1 Introduction

This document describes the algorithm for branching bisimulation reduction, including the algorithm that is used to generate counterexamples.

2 A partitioner for branching bisimulation

The partitioner for branching bisimulation calculates whether states are bisimilar, branching bisimilar and stuttering preserving branching bisimilar. It gets a state space and divides it into a number of non-intersecting subsets of states called blocks. All states in a block are bisimilar and have the same block index. Using this index it is straightforward to calculate whether two states are equivalent (they have the same index) or to construct the state space modulo this equivalence.

The algorithm works exactly as described in [2]. As a preprocessing step for (divergence preserving) branching bisimulation all states that are strongly connected via internal transitions are replaced by a single state. In case of divergence preserving branching bisimulation this state gets a tau loop. In case of ordinary branching bisimulation, there will not be a tau loop.

Then the algorithm for branching bisimulation is started. The state space is partitioned into blocks. Initially, all states are put in one block. Repeatedly, a block is split in two blocks until the partitioning has become stable. For details see [2].

Furthermore, there is an option to obtain counter traces or counter formulas for two non bisimilar state. The algorithm for this is inspired by [1, 3].

Given two non bisimilar states $s, t \in S$, where S is the set of all states. A *distinguishing formula* is a formula ϕ in Hennessy-Milner logic such that $s \models \phi$ and $t \not\models \phi$. A *counter trace* is a trace σa such that $s \xrightarrow{\sigma} s' \xrightarrow{a}$ for some state s', and $t \xrightarrow{\sigma} t' \not\xrightarrow{a}$ for some t', or vice versa. For two states there is always a distinghuishing formula iff these states are not bisimilar. For counter traces, the situation is not as straightforward. There are bisimilar states for which counter traces exist, e.g. $a{\cdot}b + a{\cdot}c$ has counter traces $a\,b$ and $a\,c$ for its initial state. Also, the set of all counter traces for non bisimilar states can be the same as the set of counter traces between two non bisimilar processes. E.g. comparing $a{\cdot}b + a{\cdot}c$ with itself yields the same countertraces as comparing it with $a{\cdot}b + a{\cdot}c + a{\cdot}(b + c)$. Yet, countertraces might be useful, as they provide some hint as why two states may not be bisimilar in situations where Hennessy-Milner formulas cannot be used.

Following [1, 3] we extend the algorithm in [2] as follows. Each B block that is split is annotated with its parents block B' and with the pair $\langle a, B'' \rangle$ that was used to split it.

Given two non-bisimilar states s and t first the smallest block is found that contains both s and t. Initially, it is known which stable block s and t are in. By traversing the parent relation of the blocks, the common block B_C is found (in time proportional to the length of the parent relation, which is bounded by the number of blocks, which again is bounded by the number of states). Block B_C is split by $\langle a, B'' \rangle$ into a block B_s containing s and a different block B_t containing t.

Now construct four sets of states:

$$
\begin{aligned}
B^s_{reacha} &= \{s' | s \xrightarrow{a} s' \text{ and } s' \in B''\} \\
B^s_{nonreacha} &= \{s' | s \xrightarrow{a} s' \text{ and } s' \in B''\} \\
B^t_{reacha} &= \{t' | t \xrightarrow{a} t' \text{ and } t' \in B''\} \\
B^t_{nonreacha} &= \{t' | t \xrightarrow{a} t' \text{ and } t' \in B''\}
\end{aligned}
$$

As B'' was a splitter for B_C, exactly one of B^s_{reacha} and B^t_{reacha} is empty, while the other is not empty. Assume without loss of generality that B^s_{reacha} is not empty. If $B^t_{nonreacha}$ is empty, the counter trace is a. If $B^t_{nonreacha}$ is not empty construct a set of counter traces σb for each pair of state $\langle u, u' \rangle \in B^s_{reacha} \times B^t_{nonreacha}$. The counter traces for $\langle s, t \rangle$ are the traces $a\sigma b$ obtained in this way.

References

[1] R. Cleaveland. On automatically explaining bisimulation inequivalence. In E.M. Clarke and R.P. Kurshan, editiors, *Computer Aided Verification (CAV'90)*, volume 531 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 364–372, 1990.

[2] J.F. Groote and F.W. Vaandrager. An efficient algorithm for branching bisimulation and stuttering equivalence. In M.S. Paterson, editor, Proceedings 17th ICALP, Warwick, volume 443 of Lecture Notes in Computer Science, pages 626-638. Springer-Verlag, 1990.

[3] H. Korver. Computing distinguishing formulas for branching bisimulation. In K.G. Larsen and A. Skou, editors, *Computer Aided Verification (CAV'91)*, volume 575 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 13–23, 1991.