On the Minimisation of Acyclic Models*

Pepijn Crouzen[1], Holger Hermanns[1,2], and Lijun Zhang[1]

[1] Universität des Saarlandes, Saarbrücken, Germany
[2] INRIA, VASY, Grenoble Rhône-Alpes, France
{crouzen,hermanns,zhang}@cs.uni-sb.de

Abstract. This paper presents a novel algorithm to compute weak bisimulation quotients for finite acyclic models. It is developed in the setting of interactive Markov chains, a model overarching both labelled transition systems and continuous-time Markov chains. This model has lately been used to give an acyclic compositional semantics to dynamic fault trees, a reliability modelling formalism.

While the theoretical complexity does not change substantially, the algorithm performs very well in practice, almost linear in the size of the input model. We use a number of case studies to show that it is vastly more efficient than the standard bisimulation minimisation algorithms. In particular we show the effectiveness in the analysis of dynamic fault trees.

1 Introduction

Determining the minimum bisimulation quotient of a behavioural model is one of the principal algorithmic challenges in concurrency theory, with concrete applications in many areas. Together with substitutivity properties enjoyed by process algebraic composition operators, bisimulation is at the heart of *compositional aggregation*, one of the most elegant ways to alleviate the state space explosion problem: In compositional aggregation, a model is composed out of sub-models. During generation of its state-space representation, composition and minimisation steps are intertwined along the structure of the compositional specification. This strategy is central to explicit-state verification tools such as μCRL [1] and CADP [12], and arguably a central backbone of their successes in industrial settings [7, 18].

The algorithmic problem to minimise a labelled transition system with respect to bisimulation is well studied. For strong bisimulation, a partition refinement based approach [22] can be used to achieve an algorithm with complexity $\mathcal{O}(m \log n)$, where m and n denote the number of transitions and states of the model. The computation of weak and branching bisimulation is theoretically dominated by the need to compute the transitive closure of internal transitions. This directly determines the overall complexity to be $\mathcal{O}(n^3)$ (disregarding some very specialized algorithms for transitive closure such as [6]). As first noted in [15], the transitive closure computation does *not* dominate in practical applications, and then the complexity is $\mathcal{O}(m^* \log n)$, where m^* is the number of transitions after closure.

* This work is supported by the DFG as part of the Transregional Collaborative Research Center SFB/TR 14 AVACS and by the European Commission under the IST framework 7 project QUASIMODO.

F. van Breugel and M. Chechik (Eds.): CONCUR 2008, LNCS 5201, pp. 295–309, 2008.

Lately, the growing importance of the minimisation problem has triggered work in at least three different directions. Orzan and Blom have devised an efficient *distributed* algorithm for bisimulation minimisation, based on the notion of *signatures* [2]. Wimmer et al. [25] have taken up this idea to arrive at a fully *symbolic* implementation of the signature-refinement approach, to effectively bridge to BDD-based representations of state spaces. In [9], an algorithm with $\mathcal{O}(m)$ complexity has been proposed for deciding strong bisimulation on directed acyclic graphs. Mateescu [20] developed an $\mathcal{O}(m)$ algorithm for checking modal mu calculus on acyclic LTS, which can be instantiated to checking weak bisimulation, then requiring $\mathcal{O}(m^2)$.

Stochastic behavioural models are among the most prominent application areas for bisimulation minimisation and compositional aggregation techniques [3, 17]. They are used to model and study 'quantified uncertainty" in many areas, such as embedded, networked, randomized, and biological systems. Interactive Markov chains (IMCs) [16] constitute a process algebraic formalism to construct such models. Recently, an extension of IMCs with input and output (IOIMCs) [4] has been introduced to define a rigorous compositional semantics for dynamic fault trees (DFTs). Fault trees and DFTs are in daily use in industrial dependability engineering [10, 24]. The analysis of DFTs via their IOIMC semantics relies heavily on compositional aggregation and weak bisimulation minimisation [4]. Remarkably, the IOIMC semantics maps on acyclic structures. This is the main motivation for the work presented in this paper. We show how to effectively exploit acyclicity of the model in weak bisimulation minimisation. Since (IO)IMCs are a strict superset of LTSs, our results apply to LTSs as well.

The problem of weak bisimulation minimisation on acyclic models is substantially different from the strong bisimulation problem. While not directly developed for LTSs, the rough idea of [9] is to assign to each state a rank which corresponds to the length of the longest path from the state to one of the absorbing states. Observing that *(i)* transitions always move from higher rank states to lower rank states, and *(ii)* only states on the same rank can be bisimilar. allows one to arrive at a linear algorithm. Especially condition *(ii)* is invalid in the weak setting. To overcome this, we use elaborated rank-based techniques to partition the state space on-the-fly during the computation of the weak bisimulation quotient. The resulting algorithm is of linear time complexity in m^*, the size of the weak transition relation. We provide experimental evidence that in practice, the algorithm runs even in linear time in the size of the original relation m. The contributions are developed in the setting of IMCs.

Organisation. The paper is organised as follows. After giving preliminary definitions we discuss in Section 3 how to adapt the strong bisimulation algorithm for acyclic digraphs in [9] to IMCs. Section 4 is devoted to a novel algorithm for weak bisimulation on acyclic IMCs. Section 5 discusses two extensions which allow us to handle the models appearing in DFT analysis. Section 6 presents experimental results after we which we conclude.

Detailed proofs for the theorems in this paper can be found in [8].

2 Preliminaries

In this section we introduce the definition of acyclic interactive Markov chains, strong and weak bisimulations [16].

Definition 1. *An interactive Markov chain (IMC) is a tuple* $\langle S, s^0, A, R_i, R_M \rangle$ *where:* S *is a finite set of states,* $s^0 \in S$ *is the starting state,* A *is a finite set of actions,* $R_i \subseteq S \times A \times S$ *is the set of interactive transitions, and* $R_M \subseteq S \times \mathbb{R}^{>0} \times S$ *is the set of Markovian transitions.*

The label τ is a assumed to be contained in A, it denotes the internal, unobservable action. For $(s, a, t) \in R_i$ we write $s \xrightarrow{a}{}^{i} t$ and for $(s, \lambda, t) \in R_M$ we write $s \xrightarrow{\lambda}{}^{M} t$. Let $R = R_i \cup R_M$. We write $s \xrightarrow{x} s'$ if $(s, x, s') \in R$. In this case x is either an action or a Markovian rate. We write $s \rightarrow^{i} t$ for any interactive transition from s to t and $s \rightarrow^{M} t$ for any such Markovian transition.

States with outgoing τ-transitions are called *unstable*. States without outgoing τ-transitions are called *stable*. We write the reflexive and transitive closure of all internal transitions $s \xrightarrow{\tau}{}^{i} s'$ as $\stackrel{\tau}{\Longrightarrow}$ and say that if $s \stackrel{\tau}{\Longrightarrow} s'$ then s may move internally to state s'. For $s \stackrel{\tau}{\Longrightarrow} s' \xrightarrow{a}{}^{i} s'' \stackrel{\tau}{\Longrightarrow} s'''$ we write $s \stackrel{a}{\Longrightarrow}{}^{i} s'''$. For $s \stackrel{\tau}{\Longrightarrow} s' \xrightarrow{\lambda}{}^{M} s'' \stackrel{\tau}{\Longrightarrow} s'''$ we write $s \stackrel{\lambda}{\Longrightarrow}{}^{M} s'''$. For $s \stackrel{\tau}{\Longrightarrow} s' \xrightarrow{x} s'' \stackrel{\tau}{\Longrightarrow} s'''$ we write $s \stackrel{x}{\Longrightarrow} s'''$.

The cumulative rate from a state s to a set of states C, denoted $\gamma_M(s, C)$, is the sum of the rates of all Markovian transitions from s to states in C: $\gamma_M(s, C) = \sum\{|\lambda \mid (s, \lambda, t) \in R_M \wedge t \in C|\}$, where $\{|\ldots|\}$ denotes a multi-set. The internal backwards closure of a set of states C, denoted C^τ is the set of all states that can reach a state in C via zero or more τ-transitions: $C^\tau = \{s \mid s \stackrel{\tau}{\Longrightarrow} t \wedge t \in C\}$.

A (finite) path π is a sequence $\pi = s_0 x_0 s_1 x_1 \ldots s_n$ in $(S \times (A \cup \mathbb{R}^{>0}))^* \times S$ such that $s_i \xrightarrow{x_i} s_{i+1}$ for all $i = 0, 1, \ldots, n-1$. For a path π we let $first(\pi)$ denote the first state s_0 of π, $last(\pi)$ denote the last state of a finite π, $\pi[i]$ denote the $i+1$-th state s_i of π, $\pi_\alpha[i]$ denote the $i+1$-th label x_i of π, and $len(\pi)$ denote the length n of π. Moreover, we let $wlen(\pi) = \{|\pi_\alpha[i] \mid i = 0, \ldots, len(\pi) - 1 \wedge \pi_\alpha[i] \neq \tau|\}$ denote the weak length of π, which corresponds to the number of observable actions of π. We write $s \overset{\pi}{\rightsquigarrow} s'$ if $first(\pi) = s$ and $last(\pi) = s'$. We write the set of all paths starting from a state s as $Paths(s) = \{\pi \mid \exists s' \in S \cdot s \overset{\pi}{\rightsquigarrow} s'\}$. A path π such that $s \overset{\pi}{\rightsquigarrow} s$ and $len(\pi) > 0$ is called a *cycle*.

The *maximal progress* assumption is usually employed when working with IMCs. It states that if an unobservable (τ) transition is possible in a state, no time may advance prior to taking this (or any other action) transition. In other words, in an unstable state the chance of taking a Markovian transition is given by the probability that the delay associated with the transition is less than or equal to 0. However, this probability is 0, and thus we qualify such a transition as not *plausible*. Semantically, their existence is negligible, which will become apparent in the definition of strong and weak bisimulation. On the other hand, all interactive transitions and all Markovian transitions from stable states are plausible. A path $\pi = s_0 x_0 \ldots s_n$ is plausible if it holds that: for all $i = 0, \ldots, n-1$, if $s_i \xrightarrow{x_i}{}^{M} s_{i+1}$ then s_i is stable. We write the set of all plausible paths starting from a state s as $Paths^P(s)$. A plausible path π from s to t is denoted $s \overset{\pi}{\rightsquigarrow}{}^{P} t$.

Definition 2. *An IMC* $\mathcal{P} = \langle S, s^0, A, R_i, R_M \rangle$ *is* acyclic *if it does not contain any plausible path* π *with* $s \overset{\pi}{\rightsquigarrow}^P s$ *and* $len(\pi) > 0$ *for any* $s \in S$.

An acyclic IMC only contains finite plausible paths since the set of states is by definition finite. We recall the definition of strong and weak bisimulations.

Definition 3. *Let* $\mathcal{P} = \langle S, s^0, A, R_i, R_M \rangle$ *be an IMC. An equivalence relation* $\mathcal{E}$ *on* S *is a strong bisimulation if and only if* $s\mathcal{E}t$ *implies for all* $a \in A$ *and all equivalence classes* C *of* $\mathcal{E}$*, that* $s \xrightarrow{a}{}^i s'$ *implies* $t \xrightarrow{a}{}^i t'$ *with* $s'\mathcal{E}t'$*, and* s *stable implies* $\gamma_M(s, C) = \gamma_M(t, C)$.

Two states s, t *of* $\mathcal{P}$ *are strongly bisimilar, written* $s \sim t$ *if there exists a strong bisimulation* $\mathcal{E}$ *such that* $s\mathcal{E}t$.

Definition 4. *Let* $\mathcal{P} = \langle S, s^0, A, R_i, R_M \rangle$ *be an IMC. An equivalence relation* $\mathcal{E}$ *on* S *is a weak bisimulation if and only if* $s\mathcal{E}t$ *implies for all* $a \in A$ *(including* τ*) and all equivalence classes* C *of* $\mathcal{E}$*, that* $s \overset{a}{\Longrightarrow}{}^i s'$ *implies* $t \overset{a}{\Longrightarrow}{}^i t'$ *with* $s'\mathcal{E}t'$*, and* $s \overset{\tau}{\Longrightarrow} s'$ *and* s' *stable imply* $t \overset{\tau}{\Longrightarrow} t'$ *for some stable* t' *and* $\gamma_M(s', C^\tau) = \gamma_M(t', C^\tau)$.

Two states s, t *of* $\mathcal{P}$ *are weakly bisimilar, written* $s \approx t$ *if there exists a weak bisimulation* $\mathcal{E}$ *such that* $s \approx t$.

Strong, respectively weak bisimilarity is the largest strong, respectively weak bisimulation [16]. For an IMC $\langle S, s^0, A, R_i, \emptyset \rangle$ the above definitions reduce to Milner's standard definitions on labelled transition systems [21].

3 Strong Bisimulation for Acyclic IMCs

To decide strong bisimulation on unlabelled acyclic digraphs, a linear-time algorithm has been developed in [9], which is based on state ranking. To handle labelled transition systems, the authors encode them into unlabeled graphs, for which strong bisimulation can then be decided. In this section, we extend their rank-based algorithm in [9] to decide strong bisimulation directly for IMCs.

We adapt the notion of ranks for acyclic IMCs. The rank of absorbing states is 0, for other states it is the longest distance to a state on rank 0. So the rank of a state is the length of the longest path starting in that state.

Definition 5. *The rank function* $\mathsf{R} : S \to \mathbb{N}$ *is defined by:* $\mathsf{R}(s) = \max\{len(\pi) \mid \pi \in Paths^P(s)\}$.

Since in acyclic IMCs all plausible paths are finite, we have that $\mathsf{R}(s) < \infty$. If state s has a higher rank than state t, we say also that s is higher than t. By definition, transitions always go from higher states to lower states. Before continuing, we state an important observation about the relationship between strong bisimilarity and ranks.

Theorem 1. *If two states of* $\mathcal{P}$ *are strongly bisimilar they are on the same rank:* $\forall s, t \in S \cdot s \sim t \to \mathsf{R}(s) = \mathsf{R}(t)$.

Algorithm 1. Determining the strong bisimilarity quotient for an acyclic IMC

Require: $\mathcal{P} = \langle S, s^0, A, R_i, R_M \rangle$ is an acyclic IMC.

1: $\mathsf{R} = \textsc{ComputeRanks}()$
2: $maxrank = max\{n \mid s \in S \wedge \mathsf{R}(s) = n\}$
3: $BLOCKS = \langle \{\{s \mid s \in S \wedge \mathsf{R}(s) = n\}\} \mid n \leftarrow \langle 0 \ldots maxrank \rangle \rangle$
4: $matrix = \underline{0}$; $rate = \underline{0}$
5: **for** $i = 0$ to $maxrank - 1$ **do**
6: **for all** $(s, a, B) \in \{(s, a, B) \mid B \in BLOCKS[i] \wedge \exists t \in B \cdot (s, a, t) \in R_i\}$ **do**
7: $matrix[s][a][B] = 1$
8: **for all** $(s, \lambda, B) \in \{|(s, \lambda, B) \mid B \in BLOCKS[i] \wedge \exists t \in B \cdot (s, \lambda, t) \in R_M|\}$ **do**
9: **if** s stable **then**
10: $rate[s][B] = rate[s][B] + \lambda$
11: **for** $j = i + 1$ to $maxrank$ **do**
12: $BLOCKS[j] = \bigcup\{\{\{t \mid t \in B \wedge matrix[t] = matrix[s] \wedge rate[t] = rate[s]\} \mid s \in B\} \mid B \in BLOCKS[j]\}$

The above theorem mirrors Proposition 4.2 in [9] and implies that only states on the same rank can be bisimilar. Since transitions go from higher states to lower states, whether two states on the same rank are bisimilar depends only on the states below them. The main idea of the algorithm is to start with the states on rank 0 and to process states rank by rank in a backward manner. The algorithm is presented in Algorithm 1. The input is an acyclic IMC. Lists (and tuples) are written by: $\langle ... \rangle$. The state ranks are computed in line 1 with a simple depth first search (time-complexity $\mathcal{O}(m)$). The partition $BLOCKS$ is initialised such that states with the same rank are grouped together. During the algorithm, we use $BLOCKS[i]$ to denote the partition of the states with rank i. The matrices $matrix$ and $rate$ respectively denote the interactive transitions and the cumulative rates from states to blocks of bisimilar states. The algorithm starts with the rank 0 states which are all strongly bisimilar. Then, it traverses the transitions backwards to refine blocks on the higher level according to the bisimulation definition. Observe that in iteration i all states with ranks lower than i have been processed. Since each transition is visited at most once, the algorithm runs in linear time.

Theorem 2. *Given an acyclic IMC $\mathcal{P}$, Algorithm 1 computes the strong bisimulation correctly. Moreover, the time-complexity of the algorithm is $\mathcal{O}(m)$.*

4 Weak Bisimulation Minimisation

Weak bisimulation (or observational equivalence [21]) differs from strong bisimulation in that only visible behavior has to be mimicked (see Definition 4). In general weak bisimulation can be computed by first computing the reflexive transitive closure of internal transitions $\stackrel{\tau}{\Longrightarrow}$ and then computing the *weak* transitions $s \stackrel{a}{\Longrightarrow} t$ from $s \stackrel{a}{\Longrightarrow} t \leftrightarrow s \stackrel{\tau}{\Longrightarrow} s' \stackrel{a}{\rightarrow} t' \stackrel{\tau}{\Longrightarrow} t$. Once we have computed the weak transition relation we can then compute weak bisimulation simply by computing strong bisimulation on the weak transition relation.

If we try this strategy for acyclic models we quickly run into a problem, since the weak transition relation of an acyclic model is not acyclic, it contains cycles $s \stackrel{\tau}{\Longrightarrow} s$ for each state s. Thus we cannot simply apply Algorithm 1 to the weak transition relation.

Of course it is easy to see that these τ self-loops will be the only cycles. A naive approach would then simply remove these self-loops from the weak transition relation and apply Algorithm 1 to this modified weak transition relation. This approach however does not work. Consider IMC $\mathcal{P}$ in Figure 1. It is obvious that states s_1 and s_3 are weakly bisimilar, while the naive approach would decide that they are not, since s_1 can do the weak transition $s_1 \stackrel{\tau}{\Longrightarrow} s_3$, which s_3 seemingly cannot simulate if we do not consider the τ-loop $s_3 \stackrel{\tau}{\Longrightarrow} s_3$. Even if we would memorize that each state has a τ-loop, and treat this case separately in the algorithm, this is not enough, since there is a fundamental difference to the strong case. We find in fact that Theorem 1 does not hold for weak bisimulation! States s_1 and s_3 have different ranks (2 and 1 respectively) but they are still weakly bisimilar. We can however, define a different ranking of states for which a similar theorem does hold.

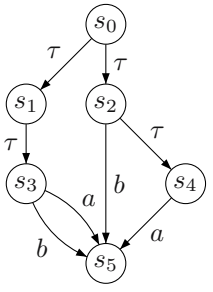

Fig. 1. An acyclic IMC $\mathcal{P}$

4.1 Weak Ranks

Let $\mathcal{P} = \langle S, s^0, A, R_i, R_M \rangle$ be an acyclic IMC. To find the *weak rank* of a state we do not look at the longest path starting in that state but we find the longest path counting only the observable transitions.

Definition 6. *We define the notion of* weak rank *(or the observable rank)* $\mathsf{R}^W : S \to \mathbb{N}$ *of a state* s *as the maximum weak length of all plausible paths starting in* s*:* $\mathsf{R}^W(s) = max\{wlen(\pi) \mid \pi \in Paths^P(s)\}$.

For weak ranks we can establish a theorem similar to Theorem 1.

Theorem 3. *If two states of* $\mathcal{P}$ *are weakly bisimilar they have the same weak rank:* $\forall s, t \in S \cdot s \approx t \to \mathsf{R}^W(s) = \mathsf{R}^W(t)$.

For strong bisimulation and ranks we found the property that strong bisimulation for states on a certain rank only depends on the states below this rank. Unfortunately, this property does not hold for weak ranks. Consider again IMC $\mathcal{P}$ in Figure 1. State s_5 has weak rank 0 whereas all other states have weak rank 1. If we consider only weak transitions to the states on rank 0 we are tempted to conclude that s_0, s_1, s_2 and s_3 are all weakly bisimilar, since they can all do the weak moves $\stackrel{a}{\Longrightarrow} s_5$ and $\stackrel{b}{\Longrightarrow} s_5$, while state s_4 can only do the weak move $\stackrel{a}{\Longrightarrow} s_5$. However state s_0 can also do the move $s_0 \stackrel{\tau}{\Longrightarrow} s_4$ which s_1, for instance, cannot simulate and thus s_0 and s_1 are actually not weakly bisimilar.

4.2 A Different Way of Partitioning the State Space

To make use of the acyclicity of our models in computing weak bisimulation we need to order the state space such that *(i)* weak bisimilarity of states on order x only depends on the bisimilarity of states of a lower order, and *(ii)* all states that are weakly bisimilar have the same order. However, we have seen that the weak rank does not satisfy the first requirement and the rank does not satisfy the second.

We introduce the notion of *level* of a state, which is computed on-the-fly, to order the state space. A level function L maps states to some ordered well-founded set W, such that for state s, level $L(s)$ is the smallest value satisfying the following conditions

1. State s has no outgoing transitions to any other state t on a level higher than $L(s)$: $s \xrightarrow{a} t$ implies $L(s) \geq L(t)$.
2. State s has no outgoing observable transitions to any state t on *level* $L(s)$: $s \xrightarrow{a} t \wedge L(s) = L(t)$ implies $a = \tau$.
3. State s has no outgoing τ-transitions to any state t on *level* $L(s)$, unless s is weakly bisimilar to t: $s \xrightarrow{\tau} t \wedge L(s) = L(t)$ implies $s \approx t$.

Notably, partitioning the state space in such levels satisfies the two requirements *(i)* and *(ii)* above: If a state s has a level higher than t they cannot be weakly bisimilar since s must have a transition to a non-bisimilar state that is at least on the same level as t. Furthermore the bisimilarity of two states on a level i depends only on lower levels since weak transitions to bisimilar states can, by definition, always be simulated. However, if we want to use such a level function to compute the weak bisimulation for an acyclic IMC $\mathcal{P} = \langle S, s^0, A, R_i, R_M \rangle$, we are in a trap, because condition 3 relies on knowledge about the weak bisimulation classes.

Our algorithm exploits that we can – for a particular level – obtain the required bisimulation knowledge by only looking at an IMC that is spanned by lower levels. This allows us to increment our knowledge about both L and $\approx$ while 'climbing up' the transitions, starting with absorbing states. Technically, we are working with a sequence of partial functions $L'_1, \ldots, L'_k$ $(L'_i : S \to W)$ that satisfy (in set notation) $L'_i \subset L'_{i+1}$ and that must converge towards the total function L in finitely many steps. For a given partial function L', and a fixed level $w \in W$, the IMC spanned by level w is defined by restricting the transitions of $\mathcal{P}$ to only those transitions which are part of the weak transition relation to states s with level $L'(s) \leq w$.

Definition 7. *For an acyclic IMC $\mathcal{P}$, a partial function $L' : S \to W$ to an ordered well-founded set, and an element of the ordered set $w \in W$ the* IMC spanned by level w *is $\mathcal{P}_w = \langle S, s^0, A, R_i^w, R_M^w \rangle$, where:*

$$
\begin{aligned}
R_i^w &= \{(s,\tau,t) \mid \exists t' \cdot t \stackrel{x}{\Longrightarrow} t' \wedge L'(t') \leq w \wedge (s,\tau,t) \in R_i\} \\
&\quad \cup \{(s,a,t) \mid \exists t' \cdot t \stackrel{\tau}{\Longrightarrow}{}^{i} t' \wedge L'(t') \leq w \wedge (s,a,t) \in R_i\} \\
R_M^w &= \{(s,\lambda,t) \mid \exists t' \cdot t \stackrel{\tau}{\Longrightarrow}{}^{i} t' \wedge L'(t') \leq w \wedge (s,\lambda,t) \in R_M\}
\end{aligned}
$$

Algorithm 2. Determining the weak bisimilarity quotient for an acyclic IMC

Require: $\mathcal{P} = \langle S, s^0, A, R_i, R_M \rangle$ is an acyclic IMC.

1: $(\mathsf{R}^W, \#wout) = \textsc{ComputeRanks}(\mathcal{P})$
2: $maxwrank = max\{n \mid s \in S \wedge \mathsf{R}^W(s) = n\}$
3: $BLOCKS = \langle\{\{s \mid s \in S \wedge \mathsf{R}^W(s) = n\}\} \mid n \leftarrow \langle 0 \ldots maxwrank \rangle\rangle$
4: $matrix = \underline{0}$; $rate = \underline{0}$
5: **for** $i = 0$ to $maxrank$ **do**
6: $\quad LSTATES = \{s \mid \mathsf{R}^W(s) = i \wedge \#wout(s) = 0\}$
7: $\quad NSTATES = \emptyset$
8: $\quad j = 0$
9: $\quad$ **while** $LSTATES \neq \emptyset$ **do**
10: $\quad\quad \textsc{ComputeLevel}(\mathcal{P}, BLOCKS[i], LSTATES, NSTATES)$
11: $\quad\quad LBLOCKS = \{\{s \mid s \in B \wedge s \in LSTATES\} \mid B \in BLOCKS[i]\}$
12: $\quad\quad$ **for all** $(s, a, B) \in \{(s, a, B) \mid B \in LBLOCKS \wedge \exists t \in B \cdot s \stackrel{a}{\Longrightarrow}^i t\}$ **do**
13: $\quad\quad\quad matrix[s][a][B] = 1$
14: $\quad\quad$ **for all** $(s, \lambda, B) \in \{(s, \lambda, B) \mid B \in LBLOCKS \wedge \exists t \in S \cdot s \stackrel{\tau}{\Longrightarrow}^i t \wedge t$ stable $\wedge$ $\gamma_M(t, B^\tau) = \lambda\}$ **do**
15: $\quad\quad\quad rate[s][B] = \lambda$
16: $\quad\quad$ **for** $k = i$ to $maxrank$ **do**
17: $\quad\quad\quad BLOCKS[k] = \bigcup\{\{\{t \mid t \in B \wedge matrix[t] = matrix[s] \wedge rate[t] = rate[s]\} \mid s \in B\} \mid B \in BLOCKS[k]\}$
18: $\quad\quad LSTATES = \{s \mid s \in NSTATES \wedge \#wout(s) = 0\}$
19: $\quad\quad NSTATES = \emptyset$
20: $\quad\quad j = j + 1$

Our intention is to reformulate conditions 2 and 3 above to the following: if a state s has a level $L'(s) = w$ and w' is the largest value smaller than w appearing in the range of L', then for all transitions $s \xrightarrow{\tau} t$ we find:

1. State t has a level strictly lower than w, or
2. State t has level w and is weakly bisimilar to s on the IMC spanned by level w'.

This property holds indeed for our algorithm, because the sequence $L'_1, \ldots, L'_k$ is constructed level-by-level, starting from the bottom level. The algorithm also implies that if $L'(s)$ is defined, then L' is defined for all states reachable from s as well, which is a requirement for the above idea to work.

Algorithm 2 computes the levels while traversing the state space, and while computing the weak bisimulation relation for an acyclic IMC. Line 1 calculates the weak rank and the number of outgoing weak transitions for every state. Notably, the levels (line 10) and the weak bisimulation relation (lines 12 to 17) are calculated per weak rank. This is justified by Theorem 3. In other words, we fix $W = \mathbb{N} \times \mathbb{N}$ where the first component is the weak rank, and use the (lexicographic) order on pairs $(x, y) \geq (x', y') \iff x > x' \vee (x = x' \wedge y \geq y')$. For each iteration of the loop 9–20 the set *LSTATES* then contains exactly those states which have level (i, j), see Definition 8 below, and the set *LBLOCKS* partitions *LSTATES* into sets of weakly bisimilar states. The set *NSTATES*

Algorithm 3. ComputeLevel($\mathcal{P}, BLOCKS, LSTATES, NSTATES$)

```
1: for all States s in LSTATES do
2:   for all Transitions t -τ-> s ∈ R_i with R^W(t) = R^W(s) do
3:     #wout(t) = #wout(t) − 1
4:     if ¬∃B ∈ BLOCKS · s, t ∈ B then
5:       NSTATES = NSTATES ∪ {t}
6:     else
7:       if #wout(t) = 0 ∧ t ∉ NSTATES then
8:         LSTATES = LSTATES ∪ {t}
```

contains all states with weak rank i, level greater than (i, j) and at least one transition to a state on level (i, j).

Theorem 4. *Given an acyclic IMC $\mathcal{P}$, Algorithm 2 computes the weak bisimulation correctly. Moreover, the time-complexity of the algorithm, given the weak transition relation, is $\mathcal{O}(n^2)$. The space complexity is $\mathcal{O}(n^2)$.*

4.3 Correctness

We give here an extended sketch of the proof of correctness for Algorithm 2. For the full proof we refer to [8]. First we define the notion of the level of a state based on the two conditions given at the end of Subsection 4.2.

Definition 8 (Level (i, j)). *Let $\mathcal{P} = \langle S, s^0, A, R_i, R_M \rangle$ be an acyclic IMC. We define the set of all states in $\mathcal{P}$ with* level (i, j)*, written $\mathcal{L}_{(i,j)}$ as the largest set for which the following holds:*

$$s \in \mathcal{L}_{(i,j)} \rightarrow \mathsf{R}^W(s) = i \wedge \neg\exists j' < j \cdot s \in \mathcal{L}_{(i,j')} \wedge \forall s \xrightarrow{x} t \cdot L(t) < (i, j) \vee (t \in \mathcal{L}_{(i,j)} \wedge s \approx t)$$

We write $L(s) = (i, j)$ if and only if $s \in \mathcal{L}_{(i,j)}$.

To prove that Algorithm 2 is correct we prove that it computes in each iteration (i, j) of the loop 9–20, the set of states on level (i, j) *LSTATES* (line 10) and weak bisimulation on the IMC spanned by level (i, j) *BLOCKS* (lines 16–17). By computing the set of states on level (i, j) we also further refine the partial function L' in each iteration, which is initially completely undefined. By iteration (i, j) we mean that pass of loop 9–20 where variables i and j have those particular values. Line 1 computes, for each state, its weak rank and the number of outgoing transitions to states on the same weak rank ($\#wout(s)$). For each weak rank i the loop 9–20 terminates having computed the weak bisimulation on the IMC spanned by the maximum possible level for weak rank i, denoted (i, max_i). The algorithm then terminates after computing weak bisimulation for the IMC spanned by the maximal level for the maximal weak rank which is equivalent to the IMC itself.

First we consider the computation of weak bisimulation on IMCs spanned by the different levels. Line 3 initializes *BLOCKS* such that all states are partitioned

according to weak rank, this is justified by Theorem 3. For level $(0,0)$ weak bisimulation on $\mathcal{P}_{(0,0)}$ is computed in lines 12–19. For a level $(i,0) > (0,0)$ we compute weak bisimulation on $\mathcal{P}_{(i,0)}$ by refining weak bisimulation on $\mathcal{P}_{(i-1,\mathsf{max}_{i-1})}$. We do this by considering all the weak transitions to states on level $(i,0)$ in lines 12–15[1], note that we compute the set of states on level $(i,0)$ in that same iteration on line 10. Here it is assumed that the partition of states on level $(i,0)$, *LBLOCKS*, is the partition according to weak bisimilarity on $\mathcal{P}$. This is correct since the only outgoing weak transitions for states on level $(i,0)$ we have not yet considered are those that go to other states on level $(i,0)$. But, by Definition 8, such transitions are τ transitions which go to bisimilar states and such transitions can always be simulated. This then means that in line 17 weak bisimulation on $\mathcal{P}_{(i,0)}$ is computed. The same holds for iteration (i,j) with $j > 0$ where weak bisimulation on $\mathcal{P}_{(i,j-1)}$ is refined to weak bisimulation on $\mathcal{P}_{(i,j)}$.

Now we consider the computation of levels. For every weak rank i line 6 initializes *LSTATES* to all states on weak rank i with only transitions to states levels lower than $(i,0)$. By definition these states must have level $(i,0)$. Line 7 initializes *NSTATES* to $\emptyset$. The function ComputeLevel then considers all τ-transitions to states in *LSTATES*. If a state is found which has a τ-transition to a non-bisimilar state - with respect to transitions to states on levels below $(i,0)$, and note that we have computed this relation in the previous iteration - on level $(i,0)$ then we add this state to *NSTATES* since we are sure it has level higher than $(i,0)$. If the condition of line 7 of ComputeLevel is met for a state t then all outgoing transitions of t must go to lower levels or to bisimilar states on level $(i,0)$ which means that t also has level $(i,0)$. When ComputeLevel terminates we must have found all states on level $(i,0)$ because the model is acyclic. Now *NSTATES* contains all states with at least one transition to level $(i,0)$. For those states s in *NSTATES* which now have $\#wout(s) = 0$ (line 18 of Algorithm 2) we know that they only have transitions to states lower than $(i,1)$ and thus they must have level $(i,1)$. In this way we compute all the levels recursively. For each weak rank loop 9–20 must terminate since there are only a finite number of states on that weak rank. Acyclicity also ensures that we must encounter and assign a level to all states in the function ComputeLevel.

This shows that in each iteration (i,j) of loop 9–20 the states on level (i,j) and $\mathcal{P}_{(i,j)}$ are computed. Finally then weak bisimulation on $\mathcal{P}$ spanned by the maximum level (i_m, j_m) is computed. Since all states must have a level smaller than or equal to the maximum we find that $\mathcal{P}_{(i_m,j_m)} = \mathcal{P}$.

4.4 Complexity

We first discuss the complexity of the algorithm itself. Afterwards we discuss the time needed to compute the weak transition relation. In the following n, m and l are used the denote the number of states ($|S|$), transitions ($|R_i| + |R_M|$) and the number of actions ($|A|$) respectively.

[1] We only consider the weak transition relation to states on level $(i,0)$ while $\mathcal{P}_{(i,0)}$ also contains transitions above this level. However, we will consider these transitions in later iterations of the algorithm.

The ranks are computed with a simple depth-first search which can be done in time $\mathcal{O}(m)$. The level computations are also done in time $\mathcal{O}(m)$ as each state is evaluated exactly once in Algorithm 3 and in each such evaluation all incoming transitions are considered once. For the partitioning of the state space in weakly bisimilar blocks every state is considered exactly once, since we only consider states in blocks that we do not have to partition anymore. For each state we must consider each incoming weak move in the weak transition relation. The time complexity is then in the order of the size of the weak transition relation which is $\mathcal{O}(n^2)$. There can be at most n partitions (in the case that there are no bisimilar states at all) so we must make at most n partitions which then takes $\mathcal{O}(n)$ time.

There are at most $\mathcal{O}(ln^2)$ transitions in an acyclic IMC. If we consider the number of actions to be constant then m will be in $\mathcal{O}(n^2)$ which proves that Algorithm 2 computes the weak bisimulation quotient for an acyclic IMC in time $\mathcal{O}(n^2)$ given that we know the weak transition relation. For the space complexity we find that we need to store several attributes of the states, but most importantly we must store the weak transition relation which is again of size $O(n^2)$.

However computing the weak transition relation theoretically dominates the rest of the algorithm in the general case as well as the acyclic case. We will see in Section 6 that this is usually not the case in practice, as also noted in [15]. In the general case the best theoretical complexity for computing the reflexive transitive closure is $\mathcal{O}(n^{2.376})$ as given by [6]. For acyclic graphs [23] gives an algorithm which computes the reflexive transitive closure in time $\mathcal{O}(mk)$ where k is the size of the chain decomposition of the model (which is at most n). In our implementation we have adapted the simple algorithm in [13] which has worst-case time complexity $\mathcal{O}(mn)$. Theoretically the algorithm is then still cubic in the number of states, but we will discuss why this is often not the case in practice.

For acyclic models m^*, the number of transitions in the weak transition relation is at most $\mathcal{O}(ln^2)$. For some state s, m_s^*, the number of outgoing weak transitions of s will be at most nl. As in [13] we compute $\Longrightarrow$ by starting at the states with rank 0 and then moving up the ranks. This means that for any state s we will compute $out^*(s)$ (the outgoing weak moves of state s) by merging $out^*(t')$ for all transitions $s \rightarrow t'$. This is possible since the state t' must have a lower rank than s. We will then find every move $s \overset{a}{\Longrightarrow} t$ at most once for each outgoing transition. This means that the complexity is $\mathcal{O}(nm^*) = \mathcal{O}(n^3)$. However, since we have different action labels, the situation is more nuanced. We can in fact only find the move $s \overset{a}{\Longrightarrow} t$ for outgoing τ- or a-transitions. So we will find every move $s \overset{a}{\Longrightarrow} t$ exactly $m_{s,\tau} + m_{s,a}$ times, where $m_{s,\tau}$ is the number of outgoing τ-transitions of state s and $m_{s,a}$ the number of outgoing a-transitions. For a state s and an action a we can find the, at most n, a-moves in m_s^* at most $m_{s,\tau} + m_{s,a}$ times. If we now sum over all actions and all states we find:

$$\sum_{a}^{a \in A \setminus \{\tau\}} \sum_{s}^{s \in S} nm_{s,\tau} + nm_{s,a} = \sum_{a}^{a \in A \setminus \{\tau\}} nm_\tau + nm_a = lnm_\tau + nm_{A \setminus \{\tau\}}$$

In the worst case $lnm_\tau + nm_{A\setminus\{\tau\}}$ is still cubic in the number of states. However, we can make an interesting observation about models which are observably deterministic, i.e. each state has at most one outgoing transition for each visible action. We then find that $m_{A\setminus\{\tau\}}$ is at most ln and, if we then assume the number of actions is constant we find complexity $\mathcal{O}(nm_\tau + n^2)$. The models used in the dynamic fault tree domain [4] are all observably deterministic.

5 Extensions to the Algorithm

The algorithm presented in this paper was developed with a particular application in mind, namely dynamic fault tree analysis. We have therefore extended the algorithm to compute the desired equivalence relation, weak Markovian bisimulation, for the desired formalism, IOIMCs.

Weak Markovian bisimulation. Weak Markovian bisimulation for IMCs (introduced as weak bisimulation in [5], refereed to here as weak Markovian bisimulation to avoid confusion) differs from weak bisimulation in that Markovian transitions to otherwise bisimilar states are not observable. This bisimulation relation is justified by the fact that IMCs are memoryless in time [16]. A Markovian transition from state s to state t means that we may move from s to t after some time. If we find, however, that the behavior of s and t are the same then this extra time span does not influence the behavior, since the memoryless property ensures that the behavior of t does not depend on the time we arrive in state t.

Adapting our algorithm to weak Markovian bisimulation is very simple. All that needs to be changed is that in the adapted algorithm Markovian transitions are considered unobservable, like τ transitions. This only has a direct effect on the definition, and computation, of weak ranks (see Definition 6).

Input/output interactive Markov chains. In compositional dynamic fault tree (DFT) analysis [4] the components of a DFT are modelled using IOIMCs. The IOIMC formalism is a variant of the IMC formalism, inspired by I/O automata [19], in which the actions of a model are divided into input, output and internal transitions. The difference between weak Markovian bisimulation for IMCs and for IOIMCs is that for IOIMCs there may be more internal actions, while IMCs only use τ and the maximal progress assumption is also applied to output transitions. The first difference is handled by renaming all the different internal actions of an IOIMC to τ. The second difference is covered by refining the stability check in Algorithm 2 (line 17).

Of course we can apply our algorithm in DFT analysis only if the models encountered are acyclic. At first glance this is the case indeed, since a DFT describes the degradation of a system towards system failure, i.e. how the interplay of component failures may lead to a (possibly catastrophic) system failure. Since repairs are not considered, the behaviour is structurally acyclic.

There is a fine point because input-enabledness (typical of I/O automata) leads to models that may choose to do nothing in response to certain inputs.

This means that the state does not change, so the models are actually not acyclic. However, using the fact that the models are input-enabled and observably deterministic (see [4]), allows us to cut input loops without losing any information. To deal with the fact that two states $s \xrightarrow{a?} t$ may be bisimilar we make input-actions unobservable (just as we did for Markovian transitions). The adapted algorithm also maintains the information on removed input self-loops in order to properly build the weak transition relation.

6 Experimental Results

In this section we show, with a number of case studies that for acyclic models our algorithm is much more efficient than the existing algorithms. We compare the performance of the acyclic algorithm in minimising IOIMCs in the context of DFT analysis with the BCG_MIN tool from the CADP toolset [12]. We have so far been using BCG_MIN in the analysis of DFTs within the CORAL tool [4], which computes branching bisimulation for (IO)IMCs. However a weak bisimulation minimiser fits the theory better. Branching bisimulation is usually faster to compute than weak bisimulation and will give the same numerical results for DFT analysis, although intermediate models may be larger when using branching bisimulation. All experiments were run on a Linux machine with an AMD Athlon(tm) XP 2600+ processor at 2 GHz equipped with 2GB of RAM. In the following tables, n, m and m' stand for the number of states, transitions and the size of the weak transition relation from states in the input model to states in the reduced output model. m' is a lower bound on the size m^* of the weak transition relation of the input model. The computation of the latter is avoided by our algorithm, so we cannot report the sizes.

The first example we consider is a large version of a fault-tolerant parallel processor case study (FTPP-6) from [4]. It has 6 network elements and 6 groups of 4 processors. As explained earlier, the state spaces are constructed by compositional aggregation, where composition and minimisation steps are intertwined. Using the BCG_MIN tool in this process, leads to an overall time consumption of 7 hours, 3 minutes and 48 seconds. Using our acyclic minimization algorithm instead the same process only required 30 minutes and 20 seconds. Some insight in this 14-fold speed up is provided in the first rows of Table 1, where some representative intermediate state spaces and their reduction times are listed.

Another case study, an extended version of the cardiac assist system case study from [4] (CAS-l) is considered below. We found that the overall time for compositional aggregation decreased from 1 hour 57 minutes, 13 seconds (using BCG_MIN) to 1 minute 53 seconds using our dedicated algorithm. Again some intermediate steps are listed in the table.

For a larger version of the multi-processor distributed computing system case study (MDCS-l), also taken from [4], we found that the acyclic algorithm requires 5 minutes, 41 seconds to do compositional aggregation while BCG_MIN requires 30 minutes and 9 seconds. Table 1 shows the timing results for the minimization of various intermediate models.

Table 1. Minimisation times for acyclic IOIMCs in seconds

Case study	n	m	m'	BCG_MIN	acyclic
FTPP-6	9237	68419	93214	18.21	0.67
	32909	293955	398661	129.09	2.34
	79111	1324359	1364850	104.43	9.36
	101426	1043520	1402507	630.33	12.52
	255397	2920996	3883860	2028.29	83.60
	464762	5756020	7553746	4696.40	289.36
	464762	6066486	7833720	3742.70	222.15
	1180565	22147378	22502816	13631.44	666.28
CAS-1	38396	389714	482657	160.22	5.73
	61055	378219	548331	4455.71	6.27
	62860	419459	601135	1457.25	6.94
	66137	483157	672742	688.83	8.05
	57373	675330	1310517	14.05	5.42
MDCS-1	26466	244003	326916	13.37	1.39
	55578	645119	856984	41.78	3.3
	99242	1395459	1816748	75.21	6.98
	97482	1606231	2049696	101.38	9.38

As evident from the table and the reported time savings, the effectiveness of bisimulation minimization and of compositional aggregation is improved drastically for acyclic models.

7 Conclusion

This paper has developed bisimulation minimisation algorithms for acyclic IMC models. While this work is motivated in a very concrete application, namely the analysis of very large dynamic fault tree specifications, the results equally apply to acyclic labelled transition systems. We are rather happy with the speedup achieved over algorithms for possibly cyclic models. One interesting question we have not yet considered is how the algorithm can be twisted towards branching bisimulation, possibly also exploring links to normed simulation [14] and the cones-and-foci method [11]. Another promising avenue of research lies in extending the algorithm to also minimise cyclic models along the same lines as [9].

References

1. Blom, S., Fokkink, W., Groote, J.F., van Langevelde, I., Lisser, B., van de Pol, J.: μcrl: A toolset for analysing algebraic specifications. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 250–254. Springer, Heidelberg (2001)
2. Blom, S., Orzan, S.: Distributed branching bisimulation reduction of state spaces. Electr. Notes Theor. Comput. Sci. 89(1) (2003)
3. Böde, E., Herbstritt, M., Hermanns, H., Johr, S., Peikenkamp, T., Pulungan, R., Wimmer, R., Becker, B.: Compositional performability evaluation for statemate. In: QEST, pp. 167–178 (2006)
4. Boudali, H., Crouzen, P., Stoelinga, M.: A compositional semantics for dynamic fault trees in input/output interactive markov chains. In: ATVA, pp. 441–456 (2007)
5. Bravetti, M.: Revisiting interactive markov chains. Electronic Notes in Theoretical Computer Science 68(5) (2002)
6. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. In: ACM Symposium on Theory of Computing (1987)

7. Coste, N., Garavel, H., Hermanns, H., Hersemeule, R., Thonnart, Y., Zidouni, M.: Quantitative evaluation in embedded system design: Validation of multiprocessor multithreaded architectures. In: DATE (2008)
8. Crouzen, P., Hermanns, H., Zhang, L.: No Cycling? Go Faster! On the minimization of acyclic models. Reports of SFB/TR 14 AVACS 41, SFB/TR 14 AVACS (July 2008), http://www.avacs.org ISSN: 1860-9821
9. Dovier, A., Piazza, C., Policriti, A.: A fast bisimulation algorithm. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 79–90. Springer, Heidelberg (2001)
10. Dugan, J., Bavuso, S., Boyd, M.: Dynamic fault-tree models for fault-tolerant computer systems. IEEE Transactions on Reliability 41(3), 363–377 (1992)
11. Fokkink, W., Pang, J., van de Pol, J.: Cones and foci: A mechanical framework for protocol verification. Formal Methods in System Design 29(1), 1–31 (2006)
12. Garavel, H., Lang, F., Mateescu, R., Serwe, W.: Cadp 2006: A toolbox for the construction and analysis of distributed processes. In: CAV (2006)
13. Goralcikova, A., Koubek, V.: A reduct-and-closure algorithm for graphs. Mathematical Foundations of Computer Science 74, 301–307 (1979)
14. Griffioen, W.O.D., Vaandrager, F.W.: A theory of normed simulations. ACM Trans. Comput. Log. 5(4), 577–610 (2004)
15. Groote, J.F., Vaandrager, F.: An efficient algorithm for branching bisimulation and stuttering equivalence. In: Paterson, M. (ed.) ICALP 1990. LNCS, vol. 443. Springer, Heidelberg (1990)
16. Hermanns, H.: Interactive Markov Chains: The Quest for Quantified Quality. LNCS, vol. 2428. Springer, Heidelberg (2002)
17. Katoen, J.-P., Kemna, T., Zapreev, I.S., Jansen, D.N.: Bisimulation minimisation mostly speeds up probabilistic model checking. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 87–101. Springer, Heidelberg (2007)
18. Lang, F.: Refined interfaces for compositional verification. In: Najm, E., Pradat-Peyre, J.-F., Donzeau-Gouge, V.V. (eds.) FORTE 2006. LNCS, vol. 4229, pp. 159–174. Springer, Heidelberg (2006)
19. Lynch, N., Tuttle, M.: An introduction to input/output automata. CWI Quarterly 2(3), 219–246 (1989)
20. Mateescu, R.: Local model-checking of modal mu-calculus on acyclic labeled transition systems. In: Katoen, J.-P., Stevens, P. (eds.) TACAS 2002. LNCS, vol. 2280, pp. 281–295. Springer, Heidelberg (2002)
21. Milner, R.: Communication and Concurrency. Prentice Hall, Englewood Cliffs (1989)
22. Paige, R., Tarjan, R.E.: Three partition refinement algorithms. SIAM J. Comput. 16(6), 973–989 (1987)
23. Simon, K.: An improved algorithm for transitive closure on acyclic digraphs. In: Kott, L. (ed.) ICALP 1986. LNCS, vol. 226, pp. 376–386. Springer, Heidelberg (1986)
24. Vesely, W., Goldberg, F., Roberts, N., Haasl, D.: Fault Tree Handbook. NASA (1981)
25. Wimmer, R., Herbstritt, M., Hermanns, H., Strampp, K., Becker, B.: Sigref- a symbolic bisimulation tool box. In: ATVA, pp. 477–492 (2006)