NAME

projector − semi-composition and generation of Labelled Transition Systems

SYNOPSIS

bcg_open [*bcg_opt*] *spec*[**.bcg**] [*cc_opt*] **projector** [*projector_opt*] *interface*[**.bcg**] *result*[**.bcg**]

or:

exp.open [*exp_opt*] *spec*[**.exp**] [*cc_opt*] **projector** [*projector_opt*] *interface*[**.bcg**] *result*[**.bcg**]

or:

fsp.open [*fsp_opt*] *spec*[**.lts**] [*cc_opt*] **projector** [*projector_opt*] *interface*[**.bcg**] *result*[**.bcg**]

or:

lnt.open [*lnt_opt*] *spec*[**.lnt**] [*cc_opt*] **projector** [*projector_opt*] *interface*[**.bcg**] *result*[**.bcg**]

or:

lotos.open [*lotos_opt*] *spec*[**.lotos**] [*cc_opt*] **projector** [*projector_opt*] *interface*[**.bcg**] *result*[**.bcg**]

or:

seq.open [*seq_opt*] *spec*[**.seq**] [*cc_opt*] **projector** [*projector_opt*] *interface*[**.bcg**] *result*[**.bcg**]

DESCRIPTION

projector takes two different inputs:

- A Labelled Transition System (LTS), expressed either as a BCG graph *spec***.bcg,** a composition expression *spec***.exp,** an FSP program *spec***.lts,** an LNT program *spec***.lnt,** a LOTOS program *spec***.lotos,** or a sequence file *spec***.seq.**

- An LTS contained in the file *interface*[**.bcg**].

projector performs the semi-composition of *spec* with respect to the LTS *interface* and using a synchronization set determined by the **-sync** option.

The resulting LTS is stored in *result*, in BCG format.

The semantics of the semi-composition operator is given in **$CADP/doc/*/Krimm-Mounier-97.*** and in Section **FORMAL DEFINITION OF SEMI-COMPOSITION** below.

In a few words, **projector** reduces the LTS *spec* by suppressing the transitions that are not possible when *spec* is synchronized with *interface*, with respect to a given synchronization set (see **-sync** option below). The way transitions are suppressed depends on the nature of the interface. See option **-userinterface** for details.

OPTIONS

bcg_opt	if any, are passed to **bcg_lib**(LOCAL).
exp_opt	if any, are passed to **exp.open**(LOCAL).
fsp_opt	if any, are passed to **fsp.open**(LOCAL).

lnt_opt if any, are passed to **lnt.open**(LOCAL).

lotos_opt
if any, are passed to **caesar**(LOCAL) and **caesar.adt**(LOCAL).

seq_opt if any, are passed to **seq.open**(LOCAL).

cc_opt if any, are passed to the C compiler.

The following options *projector_opt* are currently available:

–monitor
Open a window for monitoring in real-time the generation of *result.bcg*. This is not a default option.

–sync [–total|–partial|–gate] *sync-file*
Use the synchronization rules defined in *sync-file*. The **-total**, **-partial**, and **-gate** options specify the "total matching", "partial matching", and "gate matching" semantics, respectively.

In "total matching" semantics, the regular expressions contained in *sync-file* denote full labels (i.e., gates possibly followed by experiment offers). Each transition of *spec* carrying a label that matches some rule in *sync-file* must be synchronized with a transition of *interface* carrying the same label.

In "partial matching" semantics, the regular expressions contained in *sync-file* denote substrings of labels. Each transition of *spec* carrying a label, a substring of which matches some rule in *sync-file*, must be synchronized with a transition of *interface* carrying the same label.

At last, in "gate matching" semantics, the regular expressions contained in *sync-file* denote gates. Each transition of *spec* carrying a label, the first word of which (called gate) matches some rule in *sync-file*, must be synchronized with a transition of *interface* carrying the same label. In this case, regular expressions in the synchronization set should not contain characters forbidden in gate identifiers (e.g., " ", "(", or "!").

Option **-total** is the default.

–hide [–total | –partial | –gate] *hiding_filename*
Use the hiding rules defined in *hiding_filename* to hide (on the fly) the labels of the LTS being generated. See the **caesar_hide_1**(LOCAL) manual page for a detailed description of the appropriate format for *hiding_filename*.

The **-total**, **-partial**, and **-gate** options specify the "total matching", "partial matching", and "gate matching" semantics, respectively. See the **caesar_hide_1**(LOCAL) manual page for more details about these semantics. Option **-total** is the default.

Note that label hiding does not operate on *spec*, but on the LTS resulting from the semi-composition of *spec* with respect to *interface*.**bcg**.

–rename [–total|–single|–multiple|–gate] *renaming_filename*

Use the renaming rules defined in *renaming_filename* to rename (on the fly) the labels of the LTS being generated. See the **caesar_rename_1**(LOCAL) manual page for a detailed description of the appropriate format for *renaming_filename*.

The **-total**, **-single**, **-multiple**, and **-gate** options specify the "total matching", "single partial matching", "multiple partial matching", and "gate matching" semantics, respectively. See the **caesar_rename_1**(LOCAL) manual page for more details about these semantics. Option **-total** is the default.

As for the **bcg_labels**(LOCAL) tool, several hiding and/or renaming options can be present on the command-line, in which case they are processed from left to right. However, the semi-composition (synchronizations) is always performed before hiding and/or renaming. Note that hiding and renaming only affect the part of refused labels that follow the "fail: " string.

Note that label renaming does not operate on *spec*, but on the LTS resulting from the semi-composition of *spec* with respect to *interface*.**bcg**.

–userinterface | -interfaceuser

This option is used when *interface* is a "user-given" interface, in which case **projector** computes validation predicate in the form of *refused transitions*, in order to check the validity of this interface. A refused transition is identified by its label, which begins with the string "fail: ". Such a label may be present in *spec*, but not in *interface*. When **–userinterface** is specified, **projector** may produce new refused transitions in addition to those already existing in *spec*.

Note that this option does not make sense outside an entire compositional generation process and it is mainly used by the **svl** tool. The validation predicates are checked by the **exp.open** tool, with **-interfaceuser** option. This is not a default option.

-uncompress, -compress, -register, -short, -medium, -size

These options control the form under which the BCG graph *result***.bcg** is generated. See the **bcg**(LOCAL) manual page for a description of these options.

-tmp This option specifies the directory in which temporary files are to be stored. See the **bcg**(LOCAL) manual page for a description of this option.

PROJECTOR USAGE

Even if **projector** may be used by the command line described in the **SYNOPSIS** section, it is mainly used by **svl***(LOCAL)* with **svl** script lines like:

```
"spec.lotos" -|| "interface.bcg"
"spec.lotos" -|[G1, G2]| "interface.bcg"
```

FORMAT FOR INPUT LABELLED TRANSITION SYSTEMS

The LTS expressed by *spec* does not have any restriction. In particular, it may contain refused transitions (i.e., labels prefixed by the special marker "fail: ").

The LTS contained in *interface* may not contain refused transitions. No other constraint is imposed (for instance, it may have non deterministic transitions).

Note: Minimizing *interface* for safety equivalence after hiding all labels that are not accepted by the synchronization set is likely to improve the time and space performances of **projector**, while not changing the resulting LTS. In **svl**, those two operations are automatically performed before calling **projector**.

FORMAT FOR OUTPUT LABELLED TRANSITION SYSTEM

The resulting LTS (*result*[**.bcg**]) may contain refused transitions. Such refused transitions form a loop from one state to the same state with a label prefixed by "fail: ".

FORMAT FOR SYNCHRONIZATION SETS

The description of the synchronization set is made through a file, which must respect the following grammar:

```
<sync-set>          ::=  <positive-header> '\n' <label-list>
                     |   <negative-header> '\n' <label-list>
<label-list>        ::=  <label>
                     |   <label> '\n' <label-list>
<label>             ::=  <regexp-denoting-a-label>
<positive-header> ::=  'Sync'
                     |   'sync'
<negative-header> ::=  'Sync all but'
                     |   'sync all but'
```

If the header is a positive header, any label matching a regular expression of the file will be considered as accepted by the synchronization set. On the contrary, if the header is a negative header, any label that does not match any regular expression of the file will be considered as accepted by the synchronization set.

The synchronization set should not accept any label prefixed by "fail: " if there is any in the graph module. The special label "i" should not be accepted by the synchronization set. If it does, a warning is issued but "i" is not synchronized.

When no synchronisation set is given, the synchronisation is performed on all labels of *spec* and *interface*, except the hidden label "i".

FORMAL DEFINITION OF SEMI-COMPOSITION

First of all, remind that an LTS is defined formally as a tuple (Q, A, T, q0), where Q is the set of states, A is a set of actions, T is the transition relation between states, and q0 is the initial state.

Let S1 and S2 be two LTSs, and SYNC be a set of labels (the synchronization set). We write (S1 |[SYNC]| S2) the parallel composition of S1 and S2, using an extension of LOTOS parallel composition, where SYNC contains full labels instead of gates.

The semi-composition (S1 -|[SYNC]| S2) is the LTS (Q, A, T, q0) defined as follows:

- Q is the set of states p1 of S1 such that for some state p2 of S2, (p1, p2) is reachable in (S1 |[SYNC]| S2).
- T is the set of transitions (p1, a, q1) of S1 such that for some states p2, q2 of S2 there exists a transition ((p1, p2), a, (q1, q2)) in (S1 |[SYNC]| S2).
- q0 is the initial state of S1.

FREQUENTLY ASKED QUESTIONS ABOUT SEMI-COMPOSITION

In the sequel, we describe LTSs as sets of transitions of the form q -a-> q'. The initial state will be generally noted q0, and sets of states and actions can be easily reconstructed from the transition relation.

Q: Does (S1 -|[SYNC]| S2) have always less states and transitions than S1?

A: Yes. One can see from the mathematical definition that (S1 -|[SYNC]| S2) cannot have more than the states and transitions of S1.

Q: If LTS S1 is minimal modulo an equivalence relation (e.g., strong, branching, etc.), will (S1 -|[SYNC]| S2) be also a minimal LTS?

A: No, as shows the following example. Take the minimal LTS S1 = {q0 -a-> q1, q0 -a-> q2, q1 -b-> q3}. The semi-composition (S1 -|[b]| S2), where S2 contains a unique state and no transition, results in {q0 -a-> q1, q0 -a-> q2}, which is not minimal (q1 and q2 are equivalent states).

Q: Starting from an LTS S1, I made the following two experiments: (1) simply minimize S1; (2) first restrict S1 by semi-composition and then minimize the result. Amazingly the LTS obtained by experiment (2) was larger than that obtained by experiment (1). Is this normal?

A: Yes, this may happen, for instance, if semi-composition breaks some symmetry in S1. Take S1 = {q0 -a-> q1, q1 -a-> q2, q2 -a-> q0}. Experiment (1), i.e., minimizing S1 yields S2 = {q0 -a-> q0}. Experiment (2) using (S1 -|[a]| S3) with S3 = {q0 -a-> q1, q1 -a-> q2} yields S3, which is minimal. S3 is effectively larger than S2.

Q: Are standard equivalence relations congruences for the semi-composition operator.

A: No. Take LTSs S1, S2, and S3 defined in previous example. S1 and S2 are equivalent, but (S1 -|[a]| S3) (which is equal to S3) is not equivalent to (S2 -|[a]| S3) (which is equal to S2). However, most equivalence relations (strong, branching, observational, tau*.a, safety) are semi-congruences for semi-composition, in the sense that minimizing the interface will not alter the result of semi-composition.

EXIT STATUS

Exit status is 0 if everything is alright, 1 otherwise.

DIAGNOSTICS

When the source is erroneous, error messages are issued.

BIBLIOGRAPHY

- Jean-Pierre Krimm. Une Approche Pratique pour la Verification Compositionnelle de Programmes LOTOS. Memoire de Magistere, September 1996. (In French.)
- Jean-Pierre Krimm and Laurent Mounier. Compositional State Space Generation from Lotos Programs. In Proceedings of TACAS'97, Tools and Algorithms for the Construction and Analysis of Systems (University of Twente, Enschede, The Netherlands), April 1997. Available from http://cadp.inria.fr/publications/Krimm-Mounier-97.html
- Hubert Garavel. OPEN/CAESAR: An Open Software Architecture for Verification, Simulation, and Testing. In Proceedings of the First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98, March 1998. Available from http://cadp.inria.fr/publications/Garavel-98.html
- Hubert Garavel, Frederic Lang. SVL: a Scripting Language for Compositional Verification. In Proceedings of the 21st IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2001 (Cheju Island, Korea), August 2001. Available from http://cadp.inria.fr/publications/Garavel-Lang-01.html

AUTHORS

Version 1.x of **projector** was developped by Jean-Pierre Krimm using an algorithm written by Laurent Mounier and himself (VERIMAG).

Version 2.x of **projector** was totally rewritten from scratch by Gordon Pace, Bruno Ondet, Nicolas Descoubes, and Frederic Lang (INRIA Rhone-Alpes/VASY).

Version 3.x of **projector** was partially rewritten by Remi Herilier and Frederic Lang (INRIA Rhone-Alpes/VASY).

OPERANDS

spec.**bcg**	BCG graph (input)
spec.**exp**	network of communicating LTSs (input)

spec.**lts**	FSP specification (input)
spec.**lnt**	LNT specification (input)
spec.**lotos**	LOTOS specification (input)
spec.**seq**	sequence file (input)
interface.**bcg**	interface LTS (input)
sync-file.**sync**	set of synchronization labels (input)
result.**bcg**	BCG graph (output)

FILES

The binary code of **projector** is available in **$CADP/bin.‘arch‘/projector.a**

SEE ALSO

CAESAR Reference Manual, OPEN/CAESAR Reference Manual, **bcg**(LOCAL), **bcg_open**(LOCAL), **caesar_hide_1**(LOCAL), **caesar_rename_1**(LOCAL), **caesar**(LOCAL), **caesar.adt**(LOCAL), **exp**(LOCAL), **exp.open**(LOCAL), **fsp.open**(LOCAL), **lnt.open**(LOCAL), **lotos**(LOCAL), **lotos.open**(LOCAL), **seq**(LOCAL), **seq.open**(LOCAL), **svl**(LOCAL)

Additional information is available from the CADP Web page located at http://cadp.inria.fr

Directives for installation are given in files **$CADP/INSTALLATION_*.**

Recent changes and improvements to this software are reported and commented in file **$CADP/HISTORY.**

BUGS

Please report new bugs to cadp@inria.fr