NAME

bcg_min − minimization of normal, probabilistic, or stochastic labeled transitions systems (LTS) encoded in the BCG format

SYNOPSIS

bcg_min [*bcg_options*] [**-strong** | **-branching** | **-divbranching** | **-observational**] [**-normal** | **-prob** | **-rate** [**-self**]] [**-epsilon** *eps*] [**-format** *format_string*] [**-class** *class_file*] *input***.bcg** [*output***.bcg**]

where *bcg_options* is defined below (see GENERAL OPTIONS).

bcg_min takes as input the BCG graph *input***.bcg**, minimizes this graph according to some bisimulation relation, and writes the resulting reduced graph to *output***.bcg**, replacing *input***.bcg** if *output***.bcg** is omitted.

DESCRIPTION

bcg_min implements various algorithms to perform minimization of graphs encoded in the BCG format according to strong bisimulation, branching bisimulation, divergence-sensitive branching bisimulation, or observational equivalence. A graph input or output by **bcg_min** can be:

- either a "normal" LTS, whose transitions are either labelled with "normal" labels or with the "internal" label (usually noted "tau" in the scientific literature and displayed as the character string "**i**" by the various BCG tools),

- or a "probabilistic LTS": these are LTS with "normal" labelled transitions, as well as "special" transitions, whose labels are either of the form "**prob %p**" or "*label*; **prob %p**", where **%p** denotes a floating-point number in the range]0..1] and *label* denotes a character string that does not contain the "**;**" character. For each state, the sum of "**%p**" values on special transitions leaving the state must be less or equal than 1 (see ANNEX 1 below for a discussion on how the probabilistic LTS model of **bcg_min** generalizes other theoretical models published in the literature),

- or a "stochastic LTS": these are LTS with "normal" labelled transitions, as well as "special" transitions, whose labels are either of the form "**rate %f**" or "*label*; **rate %f**", where **%f** denotes a strictly positive floating-point number and *label* denotes a character string that does not contain the "**;**" character (see ANNEX 2 below for a discussion on how the stochastic LTS model of **bcg_min** generalizes other theoretical models published in the literature).

GENERAL OPTIONS

The following *bcg_options* are currently supported: **-version**, **-create**, **-update**, **-remove**, **-cc**, **-tmp**, **-uncompress**, **-compress**, **-register**, **-short**, **-medium**, and **-size**. See the **bcg**(LOCAL) manual page for a description of these options.

PARTICULAR OPTIONS

The following options are also supported:

-strong

Perform LTS reduction according to strong bisimulation. On (Discrete or Continuous Time) Markov Chains and on Markov Reward Models, this reduction agrees with lumpability of [KS76] (see ANNEX 1, ANNEX 2, and BIBLIOGRAPHY below). Default option.

-branching

Perform LTS reduction according to branching bisimulation. It is worth noticing that the notion of branching bisimulation is rather meaningless for probabilistic systems. Not a default option.

-divbranching

Perform LTS reduction according to divergence-sensitive branching bisimulation [GW96]. Divergence-sensitive branching bisimulation differs from branching bisimulation only in the way cycles of internal transitions (also called divergences) are treated. It is known that all states traversed by a cycle of internal transitions belong to the same branching equivalence class. While divergences are eliminated in the LTS obtained by reduction modulo ordinary branching bisimulation, a self-looping internal transition is kept in each such equivalence class in the LTS obtained by divergence-sensitive branching bisimulation. Unlike branching bisimulation, divergence-sensitive branching bisimulation preserves inevitability properties. Like branching bisimulation, it is worth noticing that the notion of divergence-sensitive branching bisimulation is rather meaningless for probabilistic systems. Not a default option.

-observational

Perform LTS reduction according to observational equivalence [Mil89]. It is worth noticing that observational equivalence is computationally more expensive than branching bisimulation, so that reduction may fail even for graphs containing only few thousands of states. To reduce the risk of failure, in a first step the input graph is automatically reduced according to branching bisimulation. This is sound because branching bisimulation is a graph relation stronger than observational equivalence. However, this optimisation is not applied if the **-class** option is set, so that **bcg_min** can print the equivalence classes relatively to the states of the input graph, instead of the states of the branching minimal intermediate graph produced in the first step. This option cannot be combined with neither **-prob** nor **-rate** options. Not a default option.

-normal

Consider *input***.bcg** as a normal LTS. With this option, labels of the form "**`rate %f`**" or "*label*`; rate %f`" or "**`prob %p`**" or "*label*`; prob %p`" are processed as ordinary labels, without special meaning attached. Default option.

-prob

Consider *input***.bcg** as a probabilistic LTS. With this option, each label of the form "**`prob %p`**" or "*label*`; prob %p`" is recognized as denoting a probabilistic transition with probability **`%p`**. **bcg_min** will stop with an error message if, for some probabilistic transition, **`%p`** is out of]0..1], or if the probabilistic transitions going out of the same state have a cumulated sum strictly greater than 1. With this option, labels of the form "**`rate %f`**" or "*label*`; rate %f`" are processed as ordinary labels. Not a default option.

-rate [**-self**]

Consider *input***.bcg** as a stochastic LTS. With this option, each label of the form "**`rate %f`**" or "*label*`; rate %f`" is recognized as denoting a stochastic transition with rate **`%f`**. **bcg_min** will stop with an error message if, for some stochastic transition, **`%f`** is less or equal to 0. If the **-branching** or the **-divbranching** option is selected, and some state has both an outgoing stochastic transition and an outgoing internal (i.e., "tau") transition, **bcg_min** will print a warning and remove the stochastic transition in order to preserve the notion of maximal progress. With this option, labels of the form "**`prob %p`**" or "*label*`; prob %p`" are processed as ordinary labels. Not a default option.

If **-self** sub-option is given, all self loops (i.e., transitions that remain within the same equivalence class) having labels of the form "**`rate %f`**" are removed. This implements the weak Markovian equivalences described in [Bra02] and [BHKW05]. Not a default sub-option.

-epsilon *eps*

Set the precision of floating-point comparisons to *eps*, where *eps* is a real value. When *eps* is out

of [0..1[, **bcg_min** reports an error. Default value for *eps* is 1E-6.

-format *format_string*

Use *format_string* to control the format of the floating-point numbers contained in transition labels (these numbers correspond to the occurrences of **`%f`** and **`%p`** mentioned in section DESCRIPTION above). The value of *format_string* should obey the same conventions as the *format* parameter of function **sprintf**(3C) for values of type **`double`**. Default value for *format_string* is **`"%g"`**, meaning that floating-point numbers are printed with at most six digits after the "." (i.e., the radix character). Other values can be used, for instance **`"%.9g"`** to obtain nine digits instead of six, or by replacing the **`"%g"`** flag with other flags, namely **`"%e"`**, **`"%E"`**, **`"%f"`**, **`"%G"`**, possibly combined with additional flags (e.g., to specify precision).

-class *class_file*

If *class_file* is the character '-', then display the equivalence classes on the standard output. Otherwise, display the equivalence classes in a file named *class_file*. Not a default option.

Note: In **bcg_min** versions up to 2.1, option **-class** was not followed by a *class_file* argument and equivalence classes were always displayed on the standard output. The *class_file* argument was introduced in **bcg_min** version 2.2. Because such evolution breaks backward compatibility, **bcg_min** issues an error message and stops if the argument following option **-class** is either an option (i.e., starts with a hyphen) or a file name with extension **.bcg**, which prevents overwriting an existing BCG file.

Note: Options **-strong**, **-branching**, and **-divbranching** are mutually exclusive. If they occur simultaneously on the command-line, the option occurring last is selected.

Note: Options **-normal**, **-prob**, and **-rate** (with or without **-self** sub-option) are mutually exclusive. If they occur simultaneously on the command-line, the option occurring last is selected.

ENVIRONMENT VARIABLES

See the **bcg**(LOCAL) manual page for a description of the environment variables used by all the BCG application tools.

EXIT STATUS

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

AUTHORS

Version 1.* of **bcg_min** was developed by Damien Bergamini (INRIA/VASY), Moez Cherif (INRIA/VASY), Hubert Garavel (INRIA/VASY) and Holger Hermanns (University of Twente). Pepijn Crouzen added **-self** sub-option. Version 1.* of **bcg_min** used the following algorithms:

- It used the algorithm of [KS90] to compute strong bisimulation on a normal LTS.
- It used the algorithm of [HS99] to compute strong bisimulation on probabilistic and stochastic LTS.
- It used the algorithm of [GV90] to compute branching bisimulation on a normal LTS. The implementation in **bcg_min** was derived from a Pascal program written by Jan Friso Groote (CWI).
- It used a variant of the algorithm of [HS99] to compute branching bisimulation on a stochastic (resp. probabilistic) LTS: the branching bisimulation condition was applied only to the "normal" fragment of the transition relation, while the stochastic (resp. probabilistic) fragments were mimimized with respect to strong bisimulation.

Version 2.* of **bcg_min** was developed by Frederic Lang (INRIA/VASY). It uses (sequential) variants of the signature-based algorithm of [BO03] to compute strong, branching, and divergence-sensitive vranching bisimulation on normal, probabilistic and stochastic LTS.

OPERANDS

input.**bcg**	BCG graph (input)
output.**bcg**	BCG graph (output)
input@**1.o**	dynamic library (input or output)

FILES

$CADP/bin.'arch'/bcg_min	"**bcg_min**" binary program

See the **bcg**(LOCAL) manual page for a description of the other files.

SEE ALSO

bcg(LOCAL), **sprintf**(3C)

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 Hubert.Garavel@inria.fr.

ANNEX 1 - PROBABILISTIC MODELS

The probabilistic LTS model used in **bcg_min** is general enough to support the following models, which can be considered as special cases of probabilistic LTS:

Discrete Time Markov Chains [Nor97]
The graph contains transitions of the form "**`prob %p`**" only.

Discrete Time Markov Reward Models [How71]
The graph contains transitions of the form "**`prob %p`**" to represent transitions not obtaining an impulse reward. State rewards are associated to states by "normal" transitions with source and target states being identical. The label indicates the state reward such as "**`reward 5`**". Impulse rewards are associated to probabilistic transitions by prefixing the "**`prob %p`**" label with a label indicating the reward obtained by taking this transition, as in "**`impulse 4; prob %p`**".

Alternating Probabilistic LTS [Han91]
The graph contains transitions of the form "**`prob %p`**", as well as normal transitions in such a way that there is no state possessing both normal as well as "**`prob %p`**" transitions.

Discrete Time Markov Decision Processes [Put94]
The graph contains transitions of the form "**`prob %p`**", as well as normal transitions in such a way that there is no state possessing both normal as well as "**`prob %p`**" transitions. Normal and probabilistic transitions strictly alternate, i.e. normal (resp. "**`prob %p`**") transitions are not directly followed by normal (resp. "**`prob %p`**") transitions. Uses an encoding of Discrete Time Markov Decision Processes into strictly Alternating Probabilistic LTS proposed in [Arg00].

Generative Probabilistic LTS [GSS95]
The graph contains transitions of the type "*label*`; prob %p`" only, and for each state the sum of "**`%p`**" values leaving a state is equal to (or smaller than) 1.

Reactive Probabilistic LTS [GSS95]

The graph contains transitions of the type "*label*`; prob %p`" only, and for each state and each "*label*" the sum of "`%p`" values leaving a state is equal to (or smaller than) 1.

Stratified Probabilistic LTS [GSS95]

The graph contains transitions of the form "`prob %p`", as well as normal transitions in such a way that there is no state possessing both normal as well as "`prob %p`" transitions. Normal transitions are not directly followed by normal transitions.

ANNEX 2 - STOCHASTIC MODELS

The stochastic LTS model used in **bcg_min** is general enough to support the following models, which can be considered as special cases of stochastic LTS:

Continuous Time Markov Chains [Nor97]

The graph contains transitions of the form "`rate %f`" only.

Continuous Time Markov Reward Models [How71]

The graph contains transitions of the form "`rate %f`" to represent transitions not obtaining an impulse reward. State rewards are assigned to states by "normal" transitions with source and target states being identical. The label indicates the state reward such as "`reward 5`". Impulse rewards are assigned to probabilistic transitions by prefixing the "`rate %f`" label with a label indicating the reward obtained, as in "`impulse 4; rate %f`".

Continuous Time Markov Decision Processes [Put94]

The graph contains transitions of the form "`rate %f`", as well as normal transitions in such a way that there is no state possessing both normal as well as "`rate %f`" transitions. Normal and stochastic transitions strictly alternate, meaning that normal (resp. "`rate %f`") transitions are not directly followed by normal (resp. "`rate %f`") transitions. Inspired by an encoding proposed in [Arg00].

Interactive Markov Chains [Her98]

The graph contains transitions of the form "`rate %f`", as well as normal transitions.

Timed Processes for Performance (TIPP) Models [HHM98]

The graph contains transitions of the form "*label*`; rate %f`", as well as normal transitions.

Performance Evaluation Process Algebra (PEPA) Models [Hil96]

The graph contains transitions of the form "*label*`; rate %f`" only. Passsive transitions are represented by abuse of the "`rate`" keyword. The transition label of a passive transition is augmented by a distinguishing string to indicate that the rate value has to be interpreted as a weight, such as in "**THIS IS A PASSIVE TRANSITION LABELLED** *label* **WITH WEIGHT**`; rate %f`". The actual distinguishing string used for this purpose is of no importance for **bcg_min**, but it must not contain "`;`" and must not start with the keyword "`rate`".

Extended Markovian Process Algebra (EMPA) Models [BG98]

The graph contains transitions of the form "*label*`; rate %f`" only. Passive and immediate transitions are represented by abuse of the "`rate`" keyword. The transition label of a passive transition is augmented by a distinguishing string to indicate that the rate value has to be interpreted as a weight, such as in "**THIS IS A PASSIVE TRANSITION LABELLED** *label* **WITH PRIORITY** *p* **AND WEIGHT**`; rate %f`". The transition label of an immediate transition is augmented in a similar way, as in "**THIS IS AN IMMEDIATE TRANSITION LABELLED** *label* **WITH PRIORITY** *p* **AND WEIGHT**`; rate %f`". The actual distinguishing strings used for these purposes are of no importance for **bcg_min**, but they must not contain "`;`" and must not start with the keyword "`rate`".

BIBLIOGRAPHY

[Arg00] P. R. D'Argenio (2000). On the relation among different probabilistic transition systems and probabilistic bisimulations. CTIT Tech Report, to appear 2000.

[BG98] M. Bernardo and R. Gorrieri (1998). A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time. Theoretical Computer Science 202, pp. 1-54, 1998.

[BHKW05] C. Baier, H. Hermanns, J.P. Katoen, V. Wolf. Comparative branching-time semantics for Markov chains. Information and Computation, vol. 200(2), pp. 149-214, 2005.

[BO03] S. Blom, S. Orzan. Distributed State Space Minimization. Electronic Notes in Theoretical Computer Science, vol. 80, 2003.

[Bra02] M. Bravetti. Revisiting Interactive Markov Chains. Electronic Notes on Theoretical Computer Science, vol. 68(5), 2002.

[GH02] H. Garavel and H. Hermanns (2002). On Combining Functional Verification and Performance Evaluation using CADP. Proc. 11th Int. Symp. of Formal Methods Europe FME'2002 (Copenhagen, Denmark), LNCS 2391, July 2002. Available from http://cadp.inria.fr/publications/Garavel-Hermanns-02.html

[GSS95] R. J. van Glabbeek, S. A. Smolka, and B. Steffen (1995). Reactive, generative, and stratified models of probabilistic processes. Information and Computation 121, pp. 59-80, 1995.

[GW96] R.J. van Glabbeek and W.P. Weijland (1996). Branching Time and Abstraction in Bisimulation Semantics. Journal of the ACM 43(3):555-600.

[GV90] J. F. Groote and F. Vaandrager (1990). An efficient algorithm for branching bisimulation and stuttering equivalence. Proceedings of the 17th ICALP (Warwick), LNCS 443, pp. 626-638, 1990.

[Han91] H. A. Hansson (1991). Time and Probability in Formal Design of Distributed Systems. PhD thesis, University of Uppsala, 1991.

[Her98] H. Hermanns (1998). Interactive Markov Chains. Ph.D. Thesis, University of Erlangen-Nuernberg, Germany, 1998.

[HHM98] H. Hermanns, U. Herzog, and V. Mertsiotakis (1998). Stochastic Process Algebras - Between LOTOS and Markov Chains. Computer Networks and ISDN Systems 30, pp. 901-924, 1998.

[Hil96] J. Hillston (1996). A Compositional Approach to Performance Modelling. Cambridge University Press, 1996.

[How71] R. A. Howard (1971). Dynamic Probabilistic Systems, Vol II: Semi-Markov and Decision Processes. Wiley, 1971.

[HS99] H. Hermanns and M. Siegle (1999). Bisimulation algorithms for stochastic process algebras and their BDD-based implementation. Proceedings of the 5th ARTS, LNCS 1601, pp. 244-265, 1999.

[KS76] J. G. Kemeny and J. L. Snell (1976). Finite Markov Chains. Springer, 1976.

[KS90] P. C. Kanellakis and S. A. Smolka (1990). CCS expressions, finite state processes, and three problems of equivalence. Information and Computation 86(1), pp. 43-68, May 1990.

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

[Nor97] J. R. Norris (1997). Markov Chains. Cambridge University Press, 1997.

[Put94] M. L. Puterman (1994). Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York, NY, 1994.