NAME

bisimulator − on-the-fly equivalence/preorder checking

SYNOPSIS

bcg_open [*bcg_opt*] *lts1*[**.bcg**] [*cc_opt*] **bisimulator** [*bisimulator_opt*] *lts2*[**.bcg**]

or:

exp.open [*exp_opt*] *lts1*[**.exp**] [*cc_opt*] **bisimulator** [*bisimulator_opt*] *lts2*[**.bcg**]

or:

fsp.open [*fsp_opt*] *lts1*[**.lts**] [*cc_opt*] **bisimulator** [*bisimulator_opt*] *lts2*[**.bcg**]

or:

lnt.open [*lnt_opt*] *lts1*[**.lnt**] [*cc_opt*] **bisimulator** [*bisimulator_opt*] *lts2*[**.bcg**]

or:

lotos.open [*lotos_opt*] *lts1*[**.lotos**] [*cc_opt*] **bisimulator** [*bisimulator_opt*] *lts2*[**.bcg**]

or:

seq.open [*seq_opt*] *lts1*[**.seq**] [*cc_opt*] **bisimulator** [*bisimulator_opt*] *lts2*[**.bcg**]

DESCRIPTION

bisimulator takes as inputs two Labelled Transition Systems (LTSs), the first one being represented either as a BCG graph *lts1***.bcg**, a composition expression *lts1***.exp**, an FSP program *lts1***.lts**, an LNT program *lts1***.lnt**, a LOTOS program *lts1***.lotos**, or a sequence file *lts1***.seq**, and the second one being represented as a BCG graph *lts2***.bcg**. Traditionally, *lts1* represents the behaviour of a *protocol* and *lts2* represents the behaviour of its *service*.

bisimulator performs an on-the-fly comparison of the two LTSs *lts1* and *lts2* modulo a given equivalence/preorder relation (see EQUIVALENCE RELATIONS below). The result of this verification (TRUE or FALSE) is displayed on the standard output, possibly accompanied by a diagnostic (see OPTIONS below).

Note: The verification method underlying the current version of **bisimulator** is based upon a translation of the equivalence/preorder checking problem into the resolution of a Boolean Equation System (BES), which is performed on-the-fly using the algorithms provided by the **caesar_solve_1**(LOCAL) library of OPEN/CAESAR (see the corresponding manual page and the article [Mat06] for details).

OPTIONS

The options *bcg_opt*, if any, are passed to **bcg_lib**(LOCAL).

The options *exp_opt*, if any, are passed to **exp.open**(LOCAL).

The options *fsp_opt*, if any, are passed to **fsp.open**(LOCAL).

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

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

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

The options *cc_opt*, if any, are passed to the C compiler.

The options *bisimulator_opt* currently available are described below.

The options below specify the equivalence relation used for comparing *lts1* and *lts2*.

−branching
Use branching equivalence (resp. its corresponding preorder) as equivalence (resp. preorder) relation for comparing *lts1* and *lts2*. Not a default option.

−observational
Use observational equivalence (resp. its corresponding preorder) as equivalence (resp. preorder) relation for comparing *lts1* and *lts2*. Not a default option.

−safety Use safety equivalence (resp. its corresponding preorder) as equivalence (resp. preorder) relation for comparing *lts1* and *lts2*. Not a default option.

−strong Use strong equivalence (resp. its corresponding preorder) as equivalence (resp. preorder) relation for comparing *lts1* and *lts2*. Default option.

−taustar
Use *tau*.a* equivalence (resp. its corresponding preorder) as equivalence (resp. preorder) relation for comparing *lts1* and *lts2*. Not a default option.

−trace Use trace equivalence (resp. its corresponding preorder) as equivalence (resp. preorder) relation for comparing *lts1* and *lts2*. Not a default option.

−weaktrace
Use weak trace equivalence (resp. its corresponding preorder) as equivalence (resp. preorder) relation for comparing *lts1* and *lts2*. Not a default option.

The options below specify the kind of comparison between *lts1* and *lts2*.

−smaller
Check whether *lts1* is included in *lts2* modulo the preorder corresponding to the equivalence relation considered (if the two LTSs are equivalent, they are also included one into the other modulo the corresponding preorder). Not a default option.

−equal Check whether *lts1* is equivalent to *lts2* modulo the equivalence relation considered. Default option.

−greater
Check whether *lts2* is included in *lts1* modulo the preorder corresponding to the equivalence relation considered (if the two LTSs are equivalent, they are also included one into the other modulo the corresponding preorder). Not a default option.

The options below specify the algorithm used for comparing *lts1* and *lts2*.

−bfs Compare *lts1* and *lts2* using a breadth-first search algorithm. Compared to **-dfs**, this option is generally slower, but produces counterexamples of smaller depth. Not a default option.

−dfs Compare *lts1* and *lts2* using a depth-first search algorithm. Compared to **-bfs**, this option produces counterexamples of greater depth, but is generally faster and consumes less memory if *lts2* is deterministic (for strong equivalence) and has no invisible actions (for weak equivalences). Default option.

The options below specify various features available in addition to the comparison of *lts1* and *lts2*.

-bes [*file*[**.bes**[*.ext*]]]

Print in *file*[**.bes**] or, if the file name argument is missing, in file **bisimulator.bes**, a textual description of the BES corresponding to the comparison of *lts1* and *lts2* modulo the equivalence/preorder relation considered. If present, the extension *.ext* must correspond to a known file compression format (e.g., *.Z*, *.gz*, *.bz2*, etc.). In this case, the file containing the BES is compressed according to the corresponding format. The list of currently supported extensions and compression formats is given by the **$CADP/src/com/cadp_zip** shell-script. This option does not influence the comparison between the two LTSs. Not a default option.

-diag [**-minimal**] [*diag*[**.bcg**]]

When the comparison of *lts1* and *lts2* yields FALSE, generate a diagnostic (counterexample) in BCG format (see the **bcg**(LOCAL) manual page for details) explaining this result. The diagnostic is generated in the file *diag*.**bcg** or, if the file name argument is missing, in file **bisimulator.bcg**. This option has no effect when the comparison of *lts1* and *lts2* yields TRUE, since in this case the diagnostic would be larger than *lts1* and *lts2*, and would not bring any useful information. The BCG file containing the diagnostic can be visualized using the **bcg_draw**(LOCAL) and **bcg_edit**(LOCAL) tools of CADP (see respective manual pages for details).

The diagnostic is a directed acyclic graph included (modulo the preorder corresponding to the equivalence relation considered) both in *lts1* and *lts2*. Each state *p* of the diagnostic corresponds to a couple of states (*q*, *r*) belonging to *lts1* and *lts2*, respectively; the portion of diagnostic going out of *p* illustrates why the two corresponding states *q* and *r* are not equivalent. The terminal states of the diagnostic have additional "error" outgoing transitions with labels of the form "Present in lts2.bcg: *b*" or "Absent in lts2.bcg: *b*", indicating that the action *b* does not occur either in *lts1*, or in *lts2*, respectively (*b* can be either a visible action, or the invisible action *tau*, see EQUIVALENCE RELATIONS below for naming conventions). Intuitively, all transition sequences contained in the diagnostic lead, when executed simultaneously in *lts1* and *lts2*, to states which are unrelated modulo the equivalence/preorder relation considered. Note that for weak equivalences, any transition *p1--b-->p2* in the diagnostic may correspond to sequences of the form *q1--tau*.b-->q2* and *r1--tau*.b-->r2* contained in *lts1* and *lts2*, respectively. Also, any transition *p1--tau-->p2* in the diagnostic may correspond to a sequence *q1--tau-->q2* contained in *lts1* and possibly to a sequence *r1--tau*-->r2* contained in *lts2*, or vice-versa.

In the case of branching equivalence, the diagnostic may also contain some transitions of the form *p1--b-->p2* leading to sink states. Considering that *p1* corresponds to the couple of states (*q1*, *r1*), such a transition indicates the existence of two sequences of the form *q1--tau*-->q2--b-->q3* and *r1--tau*-->r2--b-->r3* in *lts1* and *lts2*, respectively, such that the states *q2* and *r2* are not branching equivalent. For each transition *p1--b-->p2* leading to a sink state in the diagnostic, the remainder of the diagnostic going out of *p1* illustrates the non equivalence of the states *q2* and *r2*. This specific handling of branching equivalence is due to the nature of this relation, which (at the opposite of other relations, such as strong, observational, *tau*.a*, and safety) requires that not only the target states of transitions, but also their source states are equivalent.

If the additional option **-minimal** is specified, a small-depth diagnostic is generated (the depth is guaranteed to be minimal only when the diagnostic is a tree).

If the diagnostic is a sequence of transitions, it will also be displayed on standard output using the SEQ format (see the **seq**(LOCAL) manual page for the definition of this format). Not a default option.

−stat Display statistical information about the resolution of the BES corresponding to the comparison of *lts1* and *lts2* modulo the equivalence/preorder relation considered. Not a default option.

−tauconfluence
Reduce *lts1* on-the-fly modulo tau-confluence (a form of partial order reduction that preserves branching equivalence) while performing the comparison with *lts2*. This option can be used only in conjunction with options **-branching** and **-observational**, and in some cases it may improve speed and memory consumption significantly. Not a default option.

EQUIVALENCE RELATIONS

An LTS is a quadruple *M* = (*Q*, *A*, *T*, *q0*), where: *Q* is the set of *states*, *A* is the set of *actions* (transition labels), *T* included in *Q* ∗ *A* ∗ *Q* is the *transition relation*, and *q0* is the *initial state*. The set *A* contains the invisible action *tau*, which denotes internal (unobservable) activity. A transition (*p*, *a*, *q*) in *T* (also noted *p--a-->q*) means that the system can evolve from state *p* to state *q* by performing action *a*. If *L* is a language included in *A*∗, then *p--L-->q* denotes a transition sequence *p--a1-->q2--a2-->...--an-->q* such that the word *a1a2...an* belongs to *L*. All states *q* of *Q* are assumed to be reachable from the initial state *q0* via sequences of transitions in *T* (i.e., *q0--A*∗*-->q*). In the sequel, visible actions of *A* are denoted by *a*, and (both visible and invisible) actions of *A* are denoted by *b*. The transitive and reflexive closure of *T* is denoted by *T*∗.

Two LTSs *M1* = (*Q1*, *A*, *T1*, *q01*) and *M2* = (*Q2*, *A*, *T2*, *q02*) are related modulo an equivalence relation *R* (noted *M1 R M2*) if and only if their initial states are related modulo *R* (noted *q01 R q02*). The equivalence relations currently supported by **bisimulator** are defined below. For each equivalence *R_equ*, the corresponding preorder relation *I_equ*, which indicates whether a state *p* is ‘‘simulated’’ by a state *q* (resp. *q* is ‘‘simulated’’ by *p*) is obtained by keeping only condition 1 (resp. 2) in the definition of *R_equ*.

Strong equivalence [Par81]

This is the largest relation *R_str* such that two states *p* and *q* are related modulo strong equivalence (*p R_str q*) if and only if:

1. for each transition *p--b-->p’* in *T1*
   there is a transition *q--b-->q’* in *T2*
   such that *p’ R_str q’*
2. for each transition *q--b-->q’* in *T2*
   there is a transition *p--b-->p’* in *T1*
   such that *p’ R_str q’*

Branching equivalence [GW89]

This is the largest relation *R_bra* such that two states *p* and *q* are related modulo branching equivalence (*p R_bra q*) if and only if:

1. for each transition *p--b-->p’* in *T1*

a. either *b = tau* and *p' R_bra q*, or
b. there is a sequence *q--tau*-->q'--b-->q''* in *T2**
such that *p R_bra q'* and *p' R_bra q''*

2. for each transition *q--b-->q'* in *T2*
a. either *b = tau* and *p R_bra q'*, or
b. there is a sequence *p--tau*-->p'--b-->p''* in *T1**
such that *p' R_bra q* and *p'' R_bra q'*

Observational equivalence [Mil89]

This is the largest relation *R_obs* such that two states *p* and *q* are related modulo observational equivalence (*p R_obs q*) if and only if:

1. a. for each transition *p--tau-->p'* in *T1*
there is a sequence *q--tau*-->q'* in *T2**
such that *p' R_obs q'*
b. for each transition *p--a-->p'* in *T1*
there is a sequence *q--tau*.a.tau*-->q'* in *T2**
such that *p' R_obs q'*

2. a. for each transition *q--tau-->q'* in *T2*
there is a sequence *p--tau*-->p'* in *T1**
such that *p' R_obs q'*
b. for each transition *q--a-->q'* in *T2*
there is a sequence *p--tau*.a.tau*-->p'* in *T1**
such that *p' R_obs q'*

Tau*.a equivalence [FM91]

This is the largest relation *R_tau* such that two states *p* and *q* are related modulo *tau*.a* equivalence (*p R_tau q*) if and only if:

1. for each sequence *p--tau*.a-->p'* in *T1**
there is a sequence *q--tau*.a-->q'* in *T2**
such that *p' R_tau q'*

2. for each transition *q--tau*.a-->q'* in *T2**
there is a sequence *p--tau*.a-->p'* in *T1**
such that *p' R_tau q'*

Safety equivalence [BFG+91]

This is the largest relation *R_saf* such that two states *p* and *q* are related modulo safety equivalence (*p R_saf q*) if and only if:

1. *p I_tau q*

2. *q I_tau p*

Safety equivalence is defined in terms of the *tau*.a* preorder *I_tau*. It is a *simulation equivalence* rather than a *bisimulation* (e.g., like *tau*.a* equivalence), because it only requires that states *p* and *q* are included one into the other modulo *I_tau*, and does not require that each *tau*.a*-successor of *p* (resp. *q*) is equivalent to a corresponding *tau*.a*-successor of *q* (resp. *p*). Therefore, safety equivalence is weaker than *tau*.a* equivalence (see the note below), but it has the same associated preorder (i.e., *I_saf = I_tau*).

Trace equivalence (a.k.a. language equivalence)

This is the largest relation *R_tra* such that two states *p* and *q* are related modulo trace equivalence (p R_tra q) if and only if:

1. for each sequence p--b1...bn-->p' in T1*
   there is a sequence q--b1...bn-->q' in T2*
2. for each sequence q--b1...bn-->q' in T2*
   there is a sequence p--b1...bn-->p' in T1*

Weak trace equivalence [BHR84]

Two states p and q are related modulo weak trace equivalence (p R_wtr q) if and only if:

1. for each sequence p--tau*.a1...tau*.an-->p' in T1*
   there is a sequence q--tau*.a1...tau*.an-->q' in T2*
2. for each sequence q--tau*.a1...tau*.an-->q' in T2*
   there is a sequence p--tau*.a1...tau*.an-->p' in T1*

Note: A relation R1 is said to be *stronger* than another relation R2 (noted R1 <= R2) iff p R1 q implies p R2 q for any states p, q. The relations above are ordered w.r.t. their strength as follows:

R_str <= R_bra <= R_obs <= R_saf <= R_wtr

R_str <= R_tra <= R_wtr

R_bra <= R_tau <= R_saf

As opposed to R_str and R_tra (the strong and trace equivalences), which handle all transition labels in the same way, the relations R_bra, R_obs, R_tau, R_saf, and R_wtr are called *weak* equivalences, since each of them performs a kind of abstraction over invisible actions.

Note: To obtain maximal performance, it is recommended to put the ‘‘bigger’’ LTS (the protocol) in argument *lts1* and the ‘‘smaller’’ LTS (the service) in argument *lts2*. In addition, the service LTS *lts2* can be minimized before comparison, either modulo strong equivalence (when strong equivalence is considered for comparing *lts1* and *lts2*), or modulo branching equivalence (if a weak equivalence is considered) using the **bcg_min**(LOCAL) tool of CADP (see the corresponding manual page for details). This restriction will be eliminated in a future version of **bisimulator**.

EXIT STATUS

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

BIBLIOGRAPHY

[BHR84]

S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A Theory of Communicating Sequential Processes. Journal of the ACM 31(3):560-599, July 1984.

[BDJ+05]

D. Bergamini, N. Descoubes, C. Joubert, and R. Mateescu. BISIMULATOR: A Modular Tool for On-the-Fly Equivalence Checking. In N. Halbwachs and L. Zuck (Eds.), Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2005 (Edinburgh, Scotland, UK), Lecture Notes in Computer Science vol. 3440, p. 581-585. Springer Verlag, April 2005. Available from http://cadp.inria.fr/publications/Bergamini-Descoubes-Joubert-Mateescu-05.html

[BFG+91]

A. Bouajjani, J-C. Fernandez, S. Graf, C. Rodriguez, and J. Sifakis. Safety for Branching Time

Semantics. In Proceedings of 18th ICALP. Springer Verlag, July 1991.

[FM91] J-C. Fernandez and L. Mounier. ''On the Fly'' Verification of Behavioural Equivalences and Preorders. In K. G. Larsen and A. Skou (Eds.), Proceedings of the 3rd Workshop on Computer-Aided Verification CAV'91 (Aalborg, Denmark), Lecture Notes in Computer Science vol. 575. Springer Verlag, July 1991.

[GW89] R. J. van Glabbeek and W. P. Weijland. Branching-Time and Abstraction in Bisimulation Semantics. In Proceedings of the IFIP 11th World Computer Congress (San Francisco, USA), 1989.

[Mat06] R. Mateescu. CAESAR_SOLVE: A Generic Library for On-the-Fly Resolution of Alternation-Free Boolean Equation Systems. Springer International Journal on Software Tools for Technology Transfer (STTT), 8(1):37-56, 2006. Full version available as INRIA Research Report RR-5948. Available from http://cadp.inria.fr/publications/Mateescu-06-a.html

[Mil89] R. Milner. Communication and Concurrency. Prentice-Hall, 1989.

[Par81] D. Park. Concurrency and Automata on Infinite Sequences. In Peter Deussen (Ed.), Theoretical Computer Science, Lecture Notes in Computer Science vol. 104, p. 167-183. Springer Verlag, March 1981.

AUTHORS

Radu Mateescu, with the help of Damien Bergamini (both at INRIA/VASY), who implemented a first version of the encoding of branching equivalence in terms of boolean equation systems.

OPERANDS

lts1.**bcg**	BCG graph (input)
lts1.**exp**	network of communicating LTSs (input)
lts1.**lts**	FSP specification (input)
lts1.**lnt**	LNT specification (input)
lts1.**lotos**	LOTOS specification (input)
lts1.**seq**	sequence file (input)
lts2.**bcg**	BCG graph (input)
diag.**bcg**	diagnostic in BCG format (output)
file.**bes**	BES in textual format (output)

FILES

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

SEE ALSO

bcg(LOCAL), **bcg_open**(LOCAL), **exp**(LOCAL), **exp.open**(LOCAL), **fsp.open**(LOCAL), **lnt.open**(LOCAL), **lotos.open**(LOCAL), **seq**(LOCAL), **seq.open**(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 bugs to Radu.Mateescu@inria.fr