Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER Electronic Notes in Theoretical Computer Science 104 (2004) 181–197
www.elsevier.com/locate/entcs

Compositional Minimization in Span(Graph): Some Examples

Piergiulio Katis[1] ,[2] N. Sabadini,[1] ,[3] R.F.C. Walters[1] ,[4]

Dip. Scienze CC. FF. MM.
Università dell'Insubria
Como, Italy

Abstract

We study a class of examples of minimizing automata with respect to branching bisimulation in the context of the Span(Graph) model. Compositional minimization is particularly efficient for the class which includes the classical dining philospher problem and variants. The reason for the efficiency is that finite subsets of the class generate finite submonoids of the *bisimulation monoid*. We indicate how this may be used in studying deadlock. In the case of the dining philosopher the critical fact is that $(F \cdot P)^3 = (F \cdot P)^2$ in the bisimulation monoid, where F is the fork and P the philosopher.

Keywords: model checking, minimization, branching bisimulation, automata, monoid, dining philosopher

1 Introduction

The classical notion of minimization of an automaton has been extended to the compositional setting of Span(Graph) in [9], in particular with respect to branching bisimulation. Similar minimizations are the basis for compositional model checking (e.g. [2], [10], [12]), the attempt to circumvent the problem

[1] The research has been financially supported by the Dipartimento di Scienze Chimiche, Fisiche e Matematiche of the University of Insubria, Como, Italy, and by the Italian Progetti Cofinanziati MUIR: *Computational Metamodels* (COMETA).
[2] Email: piergiulio_katis@hotmail.com
[3] Email: nicoletta.sabadini@uninsubria.it
[4] Email: robert.walters@uninsubria.it

doi:10.1016/j.entcs.2004.08.025

of the exponential growth in state that may occur in exhaustive automated state-space searches. The method has been used in practical model checking programs (e.g. [4]) with success in a variety of problems, though the approach has been shown to have its limitations ([3]).

It seems to be a very difficult mathematical problem to determine the class of problems for which compositional minimization reduces the state space from exponential to polynomial size. This paper is an investigation into a class of examples where the method is shown to be extremely efficient.

The general algebraic situation of this paper is described in [9], but to concentrate attention on the problem of minimization and to avoid an overload of algebraic details, in this paper we consider only two types of spans: those with two equal interfaces - a left and a right interface; and those with no interfaces. We consider spans with given initial states and we consider only two operations. If X and Y have two interfaces, we denote the reachable part of the span composite (the communicating parallel product) as $X \cdot Y$ which is again a span with two interfaces; this operation models the construction of identifying the right interface of X with the left interface of Y (with the effect of hiding the interface they now share). If X has two interfaces, we also define $\mathsf{Fb}(X)$ which is a span with no interfaces; the construction of feeding back the right interface of X to its left interface. These operations allow us to model problems which consist of a line, or a ring, of parallel processes each communicating with its two adjacent processes. Note that Span(Graph) [8] permits the description of more general systems with a circuit like geometry.

In section 2 we review briefly what we require of our span model, together with the notion of bisimilarity we will use – essentially branching bisimulation (further details may be found in [9]). We introduce the notion of the bisimulation monoid **BisAut**, whose elements are spans modulo branching bisimulation. Section 3 reviews the compositional minimization of spans with respect to branching bisimulation - we denote the minimization of X by $\min(X)$.

The fundamental complexity problem in compositional model checking is as follows. Consider an infinite sequence $X_1, X_2, ...$ of spans each with a maximum of c states and transitions. Then $\min(X_1 \cdot X_2 \cdot \cdots \cdot X_n)$ may be calculated by a sequence of elementary minimizations and compositions. The number of states and transitions of $\min(X_1 \cdot X_2 \cdot \cdots \cdot X_n)$ is clearly bounded by c^n. The question is for which sequences of spans may c^n be replaced by a polynomial in c. The technical content of the paper is dedicated to providing an infinite number of sequences of spans with this property, including the classical dining philosopher problem and variants of it. We obtain these examples by a detailed analysis of spans which communicate through locking and unlocking.

The families of spans considered in these examples in fact generate *finite* submonoids of **BisAut**; from this fact it can be immediately deduced that the minimization algorithm applied to these families is linear. In the case of the dining philosopher, the critical fact is that three forks and philosophers in a row are bisimilar to two philosophers and forks in a row – in terms of the bisimulation monoid, $(F \cdot P)^3 = (F \cdot P)^2$. This implies that a row of (more than two) alternating forks and philosophers is bisimilar to just two forks and philosophers. Though the dining philosophers is one of the most studied examples of a concurrent system, this fact seems to be previously unremarked.

In the last section we describe how various results about the existence/non-existence of deadlocks may be deduced. An implementation of the minimization algorithm by Pierantonio Redaelli, Mara Barbera and R.F.C. Walters was helpful in obtaining the particular results of the paper.

2 The model

2.1 Spans

A reflexive graph is a directed graph in which parallel edges may exist, and which has for each vertex a specified edge beginning and ending at that vertex, the *reflexive* edge at that vertex. We will consider particular spans of reflexive graphs, namely reflexive graphs whose edges have two labellings in a finite set A. We assume that the set A has a distinguished element which we denote '$-$', which is to be thought of as a silent action.

Definition 2.1 A *span of graphs with two equal interfaces* A, or just a span, is a tuple (G, g_0, l, r), where:
(i) G is a finite reflexive graph. Its vertices and edges are sometimes referred to as the *states* and *transitions* of the span. The reflexive edges are also called *idling* transitions.
(ii) g_0 is a vertex of G such that every vertex of G is reachable from g_0, the *initial state* of the span.
(iii) l and r are functions $l, r : \mathsf{Edge}(G) \to A$ such that if α is a reflexive edge then $l(\alpha) = r(\alpha) = -$. For each transition α of G, we call $l(\alpha)$ and $r(\alpha)$ the *left* and *right labellings* of the transition respectively. If $l(\alpha) = -$ then we say the transition e is *invisible on the left interface.* If $r(\alpha) = -$ then we say the transition α is *invisible on the right interface.* A *span with no interfaces* is just a pair (G, g_0), where G is a finite reflexive graph and g_0 is a vertex of G such that every vertex of G is reachable from g_0. If there is no cause for confusion, we denote the span (G, g_0, l, r), or the span (G, g_0), merely by G. A *behaviour* of G is a path (finite or infinite) in the graph G which starts at the initial state.

Example 2.2 We describe here two spans which can be used to model the example of the dining philosophers. Each has two interfaces. When we list the transitions of a span, we do not bother to list the reflexive edges, and a transition $\alpha : v \to w$ is denoted by $l(\alpha)/r(\alpha) : v \to w$. For this and other examples in this paper, we assume A contains the symbols 'l' and 'u', which stand for 'lock' and 'unlock' respectively.
The first span F models a *fork*. It has three states $0, 1, 2$ and the following transitions: $\mathsf{l}/- : 0 \to 1$, $\mathsf{u}/- : 1 \to 0$, $-/\mathsf{l} : 0 \to 2$, $-/\mathsf{u} : 2 \to 0$. The state 0 corresponds to the fork not being used, the state 1 to it being used, or locked, on the left, and the state 2 to it being used on the right. The initial state for F is 0. The second span P models a *right-handed philosopher*. It has four states $0, 1, 2, 3$ and the following transitions:

$$-/\mathsf{l} : 0 \to 1 \quad \mathsf{l}/- : 1 \to 2 \quad -/\mathsf{u} : 2 \to 3 \quad \mathsf{u}/- : 3 \to 0.$$

The state 0 corresponds to the philosopher not holding either fork, the state 1 to holding, or having locked, the fork on the right, the state 2 to holding both forks, the state 3 to only holding the fork to the left. The initial state for P is 0.

2.2 Operations on spans

Definition 2.3 Given two spans (G, g_0, l_G, r_G) and (H, h_0, l_H, r_H) we define their *composite* $(G \cdot H, k_0, l_{G \cdot H}, r_{G \cdot H})$ as follows. First we define a reflexive graph $G \times_A H$ called the restricted product of G and H. A vertex of $G \times_A H$ is an arbitrary pair of vertices $(g, h) \in G \times H$. An edge of $G \times_A H$ is a pair of edges $(\alpha, \beta) \in G \times H$ such that $r_G(\alpha) = l_H(\beta)$. The reflexive graph $G \cdot H$ is the subgraph of $G \times_A H$ reachable from the vertex (g_0, h_0), this vertex being the initial state k_0 of the composite. The left labelling of an edge $(\alpha, \beta) \in G \cdot H$ is defined by $l_{G \cdot H}(\alpha, \beta) = l_G(\alpha)$, and its right labelling by $r_{G \cdot H}(\alpha, \beta) = r_H(\beta)$.

Definition 2.4 If G is a span with two interfaces then the feedback of G, denoted $\mathsf{Fb}(G)$, is the span with no interfaces defined as follows. Define $G^\dagger$ to be the reflexive graph with the same vertices as G, but only those edges for which the left labelling equals the right labelling. $\mathsf{Fb}(G)$ is the subgraph of $G^\dagger$ reachable from the initial state of G.

We define a *straight-line system*, or just a system, to be a composite $X_1 \cdot X_2 \cdot ... \cdot X_n$ of spans, sometimes also denoted $X_1 X_2 ... X_n$. A *ring system* is an expression of spans of the form $\mathsf{Fb}(X_1 X_2 ... X_n)$.

Example 2.5 The system $(FP)^n F$ models a system of n dining philosophers sitting in a row. The classical problem of n dining philosophers sitting around a table is modelled by the system $\mathsf{Fb}((FP)^n)$.

Remark 2.6 The algebraic structure considered above is essentially a monoid together with a feedback (or trace) function – that is, a monoid M, a set S and a function $\mathsf{Fb} : M \to S$ which satisfies the property that $\mathsf{Fb}(a \cdot b) = \mathsf{Fb}(b \cdot a)$. Here, the set of spans with two interfaces plays the role of the monoid M, and the set of spans with no interfaces plays the role of the set S.

2.3 Bisimulation

The notion of bisimulation we define here is a slight variation of the notion of branching bisimulation. By the Stuttering Lemma of [6], our notion gives the same bisimilarity equivalence relation on spans as does branching bisimulation.

Definition 2.7 Suppose G and H are spans with two interfaces. A *bisimulation* between G and H is a relation $R \subseteq \mathsf{Vertex}(G) \times \mathsf{Vertex}(H)$ such that $(g_0, h_0) \in R$, and if $(g, h) \in R$ then:
(a) if there is an edge $a/b : g \to g'$ (where a and b may equal $-$), then there is a path $h = h_0 \to h_1 \to \ldots \to h_n$ in H such that (i) the first $n-1$ edges are invisible on both interfaces, (ii) the last edge is labelled a/b, (iii) for $0 \leq i < n$, $(g, h_i) \in R$ and (iv) (g', h_n).
(b) The symmetric property for h in terms of g.

We denote such a bisimulation by $R : G \Longrightarrow H$. We say two spans G and H are *bisimilar* if there exists a bisimulation $R : G \Longrightarrow H$, and we write $G \sim H$.

The notion of bisimulation between spans with no interfaces is trivial – we define any two spans with no interfaces to be bisimilar. This does not mean we are not interested in such systems – the dining philosophers has no interfaces. However, analysis of the internal structure of a system (with or without interfaces) may be facilitated by the use of bisimulations by abstracting parts of the system to simpler processes - see Section 5

The following theorem (from [9]) regards the compositionality of bisimulations.

Theorem 2.8 *(i) If $R : G \Longrightarrow H$ and $S : H \Longrightarrow K$ are bisimulations, then the composite relation $R \circ S \subseteq \mathsf{Vertex}(G) \times \mathsf{Vertex}(K)$ defines a bisimulation $R \circ S : G \Longrightarrow K$.*
(ii) The identity relation on $\mathsf{Vertex}(G)$ is a bisimulation $1 : G \Longrightarrow G$.
(iii) If $R : G \Longrightarrow H$ is a bisimulation, then the dual relation $R^\circ : H \Longrightarrow G$ is a bisimulation.
(iv) If $R : F \Longrightarrow G$ and $S : H \Longrightarrow K$ are bisimulations, then the relation

$$R \times S \subseteq (\mathsf{Vertex}(F) \times \mathsf{Vertex}(H)) \times (\mathsf{Vertex}(G) \times \mathsf{Vertex}(K))$$

when restricted to

$$\mathsf{Vertex}(F\cdot H)\times\mathsf{Vertex}(G\cdot K)\subseteq(\mathsf{Vertex}(F)\times\mathsf{Vertex}(H))\times(\mathsf{Vertex}(G)\times\mathsf{Vertex}(K))$$

defines a bisimulation $R\cdot S : F\cdot H \Longrightarrow G\cdot K$.

Corollary 2.9 *Bisimilarity is an equivalence relation on spans. Composition of spans induces a monoid structure on the set* **BisAut** *of equivalence classes of spans under the bisimilarity equivalence relation.*

Proof. The identity for the monoid **BisAut** is the equivalence class of the span $(S(A), *, 1_A, 1_A)$, where $S(A)$ is the graph with one vertex $*$, set of edges A, and reflexive edge $-$. It is straightforward to check that for any spans G, H, K we have $S(A)\cdot G \sim G \sim G\cdot S(A)$ and $G\cdot(H\cdot K)\sim(G\cdot H)\cdot K$. □

3 Compositional minimization

In this section we show how bisimulations give an example of a compositional quotient structure. We use this structure to define an algorithm for calculating the minimization of a system.

First we give the algorithm for constructing the maximal auto-bisimulation of any span G with two interfaces. The construction follows the standard method for constructing maximal bisimulations of transition systems (for example, see [1]).

Given a relation R on the states of G define a new relation $E(R)$ as follows:
$(x, y) \in E(R)$ if
(i) for any transition $a/b : x \to x'$ in G there is a path $y = y_0 \to y_1 \to y_2 \to \ldots \to y_n$ with the first $n-1$ transitions invisible on the left and the right, with $(x, y_i) \in R$ for $i = 0, 1, .., n-1$, and with $(x', y_n) \in R$, *and*
(ii) the symmetric property for y in terms of x.
Commence with the total relation T - any x is related to any y. Then $E^k(T)$ $(k = 1, 2, 3, \ldots)$ eventually stabilizes as an equivalence relation on the states of G, which is a bisimulation from G to itself. We call this equivalence relation the maximal auto-bisimulation of G.

For any G we define the minimization $\min(G)$ of G to be the following span. Its set of states consist of the equivalence classes of states of G under the maximal auto-bisimulation relation. There is a transition $a/b : [g] \to [h]$ in $\min(G)$ if there is such a labelled transition in G for any representatives of the classes. The initial state of $\min(G)$ is $[g_0]$.

Note that the construction of the minimization of G not only produces a span $\min(G)$ but also a bisimulation $\varepsilon_G : G \Longrightarrow \min(G)$ which is actually a function: $\varepsilon_G : g \mapsto [g]$. The functions ε may be used for locating deadlocks in a system.

We note that minimization with respect to branching bisimulation can be done in $O(n(n+m))$ time, where n is the number of states and m is the number of transitions ([7],[5]).

The crucial property of the construction min is that it is compositional in the sense of the following proposition.

Proposition 3.1 *For any spans G and H,*
(i) $\min(\min(G)) = \min(G)$*,*
(ii) $\min(G \cdot H) = \min(\min(G) \cdot \min(H))$ *and,*
(iii) $\varepsilon_{G \cdot H} = (\varepsilon_G \cdot \varepsilon_H) \circ \varepsilon_{\min(G) \cdot \min(H)}$ *.*

Other notions of bisimulation (e.g. Milner's strong or weak bisimulation) have similar properties.

The *compositional minimization algorithm* applied to a system $X_1 ... X_n$ is simply to calculate min of the value of the system by alternately calculating min and evaluating composition from the left. We assume the X_i's are already minimal. (If not, minimize them first.) Explicitly: initially set $M_1 = X_1$; for $i = 2$ to n {calculate $M_{i-1} \cdot X_i$; calculate $\min(M_{i-1} \cdot X_i)$ and $\varepsilon_{M_{i-1} \cdot X_i}$; set M_i equal to $\min(M_{i-1} \cdot X_i)$}.

Proposition 3.1 guarantees the correctness of this algorithm; namely that $M_n = \min(X_1 ... X_n)$ and that

$$\varepsilon_{X_1 \cdot ... \cdot X_n} = (\varepsilon_{M_1 \cdot X_2} \cdot 1_{X_3} \cdot ... \cdot 1_{X_n}) \circ (\varepsilon_{M_2 \cdot X_3} \cdot 1_{X_4} \cdot ... \cdot 1_{X_n}) \circ ... \circ \varepsilon_{M_{n-1} \cdot X_n}$$

3.1 Polynomial problems

Definition 3.2 A minimization problem is an infinite sequence $X_1, X_2, ...$ of spans for which there exists a number c such that, for all i, the number of states and transitions of X_i is less than c. The problem is said to be bounded by a function $b : \mathbf{N} \to \mathbf{N}$ if the number of states and transitions of $\min(X_1 \cdot X_2 \cdot \cdots \cdot X_n)$ is less than $b(n)$ (for all n). A problem is said to be *polynomial* (resp. linear or constant) if it is bounded by a polynomial (resp. linear or constant) function.

Every problem $X_1, X_2, ...$ is bounded by the exponential function $b(n) = c^n$, where c is the maximum number of states and transitions that an X_i may have. With regards to model checking, we are interested in problems that are bounded by polynomial functions.

In most cases, the X_i's will all belong to a finite family $\mathcal{F}$. For an example of a typical problem, let $\mathcal{F} = \{P, P^\circ\}$, where P° is the span obtained by swapping the left and right labellings of P; P° may be thought of as a philosopher who wants to pick up his left fork first. A consequence of what we show below is that any problem of the form $F, P_1, F, P_2, F, P_3, \ldots$ where $P_i \in \mathcal{F}$ is bounded by a constant function. (Note there is no symmetry in this class of problems, as we have not made any assumption about the distribution of the P_i's.)

To get an example bounded by a linear function, consider the span B with two states $0, 1$ and two (non-reflexive) transitions $m/- : 0 \to 1$ and $-/m : 1 \to 0$. This models a buffer, or message passer, of capacity one: the state 0 corresponds to the buffer being empty, and the state 1 to the buffer being full. It can be shown that $\min(B^n)$ has $n + 1$ states: the intuition is that B^n is a model of a buffer of capacity n. So the problem $B, B, B, \ldots$ is bounded by the linear function $n + 2$.

To obtain an example that is not bounded by a polynomial function, consider the span C with three states $0, 1, 2$ and three (non-reflexive) transitions $m/- : 0 \to 1$, $m/- : 1 \to 2$ and $-/m : 2 \to 0$. Think of this as a device which waits to receive two messages, then outputs only one and returns to its initial state. The problem $C, C, C, \ldots$ is not polynomial. To see this notice that the span C^n can accept $\sum_{k=1}^{n} 2^k$ – and no more – messages before outputting a message. Thus it contains at least $\sum_{k=1}^{n} 2^k$ states which are not bisimilar.

The following simple observation will be useful in our analysis of spans that lock and unlock.

Proposition 3.3 *Suppose the submonoid $\langle \mathcal{F} \rangle$ of* **BisAut** *generated by the bisimulation equivalence classes of the members of a finite family $\mathcal{F}$ is finite. Then any problem $X_1, X_2, \ldots$, where each $X_i \in \mathcal{F}$, is bounded by a constant function.*

Proof. Take the constant to be the maximum number of states that a minimal span in any equivalence class of $\langle \mathcal{F} \rangle$ may have. □

The following propositions relate the notion of polynomial and linear problems to the time complexity of the minimization algorithm.

Proposition 3.4 *Suppose $X_1, X_2, \ldots$ is a polynomial problem. Then there exists a polynomial function $t : \mathbf{N} \to \mathbf{N}$ such that, for any n, the time taken to apply the minimization algorithm to the system $X_1 \ldots X_n$ is less than $t(n)$.*

Proof. The minimization of a labelled transition system with respect to branching bisimulation can be done in $O(s(s+t))$ time, where s is the number of states and t is the number of transitions ([7]). Since we assume the alphabet A (in which the interfaces are labelled) is finite, it can be done in polynomial time with respect to the number of states. Let us say that the time taken to minimize a span with s states is bounded by a polynomial $f(s)$. Now suppose the polynomial q bounds the problem $X_1, X_2, \ldots$. Let c be the upper bound on the number of states that a span X_i may have. Then the number of states of $\min(X_1...X_n) \cdot X_{n+1}$ is bounded by the polynomial $q(n) \times c$. Let $g(n)$ be a polynomial which is an upper bound on the time taken to calculate the product of $\min(X_1...X_n)$ and X_{n+1}. Now the function $t(n) = \sum_{i=1}^{n}(g(i) + f(q(i) \times c))$ bounds the time taken to minimize $X_1...X_n$. The result now follows (since the integral of a polynomial is a polynomial). □

Proposition 3.5 *Suppose $X_1, X_2, ...$ is a constant problem. Then then there exists linear function $t : \mathbf{N} \to \mathbf{N}$ such that, for any n, the time taken to apply the minimization algorithm to the system $X_1...X_n$ is less than $t(n)$.*

Proof. There exists a constant c such that, for all i, the time taken to calculate the product of $\min(X_1...X_i)$ and X_{i+1} and then minimize the result is less than c. Thus the time taken to apply the minimization algorithm to $X_1...X_n$ is less than $c \times n$. □

Corollary 3.6 *Suppose $\mathcal{F}$ is a finite family of spans and the submonoid $\langle \mathcal{F} \rangle$ of* **BisAut** *generated by the bisimulation equivalence classes of the members of the family $\mathcal{F}$ is finite. Then the time taken to apply the minimization algorithm to any system $X_1...X_n$, where $X_i \in \mathcal{F}$, is linear in n.*

4 The Examples

In this section we consider some examples of finite submonoids of **BisAut**, and thus of problems which are bounded by constants and can be minimized in linear time. In particular, we study classes of, what we call, 'lock-unlock' spans. One of these classes includes the example of the dining philosopher.

1. Let Z be the span with one state, and whose only transition is the reflexive edge. This span has the property that $ZGZ \sim Z$, for any G. Immediate consequences of this are that $(ZG)(ZH) \sim ZH$ and $(GZ)(HZ) \sim GZ$, for any G and H. So any finite family comprising spans of the form ZG and HZ generate a finite submonoid of **BisAut**. The span Z may be thought of as one which blocks an interface, and thus prohibits the exponential growth of state in model checking. For example, if we wanted to check a property such as deadlock of the system $ZG_1ZG_2G_3ZG_4...ZG_nZ$, it is intuitively clear we only need check the spans $ZG_1Z, ZG_2G_3Z, ..., ZG_nZ$ separately for this property; and the time taken for such a process will increase linearly with respect to n.

2. Let N be the span with two states $0, 1$ and transitions $m/- : 0 \to 1$, $m/- : 1 \to 1$, $-/m : 1 \to 1$, $m/m : 1 \to 1$ and $-/m : 1 \to 0$. We call this span a non-deterministic buffer or a buffer of an undetermined capacity. It has the property that $N^2 \sim N$; that is, its equivalence class is idempotent in

BisAut. The problem $N, N, \ldots$ is bounded by a constant.

3. In Example 1. above constant problems were identified by using the span Z to 'break up' systems – i.e. to limit the communication that can occur in a system. We will consider a more interesting instance of this phenomenon in this example. Instead of Z we will use F, the fork span of Example 2.2.

We call X a *lock-unlock* span if it has the following property: from every state of FXF there is a path to the initial state. The idea is that X is a process which wants to complete a 'cycle', and to do so it may require to lock and unlock a resource on its left and one on its right, perhaps many times and perhaps non-deterministically. A simple example of a lock-unlock span is the philosopher P, described in Example 2.2.

We will define four infinite classes $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_3$ and $\mathcal{F}_4$ of lock-unlock spans (one of which will include P) such that for X in any of these classes $(FX)^2 \sim (FX)^3$. Moreover, we will show that any finite subset of

$$F(\mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3 \cup \mathcal{F}_4) = \{FX \mid X \in \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3 \cup \mathcal{F}_4\}$$

generates a finite submonoid of **BisAut**. To do so we require the following definitions.

We say a path in G is invisible if every transition in the path is invisible on the left and on the right.

A state $(2, x, 2) \in FXF$ is said to be *bound* if there does not exist a $y \in X$ and an invisible path in FX from $(2, x)$ to $(0, y)$. In other words, $(2, x, 2)$ is bound if in order for X to unlock its left fork X must gain access to its right fork.

Similarly, a state $(1, x, 1) \in FXF$ is said to be *bound* if there does not exist a $y \in X$ and an invisible path in XF from $(x, 1)$ to $(y, 0)$.

A state $(f, x, 2) \in FXF$ is said to be *free* if there is a $y \in X$ and an invisible path in FX from (f, x) to $(0, y)$, and if there is no invisible path in FX from (f, x) to $(2, z)$, where $(2, z, 2)$ is a bound state. Note that this means $f \neq 1$.

Similarly, we say a state $(1, x, f) \in FXF$ is *free* if there is a $y \in X$ and an invisible path in XF from (x, f) to $(y, 0)$, and if there is no invisible path in XF from (x, f) to $(z, 1)$, where $(1, z, 1)$ is a bound state. Note that this means $f \neq 2$.

A state $(f, x, 2) \in FXF$ is said to be *free-bound* if one of the following two conditions hold: (i) $(f, x, 2)$ is bound; or (ii) there is a $y \in X$ and an invisible path in FX from (f, x) to $(0, y)$, and there is no invisible path in FX from (f, x) to (g, z), where $(g, z, 2)$ is a free state. So a free-bound state is bound or can become bound.

Similarly, we say a state $(1, x, f) \in FXF$ is *free-bound* if one of the following two conditions hold: (i) $(1, x, f)$ is bound; or (ii) there is a $y \in X$ and an invisible path in XF from (x, f) to $(y, 0)$, and there is no invisible path in XF from (x, f) to (z, g), where $(1, z, g)$ is a free state.

We now define the four classes of lock-unlock spans.

(i) $X \in \mathcal{F}_1$ if and only if all states of the forms

$$(1, x, 1), (1, x, 0), (0, x, 2), (2, x, 2) \in FXF$$

are free. So if X performs a lock on the left (resp. right), it is able to follow that action with an unlock on the left (resp. right).

(ii) $X \in \mathcal{F}_2$ if and only if all states of the forms

$$(1, x, 1), (1, x, 0), (0, x, 2), (2, x, 2) \in FXF$$

are free-bound.

(iii) $X \in \mathcal{F}_3$ if and only if all states of the forms $(1, x, 1), (1, x, 0) \in FXF$ are free and all states of the form $(0, x, 2), (2, x, 2) \in FXF$ are free-bound.

(iv) $X \in \mathcal{F}_4$ if and only if all states of the forms $(1, x, 1), (1, x, 0) \in FXF$ are free-bound and all states of the form $(0, x, 2), (2, x, 2) \in FXF$ are free.

An example of a span in $\mathcal{F}_1$ is F; Z is also an example. An example of a span in $\mathcal{F}_4$ is P. An example of a span in $\mathcal{F}_3$ is P°, the left-handed philosopher defined earlier this section. An example of a span in $\mathcal{F}_2$ is $P + P^\circ$, the span formed by taking the disjoint union of P and P° and then smashing their initial states together (it has 7 states and 8 non-reflexive edges).

The following is an example of a lock-unlock span P' which is not in one of the above four classes. P' has six states 0,1,2,3,4,5 and the following transitions:

$$l/- : 0 \to 1 \quad -/l : 1 \to 2 \quad -/u : 2 \to 3$$
$$u/- : 3 \to 4 \quad -/l : 4 \to 5 \quad -/u : 5 \to 0$$

The reason why it does not fit into one of these classes is that the state $(1, 1, 2) \in FP'F$ is bound and the state $(0, 4, 2) \in FP'F$ is free. As an aside we note that $(FP')^2$ is not bisimilar to $(FP')^3$, but $(FP')^3 \sim (FP')^4$.

Let $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3 \cup \mathcal{F}_4$.

Claim 4.1 *Suppose $T \subset \mathcal{F}$ is finite. Any problem of the form $F, X_1, F, X_2, ...,$ where each $X_i \in T$, is bounded by a constant.*

We prove that if $S \subset F\mathcal{F}$ is finite then the submonoid $\langle S \rangle$ of **BisAut** is finite. This implies that problems of the form $FX_1, FX_2, FX_3, ...$ are bounded by a constant; and from this the Claim is immediate.

Lemma 4.2 *If X and Y are lock-unlock spans then XFY is a lock-unlock span.*

Proof. Consider a state $(f, x, g, y, h) \in FXFYF$. Either X or Y has, or has access to, the middle fork. Assume X does. Then there is a path from (f, x, g, y, h) to $(0, 0, 0, y, h)$. There is a path from $(0, 0, 0, y, h)$ to $(0, 0, 0, 0, 0)$.$\square$

We define a monoid structure on the four element set $\{1, 2, 3, 4\}$ as follows:

$*$	1	2	3	4
1	1	1	1	1
2	1	2	3	4
3	1	3	3	1
4	1	4	1	4

Lemma 4.3 *If $X \in \mathcal{F}_i$ and $Y \in \mathcal{F}_j$ then $XFY \in \mathcal{F}_{i*j}$.*

Proof. First consider $i = 1$ and any $j = 1, 2, 3, 4$. Consider a state $(2, x, f, y, 2)$ in $FXFYF$. If $f \neq 2$ then, since X is a lock-unlock span, there is an invisible path in FXF from $(2, x, f)$ to $(0, 0, 0)$, and thus an invisible path in $FXFYF$ from $(2, x, f, y, 2)$ to $(0, 0, 0, y, 2)$. Suppose $f = 2$. Since $(2, x, f) \in FXF$ is free, there is an invisible path in FX from $(2, x)$ to $(0, x)$, and thus an invisible path in $FXFYF$ from $(2, x, f, y, 2)$ to $(0, x, f, y, 2)$. Since we have not placed any restrictions on $(x, f, y) \in XFY$, the states $(2, x, f, y, 2)$ are free as they can never become bound. Consider a state of the form $(1, x, f, y, 1)$. If $f \neq 1$ then there is an invisible path in FYF from $(f, y, 1)$ to $(0, 0, 0)$, since Y is a lock-unlock span; and thus an invisible path from $(1, x, f, y, 1)$ to $(1, x, 0, 0, 0)$. If $f = 1$ then, since $(1, x, 1) \in FXF$ is free, there is an invisible path in XF from $(x, 1)$ to $(x', 0)$, and thus an invisible path in $FXFYF$ from $(2, x, f, y, 2)$ to $(0, z, 0, y, 2)$, which brings us to the previous case $f \neq 1$. Since we have not placed any restrictions on $(x, f, y) \in XFY$, the states $(2, x, f, y, 2)$ are free, as they can never become bound. This proves that if $X \in \mathcal{F}_1$ and $Y \in \mathcal{F}_j$ then $XFY \in \mathcal{F}_1$. A symmetric argument shows that if $X \in \mathcal{F}_i$ and $Y \in \mathcal{F}_1$ then $XFY \in \mathcal{F}_1$.

We now consider the case $X \in \mathcal{F}_2$ and $Y \in \mathcal{F}_3$. Arguments similar to those given in the previous paragraph can be used to show any state $(1, x, f, y, 1)$ is free. We need to show that there is an invisible path from any state

$(2, x, f, y, 2)$ to a bound state. If $f \neq 2$ then there is an invisible path to a state $(2, x', 2, y, 2)$, since X is a lock-unlock span. So assume $f = 2$. Since $X \in \mathcal{F}_2$ there is an invisible path in FX from $(2, x)$ to $(2, x')$, where $(2, x', 2) \in FXF$ is bound. Since $Y \in \mathcal{F}_3$ there is an invisible path in FY from $(2, y)$ to a state $(2, y', 2) \in FYF$ which is bound. Thus there is an invisible path in $FXFYF$ from any $(2, x, f, y, 2)$ to a bound state $(2, x', 2, y', 2)$. So $XY \in \mathcal{F}_3$. The remaining cases can be argued in similar ways. □

Lemma 4.4 *For $i = 1, 2, 3, 4$, if $X, Y \in \mathcal{F}_i$ then $FXF \sim FYF$.*

Proof. For any $X \in \mathcal{F}$, we partition the states of FXF into eight sets U_1, U_2, U_3, U_4, U_5, U_6, U_7, U_8 (some of which may be empty).

1. For all x, $(f, x, g) \in U_1$ if $f \in \{0, 2\}$ and $g \in \{0, 1\}$.
2. For all x, $(1, x, 2) \in U_2$.
3. $(f, x, 2) \in U_3$ if it is free.
4. $(f, x, 2) \in U_4$ if it is free-bound and not bound.
5. $(f, x, 2) \in U_4$ if it is bound.
6. $(1, x, f) \in U_5$ if it is free.
7. $(1, x, f) \in U_6$ if it is free-bound and not bound.
8. $(1, x, f) \in U_6$ if it is bound.

For $X \in \mathcal{F}$, this partition is well defined. Notice that, by definition of the $\mathcal{F}_i$'s, for each X either U_3 or $(U_4 \cup U_5)$ will be empty, and either U_6 or $(U_7 \cup U_8)$ will be empty. With this last observation in mind, it is easy to check that the equivalence relation induced by this partition is an auto-bisimulation (in fact, maximal). Furthermore, it is straightforward to show that if $X, Y \in \mathcal{F}_i$ then the minimal spans of X and Y are the same. □

We can use the previous results to deduce that $(FX)^2 \sim (FX)^3$ for $X \in \mathcal{F}_i$. Lemma 4.3, together with the four element multiplication table above it, implies that $XFX \in \mathcal{F}_i$. Lemma 4.4 now implies that $FXFXF \sim FXF$. Hence, the desired result.

Lemma 4.5 *If $S \subset F\mathcal{F}$ has s elements, then the number of elements of the submonoid $\langle S \rangle$ of* **BisAut** *is less than or equal to $4 \times s + 1$.*

Proof. Let F_1, F_2, F_3, F_4 be the four minimal spans such that for any $X \in \mathcal{F}_i$, $FXF \sim F_i$. By the previous two Lemmas, for any $X_1, ..., X_n \in \mathcal{F}$, $FX_1F...FX_nF \sim F_i$, for some i. So any element of $\langle S \rangle$ equals the equivalence class of F_iX, for some $i \in \{1, 2, 3, 4\}$ and X such that $FX \in S$. (The additional 1 in the sum comes from counting the identity of the monoid.) □

This completes the proof of the claim.

Remark 4.6 We give explicit descriptions of the four minimal spans F_1, F_2, F_3, F_4. From the proof of Lemma 4.4, it is clear that F_1 has 4 states (corresponding to the cases 4,5,7 and 8 being empty), F_2 has 6 states (cases 3 and 6 empty), and F_3 (cases 3,7 and 8 empty) and F_4 (cases 4,5 and 6 empty) have 5 states. We describe the minimal spans as sub-spans of the span Q which has 8 states (each state corresponds to one of the cases of the partition described in the proof of Lemma 4.4) and the following transitions:

$$
\begin{aligned}
l/l &: 1{\to}2,\ -/l{:}1{\to}3,\ -/l{:}1{\to}4,\ -/l{:}1{\to}5,\ l/-{:}1{\to}6,\ l/-{:}1{\to}7,\\
l/- &: 1{\to}8,\ u/u{:}2{\to}1,\ u/-{:}2{\to}3,\ u/-{:}2{\to}4,\ -/u{:}2{\to}6,\ -/u{:}2{\to}7,\\
-/u &: 3{\to}1,\ l/-{:}3{\to}2,\ l/u{:}3{\to}6,\ l/u{:}3{\to}7,\ -/u{:}4{\to}1,\ l/-{:}4{\to}2,\\
-/- &: 4{\to}5,\ l/u{:}4{\to}6,\ l/u{:}4{\to}7,\ -/u{:}5{\to}1,\ u/-{:}6{\to}1,\ -/l{:}6{\to}2,\\
u/l &: 6{\to}3,\ u/l{:}6{\to}4,\ u/-{:}7{\to}1,\ -/l{:}7{\to}2,\ u/l{:}7{\to}3,\ u/l{:}7{\to}4,\ -/- : 7{\to}8,\ u/-{:}8{\to}1.
\end{aligned}
$$

F_1 is the sub-span of Q with states 1,2,3 and 6. F_2 is the sub-span of Q with states 1,2,4,5,7 and 8. F_3 is the sub-span of Q with states 1,2,4,5 and 6. F_4 is the sub-span of Q with states 1,2,3,7 and 8.

Remark 4.7 The above calculations may be interpreted as follows. Lock-unlock spans form a submonoid $\mathcal{LU}$ of **BisAut** comprising bisimulation classes of spans of the form FX where X is a lock-unlock span. Though the submonoid $\langle F\mathcal{F}_1 \cup F\mathcal{F}_2 \cup F\mathcal{F}_3 \cup F\mathcal{F}_4 \rangle$ of $\mathcal{LU}$ is infinite, any finite subset S of it generates a finite submonoid.

Remark 4.8 Exploration with a program implementing the algorithm have revealed other examples of finite submonoids of **BisAut**. One in particular is of interest. Suppose the fork F and the philosopher P are both modified as follows: each nonreflexive label is replaced by two labels (for example lock becomes beginlock and endlock) and each noneflexive transition is replace by two transitions, the beginning of the transition and the end of the transition, with the intermediate state being a new one. Then the new spans $F'$and P' satisfy $(F' \cdot P')^3 = (F' \cdot P')^2$ in **BisAut**.

5 Deadlock

In this section we describe a simple result which in many cases allows us to deduce the nonexistence of reachable deadlocks. The situation is rather analogous to one in Number Theory [11] where the idea of reducing modulo a natural number can lead to the conclusion that a diophantine equation has no solutions in integers. To illustrate the point more concretely consider the sequence of diophantine equations:

$$5^k x^2 - 3^{4k} y^2 = 7^{2k} + 1 \qquad (k = 1, 2, \ldots).$$

Even to check a single pair of integral values x, y seems to be, on the surface, an exponential calculation in k. However considering modulo 4 and using the compositional reduction modulo 4 the sequence of equations becomes the single congruence $x^2 - y^2 \equiv 2 \mod 4 \quad (k = 1, 2, \ldots)$ which is easily seen to have no solutions by just checking 16 cases. This implies that none of the original diophantine equations has a solution in integers. Notice that using compositional reduction modulo 2 does not allow the same deduction. It is fundamental to the utility of congruences that information is lost, and that they yield only negative information about the diophantine equation.

We use precisely analogous arguments to prove that many systems do not have deadlock.

Definition 5.1 A deadlock of a span G (with two or no interfaces) is a state g such that the only transition out of g is the idling transition.

An example of a deadlock state is the state $(1,1,1,1,1,1)$ of $\mathsf{Fb}((FP)^3)$, which is the state corresponding to each philosopher having their right forks.

Proposition 5.2 *If g is a deadlock of span G then $\varepsilon_G(g)$ is a deadlock of* $\min(G)$. *If g is a deadlock of* $\mathsf{Fb}(G)$ *then $\varepsilon_G(g)$ is a deadlock of* $\mathsf{Fb}(\min(G))$.

Proof straightforward.

We wish to use these results together with the results of the previous section to investigate the existence of deadlocks in systems of the form

$$\mathsf{Fb}(FX_1FX_2...FX_n) \text{ where } X_1, X_2, ..., X_n \in \mathcal{F}.$$

By the Proposition 5.2 *if* $\mathsf{Fb}(\min(FX_1FX_2...FX_n))$ *has no deadlocks then also* $\mathsf{Fb}(FX_1FX_2...FX_n)$ *has no deadlocks.* But it is straightforward using the results of the previous section to calculate $\min(FX_1FX_2...FX_n)$. The span $\min(FX_1FX_2...X_{n-1}F)$ is one of the four minimal spans (Remark 4.6) F_1, F_2, F_3, F_4, say F_i. Which one can be calculated using the monoid structure (Lemma 4.3) on the classes $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4$. Then $\mathsf{Fb}(\min(FX_1FX_2...FX_n)) = \mathsf{Fb}(\min(F_iX_n))$. If this span has no deadlock we are able to deduce that the ring of processes $\mathsf{Fb}(FX_1FX_2...FX_n)$ also has no deadlocks.

Let us apply this in the case that $X_1, X_2, ..., X_n \in \{F, P + P^\circ, P^\circ, P\}$. We are only able to obtain a result when $\mathsf{Fb}(\min(F_iX_n))$ has no deadlock. A simple calculation shows that this occurs exactly in the cases where $F_i = F$ or $X_n = F$ or the following (i) $\mathsf{Fb}(\min(F_3P))$, (ii) $\mathsf{Fb}(\min(F_4P^\circ))$. We illustrate with two simple examples.

Example 5.3 Consider the ring of philosophers containing four right-handed and two left-handed philosophers defined by the expression

$$\mathsf{Fb}(FPFP^\circ FPFP^\circ FPFP).$$

Applying Lemma 4.3 repeatedly we see that $PFP^\circ FPFP^\circ FP$ is in $\mathcal{F}_1$, and hence $\min(FPFP^\circ FPFP^\circ FPF) = F_1$. Finally

$$\mathsf{Fb}(\min(FPFP^\circ FPFP^\circ FPFP)) = \mathsf{Fb}(\min(F_1P))$$

which last has no deadlock and hence we deduce that the original ring of philosophers has no deadlock.

Example 5.4 Consider the ring of four right-handed philosophers defined by the expression $\mathsf{Fb}(FPFPFPFPFP)$. Applying Lemma 4.3 repeatedly we see that $PFPFPFP$ is in $\mathcal{F}_4$ and hence that $\min(FPFPFPFPF)$ is F_4. Finally $\mathsf{Fb}(\min(FPFPFPFPFP)) = \mathsf{Fb}(\min(F_4P))$ which last *does* have a deadlock. We can make no deduction about deadlocks by this argument (although more careful examination of the minimization function ε in this case does allow the detection of the single deadlock).

References

[1] A. Arnold, Finite transition systems, Prentice Hall, 1994.

[2] E. Clarke, D. Long, K. McMillan. Compositional model checking. In Proceedings of the Fourth Annual IEEE Symposium on Logic in Computer Science, pp. 353-362, 1989.

[3] J.C. Corbett and G. S. Avrunin, Towards Scalable Compositional Analysis, Proceedings of the Second Symposium on Foundations of Software Engineering, ed. David Wile, December, 1994.

[4] J Fernandez, Aldébaran: Manuel de l'utilisateur. Technical Report, LGI-IMAG Grenoble, 1988.

[5] J Fernandez, L. Mounier. A tool set for deciding behavioural equivalences. Preprint.

[6] R. van Glabbeek, P. Weijland, Branching time and abstraction in bisimulation semantics, in: JACM 43(3), 1996, pp. 555-600.

[7] Groote J and Vaandrager F, An efficient algorithm for branching bisimulation and stuttering equivalence. CS-R 9001, CWI, Amsterdam, 1989.

[8] P. Katis, N. Sabadini, R.F.C. Walters, Span(Graph): A categorical algebra of transition systems, Proceedings Algebraic Methodology and Software Technology, volume 1349 of Lecture Notes in Computer Science, 307–321, Springer Verlag, 1997.

[9] Rosebrugh R., Sabadini N, Walters RFC, Minimization and minimal realization in Span(Graph), in press, MSCS.

[10] B. Steffen S. Graf, G Lüttgen. Compositional minimization of finite state systems. International Journal of Formal Aspects of Computing, Vol 8, pp 607-616, 1996.

[11] R.F.C. Walters, Number Theory: an Introduction, Carslaw Publications, 1987.

[12] W Yeh, M Young. Compositional reachability analysis using process algebra. In Proc. Symposium on Testing, Analysis, and Verification (TAV4), pp 178-187, New York, Oct 1991. ACM SIGSOFT.