On the Complexity of Equation Solving in Process Algebra

Bengt Jonsson
Swedish Institute of Computer Science *

Kim Guldstrand Larsen
Aalborg University, Denmark †

Abstract

The problem of designing a system which in a given environment C should satisfy a given specification S can be formulated as "find a system P such that $C(P)$ satisfies the specification S". In process algebra, such problems take the form of equations. We investigate the complexity of solving such equations in process algebra. We consider the problem of deciding whether there is a process P which satisfies an equation of one of the following forms:

$$C(P) \sim Q \qquad C(P) \lhd S \qquad (A \,|\, P)\backslash L \sim B \qquad (A \,|\, P)\backslash L \approx B$$

where C is an arbitrary context of some process algebra, A, B and Q are given processes, S is a modal specification, $\sim$ ($\approx$) is (weak) bisimulation equivalence, $\lhd$ is refinement between modal specifications (a generalization of bisimulation equivalence), and $|$ and $\backslash L$ is the parallel and restriction operator of CCS respectively. The main result is that all four problems are PSPACE-hard in the size of the given contexts, processes and specifications. We also give constraints under which the first and third problem can be solved in polynomial time.

Introduction

One of the most difficult and important problems in computer science is to develop methods for design and construction of concurrent systems. One way of automating the problem is to formulate the design problem as a model construction problem, "find a system P which satisfies a specification S," for which automatic decision procedures can be found. The specification S can for instance be a formula in temporal logic as in [MW84, PR89, CE82], or an abstract system as in [KS].

In this paper, we consider the case where the specification S does not specify the system P directly, but rather P placed in a given environment. Process algebra provides an elegant way to represent such environments formally as contexts. The design problem is then formulated as the problem of finding a system P which satisfies

$$C(P) \text{ sat } S \tag{1}$$

where C is a context representing the given environment, and sat is a suitably chosen satisfaction relation. As an example, S can be an abstract system which specifies a system in which P is an

*Full Address: Bengt Jonsson, Swedish Institute of Computer Science, Box 1263, 164 28 Kista, Sweden. E-mail: bengt@sics.se. Supported in part by the Swedish Board for Technical Development (STU) under contract No. 89-01220P as part of Esprit BRA project SPEC, No. 3096

†Full Address: Kim Guldstrand Larsen, Aalborg University, Department of Mathematics and Computer Science, Fredrik Bajersvej 7, 9220 Aalborg, Denmark. E-mail: kgl@iesd.auc.dk

unknown component which executes in parallel with several other known components, represented by the system A. This problem can be formulated in CCS [Mil89] as finding P which satisfies

$$(A \mid P)\backslash L \textbf{ sat } S \tag{2}$$

and a restriction on which actions P may use. Here L is the set of actions over which A and P communicate. The operation $\mid$ puts two systems in parallel, and the operation $\backslash L$ makes the actions in the set L internal and unobservable,

Methods for solving (2) have been presented by Shields [Shi89], by Parrow [Par89], by Lewis and Qin [LQ90] and by Merlin and Bochmann [MB83]. For the more general problem (1), Larsen and Xinxin [LX90b] have developed a language-independent theory of contexts in process algebra, in which they give a characterization of the solutions of (1) with **sat** being bisimulation equivalence, which induces a single exponential time decision algorithm.

Common to all proposed methods [Shi89, Par89, MB83, LX90b, LQ90] is that the proposed algorithms require exponential time, even in spite of the fact that they impose restrictions on the involved contexts and processes. No restrictions have been presented under which the problem can be solved in polynomial time, and there have not been presented any lower bounds on the complexity of the problem.

In this paper, we establish lower bounds on the complexity of solving equations of form (1) and (2), and also present some restrictions under which the equations can be solved in polynomial time. We consider three different satisfaction relations **sat** in (1): bisimulation equivalence, $\sim$ and weak bisimulation or observational equivalence, $\approx$, with S being a system, and modal refinement, $\lhd$, where S is a modal specification. The equivalences $\sim$ and $\approx$ are well-established and often used satisfaction relations for the correctness of concurrent systems [BK84, Koo85, LM87, Par87, SFD85]. Modal refinement $\lhd$ is a generalization of bisimulation equivalence: a modal specification can distinguish between mandatory and optional transitions, thus allowing more loose specifications [LT88, HL89, Lar90, BL90].

The main decision problems that we consider are the following:

- CcsEq Given (finite-state) systems A and B and a set of actions L, does there exist any process P satisfying the equation $(A \mid P)\backslash L \sim B$.
- CcsObs is the same as CcsEq but for observation equivalence $\approx$.
- Eq Given a (finite-state) context C and a (finite-state) transition system Q, does there exist any process P satisfying the equation $C(P) \sim Q$?
- InEq Given a (finite-state) context C and a (finite-state) modal transition system S, does there exist any process P satisfying the inequation $C(P) \lhd S$?

Also we are concerned with the identification of subclasses of the above problems that are solvable in polynomial time. The results concerning these problems that we obtain are:

- The problems CcsEq and CcsObs are PSPACE-hard in the size of A and B.
- The problems Eq and InEq are PSPACE-hard in the size of C and S (C and Q for Eq), even in the case where S (Q in Eq) is deterministic.
- Under certain conditions on the context C, we obtain a subproblem of Eq which is solvable in polynomial time. This also yields conditions under which the problem CcsEq is solvable in polynomial time.

The lower bounds are obtained through a series of reductions from the known PSPACE–complete graph theoretical problem called Generalized Geography, GENGEO , in [GJ79]. The series of reductions is the following:

$$\text{GENGEO} \longrightarrow \text{INEQ} \longrightarrow \text{EQ} \longrightarrow \text{CCSEQ} \longrightarrow \text{CCSOBS}$$

All problems are PSPACE–hard even when we assume a small fixed-size set of allowed actions for the processes, specifications and contexts. A deterministic exponential time upper bound for all problems can be obtained from the solution method in [LX90b]. It still remains an open problem whether these problems are in PSPACE or not.

Equations of form (2) when **sat** is observation equivalence have been studied by Shields [Shi89], and by Parrow [Par89]. Both of these works impose restrictions on C and B to obtain methods and algorithms for a solution. Our results show that the problems they consider are PSPACE–hard. The case when **sat** is trace equivalence has been treated by Merlin and Bochmann [MB83]. None of these works obtain any complexity results.

A related design problem is that of constructing a system P which satisfies a given formula in a temporal logic. For linear time temporal logic this problem has been considered by Manna and Wolper [MW84] obtaining a PSPACE-complete problem, and in a different framework by Pnueli and Rossner [PR89] obtaining a double exponential time algorithm. For branching time temporal logic (CTL) the problem has been considered by Clarke and Emerson [CE82].

In the next section, we introduce our framework of processes, contexts and bisimulations. Section 2 introduces modal transition systems and refinement. Section 3 presents the INEQ problem and states that it is PSPACE–hard. The actual proof of this fact is found in the appendix, since the GENGEO problem and its reduction to INEQ are not needed for understanding the remainder of the paper. Section 4 contains the reduction from INEQ to EQ . Section 5 contains the reduction from EQ to CCSEQ . Section 6 presents a polynomial time solution to a subproblem of EQ . Finally we discuss open problems and future work. The appendix contains an outline of the proof that INEQ is PSPACE–hard. For a version with complete proofs we refer the interested reader to [JL91].

1 Bisimulation and Contexts

In this paper we follow the *reactive* view of concurrent systems advocated by Pnueli [Pnu85]. We describe concurrent systems (or *processes*) in terms of their interaction with their environment using the well–established model of *labelled transition systems* [Plo81].

Definition 1.1 A labelled transition system *is a structure* $\mathcal{P} = (W, A, \longrightarrow)$ *where* W *is a set of processes (states or configurations),* A *is a set of actions and* $\longrightarrow \subseteq W \times A \times W$ *is a transition relation.*

Notation 1.2 Let $\mathcal{P} = (W, A, \longrightarrow)$ be a labelled transition system. A *derivation sequence* d is a finite or infinite sequence of transitions of the form:

$$d = P_0 \xrightarrow{a_0} P_1, P_1 \xrightarrow{a_1} P_2, P_2 \xrightarrow{a_2} P_3, \ldots$$

which we shall often abbreviate to:

$$d = P_0 \xrightarrow{a_0} P_1 \xrightarrow{a_1} P_2 \xrightarrow{a_2} P_3 \xrightarrow{a_3} \cdots$$

We say that P_0 is the initial process of the sequence d or that d is a derivation sequence for P_0. Whenever a process Q appears in some derivation sequence of P we say that Q is a *derivative* of P.

We say that P is *finite–state* in case the set of its derivatives is finite. For $L \subseteq A$, we say that Q is *L–reachable from P*, if there exists a (finite) derivation sequence with P as initial process, Q as final process and with all actions occurring in the sequence belonging to the set L.

For $d = P_0 \xrightarrow{a_0} P_1 \xrightarrow{a_1} P_2 \xrightarrow{a_2} P_3 \xrightarrow{a_3} \cdots$ a derivation sequence we denote by $\mathrm{Proc}(d)$ the sequence of processes $P_0 P_1 P_2 P_3 \ldots$ and by $\mathrm{Act}(d)$ the sequence of actions $a_0 a_1 a_2 a_3 \ldots$.

A *computation* for a process P is a derivation sequence with P as initial process and which is maximal under the prefix ordering. Hence, the last process of any *finite* computation must be dead–locked with respect to any action. We shall use the notation $\mathrm{Comp}(P)$ to denote the set of computations of P. □

The notion of *bisimulation* [Par81, Mil83] provides a means of identifying processes based on their operational behaviour. In particular, processes at different descriptive levels of abstraction may be compared.

Definition 1.3 *Let $\mathcal{P} = (W, A, \longrightarrow)$ be a labelled transition system. Then a bisimulation $\mathcal{B}$ is a binary relation on W such that whenever $(P, Q) \in \mathcal{B}$ and $a \in A$ then the following holds:*

1. *Whenever $P \xrightarrow{a} P'$, then $Q \xrightarrow{a} Q'$ for some Q' with $(P', Q') \in \mathcal{B}$,*
2. *Whenever $Q \xrightarrow{a} Q'$, then $P \xrightarrow{a} P'$ for some P' with $(P', Q') \in \mathcal{B}$*

P and Q are said to be bisimilar in case (P, Q) is contained in some bisimulation $\mathcal{B}$. We write $P \sim Q$ in this case.

A straightforward generalization allows us to compare processes from different transition systems (essentially by applying the above definition to disjoint sums of transition systems). Bisimulation treats all actions in A equally. One sometimes wants to distinguish between observable and unobservable actions, and define an analogous equivalence in which only the observable actions of transitions are significant. This is achieved by assuming that the action set A contains an *unobservable action* τ. A labelled transition system $\mathcal{P} = (W, A, \longrightarrow)$ now induces an *observational* transition system $\mathcal{P}_O = (W, (A \backslash \{\tau\}) \cup \{\epsilon\}, \Longrightarrow)$, where $P \overset{\epsilon}{\Longrightarrow} Q$ if and only if there is a (possibly empty) sequence $P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} Q$ and $P \overset{a}{\Longrightarrow} Q$ if and only if $P \overset{\epsilon}{\Longrightarrow} P_1 \xrightarrow{a} P_2 \overset{\epsilon}{\Longrightarrow} Q$ for $a \in A \backslash \{\tau\}$. We say that $\mathcal{B}$ is a *weak bisimulation* in case $\mathcal{B}$ is a bisimulation with respect to $\mathcal{P}_O$. We write $P \approx Q$ whenever (P, Q) is contained in some weak bisimulation. The equivalence $\approx$ is often referred to as *observational equivalence.*

Process algebra [Mil80, Mil89, Hoa78, BK85, Bou85] provides a framework for describing both the modular structure and the operational behaviour of reactive systems (or processes). In particular, a process algebra enables processes to be constructed (syntactically) through a number of operators (normally including some operator for parallel composition). Semantically, these operators are described through a number of inference–rules from which the operational behaviour of a composite process may be inferred from that of its components. In Figure 1 is shown the well–known inference rules for the parallel composition operator $\mid$ and the restriction operator $\backslash L$ ($L \subseteq A$) of CCS.

$$\frac{P \xrightarrow{a} P'}{P \mid Q \xrightarrow{a} P' \mid Q} \qquad \frac{Q \xrightarrow{a} Q'}{P \mid Q \xrightarrow{a} P \mid Q'} \qquad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \frac{P \xrightarrow{a} P'}{P \backslash L \xrightarrow{a} P' \backslash L} a, \overline{a} \notin L$$

Figure 1: Inference rules for $\mid$ and $\backslash L$ of CCS

In process algebra, derived operators (or *contexts*) are normally represented syntactically as terms with free variables possibly occurring. In order to facilitate a general investigation of the problem of equation solving, we introduced in [LX90a, LX90b] an operational theory of contexts in terms of action transducers. That is, a (unary) context is semantically viewed as an object which consumes actions provided by its internal process and in return produces actions for an external observer. We shall allow transductions in which the context produces actions on its own without involving the inner process, and also, we shall assume that the context may change during transductions.

Definition 1.4 *A context system $\mathcal{C}$ is a structure $\mathcal{C} = (K, A, \longrightarrow)$, where K is a set of* contexts, *A is a set of* actions, *$\longrightarrow \subseteq K \times (A_0 \times A) \times K$ is a* transduction *relation, $A_0 = A \cup \{0\}$ with 0 being a distinguished* no–action *symbol (i.e. $0 \notin A$).*

For $(C, (a, b), C') \in \longrightarrow$ we shall adopt the notation $C \underset{a}{\overset{b}{\rightarrow}} C'$ and interpret this as: "by consuming the action a the context C can produce the action b and change into C'". For $a = 0$ the production of b does not involve consumption of any action.

Example 1.5 Consider the CCS context $P \mid [\,]$ (we use $[\,]$ as a free variable as this notation suggests the existence of a hole in which to place an argument process). The first inference rule of Figure 1 indicates that $P \mid [\,]$ may produce an action without consulting its argument process Q whenever P has transitions. Stated in terms of transductions, this can be expressed as $P \mid [\,] \underset{0}{\overset{a}{\rightarrow}} P' \mid [\,]$ whenever $P \xrightarrow{a} P'$. The second inference rule of Figure 1 allows the inner process to interact directly with the environment without involving the context. As a transduction we have $P \mid [\,] \underset{a}{\overset{a}{\rightarrow}} P \mid [\,]$ for any action a. Finally, the third inference rule of Figure 1 indicates that the context may produce a τ–action as a result of an internal communication between the inner process (contributing $\overline{a}$) and the process P (contributing a). As a transduction this becomes $P \mid [\,] \underset{\overline{a}}{\overset{\tau}{\rightarrow}} P' \mid [\,]$ whenever $P \xrightarrow{a} P'$.

Now consider a restriction context $[\,]\backslash L$. Then the inference rule given for restriction in Figure 1 may be represented as the transductions $[\,]\backslash L \underset{a}{\overset{a}{\rightarrow}} [\,]\backslash L$ whenever $a, \overline{a} \notin L$. □

Now, given a (unary) context C and a process P we may syntactically form the *combined process* $C(P)$ by substituting P for the free variable (normally denoted $[\,]$) in C. The semantics of $C(P)$ is such that if $P \xrightarrow{a} P'$ and C has an a–consuming transduction $C \underset{a}{\overset{b}{\rightarrow}} C'$ then $C(P) \xrightarrow{b} C'(P')$ should hold. Also, whenever $C \underset{0}{\overset{b}{\rightarrow}} C'$, i.e. C has a transduction which does not involve any consumption, we require the transition $C(P) \xrightarrow{b} C'(P)$. Extending the transition relation for processes such that $P \xrightarrow{0} Q$ if and only if $P = Q$ [1], the above expectations are both met by the following (single) inference rule for combined processes:

$$\frac{C \underset{a}{\overset{b}{\rightarrow}} C' \qquad P \xrightarrow{a} P'}{C(P) \xrightarrow{b} C'(P')} \tag{3}$$

For a combined process of the form $D(C(P))$, a combined context $D \circ C$ may be defined from D and C (see [LX90a]) such that $D(C(P)) = (D \circ C)(P)$.

[1]Note, that this extension does *not* change which processes are bisimular.

2 Modal Transition Systems and Refinement

Modal Transition Systems provides a (graphical) specification formalism for processes and is studied at length in [LT88, HL89, Lar90, BL90, LX90b]. By graphical specification formalism we mean a formalism which uses transition systems, in contrast to logical formalisms. The specifications expressible using Modal Transition Systems (Modal Specifications) typically impose restrictions on the transitions of possible implementations by telling which transitions are *necessary* and which are *admissible*. This is reflected by the structure of a Modal Transition System which contains *two* transition relations: $\longrightarrow_\Box$ for describing the *required* transitions and $\longrightarrow_\Diamond$ for describing the *allowed* transitions.

Definition 2.1 *A* modal transition system *is a structure* $\mathcal{S} = (Q, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$, *where* Q *is a set of (modal) specifications,* A *is a set of actions and* $\longrightarrow_\Box, \longrightarrow_\Diamond \subseteq Q \times A \times Q$ *are two transition relations satisfying the condition* $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$.

The condition $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$ says that anything required is also allowed, ensuring that any modal specification is consistent. A modal specification S is *deterministic* if $T \xrightarrow{a}_\Diamond T_1$ and $T \xrightarrow{a}_\Diamond T_2$ implies $T_1 = T_2$ whenever T is a derivative of S. For a standard labelled transition system $\mathcal{P} = (W, A, \longrightarrow)$, we may consider the derived modal transition system $\mathcal{S} = (W, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$, with $\longrightarrow_\Box = \longrightarrow_\Diamond = \longrightarrow$; i.e. we view processes as modal specifications where all requirements are necessary ones.

Now, the more a specification requires and the less it allows the stronger we expect the specification to be. Using the derivation relations $\longrightarrow_\Box$ and $\longrightarrow_\Diamond$ this may be formalized by the following notion of *refinement*.

Definition 2.2 *Let* $\mathcal{S} = (Q, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$ *be a modal transition system. A* refinement $\mathcal{R}$ *is a binary relation on* Q *such that whenever* $(S, T) \in \mathcal{R}$ *and* $a \in A$ *then the following holds:*

1. *Whenever* $S \xrightarrow{a}_\Diamond S'$, *then* $T \xrightarrow{a}_\Diamond T'$ *for some* T' *with* $(S', T') \in \mathcal{R}$,
2. *Whenever* $T \xrightarrow{a}_\Box T'$, *then* $S \xrightarrow{a}_\Box S'$ *for some* S' *with* $(S', T') \in \mathcal{R}$.

S is said to be a refinement of T in case (S, T) is contained in some refinement R. We write $S \lhd T$ in this case.

As for bisimulation the notion of refinement may be generalized so that specifications from different modal transition systems can be compared. If $P \lhd S$, where P is a process (viewed as a specification through the derived modal transition system) and S is a specification, we will say that P is an *implementation* of S. For P and Q processes it is easy to see that $P \lhd Q$ becomes equivalent to $P \sim Q$.

3 Inequation Solving

The problem of Inequation Solving INEQ is defined as follows:

Instance: A finite collection of pairs

$$\mathcal{E} = \{(C_i, S_i) \mid i \in I\}$$

where I is a finite index set, and for all $i \in I$, C_i is a finite-state context and S_i is a finite-state modal specification.

Question: Does there exist a process P such that the inequation $C_i(P) \lhd S_i$ is satisfied for all $i \in I$?

Let INEQ^1 be the subproblem of INEQ where only singleton collections are allowed. Despite the restriction, it turns out that any instance $\{(C_i, S_i) | i \in I\}$ of INEQ may be transformed (in polynomial time) into an equivalent instance (C, S) of INEQ^1 : simply let $C \xrightarrow[0]{a_i} C_i$ and $S \xrightarrow{a_i}_\Box S_i$ for all $i \in I$ (assuming that the actions a_i are all different), then it is easy to see that the solutionsets coincide:

$$\{P \mid \forall i \in I.\, C_i(P) \lhd S_i\} \;=\; \{P \mid C(P) \lhd S\}$$

Let INEQ_d be the subproblem of INEQ , where the modal specifications of an instance are all required to be deterministic.

Theorem 3.1 *The decision problems* INEQ *and* INEQ_d *are both PSPACE–hard.*

Proof: As announced in the Introduction, we have transferred the proof of this theorem to the Appendix. The proof is a reduction from the known PSPACE–complete graph theoretical problem called Generalized Geography, GENGEO , in [GJ79]. □

From the remark in the beginning of this section it follows that also the decision problem INEQ^1 is PSPACE–hard.

4 Equation Solving

The problem of equation solving EQ is the subproblem of INEQ obtained by restricting the instances to be collections $\{(C_i, Q_i) \mid i \in I\}$ where Q_i is a *process* for all $i \in I$ (recall that any process can be viewed as a modal specification). The problem of INEQ then reduces to whether there exists a process P satisfying the *equation* $C_i(P) \sim Q_i$ for all $i \in I$.

The following lemma provides the basis for establishing PSPACE–hardness of EQ :

Lemma 4.1 *Let S be a deterministic, finite–state modal specification. Then there exists a context C_S and a process Q_S such that for all processes P the following equivalence holds:*

$$P \lhd S \;\Leftrightarrow\; C_S(P) \sim Q_S$$

Moreover, the size of both C_S and Q_S is linear in the size of S, and Q_S is deterministic.

Proof: For each specification S we define contexts C_S, D_S and a process Q_S. We state just he inference rules defining the behaviours of C_S, D_S, and Q_S in terms of that of S:

$$\frac{S \xrightarrow{a}_\Diamond S'}{C_S \xrightarrow[a]{a} C_{S'}} \qquad \frac{S \xrightarrow{a}_\Diamond S' \quad S \not\xrightarrow{a}_\Box S'}{C_S \xrightarrow[0]{a} D_{S'}} \qquad \frac{S \not\xrightarrow{a}_\Diamond}{C_S \xrightarrow[a]{x}} \qquad \frac{S \xrightarrow{a}_\Diamond S'}{D_S \xrightarrow[0]{a} D_{S'}} \qquad \frac{S \xrightarrow{a}_\Diamond S'}{Q_S \xrightarrow{a} Q_{S'}}$$

where x is a new action symbol. The idea is that C_S is a context which behaves like the inner process P (by the leftmost rule). However, in case S does not require a transition, then C_S must be able to perform that transition even when its inner process cannot. This is attained by the transition to some $D_{S'}$ (the second rule), whereafter $D_{S'}$ behaves exactly like S'. Finally, C_S prohibits disallowed moves by P by translating them to some distinguished action x. □

Theorem 4.2 *The decision problem* EQ *is PSPACE-hard.*

Proof: let $\mathcal{E} = \{(C_i, S_i) \mid i \in I\}$ be an instance of INEQ$_d$. Then $\mathcal{E}^* = \{(C_{S_i} \circ C_i, Q_{S_i}) \mid i \in I\}$ is an instance of EQ and it follows from Lemma 4.1 that the solutionsets to $\mathcal{E}$ and $\mathcal{E}^*$ coincide, and that the size of $\mathcal{E}^*$ is polynomial in the size of $\mathcal{E}$. □

Now, let EQ^{1}be the subproblem of EQ where only singleton collections are allowed. Then PSPACE–hardness follows from the PSPACE–hardness of INEQ^1 . Also the subproblem EQ$_d$ of EQ where only deterministic processes is allowed is PSPACE–hard as the process Q_S constructed in Lemma 4.1 is deterministic provided S is.

5 Equation Solving in Process Algebra

In this section, we first consider the problem CCSEQ , which is the subproblem of EQ^{1}obtained by restricting the context C to be of the form $(A \mid P)\backslash L$ for given A and L. We thereafter consider CCSOBS which is obtained from CCSEQ by replacing bisimulation equivalence $\sim$ by observation equivalence $\approx$.

The problem CCSEQ is the following:

> *Given (finite–state) processes A and B and a set of actions L, does there exist a process P satisfying the equation $(A \mid P)\backslash L \sim B$.*

Just as for the problem INEQ^1 , it does not matter whether we consider one equation or a collection of equations, represented by pairs $\{(A_i, B_i) \mid i \in I\}$ as long as the set L is the same in all equations. We can simply let $A \xrightarrow{a_i} A_i$ and $B \xrightarrow{a_i} B_i$ for all $i \in I$ for different a_i's which are not in L and do not occur elsewhere in any B_i.

We shall prove that CCSEQ is PSPACE–hard by a reduction from the problem EQ^1, presented in the previous section. The following lemma provides the basis for this result.

Lemma 5.1 *Let C be a context and Q be a process. Let L be the union of the sorts of C and Q. Then there are processes A_C and B_Q, such that for any process P:*

$$C(P) \sim Q \;\Leftrightarrow\; (A_C \mid P\backslash L^c)\backslash L \sim B_Q$$

Proof: The sorts of A_C and B_Q will be the union of L, L' and $\{w\}$, where $L' = \{a' \mid a \in L\}$ is a tagged copy of I, and w is a distinguished action. We just state the inference rules defining the transitions of A_C and B_Q in terms of the transductions of C and the transitions of Q:

$$\frac{C \xrightarrow[a]{b} C'}{A_C \xrightarrow{\overline{a}} \cdot \xrightarrow{b'} A_{C'}} \qquad \frac{Q \xrightarrow{b} Q'}{B_Q \xrightarrow{\tau} \cdot \xrightarrow{b'} B_{Q'}} \qquad A_C \xrightarrow{w} NIL \qquad B_Q \xrightarrow{w} NIL$$

where NIL is a process that can not perform any actions. Thus, any transduction of C corresponds to two consecutive transitions of A_C. Similarly, any transition of Q corresponds to a sequence of two transitions of B_Q. In the rules above $\cdot$ abbreviates the intermediate states, having precisely one transition. The use of w is to insure that intermediate states are matched with intermediate states. □

Theorem 5.2 *The decision problem* CCSEQ *is PSPACE–hard.*

Proof: According to lemma 5.1 any EQ^1problem can be reduced to an equation solving problem of the form:

$$\exists P.(A \mid P\backslash L^c)\backslash L \sim B$$

which is equivalent to the existence of a process P satisfying $(A \mid P)\backslash L \sim B$ and sort$(P) \subseteq L$. Now, the restriction that the sort of P is included in L may be expressed by the following extra equation $(U \mid P)\backslash L \sim V$, where:

$$\begin{aligned} U &= \sum_{a \in L} (\overline{a}.w.U + a.w.U + \tau.w.U) \\ V &= \tau.w.V \end{aligned}$$

w being a distinguished action not in L. Using the technique described above, the two equations $(A \mid P)\backslash L \sim B$ and $(U \mid P)\backslash L \sim V$ can be combined into one equation, whence an instance of CcsEq . □

An alternative reduction shows that CcsEq is PSPACE–hard even for instances where the right–hand process is restricted to a deterministic process.

Next we consider the problem CcsObs , which is similar to CcsEq except that the satisfaction relation is that of observation equivalence weak bisimularity. That is the problem is:

> *Given (finite–state) processes A and B and a set of actions L, does there exist a process P satisfying the equation $(A \mid P)\backslash L \approx B$.*

We shall prove that CcsObs is PSPACE–hard by a reduction from the problem CcsEq .

Define the *rigidification* P_r of a process P as the process which has behaviour as follows: $P_r \xrightarrow{a} Q$ if and only if $P \overset{a}{\Longrightarrow} P'$ and $Q = P'_r$ where $a \neq \tau$. Note that P_r is *rigid* in the sense that no derivative of P_r has τ–transitions.

Lemma 5.3 *Let A_C, B_Q and L be as in the proof of Lemma 5.1. Then for any process P with sort$(P) \subseteq L$ the following holds:*

$$(A_C \mid P)\backslash L \approx B_Q \;\Rightarrow\; (A_C \mid P_r)\backslash L \sim B_Q$$

Theorem 5.4 *The decision problem* CcsObs *is PSPACE–hard.*

Proof: Let A_C, B_Q and L be as in the proof of Lemma 5.1. As $\approx$ is a weaker equivalence than $\sim$ it follows that

$$\exists P.\,(A_C \mid P)\backslash L \sim B_Q \implies \exists P.\,(A_C \mid P)\backslash L \approx B_Q$$

The opposite implication follows from Lemma 5.3. Thus the two sides in the implication are equivalent, whence the theorem follows from Theorem 5.2. □

6 Polynomial Equation Solving

As argued in the Introduction both equation and inequation solving occur during (top–down) development of concurrent systems. As such it is important to find conditions under which these problems may be dealt with efficiently. In this section we identify conditions on contexts which will induce a subproblem of EQ^1which is solvable in polynomial time.

Definition 6.1 *A context C is* deterministic *if $D \underset{a}{\overset{b_1}{\longrightarrow}} D_1$ and $D \underset{a}{\overset{b_2}{\longrightarrow}} D_2$ implies $b_1 = b_2$ and $D_1 = D_2$ for any derivative D of C.*

Definition 6.2 *Let $\mathcal{P} = (S, A, \longrightarrow)$ be a labelled transition system and let $\mathcal{C} = (K, A, \longrightarrow)$ be a context system. A* consistency *relation $\mathcal{K}$ is a subset of $K \times S$ such that whenever $(C, Q) \in \mathcal{K}$ then the following holds:*

$$\text{Whenever } Q \xrightarrow{b} Q', \text{ then } C \underset{a}{\overset{b}{\longrightarrow}} C' \text{ for some } a, C' \text{ such that } (C', Q') \in \mathcal{K}.$$

We say that C and P are consistent if (C, P) is contained in some consistency relation.

Note that the notion of consistency is very similar to that of simulation (being "half" of bisimulation [Mil89]) for which there is a well-known polynomial time decision procedure [KS]. For deterministic contexts the notion of consistency captures exactly that of solvability:

Theorem 6.3 *Let C be a deterministic context and Q a process. Then there exists a process P such that $C(P) \sim Q$ if and only if C and Q are consistent.*

Proof:
$\Rightarrow$ Let $\mathcal{K} = \{(C, Q) \mid \exists P. C(P) \sim Q\}$. We show that $\mathcal{K}$ is a consistency relation. So let $(C, Q) \in \mathcal{K}$ and let $Q \xrightarrow{b} Q'$. Now assume $C(P) \sim Q$, then $C(P) \xrightarrow{b} R$ with $R \sim Q'$. According to the rule of inference for combined processes, $C \underset{a}{\overset{b}{\longrightarrow}} C'$ and $P \xrightarrow{a} P'$ for some a, C' and P' with $R = C'(P')$. Obviously, $(C', Q') \in \mathcal{K}$.

$\Leftarrow$ Let $\mathcal{K}$ be a consistency relation. Define a transition system with states $P_{C,Q}$ for $(C, Q) \in \mathcal{K}$ and transtions:

$$P_{C,Q} \xrightarrow{a} P_{C',Q'} \quad \Leftrightarrow^{\Delta} \quad \exists b. Q \xrightarrow{b} Q' \wedge C \underset{a}{\overset{b}{\longrightarrow}} C'$$

Then $C(P_{C,Q}) \sim Q$ for all $(C, Q) \in \mathcal{K}$. To see this we show that the relation below is a bisimulation:

$$\mathcal{B} = \{(C(P_{C,Q}), Q) \mid (C, Q) \in \mathcal{K}\}$$

Let $(C(P_{C,Q}), Q) \in \mathcal{B}$ and assume $Q \xrightarrow{b} Q'$ As $(C, Q) \in \mathcal{K}$, $C \underset{a}{\overset{b}{\longrightarrow}} C'$ with $(C', Q') \in \mathcal{K}$ for some a, C'. But then $P_{C,Q} \xrightarrow{a} P_{C',Q'}$. Hence, using the inference rule for combined processes, $C(P_{C,Q}) \xrightarrow{b} C'(P_{C',Q'})$ and clearly $(C'(P_{C',Q'}), Q') \in \mathcal{B}$. Let $C(P_{C,Q}) \xrightarrow{b} R$. That is $C \underset{a}{\overset{b}{\longrightarrow}} C'$ and $P_{C,Q} \xrightarrow{a} P_{C'',Q''}$ for some a, C' and $P_{C'',Q''}$. Now according to the definition of $P_{C,Q}$'s transtions $C \underset{a}{\overset{b'}{\longrightarrow}} C''$ and $Q \xrightarrow{b'} Q''$ for some b'. But as C is assumed to be *deterministic* it follows that $b = b'$ and $C' = C''$. Thus $R = C'(P_{C',Q''})$ and obviously $Q \xrightarrow{b} Q''$ is a matching transition. □

Since consistency relations can be found in polynomial time, we get the following theorem.

Theorem 6.4 *Let C be a deterministic context and Q a process. Then the problem whether there exists a process P such that $C(P) \sim Q$ can be decided in polynomial time.*

Examples of deterministic contexts are:

- the CCS context $(A \mid [\,])\backslash L$, where A is deterministic and rigid (i.e. the derivatives of A have no internal transitions) and the sort of A is included in L,
- the CSP [BHR84] context $A \| [\,]$, where A is a deterministic process.

Open Problems and Future Work

This paper leaves as open problems whether or not the decision problems InEq and Eq are members of PSPACE. The problems can all be solved in single exponential time, using the procedure proposed by Larsen and Xinxin [LX90b]. This procedure involves checking for consistency in a (disjunctive) modal transition system with an *exponential size* in that of the underlying context and process.

On the positive side, a more careful examination shows that the procedure for Eq presented in [LX90b] has time complexity exponential in the size of the contexts but *polynomial* in the size of the processes. Thus, equation systems with small or bounded size contexts may be dealt with efficiently.

As for future work we should like to continue the work of Section 6 in identifying more liberal conditions on contexts which will make (in)equation solving efficient.

Acknowledgement

The authors are greateful to Pierre Wolper for fruitful discussions, and to the referees for helpful comments.

References

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, 1984.

[BK84] J Bergstra and J Klop. Verification of an alternating bit protocol by means of process algebra. Technical report, Centrum voor Wiskunde en Informatica, 1984.

[BK85] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37:77–121, 1985.

[BL90] Gérard Boudol and Kim G. Larsen. Graphical versus logical specifications. *Lecture Notes in Computer Science*, 431, 1990. In Proceedings of CAAP90.

[Bou85] G. Boudol. Calcul de processus et verification. Technical Report 424, INRIA, 1985.

[CE82] E. Clarke and E.A. Emerson. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2:241–266, 1982.

[GJ79] Garey and Johnson. *Computers and Intractability: A guide to the Theory of NP-Completeness*. Freeman, 1979.

[HL89] Hans Hüttel and Kim G. Larsen. The use of static constructs in a modal process logic. *Lecture Notes in Computer Science*, 363, 1989.

[Hoa78] C.A.R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8), 1978.

[JL91] B. Jonsson and K.G. Larsen. On the complexity of equation solving in process algebra. Technical report, Swedish Institute of Computer Science, 1991.

[Koo85] C. Koomen. Algebraic specification and verification of protocols. *Science of Computer Programming*, 5(1):1–36, 1985.

[KS] Kannellakis and Smolka. CCS expressions, finite state processes, and three problems of equivalence. To appear in Information and Computation.

[Lar90] K.G. Larsen. Modal specifications. *Lecture Notes in Computer Science*, 407, 1990.

[LM87] Kim G. Larsen and Robin Milner. Verifying a protocol using relativized bisimulation. In *Proc. ICALP '87*, volume 267 of *Lecture Notes of Computer Science*. Springer Verlag, 1987.

[LQ90] P. Lewis and H. Qin. Factorization of finite state machines under observational equivalence. *Lecture Notes in Computer Science*, 458, 1990.

[LT88] Kim G. Larsen and Bent Thomsen. A modal process logic. In *Proceedings on Logic in Computer Science*, 1988.

[LX90a] K.G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. *Lecture Notes in Computer Science*, 1990. To appear in Proceedings of ICALP90.

[LX90b] K.G. Larsen and L. Xinxin. Equation solving using modal transition systems. In *Proceedings on Logic in Computer Science*, 1990.

[MB83] Philip Merlin and Gregor von Bochmann. On the construction of submodule specifications and communication protocols. *ACM Transactions on Programming Languages and Systems*, 5(1):1–25, 1983.

[Mil80] R. Milner. *Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer Verlag, 1980.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25, 1983.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice–Hall, 1989.

[MW84] Zohar Manna and Pierre Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 6(1):68–93, 1984.

[Par81] D. Park. Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science*, 104, 1981. in Proc. of 5th GI Conf.

[Par87] Joachim Parrow. Verifying a csma/cd-protocol with ccs. Technical Report ECS-LFCS-87-18, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, 1987. To appear in *Protocol Specification, Testing, and Verification VIII (1988)*.

[Par89] J. Parrow. Submodule construction as equation solving in CCS. *Theoretical Computer Science*, 68:175–202, 1989.

[Plo81] G. Plotkin. A structural approach to operational semantics. FN 19, DAIMI, Aarhus University, Denmark, 1981.

[Pnu85] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. *Lecture Notes in Computer Science*, 194, 1985. in Proc. of ICALP'87.

[PR89] A. Pnueli and R. Rossner. On the synthesis of a reactive module. In *Proc. 16:th ACM Symp. on Principles of Programming Languages*, pages 179–190, 1989.

[SFD85] S.A. Smolka, A.J. Frank, and S.K. Debray. Testing protocol robustness the CCS way. In *Protocol Specification, Testing, and Verification IV (1984)*, pages 93–108, 1985. North-Holland.

[Shi89] M.W. Shields. A note on the simple interface equation. *The Computer Journal*, 32(5), 1989.

A Proof that InEq is PSPACE–hard

A.1 The Generalized Geography Problem

Definition A.1 *A* rooted, directed *graph is a structure* $G = (V, E, v_0)$, *where* V *is a (finite) set of vertices,* $E \subseteq V \times V$ *is a set of edges and* $v_0 \in V$ *is the root (the initial vertex).*

Let $G = (V, E, v_0)$ be a rooted, directed graph. For $e = (u, v) \in E$ we write $\text{hd}e$ for v and $\text{tl}e$ for u. A path of G is any finite sequence $p = e_0 e_1 e_2 \ldots e_n$, where $\text{tl}e_0 = v_0$ and $\text{hd}e_i = \text{tl}e_{i+1}$ for all $i \in [0, n[$. We write $\text{Path}(G)$ for the set of paths of G. For $e \in E$ we define the set $\text{Follow}(e) = \{f \in E \mid \text{hd}e = \text{tl}f\}$. Also, $\text{Init} = \{e \in E \mid \text{tl}e = v_0\}$. In fact, a sequence of edges $e_0 e_1 e_2 \ldots e_n$ is a path of G just in case $e_0 \in \text{Init}$ and $e_{i+1} \in \text{Follow}(e_i)$ for all $i \in [0, n[$.

Given a rooted, directed graph $G = (V, E, v_0)$ the (two–player) *Generalized Geography* game on G is played according to the following rules [GJ79]:

> The two players alternate choosing a new edge from E. The first edge chosen (by player 1) must have its tail at v_0 and each subsequently chosen edge must have its tail at the vertex that was the head of the previous edge, and must not have been previously chosen in the game. The first player unable to choose such a new edge loses.

Now, the Generalized Geography problem GenGeo may be described as below. Also, we recall from [GJ79] that GenGeo is PSPACE–complete.

> **Instance:** A rooted, directed graph G.
>
> **Question:** Does player 1 have a forced win in the Generalized Geography game played on G?

We want to reformulate (or formalize) the GenGeo problem into a question of existence of a process (of some labelled transition system) expressing in an *explicit* way a winning strategy for player 1 on a given graph G. Thus, let $G = (V, E, v_0)$ be a rooted, directed graph and let P be a process of some labelled transition system *with* E *as action set*. Then:

> P *respects* G if $\text{Act}(d)$ is a path of G for any finite derivation sequence d of P.
>
> P *obeys the* GenGeo *game* if the actions (i.e. edges of G) occurring in any derivation sequence of P are all different.

The idea is that the computations of P should correspond to complete GenGeo games on G (with player 1 as winner if the length of the computation is odd). Now, we want P to capture several GenGeo games; in particular we want P to provide player 1 with a strategy for *any* legal move of the opponent:

P provides a strategy with respect to G if whenever

$$P \xrightarrow{e_0} P_1 \xrightarrow{e_1} P_2 \xrightarrow{e_2} \cdots \xrightarrow{e_j} P_j$$

is an odd length derivation sequence of P, then for any $e \in \mathrm{Follow}(e_j)\backslash\{e_0, \ldots, e_j\}$:

$$P_j \xrightarrow{e} P^e_{j+1}$$

for some P^e_{j+1}.

Here $\mathrm{Follow}(e_j)\backslash\{e_0, \ldots, e_j\}$ is the set of legal moves of the opponent (only *new* edges can be chosen), and P^e_{j+1} describes player 1's strategy after the move e. Finally, P should only contain computations with player 1 as winner. I.e.:

P provides a winning strategy with respect to G if P respects G, obeys the GENGEO game and provides a strategy wrt. G such that all computations of P has odd length.

We now reformulate (or formalize) the GENGEO problem as follows:

Instance: A rooted, directed graph G.

Question: Does there exist a process P providing a winning strategy with respect to G?

A.2 Inequation Solving

We recall that the problem of Inequation Solving INEQ is defined as follows:

Instance: A finite collection of pairs

$$\mathcal{E} = \{(C_i, S_i) \mid i \in I\}$$

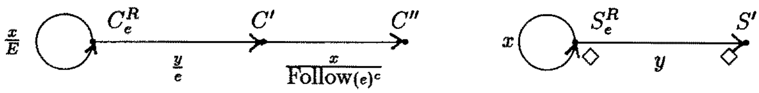
where I is a finite index set, and for all $i \in I$, C_i is a finite–state context and S_i is a finite–state modal specification.

Question: Does there exist a process P such that the inequation $C_i(P) \triangleleft S_i$ is satisfied for all $i \in I$?

In the remainder of this section we shall show how to transform (in polynomial time) any instance $G = (V, E, v_0)$ of GENGEO into an equivalent instance $\mathcal{E}_G$ of INEQ . That is: there will be a winning strategy for player 1 on G just in case the inequation system $\mathcal{E}_G$ has a solution. In fact, the transformation offered will be such that the solutionset to $\mathcal{E}_G$ is exactly the winning strategies with respect to G. As GENGEO is a PSPACE–complete decision problem it will follow that INEQ is PSPACE–hard! Whether or not INEQ is *in* PSPACE is as yet an open problem on which we shall comment in the conclusion.

In the remainder of this section let $G = (V, E, v_0)$ be a given rooted, directed graph. We construct, in the following lemmas, inequation systems which will be equivalent to the four conditions on a winning strategy for G. For full proofs we refer to [JL91].

Lemma A.2 *For $e \in E$ let C_e^R and S_e^R have the following behaviours* [2] *:*



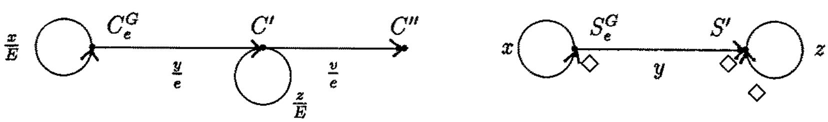
where x and y are different actions. Also let C_I and S_I be defined by:

Then P is a solution to the inequation system:

$$\mathcal{E}_R = \{(C_e^R, S_e^R) \mid e \in E\} \cup \{(C_I, S_I)\}$$

if and only if P respects G.

Lemma A.3 *For $e \in E$ let C_e^G and S_e^G have the following behavious:*

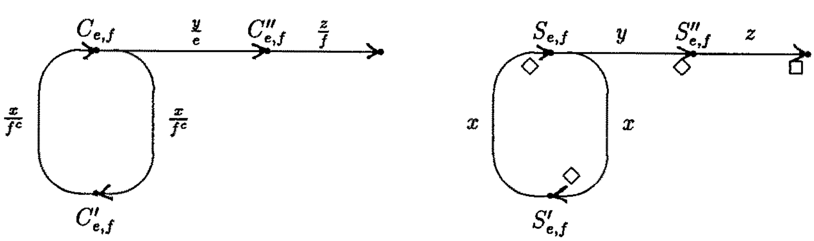


where x, y, z, v are all different actions. Then P is a solution to the inequation system:

$$\mathcal{E}_G = \{(C_e^G, S_e^G) \mid e \in E\}$$

if and only if P obeys the GENGEO *game.*

Lemma A.4 *For $e \in E$ and $f \in \mathrm{Follow}(e) \backslash \{e\}$ let $C_{e,f}$ and $S_{e,f}$ have the following behaviours:*



---

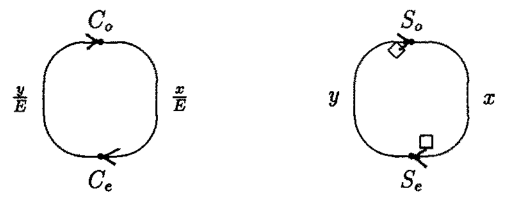
[2]If, for $B_1, B_2 \subseteq E$, $\frac{B_2}{B_1}$ labels an edge between contexts C and D, this means by convention that $C \xrightarrow[a]{b} D$ for any $a \in B_1$ and $b \in B_2$. Singleton sets over A are identified with their element. For a set A, A^c denotes the complementary set.

with x, y, z being different actions. Then P is a solution to the inequation system:

$$\mathcal{E}_S = \{(C_{e,f}, S_{e,f}) \mid e \in E, f \in \text{Follow}(e)\backslash\{e\}\}$$

if and only if P provides a strategy with respect to G.

Lemma A.5 *Let C_o and S_o have the following behaviours:*



where x and y are different actions. Then P is a solution to the (singleton) inequationsystem:

$$\mathcal{E}_o = \{(C_o, S_o)\}$$

if and only if all finite *computations of P has odd length.*

We can now state and prove Theorem 3.1.

Theorem A.6 *The decision problems* InEq *and* InEq$_d$ *are both PSPACE–hard.*

Proof: It follows easily from Lemma A.2 – A.5 that P is a solution to the inequation system $\mathcal{E} = \mathcal{E}_R \cup \mathcal{E}_G \cup \mathcal{E}_S \cup \mathcal{E}_o$ if and only if P provides a winning strategy with respect to G. Thus PSPACE–hardness of InEq follows directly from PSPACE–completeness of GenGeo and the fact that the inequation system constructed by Lemma A.2 – A.5 has polynomial size relative to the original graph. PSPACE–hardness of InEq$_d$ follows by noting that the modal specifications of inequations constructed in Lemmas A.2 – A.5 are all deterministic. □