

Electronic Notes in Theoretical Computer Science 262 (2010) 33–48

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Controller/Orchestrator Synthesis via Filtration

Philippe Balbiani

*CNRS — Universit´e de Toulouse*

*Institut de recherche en informatique de Toulouse*

*118 ROUTE DE NARBONNE, 31062 TOULOUSE CEDEX 9, FRANCE*

[*Philippe.Balbiani@irit.fr*](mailto:Philippe.Balbiani@irit.fr)

Fahima Cheikh

*UT1 — Universit´e de Toulouse*

*Institut de recherche en informatique de Toulouse*

*118 ROUTE DE NARBONNE, 31062 TOULOUSE CEDEX 9, FRANCE*

[*Fahima.Cheikh@irit.fr*](mailto:Fahima.Cheikh@irit.fr)

Guillaume Feuillade

*UPS — Universit´e de Toulouse*

*Institut de recherche en informatique de Toulouse*

*118 ROUTE DE NARBONNE, 31062 TOULOUSE CEDEX 9, FRANCE*

[*Guillaume.Feuillade@irit.fr*](mailto:Guillaume.Feuillade@irit.fr)

**Abstract**

The present paper is interested in the following decision problems: (1) given finite frames *F, F'*, determine if there exists a frame *F''* such that *F* and *F' ⊗F''*, the synchronous product of *F'* and *F''*, are bisimilar;

(2) given finite frames *F, F'*, determine if there exists a frame *F''* such that *F* and *F'⊕F''*, the asynchronous product of *F'* and *F''*, are bisimilar. It shows that variants of the filtration method are adequate for solving them.

*Keywords:* Modal logic, filtration, bisimulation, controller synthesis, orchestrator synthesis.

# Introduction

Multifarious controller synthesis problems, as introduced by Maler *et al.* [[11](#_bookmark27)] and Ramadge and Wonham [[14](#_bookmark30)], amount, given finite transition systems *S, S'*, to deter- mine if there exists a transition system *S''* such that *S* and *S' ⊗S''*, the synchronous product of *S'* and *S''*, are equivalent. The role of *S''* is to restrict the behaviours of *S'*. Hence, in this setting, *S*, *S'* and *S''* can be respectively seen as the control objective, the reactive system to be controlled and the controller whereas *S' ⊗ S''*

1571-0661 © 2010 Elsevier B.V. Open access under [CC BY-NC-ND license](http://creativecommons.org/licenses/by-nc-nd/3.0/).

doi:10.1016/j.entcs.2010.04.004

denotes the restricted system. Controller synthesis problems arise in a variety of contexts ranging from computer operating systems to complex multimode processes. The exponential-time method proposed by Arnold *et al.* [[3](#_bookmark18)] to solve them consists in transforming them into formula satisfiability problems in *μ*-calculus [[2](#_bookmark19)].

Several orchestrator synthesis problems, as introduced by Berardi *et al.* [[4](#_bookmark20)] and Berardi *et al.* [[5](#_bookmark22)], amount, given finite distributed systems *S, S'*, to determine if there exists a distributed system *S''* such that *S* and *S' ⊕S''*, the asynchronous product of *S'* and *S''*, are equivalent. The role of *S''* is to enhance the behaviours of *S'*. Hence, in this setting, *S*, *S'* and *S''* can be respectively seen as the orchestration objective, the reactive system to be orchestrated and the orchestrator whereas *S' ⊕S''* denotes the enhanced system. Orchestrator synthesis problems arise in a variety of contexts ranging from service oriented computing to ambiant intelligence. The exponential- time method proposed by Berardi *et al.* [[6](#_bookmark23)] to solve them consists in transforming them into formula satisfiability problems in propositional dynamic logic [[10](#_bookmark28)].

Transition systems and distributed systems can be abstracted as frames. Hence, the present paper is interested in the following controller/orchestrator synthesis problems: (1) given finite frames *F, F'*, determine if there exists a frame *F''* such that *F* and *F' ⊗F''*, the synchronous product of *F'* and *F''*, are bisimilar; (2) given finite frames *F, F'*, determine if there exists a frame *F''* such that *F* and *F' ⊕ F''*, the asynchronous product of *F'* and *F''*, are bisimilar. It is probably correct to say that these decision problems are motivated more by model-theoretic and complexity-theoretic characteristics than by tools for the philosophical analysis of modal concepts. Nevertheless, there are various reasons to believe that they are very similar to the formula satisfiability problems traditionally considered in modal logic.

What the present paper shows is that variants of the filtration method are ad- equate for solving them, i.e. we will use these variants to give exponential-time algorithms for solving our controller/orchestrator synthesis problems. Its section- by-section breakdown is as follows. Section [2](#_bookmark0) establishes the concepts of frame, bisimulation, synchronous product and asynchronous product. In section [3](#_bookmark3), basic definitions concerning the controller synthesis problem and the orchestrator synthe- sis problem are given. Based on variants of the filtration method, ways of solving both problems are presented in sections [4](#_bookmark5) and [5](#_bookmark16). Section [6](#_bookmark17) studies variants of our synthesis problems. We assume the reader is at home with tools and techniques in modal logic. For more on these see [[7](#_bookmark24)].

# Basic notions

This section presents the basic notions needed to introduce the controller synthesis problem and the orchestrator synthesis problem.

* 1. *Frame*

Let *PG* be a set of program variables (with typical members denoted *a*, *b*, etc). A frame over *PG* is a structure of the form *F* = (*W, R*) where *W* is a nonempty set

of states (with typical members denoted *x*, *y*, etc) and *R* is a function from *W* 2 to 2*PG*. The set *W* of states is to be regarded as the set of all possible states in a computational process whereas the function *R* from *W* 2 to 2*PG* associates with each pair of states a set of program variables with *a ∈ R*(*x, y*) meaning that state *y* can be reached from state *x* by performing program *a*. In this paper, we shall always consider that there exists a root *x*0 *∈ W* such that for all *x ∈ W* , there exists a nonnegative integer *n* and *a*1*,..., an ∈ PG* such that *x* can be reached from *x*0 by performing programs *a*1, *.. .*, *an*. For all *a ∈ P G*, let *Ra ⊆ W × W* be the binary relation such that for all *x, y ∈ W* ,

* *x Ra y* iff *a ∈ R*(*x, y*).

*F* is said to be finite iff *W* is finite. We shall say that *F* is deterministic iff for all *x ∈ W* , for all *a ∈ P G*, the set of all *y ∈ W* such that *x Ra y* has cardinality 0 or 1 whereas we shall say that *F* is serial iff for all *x ∈ W* , for all *a ∈ P G*, the set of all *y ∈ W* such that *x Ra y* has cardinality 1 or more. *F* is said to be an equivalence frame iff for all *a ∈ P G*, *Ra* is reflexive, symmetrical and transitive. We shall say that *F* is reflexive (respectively: symmetrical, transitive) iff for all *a ∈ P G*, *Ra* is reflexive (respectively: symmetrical, transitive).

* 1. *Bisimulation*

Let *F* = (*W, R*), *F'* = (*W', R'*) be frames over *P G*. A binary relation *Z ⊆ W × W'* is called a bisimulation between *F* and *F'*, in symbols *Z*: *F ←→ F'*, iff for all *x ∈ W* , for all *x' ∈ W'*, if *x Z x'* then

* for all *a ∈ P G*, for all *y ∈ W* , if *x Ra y* then there exists *y' ∈ W'* such that *x' R'a y'* and *y Z y'*,
* for all *a ∈ P G*, for all *y' ∈ W'*, if *x' R'a y'* then there exists *y ∈ W* such that *x Ra y* and *y Z y'*.

If *x ∈ W* , *x' ∈ W'* are such that *x Z x'* then we say that *x* and *x'* are bisimilar, in symbols *Z*: *F,x ←→ F', x'*. If *x ∈ W* , *x' ∈ W'* are such that there exists a bisimulation *Z* between *F* and *F'* such that *Z*: *F,x ←→ F', x'* then we write *F,x ←→ F', x'*. It is a well-known fact that the following decision problem is in *PTIME*:

* Given a finite set *PG* of program variables, finite frames *F* = (*W, R*), *F'* = (*W', R'*) over *PG*, roots *x*0 *∈ W* , *x'*0 *∈ W'*, determine whether *F, x*0 *←→ F', x'*0.

See [[1](#_bookmark21)] for details.

* 1. *Synchronous product and asynchronous product*

Let *F* = (*W, R*), *F'* = (*W', R'*) be frames over *P G*. By *F ⊗ F'*, we denote the synchronous product of *F* and *F'*, i.e. the frame *F''* = (*W '', R''*) over *PG* where

* *W''* = *W × W'*,
* *R''* is the function from *W''*2 to 2*PG* such that for all (*x, x'*)*,* (*y, y'*) *∈ W''*,

*R''*((*x, x'*)*,* (*y, y'*)) is the set of all *a ∈ PG* such that *a ∈ R*(*x, y*) and *a ∈ R'*(*x', y'*). Let *G* = (*V, E*), *G'* = (*V ', E'*) be frames over *P G*, *x*0 *∈ W* , *x'*0 *∈ W'*, *v*0 *∈ V* , *v'*0 *∈*

*V '* be roots. The proof of the following lemma is left to the reader.

**Lemma 2.1** *If F, x*0 *←→ G, v*0 *and F', x'*0 *←→ G', v'*0 *then F ⊗ F',* (*x*0*, x'*0) *←→ G⊗ G',* (*v*0*, v'*0)*.*

By *F ⊕ F'*, we denote the asynchronous product of *F* and *F'*, i.e. the frame *F''*

= (*W '', R''*) over *PG* where

* *W''* = *W × W'*,
* *R''* is the function from *W''*2 to 2*PG* such that for all (*x, x'*)*,* (*y, y'*) *∈ W''*, *R''*((*x, x'*)*,* (*y, y'*)) is the set of all *a ∈ PG* such that *a ∈ R*(*x, y*) and *x'* = *y'* or *x* = *y* and *a ∈ R'*(*x', y'*).

Let *G* = (*V, E*), *G'* = (*V ', E'*) be frames over *P G*, *x*0 *∈ W* , *x'*0 *∈ W'*, *v*0 *∈ V* , *v'*0 *∈*

*V '* be roots. The proof of the following lemma is left to the reader.

**Lemma 2.2** *If F, x*0 *←→ G, v*0 *and F', x'*0 *←→ G', v'*0 *then F ⊕ F',* (*x*0*, x'*0) *←→* *G⊕ G',* (*v*0*, v'*0)*.*

# Controller synthesis and orchestrator synthesis

This section presents our controller/orchestrator synthesis problems. Let us consider a finite set *PG* of program variables.

* 1. *Decision problems*

Let *F* = (*W, R*), *F'* = (*W', R'*) be finite frames over *P G*, *x*0 *∈ W* , *x'*0 *∈ W'* be roots. Given a frame *F''* = (*W '', R''*) over *P G*, a root *x''*0 *∈ W''*, we say that (*F'', x''*0) controls (*F', x'*0) within (*F, x*0) iff *F, x*0 *←→ F' ⊗ F'',* (*x'*0*, x''*0). The synthesis of controllers is the following decision problem:

(*SC*) Given a finite set *PG* of program variables, finite frames *F* = (*W, R*), *F'* = (*W', R'*) over *P G*, roots *x*0 *∈ W* , *x'*0 *∈ W'*, determine whether there exists a frame *F''* = (*W '', R''*) over *P G*, a root *x''*0 *∈ W''* such that (*F'', x''*0) controls (*F', x'*0) within (*F, x*0).

Given a frame *F''* = (*W '', R''*) over *P G*, a root *x''*0 *∈ W''*, we say that (*F'', x''*0) orchestrates (*F', x'*0) within (*F, x*0) iff *F, x*0 *←→ F' ⊕F'',* (*x'*0*, x''*0). The synthesis of orchestrators is the following decision problem:

(*SO*) Given a finite set *PG* of program variables, finite frames *F* = (*W, R*), *F'* = (*W', R'*) over *P G*, roots *x*0 *∈ W* , *x'*0 *∈ W'*, determine whether there exists a frame *F''* = (*W '', R''*) over *P G*, a root *x''*0 *∈ W''* such that (*F'', x''*0) orchestrates (*F', x'*0) within (*F, x*0).

(*SC*) and (*SO*) are deeply related to several important topics considered in the theory of controller synthesis [[3](#_bookmark18),[11](#_bookmark27),[14](#_bookmark30)] and in the theory of orchestrator synthe- sis [[4](#_bookmark20),[5](#_bookmark22),[6](#_bookmark23)]. In the theory of controller synthesis, the basic problem is to restrict, by

means of a controller, the behaviours of a given transition system, the reactive sys- tem to be controlled, so that it satisfies the given control objective. In the theory of orchestrator synthesis, the basic problem is to enhance, by means of an orchestrator, the behaviours of a given distributed system, the multiagent system to be orches- trated, so that it satisfies the given orchestration objective. In [[3](#_bookmark18)] and [[6](#_bookmark23)], methods consisting in transforming every instance of the controller synthesis problem or the orchestrator synthesis problem into an instance of the formula satisfiability problem in *μ*-calculus or the formula satisfiability problem in propositional dynamic logic are proposed. What sections [4](#_bookmark5) and [5](#_bookmark16) show is that alternative methods based on variants of the filtration method are adequate for solving (*SC*) and (*SO*).

* 1. *Bisimulations and products*

Let *F''*

= (*W '', R''*), *F''*

= (*W '', R''*) be frames over *P G*, *x''*

*∈ W''*, *x''*

*∈ W''*

1 1 1 2 2 2 1 1 2 2

be roots. The following lemma demonstrates that control and orchestration are invariant under bisimulations.

**Lemma 3.1** *If F'', x'' ←→ F'', x''*

*then for all ﬁnite frames F* = (*W, R*)*, F'* =

1 1 2 2

(*W', R'*) *over P G, for all roots x*0 *∈ W, x'*0 *∈ W',*

* (*F'', x''*) *controls* (*F', x'*0) *within* (*F, x*0) *iff* (*F'', x''*) *controls* (*F', x'*0) *within*

1 1 2 2

(*F, x*0)*,*

* (*F'', x''*) *orchestrates* (*F', x'*0) *within* (*F, x*0) *iff* (*F'', x''*) *orchestrates* (*F', x'*0)

1 1 2 2

*within* (*F, x*0)*.*

**Proof.** By lemmas [2.1](#_bookmark1) and [2.2](#_bookmark2).

We say that

* *F'', x''* and *F'', x''* are control-equivalent, in symbols *F'', x'' ≡c F'', x''*, iff for all

1 1 2 2 1 1 2 2

finite frames *F* = (*W, R*), *F'* = (*W', R'*) over *PG*, for all roots *x*0 *∈ W* , *x'*0 *∈*

*W'*, (*F'', x''*) controls (*F', x'*0) within (*F, x*0) iff (*F'', x''*) controls (*F', x'*0) within

1 1 2 2

(*F, x*0),

* *F'', x''* and *F'', x''* are orchestration-equivalent, in symbols *F'', x'' ≡o F'', x''*, iff

1 1 2 2 1 1 2 2

for all finite frames *F* = (*W, R*), *F'* = (*W', R'*) over *P G*, for all roots *x*0 *∈ W* ,

*x'*0 *∈ W'*, (*F'', x''*) orchestrates (*F', x'*0) within (*F, x*0) iff (*F'', x''*) orchestrates

1 1 2 2

(*F', x'*0) within (*F, x*0).

The Hennessy-Milner theorem [[7](#_bookmark24)] states that modally equivalent image-finite mod- els are bisimilar. The following lemmas show that control-equivalent frames are bisimilar and orchestration-equivalent frames are bisimilar.

**Lemma 3.2** *If F'', x'' ≡c F'', x'' then F'', x'' ←→ F'', x''.*

1 1 2 2 1 1 2 2

**Proof.** Suppose that *F'* = (*W', R'*) is a finite frame over *P G*, *x'*0 *∈ W'* is a root such that

* *R'* is the function from *W'*2 to 2*PG* such that

*·* for all *a ∈ P G*, for all *x', y' ∈ W'*, *x' R'a y'*.

The reader may easily verify that *F'', x'' ←→ F' ⊗ F'',* (*x'*0*, x''*) and *F'', x'' ←→*

1 1 1 1 2 2

*F' ⊗ F'',* (*x'*0*, x''*). Hence, *F'', x'' ←→ F'', x''*.

2 2 1 1 2 2

**Lemma 3.3** *If F'', x'' ≡o F'', x'' then F'', x'' ←→ F'', x''.*

1 1 2 2 1 1 2 2

**Proof.** Suppose that *F'* = (*W', R'*) is a finite frame over *P G*, *x'*0 *∈ W'* is a root such that

* *R'* is the function from *W'*2 to 2*PG* such that

*·* for all *a ∈ P G*, for all *x', y' ∈ W'*, not *x' R'a y'*.

The reader may easily verify that *F'', x'' ←→ F' ⊕ F'',* (*x'*0*, x''*) and *F'', x'' ←→*

1 1 1 1 2 2

*F' ⊕ F'',* (*x'*0*, x''*). Hence, *F'', x'' ←→ F'', x''*.

2 2 1 1 2 2

* 1. *Deterministic/serial frames*

Suppose that we are given finite frames *F* = (*W, R*), *F'* = (*W', R'*) over *PG*, roots *x*0 *∈ W* , *x'*0 *∈ W'*. The following lemma shows that determining whether there exists a controller of (*F', x'*0) within (*F, x*0) becomes easier if *F* is deterministic or *F'* is deterministic.

**Lemma 3.4** *If F is deterministic or F' is deterministic then there exists a frame F''* = (*W '', R''*) *over P G, a root x''*0 *∈ W'' such that* (*F'', x''*0) *controls* (*F', x'*0) *within* (*F, x*0) *iff* (*F, x*0) *controls* (*F', x'*0) *within* (*F, x*0)*.*

**Proof.** Let *F''* = (*W '', R''*) be a frame over *P G*, *x''*0 *∈ W''* be a root such that (*F'', x''*0) controls (*F', x'*0) within (*F, x*0). Hence, there exists a bisimulation *Z* between *F* and *F' × F''* such that *Z*: *F, x*0 *←→ F' ⊗ F'',* (*x'*0*, x''*0). Let *Zs ⊆ W ×* (*W' × W* ) be the binary relation such that for all *x*1 *∈ W* , for all (*x', x*2) *∈ W' × W* , *x*1 *Zs* (*x', x*2) iff there exists *x ∈ W* , there exists *x'' ∈ W''* such that *x*1 = *x*, *x*2 = *x* and *x Z* (*x', x''*). We demonstrate that *Zs*: *F ←→ F' ⊗ F*. Let *x*1 *∈ W* , (*x', x*2) *∈ W' × W* be such that *x*1 *Zs* (*x', x*2).

Let *a ∈ P G*, *y ∈ W* be such that *x*1 *Ra y*. Since *x*1 *Zs* (*x', x*2), then there exists *x ∈ W* , there exists *x'' ∈ W''* such that *x*1 = *x*, *x*2 = *x* and *x Z* (*x', x''*). Since *x*1 *Ra y*, then *x Ra y*. Since *x*2 = *x*, then *x*2 *Ra y*. Since *x Ra y* and *x Z* (*x', x''*), then there exists (*y', y''*) *∈ W' ×W''* such that *x' R'a y'*, *x'' R''a y''* and *y Z* (*y', y''*). Hence, *y Zs* (*y', y*).

Let *a ∈ P G*, (*y', y*) *∈ W' × W* be such that *x' R'a y'* and *x*2 *Ra y*. Since *x*1 *Zs* (*x', x*2), then there exists *x ∈ W* , there exists *x'' ∈ W''* such that *x*1 = *x*, *x*2 = *x* and *x Z* (*x', x''*). Since *x*2 *Ra y*, then *x Ra y*. Since *x*1 = *x*, then *x*1 *Ra y*. Since *x Z* (*x', x''*), then there exists (*z', z''*) *∈ W' × W''* such that *x' R'a z'*, *x'' R''a z''* and *y Z* (*z', z''*). Since *x' R'a y'* and *x Z* (*x', x''*), then there exists *z ∈ W* such that *x Ra z* and *z Z* (*y', z''*). If *F* is deterministic then *y* = *z*. Since *z Z* (*y', z''*), then *y Zs* (*y', y*). If *F'* is deterministic then *y'* = *z'*. Since *y Z* (*z', z''*), then *y Zs* (*y', y*).

As a result,

**Proposition 3.5** *If one considers instances* (*PG, F, F', x*0*, x'*0) *of* (*SC*) *such that*

*F is deterministic or F' is deterministic then* (*SC*) *is in PTIME.*

The following lemma shows that determining whether there exists an orchestra- tor of (*F', x'*0) within (*F, x*0) becomes easier if *F* is serial or *F'* is serial.

**Lemma 3.6** *If F is serial or F' is serial then there exists a frame F''* = (*W '', R''*) *over P G, a root x''*0 *∈ W'' such that* (*F'', x''*0) *orchestrates* (*F', x'*0) *within* (*F, x*0) *iff* (*F, x*0) *orchestrates* (*F', x'*0) *within* (*F, x*0)*.*

**Proof.** The proof is similar to the proof of lemma [3.4](#_bookmark4).

As a result,

**Proposition 3.7** *If one considers instances* (*PG, F, F', x*0*, x'*0) *of* (*SO*) *such that*

*F is serial or F' is serial then* (*SO*) *is in PTIME.*

1. **Deciding** (*SC*)

In this section, we show that (*SC*) is in *EXPTIME* . We demonstrate the existence of an *EXPTIME* algorithm using filtration.

* 1. *Synchronous ﬁltration*

We now establish a simple algorithm for solving (*SC*). This simple algorithm is based on a variant of the filtration method [[7](#_bookmark24)]. Suppose that we are given a finite set *PG* of program variables, finite frames *F* = (*W, R*), *F'* = (*W', R'*) over *PG*, roots *x*0 *∈ W* , *x'*0 *∈ W'*. Let *F''* = (*W '', R''*) be a frame over *P G*, *x''*0 *∈ W''* be a root such that *F, x*0 *←→ F' ⊗ F'',* (*x'*0*, x''*0). Hence, there exists a bisimulation *Z* between *F* and *F' ⊗F''* such that *x*0 *Z* (*x'*0*, x''*0). Let *≡ ⊆ W'' × W''* be the binary relation such that for all *x'', x'' ∈ W''*,

1 2

* *x'' ≡ x''* iff for all *x ∈ W* , for all *x' ∈ W'*, *x Z* (*x', x''*) iff *x Z* (*x', x''*).

1 2 1 2

Note that *≡* is an equivalence relation. Let *x'' ∈ W''*. The set of all states in *W''* equivalent to *x''* modulo *≡*, in symbols *| x'' |*, is called the equivalence class of *x''* in *W''* modulo *≡* with *x''* as its representative. The set of all equivalence classes of *W''* modulo *≡*, in symbols *W''/ ≡*, is called the quotient set of *W''* modulo *≡*. Suppose that *Ff* = (*Wf , Rf* ) is a frame over *PG* such that

* *Wf* = *W''/ ≡*,
* *Rf* is a function from *Wf* 2 to 2*PG* such that
  + for all *a ∈ P G*, for all *x'', y'' ∈ W''*, if there exists *z'', t'' ∈ W''* such that *x'' ≡*

*z''*, *y'' ≡ t''* and *z'' R''a t''* then *| x'' | Rf a | y'' |*,

* + for all *a ∈ P G*, for all *x'', y'' ∈ W''*, if *| x'' | Rf a | y'' |* then for all *x ∈ W* , for all *x', y' ∈ W'*, if *x' R'a y'* and *x Z* (*x', x''*) then there exists *y ∈ W* such that *x Ra y* and *y Z* (*y', y''*).

Then *Ff* is called a filtration of *F''* through *F* and *F'*. Remark that *Card*(*Wf* ) *≤* 2*Card*(*W* )*×Card*(*W '*). Let *Zf ⊆ W ×* (*W' × Wf* ) be the binary relation such that for all *x ∈ W* , for all (*x', | x'' |*) *∈ W' × Wf* , *x Zf* (*x', | x'' |*) iff *x Z* (*x', x''*). It is a simple matter to check that

**Lemma 4.1** *Zf : F ←→ F' ⊗ Ff .*

**Proof.** Let *x ∈ W* , (*x', | x'' |*) *∈ W' × Wf* be such that *x Zf* (*x', | x'' |*). Hence, *x Z* (*x', x''*).

Let *a ∈ P G*, *y ∈ W* be such that *x Ra y*. We demonstrate that there exists (*y', | y'' |*) *∈ W' × Wf* such that *x' R'a y'*, *| x'' | Rf a | y'' |* and *y Zf* (*y', | y'' |*).

Since *x Z* (*x', x''*), then there exists (*y', y''*) *∈ W' × W''* such that *x' R'a y'*, *x'' R''a*

*y''* and *y Z* (*y', y''*). Hence, there exists (*y', | y'' |*) *∈ W' × Wf* such that *x' R'a y'*,

*| x'' | Rf a | y'' |* and *y Zf* (*y', | y'' |*).

Let *a ∈ P G*, (*y', | y'' |*) *∈ W' × Wf* be such that *x' R'a y'* and *| x'' | Rf a | y'' |*. We demonstrate that there exists *y ∈ W* such that *x Ra y* and *y Zf* (*y', | y'' |*). Since *x Z* (*x', x''*), then there exists *y ∈ W* such that *x Ra y* and *y Z* (*y', y''*). Hence, there exists *y ∈ W* such that *x Ra y* and *y Zf* (*y', | y'' |*).

Hence,

**Lemma 4.2** *F, x*0 *←→ F' ⊗ Ff ,* (*x'*0*, | x''*0 *|*)*.*

There are at least two ways to define functions *Rf* from *Wf* 2 to 2*PG* that fulfil

the required conditions. Define the functions *Rf* and *Rf* from *Wf* 2 to 2*PG* as

*inf*

*sup*

follows:

* for all *a ∈ P G*, for all *| x'' |, | y'' | ∈ Wf* , *a ∈ Rf*

*inf*

(*| x'' |, | y'' |*) iff there exists

*z'', t'' ∈ W''* such that *x'' ≡ z''*, *y'' ≡ t''* and *a ∈ R''*(*z'', t''*),

* for all *a ∈ P G*, for all *| x'' |, | y'' | ∈ Wf* , *a ∈ Rf* (*| x'' |, | y'' |*) iff for all *x ∈ W* ,

*sup*

for all *x', y' ∈ W'*, if *a ∈ R'*(*x', y'*) and *x Z* (*x', x''*) then there exists *y ∈ W* such that *a ∈ R*(*x, y*) and *y Z* (*y', y''*).

## Lemma 4.3 *Rf*

*inf*

*and Rf*

*satisfy the two conditions of a ﬁltration.*

**Proof.** By definition, *Rf* satisfies the first condition of a filtration.

*sup*

*inf*

Let *a ∈ P G*, *x'', y'' ∈ W''* be such that *| x'' | Rf*

*inf*

*a*

*| y'' |*. We demonstrate that

for all *x ∈ W* , for all *x', y' ∈ W'*, if *x' R'a y'* and *x Z* (*x', x''*) then there exists *y ∈*

*W* such that *x Ra y* and *y Z* (*y', y''*). Let *x ∈ W* , *x', y' ∈ W'* be such that *x' R'a y'*

and *x Z* (*x', x''*). Since *| x'' | Rf*

*inf*

*a*

*| y'' |*, then there exists *z'', t'' ∈ W''* such that

*x'' ≡ z''*, *y'' ≡ t''* and *a ∈ R''*(*z'', t''*). Since *x Z* (*x', x''*), then *x Z* (*x', z''*). Since *x' R'a y'* and *a ∈ R''*(*z'', t''*), then there exists *y ∈ W* such that *x Ra y* and *y Z* (*y', t''*). Since *y'' ≡ t''*, then *y Z* (*y', y''*).

Let *a ∈ P G*, *x'', y'' ∈ W''* be such that there exists *z'', t'' ∈ W''* such that *x'' ≡*

*supa*

*z''*, *y'' ≡ t''* and *z'' R''a*

*t''*. We demonstrate that *| x'' | Rf*

*| y'' |*. Let *x ∈ W* ,

*x', y' ∈ W'* be such that *a ∈ R'*(*x', y'*) and *x Z* (*x', x''*). Since *x'' ≡ z''*, then *x Z* (*x', z''*). Since *a ∈ R'*(*x', y'*) and *z'' R''a t''*, then there exists *y ∈ W* such that *a ∈ R*(*x, y*) and *y Z* (*y', t''*). Since *y'' ≡ t''*, then *y Z* (*y', y''*).

By definition, *Rf* satisfies the second condition of a filtration.

*sup*

*sup*

**Lemma 4.4** *For all | x'' |, | y'' | ∈ Wf , Rf*

*inf*

(*| x'' |, | y'' |*) *⊆ Rf*

(*| x'' |, | y'' |*)*.*

**Proof.** Let *a ∈ PG* be such that *a ∈ Rf* (*| x'' |, | y'' |*). We demonstrate that *a*

*inf*

*f sup*

*∈ R*

(*| x'' |, | y'' |*). Let *x ∈ W* , *x', y' ∈ W'* be such that *a ∈ R'*(*x', y'*) and *x Z*

(*x', x''*). We demonstrate that there exists *y ∈ W* such that *a ∈ R*(*x, y*) and *y Z*

(*y', y''*). Since *a ∈ Rf* (*| x'' |, | y'' |*), then there exists *z'', t'' ∈ W''* such that *x''*

*inf*

*≡ z''*, *y'' ≡ t''* and *a ∈ R''*(*z'', t''*). Since *x Z* (*x', x''*), then *x Z* (*x', z''*). Since *a ∈ R'*(*x', y'*) and *a ∈ R''*(*z'', t''*), then there exists *y ∈ W* such that *a ∈ R*(*x, y*) and *y* *Z* (*y', t''*). Since *y'' ≡ t''*, then there exists *y ∈ W* such that *a ∈ R*(*x, y*) and *y Z* (*y', y''*).

From the discussion above, it follows that the functions *Rf*

*inf*

and *Rf*

from *Wf* 2

to 2*PG* give respectively rise to the least filtration *Ff*

*sup*

*inf*

*sup*

= (*Wf , Rf*

) of *F''* through

*F* and *F'* and the greatest filtration *Ff*

*inf*

*sup*

= (*Wf , Rf*

) of *F''* through *F* and *F'*.

* 1. *Complexity of* (*SC*)

In this section, we show how the synchronous filtration can be used for deciding (*SC*).

* + 1. *A nondeterministic exponential-time algorithm*

For our purpose, the crucial property of the above notion of synchronous filtration is the following: *Card*(*Wf* ) *≤* 2*Card*(*W* )*×Card*(*W '*). Hence, we can give a simple algorithm for solving (*SC*): guess a frame *F''* = (*W '', R''*) over *PG* such that

*Card*(*W''*) *≤* 2*Card*(*W* )*×Card*(*W '*), guess a root *x'' ∈ W''* and determine whether

0

*F, x*0 *←→ F' ⊗ F'',* (*x'*0*, x''*0). Not surprisingly, the above algorithm returns the value *true* iff there exists a frame *F''* = (*W '', R''*) over *PG*, a root *x''*0 *∈ W''* such that *F, x*0 *←→ F' ⊗ F'',* (*x'*0*, x''*0). Seeing that determining whether *F, x*0 *←→ F' ⊗ F'',* (*x'*0*, x''*0) can be done in polynomial time [[1](#_bookmark21)], it follows immediately that

**Proposition 4.5** (*SC*) *is in NEXPTIME.*

* + 1. *A deterministic exponential-time algorithm*

The truth of the matter is that (*SC*) is in *EXPTIME* . This can be proved as follows. Suppose that we are given a finite set *PG* of program variables, finite frames *F* = (*W, R*), *F'* = (*W', R'*) over *P G*, roots *x*0 *∈ W* , *x'*0 *∈ W'*. Let *F''* = (*W '', R''*) be a frame over *P G*, *x''*0 *∈ W''* be a root such that *F, x*0 *←→ F' ⊗F'',* (*x'*0*, x''*0). Hence, there exists a bisimulation *Z* between *F* and *F' ⊗F''* such that *x*0 *Z* (*x'*0*, x''*0). Let *f* be the function from *W''* to 2*W ×W '* such that for all *x'' ∈ W''*,

* + - * *f* (*x''*) = *{*(*x, x'*) *∈ W × W'*: *x Z* (*x', x''*)*}*.

By definition, for all *x'', x'' ∈ W''*, if *f* (*x''*) = *f* (*x''*) then *x'' ≡ x''*. Suppose that

1 2 1 2 1 2

*Fff* = (*Wff , Rff* ) is the frame over *PG* such that

* + - * *Wff* = *f* (*W ''*),
      * *Rff* is the function from *Wff* 2 to 2*PG* such that

*·* for all *a ∈ P G*, for all *x'', y'' ∈ W''*, *f* (*x''*) *Rff a f* (*y''*) iff for all *x ∈ W* , for all

*x', y' ∈ W'*, if *x' R'a y'* and *x Z* (*x', x''*) then there exists *y ∈ W* such that *x Ra y* and *y Z* (*y', y''*).

Let *Zff ⊆ W ×* (*W' × Wff* ) be the binary relation such that for all *x ∈ W* , for all (*x',f* (*x''*)) *∈ W' × Wff* , *x Zff* (*x',f* (*x''*)) iff *x Zf* (*x', | x'' |*). It is a simple matter to check that

**Lemma 4.6** *Zff : F ←→ F' ⊗ Fff .*

**Proof.** By lemmas [4.1](#_bookmark7) and [4.3](#_bookmark8).

Hence,

**Lemma 4.7** *F, x*0 *←→ F' ⊗ Fff ,* (*x'*0*,f* (*x''*0))*.*

We now construct a sequence *Fi* = (*Wi, Ri*), *i ≥* 0, of frames over *PG* approx- imating *Fff* = (*Wff , Rff* ) and a sequence *Zi ⊆ W ×* (*W' × Wi*), *i ≥* 0, of binary relations approximating *Zff* .

Let *F*0 = (*W* 0*, R*0) be the frame over *PG* such that

* *W* 0 = 2*W ×W '* ,
* *R*0 is the function from *W* 02 to 2*PG* such that
  + for all *a ∈ P G*, for all *x*0*, y*0 *∈ W* 0, *x*0 *R*0*a y*0 iff for all *x ∈ W* , for all *x', y' ∈ W'*, if *x' R'a y'* and (*x, x'*) *∈ x*0 then there exists *y ∈ W* such that *x Ra y* and (*y, y'*) *∈ y*0,

*Z*0 *⊆ W ×* (*W' × W* 0) be the binary relation such that for all *x ∈ W* , for all (*x', x*0)

*∈ W' × W* 0, *x Z*0 (*x', x*0) iff (*x, x'*) *∈ x*0.

Secondly, for all *i ≥* 0, let *Wi* = *{xi ∈ Wi*: there exists *a ∈ P G*, *x ∈ W* , *x' ∈*

*→*

*W'*, *y ∈ W* such that *x Zi* (*x', xi*), *x Ra y* and for all *y' ∈ W'*, for all *yi ∈ Wi*, if *x'*

*R'a y'* and *xi Ria yi* then not *y Zi* (*y', yi*)*}*, *Wi* = *{xi ∈ Wi*: there exists *a ∈ P G*,

*←*

*x ∈ W* , *x' ∈ W'*, *y' ∈ W'*, *yi ∈ Wi* such that *x Zi* (*x', xi*), *x' R'a y'*, *xi Ria yi* and for all *y ∈ W* , if *x Ra y* then not *y Zi* (*y', yi*)*}*, *Fi*+1 = (*Wi*+1*, Ri*+1) be the frame over *PG* such that

* *Wi*+1 = *Wi \* (*Wi ∪ Wi* ),

*→*

*←*

* *Ri*+1 is the function from *Wi*+12 to 2*PG* such that
  + for all *a ∈ P G*, for all *xi*+1*, yi*+1 *∈ Wi*+1, *xi*+1 *Ri*+1*a yi*+1 iff for all *x ∈ W* , for all *x', y' ∈ W'*, if *x' R'a y'* and (*x, x'*) *∈ xi*+1 then there exists *y ∈ W* such that *x Ra y* and (*y, y'*) *∈ yi*+1,

*Zi*+1 *⊆ W ×* (*W' × Wi*+1) be the binary relation such that for all *x ∈ W* , for all (*x', xi*+1) *∈ W' × Wi*+1, *x Zi*+1 (*x', xi*+1) iff (*x, x'*) *∈ xi*+1.

**Lemma 4.8** *For all i ≥* 0*,*

* *Wff ⊆ Wi,*
* *for all xff , yff ∈ Wff , Rff* (*xff , yff* ) *⊆ Ri*(*xff , yff* )*,*
* *Zff ⊆ Zi.*

**Proof.** The proof is by induction on *i ≥* 0. As the reader is asked to show, *Wff ⊆ W* 0, for all *xff , yff ∈ Wff* , *Rff* (*xff , yff* ) *⊆ R*0(*xff , yff* ) and *Zff ⊆ Z*0. Let *i ≥* 0 be such that *Wff ⊆ Wi*, for all *xff , yff ∈ Wff* , *Rff* (*xff , yff* ) *⊆ Ri*(*xff , yff* ) and *Zff ⊆ Zi*. We demonstrate that *Wff ⊆ Wi*+1, for all *xff , yff ∈ Wff* , *Rff* (*xff , yff* )

*⊆ Ri*+1(*xff , yff* ) and *Zff ⊆ Zi*+1.

Let *xff ∈ Wff* . If *xff /∈ Wi*+1 then *xff ∈ Wi* or *xff ∈ Wi* . If *xff ∈ Wi*

*→ ← →*

then there exists *a ∈ P G*, *x ∈ W* , *x' ∈ W'*, *y ∈ W* such that *x Zi* (*x', xff* ), *x Ra y* and for all *y' ∈ W'*, for all *yi ∈ Wi*, if *x' R'a y'* and *xff Ria yi* then not *y Zi* (*y', yi*). Since *x Zi* (*x', xff* ), then (*x, x'*) *∈ xff* . Hence, *x Zff* (*x', xff* ). Since *x Ra y*, then there exists *y' ∈ W'*, *yi ∈ Wi* such that *x' R'a y'* and *xff Ria yi* and *y Zff*

(*y', yi*). Hence, (*y, y'*) *∈ yi*. Hence, *y Zi* (*y', yi*): a contradiction. If *xff ∈ Wi* then

*←*

there exists *a ∈ P G*, *x ∈ W* , *x' ∈ W'*, *y' ∈ W'*, *yi ∈ Wi* such that *x Zi* (*x', xff* ), *x' R'a y'*, *xff Ria yi* and for all *y ∈ W* , if *x Ra y* then not *y Zi* (*y', yi*). Since *x Zi* (*x', xff* ), then (*x, x'*) *∈ xff* . Since *x' R'a y'* and *xff Ria yi*, then there exists *y ∈ W* such that *x Ra y* and (*y, y'*) *∈ yi*. Hence, *y Zi* (*y', yi*): a contradiction.

Let *a ∈ PG* be such that *xff Rff a yff* . We demonstrate that *xff Ri*+1*a yff* . Let *x ∈ W* , *x', y' ∈ W'*, be such that *x' R'a y'* and (*x, x'*) *∈ xff* . Hence, *x Zff* (*x', xff* ). Since *x' R'a y'* and *xff Rff a yff* , then there exists *y ∈ W* such that *x Ra y* and *y Zff* (*y', yff* ). Hence, (*y, y'*) *∈ yff* . Hence, *xff Ri*+1*a yff* .

Let *x ∈ W* , (*x', xff* ) *∈ W' ×Wff* be such that *x Zff* (*x', xff* ). We demonstrate that *x Zi*+1 (*x', xff* ). Since *x Zff* (*x', xff* ), then (*x, x'*) *∈ xff* . Hence, *x Zi*+1 (*x', xff* ).

It follows that there exists *i*0 *≥* 0 such that

**Lemma 4.9** *Zi*0 *: F ←→ F' ⊗ Fi*0 *.*

**Proof.** Since *W* 0 is finite and for all *i ≥* 0, *Wi*+1 *⊆ Wi*, then there exists *i*0 *≥* 0 such that *Wi*0+1 = *Wi*0 . Since for all *i ≥* 0, *Wff ⊆ Wi*, then *Wi*0 is nonempty.

Hence, *Fi*0 = (*Wi*0 *, Ri*0 ) is a frame over *P G*. Since *Wi*0+1 = *Wi*0 , then *Wi*0 = *∅*

*→*

and *Wi*0 = *∅*. Hence, *Zi*0 : *F ←→ F' ⊗ Fi*0 .

*←*

Hence,

**Lemma 4.10** *F, x*0 *←→ F' ⊗ Fi*0 *,* (*x'*0*,f* (*x''*0))*.*

The above construction has the following property. When applied to an arbitrary finite set *PG* of program variables, arbitrary finite frames *F* = (*W, R*), *F'* = (*W', R'*) over *P G*, arbitrary roots *x*0 *∈ W* , *x'*0 *∈ W'*, it stops with a frame *Fi*0 = (*Wi*0 *, Ri*0 ) over *PG* and a binary relation *Zi*0 *⊆ W ×* (*W' × Wi*0 ) such that if *Wi*0 */*= *∅* then *Zi*0 : *F ←→ F' ⊗ Fi*0 . Hence, we can give a simple algorithm for solving (*SC*):

* For all *i ≥* 0, construct the frame *Fi* = (*Wi, Ri*) over *PG* and the binary relation

*Zi ⊆ W ×* (*W' × Wi*) as above until if *Wi /*= *∅* then *Zi*: *F ←→ F' ⊗ Fi*.

Not surprisingly, the above algorithm returns the value *true* iff there exists a frame *F''* = (*W '', R''*) over *PG*, a root *x''*0 *∈ W''* such that *F, x*0 *←→ F' ⊗F'',* (*x'*0*, x''*0). Seeing that *F*0 and *Z*0 can be constructed in exponential time and for all *i ≥* 0,

*Fi*+1 and *Zi*+1 can be constructed in time polynomial in the size of *Fi* and *Zi*, it follows immediately that

**Proposition 4.11** (*SC*) *is in EXPTIME .*

1. **Deciding** (*SO*)

In this section, we show that (*SO*) is in *EXPTIME* . We demonstrate the existence of an *EXPTIME* algorithm using filtration.

* 1. *Asynchronous ﬁltration*

We now establish a simple algorithm for solving (*SO*). This simple algorithm is based on a variant of the filtration method [[7](#_bookmark24)] similar to the one used in section [4.1](#_bookmark6). Suppose that we are given a finite set *PG* of program variables, finite frames *F* = (*W, R*), *F'* = (*W', R'*) over *P G*, roots *x*0 *∈ W* , *x'*0 *∈ W'*. Let *F''* = (*W '', R''*) be a frame over *P G*, *x''*0 *∈ W''* be a root such that *F, x*0 *←→ F' ⊕F'',* (*x'*0*, x''*0). Hence, there exists a bisimulation *Z* between *F* and *F' ⊕ F''* such that *x*0 *Z* (*x'*0*, x''*0). Defining *Ff* = (*Wf , Rf* ) and *Zf ⊆ W ×* (*W' × Wf* ) as in section [4.1](#_bookmark6) aside from

the fact that the conditions put on the function *Rf* from *Wf* 2 to 2*PG* are now the

following:

* for all *a ∈ P G*, for all *x'', y'' ∈ W''*, if there exists *z'', t'' ∈ W''* such that *x'' ≡ z''*,

*y'' ≡ t''* and *a ∈ R''*(*z'', t''*) then *a ∈ Rf* (*| x'' |, | y'' |*),

* for all *a ∈ P G*, for all *x'', y'' ∈ W''*, if *a ∈ Rf* (*| x'' |, | y'' |*) then for all *x ∈ W* , for all *x', y' ∈ W'*, if *x'* = *y'* and *x Z* (*x', x''*) then there exists *y ∈ W* such that *a ∈ R*(*x, y*) and *y Z* (*y', y''*),

it is a simple matter to check that

**Lemma 5.1** *Zf : F ←→ F' ⊕ Ff .*

**Proof.** The proof is similar to the proof of lemma [4.1](#_bookmark7).

Hence,

**Lemma 5.2** *F, x*0 *←→ F' ⊕ Ff ,* (*x'*0*, | x''*0 *|*)*.*

As in section [4.1](#_bookmark6), there are at least two ways to define functions *Rf* from *Wf* 2

to 2*PG* that fulfil the required conditions:

* for all *a ∈ P G*, for all *| x'' |, | y'' | ∈ Wf* , *a ∈ Rf* (*| x'' |, | y'' |*) iff there exists

*inf*

*z'', t'' ∈ W''* such that *x'' ≡ z''*, *y'' ≡ t''* and *a ∈ R''*(*z'', t''*),

* for all *a ∈ P G*, for all *| x'' |, | y'' | ∈ Wf* , *a ∈ Rf* (*| x'' |, | y'' |*) iff for all *x ∈ W* ,

*sup*

for all *x', y' ∈ W'*, if *x'* = *y'* and *x Z* (*x', x''*) then there exists *y ∈ W* such that

*a ∈ R*(*x, y*) and *y Z* (*y', y''*).

## Lemma 5.3 *Rf*

*inf*

*and Rf*

*satisfy the two conditions of a ﬁltration.*

**Proof.** The proof is similar to the proof of lemma [4.3](#_bookmark8).

*sup*

**Lemma 5.4** *For all | x'' |, | y'' | ∈ Wf , Rf*

*inf*

(*| x'' |, | y'' |*) *⊆ Rf*

(*| x'' |, | y'' |*)*.*

**Proof.** The proof is similar to the proof of lemma [4.4](#_bookmark9).

*sup*

From the discussion above, it follows that the functions *Rf*

*inf*

and *Rf*

from *Wf* 2

to 2*PG* give respectively rise to the least filtration *Ff*

*sup*

*inf*

*sup*

= (*Wf , Rf*

) of *F''* through

*F* and *F'* and the greatest filtration *Ff*

*inf*

*sup*

= (*Wf , Rf*

) of *F''* through *F* and *F'*.

* 1. *Complexity of* (*SO*)

In this section, we show how the asynchronous filtration can be used for deciding (*SO*).

* + 1. *A nondeterministic exponential-time algorithm*

For our purpose, the crucial property of the above notion of synchronous filtration is the following: *Card*(*Wf* ) *≤* 2*Card*(*W* )*×Card*(*W '*). Hence, we can give a simple nondeterministic exponential-time algorithm for solving (*SO*) similar to the one considered in section [4.2.1](#_bookmark11).

**Proposition 5.5** (*SO*) *is in NEXPTIME.*

* + 1. *A deterministic exponential-time algorithm*

The truth of the matter is that (*SO*) is in *EXPTIME* . This can be proved in a way similar to the one followed in section [4.2.2](#_bookmark12). Defining *Fff* = (*Wff , Rff* ) and *Zff*

*⊆ W ×* (*W' × Wff* ) as in section [4.2](#_bookmark10) aside from the fact that the definition of the function *Rff* from *Wff* 2 to 2*PG* is now the following:

* + - * for all *a ∈ P G*, for all *x'', y'' ∈ W''*, *f* (*x''*) *Rff a f* (*y''*) iff for all *x ∈ W* , for all *x', y' ∈ W'*, if *x'* = *y'* and *x Z* (*x', x''*) then there exists *y ∈ W* such that *x Ra y* and *y Z* (*y', y''*),

it is a simple matter to check that

**Lemma 5.6** *Zff : F ←→ F' ⊕ Fff .*

**Proof.** The proof is similar to the proof of lemma [4.6](#_bookmark13).

Hence,

**Lemma 5.7** *F, x*0 *←→ F' ⊕ Fff ,* (*x'*0*,f* (*x''*0))*.*

We now construct a sequence *Fi* = (*Wi, Ri*), *i ≥* 0, of frames over *PG* approx- imating *Fff* = (*Wff , Rff* ) and a sequence *Zi ⊆ W ×* (*W' × Wi*), *i ≥* 0, of binary relations approximating *Zff* .

Let *F*0 = (*W* 0*, R*0) be the frame over *PG* such that

* + - * *W* 0 = 2*W ×W '* ,
      * *R*0 is the function from *W* 02 to 2*PG* such that

*·* for all *a ∈ P G*, for all *x*0*, y*0 *∈ W* 0, *x*0 *R*0*a y*0 iff for all *x ∈ W* , for all *x', y' ∈ W'*, if *x'* = *y'* and (*x, x'*) *∈ x*0 then there exists *y ∈ W* such that *x Ra y* and (*y, y'*) *∈ y*0,

*Z*0 *⊆ W ×* (*W' × W* 0) be the binary relation such that for all *x ∈ W* , for all (*x', x*0)

*∈ W' × W* 0, *x Z*0 (*x', x*0) iff (*x, x'*) *∈ x*0.

Secondly, for all *i ≥* 0, let *Wi* = *{xi ∈ Wi*: there exists *a ∈ P G*, *x ∈ W* , *x' ∈*

*→*

*W'*, *y ∈ W* such that *x Zi* (*x', xi*), *x Ra y* and for all *y' ∈ W'*, for all *yi ∈ Wi*, if *x'*

*R'a y'* and *xi* = *yi* or *x'* = *y'* and *xi Ria yi* then not *y Zi* (*y', yi*)*}*, *Wi* = *{xi ∈ Wi*:

*←*

there exists *a ∈ P G*, *x ∈ W* , *x' ∈ W'*, *y' ∈ W'*, *yi ∈ Wi* such that *x Zi* (*x', xi*), *x' R'a y'* and *xi* = *yi* or *x'* = *y'* and *xi Ria yi* and for all *y ∈ W* , if *x Ra y* then not *y Zi* (*y', yi*)*}*, *Fi*+1 = (*Wi*+1*, Ri*+1) be the frame over *PG* such that

* *Wi*+1 = *Wi \* (*Wi ∪ Wi* ),

*→*

*←*

* *Ri*+1 is the function from *Wi*+12 to 2*PG* such that

*·* for all *a ∈ P G*, for all *xi*+1*, yi*+1 *∈ Wi*+1, *xi*+1 *Ri*+1*a yi*+1 iff for all *x ∈ W* , for all *x', y' ∈ W'*, if *x'* = *y'* and (*x, x'*) *∈ xi*+1 then there exists *y ∈ W* such that *x Ra y* and (*y, y'*) *∈ yi*+1,

*Zi*+1 *⊆ W ×* (*W' × Wi*+1) be the binary relation such that for all *x ∈ W* , for all (*x', xi*+1) *∈ W' × Wi*+1, *x Zi*+1 (*x', xi*+1) iff (*x, x'*) *∈ xi*+1.

**Lemma 5.8** *For all i ≥* 0*,*

* *Wff ⊆ Wi,*
* *for all xff , yff ∈ Wff , Rff* (*xff , yff* ) *⊆ Ri*(*xff , yff* )*,*
* *Zff ⊆ Zi.*

**Proof.** The proof is similar to the proof of lemma [4.8](#_bookmark14).

It follows that there exists *i*0 *≥* 0 such that

**Lemma 5.9** *Zi*0 *: F ←→ F' ⊕ Fi*0 *.*

**Proof.** The proof is similar to the proof of lemma [4.9](#_bookmark15).

Hence,

**Lemma 5.10** *F, x*0 *←→ F' ⊕ Fi*0 *,* (*x'*0*,f* (*x''*0))*.*

The above construction has the following property. When applied to an arbitrary finite set *PG* of program variables, arbitrary finite frames *F* = (*W, R*), *F'* = (*W', R'*) over *P G*, arbitrary roots *x*0 *∈ W* , *x'*0 *∈ W'*, it stops with a frame *Fi*0 = (*Wi*0 *, Ri*0 ) over *PG* and a binary relation *Zi*0 *⊆ W ×* (*W' × Wi*0 ) such that if *Wi*0 */*= *∅* then *Zi*0 : *F ←→ F' ⊕ Fi*0 . Hence, we can give a simple algorithm for solving (*SO*):

* For all *i ≥* 0, construct the frame *Fi* = (*Wi, Ri*) over *PG* and the binary relation

*Zi ⊆ W ×* (*W' × Wi*) as above until if *Wi /*= *∅* then *Zi*: *F ←→ F' ⊕ Fi*.

Not surprisingly, the above algorithm returns the value *true* iff there exists a frame

*F''* = (*W '', R''*) over *PG*, a root *x''*0 *∈ W''* such that *F, x*0 *←→ F' ⊕F'',* (*x'*0*, x''*0).

Seeing that *F*0 and *Z*0 can be constructed in exponential time and for all *i ≥* 0, *Fi*+1 and *Zi*+1 can be constructed in time polynomial in the size of *Fi* and *Zi*, it follows immediately that

**Proposition 5.11** (*SO*) *is in EXPTIME .*

# Conclusion

We have considered the decision problems (*SC*) and (*SO*) of controller/orchestra- tor synthesis. Deterministic algorithms that check in exponential-time whether a controller/orchestrator exists have been proposed. An interesting (and still open) question is to evaluate the exact complexity of (*SC*) and (*SO*). Let us remark that the following decision problem is known to be *EXPTIME* -hard: given a finite set *PG* of program variables, deterministic finite frames *F* = (*W, R*), *F*1 = (*W* 1*, R*1),

*.. .*, *Fn* = (*Wn, Rn*) over *P G*, roots *x*0 *∈ W* , *x*10 *∈ W* 1, *.. .*, *xn*0 *∈ Wn*, determine if *F, x*0 is simulated by *F*1 *⊕ ... ⊕ Fn,* (*x*10*,..., xn*0). See [[12](#_bookmark29)] for details. Are (*SC*) and (*SO*) *EXPTIME* -hard too? If (*SC*) and (*SO*) prove to be *EXPTIME* -hard too then we doubt the practicality of any decision method for them. In this re- spect, the use of symbolic techniques should permit to reduce the practical cost of controller/orchestrator synthesis. Possible solutions would demand to use compact data structures for the representation of frames [[8](#_bookmark25)] and to apply the techniques of abstraction and refinement used within the context of computer-aided verifica- tion [[9](#_bookmark26)]. Variants of (*SC*) and (*SO*) can be considered as well. For instance, one may consider that the controller/orchestrator must be transitive, reverse well-founded, etc. For such a variant, although we believe that our filtration approach can pro- vide a solution, the complexity of controller/orchestrator synthesis is still unknown. Take another variant: one may replace “bisimilar” by “trace equivalent”. For such a variant, although Ramadge and Wonham [[14](#_bookmark30)] and Tsitsiklis [[15](#_bookmark31)] have indirectly and partially addressed it, the complexity of controller/orchestrator synthesis is still unknown. Finally, one may involve atomic propositions and do everything on the level of finite models which are a more natural framework for the synthesis prob- lems. Involving atomic propositions can make the synthesis problems much harder, at least in some cases. For instance, every two finite serial frames are bisimilar, hence the synthesis problems in the case of serial frames are trivial; but not so for serial models.

# Acknowledgements

We make a point of thanking the colleagues of the *Institut de recherche en infor- matique de Toulouse* who, by the discussions we had with them, contributed to the development of the work we present today. Special acknowledgement is heartly granted as well to the anonymous referees who made several helpful comments for improving the readability of the paper and suggesting variants of (*SC*) and (*SO*).

# References

1. Alvarez, C., Balc´azar, J., Gabarr´o, J., S´antha, M.: *Parallel complexity in the design and analysis of concurrent systems.* In: *PARLE ’91, Parallel Architectures and Languages Europe.* Springer-Verlag (1991) 288–303.
2. Arnold, A., Niwin´ski, D.: *Rudiments of μ-Calculus.* Elsevier (2001).
3. Arnold, A., Vincent, A., Walukiewicz, I.: *Games for synthesis of controllers with partial observation.*

Theoretical computer science **303** (2003) 7–34.

1. Berardi, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Mecella, M.: *Automatic composition of* *e-services that export their behavior.* In: *Service-Oriented Computing — ICSOC 2003.* Springer (2003) 43–58.
2. Berardi, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Mecella, M.: *Synthesis of underspecified* *composite e-services based on automated reasoning.* In: *ICSOC’04 — Proceedings of the Second International Conference on Service Oriented Computing.* ACM Press (2004) 105–114.
3. Berardi, D., Calvanese, D., De Giacomo, G., Hull, R., Mecella, M.: *Automated composition of transition- based semantic Web services with messaging.* In: *Proceedings of the 31st International Conference on* *Very Large Data Bases.* VLDB Endowment (2005) 613–624.
4. Blackburn, P., de Rijke, M., Venema, Y.: *Modal Logic.* Cambridge University Press (2001).
5. Burch, J., Clarke, E., McMillan, K., Dill, D., Hwang, L.: *Symbolic model checking:* 1020 *states and beyond.* Information and Computation **98** (1992) 142–170.
6. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: *Counterexample-guided abstraction refinement for symbolic model checking.* Journal of the ACM **50** (2003) 752–794.
7. Harel, D., Kozen, D., Tiuryn, J.: *Dynamic Logic.* MIT Press (2000).
8. Maler, O., Pnueli, A., Sifakis, J.: *On the synthesis of discrete controllers for timed systems.* In: *STACS*

*95.* Springer (1995) 229–242.

1. Muscholl, A., Walukiewicz, I.: *A lower bound on Web services composition.* In: *Foundations of Software Science and Computational Structures.* Springer (2007) 274–286.
2. Pistore, M., Marconi, A., Bertoli, P., Traverso, P.: *Automated composition of Web services by planning at the knowledge level.* In: *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence.* International Joint Conferences on Artificial Intelligence (2005) 1252–1259.
3. Ramadge, P., Wonham, W.: *The control of discrete event systems.* In: *Proceedings of the IEEE.* IEEE (1989) 81–98.
4. Tsitsiklis, J.: *On the control of discrete-event dynamical systems.* Mathematics of Control, Signals, and Systems **2** (1989) 95–107.