *<file>_cadp.lot*.

In the case of GS or WS, a number of files are copied from *gs/<diagram>* or *ws/<diagram>* to *bpel/<diagram>*: **.bpel*, **.pdd*, **.properties*, **.wsdl*. **Note that directory *bpel/<diagram>* is removed before copying!** Also note that WSDL files older than the BPEL file are ignored and not copied. This is to ensure that WSDL files created on a previous run (e.g. due to a different feature selection) are ignored.

In the case of an SDL file, a phase follows the filename (and must be *parse* for any action to be taken). Later versions of SDT follow the phase by a directory (which is silently ignored here).

Cress macro calls appear in the file as:

Macro	Purpose
<i>Cress(Diag1,Diag2)</i>	expand diagrams Diag1 and Diag2
<i>Cress(Diag1:10,Diag2:20)</i>	ditto with feature priorities 10 and 20 respectively
<i>Cress(-e 2 -g -v sip,Diag1,Diag2)</i>	ditto with translator options -e 2 -g -v sip
<i>Cress(Features)</i>	expand feature definitions (from the <i>CONFIG_<VOCAB></i> diagram, extracting options/diagrams and recursing)
<i>Cress(Gates)</i>	expand gate definitions (from the <i>CONFIG_<VOCAB></i> diagram)
<i>Cress(Profiles)</i>	expand profile definitions (from the <i>CONFIG_<VOCAB></i> diagram)
<i>Cress(Types)</i>	expand type definitions (via the <i>TYPES</i> diagram)

In the case of an SDL file, a macro call in the file is preceded by macro and ended by semicolon. White space and comments are ignored in macro calls.

Command-line options are as follows:

Option	Purpose
<i>-d</i>	deploy grid, web or CPL services (not meaningful for other domains)
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-h</i>	print usage help

<i>-k key</i>	key for CPL server authorisation <i>user:password@host</i>
<i>-v vocab</i>	use the named vocabulary (<i>ds, gs, in, ivr, sip, voip, ws</i> ; default is current directory, e.g. <i>in</i>)
<i>file.bpel</i> <i>file.cpl</i> <i>file.lot</i> <i>file.pr phase</i> <i>file.vxml</i>	BPEL, CPL, Lotos or VoiceXML file to generate, or SDL file to generate plus Tau SDT analysis phase

7.10 cress_lotos

This script translates Cress diagram files given on the command line to Lotos. These may optionally be given with a diagram suffix. A single Lotos file **.lot* is generated in the directory of the last-named diagram. (If the configuration diagram defines the diagram list, note that diagrams are sorted by priority. The 'last-named' one may therefore not be the last in the list.)

For GS/WS, if the file *<partner>.lot* is found in the directory of its owning service, this will be treated as specifying the partner process. Otherwise a dummy process will be created. For a normal partner, the owning diagram is the last (highest) one in the hierarchy that refers to the diagram. For a phantom partner, the owning diagram is the first one in the hierarchy.

Command-line options are as follows (*-b, -o, -p* are unused and ignored):

Option	Purpose
<i>-a limits</i>	also generate <i><service>.lotos</i> with annotations for CADP; '.' means no type limits, while a comma-separated list has <i>type=integer</i> for a specific limit or <i>type</i> for the default limit of 3 (default no annotations)
<i>-c</i>	provide comments
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-f partners</i>	foreign partners as a comma-separated list; this is needed for partners considered to be external to the specification
<i>-h</i>	print usage help
<i>-i</i>	interleave parallel signals (default is serialise them)
<i>-l</i>	levels of code shown by indenting (default is no level indenting)
<i>-m partners</i>	merge partners as a comma-separated list; this is needed only if one partner is shared by several business processes, the merged processes being extracted to the top level of the specification
<i>-n number</i>	number of top-level call instances (default 3)
<i>-p</i>	suppress prompt count (default maintain IVR prompt count)
<i>-q</i>	qualified dispatchers (one per scope, default integrated)
<i>-r services</i>	repeat behaviour for given services; '.' means all services, while a comma-separated list identifies specific services (default is stop at a leaf node); when translating for CADP ('-a' option), despite this option there is no repeat inside a fault handler (including an implicit CatchAll)
<i>-s</i>	swap uses new diagram label (default is old diagram label)
<i>-v vocab</i>	the named vocabulary (<i>ds, gs, in, ivr, sip, ws</i> ; default is current directory, e.g. <i>in</i>)
<i>-w</i>	width of Lotos output line in characters (default 80)

<i>file ...</i>	diagram files to translate into Lotos (<i>FEATURES, GATES, PROFILES, TYPES</i> are special cases)
-----------------	--

7.11 *cress_realise*

This realises Cress creation (and deployment) for the domain given on the command line. The services/features deployed are those given in the corresponding configuration diagram.

Command-line options are as follows:

Option	Purpose
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-h</i>	print usage help
<i>-k key</i>	key for server authorisation (CPL, <i>user:password@host</i>)
<i>-t language</i>	target language (<i>bpel, cpl, lotos, sdl, vxml</i> ; must be given)
<i>-v vocab</i>	use the named vocabulary (<i>ds, gs, in, ivr, sip, voip, ws</i> ; must be given)
<i>file ...</i>	diagram files to check

7.12 *cress_sdl*

This script translates Cress diagram files given on the command line to SDL. These may optionally be given with a diagram suffix. A single SDL file **.pr* is generated in the directory of the last-named diagram. (If the configuration diagram defines the diagram list, note that diagrams are sorted by priority. The ‘last-named’ one may therefore not be the last in the list.)

Command-line options are as follows (*-a, -b, -f, -m, -o, -q* are unused and ignored):

Option	Purpose
<i>-c</i>	provide comments
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-h</i>	print usage help
<i>-i</i>	interleave parallel signals (default is serialise them, and is currently the only possibility)
<i>-l</i>	levels of code shown by indenting (default is no level indenting)
<i>-n number</i>	number of top-level call instances (default 3)
<i>-p</i>	parameter check on input (default is to allow any parameters); incorrect input is handled according to <i>-r</i> (repeat input or stop)
<i>-r services</i>	repeat behaviour for given services; ‘.’ means all services, while a comma-separated list identifies specific services (default is stop at a leaf node, other values not currently checked)
<i>-s</i>	swap uses new diagram label (default is old diagram label)
<i>-v vocab</i>	use the named vocabulary (<i>in, ivr, sip</i> ; default is current directory, e.g. <i>in</i>)
<i>-w</i>	width of SDL output line in characters (default 80)
<i>file ...</i>	diagram files to translate into SDL (<i>FEATURES, PROFILES, TYPES</i> are special cases)

7.13 cress_test

Distributions for JUnit and a JDBC driver (e.g. for MySQL) are required to use this script. The paths and JAR names are hard-coded in the *customise* subroutine.

The script runs the JUnit test suite for the BPEL services given on the command line. The test files may optionally be compiled first. The list of known services is defined in the *customise* subroutine. Test suites must be written in advance, and must be present in the *bpel/<service>/test* directory. Cress is supplied with test suites for its GS and WS service examples.

Command-line options are as follows:

Option	Purpose
<i>-c</i>	compile test files first
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>service ...</i>	services to test

7.14 cress_validate

This validates a Cress specification implied by the command line.

For Lotos, the specification is re-built according to the configuration diagram. Any features named for validation must be present in this. For SDL, the specification is *not* re-built. It must therefore have already been generated with the features to be validated.

Command-line options are as follows:

Option	Purpose
<i>-b memory</i>	bit state hash memory size in MB (default 5) – Lotos
<i>-d depth</i>	maximum depth of exploration (default 100) – Lotos
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-h</i>	print usage help
<i>-m</i>	manual (run tests manually, default automatic)
<i>-p mode[runs]</i>	performance test (c – concurrent, s – sequential), followed by the number of runs (default 40) – BPEL
<i>-q qualifier</i>	qualifier for macro names (default ")
<i>-t language</i>	target language (<i>bpel</i> , <i>lotos</i> , <i>sdl</i> ; must be given)
<i>-v vocabulary</i>	use the named vocabulary (<i>ds</i> , <i>gs</i> , <i>in</i> , <i>ivr</i> , <i>sip</i> , <i>ws</i> ; must be given)
<i>feature ...</i>	optional features to validate (all by default)

7.15 cress_verify

This verifies a Cress specification implied by the command line. Any features named for verification must be present in this.

For Lotos, the specification is re-built according to the configuration diagram. Any features named for verification must be present in this. Note that *<spec>.custom*, *<spec>.f*, *<spec>.lotos* and *<spec>.t* are moved to the temporary directory after verification.

Command-line options are as follows:

Option	Purpose
--------	---------

Option	Purpose
<i>-d</i>	suppress deadlock freedom check (checked by default)
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-h</i>	print usage help
<i>-i</i>	suppress initials safety check (checked by default)
<i>-l</i>	suppress livelock freedom check (checked by default)
<i>-m</i>	manual verification (automatic by default)
<i>-p</i>	show BCG progress (close window when done, default none)
<i>-q qualifier</i>	Clove macro qualifier prefix
<i>-r relation</i>	reduction relation (none, branch, strong – default, taucomp – tau compression with branching reduction, tauconf – tau confluence with branching reduction)
<i>-t language</i>	target language (lotos - no default)
<i>-u</i>	user iteration exists in <i>file.f</i> (generated from Clove by default)
<i>-v vocab</i>	use the named vocabulary (default is the basic filename ignoring any ‘_’ suffix, e.g. file ‘ws.lot’ has vocabulary ‘ws’, file ‘gs_matcher_scorer.lot’ has vocabulary ‘gs’)
<i>-x</i>	perform exit check (not checked by default)
<i>feature ...</i>	optional features to verify (all by default)

7.16 cress_vxml

This script translates Cress diagram files given on the command line to VoiceXML. These may optionally be given with a diagram suffix. A single VoiceXML file **.vxml* is generated in the directory of the last-named diagram. (If the configuration diagram defines the diagram list, note that diagrams are sorted by priority. The ‘last-named’ one may therefore not be the last in the list.)

Command-line options are as follows (*-a, -b, -f, -i, -m, -n, -o, -p, -q, -r, -s* are unused and ignored):

Option	Purpose
<i>-c</i>	provide comments
<i>-e level</i>	error report level (3 – internal errors, 2 – these plus user errors, 1 (default) – these plus informative notes, 0 – these plus diagnostics)
<i>-h</i>	print usage help
<i>-i</i>	interleave parallel actions (default is serialise them, and is currently the only possibility)
<i>-l</i>	levels of code shown by indenting (default is no level indenting)
<i>-v vocab</i>	use the named vocabulary (<i>ivr</i> ; default is current directory, e.g. <i>ivr</i>)
<i>-w</i>	width of VoiceXML output line in characters (default 80)
<i>file ...</i>	diagram files to translate into VoiceXML (<i>FEATURES, PROFILES, TYPES</i> are special cases)

8. Diagram Editors

The best alternative is to make use of the [Chive](#) graphical editor created for Cress. This is Java-based and should run on many platforms. Virtually all Cress diagrams have been provided in Chive XML format.

If Java is supported on your platform, you can use yEd instead. Download this from yWorks. Diagrams need to be saved in GML format for Cress to use them. yEd will read *.gml* format as well as its own *.ygf* format. The following types of symbols must be used:

comment	3D rectangle
rule box	rounded rectangle
node	anything else, but typically an ellipse

If you are able to run NextStep/OpenStep, you can edit and create Cress diagrams. You may be able to download *Diagram!* and a licence key from the web, though the author has local copies. The file *cress.dpalette2* is a palette for this diagram editor. The following types of symbols must be used:

comment	parallel lines
rule box	rounded rectangle
node	anything else, but typically an ellipse

Diagrams can be checked individually or in groups by running *cress_check*.

CRESS Syntax

9. Diagrams

9.1 Diagram Structure

Diagram names must consist of letters or underscores, finishing with a letter; the case of letters is not significant. The diagram names *A*, *Features*, *Gates*, *Profiles* and *Types* are reserved.

In signal names, British spelling ‘Dialogue’, ‘Dialled/Dialling’ and ‘Analyse’ may be used as well as American ‘Dialog’, ‘Dialed/Dialing’ and ‘Analyze’.

Diagrams are structured according to the following rules:

- Multiple plain arcs joining the same pair of nodes are counted as one.
- Multiple guarded arcs joining the same pair of nodes are counted as separate.
- In a list of guards, one may be designated as ‘Else’ to mean the negation of the other guards.
- A guard may have associated assignments. An arrow with assignments is treated as an empty guard with assignments.
- In a graph with loops, a Source or explicit Start node is required if the top node would be ambiguous. An implicit Start node is supplied if not given in an ordinary graph. A **Start** node may be used in a Source or Target.
- Node numeric labels may optionally end with one of ‘<’, ‘>’, ‘&’, then ‘!’, then ‘*’. An example is ‘<!’ or ‘>*’.
- A feature may define a template. The initial node of a template has a numeric label that ends with ‘+’ (append to matching node), ‘-’ (prefix to matching node), ‘=’ (replace matching node). The initial node defines a single event. Any binding associated with the start is appended to the corresponding original node. The template must end with a single Finish (or empty) node, though other non-empty leaf nodes are allowed. An event node whose numeric label ends with ‘!’ is not template-expanded (e.g. when a Dial event is for a PIN and not for an address). Apart from this, template event nodes are subject to further template expansion.
- If the direction of a signal is ambiguous (i.e. a node could be either input or output), the actual node (not signal) direction can be given explicitly by using ‘<’ (input) or ‘>’ (output) after the node number. A node number ending in ‘&’ is shorthand for an input node followed by an output node for the same signal content; the input node is suffixed ‘A’, while the output node is suffixed ‘B’.
- The diagram in an Arrow, Sink, Source or Target node may be omitted, meaning the current diagram.
- Arrow and Target nodes are distinguished as comprising alphabetic characters (including ‘_’ and ‘.’). Neither may have a binding. A Target node may optionally be followed by a boolean constraint.
- Update bindings for a signal may have to be disambiguated with extra slashes between them. Normally the rightmost space between bindings is taken as separator. Say ‘/ Last B <- A / Busy B <- Check C’ so that ‘A Busy’ is not combined.
- The binding in a Sink, Source or Swap may be omitted, meaning the current binding; useless bindings like ‘A <- A’ are ignored. If two arrows converge on the same Sink node, this may not have an associated binding. Instead, the Sink node must be repeated with just a single arrow to each copy.
- The binding in a Swap node is redundant and may be omitted.
- The label in a Source node may be given as ‘Start’ to mean the notional parent node in the root diagram.

- A feature diagram may have several Source nodes. A Source node may not mix guarded and unguarded descendants.
- As a translator option (-r), a leaf node is implicitly followed by the root diagram node. By default, a leaf node terminates behaviour.

9.2 Rule Boxes

A diagram may have a rule box that gives rules in the formats below.

The following gives optional address parameters and optional diagram names that are required by the current diagram:

```
Uses Address A,B / POTS
```

Parameters are cumulative, i.e. they are inherited by parent diagrams. A diagram will not be parsed more than once, even if referenced in different places. Diagrams are parsed lazily, e.g. a single top-level diagram reference may cause all dependent diagrams to be included.

The following declares and initialises a variable:

```
Variable := Expression
```

The following declares a function as a macro, to be replaced by its definition any place the function is used:

```
Function Parameters <- Definition
```

The function definition may not refer to the function. A later definition of a function replaces the earlier definition.

The following declares assignment(s) to be made when the signal occurs:

```
Signal Parameters / Variable Parameters <- Expression
```

The variable parameters must have been included in those of the signal. Several such rules may be given for a signal, and are cumulative. Such assignments are implicit. In addition, explicit assignments may also be given in an Event node. The latter override any implicit assignments for the same variables.

A signal may be replaced by another:

```
Signal Parameters / Signal Parameters
```

9.3 Node Labels

Node labels are used as follows:

- An input or output node normally has label <diagram>.<node number>.
- If a node is marked with '&', its implied input/output nodes have suffix 'A'/'B'.
- If an input or output node is reached by more than one path, it is labelled <to diagram>.<to node number>.<from diagram>.<from node number> (the two numbers are the same if the node loops back to itself).
- A node instantiated in a template is numbered <diagram>_<template instance number>.<node number>.
- An Empty node is labelled <diagram>_NULL.<noevent instance number> unless explicitly numbered.
- An Empty node instantiated in a template is labelled <diagram>_NULL_<template instance number>.<noevent instance number> unless explicitly numbered.
- A repeated guard is treated as a node numbered GUARD.<guard instance number>.

- A Source node is labelled [*<diagram>*]*<node number>*; it identifies the node in another graph to which something should be added.
- A Target node is labelled [*<diagram>*]*<node number>*; it identifies the node in another graph from which flow continues.
- A Swap node is labelled [*<diagram>*]*<node number>*; it identifies a matching node in another graph that is to be modified.
- An Arrow node is labelled *<link label>* (alphabetical, ‘_’ and ‘.’ characters); it identifies a matching Target node in the same graph from which flow continues.
- A Target node is labelled *<link label>* (alphabetical, ‘_’ and ‘.’ characters); it identifies a matching Arrow node in the same graph that is the source of the flow.

9.4 Diagram Parsing

Graphs are parsed as follows:

- Text from ‘//’ to the end of line is treated as a comment and removed. For such a comment to be recognised, ‘//’ must be at the start of a line or must be preceded by white space.
- The generated abstract syntax graph contains primary nodes corresponding to those of the original Cress diagram. Each primary node is linked by means of the ‘offspring’ field: a list of one or more primary nodes that follow. A labelled arrow corresponding to a guard or a replacement event node is separated and treated as a primary node.
- Each primary node has a parameter field pointing to a secondary node or node list corresponding to parameters. The secondary nodes have a subsidiary parameter node or node list.

Kind	Type	Label	Parameter
secondary	Assignment	<i>variable</i>	<i>expression</i>
secondary	Binding	<i>variable/expression</i>	<i>variable/expression</i>
secondary	Connector (1)	<i>DIAGRAM.number</i>	<i>Diagram</i> (no Bindings)
secondary	Diagram	<i>DIAGRAM</i>	Bindings
primary	Event (3)	<i>condition</i>	Assignments
primary	Guard	<i>condition/Else</i>	Assignments
primary	Null (4)	<i>DIAGRAM.number</i>	[]
secondary	Replacement (5)	<i>DIAGRAM.number</i>	Diagram
primary	Start	<i>DIAGRAM</i>	[]
primary/secondary	Statement (2)	<i>DIAGRAM.number</i>	Steps/Updates
secondary	Step (6)	<i>step_name</i>	<i>Par;Par;...</i>
secondary	Update	Step	Bindings/Assignments

1. Connector is one of ‘Arrow’, ‘Target’
2. Statement is one of ‘Action’, ‘Input’, ‘Output’
3. Event is vocabulary-defined
4. None is an ‘Empty’, ‘Finish’, ‘NoEvent’, ‘Null’, empty node
5. Replacement is one of ‘Sink’, ‘Source’, ‘Swap’
6. Step is vocabulary-defined

After a graph has been parsed, the Arrow and Target nodes disappear after being linked, while the Source and Swap nodes disappear after merging with the root graph. The Sink nodes remain in the composite graph.

9.5 Diagram Checking

Graphs are checked for consistency with the following rules:

- There must be a unique top node. This must be an Event or Start node for the whole diagram. It must be a Source for a feature diagram.
- A Source node may not be followed by a Guard unless a Swap node then follows.
- The diagram and node number in a Sink, Source or Swap must exist.
- Each Arrow node needs a Target node with the same label.
- An Empty node cannot be a leaf in an ordinary diagram, but must be the single leaf for a template. An Empty node cannot lead to itself directly.
- A leaf node must be an Event or Sink node.
- Event nodes must have a unique numeric node identifier.
- Signals must have valid names and the correct parameters.
- If multiple Inputs of the same signal follow a node, they must be distinguished by one or more corresponding parameters with different known values (otherwise the Inputs are non-deterministic).
- 'Else' may be used exactly once where there are multiple guards.

10. Expressions

10.1 General Rules

Expressions may use logical, arithmetic and string operators; these are translated as follows to a target language. Note that operators in Cress do not have an intrinsic precedence; this may be acquired only from the translation to another language. As a result, it is almost essential to parenthesise complex expressions in order to achieve the same result in all target languages.

10.2 Translation to BPEL

Cress	BPEL
(...)	(...)
- ~	- not
* \ %	* div mod
+ -	+ -
= != < <= >= >	= != < <= >= >
&&	and or
s1 After s2	substring-after(s1,s2)
s1 Before s2	substring-before(s1,s2)
Ceiling(n)	ceiling(n)
Concat(s1,s2)	concat(s1,s2)
Contains(s1,s2)	contains(s1,s2)
Decimal(s)	decimal(s)
Ends(s1,s2)	ends-with(s1,s2)

Cress	BPEL
False	false()
Floor(n)	floor(n)
Global Pars (LHS)	<assign> <copy> <from variable="name" part="name"/> <to variable="name" part="name"/> </copy> </assign>
Global Pars (RHS)	variable
Infinity	1 div 0E0
Length(s)	string-length(s)
Number(s)	number(s)
Round(n)	round(n)
Starts(s1,s2)	starts-with(s1,s2)
String(v)	string(v)
SubString(s,b)	substring(s,b)
SubString(s,b,e)	substring(s,b,e)
Translate(s1,s2,s3)	translate(s1,s2,s3)
True	true()

10.3 Translation to CPL

Cress	CPL
=	is
<	subdomain-of
~	contains

10.4 Translation to Lotos

Cress	Lotos
(...)	(...)
- ~	- not
* \ %	* / mod
+ -	+ -
= != < <= >= >	eq ne lt le ge gt (normally) = /= < <= >= > (IVR)
&& ^^	and or xor
s1 After s2	s1 after s2
Any (binding/expression)	AnyAddress
Any (event)	dummy

Cress	Lotos
s1 Before s2	s1 before s2
Ceiling(n)	ceiling(n)
Concat(s1,s2)	s1 ~ s2
Contains(s1,s2)	contains(s1,s2)
Decimal(s)	
Ends(s1,s2)	ends(s1,s2)
False	False
Floor(n)	floor(n)
Global Pars (LHS)	Stat!Write!Global!Pars!Val
Global Pars (RHS)	Stat!Read!Global!Pars?Val
If c Then e1 Else e2 Fi	conditional(c,e1,e2)
e In s	e isIn s
IndexOf(s1,s2)	indexOf(s1,s2)
Infinity	Infinity
Length(s)	length(s)
e NotIn s	e notIn s
Round(n)	round(n)
Slice(s,b)	slice(s,b)
Slice(s,b,e)	slice(s,b,e)
Starts(s1,s2)	starts(s1,s2)
String(v)	
SubStr (s,b,c)	substr (s,b,c)
SubString(s,b)	substring(s,b)
SubString(s,b,e)	substring(s,b,e)
Time	Stat!Read!Clock?Val
Translate(s1,s2,s3)	translate(s1,s2,s3)
True	True

10.5 Translation to SDL

Cress	SDL
(...)	(...)
- ~	- Not
* \ %	* / Mod
+ -	+ -
= != < <= >= >	= != < <= >= >
&& ^^	And Or Xor

Cress	SDL
s1 After s2	After(s1,s2)
Any (binding/expression)	AnyAddress
Any (event)	Dummy
False	False
Global Pars (LHS)	Output Update(Global,Pars)
Global Pars (RHS)	View(Global)(Pars)
If c Then e1 Else e2 Fi	If c Then e1 Else e2 Fi
e In s	e In s
IndexOf(s1,s2)	IndexOf(s1,s2)
Length(s)	Length(s)
e NotIn s	Not(e In s)
Slice(s,b)	Slice(s,b)
Slice(s,b,e)	Slice(s,b,e)
SubString(s,b)	SubString(s,b)
SubString(s,b,e)	SubString(s,b,e)
Time	Now
True	True

10.6 Translation to VoiceXML

Cress	VoiceXML
(...)	(...)
- ~	- !
* \ %	* / %
+ -	+ -
= != < <= > >=	= != < <= >= >
&& ^^	&& ^^
Any (binding/expression)	Any
Any (event)	Any
False	false
Global Pars (LHS)	<property name="variable" value="expr"/>
Global Pars (RHS)	variable
If c Then e1 Else e2 Fi	c ? e1 : e2
IndexOf(s1,s2)	s1.indexOf(s2)
Length(s)	s.length
Slice(s,b)	s.slice(b)
Slice(s,b,e)	s.slice(b,e)

SubString(s,b)	s.substring(b)
SubString(s,b,e)	s.substring(b,e)
True	true

CRESS Application Domains

11. Overview

See the [Cress home page](#) for an overview of Cress and some references. Cress (Chisel Representation Employing Systematic Specification) is designed for modular and extensible support of a variety of application domains. Currently it supports services for:

- DS (Devices Services, i.e. BPEL and WSDL)
- GS (Grid Services, i.e. BPEL and WSDL)
- IN (Intelligent Network)
- IVR (Interactive Voice Response, i.e. VoiceXML)
- SIP (Session Initiation Protocol, i.e. Internet Telephony)
- VoIP (Voice over Internet Protocol, i.e. Internet Telephony with SIP/CPL)
- WS (Web Services, i.e. BPEL and WSDL)

Cress is also designed to support a variety of target languages. Currently it supports translation to:

- BPEL/WSDL
- CPL
- Lotos
- SDL
- VoiceXML

though the supported translations vary among domains. These notes focus on how Cress is used with various application domains and, where relevant, how support varies among target languages.

12. DS (Device Services)

Device services resemble web services in allowing device-to-device communication using web-like mechanisms. A device service exists as a BPEL process. DS applications typically make use of features, but these must start with `<root>_` to allow feature diagrams to be distinguished from root diagrams during translation to Lotos.

A device service may receive *device_in* events from OSGi, and may send *device_out* events to OSGi – both via a SOAP proxy. OSG device events carry the following fields:

Argument	Purpose
arg1	message type
arg2	entity name
arg3	entity instance
arg4	message period
arg5	parameter values

These are mapped to web services as partner *entity_name*, port *in* or *out*, operation *message_type*. The remaining arguments (*entity_instance*, *message_period*, *parameter_values*) are mapped to a *device* record that is sent along with the web request. This type is predefined in Cress with name **Device**, fields *instance*, *period* and *params*.

A BPEL process first receives a device event from OSGi using ‘**Receive** *entity.in.message*’ or ‘**Receive** *entity.out.message*’. A BPEL process sends a device event to OSGi using ‘**Device** *entity.in.message*’ or ‘**Device** *entity.out.message*’. This is just a disguised Invoke: the **Device** partner name has ‘_’ appended to it to indicate the corresponding (‘shadow’) partner in OSGi.

12.1 Configuration Diagram

The configuration diagram starts with a **Deploys** line and then a blank line (see section 4).

Service configuration lines are then the same as for WS (section 18.1), except that a *HOME* partner must be defined first. This gives the base URI of the OSGi web service that supports shadow device partners. Subsequent partners in the DS configuration diagram correspond to business processes in ActiveBPEL.

When translating to BPEL, a second partner is automatically declared for each partner. This is similar to the main partner, but the partner name, prefix and URI have ‘_’ appended to them, and the URL is that of *HOME* (i.e. OSGi). These ‘shadow partners’ correspond to device classes in OSGi.

12.2 Services/Features

Most DS features are service-specific.

Feature	Purpose
CUPBOARD	monitor cupboard status (root)
DOOR	monitor door status (root)
DOOR_ALL	lock/unlock all doors (feature)
DOOR_LIGHT	turn on light on entering house (feature)
DOOR_LOUNGE	set lounge environment on entry (feature)
FALL	monitor fall status (root)
FALL_MOVEMENT	report alert if no movement after fall (feature)
HEATING	monitor heating status (root)
HEATING_FROST	report freezing alert if heating turned off when frosty (feature)
SPEECH	monitor speech as text (root)
SPEECH_HELP	provide weather/help spoken advice (feature)
WEATHER	dummy weather service (root)
WINDOW	monitor window status (root)

12.3 Signals

These are the same as for WS (section 18.3).

12.4 Actions

These are the same as for WS (section 18.4).

12.5 Events

These are the same as for WS (section 18.3).

12.6 Dynamic (Run-Time) Variables

These are the same as for WS (section 18.6).

12.7 Variable Types

These are the same as for WS (section 18.7) except that the following is also allowed:

Variable
Certificate

12.8 Specification Validation

Lotos Validation: performed using Mustard on a 2.67GHz i920 processor, 4Gb memory, Windows XP, CygWin 1.7.1-1, Lola 3.7.2 and Mustard 1.8.

Validation of *DOOR* and its features using the generated Lotos specification:

Test DOOR ALL Lock Doors ...	Pass	1 succ	0 fail	1.3 secs
Test DOOR ALL Unlock Doors ...	Pass	1 succ	0 fail	0.5 secs
Test DOOR LIGHT Front Open ...	Pass	1 succ	0 fail	0.4 secs
Test DOOR LIGHT Back Open ...	Pass	1 succ	0 fail	0.4 secs
Test DOOR LIGHT Other Open ...	Pass	1 succ	0 fail	0.4 secs
Test DOOR LOUNGE Dark Open ...	Pass	2 succ	0 fail	0.4 secs
Test DOOR LOUNGE Light Open ...	Pass	3 succ	0 fail	0.4 secs
Test DOOR LOUNGE Other Open ...	Pass	1 succ	0 fail	0.4 secs

Validation of *FALL* and its features using the generated Lotos specification:

Test FALL MOVEMENT Fall Alert ...	Pass	2 succ	0 fail	1.3 secs
-----------------------------------	------	--------	--------	----------

Validation of *HEATING* and its features using the generated Lotos specification:

Test HEATING FROST Off Freezing ...	Pass	1 succ	0 fail	1.2 secs
Test HEATING FROST Off Mild ...	Pass	1 succ	0 fail	0.4 secs

Validation of *SPEECH* and its features using the generated Lotos specification:

Test SPEECH HELP Weather ...	Pass	1 succ	0 fail	1.3 secs
Test SPEECH HELP Help Friend ...	Pass	1 succ	0 fail	0.4 secs
Test SPEECH HELP Help Doctor ...	Pass	1 succ	0 fail	0.4 secs
Test SPEECH HELP Help Other ...	Pass	1 succ	0 fail	0.5 secs
Test SPEECH HELP Other Request ...	Pass	1 succ	0 fail	0.4 secs

13. GS (Grid Services)

Grid services resemble web services in allowing application-to-application communication using web-like mechanisms. Grid services differ through use of resources and specific grid mechanisms. GS applications usually do not make use of features.

13.1 Configuration Diagram

The configuration diagram starts with a **Deploys** line and then a blank line (see section 4).

Service configuration lines are the same as for WS (section 18.1), except that each finishes with a resource declaration (a Cress variable declaration). If no resource is required, '-' must be given as the presence of a resource is how grid services are distinguished from web services.

13.2 Services/Features

Some GS features are general-purpose and could be used with many services, while others are service-specific.

Feature	Purpose
ANALYSER	conditional frequency analysis of occupational survey data (root, used by SPLITTER)
MATCHER	document comparison, based on clause length and word frequency (root, making use of SCORER)
SCORER	compute document clause length and word frequency (root, used by MATCHER)
SPLITTER	conditional frequency analysis of occupational survey data, splitting on some criterion (root, making use of ANALYSER)

13.3 Signals

These are the same as for WS (section 18.3).

13.4 Actions

These are the same as for WS (section 18.4).

13.5 Events

These are the same as for WS (section 18.5).

13.6 Dynamic (Run-Time) Variables

These are the same as for WS (section 18.6).

13.7 Variable Types

These are the same as for WS (section 18.6) except that the following is also allowed:

Variable
Certificate

13.8 Specification Validation

Lotos validation was performed using Mustard on a 2.67GHz i920 processor, 4Gb memory, Windows XP, CygWin 1.7.1-1, Lola 3.7.2 and Mustard 1.8.

Validation of *DOUBLEMAP* using the generated Lotos specification:

```

Test DOUBLEMAP No Records ... Pass 1 succ 0 fail 0.5 secs
Test DOUBLEMAP Invalid Position ... Pass 1 succ 0 fail 0.5 secs
Test DOUBLEMAP Map 2 To Any ... Pass 1 succ 0 fail 0.5 secs
Test DOUBLEMAP Map 2 To 4 ... Pass 1 succ 0 fail 0.6 secs

```

Validation of *LOOKUP* (and implicitly *ALLOCATOR*) using the generated Lotos specification:

```

Test ALLOCATOR Unknown Scheme ... Pass 3 succ 0 fail 1.5 secs
Test ALLOCATOR Unknown Job ... Pass 1 succ 0 fail 0.6 secs
Test ALLOCATOR SOC2000 Nurse ... Pass 1 succ 0 fail 0.5 secs

```

Test ALLOCATOR SIC92 Nurse ...	Pass	1 succ	0 fail	0.5 secs
Test LOOKUP Unknown Scheme1 ...	Pass	7 succ	0 fail	0.7 secs
Test LOOKUP Unknown Scheme2 ...	Pass	7 succ	0 fail	0.6 secs
Test LOOKUP Unknown Job ...	Pass	20 succ	0 fail	0.5 secs
Test LOOKUP Bookbinder Codes ...	Pass	1 succ	0 fail	0.6 secs
Test LOOKUP Nurse Codes ...	Pass	1 succ	0 fail	0.5 secs
Test LOOKUP Parallel Codes ...	Pass	2 succ	0 fail	0.7 secs

Validation of *MATCHER* (and implicitly *SCORER*) using the generated Lotos specification:

Test MATCHER No Clauses ...	Pass	191 succ	0 fail	9.1 secs
Test MATCHER One Shared ...	Pass	2 succ	0 fail	15.9 secs
Test MATCHER Nothing Shared ...	Pass	2 succ	0 fail	32.1 secs
Test MATCHER All Shared ...	Pass	2 succ	0 fail	27.9 secs
Test MATCHER Lengthy Text ...	Pass	2 succ	0 fail	9.0 mins

Validation of *SPLITTER* (and implicitly *ANALYSER*) using the generated Lotos specification:

Test SPLITTER No Jobs ...	Pass	2 succ	0 fail	1.5 secs
Test SPLITTER Invalid Query ...	Pass	16 succ	0 fail	0.6 secs
Test SPLITTER Valid Query ...	Pass	31 succ	0 fail	0.6 secs
Test SPLITTER No Plumbers ...	Pass	2 succ	0 fail	0.7 secs
Test SPLITTER All Plumbers ...	Pass	2 succ	0 fail	0.6 secs
Test SPLITTER Plumber Welder ...	Pass	2 succ	0 fail	0.9 secs
Test SPLITTER Mixed Jobs ...	Pass	2 succ	0 fail	1.3 secs
Test SPLITTER Female Jobs ...	Pass	2 succ	0 fail	1.7 secs
Test SPLITTER Over 20 Jobs ...	Pass	2 succ	0 fail	1.4 secs
Test SPLITTER Under 20 Jobs ...	Pass	2 succ	0 fail	1.3 secs
Test SPLITTER Sequential Use ...	Pass	7 succ	0 fail	10.4 secs
Test SPLITTER Concurrent Use ...	Pass	14 succ	0 fail	10.9 secs

13.9 Specification Verification

Lotos Verification: performed using CADP 2009-c on a 2.67GHz i920 processor, 4Gb memory, Windows XP, CygWin 1.7.7-1 and Clove 1.3.

Verification of *DOUBLEMAP* using the annotated Lotos specification and options ‘-x -r taucomp’:

Generating properties for DOUBLEMAP ...	CPU Time (Real Time)
Generating graph for DOUBLEMAP ...	22.0 secs (1.7 mins)
	(states/transitions/labels: 15/19/7 -> 4/6/7)
Verifying DOUBLEMAP Missing Records ...	Success 6.6 secs (12.0 secs)
Verifying DOUBLEMAP Wrong Occupation Position ...	Success 6.5 secs (11.0 secs)
Verifying DOUBLEMAP Any Map Response ...	Success 6.5 secs (12.0 secs)
Verifying DOUBLEMAP General Map Response ...	Success 6.5 secs (12.0 secs)
Verifying DOUBLEMAP Map Position 2 To 4 ...	Success 6.4 secs (11.0 secs)
Verifying DOUBLEMAP Livelock Freedom ...	Success 6.6 secs (14.0 secs)
Verifying DOUBLEMAP Initials Safety ...	Success 6.4 secs (11.0 secs)

Verification of *LOOKUP* (and implicitly *ALLOCATOR*) using the annotated Lotos specification and options ‘-d -x -r taucomp’:

Generating properties for ALLOCATOR ...	CPU Time (Real Time)
Generating graph for ALLOCATOR ...	22.0 secs (2.1 mins)
	(states/transitions/labels: 39/55/11 -> 2/10/11)
Verifying ALLOCATOR Livelock Freedom ...	Success 6.3 secs (15.0 secs)
Verifying ALLOCATOR Initials Safety ...	Success 6.4 secs (12.0 secs)
Verifying ALLOCATOR Always Exit ...	Success 6.5 secs (12.0 secs)
Generating properties for LOOKUP ...	CPU Time (Real Time)
Generating graph for LOOKUP ...	22.1 secs (2.1 mins)
	(states/transitions/labels: 49/221/7 -> 5/6/7)
Verifying LOOKUP Unknown Scheme1 ...	Success 6.5 secs (12.0 secs)
Verifying LOOKUP Unknown Scheme2 ...	Success 6.2 secs (12.0 secs)
Verifying LOOKUP Unknown Job ...	Success 6.3 secs (11.0 secs)
Verifying LOOKUP Any Translation Response ...	Success 6.4 secs (12.0 secs)
Verifying LOOKUP General Translation Response ...	Success 6.9 secs (13.0 secs)
Verifying LOOKUP Bookbinder Codes ...	Success 6.5 secs (12.0 secs)
Verifying LOOKUP Livelock Freedom ...	Success 6.3 secs (14.0 secs)

Verifying LOOKUP Initials Safety ...	Success	6.5 secs	(12.0 secs)
Verifying LOOKUP Always Exit ...	Success	6.3 secs	(12.0 secs)

Verification of *MATCHER* (and implicitly *SCORER*) using the annotated Lotos specification and options '-d -x -r taucomp':

Generating properties for SCORER ...		CPU Time	(Real Time)
Generating graph for SCORER ...		22.1 secs	(57.0 secs)
	(states/transitions/labels: 6/13/10 -> 2/9/10)		
Verifying SCORER Livelock Freedom ...	Success	6.2 secs	(10.0 secs)
Verifying SCORER Initials Safety ...	Success	6.2 secs	(7.0 secs)
Verifying SCORER Always Exit ...	Success	6.0 secs	(7.0 secs)
Generating properties for MATCHER ...		CPU Time	(Real Time)
Generating graph for MATCHER ...		22.4 secs	(1.7 mins)
	(states/transitions/labels: 36928/100242/74 -> 11/73/74)		
Verifying MATCHER Missing First Text ...	Success	6.3 secs	(7.0 secs)
Verifying MATCHER Missing Second Text ...	Success	6.4 secs	(8.0 secs)
Verifying MATCHER Any Metric Response ...	Success	6.4 secs	(7.0 secs)
Verifying MATCHER General Metric Response ...	Success	6.3 secs	(8.0 secs)
Verifying MATCHER Specific Texts ...	Success	6.3 secs	(8.0 secs)
Verifying MATCHER Same Texts ...	Success	6.3 secs	(7.0 secs)
Verifying MATCHER Livelock Freedom ...	Success	6.4 secs	(10.0 secs)
Verifying MATCHER Initials Safety ...	Success	6.3 secs	(7.0 secs)
Verifying MATCHER Always Exit ...	Success	6.1 secs	(7.0 secs)

Verification of *SPLITTER* (and implicitly *ANALYSER*) using the annotated Lotos specification and options '-d -x -r taucomp':

Generating properties for ANALYSER ...		CPU Time	(Real Time)
Generating graph for ANALYSER ...		22.6 secs	(59.0 secs)
	(states/transitions/labels: 81/81/82 -> 2/81/82)		
Verifying ANALYSER Livelock Freedom ...	Success	6.5 secs	(9.0 secs)
Verifying ANALYSER Initials Safety ...	Success	6.2 secs	(8.0 secs)
Verifying ANALYSER Always Exit ...	Success	6.3 secs	(7.0 secs)
Generating properties for SPLITTER ...		CPU Time	(Real Time)
Generating graph for SPLITTER ...		22.6 secs	(1.5 mins)
	(states/transitions/labels: 66/285/7 -> 5/6/7)		
Verifying SPLITTER Invalid Query ...	Success	6.4 secs	(7.0 secs)
Verifying SPLITTER No Jobs ...	Success	6.5 secs	(8.0 secs)
Verifying SPLITTER No Plumbers ...	Success	6.3 secs	(7.0 secs)
Verifying SPLITTER Any Analysis Response ...	Success	6.3 secs	(8.0 secs)
Verifying SPLITTER General Analysis Response ...	Success	6.5 secs	(8.0 secs)
Verifying SPLITTER Female Jobs ...	Success	6.6 secs	(7.0 secs)
Verifying SPLITTER Livelock Freedom ...	Success	6.4 secs	(10.0 secs)
Verifying SPLITTER Initials Safety ...	Success	6.3 secs	(7.0 secs)
Verifying SPLITTER Always Exit ...	Success	6.5 secs	(7.0 secs)

13.10 Implementation Validation

BPEL Validation: performed using Mustard on a 2.67GHz i920 processor, 4Gb memory, Windows XP, CygWin 1.7.1-1, Mint 1.2 and Mustard 1.8.

Validation of *DOUBLEMAP* using the generated Bpel implementation:

Test DOUBLEMAP No Records ...	Pass	1 succ	0 fail	1.4 secs
Test DOUBLEMAP Invalid Position ...	Pass	1 succ	0 fail	0.5 secs
Test DOUBLEMAP Map 2 To Any ...	Pass	1 succ	0 fail	0.4 secs
Test DOUBLEMAP Map 2 To 4 ...	Pass	1 succ	0 fail	0.5 secs

Validation of *LOOKUP* (and implicitly *ALLOCATOR*) using the generated Bpel implementation:

Test ALLOCATOR SOC2 Nurse ...	Pass	1 succ	0 fail	0.5 secs
Test ALLOCATOR SIC Nurse ...	Pass	1 succ	0 fail	0.6 secs
Test ALLOCATOR SOC2 Unknown ...	Pass	1 succ	0 fail	0.6 secs
Test ALLOCATOR Unknown Nurse ...	Pass	1 succ	0 fail	0.5 secs
Test LOOKUP SOC SIC Bookbinder ...	Pass	1 succ	0 fail	0.6 secs

Test LOOKUP SOC SIC Nurse ...	Pass	1 succ	0 fail	0.6 secs
Test LOOKUP SOC SIC Parallel ...	Pass	4 succ	0 fail	0.9 secs
Test LOOKUP SOC SIC Unknown ...	Pass	1 succ	0 fail	0.6 secs
Test LOOKUP Unknown SIC Nurse ...	Pass	1 succ	0 fail	0.5 secs
Test LOOKUP SOC Unknown Nurse ...	Pass	1 succ	0 fail	0.7 secs

Validation of *MATCHER* (and implicitly *SCORER*) using the generated Bpel implementation:

Test MATCHER No Clauses ...	Pass	1 succ	0 fail	0.7 secs
Test MATCHER One Shared ...	Pass	1 succ	0 fail	0.6 secs
Test MATCHER Nothing Shared ...	Pass	1 succ	0 fail	0.6 secs
Test MATCHER All Shared ...	Pass	1 succ	0 fail	0.6 secs
Test MATCHER Lengthy Text ...	Pass	1 succ	0 fail	0.7 secs

Validation of *SPLITTER* (and implicitly *ANALYSER*) using the generated Bpel implementation:

Test SPLITTER No Jobs ...	Pass	1 succ	0 fail	1.7 secs
Test SPLITTER Invalid Query ...	Pass	1 succ	0 fail	0.7 secs
Test SPLITTER Valid Query ...	Pass	1 succ	0 fail	0.8 secs
Test SPLITTER No Plumbers ...	Pass	1 succ	0 fail	0.8 secs
Test SPLITTER All Plumbers ...	Pass	1 succ	0 fail	0.8 secs
Test SPLITTER Plumber Welder ...	Pass	1 succ	0 fail	1.1 secs
Test SPLITTER Mixed Jobs ...	Pass	1 succ	0 fail	0.9 secs
Test SPLITTER Female Jobs ...	Pass	1 succ	0 fail	1.0 secs
Test SPLITTER Over 20 Jobs ...	Pass	1 succ	0 fail	1.0 secs
Test SPLITTER Under 20 Jobs ...	Pass	1 succ	0 fail	1.0 secs
Test SPLITTER Sequential Use ...	Pass	1 succ	0 fail	1.6 secs
Test SPLITTER Concurrent Use ...	Pass	4 succ	0 fail	3.0 secs

14. IN (Intelligent Network)

The Intelligent Network was designed to allow easy development and deployment of features in addition to the basic call. It is governed by the ITU Q.1200 series of recommendations. IN features are typically implemented in Service Control Points using specialised service creation software.

14.1 Configuration Diagram

The configuration diagram starts with a **Deploys** line and then a blank line (see section 4).

Feature configuration lines are then per subscriber and have the form:

feature number parameters

for example:

INFR 6 **From Any To 9 Start 8 Finish 18**

The configuration lines vary according to the feature, so here are concrete examples for CC (Charge Call), CFBL (Call Forward Busy Line), CFU (Call Forward Unconditional), CW (Call Waiting), INCF (Intelligent Network Call Forward), INFB (Intelligent Network Freephone Billing), INFR (Intelligent Network Freephone Routing), INTL (Intelligent Network Teen Line) and TCS (Terminating Call Screening). RC (return call) and TWC (three-way calling) are service options are not configured per subscriber.

CC	4	Pin 5	// 4 accepts charge requests with PIN 5
CFBL	3	To 5	// 3 forwards on busy to 5
CFU	5		// 5 forwards unconditionally
CND	3		// 3 has caller display
CW	2		// 2 has call waiting
INCF	5	To 10	// 5 forwards to 10
INFB	1		// 1 pays for calls it receives
INFR	6	From Any To 9 Start 8 Finish 18	// 6 forwards from any number to 9, 08.00–18.00
INTL	2	Pin 3 Start 9 Finish 17	// 2 needs PIN 3, 09:00–17:00

14.2 Services/Features

The following features for the IN are closely patterned after those in the first Feature Interaction Contest.

Feature	Purpose
CC	Charge Card (feature)
CFBL	Call Forward Busy Line (feature)
CND	Calling Number Display (feature)
INCF	Intelligent Network Call Forward (feature)
INFB	Intelligent Network FreePhone Billing (feature)
INFR	Intelligent Network FreePhone Routing (feature)
INTL	Intelligent Network Teen Line (feature)
POTS	Plain Old Telephone Service (root)
RC	Return Call (feature)
TCS	Terminating Call Screening (feature)
TWC	Three-Way Call (feature)

14.3 Signals

Signal	Parameters
SCP to Switch	
AnalyzeRoute (1)	Address,Address,Address,Address
Continue (1,2)	Address,Address,Address
ForwardCall (1)	Address,Address,Address
Resource (1,3)	Address,Address
SendToResource (1)	Address,Address,Message
Terminate (1,3)	Address,Address
Switch to Billing	
AirBegin	Address,Time
AirEnd	Address,Time
LogBegin	Address,Address,Address,Time
LogEnd	Address,Address,Time
Switch to SCP	
InfoAnalyzed (4,6)	Address,Address,Address
InfoCollected (4)	Address,Address,Address
NetworkBusy (4)	Address,Address,Address
OriginationAttempt (4)	Address,Address,Address
TerminationAttempt (4)	Address,Address,Address

Signal	Parameters
Resource (4)	Address,Address
ResourceAbort (4,5)	Address,Address
Switch to Status Manager	
StartBilling (7)	Address,Address
StopBilling (7)	Address,Address
UpdateAA (7,8)	Status,Address,Address
UpdateAB (7,8)	Status,Address,Boolean
UpdateAL (7,8)	Status,Address,List
UpdateAT (7,8)	Status,Address,Time
UpdateAAA (7,8)	Status,Address,Address,Address
UpdateAAB (7,8)	Status,Address,Address,Boolean
UpdateAAT (7,8)	Status,Address,Address,Time
Switch to User	
Announce	Address,Message
DialTone	Address
Disconnect	Address,Address
Display	Address,Message
LineBusyTone	Address
StartAudibleRinging (9)	Address,Address
StartCallWaitingTone (9)	Address,Address
StartRinging (9,10)	Address,Address[,Address]
StopAudibleRinging (9)	Address,Address
StopCallWaitingTone (9)	Address,Address
StopRinging (9)	Address,Address
UnobtainableTone	Address
User to Switch	
Answer (11)	Address
Dial	Address,Address
Flash	Address
OffHook	Address
OnHook	Address

1. Preceded in Chisel by 'Response' but automatically removed. Since 'Response Disconnect' is then ambiguous, 'Terminate' is used instead.
2. Translated as 'Continued' in SDL since 'Continue' is a reserved word.
3. Incorrectly shown in Chisel as having three address parameters.
4. Preceded in Chisel by 'Trigger' but automatically removed.
5. Not defined in Chisel but needed on hang-up after 'SendToResource'.

6. British spelling 'Analyse' accepted but converted to 'Analyze'.
7. Not defined in Chisel.
8. Used in the SDL translation.
9. As in Chisel, a space after 'Start'/'Stop' is removed.
10. The optional 'address' is the code for a cadence.
11. Not defined in Chisel, but needed to disambiguate 'OffHook'.

14.4 Static (Profile) Variables

Variable	Parameters	Result
CallingNumber (4)	Address	Boolean
CallWaiting	Address	Boolean
Cellular	Address	Boolean
ForwardBusy (2)	Address	Address
ForwardTo	Address	Address
Freephone (4)	Address	Boolean
RedirectAddress	Address,Address	Address
RedirectTime1	Address,Address	Time
RedirectTime2	Address,Address	Time
ScreenIn	Address	Addresses
ScreenOut (3)	Address	Addresses
TeenPIN	Address	Address
TeenTime1	Address	Time
TeenTime2	Address	Time

1. *Charge* in Chisel.
2. *BLForward* in Chisel.
3. *Screened* in Chisel.
4. Not defined in Chisel.

14.5 Dynamic (Run-Time) Variables

Variable	Parameters	Result
AudibleRinging	Address,Address	Boolean
Bill (2)	Address,Address	Address
Busy	Address	Boolean
By (2)	Address,Address	Address
Dialing (1)	Address	Boolean
LastIncoming	Address	Address
ReturnCall	Address,Address	Boolean
Ringing	Address,Address	Boolean
ThreeWay	Address	Boolean

1. British spelling ‘Dialling’ accepted but converted to ‘Dialing’.
2. Not defined in Chisel.

14.6 Announcement Messages

Variable
AnyMessage
AskForPIN (1)
EnterPhoneNumber
EnterPIN
InvalidPIN
ScreenedMessage

1. Accepted as in Chisel, but converted to ‘EnterPIN’.

14.7 Variable Types

Variable
Address
Message
Time

14.8 Specification Validation

Lotos Validation: performed using Mustard on a 2.67GHz i920 processor, 4Gb memory, Windows XP, CygWin 1.7.1-1, Lola 3.7.2 and Mustard 1.8.

Validation of *POTS* and its features using the generated Lotos specification:

Test POTS Clear Before Dial ...	Pass	3 succ	0 fail	1.7 secs
Test POTS Clear Before Answer ...	Pass	3 succ	0 fail	0.7 secs
Test POTS Caller Clear ...	Pass	6 succ	0 fail	0.7 secs
Test POTS Callee Clear ...	Pass	6 succ	0 fail	0.7 secs
Test POTS Call Self ...	Pass	3 succ	0 fail	0.6 secs
Test POTS Call Busy ...	Pass	11 succ	0 fail	0.8 secs
Test POTS Simultaneous Call ...	Pass	11 succ	0 fail	0.9 secs
Test INTL No Teen Line ...	Pass	6 succ	0 fail	0.7 secs
Test INTL Early ...	Pass	6 succ	0 fail	0.7 secs
Test INTL No PIN ...	Pass	3 succ	0 fail	0.6 secs
Test INTL Wrong PIN ...	Pass	3 succ	0 fail	0.5 secs
Test INTL Right PIN ...	Pass	6 succ	0 fail	0.7 secs
Test TWC Add1 Hang321 ...	Pass	3 succ	0 fail	0.9 secs
Test TWC Add2 Hang123 ...	Pass	6 succ	0 fail	1.0 secs
Test TWC Add1 Hang3 Ring2 ...	Pass	3 succ	0 fail	0.9 secs
Test TWC Add1 Hang31 Ring2 ...	Pass	6 succ	0 fail	0.9 secs
Test TWC Add1 Busy3 Hang21 ...	Pass	6 succ	0 fail	0.9 secs
Test TWC Add2 Busy3 End2 Hang31 ...	Pass	6 succ	0 fail	0.9 secs
Test TWC Add2 Hang312 ...	Pass	6 succ	0 fail	1.0 secs
Test TWC Add2 End2 Hang312 ...	Pass	6 succ	0 fail	0.9 secs
Test TWC Add2 End2 Hang231 ...	Pass	3 succ	0 fail	0.9 secs
Test TWC Add2 End2 Hang123 ...	Pass	6 succ	0 fail	0.9 secs
Test TWC Add2 Hang1 Hang32 ...	Pass	6 succ	0 fail	0.9 secs
Test TWC Add2 Hang2 Ring2 Hang12 ...	Pass	6 succ	0 fail	0.9 secs
Test TWC Add2 Hang2 Ring2 Hang1 ...	Pass	3 succ	0 fail	0.9 secs
Test TWC End2 Hang12 ...	Pass	6 succ	0 fail	0.7 secs
Test TWC Add2 Hang12 ...	Pass	6 succ	0 fail	0.9 secs
Test TWC Hang1 Add2 Hang32 ...	Pass	6 succ	0 fail	1.1 secs

Test TWC Hang2 Ring2 Hang1 ...	Pass	3 succ	0 fail	0.8 secs
Test TWC Add2 Hang21 ...	Pass	6 succ	0 fail	1.1 secs
Test TWC Add2 Hang132 ...	Pass	6 succ	0 fail	0.9 secs
Test TWC Add2 End2 Hang31 ...	Pass	6 succ	0 fail	0.9 secs
Test TWC Add1 Hang3 End1 Hang12 ...	Pass	6 succ	0 fail	0.9 secs
Test TWC Add2 Hang2 Ring2 Hang21 ...	Pass	3 succ	0 fail	0.9 secs
Test TWC Add1 Hang1 Ring1 Hang2 ...	Pass	3 succ	0 fail	0.8 secs
Test CC No Account ...	Pass	3 succ	0 fail	0.6 secs
Test CC No PIN ...	Pass	3 succ	0 fail	0.6 secs
Test CC Wrong Account ...	Pass	3 succ	0 fail	0.6 secs
Test CC Wrong PIN ...	Pass	3 succ	0 fail	0.6 secs
Test CC Right PIN ...	Pass	6 succ	0 fail	0.7 secs
Test RC Call Back ...	Pass	10 succ	0 fail	0.9 secs
Test RC No Caller ...	Pass	3 succ	0 fail	0.6 secs
Test CND Display ...	Pass	6 succ	0 fail	0.7 secs
Test CND No Display ...	Pass	6 succ	0 fail	0.7 secs
Test INFB Forward ...	Pass	6 succ	0 fail	0.7 secs
Test INFB No Forward ...	Pass	6 succ	0 fail	0.7 secs
Test INFR Early ...	Pass	6 succ	0 fail	0.7 secs
Test INFR Any Forward ...	Pass	9 succ	0 fail	0.9 secs
Test INFR No Forward ...	Pass	6 succ	0 fail	0.7 secs
Test INFR Not Caller ...	Pass	6 succ	0 fail	0.9 secs
Test INFR Caller Forward ...	Pass	9 succ	0 fail	0.9 secs
Test INCF Forward ...	Pass	9 succ	0 fail	0.8 secs
Test INCF No Forward ...	Pass	6 succ	0 fail	0.8 secs
Test CFBL Busy Forward ...	Pass	23 succ	0 fail	1.3 secs
Test CFBL No Forward ...	Pass	11 succ	0 fail	0.7 secs
Test CFBL Free ...	Pass	6 succ	0 fail	0.8 secs
Test TCS Reject ...	Pass	3 succ	0 fail	0.7 secs
Test TCS No Reject ...	Pass	6 succ	0 fail	1.0 secs
Test TCS No Screen ...	Pass	6 succ	0 fail	0.8 secs

SDL Validation: performed using Mustard on a 3.8GHz/1Gb Pentium with Windows XP, CygWin 1.5.24-2, Mustard 1.3 and Tau 4.6.

SDL validation of *POTS* and its features using the generated SDL specification:

Test POTS Clear Before Dial ...	Pass	1 succ	0 fail	0.0 secs
Test POTS Clear Before Answer ...	Pass	1 succ	0 fail	0.1 secs
Test POTS Caller Clear ...	Pass	1 succ	0 fail	0.1 secs
Test POTS Callee Clear ...	Pass	1 succ	0 fail	0.1 secs
Test POTS Call Self ...	Pass	1 succ	0 fail	0.0 secs
Test POTS Call Busy ...	Pass	1 succ	0 fail	0.1 secs
Test POTS Simultaneous Call ...	Pass	1 succ	0 fail	0.0 secs
Test INTL No Teen Line ...	Pass	1 succ	0 fail	0.1 secs
Test INTL Early ...	Pass	1 succ	0 fail	0.0 secs
Test INTL No PIN ...	Pass	1 succ	0 fail	0.1 secs
Test INTL Wrong PIN ...	Pass	1 succ	0 fail	0.0 secs
Test INTL Right PIN ...	Pass	1 succ	0 fail	0.0 secs
Test TWC Add1 Hang321 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add2 Hang123 ...	Pass	1 succ	0 fail	0.0 secs
Test TWC Add1 Hang3 Ring2 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add1 Hang31 Ring2 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add1 Busy3 Hang21 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add2 Busy3 End2 Hang31 ...	Pass	1 succ	0 fail	0.0 secs
Test TWC Add2 Hang312 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add2 End2 Hang312 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add2 End2 Hang231 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add2 End2 Hang123 ...	Pass	1 succ	0 fail	0.0 secs
Test TWC Add2 Hang1 Hang32 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add2 Hang2 Ring2 Hang12 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add2 Hang2 Ring2 Hang1 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC End2 Hang12 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add2 Hang12 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Hang1 Add2 Hang32 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Hang2 Ring2 Hang1 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add2 Hang21 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add2 Hang132 ...	Pass	1 succ	0 fail	0.1 secs
Test TWC Add2 End2 Hang31 ...	Pass	1 succ	0 fail	0.1 secs

```

Test TWC Add1 Hang3 End1 Hang12 ... Pass          1 succ    0 fail    0.1 secs
Test TWC Add2 Hang2 Ring2 Hang21 ... Pass          1 succ    0 fail    0.1 secs
Test TWC Add1 Hang1 Ring1 Hang2 ... Pass          1 succ    0 fail    0.0 secs
Test CC No Account ... Pass          1 succ    0 fail    0.1 secs
Test CC No PIN ... Pass          1 succ    0 fail    0.0 secs
Test CC Wrong Account ... Pass          1 succ    0 fail    0.1 secs
Test CC Wrong PIN ... Pass          1 succ    0 fail    0.0 secs
Test CC Right PIN ... Pass          1 succ    0 fail    0.1 secs
Test RC Call Back ... Pass          1 succ    0 fail    0.1 secs
Test RC No Caller ... Pass          1 succ    0 fail    0.1 secs
Test CND Display ... Pass          1 succ    0 fail    0.0 secs
Test CND No Display ... Pass          1 succ    0 fail    0.1 secs
Test INFB Forward ... Pass          1 succ    0 fail    0.1 secs
Test INFB No Forward ... Pass          1 succ    0 fail    0.0 secs
Test INFR Early ... Pass          1 succ    0 fail    0.0 secs
Test INFR Any Forward ... Pass          1 succ    0 fail    0.0 secs
Test INFR No Forward ... Pass          1 succ    0 fail    0.0 secs
Test INFR Not Caller ... Pass          1 succ    0 fail    0.1 secs
Test INFR Caller Forward ... Pass          1 succ    0 fail    0.0 secs
Test INCF Forward ... Pass          1 succ    0 fail    0.0 secs
Test INCF No Forward ... Pass          1 succ    0 fail    0.0 secs
Test CFBL Busy Forward ... Pass          1 succ    0 fail    0.1 secs
Test CFBL No Forward ... Pass          1 succ    0 fail    0.1 secs
Test CFBL Free ... Pass          1 succ    0 fail    0.0 secs
Test TCS Reject ... Pass          1 succ    0 fail    0.0 secs
Test TCS No Reject ... Pass          1 succ    0 fail    0.1 secs
Test TCS No Screen ... Pass          1 succ    0 fail    0.0 secs

```

15. IVR (Interactive Voice Response)

Interactive Voice Response supports a speech interface that allows the user to make spoken enquiries and receive spoken responses. IVR services are normally referred to as applications. Although IVR has the concept of sub-dialogue, it does not have the telephony concept of feature.

15.1 Configuration Diagram

The configuration diagram starts with a **Deploys** line (see section 4). No further configuration lines are required.

15.2 Services/Features

Some IVR features are general-purpose and could be used with many services, while others are service-specific.

Feature	Purpose
ACCOUNT	bank account number (general-purpose feature, used with BOOKING and DONATION)
BOOKING	hotel booking (root)
CONFIRM	confirm submission (general-purpose feature, used with BOOKING, DONATION and ORDER)
CONTACT	contact telephone number (feature, used with BOOKING)
CUSTOMER	customer number (feature, used with ORDER)
DEAF	no user barge-in (general-purpose feature, used with BOOKING, DONATION and ORDER)
GLUCOSE	blood sugar checking (root), based on original work by Brian O'Neill (Brain Injury Rehabilitation Trust)

Feature	Purpose
HANDWASH	hand washing (root), based on original work by Alex Mihailidis (University of Toronto)
INTRODUCTION	introduction (general-purpose feature, used with BOOKING, DONATION and ORDER)
LIMB	limb donning (root), based on original work by Alex Gillespie (University of Stirling) and Brian O'Neill (Brain Injury Rehabilitation Trust)
PIN	personal identification number (general-purpose feature, used with BOOKING and ORDER)
RESTART	re-start (feature, used with DONATION)
SMOOTHIE	making a strawberry smoothie (root), based on original work by Alex Gillespie (University of Stirling)
WAIT	no prompt time-out (general-purpose feature, used with BOOKING, DONATION and ORDER)

15.3 Signals

Signal	Parameters
Application to Recogniser	
Menu (1)	String,String,[DTMF]
Option	Variable,String,String,[Boolean]
Query	String
Request	Variable,String,Grammar,[Boolean]
Application to Server	
Data	URI,String
Application to User	
Audio	String
Recogniser to Application	
Failed	Event
Failed	Value
Recogniser to User	
Audio	String
User to Recogniser	
Event	Event
Tone	DTMF
Voice	Speech

1. Not yet (fully) implemented

15.4 Actions

Action	Parameters
--------	------------

Action	Parameters
Clear	String
Data (1)	URI,String
Exit	-
Prompt	String,[Boolean]
Reprompt	-
Retry	-
Throw	String

1. Not yet (fully) implemented as VoiceXML support is limited. The Cress interpretation of this is as one-way to the server, whereas VoiceXML expects one-way from or two-way to/from the server.

15.5 Events

Event	Parameters
Cancel	[Count,[Boolean]]
Catch	String,[Count,[Boolean]]
Error	[Count,[Boolean]]
Exit	[Count,[Boolean]]
Filled	-
Help	[Count,[Boolean]]
NoInput	[Count,[Boolean]]
NoMatch	[Count,[Boolean]]

15.6 Variable Interpolation

A variable \$Variable in text is replaced by its value except in the following cases:

Variable	Meaning
\$Class(text)	string equivalent of <i>text</i>
\$Emp(text)	string equivalent of <i>text</i>
\$Enumerate	list of current options
\$Sub(text1,text2)	string equivalent of <i>text2</i>

15.7 Platform Variables

Variable	Parameters
bargin	Boolean (default false)
timeout	Integer (default 0)
vxevent	Event
vxfield	Integer (initially 0)
vxoptions	Strings (initially " " ")

Variable	Parameters
vxprompt	Strings (initially "")

15.8 Variable Types

Variable names of the form ‘query’ followed by digits are reserved (being automatically generated by the **Query** activity).

Variable
Value

15.9 Specification Validation

Lotos Validation: performed using Mustard on a 2.67GHz i920 processor, 4Gb memory, Windows XP, CygWin 1.7.1-1, Lola 3.7.2 and Mustard 1.8.

Validation of *BOOKING* and its features using the generated Lotos specification:

Test BOOKING Normal ...	Pass	1 succ	0 fail	1.7 secs
Test BOOKING Errors ...	Pass	1 succ	0 fail	0.6 secs
Test INTRODUCTION Exit ...	Pass	1 succ	0 fail	0.5 secs
Test INTRODUCTION Incorrect ...	Pass	1 succ	0 fail	0.6 secs
Test INTRODUCTION Retry Limit ...	Pass	1 succ	0 fail	0.5 secs
Test ACCOUNT Correct ...	Pass	1 succ	0 fail	0.6 secs
Test ACCOUNT Incorrect ...	Pass	1 succ	0 fail	0.6 secs
Test CONTACT Correct ...	Pass	1 succ	0 fail	0.6 secs
Test CONTACT Incorrect ...	Pass	1 succ	0 fail	0.6 secs
Test CONFIRM Correct ...	Pass	1 succ	0 fail	0.6 secs
Test CONFIRM Incorrect ...	Pass	1 succ	0 fail	0.6 secs
Test CONFIRM Retry ...	Pass	1 succ	0 fail	0.6 secs

Validation of *DONATION* and its features using the generated Lotos specification:

Test DONATION Normal ...	Pass	1 succ	0 fail	1.6 secs
Test DONATION Errors ...	Pass	1 succ	0 fail	0.6 secs
Test INTRODUCTION Exit ...	Pass	1 succ	0 fail	0.6 secs
Test INTRODUCTION Incorrect ...	Pass	1 succ	0 fail	0.5 secs
Test INTRODUCTION Retry Limit ...	Pass	1 succ	0 fail	0.5 secs
Test ACCOUNT Correct ...	Pass	1 succ	0 fail	0.5 secs
Test ACCOUNT Incorrect ...	Pass	1 succ	0 fail	0.5 secs
Test PIN Correct ...	Pass	1 succ	0 fail	0.5 secs
Test PIN Incorrect ...	Pass	1 succ	0 fail	0.6 secs
Test CONFIRM Correct ...	Pass	1 succ	0 fail	0.5 secs
Test CONFIRM Incorrect ...	Pass	1 succ	0 fail	0.5 secs
Test CONFIRM Retry ...	Pass	1 succ	0 fail	0.6 secs

Validation of *GLUCOSE* using the generated Lotos specification:

Test GLUCOSE No Problems ...	Pass	1 succ	0 fail	6.0 secs
Test GLUCOSE Help Needed ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE Find Lance ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE Find Glucose Tester ...	Pass	1 succ	0 fail	2.6 secs
Test GLUCOSE Find Strips ...	Pass	1 succ	0 fail	2.5 secs
Test GLUCOSE Find Wipe ...	Pass	1 succ	0 fail	2.6 secs
Test GLUCOSE Find Sharps Box ...	Pass	1 succ	0 fail	2.5 secs
Test GLUCOSE Hands Not Wiped ...	Pass	1 succ	0 fail	2.5 secs
Test GLUCOSE Strip Not Found ...	Pass	1 succ	0 fail	2.5 secs
Test GLUCOSE Strip Not Inserted ...	Pass	1 succ	0 fail	2.5 secs
Test GLUCOSE No Strip Number ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE No Number Match ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE No Lance Click ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE No Yellow Line ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE No Button Push ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE No Blood ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE No Blood Sample ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE Finger Bleeding ...	Pass	1 succ	0 fail	2.4 secs

Test GLUCOSE High Glucose ...	Pass	1 succ	0 fail	2.5 secs
Test GLUCOSE Medium Glucose ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE Low Glucose Unfixed ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE Low Glucose Fixed ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE Lance Not Out ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE No Lance Disposal ...	Pass	1 succ	0 fail	2.4 secs
Test GLUCOSE New Lance Fitted ...	Pass	1 succ	0 fail	2.5 secs
Test GLUCOSE Old Strip Removal ...	Pass	1 succ	0 fail	2.5 secs
Test GLUCOSE Old Strip Disposal ...	Pass	1 succ	0 fail	2.4 secs

Validation of *HANDWASH* using the generated Lotos specification:

Test HANDWASH No Problems ...	Pass	1 succ	0 fail	1.7 secs
Test HANDWASH Help Needed ...	Pass	1 succ	0 fail	0.6 secs
Test HANDWASH Turn On Water ...	Pass	1 succ	0 fail	0.6 secs
Test HANDWASH Use Soap ...	Pass	1 succ	0 fail	0.6 secs
Test HANDWASH Do Rinse ...	Pass	1 succ	0 fail	0.6 secs
Test HANDWASH Turn Off Water ...	Pass	1 succ	0 fail	0.6 secs
Test HANDWASH Dry Hands ...	Pass	1 succ	0 fail	0.6 secs

Validation of *LIMB* using the generated Lotos specification:

Test LIMB No Problems ...	Pass	1 succ	0 fail	4.8 secs
Test LIMB Help Needed ...	Pass	1 succ	0 fail	1.8 secs
Test LIMB Find Limb ...	Pass	1 succ	0 fail	1.8 secs
Test LIMB Find Liner ...	Pass	1 succ	0 fail	2.0 secs
Test LIMB Find Socks ...	Pass	1 succ	0 fail	2.0 secs
Test LIMB Apply Brakes ...	Pass	1 succ	0 fail	2.0 secs
Test LIMB Remove Boards ...	Pass	1 succ	0 fail	2.0 secs
Test LIMB Remove Footplates ...	Pass	1 succ	0 fail	2.4 secs
Test LIMB Remove Obstacle ...	Pass	1 succ	0 fail	1.9 secs
Test LIMB Remove Shrinker Sock ...	Pass	1 succ	0 fail	2.3 secs
Test LIMB Wear Thick Sock ...	Pass	1 succ	0 fail	1.9 secs
Test LIMB Smooth Thick Sock ...	Pass	1 succ	0 fail	1.9 secs
Test LIMB Wear Thin Sock ...	Pass	1 succ	0 fail	1.9 secs
Test LIMB Smooth Thin Sock ...	Pass	1 succ	0 fail	1.9 secs
Test LIMB Blocked Liner ...	Pass	1 succ	0 fail	1.9 secs
Test LIMB Tight Liner ...	Pass	1 succ	0 fail	2.5 secs
Test LIMB Reversed Liner ...	Pass	1 succ	0 fail	1.9 secs
Test LIMB Old Liner ...	Pass	1 succ	0 fail	2.1 secs
Test LIMB Pull Sock Up ...	Pass	1 succ	0 fail	2.3 secs
Test LIMB Wear Extra Sock ...	Pass	1 succ	0 fail	2.0 secs
Test LIMB Put Leg On ...	Pass	1 succ	0 fail	1.9 secs
Test LIMB Check Fit ...	Pass	1 succ	0 fail	1.9 secs
Test LIMB Pull Up Sleeve ...	Pass	1 succ	0 fail	1.9 secs
Test LIMB All Problems ...	Pass	1 succ	0 fail	2.8 secs

Validation of *ORDER* and its features using the generated Lotos specification:

Test ORDER Normal ...	Pass	1 succ	0 fail	1.8 secs
Test ORDER Errors ...	Pass	1 succ	0 fail	0.7 secs
Test INTRODUCTION Exit ...	Pass	1 succ	0 fail	0.8 secs
Test INTRODUCTION Incorrect ...	Pass	1 succ	0 fail	0.8 secs
Test INTRODUCTION Retry Limit ...	Pass	1 succ	0 fail	0.6 secs
Test CUSTOMER Correct ...	Pass	1 succ	0 fail	0.7 secs
Test CUSTOMER Incorrect ...	Pass	1 succ	0 fail	0.7 secs
Test PIN Correct ...	Pass	1 succ	0 fail	0.5 secs
Test PIN Incorrect ...	Pass	1 succ	0 fail	0.6 secs
Test CONFIRM Correct ...	Pass	1 succ	0 fail	0.5 secs
Test CONFIRM Incorrect ...	Pass	1 succ	0 fail	0.5 secs
Test CONFIRM Retry ...	Pass	1 succ	0 fail	0.5 secs

Validation of *SMOOTHIE* using the generated Lotos specification:

Test SMOOTHIE No Problems ...	Pass	1 succ	0 fail	4.5 secs
Test SMOOTHIE Help Needed ...	Pass	1 succ	0 fail	1.7 secs
Test SMOOTHIE All Problems ...	Pass	1 succ	0 fail	1.8 secs

SDL Validation: performed using Mustard on a 3.8GHz/1Gb Pentium with Windows XP, CygWin 1.5.24-2, Mustard 1.3 and Tau 4.6.

Validation of *BOOKING* and its features using the generated SDL specification:

Test BOOKING Normal ...	Pass	1 succ	0 fail	0.0 secs
Test BOOKING Errors ...	Pass	1 succ	0 fail	0.1 secs
Test INTRODUCTION Exit ...	Pass	1 succ	0 fail	0.0 secs
Test INTRODUCTION Incorrect ...	Pass	1 succ	0 fail	0.0 secs
Test INTRODUCTION Retry Limit ...	Pass	1 succ	0 fail	0.0 secs
Test ACCOUNT Correct ...	Pass	1 succ	0 fail	0.0 secs
Test ACCOUNT Incorrect ...	Pass	3 succ	0 fail	0.0 secs
Test CONTACT Correct ...	Pass	1 succ	0 fail	0.1 secs
Test CONTACT Incorrect ...	Pass	3 succ	0 fail	0.1 secs
Test CONFIRM Correct ...	Pass	1 succ	0 fail	0.1 secs
Test CONFIRM Incorrect ...	Pass	3 succ	0 fail	0.1 secs
Test CONFIRM Retry ...	Pass	1 succ	0 fail	0.1 secs

Validation of *DONATION* and its features using the generated SDL specification:

Test DONATION Normal ...	Pass	1 succ	0 fail	0.1 secs
Test DONATION Errors ...	Pass	1 succ	0 fail	0.0 secs
Test INTRODUCTION Exit ...	Pass	1 succ	0 fail	0.1 secs
Test INTRODUCTION Incorrect ...	Pass	1 succ	0 fail	0.1 secs
Test INTRODUCTION Retry Limit ...	Pass	1 succ	0 fail	0.1 secs
Test ACCOUNT Correct ...	Pass	1 succ	0 fail	0.1 secs
Test ACCOUNT Incorrect ...	Pass	3 succ	0 fail	0.1 secs
Test PIN Correct ...	Pass	1 succ	0 fail	0.1 secs
Test PIN Incorrect ...	Pass	3 succ	0 fail	0.1 secs
Test CONFIRM Correct ...	Pass	1 succ	0 fail	0.1 secs
Test CONFIRM Incorrect ...	Pass	3 succ	0 fail	0.1 secs
Test CONFIRM Retry ...	Pass	1 succ	0 fail	0.1 secs

Validation of *ORDER* and its features using the generated SDL specification:

Test ORDER Normal ...	Pass	1 succ	0 fail	0.0 secs
Test ORDER Errors ...	Pass	1 succ	0 fail	0.0 secs
Test INTRODUCTION Exit ...	Pass	1 succ	0 fail	0.1 secs
Test INTRODUCTION Incorrect ...	Pass	1 succ	0 fail	0.1 secs
Test INTRODUCTION Retry Limit ...	Pass	1 succ	0 fail	0.0 secs
Test CUSTOMER Correct ...	Pass	1 succ	0 fail	0.0 secs
Test CUSTOMER Incorrect ...	Pass	3 succ	0 fail	0.0 secs
Test PIN Correct ...	Pass	1 succ	0 fail	0.0 secs
Test PIN Incorrect ...	Pass	3 succ	0 fail	0.0 secs
Test CONFIRM Correct ...	Pass	1 succ	0 fail	0.1 secs
Test CONFIRM Incorrect ...	Pass	3 succ	0 fail	0.1 secs
Test CONFIRM Retry ...	Pass	1 succ	0 fail	0.0 secs

15.10 Specification Verification

Lotos Verification: performed using CADP 2009-c on a 2.67GHz i920 processor, 4Gb memory, Windows XP, CygWin 1.7.7-1 and Clove 1.3.

Verification of *GLUCOSE* using the annotated Lotos specification and options '-d -x -r taucomp':

Generating properties for GLUCOSE ...	CPU Time (Real Time)
Generating graph for GLUCOSE ...	23.7 secs (4.5 mins)
(states/transitions/labels: 2226/12417/102 -> 141/712/102)	
Verifying GLUCOSE Can Finish ...	Success 6.4 secs (7.0 secs)
Verifying GLUCOSE May Not Finish ...	Success 6.5 secs (7.0 secs)
Verifying GLUCOSE Finish Or Help ...	Success 6.3 secs (7.0 secs)
Verifying GLUCOSE Numbers Agree ...	Success 6.3 secs (14.0 secs)
Verifying GLUCOSE Livelock Freedom ...	Success 6.4 secs (9.0 secs)
Verifying GLUCOSE Initials Safety ...	Success 6.3 secs (7.0 secs)
Verifying GLUCOSE Always Exit ...	Success 6.4 secs (8.0 secs)

Verification of *HANDWASH* using the annotated Lotos specification and options '-d -x -r taucomp':

Generating properties for HANDWASH ...	CPU Time (Real Time)
Generating graph for HANDWASH ...	22.7 secs (2.3 mins)

```

(states/transitions/labels: 753/4210/45 -> 48/229/45)
Verifying HANDWASH Can Finish ... Success 6.3 secs ( 7.0 secs)
Verifying HANDWASH May Not Finish ... Success 6.3 secs ( 7.0 secs)
Verifying HANDWASH Finish Or Help ... Success 6.3 secs ( 7.0 secs)
Verifying HANDWASH Hands Rinsed ... Success 6.4 secs (12.0 secs)
Verifying HANDWASH Livelock Freedom ... Success 6.3 secs ( 9.0 secs)
Verifying HANDWASH Initials Safety ... Success 6.4 secs ( 7.0 secs)
Verifying HANDWASH Always Exit ... Success 6.3 secs ( 7.0 secs)

```

Verification of *LIMB* using the annotated Lotos specification and options ‘-d -x -r taucomp’:

```

Generating properties for LIMB ... CPU Time (Real Time)
Generating graph for LIMB ... 24.5 secs (10.0 mins)
(states/transitions/labels: 4595/26252/106 -> 176/1111/106)
Verifying LIMB Can Finish ... Success 6.5 secs ( 7.0 secs)
Verifying LIMB May Not Finish ... Success 6.6 secs ( 7.0 secs)
Verifying LIMB Finish Or Help ... Success 6.4 secs ( 7.0 secs)
Verifying LIMB Boards Removed ... Success 6.3 secs (13.0 secs)
Verifying LIMB Livelock Freedom ... Success 6.4 secs (10.0 secs)
Verifying LIMB Initials Safety ... Success 6.6 secs ( 7.0 secs)
Verifying LIMB Always Exit ... Success 6.4 secs ( 7.0 secs)

```

Verification of *SMOOTHIE* using the annotated Lotos specification and options ‘-d -x -r taucomp’:

```

Generating properties for SMOOTHIE ... CPU Time (Real Time)
Generating graph for SMOOTHIE ... 26.2 secs (13.6 mins)
(states/transitions/labels: 6853/39066/173 -> 286/1819/173)
Verifying SMOOTHIE Can Finish ... Success 6.4 secs ( 7.0 secs)
Verifying SMOOTHIE May Not Finish ... Success 6.2 secs ( 7.0 secs)
Verifying SMOOTHIE Finish Or Help ... Success 6.4 secs ( 8.0 secs)
Verifying SMOOTHIE Contents Blended ... Success 6.4 secs (19.0 secs)
Verifying SMOOTHIE Livelock Freedom ... Success 6.5 secs (10.0 secs)
Verifying SMOOTHIE Initials Safety ... Success 6.4 secs ( 7.0 secs)
Verifying SMOOTHIE Always Exit ... Success 6.5 secs ( 7.0 secs)

```

16. SIP (Session Initiation Protocol)

The Session Initiation Protocol is typically used to support Internet Telephony, though it is really a general-purpose signalling protocol. SIP features are typically implemented in a user agent, proxy server or redirect server using CPL (Call Processing Language) or CGI-BIN (Common Gateway Interface – Binary).

16.1 Configuration Diagram

The configuration diagram starts with a **Deploys** line and then a blank line (see section 4).

Feature configuration lines are then per subscriber and have the same form as for IN (see section 14.1).

16.2 Services/Features

Only a limited range of features is currently defined, patterned after those used with Intelligent Networks. Features vary according to where they are deployed.

Feature	Purpose
AGENT	User Agent (root)
AGENT CFBL	User Agent Call Forward Busy Line (feature)
AGENT TCS	User Agent Terminating Call Screening (feature)
PROXY	Proxy Server (root)
PROXY CFBL	Proxy Server Call Forward Busy Line (feature)

PROXY TCS	Proxy Server Terminating Call Screening (feature)
REDIRECT	Redirect Server (root)

16.3 Signals

Signal	Parameters
Agent to Billing	
AirBegin	Address,Time
AirEnd	Address,Time
LogBegin	Address,Address,Address,Time
LogEnd	Address,Address,Time
Agent to Protocol, Protocol to Agent	
Ack	Address,Address
Bye	Address,Address
Invite	Address,Address
Response	Address,Address,Response
Agent to Status Manager	
StartBilling	Address,Address,Address
StopBilling	Address,Address,Address
UpdateAA (1)	Status,Address,Address
UpdateAB (1)	Status,Address,Boolean
UpdateAL (1)	Status,Address,List
UpdateAT (1)	Status,Address,Time
UpdateAAA (1)	Status,Address,Address,Address
UpdateAAB (1)	Status,Address,Address,Boolean
UpdateAAT (1)	Status,Address,Address,Time
Agent to User	
Announce	Address,Message
Disconnect	Address,Address
StartRing	Address,Address
StopRing	Address,Address
User to Agent	
Answer	Address
Dial	Address,Address
OffHook	Address
OnHook	Address
Reject	Address,Message

1. Used in the SDL translation

16.4 Static (Profile) Variables

Variable	Parameters	Result
BillPIN (1)	Address	Boolean
CallingNumber (4)	Address	Boolean
CallWaiting	Address	Boolean
Cellular	Address	Boolean
ForwardBusy (2)	Address	Address
ForwardTo	Address	Address
Freephone (4)	Address	Boolean
MovedTo (4)	Address	Boolean
RedirectAddress	Address,Address	Address
RedirectTime1	Address,Address	Time
RedirectTime2	Address,Address	Time
ScreenIn	Address	Addresses
ScreenOut (3)	Address	Addresses
TeenPIN	Address	Address
TeenTime1	Address	Time
TeenTime2	Address	Time

1. 'Charge' in Chisel
2. 'BLForward' in Chisel
3. 'Screened' in Chisel
4. Not defined in Chisel

16.5 Dynamic (Run-Time) Variables

Variable	Parameters	Result
Bill (1)	Address,Address	Address
Busy	Address	Boolean
By (1)	Address,Address	Address
LastIncoming	Address	Address
ReturnCall	Address,Address	Boolean

1. Not defined in Chisel.

16.6 Announcement Messages

Message
AnyMessage
AskForPIN (1)
BusyHere

Message
Decline
DialTone
EnterPhoneNumber
EnterPIN
InvalidPIN
ScreenedMessage
Success
Terminated
Trying
Unobtainable

1. Accepted as in Chisel, but converted to ‘EnterPIN’

The following functions apply to announcement messages (i.e. response codes):

Function	Parameters	Result
Moved	Message	Boolean
MovedTo	Message	Address
Terminal	Message	Boolean

16.7 Variable Types

Variable
Address
Message
Time

16.8 Specification Validation

Lotus Validation: performed using Mustard on a 2.67GHz i920 processor, 4Gb memory, Windows XP, CygWin 1.7.1-1, Lola 3.7.2 and Mustard 1.8.

Validation of *AGENT* and its features using the generated Lotus specification:

Test AGENT CFBL Busy Forward ...	Pass	222 succ	0 fail	12.4 secs
Test AGENT CFBL No Forward ...	Pass	244 succ	0 fail	3.0 secs
Test AGENT CFBL Free ...	Pass	57 succ	0 fail	1.5 secs
Test AGENT TCS Reject ...	Pass	57 succ	0 fail	1.1 secs
Test AGENT TCS No Reject ...	Pass	57 succ	0 fail	1.4 secs
Test AGENT TCS No Screen ...	Pass	24 succ	0 fail	1.2 secs

Validation of *PROXY* and its features using the generated Lotus specification:

Test PROXY Clear Before Dial ...	Pass	12 succ	0 fail	1.5 secs
Test PROXY Clear Before Answer ...	Pass	24 succ	0 fail	1.0 secs
Test PROXY Caller Clear ...	Pass	23 succ	0 fail	1.4 secs
Test PROXY Callee Clear ...	Pass	24 succ	0 fail	1.3 secs
Test PROXY Call Self ...	Pass	57 succ	0 fail	0.8 secs
Test PROXY Simultaneous Call ...	Pass	248 succ	0 fail	1.7 secs
Test PROXY CFBL Busy Forward ...	Pass	42 succ	0 fail	4.4 secs
Test PROXY CFBL No Forward ...	Pass	249 succ	0 fail	1.6 secs

Test PROXY CFBL Free ...	Pass	57 succ	0 fail	1.3 secs
Test PROXY TCS Reject ...	Pass	4 succ	0 fail	0.4 secs
Test PROXY TCS No Reject ...	Pass	24 succ	0 fail	1.4 secs
Test PROXY TCS No Screen ...	Pass	24 succ	0 fail	1.0 secs

Validation of *REDIRECT* and its features using the generated Lotos specification:

Test REDIRECT Forward ...	Pass	24 succ	0 fail	1.7 secs
Test REDIRECT Forward Busy ...	Pass	57 succ	0 fail	0.7 secs

SDL Validation: performed using Mustard on a 3.8GHz/1Gb Pentium with Windows XP, CygWin 1.5.24-2, Mustard 1.3 and Tau 4.6.

Validation of *AGENT* and its features using the generated SDL specification:

Test AGENT CFBL Busy Forward ...	Pass	1 succ	0 fail	0.0 secs
Test AGENT CFBL No Forward ...	Pass	1 succ	0 fail	0.1 secs
Test AGENT CFBL Free ...	Pass	1 succ	0 fail	0.0 secs
Test AGENT TCS Reject ...	Pass	1 succ	0 fail	0.1 secs
Test AGENT TCS No Reject ...	Pass	1 succ	0 fail	0.1 secs
Test AGENT TCS No Screen ...	Pass	1 succ	0 fail	0.0 secs

Validation of *PROXY* and its features using the generated SDL specification:

Test PROXY Clear Before Dial ...	Pass	1 succ	0 fail	0.0 secs
Test PROXY Clear Before Answer ...	Pass	1 succ	0 fail	0.1 secs
Test PROXY Caller Clear ...	Pass	1 succ	0 fail	0.0 secs
Test PROXY Callee Clear ...	Pass	1 succ	0 fail	0.1 secs
Test PROXY Call Self ...	Pass	1 succ	0 fail	0.1 secs
Test PROXY Simultaneous Call ...	Pass	1 succ	0 fail	0.0 secs
Test PROXY CFBL Busy Forward ...	Pass	1 succ	0 fail	0.0 secs
Test PROXY CFBL No Forward ...	Pass	1 succ	0 fail	0.1 secs
Test PROXY CFBL Free ...	Pass	1 succ	0 fail	0.0 secs
Test PROXY TCS Reject ...	Pass	1 succ	0 fail	0.0 secs
Test PROXY TCS No Reject ...	Pass	1 succ	0 fail	0.0 secs
Test PROXY TCS No Screen ...	Pass	1 succ	0 fail	0.1 secs

Validation of *REDIRECT* and its features using the generated SDL specification:

Test REDIRECT Forward ...	Pass	1 succ	0 fail	0.1 secs
Test REDIRECT Forward Busy ...	Pass	1 succ	0 fail	0.0 secs

17. VoIP (Voice over Internet Protocol)

Voice over Internet Protocol refers to use of CPL, to distinguish it from the SIP domain. This allows a user to define a single call-handling service that is executed by a SIP server when calls are made or received. VoIP support is due to Dean McMenemy (University of Stirling). As this was a student project, the code is not as thoroughly tested or complete as the other applications domains developed by the author.

17.1 Configuration Diagram

The configuration diagram starts with a **Deploys** line (see section 4). No further configuration lines are required.

17.2 Services/Features

A SIP user is allowed only one CPL script, so only one service at a time is deployed. There are currently no features defined for VoIP. The following services are based on the examples given in the CPL standard (RFC 3880).

Service	Purpose
AGENT_CHECK	presuming the "DodgySIP" package is not compatible with the "telephony.com" network, a call to Alice on this network is removed

	from the list of destinations.
FORWARD_ALWAYS	calls are always forwarded to a friend
FORWARD_LOOKUP	forward to the location returned by registration lookup
FORWARD_SUBJECT	calls about academic discounts are rejected, those about sales are forwarded to the sales address, and all others are sent to an information address
FORWARD_VOICEMAIL	forward to voicemail on busy or no answer
PRIORITY_LANGUAGE	non-urgent calls are forwarded to a Spanish or English speaking operator as appropriate
REJECT_BUSY	reject calls from Bob as being busy
REJECT_OFFLINE	while the system is offline, mail the administrator of outgoing call attempts and inform the user
REJECT_SUNDAY	reject calls on Sunday from 1st Jan 2007
SCREEN_PREMIUM	reject premium rate calls to 0870 numbers

17.3 Actions

Action	Parameters
Location	<ul style="list-style-type: none"> lookup [source] URL, followed by guard expressions of the form busy default failure noanswer notfound not-present otherwise redirection success remove URL, followed by guard expressions of the form busy default failure noanswer notfound not-present otherwise redirection success URL, followed by guard expressions of the form busy default failure noanswer notfound not-present otherwise redirection success
Log	name comment
Mail	email subject body
Proxy	<ul style="list-style-type: none"> nothing ordering first-only parallel sequential recurse no yes timeout
Redirect	<ul style="list-style-type: none"> nothing URL
Reject	status reason
Switch	<ul style="list-style-type: none"> address destination origin original-destination address-type display host port tel user, followed by guard expressions of the form = < ~ <i>address</i> or else language, followed by guard expressions of the form = < ~ <i>language</i> or else priority = eq equal > gt great < lt less priority, followed by guard expressions such as else string display organization subject user-agent, followed by guard expressions of the form = < ~ <i>string</i> or else time zone [URL], followed by guard expressions of the form byday byhour byminute bymonth bymonthday bysecond bysetpos byweekno byyear count dtstart dtend duration freq interval time until wkst = <i>value</i>

17.4 Guards

Guard	Parameters
=	address, language, string
<	address, language, string
~	address, language, string
busy, default, failure, noanswer, notfound, not-present, otherwise, redirection, success	-
byday, byhour, byminute, bymonth, bymonthday, bysecond, bysetpos, byweekno, byyear, count, dtstart, dtend, duration, freq, interval, time, until, wkst	time
else	-

18. WS (Web Services)

Web services allow application-to-application communication using web-like mechanisms. WS applications usually do not make use of features.

18.1 Configuration Diagram

The configuration diagram starts with a **Deploys** line and then a blank line (see section 4).

Service configuration lines then have the form:

```
partner_name ns_prefix ns_URI service_URL
```

for example:

```
APPROVER app urn:FirstRate localhost:8080/active-bpel
```

This gives the partner name, XML namespace prefix, XML namespace URI (normally a URN), and the base URL where the service is deployed. In conformity to Axis conventions, *services* is automatically added to the last of these.

18.2 Services/Features

Some WS features are general-purpose and could be used with many services, while others are service-specific.

Feature	Purpose
ADDRESS	address validation (general-purpose feature, not fully supported currently)
BROKER	car broker (root, making use of BROKER and SUPPLIER)
LENDER	loan arranger (root, used by BROKER)
NORMALISE	name normalisation (general-purpose feature, not fully supported currently)
SUPPLIER	car supplier (root, used by BROKER)

18.3 Signals

Signal	Parameters
--------	------------

Signal	Parameters
Partner to Service	
Reply	QName,[Identifier Fault]
Service to Partner	
Invoke	QName,Identifier,[Identifier,Fault*]
Service to User	
Reply	QName,[Identifier Fault]
User to Service	
Receive	QName,[Identifier]

18.4 Actions

Action	Parameters
Compensate	[Identifier]
Empty	-
Fork	[loose strict]
Join	[Expression]
Terminate	-
Throw	QName
Wait	Date [Time] Time
While	Expression

18.5 Events

Event	Parameters
Catch	QName
CatchAll	-
Compensate	[Identifier]
Correlation (1)	[Identifier]
Timeout	Date [Time] Time

1. Not yet (fully) implemented

18.6 Dynamic (Run-Time) Variables

When translating to BPEL, note that the following variables require use of ActiveBPEL and BPEL version 2. When translating to Lotos, note that the following variables are only abstractions of real time.

Variable	Result
date	String (YYYY-MM-DD)
day	Integer

hour	Integer
month	Integer
second	Integer
time	String (HH:MM:SS)
weekday	Integer (0 = Monday)
year	Integer

18.7 Variable Types

Note that Natural cannot be mixed with other number types in the same expression. Since a literal number is translated as a floating point number, this also means that a Natural cannot be mixed with a literal (e.g. $n + 1$).

The index of an array may be any kind of number. The whole part of a floating point number is used when the index is not a Natural. Array accesses may not be nested (e.g. `a[b[2]]`).

Variable
Boolean
Byte
Date
DateTime
Decimal
Double
Float
Int
Integer
Integer
Long
Natural
NegativeInteger
NonNegativeInteger
NonPositiveInteger
PositiveInteger
Short
String
Time
UnsignedByte
UnsignedInt
UnsignedInt
UnsignedLong
UnsignedShort

18.8 Specification Validation

Lotos Validation: performed using Mustard on a 2.67GHz i920 processor, 4Gb memory, Windows XP, CygWin 1.7.1-1, Lola 3.7.2 and Mustard 1.8.

Validation of *BROKER* using the generated Lotos specification:

Test BROKER Mondeo Ken ...	Pass	1 succ	0 fail	1.6 secs
Test BROKER A5 Scotland ...	Pass	1 succ	0 fail	2.4 secs
Test BROKER Megane ...	Pass	1 succ	0 fail	1.7 secs
Test BROKER Astra ...	Pass	2 succ	0 fail	5.8 secs
Test BROKER XJ6 ...	Pass	1 succ	0 fail	0.5 secs

Validation of *LENDER* and its partners using the generated Lotos specification:

Test APPROVER Low Rate ...	Pass	1 succ	0 fail	1.3 secs
Test APPROVER Medium Rate ...	Pass	1 succ	0 fail	0.9 secs
Test APPROVER High Rate ...	Pass	1 succ	0 fail	4.4 secs
Test APPROVER Loan Unacceptable ...	Pass	1 succ	0 fail	1.3 secs
Test ASSESSOR Low Risk ...	Pass	1 succ	0 fail	0.4 secs
Test ASSESSOR Medium Risk ...	Pass	1 succ	0 fail	0.4 secs
Test ASSESSOR High Risk ...	Pass	1 succ	0 fail	0.4 secs
Test LENDER Little Low Risk ...	Pass	1 succ	0 fail	0.4 secs
Test LENDER Little Medium Risk ...	Pass	1 succ	0 fail	0.4 secs
Test LENDER Little High Risk ...	Pass	1 succ	0 fail	0.5 secs
Test LENDER Lots Ken ...	Pass	1 succ	0 fail	0.4 secs
Test LENDER Lots Scotland ...	Pass	1 succ	0 fail	0.7 secs
Test LENDER Lots Under 15000 ...	Pass	1 succ	0 fail	0.6 secs
Test LENDER Lots Exceeds 15000 ...	Pass	1 succ	0 fail	1.3 secs

Validation of *SUPPLIER* and its partners using the generated Lotos specification:

Test DEALER1 Mondeo ...	Pass	1 succ	0 fail	1.3 secs
Test DEALER1 A5 ...	Pass	1 succ	0 fail	0.4 secs
Test DEALER1 Megane ...	Pass	1 succ	0 fail	0.4 secs
Test DEALER1 XJ6 ...	Pass	1 succ	0 fail	0.4 secs
Test DEALER2 Mondeo ...	Pass	1 succ	0 fail	0.4 secs
Test DEALER2 A5 ...	Pass	1 succ	0 fail	0.4 secs
Test DEALER2 Astra ...	Pass	1 succ	0 fail	0.4 secs
Test DEALER2 XJ6 ...	Pass	1 succ	0 fail	0.5 secs
Test SUPPLIER Mondeo ...	Pass	1 succ	0 fail	0.5 secs
Test SUPPLIER A5 ...	Pass	1 succ	0 fail	0.5 secs
Test SUPPLIER Megane ...	Pass	1 succ	0 fail	0.5 secs
Test SUPPLIER Astra ...	Pass	1 succ	0 fail	0.5 secs
Test SUPPLIER XJ6 ...	Pass	1 succ	0 fail	0.5 secs

18.9 Specification Verification

Lotos Verification: performed using CADP 2009-c on a 2.67GHz i920 processor, 4Gb memory, Windows XP, CygWin 1.7.7-1 and Clove 1.3.

Verification of *BROKER* (and implicitly its partners) using the annotated Lotos specification and options '-x -r taucomp':

Generating properties for BROKER ...	CPU Time (Real Time)
Generating graph for BROKER ...	22.8 secs (2.2 mins)
(states/transitions/labels: 19789/67423/11 -> 6/10/11)	
Verifying BROKER General Arrange Response ...	Success 6.5 secs (12.0 secs)
Verifying BROKER Specific Arrange Response ...	Success 6.5 secs (12.0 secs)
Verifying BROKER Always Need Request ...	Success 6.2 secs (12.0 secs)
Verifying BROKER Deadlock Freedom ...	Success 6.5 secs (11.0 secs)
Verifying BROKER Livelock Freedom ...	Success 6.5 secs (14.0 secs)
Verifying BROKER Initials Safety ...	Success 6.3 secs (11.0 secs)
Verifying BROKER Always Exit ...	Success 6.5 secs (11.0 secs)

Verification of *LENDER* (and implicitly its partners) using the annotated Lotos specification and options '-x -r taucomp':

Generating properties for LENDER ...	CPU Time (Real Time)
Generating graph for LENDER ...	21.6 secs (1.6 mins)

```

                                (states/transitions/labels: 7425/9481/14 -> 5/13/14)
Verifying LENDER Any Loan Response ...      Success 6.3 secs (12.0 secs)
Verifying LENDER General Quote Response ...  Success 6.5 secs (12.0 secs)
Verifying LENDER Specific Quote Response ... Success 6.4 secs (12.0 secs)
Verifying LENDER Incomplete Quote Response ... Success 6.3 secs (12.0 secs)
Verifying LENDER Any Ken Low Amount Response ... Success 6.2 secs (11.0 secs)
Verifying LENDER Any Quote Refusal Response ... Success 6.4 secs (12.0 secs)
Verifying LENDER Deadlock Freedom ...      Success 6.4 secs (12.0 secs)
Verifying LENDER Livelock Freedom ...      Success 6.4 secs (13.0 secs)
Verifying LENDER Initials Safety ...       Success 6.1 secs (12.0 secs)
Verifying LENDER Always Exit ...           Success 6.1 secs (11.0 secs)

```

Verification of *SUPPLIER* (and implicitly its partners) using the annotated Lotos specification and options '-x -r taucomp':

```

Generating properties for SUPPLIER ...      CPU Time (Real Time)
Generating graph for SUPPLIER ...          22.3 secs ( 1.2 mins)
                                (states/transitions/labels: 6196/58444/59 -> 5/58/59)
Verifying SUPPLIER General Order Response ... Success 6.2 secs ( 8.0 secs)
Verifying SUPPLIER Specific Order Response ... Success 6.2 secs ( 8.0 secs)
Verifying SUPPLIER Always Need Request ... Success 6.4 secs ( 7.0 secs)
Verifying SUPPLIER Deadlock Freedom ...    Success 6.2 secs ( 7.0 secs)
Verifying SUPPLIER Livelock Freedom ...    Success 6.3 secs (10.0 secs)
Verifying SUPPLIER Initials Safety ...     Success 6.2 secs ( 7.0 secs)
Verifying SUPPLIER Always Exit ...         Success 6.2 secs ( 7.0 secs)

```

18.10 Implementation Validation

BPEL Validation: performed using Mustard on a 2.67GHz i920 processor, 4Gb memory, Windows XP, CygWin 1.7.1-1, Mint 1.2 and Mustard 1.8.

Validation of *BROKER* using the generated Bpel implementation:

```

Test BROKER Mondeo Ken ...      Pass      1 succ    0 fail    0.6 secs
Test BROKER A5 Scotland ...    Pass      1 succ    0 fail    0.6 secs
Test BROKER Megane ...         Pass      1 succ    0 fail    0.6 secs
Test BROKER Astra ...          Pass      1 succ    0 fail    0.6 secs
Test BROKER XJ6 ...            Pass      1 succ    0 fail    0.6 secs

```

Validation of *LENDER* and its partners using the generated Bpel implementation:

```

Test APPROVER Low Rate ...      Pass      1 succ    0 fail    0.5 secs
Test APPROVER Medium Rate ...   Pass      1 succ    0 fail    0.5 secs
Test APPROVER High Rate ...     Pass      1 succ    0 fail    0.5 secs
Test APPROVER Loan Unacceptable ... Pass      1 succ    0 fail    0.5 secs
Test ASSESSOR Low Risk ...      Pass      1 succ    0 fail    0.5 secs
Test ASSESSOR Medium Risk ...   Pass      1 succ    0 fail    0.5 secs
Test ASSESSOR High Risk ...     Pass      1 succ    0 fail    0.5 secs
Test LENDER Little Low Risk ... Pass      1 succ    0 fail    0.5 secs
Test LENDER Little Medium Risk ... Pass      1 succ    0 fail    0.5 secs
Test LENDER Little High Risk ... Pass      1 succ    0 fail    0.5 secs
Test LENDER Lots Ken ...        Pass      1 succ    0 fail    0.5 secs
Test LENDER Lots Scotland ...   Pass      1 succ    0 fail    0.5 secs
Test LENDER Lots Under 15000 ... Pass      1 succ    0 fail    0.5 secs
Test LENDER Lots Exceeds 15000 ... Pass      1 succ    0 fail    0.5 secs

```

Validation of *SUPPLIER* and its partners using the generated Bpel implementation:

```

Test DEALER1 Mondeo ...        Pass      1 succ    0 fail    0.5 secs
Test DEALER1 A5 ...            Pass      1 succ    0 fail    0.5 secs
Test DEALER1 Megane ...        Pass      1 succ    0 fail    0.5 secs
Test DEALER1 XJ6 ...           Pass      1 succ    0 fail    0.5 secs
Test DEALER2 Mondeo ...        Pass      1 succ    0 fail    0.5 secs
Test DEALER2 A5 ...            Pass      1 succ    0 fail    0.5 secs
Test DEALER2 Astra ...         Pass      1 succ    0 fail    0.5 secs
Test DEALER2 XJ6 ...           Pass      1 succ    0 fail    0.6 secs
Test SUPPLIER Mondeo ...       Pass      1 succ    0 fail    0.6 secs
Test SUPPLIER A5 ...           Pass      1 succ    0 fail    0.5 secs
Test SUPPLIER Megane ...       Pass      1 succ    0 fail    0.5 secs

```

Test SUPPLIER Astra ...	Pass	1 succ	0 fail	0.5 secs
Test SUPPLIER XJ6 ...	Pass	1 succ	0 fail	0.6 secs

CRESS Target Languages

19. Overview

See the [Cress home page](#) for an overview of Cress and some references. Cress (Chisel Representation Employing Systematic Specification) supports translation to the following target languages.

- BPEL/WSDL
- CPL
- Lotos
- SDL
- VoiceXML

20. Working with BPEL

To deploy and run generated device, grid or web services, you will need [Apache Tomcat](#) (e.g. version 5.5.12 onwards) and [ActiveBPEL](#) (version 5.0.0 onwards). Device service requires separate code for OSGi developed by the principal author (notably a *SoapProxy* bundle). To deploy and run grid services will also need [Globus WS Core](#) (version 4.2 onwards). To test web or grid services requires JUnit (version 4.0 onwards). Beware of some version incompatibilities in these tools:

- ActiveBPEL versions 3.*N* onwards require a JVM version 1.5.*N*

ActiveBPEL periodically checks for new (un)deployments. By default, this is every 20 seconds – perhaps too slow during development. As an example, change the file *.../Tomcat/webapps/active-bpel/WEB-INF/web.xml* as follows for scans every 5 seconds (5000 msec):

```
<init-param>
  <param-name>scan.interval</param-name>
  <param-value>5000</param-value>
</init-param>
```

ActiveBPEL provides a SOAP message monitor. If it is necessary to turn this on, change the file *.../Tomcat/shared/classes/ae-server-config.wsdd* to uncomment the following:

```
<requestFlow>
  ...
  <handler type="java:org.apache.axis.handlers.SOAPMonitorHandler"/>
</requestFlow>

<responseFlow>
  <handler type="java:org.apache.axis.handlers.SOAPMonitorHandler"/>
</responseFlow>
```

If a service assigns to fields of structured types, when using BPEL version 1.1 (but not version 2) **it is essential** to select ‘Auto create target path for Copy/To’ in the ActiveBPEL console configuration. Failure to do this will result in errors like ‘query expression did not evaluate to single node’. When using BPEL version 2 onwards, this is unnecessary as the relevant BPEL extension is enabled automatically.

ActiveBPEL schema validation may incorrectly report errors (notably with the MATCHER example). In such a case, deselect ‘Validate Input/Output messages against schema’.

A partner (non-BPEL) service is deployed by ActiveBPEL/AXIS under the port name. Suppose that partner service *approver* has ports *borrow* and *loan*. This will result in the AXIS services *ApproverBorrow* and *ApproverLoan*. These services are available at <http://localhost:8080/active-bpel/services>. The first part of this is determined by the Cress configuration file. The *services* part is a configuration parameter to ActiveBPEL (*.../Tomcat/webapps/active-bpel/WEB-INF/web.xml*) and to Cress (*\$serv_path* in *cress_bpel/customise*).

A BPEL service is deployed by ActiveBPEL/AXIS under the service name. For example, BPEL service *lender* results in the AXIS service *LenderService*.

Although you can run *cress_bpel* from the command line, it is better to use the BPEL framework. In the *bpel* directory, issue a command such as:

```
cress_expand -v ws main.bpel
```

(The *ds* and *gs* vocabularies can also be used.) Optionally use *-d* to deploy the files as well. Note that *main.bpel* is a fictitious file; the framework file is actually *main.base*. A simpler way to create and deploy services is with, for example:

```
cress_realise -t bpel -v ws
```

You can use *cress_deploy* to deploy, undeploy and redeploy files.

Each service and its associated partners will be created in a directory under the *bpel* directory. For example, the supplier web service and its two partner dealer services will be created under the *bpel/supplier* directory.

Since there is no unique root diagram for BPEL, you will need to adjust the configuration diagram to suit the services required. The generated files will then be particular to these services.

Partner and service files are created in the service directory. For example, the *ws/supplier* service creates *dealer1.**, *dealer2.** and *supplier.** files in the *ws/supplier* directory. When processed with *cress_expand*, these are copied and augmented in directory *bpel/supplier*. See *bpel_create* and *cress_bpel* for more details.

A file **.[bpel|bpr|wsdl|wsr]* is pre-translated code for a service.

For GS/WS, a file *<service>.extra* is treated as extra definitions to be included in *<service>_defs.wsdl*, and a file *<partner>.extra* is treated as extra definitions to be included in *<partner>.wsdl* (disregarding any *_<parent>* suffix).

When working with grid services, note that absolute paths to Globus files are generated based on the value of *GLOBUS_LOCATION*. Cress also assumes a particular directory structure for GT4 schema files. Any GS files supplied with Cress must be re-generated according to the local GT4 installation directory.

Grid services can be deployed/undeployed in GT4 only when it is not running. If a proxy credential has been created, GT4 can be started with *globus-start-container*. To avoid conflict with ActiveBPEL, GT4 should normally be run at a port other than 8080 with a command-line parameter like *'-p 8880'*. If a proxy credential has not been created, use the *'-nosec'* (no security) option on the command line. The grid service examples supplied with Cress are defined to run at <http://localhost:8880/wsrf/services>.

Deployment of grid services assumes that Globus WS Core has been deployed locally according to the environment variable *GLOBUS_LOCATION*. This has two implications:

- GAR files are made available to Globus at this location. If Globus is not installed locally, do not request deployment of grid services. Instead, the generated GAR files can be deployed manually to Globus on the relevant system.
- WS-Addressing and WS-Resource files are imported from this location. If Globus is not installed locally, replace the local file reference with URLs as follows:

```
http://www.w3.org/2006/03/addressing/ws-addr.xsd
```

```
http://docs.oasis-open.org/wsrf/rpw-2.wsdl
```

```
http://docs.oasis-open.org/wsrf/rw-2.wsdl
```

If the Tomcat container for ActiveBPEL is running via an Internet proxy, the system properties for the proxy need to be defined, e.g.

```
CATALINA_OPTS=-DproxySet=true -DproxyHost=wwwcache.stir.ac.uk -DproxyPort=8080
```

If use of *WSDL2Java* also requires a proxy, edit the *generateStubs* target under the <java> task in *\$GLOBUS_LOCATION/share/globus_wsrfl_tools/build-stubs.xml* to add the proxy settings, e.g.:

```
<sysproperty key="proxySet" value="true"/>
<sysproperty key="proxyHost" value="wwwcache.stir.ac.uk"/>
<sysproperty key="proxyPort" value="8080"/>
```

If the grid services tests supplied are to be repeated, note that the *Converter* and *Statistics* implementations require access to JUnit 4.0 and a MySQL database with the following parameters:

- host *localhost*, port *3306*, database *test*
- user *test*, password *gs-test*

For the grid service tests supplied, note that *STATISTICS* depends on *CONVERTER*. It is therefore necessary to first build *SPLITTER* (and hence *CONVERTER*). Then copy its JAR file *bpel/splitter/converter/lib/converter_splitter.jar* to *gs/analyser/statistics/lib/converter_splitter.jar*. Provided the interface to *CONVERTER* does not change, *ANALYSER* and *SPLITTER* can now be built independently.

For the grid service tests supplied, note that *COUNTER* depends on *PARSER*. It is therefore necessary to first build *MATCHER* (and hence *COUNTER*). Then copy its JAR file *bpel/matcher/parser/lib/parser.jar* to *gs/matcher/counter/lib/parser.jar*. Provided the interface to *COUNTER* does not change, *MATCHER* can now be built directly. Note that GT4 has a *counter* service too (*globus_wsrfl_core_samples_counter*). If the Cress version is to be run, the GT4 version must be undeployed first.

When working with web services, if the file <partner>.java is present in the *ws/<service>* directory, it is used as the implementation of the *partner* service.

21. Working with CPL

To deploy and run generated CPL, you will need a CPL-capable SIP server such as [SER \(SIP Express Router\)](#) or [Vocal](#).

Although you can run *cress_cpl* from the command line, it is better to use the CPL framework. In the *cpl* directory, issue the command:

```
cress_expand -v voip main.cpl
```

Optionally use *-d* to deploy the files as well. A simpler way to create and deploy services is with, for example:

```
cress_realise -k user:password@host -t cpl -v voip
```

You can also use *cress_deploy* to deploy, undeploy and redeploy files. Both of these require *-a* to define server authorisation in the form *user:password@host*, e.g. *ken:obscure@sip.cs.stir.ac.uk*.

22. Working with Lotos

To analyse and execute the generated specifications, you will need *Lola/Topo* (e.g. version 3.6 onwards). The author has a version of these compiled for CygWin on Windows. You might also use [CADP](#) (e.g. version 2009-b onwards) for verification.

Although you can run *cress_lotos* from the command line, it is better to use the *Lotos* framework. As an example, go to the *lotos* directory and issue:

```
topo in -lola -l stir
```

or (if you get hold of the author's *lotos* interface to Topo):

```
tlotos in
```

with the proviso that when first used with a new Lotos file the *-s* flag should be given for the *stir* library. A simpler way to create and deploy services is with, for example:

```
cross_realise -t lotos -v ws
```

The corresponding commands for CADP look like:

```
cadp_annotate ws
caesar.adt -error -warning ws
caesar -error -warning ws
```

Note that *caesar.adt* will issue warnings that external operations should not have equations. It is normally possible to disregard these warnings since the equations are needed for Lola/Topo (but are ignored by CADP). *caesar* may issue warnings about unreachable processes. This typically happens with event dispatcher code and can be ignored. Make sure to set the *CADP_CC* environment variable (see section 5). Also see the notes about *cadp_annotate*. Through *cadp_annotate*, Cress generates **.custom* (customised value enumerations), **.f* (functions) and **.t* (types). These are not complete but should at least allow the files to be compiled by Caesar. They are designed to be used with Clove (which completes them automatically).

The integrated event dispatcher generated for DS/GS/IVR/WS makes significant use of recursion. Although Lola issues warnings about possible recursive process instantiation, these can be disregarded. However, CADP treats this situation as an error. It is therefore possible to generate qualified event dispatchers with one fault or compensation process per scope ('-q' option). These work with Lola and often work with CADP. However, CADP can still report an error if behaviour is repeated ('-r' option) and fault or compensation handling is required in a **Fork** branch. In such a case, avoid repeated behaviour for the service where this situation applies (omit the service in the '-r' list).

For DS/GS/WS, if a fault is not explicitly caught then the default BPEL behaviour is to compensate the scope where it occurs and to rethrow the fault to the enclosing scope. Ultimately the fault is returned to the caller. It is difficult to achieve the same effect with Lotos because the correct response event is unknown, and because the fault may occur inside a **Fork**. It is probably poor design to rely on default fault handling; explicitly dealing with faults is preferable. For these reasons, if an uncaught fault occurs (including JoinFailure) then the Lotos service will deadlock. During validation or property checking, this should identify that the service is underspecified.

Cress makes use of a shared library of data types called *stir* (for Stirling). Three *stir* files are provided in the *supp/topo* directory. They should be copied to where Topo can access them, e.g. the *lotos* directory or the *topo/stdlib* installation directory.

The file *in.lot* will be built automatically from the *in.base* framework. Now you can do whatever you do with Lotos. Delete the file *in.lot* to force a re-build if you change the configuration diagram. Alternatively *touch* the file *in.base*.

A file **.lot* is a pre-translated version of the diagram, merged *on its own* with the root diagram.

When working with web services, if the file *<partner>.lot* is present in the *ws/<service>* directory, it is used as the specification of the *partner* service.

If the '-a' flag is provided to *cress_lotos*, a file with suffix **.lotos* will be created for CADP. This is automatically annotated so that *caesar* can compile it. It is not possible to make the annotation perfect. For example, externally implemented operations will receive complaints that they have equations, and application-defined sorts and operations may need to be added to the customisation in *cadp_annotate*.

Cress makes use of shared header files *Stirling*.h* that implement key data types for CADP. These are provided in the *supp/cadp* directory. They should be copied to where CADP can access them, e.g. the *cadp/incl* installation directory.

23. Working with SDL

To analyse and execute the generated specifications, you will need [Telelogic Tau SDL Suite](#) for SDL/MSC.

An appropriate Cress filter must be set up using the Generate/Analyze command. Set the Filter Command to *sdt_in*, etc. according to the domain vocabulary. (The Tau files provided with Cress are already set up in this way, though you may need to alter the setting depending on your platform or paths.)

Although you can run *cress_sdl* from the command line, it is better to use the SDL framework. Find the *sdl* directory and open the relevant *.sdt file with Tau SDT. Click on the Analyze, Simulate or Validate button to build the specification and then process it with Tau SDT. Now you can do whatever you do with SDL. Force a re-build if you change the configuration diagram. Do this by re-saving the top-level system diagram **System.ssy*.

A file *.pr is a pre-translated version of a diagram, merged *on its own* with the root diagram.

24. Working with VoiceXML

To deploy and run Cress IVR services, you will need V-Builder and associated packages from [Nuance Corporation](#) (e.g. version 1.2 onwards). This is a substantial and complex set of downloads that needs registration and approval by Nuance. At least Nuance V-Builder, Nuance Vocalizer and a Nuance language pack will be needed.

Although you can run *cress_vxml* from the command line, it is better to use the VoiceXML framework. In the *vxml* directory, issue the command:

```
cress_expand -v ivr main.vxml
```

A simpler way to create and deploy services is with, for example:

```
cress_realise -t lotos -v ivr
```

Since there is no unique root diagram for VoiceXML, you will need to adjust the configuration diagram to suit your application. The generated *.vxml file will then be particular to that application, and should be copied if necessary.

With V-Builder (version 1.2), open *main.vbuilder* in the *vxml* directory. This file is configured to use UK English and the VoiceXML file *main.vxml* (as generated by *cress_vxml*). Edit *main.vbuilder* if required (e.g. for an appropriate language).

Start the Nuance Vocalizer and choose a particular speaker (e.g. Tim Cooper, mu-law). Then in Nuance V-Builder use Tools/Run Dialog to start the VoiceXML system. Now you can use Run Dialog to execute the script, using a microphone for input and hearing the output on speakers. The DynaViz tab also shows an overview of execution.

A file *.vxml is a pre-translated version of a diagram, merged *on its own* with the root diagram.

CRESS File Manifest

25. General Files

25.1 Naming Conventions

The following file names are used in various directories. Archived specifications can be used directly (e.g. with Mustard or Clove) without having to regenerate them.

File	Purpose
<i>.archive</i>	archive directory for previous versions of files
<i>*.bpel</i>	root plus features as BPEL
<i>*.bpr</i>	root plus features as Business Process Archive
<i>*.chx</i>	Chive XML version of Cress diagram
<i>*.clove</i>	Clove verification properties
<i>*.cpl</i>	CPL service
<i>*.custom</i>	C header file for customised CADP type enumerations
<i>*.diagram2</i>	Diagram! version of Cress diagram
<i>*.f</i>	C header file for CADP function definitions
<i>*.lot</i>	root plus features in Lotos
<i>*.lotos</i>	Lotos file for use with CADP
<i>*.mustard</i>	Mustard scenarios
<i>*.pdd</i>	root plus features as Process Deployment Descriptor
<i>*.pdf</i>	PDF version of Cress diagram
<i>*.pl</i>	Perl file
<i>*.pr</i>	root plus features as SDL/PR
<i>.properties</i>	Java properties file (used by Mint)
<i>*.svl</i>	CADP Script Verification Language file
<i>*.t</i>	C header file for CADP types
<i>*.vbuilder</i>	V-Builder project file
<i>*.vxml</i>	root plus features as VoiceXML
<i>*.wsdl</i>	root plus features as WSDL
<i>*.wsr</i>	root plus features as Web Service Archive

25.2 Binary Files (directory *bin*)

File	Purpose
<i>cadp_annotate</i>	automatically annotate a Lotos specification for CADP

File	Purpose
<i>bpel_create</i>	create partner or BPEL service archive using ActiveBPEL
<i>bpel_deploy</i>	deploy partner or BPEL service archive using ActiveBPEL
<i>cress_alias</i>	C-shell aliases for commands during internal testing
<i>cress_bpel</i>	command-line script to translate diagrams to BPEL/WSDL
<i>cress_bpel.pm</i>	BPEL/WSDL translator functions
<i>cress_check</i>	command-line script to check collection of diagrams (used by Chive)
<i>cress_common.pm</i>	common functions
<i>cress_expand</i>	command-line script to expand embedded macros
<i>cress_cpl</i>	command-line script to translate diagrams to CPL
<i>cress_cpl.pm</i>	CPL translator functions
<i>cress_lexer.pm</i>	lexical analyser functions for Diagram! diagrams
<i>cress_lola</i>	command-line script to clean up Lola (Lotos) traces
<i>cress_lotos</i>	command-line script to translate diagrams to Lotos
<i>cress_lotos.pm</i>	Lotos translator functions
<i>cress_parser.pm</i>	parser functions
<i>cress_realise</i>	create and deploy services
<i>cress_sdl</i>	command-line script to translate diagrams to SDL
<i>cress_sdl.pm</i>	SDL translator functions
<i>cress_sdt</i>	command-line script to make Tau SDT (MSC/PR) from Lola (Lotos) traces
<i>cress_test</i>	command-line script to run JUnit tests on BPEL services
<i>cress_tidy</i>	command-line script to delete temporary Lola/Tau files
<i>cress_validate</i>	command-line script to validate a specification (used by Chive)
<i>cress_verify</i>	command-line script to verify a specification (used by Chive)
<i>cress_vocab.pm</i>	domain-specific vocabulary definitions
<i>cress_vxml</i>	command-line script to translate diagrams to VoiceXML
<i>cress_vxml.pm</i>	VoiceXML translator functions
<i>sdt_in</i>	IN analyser filter for Tau SDT
<i>sdt_ivr</i>	IVR analyser filter for Tau SDT
<i>sdt_sip</i>	SIP analyser filter for Tau SDT

25.3 Documentation Files (directory *doc*)

File	Purpose
<i>cress_manual.doc, cress_manual.pdf</i>	Cress manual in Microsoft Word and Adobe PDF format
<i>cress.html, cress.css, etc.</i>	Cress overview in HTML format
<i>todo.txt</i>	notes on further work

25.4 Support Files (directory *supp*)

File	Purpose
<i>cadp</i>	directory containing implementations of Cress types for CADP (the X versions modified from the originals)
<i>gt4</i>	directory containing extra code for compatibility with GT4.2
<i>palettes</i>	directory containing Chive and <i>Diagram!</i> palettes for Cress
<i>topo</i>	directory containing Stirling Lotos library for Cress
<i>types</i>	directory for generated type files

25.5 Test Files (directory *test*)

File	Purpose
<i>bpel-ds.test</i>	Device Service tests (Knopflerfish)
<i>sdl-ivr</i>	IVR data types tests (Tau)
<i>lotos-gs.test</i>	GS data types tests (Topo)
<i>lotos-ivr.test</i>	IVR data types tests (Topo)
<i>lotos-ws.test</i>	WS data types tests (Topo)
<i>testa</i>	test arrows and targets
<i>testb</i>	test GS/WS fork/join branches
<i>testf</i>	test features (guarded and unguarded)
<i>testg</i>	test guards
<i>testh</i>	test DS/GS/WS handlers
<i>testi</i>	test inputs
<i>testp</i>	test DS/GS/WS pick
<i>testr</i>	test repeats and loops

26. Application Domain Files

26.1 Chisel Files (directory *chisel*)

These spliced features for the IN are based on the original Chisel ones.

File	Purpose
<i>cfbl</i>	Call Forward Busy Line (feature)
<i>cnd</i>	Calling Number Delivery (feature)
<i>incf</i>	Intelligent Network Call Forwarding (feature)
<i>infb</i>	Intelligent Network Freephone Billing (feature)
<i>infr</i>	Intelligent Network Freephone Routing
<i>intl</i>	Intelligent Network Teen Line (feature)
<i>pots</i>	Plain Old Telephone Service (root)

<i>tcs</i>	Terminating Call Screening (feature)
<i>twc</i>	Three-Way Calling (feature)

26.2 Device Service Files (directory *ds*)

File	Purpose
<i>cupboard</i>	monitor cupboard switch input (root)
<i>door</i>	monitor door switch input and locking output (root)
<i>door_all</i>	lock or unlock all doors (feature)
<i>door_light</i>	turn on light on entering house (feature)
<i>door_lounge</i>	set lounge environment on entry (feature)
<i>fall</i>	monitor fall detector input (root)
<i>fall_movement</i>	report alert if no movement after fall (feature)
<i>heating</i>	monitor heating output (root)
<i>heating_frost</i>	issue frost warning if heating turned off in frosty conditions (feature)
<i>speech</i>	monitor speech input and output
<i>speech_help</i>	dialogue to ask for the weather forecast or for help (feature)
<i>weather</i>	dummy weather service (root)
<i>window</i>	monitor window switch input (root)

26.3 Grid Service Files (directory *gs*)

File	Purpose
<i>allocator</i>	perform translation of a job name (root)
<i>analyser</i>	frequency analysis of occupational data (root)
<i>config_gs</i>	GS configuration diagram
<i>doublemap</i>	map occupational data from scheme X to Y to Z (root)
<i>lookup</i>	perform parallel translation of a job name (root)
<i>matcher</i>	document comparison scores (root)
<i>scorer</i>	document content analysis scores (root)
<i>splitter</i>	split frequency analysis of occupational data (root)

26.4 Intelligent Network Files (directory *in*)

File	Purpose
<i>cc</i>	Charge Call
<i>cfbl</i>	Call Forward Busy Line
<i>cnd</i>	Calling Number Delivery
<i>config_in</i>	IN configuration diagram
<i>incf</i>	Intelligent Network Call Forwarding

<i>infb</i>	Intelligent Network Freephone Billing
<i>infr</i>	Intelligent Network Freephone Routing
<i>intl</i>	Intelligent Network Teen Line
<i>pots</i>	Plain Old Telephone Service (root)
<i>rc</i>	Return Call
<i>tcs</i>	Terminating Call Screening
<i>twc</i>	Three-Way Calling

26.5 Interactive Voice Response Files (directory *ivr*)

File	Purpose
<i>account</i>	account number request and server submission
<i>booking</i>	booking for room date/nights/type (root)
<i>config_gs</i>	IVR configuration diagram
<i>confirm</i>	confirmation request
<i>contact</i>	contact phone number request
<i>customer</i>	customer number request and server submission
<i>deaf</i>	do not listen during prompts
<i>donation</i>	donation for charity/amount (root)
<i>glucose</i>	blood sugar level checking (root)
<i>handwash</i>	hand washing (root)
<i>introduction</i>	introduction and generic handlers
<i>limb</i>	limb-donning (root)
<i>order</i>	order for quarry product/weight (root)
<i>pin</i>	PIN request and server submission
<i>restart</i>	restart request
<i>smoothie</i>	strawberry smoothie making (root)
<i>wait</i>	wait indefinitely during prompts

26.6 Session Initiation Protocol Files (directory *sip*)

File	Purpose
<i>agent</i>	User Agent (root)
<i>agent_cfbl</i>	Call Forward Busy Line for User Agent
<i>agent_tcs</i>	Terminating Call Screening for User Agent
<i>config_sip</i>	SIP configuration diagram
<i>proxy</i>	Proxy Server (root)
<i>proxy_cfbl</i>	Call Forward on Busy Line for Proxy Server
<i>proxy_tcs</i>	Terminating Call Screening for Proxy Server

<i>redirect</i>	Redirect Server (root)
-----------------	------------------------

26.7 Voice over Internet Protocol Files (directory *voip*)

Service	Purpose
<i>agent_check</i>	check on user-agent
<i>config_sip</i>	VoIP configuration diagram
<i>forward_always</i>	unconditional call forwarding
<i>forward_lookup</i>	forward through registration lookup
<i>forward_subject</i>	forward on subject
<i>forward_voicemail</i>	forward to voicemail
<i>priority_language</i>	forward by priority and language
<i>reject_busy</i>	reject based on caller
<i>reject_offline</i>	use of mail and reject
<i>reject_sunday</i>	reject based on time
<i>screen_premium</i>	reject based on call number

26.8 WS Files (directory *ws*)

File	Purpose
<i>address</i>	name/address validator
<i>broker</i>	car and loan arranger (root)
<i>config_ws</i>	WS configuration diagram
<i>lender</i>	loan arranger (root)
<i>normalise</i>	name normalisation
<i>supplier</i>	car supplier (root)

27. Target Language Files

27.1 BPEL/WSDL Files (directory *bpel*)

File	Purpose
<i>main.base</i>	BPEL framework (fixed specification base)
<i>analyser</i>	frequency analysis of occupational data
<i>broker</i>	car broker service
<i>cupboard</i>	cupboard device services
<i>door</i>	door device services
<i>fall</i>	fall device services
<i>lender</i>	lender service
<i>matcher</i>	document comparison scores

<i>scorer</i>	document content analysis scores
<i>splitter</i>	split frequency analysis of occupational data
<i>supplier</i>	car supplier service
<i>window</i>	window device services

27.2 CPL Files (directory *cpl*)

File	Purpose
<i>main.base</i>	CPL framework (fixed specification base)
<i>main.cpl</i>	CPL service (generated)

27.3 Lotos Files (directory *lotos*)

File	Purpose
<i>ds.base</i>	DS framework (fixed specification base)
<i>ds.ctx</i>	DS context (Topo)
<i>ds.lot</i>	DS working specification (generated)
<i>gs.base</i>	GS framework (fixed specification base)
<i>gs.ctx</i>	GS context (Topo)
<i>gs.lot</i>	GS working specification (generated)
<i>gs.make</i>	GS specification makefile
<i>in.base</i>	IN framework (fixed specification base)
<i>in.ctx</i>	IN context (Topo)
<i>in.lot</i>	IN working specification (generated)
<i>in.make</i>	IN specification makefile
<i>ivr.base</i>	IVR framework (fixed specification base)
<i>ivr.ctx</i>	IVR context (Topo)
<i>ivr.lot</i>	IVR working specification (generated)
<i>ivr.make</i>	IVR specification makefile
<i>sip.base</i>	SIP framework (fixed specification base)
<i>sip.ctx</i>	SIP context (Topo)
<i>sip.lot</i>	SIP working specification (generated)
<i>sip.make</i>	SIP specification makefile
<i>ws.base</i>	WS framework (fixed specification base)
<i>ws.ctx</i>	WS context (Topo)
<i>ws.lot</i>	WS working specification (generated)
<i>ws.make</i>	WS specification makefile

27.4 SDL Files (directory *sdl*)

File	Purpose
<i>BillingSystem.spr</i>	IN/SIP billing system process definition
<i>CallCoordinator.spr</i>	IN call coordinator process definition
<i>IN.map</i>	IN overall name to EPS file map
<i>IN.sdt</i>	IN overall Tau SDT control file
<i>INStructure.sbk</i>	IN main block definition
<i>INSystem.pr</i>	IN working SDL/PR for IN (generated)
<i>INSystem_All.pr</i>	IN working SDL/PR POTS plus all features (archive)
<i>INSystem.ssy</i>	IN overall system definition
<i>ServiceControl.spr</i>	IN service control point process definition
<i>StatusManager.spr</i>	IN/SIP status manager process definition
<i>Switch.spr</i>	IN switch process definition
<i>Application.spr</i>	IVR application process definition
<i>Recogniser.spr</i>	IVR recogniser process definition
<i>IVR.map</i>	IVR overall name to EPS file map
<i>IVR.sdt</i>	IVR overall Tau SDT control file
<i>IVRStructure.sbk</i>	IVR main block definition
<i>IVRSystem.pr</i>	IVR SDL/PR for IVR (generated)
<i>IVRSystem_Booking.pr</i>	IVR SDL/PR for BOOKING plus all features (archive)
<i>IVRSystem_Donation.pr</i>	IVR SDL/PR for DONATION plus all features (archive)
<i>IVRSystem_Order.pr</i>	IVR SDL/PR for ORDER plus all features (archive)
<i>IVRSystem.ssy</i>	IVR overall system definition
<i>AgentCoordinator.spr</i>	SIP agent coordinator process definition
<i>BillingSystem.spr</i>	SIP/IN billing system process definition
<i>ProtocolBuffer.spr</i>	SIP protocol buffer process definition
<i>ProxyServer.spr</i>	SIP proxy server process definition
<i>RedirectServer.spr</i>	SIP redirect server process definition
<i>SIP.map</i>	SIP overall name to EPS file map
<i>SIP.sdt</i>	SIP overall Tau SDT control file
<i>SIPStructure.sbk</i>	SIP main block definition
<i>SIPSystem.pr</i>	SIP pre-translated SDL/PR for system
<i>SIPSystem_Agent.pr</i>	SIP specification for AGENT plus all features (archive)
<i>SIPSystem_Proxy.pr</i>	SIP specification PROXY plus all features (archive)
<i>SIPSystem_Redirect.pr</i>	SIP specification REDIRECT plus all features (archive)
<i>SIPSystem.ssy</i>	SIP overall system definition
<i>StatusManager.spr</i>	SIP/IN status manager process definition
<i>UserAgent.spr</i>	SIP user agent process definition

27.5 VoiceXML Files (directory *vxml*)

File	Purpose
<i>main.base</i>	VoiceXML framework (fixed specification base)
<i>main.vbuilder</i>	V-Builder project file (fixed)
<i>main.vxml</i>	VoiceXML file (generated)

28. CRESS Licence

This is not open-source software. The authors (Kenneth J. Turner and others, University of Stirling) retain copyright in it. Nonetheless, the authors will normally approve its use by others subject to the following conditions:

- The software will be provided for use only within an organisation that accepts this licence. The software may not be distributed outside this organisation.
- The software may be used only for purposes of an investigative and non-commercial nature. The software may be used freely for these purposes. Diagrams and code supplied by the author remain the property of the author. However, anything additional created by the user (including generated code) is the property of the user.
- The software is provided without any warranty or guarantee as to its reliability or fitness for purpose. No responsibility is accepted by the author for direct or indirect consequences of using the software.
- Enhancements and corrections to the Cress diagrams and code supplied must be made freely available to the author.

29. History

Version 1.N of Cress was inspired by the Chisel work at BellCore. For this reason, the first letter of 'Cress' originally stood for 'Chisel'.

29.1 Versions 1.0 – 2.0

Ken Turner, December 1999 – April 2001.

29.2 Version 2.1

Ken Turner, 1st November 2001.

29.3 Version 2.2

Ken Turner, 28th February 2002.

29.4 Version 2.3

Ken Turner, 28th June 2002:

- added SIP vocabulary and SIP diagrams
- added support for 'Charge Call' through 'BillPIN' profile variable and 'CC' profile definition
- fixed some subtle bugs with diagram parsing when loops are present

29.5 Version 2.4

Ken Turner, 31st October 2002:

- added SIP support for SDL

29.6 Version 2.5

Ken Turner, 26th March 2003:

- added VoiceXML support for Lotos

29.7 Version 2.6

Ken Turner, 31st May 2003:

- added VoiceXML support for SDL

29.8 Version 2.7

Ken Turner, 25th February 2004:

- added support for yEd and GML files

29.9 Version 2.8

Ken Turner, 28th September 2004:

- added support for Chive diagrams
- extended support for Mustard translation into SDL

29.10 Version 2.9

Ken Turner, 24th October 2004:

- extended support for Mustard translation into MSCs
- added further Mustard test files

29.11 Version 3.0

Ken Turner, 13th May 2005:

- added BPEL and Lotos support for web services
- added initial BPEL/WSDL support for web services

29.12 Version 3.1

Ken Turner, 31st August 2005:

- extended BPEL and Lotos support for web services
- all diagrams (except some for testing) converted from *Diagram!* to *Chive*

29.13 Version 3.2

Ken Turner, 21st September 2005:

- Mustard separated from Cress
- extensions to SDL and VoiceXML code generation

- fixes to VoiceXML framework

29.14 Version 3.3

Ken Turner, 22nd November 2005:

- updated for Tau 4.6 (on Windows)

29.15 Version 3.4

Ken Turner, 4th August 2006:

- included support for grid services with Globus WS Core 4.0.1
- fixed WSDL creation if there are no types
- allowed (guarded) assignments on arcs following not just *invoke*
- archived files moved to *.arch* subdirectories of top-level directories
- updated for ActiveBPEL 2.1, Apache Tomcat 5.5.12 and JUnit 4.0
- code timings updated for faster processor
- documentation completely reworked as MS Word/PDF

29.16 Version 3.5

Ken Turner, 1st September 2006:

- enhanced Lotos specifications for *CONVERTER* and *STATISTICS*
- Mustard validation of *SPLITTER* (and implicitly *ANAYSER*)
- created Lotos specifications for *COUNTER* and *PARSER*
- Mustard validation of *MATCHER* (and implicitly *SCORER*)
- command-line help improved
- all diagrams redrawn with Chive 1.4 and converted to PDF

29.17 Version 3.6

Ken Turner, 27th August 2007:

- the archive directory is now named *.archive* rather than *.arch*
- addition of *cress_test* to run JUnit tests on BPEL services
- addition of CPL support (thanks to Dean McMenemy, University of Stirling)
- graceful detection and reporting of *supp* directory not being in *CRESSPATH*
- comments (*//*) must now be at the start of a line or preceded by white space
- *cress_bpel* now correctly handles assignments for **Else** if there are no peer assignments
- Cress common data types are now in the *stir* library, following substantial work on harmonising the data type definitions
- equations have been corrected for *lt*, *le*, *gt*, *ge* in *Text*
- the manual has been extended to deal with the above changes

29.18 Version 4.0

Ken Turner and Larry Tan, 31st July 2008:

- *cadp_annotate* has been added for automatic annotation of Lotos files for CADP
- *cress_bpel* and *cress_expand* now have option *'-b'* with value 1 (BPEL4WS 1.1) or 2 (WS-BPEL 2.0)
- *cress_bpel* now supports BPEL 2 assignments, dynamic partner assignment, expressions, extensions, imports, joins, link statuses, namespaces, sources, switches, targets, terminations, transition conditions, while loops

- for parent diagrams, *cress_bpel* now creates WSDL rather differently: all types and messages are associated with the namespace of their defining diagram, so WSDL definitions of these are created for the parent and imported
- *cress_common* now exports the Cress based directory as *\$cress*, and this is used by *cress_test* to locate GS/WS test files
- for Java compilation, *cress_create* now uses Java 1.5 as target since ActiveBPEL (version 3 onwards) runs under only JRE 1.5
- *cress_create* no longer prefixes BPR files with the diagram priority
- *cress_deploy* and *cress_expand* now use '-k' for the CPL server authorisation key
- *cress_lotos* now has option '-a' for generating CADP annotations (types, SVL scripts)
- *cress_lotos* now generates a slightly different type for Message for compatibility with Mustard 1.5
- *cress_parser.pm* diagnostics for the graph now include a level number (e.g. ': 3')
- *cress_parser.pm* now disregards an empty (Null) node as an offspring for the purposes of checking the consistency of offspring types
- *cress_parser.pm* now supports a node parameter with multiple parentheses such as '(w+)\.(d\d)', being ultimately ended by white space or the end of the list
- *cress_parser.pm* no longer removes an outer level of parentheses from a node parameter
- *cress_parser.pm* now allows the first template node to contain Perl regular expressions as well as literal parameters; parenthesised parts of a regular expression are numbered 1, etc. from left to right, and can be used as \$1, etc. in the body of a template
- the *broker* and *supplier* WS examples have been changed so that an unavailable car is priced at a *million* and not 999999

29.19 Version 4.1

Ken Turner and Larry Tan, 1st September 2008:

- all code made compatible with ActiveBPEL 5.0.2 and Globus WS Core 4.2:
 - 2005/08 version of WS-Addressing
 - new locations for Globus libraries and tools
 - GAR deployment now overwrites an existing deployment
 - *cress_create* and *cress_test* use the new Globus libraries
 - all GS examples updated
- *cress_bpel* now takes the '-o' option to generate operational test files for Mint (**.properties*)
- the *stir* Lotos library has been updated and needs to be made available to Topo/Lola:
 - definition of *percent* now rounds the result to the nearest integer
 - arithmetic and comparison of numbers now deal properly with fractional parts
- the *supp* directory has been structured into sub-directories for each application
- the GS Matcher-Scorer example now has the same list of common words for the Lotos specification and the Java implementation, and the list of separators in Lotos is now *.-*@/*

29.20 Version 4.2

Ken Turner and Larry Tan, 9th April 2009:

- all archive files moved to top level 'archive' directory, and *cress_alias* changed to match
- variables (including fault variables) can now be declared with an owner in the form *variable:owner*
- *cadp_annotate* now uses named flags for sorts and operations
- *chive_font* utility added to normalise Chive diagram fonts
- *cress_bpel* now includes command-line options in the headers of generated files
- *cress_bpel* and *cress_expand* now include partners in generated headers, e.g.:
features: ANALYSER/STATISTICS SPLITTER/CONVERTER, DATABASE

- *cress_bpel* and *cress_lotos* now expect partner specifications/implementations according to new placement rules (highest diagram for normal partners, first diagram for phantom partners)
- *cress_create* and *cress_test* now run the Java compiler quietly so as to suppress unchecked type warnings due to Axis
- *cress_deploy* now reports where it is (un)deploying services, and gives a cleaner warning if Globus is currently running when deployment is attempted
- *cress_expand* no longer takes the '-a' or '-b' options – these should be specified in the configuration diagram
- *cress_expand* now generates deployment messages for BPEL files if '-d' is not selected
- *cress_lotos* now uses a different strategy for dynamic partner assignment
- *cress_test* has been updated to use a different version of the MySQL JDBC connector
- *cress_validate* now handles BPEL validation with Mustard (Mint) using the '-p' option
- the GS DynaOcc example has been replaced by the more complex Lookup-Allocator example
- the suffix 'mstd' rather than 'mst' is now defined for Mustard files
- for consistency with WS partners, *cress_bpel* now deploys GS partners with URL '*.../<partner><port>*' rather than '*.../<partner>Service*'; GS partner implementations and tests have been changed to match
- for GS and WS, the input or output parameter of Invoke, Receive and Reply may now be given as '-' (meaning it is void and omitted)
- WS Echoer and Translator examples have been removed as they were just small illustrations

29.21 Version 4.3

Ken Turner and Larry Tan, 30th January 2010:

- a new DS (Device Services) domain has been added
- most diagrams have been redrawn for Arial Narrow 10-point as a consistent font, and for use with Chive 1.7
- constants in the code have been capitalised for clarity
- *HOME* is now a reserved diagram name (for DS)
- *cadp_annotate* has been modified as follows:
 - an empty **Where** clause is removed, even if a Lotos comment separates it from the associated **EndProc**
 - the '-x' option specifies maximum size limits for specified types, to reduce the size of arrays and strings for verification
- *cadp_annotate*, *cress_bpel*, *cress_cpl*, *cress_lotos*, *cress_sdl*, *cress_vxml* no longer have the '-a' option as SVL is now handled by other tools
- *cress_bpel* has been modified as follows:
 - for consistency with XML conventions, fault owners now precede names instead of following them (e.g. *lender:refusal.error*)
 - the partner given in **Receive** (initial only) or **Reply** is checked to be the same as the diagram name
 - an embedded **Receive** from an external partner is recognised; the generated WSDL has '_' appended to the partner name to mark it as a shadow partner
 - the translation of Throw correctly deals with explicit fault owners
 - time-related aspects have been added: date/time variables, **Timeout** events, and **Wait** activities
 - the translation of **scope**, **flow** and handlers has been substantially modified because of the need to support **Timeout**
 - diagram parameters not used in messages are now correctly handled
 - the kind of parameters used in Update nodes are noted
 - multiple arcs between the *same* two nodes are handled
 - the transition condition for **Else** following multiple guards now correctly uses 'or'

- multiple guarded assignments now work correctly
- feature diagrams are included in the list of features if they start with the name of the root diagram (e.g. the *DOOR_LIGHT* feature for root *DOOR*)
- for DS, a new **Device** action is translated just like **Invoke**, except that ‘_’ is appended to the partner name as being a shadow partner
- for DS, a *HOME* service is expected in the configuration diagram; this is used as the URI base for OSGi device partners, which are automatically declared with ‘_’ appended to their names as shadow partners
- *cress_create* and *cress_lotos* now take an ‘-f’ option that lists foreign (i.e. non-Cress) partners that should be ignored for creation, deployment and specification; this option is ignored by *cress_bpel*, *cress_cpl*, *cress_sdl* and *cress_vxml*
- *cress_create* now reports only *WSDLException* errors in order to accommodate DS, where a duplicate definition of the *Device* type is sometimes unavoidable.
- *cress_deploy* has been modified as follows:
 - (un)deploying files is more specific, with reports on success or failure of deployment to ActiveBPEL and Globus
 - for DS/GS/WS, *bpel/<diagram>* is now deleted if it exists initially, and WSDL files older than the BPEL diagram are not copied (to allow for a different feature selection on a previous run)
- *cress_lotos* has been modified in a number of ways:
 - the *stir* library has been updated with an *Environment* type for DS/GS/WS, and *parseNumber* now eliminates leading and trailing zeros as required; the new version is in the *supp/topo* directory
 - this *Environment* type requires a modified version of Lola (3.7.2) that recognises the *date_time* ‘constant’
 - ‘**Fork loose**’ is translated correctly
 - time-related aspects are supported as for *cress_bpel*, using the *date_time* operation now supported by Lola
 - for DS/GS/WS, the diagram in the fork label rather than the current diagram is used when checking the join condition, thus allowing use of template features
 - for DS/GS/WS, the equations for a ‘_Result’ type now use the prefix ‘x’ for ‘partner’, ‘scope’, ‘state’, ‘states’ and ‘status’ equation variables, to avoid a possible clash with a diagram parameter of the same name
 - for DS/GS/WS, node activities may have assignments appended to them
- *cress_parser.pm* has been modified as follows:
 - for consistency with XML conventions, variable owners now precede names instead of following them (e.g. *approver:proposal2*)
 - an identifier for a diagram or variable can start with an underscore
 - for DS/GS/WS, the first signal parameter is used literally without translating it
 - a template parameter can be given in double quotes if it must literally match the original parameter (e.g. "door.in.open")
- *cress_verify* has been added to verify Lotos specifications using Clove
- *cress_vocab.pm* has been renamed *cress_vocabulary.pm*, is now above *cress_common.pm* in the module hierarchy, and predefines a *device* type/variable for DS

29.22 Version 4.4

Ken Turner, 24th May 2010:

- for DS, *cress_bpel* append the root diagram name to the names of shadow partners; this avoids WSDL name clashes where the same shadow partner is used by several root diagrams
- for DS, *cress_create* ignores shadow partners by checking if their names start with <foreign>_ using names from the ‘-f’ option (e.g. ‘alert_fall’ is ignored if ‘alert’ is a foreign partner)

- *cadp_annotate* now removes **Where** (and any following comments) in more circumstances, notably if what follows is multiple comments or **EndSpec**
- *cadp_annotate* now checks that type limits are appropriately formatted
- *cadp_annotate* has been substantially reworked to handle limited arrays, sets and strings, and to handle CADP-like annotations in partner-defined types
- various function names used by *cadp_annotate* have been changed to match the new CADP header files
- *cress_common.pm* now has the correct value for FUNC_SUFFIX, and has had other suffixes defined for the benefit of Clove
- the '-a' option for *cress_lotos* now takes a comma-separated list of type limits, with '.' meaning no limits; this replaces the former '-x' option
- *cress_lotos* can accept the '-q' option to create qualified event dispatchers, meaning there is a separate event and compensation handler for each scope; this is to ensure that Cress-generated Lotos can be processed by CADP
- *cress_lotos* now generates annotations for arrays in *supp/types/types.pl*
- for DS/GS/WS, *cress_lotos* now treats an event with name **CatchAll** as matching any other event (irrespective of any associated data)
- for *cress_lotos* and *cress_sdl*, the '-r' option now takes a comma-separated list of services whose behaviour should be repeated at a leaf node; '.' means all services, while omitting this option means that behaviour stops at a leaf node
- the 'stir' library in *supp/topo* has been updated for limited arrays, sets and strings, and definitions have been supplied for the '%' natural operation and the '^' and '%%' number operations
- the CADP header files in *supp/cadp* have been updated
- the *CONVERTER* example is no longer recursive so that it can be used with CADP
- the *XMAPY* and *YMAPZ* partners for the *DOUBLEMAP* example have been slightly changed for validation with both Lotos and BPEL
- the *SUPPLIER* and *DEALER* examples have been modified to make verification feasible: *SUPPLIER* now cancels a quotation for the dealer that is not selected; a *DEALER* ignores an *order* or *cancel* for a deal that has not been offered; *DEALER* reference numbers are now computed as hash codes instead of being incremented
- Cress has been aligned with Clove 1.1 and now includes Clove property files
- Cress has been aligned with Mint 1.2
- Cress has been aligned with Mustard 1.8

29.23 Version 4.5

Ken Turner, 11th October 2010:

- 'Deep recursion on subroutine' warnings from Perl are now suppressed unless diagnostic reporting is requested ('-e 0')
- *supp/topo/stir.** have been updated to support *Hash* for IVR
- *supp/cadp/Stirling_TEXT.h* and *cress_expand* have been updated to have a different default text enumeration ("DEFAULT", "TEXT", "ENUMERATION")
- the *supp/diagram* directory has been renamed *palettes* and now contains a Chive palette for DS
- *cadp_annotate* now ignores an empty signatures file, and has modified mappings for the *stir* library as it affects IVR
- for IVR, the **Query** activity is now supported by *cress_lotos* and *cress_vxml*; this is like **Request** but with an implicit parameter and variable (e.g. *query3*, based on the node number) and implicit grammar (*boolean*), followed by **Yes/No/Else** checks on this value after an implicit **Filled** event

- for IVR, `<Clear>` now accepts numerical variables (e.g. *103*) and converts them to query variables for the convenience of **Query** (e.g. *query103* in VoiceXML, key *103* for *xqueries* in Lotos)
- for IVR, a repeated branch to a field (**Option**, **Query**, **Request**) is now translated correctly using `<goto nexitem='...'/>`; a repeated branch to something that is not a field is therefore illegal
- for IVR, new GLUCOSE, HANDWASH, LIMB and SMOOTHIE dialogues have been added (thanks to Lynne J. McMichael, University of Stirling)
- *cress_expand* now has a subroutine for file age differences, and now uses the age difference between *types.pl* and the generated Lotos file to decide whether to run *cadp_annotate*
- *cress_expand* now handles limited hashes for use with IVR
- in *cress_lotos*, *get_unknown* now correctly handles '(' following an identifier
- in *cress_lotos* for IVR, if the top-level process a field (**Option**, **Query**, **Request**), an extra call for the top-level process is suspended; the top-level process is also ended at code level 1
- in *cress_lotos* for IVR and SOA, events following **Start** are no longer put into the *handler* array, and this array is no longer sorted into name order
- in *cress_lotos* for IVR and SOA, more detailed comments are now generated for the event dispatcher processes
- *cress_lotos* now annotates for CADP even if it generates no annotations (provided '-a .' is specified)
- *cress_validate* now has '-b' (bit state hash size), '-d' (search depth) and '-q' (macro name qualifier) options as for Mustard
- *cress_verify* now has a '-q' (macro name qualifier) option as for Clove
- *cress_verify* now takes the relations *none*, *branch* (formerly *branching*), *strong*, *taucomp* (tau compression with branching reduction) and *tauconf* (tau confluence with branch reduction)

29.24 Version 4.6

Ken Turner, 27th October 2010:

- the IVR, GS and WS examples have been re-verified with CADP 2009-c and Clove 1.3
- *cress_lotos* again eliminates '^' from strings (meaning a space) and again allows '='
- *cress_lotos* now does not repeat behaviour on **Terminate** or an implicit **CatchAll** when translating for CADP (to avoid disallowed recursion via an event handler on either side of a **Fork**)
- *cress_lotos* now creates annotations in upper case for record constructors