Automatic Verification of Parameterized Systems by Over-Approximation

by

Vladislavs Jahundovičs
This is a Swedish Licentiate’s Thesis

Swedish postgraduate education leads to a doctor’s degree and/or a licentiate’s degree.
A doctor’s degree comprises 240 ECTS credits (4 year of full-time studies).
A licentiate’s degree comprises 120 ECTS credits.

Copyright © 2015 Vladislavs Jahundovičs

ISBN 978-91-7685-918-6
ISSN 0280–7971
Printed by LiU Tryck 2015

URL: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-121776
Abstract

This thesis presents a completely automatic verification framework to check safety properties of parameterized systems. A parameterized system is a family of finite state systems where every system consists of a finite number of processes running in parallel the same algorithm. All the systems in the family differ only in the number of the processes and, in general, the number of systems in a family may be unbounded. Examples of parameterized systems are communication protocols, mutual exclusion protocols, cache coherence protocols, distributed algorithms etc.

Model-checking of finite state systems is a well-developed formal verification approach of proving properties of systems in an automatic way. However, it cannot be applied directly to parameterized systems because the unbounded number of systems in a family means an infinite state space. In this thesis we propose to abstract an original family of systems consisting of an unbounded number of processes into one consisting of a fixed number of processes. An abstracted system is considered to consist of $k+1$ components — $k$ reference processes and their environment. The transition relation for the abstracted system is an over-approximation of the transition relation for the original system, therefore, a set of reachable states of the abstracted system is an over-approximation of the set of reachable states of the original one.

A safety property is considered to be parameterized by a fixed number of processes whose relationship is in the center of attention in the property. Such processes serve as reference processes in the abstraction. We propose an encoding which allows to perform reachability analysis for an abstraction parameterized by the reference processes.

We have successfully verified three classic parameterized systems with replicated processes by applying this method.

This work has been supported by CUGS (the National Graduate School in Computer Science, Sweden).
Acknowledgments

First of all, I would like to express my gratitude to my advisor Prof. Ulf Nilsson for his help, patience and valuable comments on this thesis. I would also like to thank CUGS graduate school for supporting my research and graduate studies which resulted in this work. At last, but not least, I would like to thank my wife, Jenny, for her love and support.
Chapter 1

Introduction

The Problem in its Generality

Computer systems have been growing constantly in size and complexity for the last few decades. They trench into more and more spheres of human activities — medicine, transport, communications, industry, etc. Increasing dependency on computer systems gives rise to stronger demands on their correctness and reliability.

In some areas assurance of having a correct system is a necessity. For example, while a restart of an operating system because of a software bug irritates a computer user and causes only the loss of few minutes of his time, bugs in an aircraft control system may lead to a crash and cost human lives.

For a long time testing and simulation of the implemented systems has served as a method to assure that the desired system properties and behavior are satisfied. However, in the majority of cases it is impossible to perform exhaustive testing to cover all possible cases because their number is huge or infinite. Therefore, we never can be sure, that the system is reliable and bug-free. As early as in 1972 Dijkstra stated:

Program testing can be used to show the presence of bugs, but never to show their absence! [31]

Nevertheless, there is a long and bitter experience of notorious accidents caused by the reliance entirely on testing and simulation in the software development. Among prominent failures we can mention Therac-25 case [54] when a software bug in medical equipment cost human lives, and Ariane 5 missile failure [52] (also known as Ariane 5 Flight 501) with losses over $500 000 000. Ariane 501 Inquiry Board stated in its final report [52] that software should be assumed to be faulty until applying the currently accepted best practice methods can demonstrate that it is correct.
In practice that means that formal methods should be applied because it is the only way to demonstrate with proofs that software is correct with respect to its specification.

Formal verification of a system is a process of proving with formal methods of mathematics that the system satisfies certain properties. Therefore, having a system which has been formally verified is much more desirable than the one which has been just tested against a limited number of cases.

There are two established approaches to formal verification — theorem proving and model checking [25]. In theorem proving one has to specify both the system and its specification in a formal logic as a set of axioms and inference rules. Although there is a number of automated tools, the process of finding a proof usually requires human assistance and expertise.

Model checking [32] is an approach of an automatic verification in contrast to automated one in the theorem proving. A system is modeled as a finite state-transition graph, while the specification is expressed in a temporal logic, and an efficient algorithm checks automatically whether the transition system is a model for the temporal formula.

There exist many problems where model checking cannot be directly applied. Usually, because the number of states is either infinite or finite but too huge to be handled by computers. Automated verification of such systems may still be possible, usually by the means of abstraction of the original system to the one which can be verified by model checking ([22],[55],[59],[21],[15], etc). Obviously, abstraction must be sound with respect to the properties we wish to prove. Soundness means that whenever a property holds in an abstracted system then it also holds in the original one.

The Scope of the Thesis

This thesis considers automated formal verification of safety properties in parameterized systems. A parameterized system is a family of state transition systems of concurrently running processes, where the number of processes is a parameter. When the number of processes is unbounded the family becomes infinite. Formal verification of safety properties of parameterized systems establishes if a property holds in all states for all system sizes. Although the problem is undecidable in general [11], much work ([7], [23],[55],[58],[16],[56]) has been done recently to provide for automated verification for interesting classes of parameterized systems.

In this work we consider proving safety properties of the form

\textit{certain bad system configurations never occur}

where a “bad” unsafe system configuration is expressed as a relation between a fixed (independent of $n$) number of processes (that is, the arity of the relation is fixed, but its domains can be parameterized). Note, that the word “certain” means those states which can be thought of and formally
CHAPTER 1. INTRODUCTION

specified. Hence, in general we can never claim that a successfully verified
system is bug-free, but rather that it is free from the bugs which we could
think of, specified and verified.

A classical example is a mutual exclusion property:

no two processes can be in the critical section at the same time.

In this property we are interested in a relationship between a fixed number
of processes (two processes have the property of being in the critical section
at the same time) and consider all \( C_n^2 = \frac{n!}{2!(n-2)!} = \frac{1}{2}(n^2 - n) \) combinations
of two processes from \( n \) processes in the system. The main challenge in a
parameterized system is that \( n \) is often unbounded.

Our approach is to abstract a parameterized system to an infinite state
transition system, and apply standard techniques for infinite state systems
from model checking which may help to verify the abstracted system. (Since
the abstracted system may be infinite, termination is no longer guaranteed).

We will relax synchronization between processes to achieve abstraction.
By synchronization we mean a situation when one process can make a tran-
sition from one state to another depending on a states of other processes.
For example, when a process \( p_1 \) can perform a transition from state \( s_1 \) to
\( s_2 \) only if process \( p_2 \) is in one of the states \( s_3 \) or \( s_4 \), then we say that this
transition of \( p_1 \) is synchronized with \( p_2 \).

Let us consider the same type of mutual exclusion property as above but
limited to only two particular processes \( p_1 \) and \( p_2 \):

processes \( p_1 \) and \( p_2 \) cannot be in the critical section at the same
time.

The idea of the abstraction presented in this work is to treat only these two
processes \( p_1 \) and \( p_2 \) as the first-class reference processes w.r.t. synchronization.
This means that we disregard any synchronization between a set of
processes if it does not involve at least one reference process. The intuition
behind the idea is that we disregard any synchronization between the pro-
cesses if we are not interested in their mutual relationship w.r.t. a safety
property.

All reference processes in the abstracted system run the same algorithm
as before. The algorithm of non-reference processes (also called environment
processes) is modified so that they synchronize only with reference processes.
Otherwise synchronized transitions become unconditional. Using this fact
we have developed a symbolic representation of sets of global states even for
a system with an unbounded number of environment processes (although
the set of reference components must be bounded) together with an ap-
proximated reachability algorithm (both forward and backward reachability
analysis is possible).

Obviously, an over-approximated set of reachable states obtained with
this method will depend on which processes are chosen to be reference ones.
If we treat the set of reference processes as a parameter, we can perform the
reachability analysis for different values of the parameter in parallel even if the domain of the parameter is unbounded. In the same way we will treat the mutual exclusion property: as a relation on only two processes, but the pair of processes is a parameter and ranges over all pairs of different processes in the system.

And, finally, if we introduce into the presented scheme the number of processes in the system as a parameter, it will be possible to verify safety properties of parameterized systems.

Related Work

There exist many aspects, which should be taken into account when comparing different verification methods, such as

- **Expressiveness**: Is it possible to express the system within the given specification language? A method is to no avail if there are no means to describe the problem in the method’s input language.

- **Proving power**: Is it possible to verify the system with the given method? Formal verification of parameterized systems is undecidable [11], therefore all existing methods cover only certain classes of parameterized systems.

- **Soundness**: If the verification algorithm produce some results, they must be correct, otherwise a method is not sound (and, apparently, of no use).

- **Completeness**: The ability to verify any statement. A method which fails to produce a result for some correct statements (possibly because of the non-termination) is called *incomplete*.

- **Complexity**: Even verification with a complete method can fail if it cannot be computed within reasonable time. The availability of such techniques as, for example, abstraction or compositional proofs, can be crucial for successful application.

- **Degree of automation**: Although there may exist elegant manual proofs of properties of many interesting systems, there is a demand for methods with a high degree of automation.

- **Skill requirements**: The success of the application of many formal methods depends greatly not only on the competence in the problem-specific area but also in the area of formal verification, which hamper the use even of well-established verification methods.

The efforts toward formal verification of parameterized systems tend to be classified in two competing categories: *model checking* ([37]) and *deductive verification* [33]. Model checking exploits algorithmic methods to
perform state space exploration in order to prove a property, while deductive verification is based on theorem proving methods. The main trade-off is between the power of theorem proving and the high degree of automation assured by model checking. We limit ourselves to the description of related works which provide for (a high degree of) automated verification.

The early works on verification of parameterized systems include works of German and Sistla [37], Vernier [41], Emerson and Namjoshi [34]. A common approach then was to restrict parameterized systems to certain configurations (for example, systems with a unique control process with arbitrary number of identical user processes [37], or a class of systems which can be decomposed into two synchronized automata [41]) as well as the way of communication between the processes (e.g. communication using CCS actions [37]), which allowed to create effective model checking algorithms.

An alternative approach is to make the class of verified systems larger by the price of completeness [46, 58, 3, 43, 57]. For example, symbolic model checking with rich assertional languages [46] can deal with large classes of parameterized systems having array or tree topologies. However, the verification algorithm does not necessarily converge (even extended with acceleration [58]).

In parameterized systems the convergence problem is often hidden in loops or data structures like stacks, arrays etc which take the size of the system as an argument. The number of states becomes parameterized when the algorithm executes such a loop preventing thus the use of well established verification techniques. The acceleration techniques proposed by Abdulla et al [3] and transitive closures of regular relations by Jonsson and Nilsson [43] can handle many interesting cases of mutual exclusions protocols parameterized by the number of processes. Nevertheless, these techniques are also incomplete and do not terminate in the general case.

There has been work done on automating different steps of deductive verification to make it more applicable. A considerable amount of time and expertise required in proof construction is the main obstacle in wide application of deductive verification. For example, deductive verification with network invariants [33, 53, 20, 47] requires manual construction of network invariants. Later works on invisible invariants [57, 12, 35, 38] (also known as invisible auxiliary constructs) claimed to provide for automatic deductive verification of safety and liveness properties. These methods automatically generate assertions which are required for deductive proofs and which should be provided manually by the user otherwise. The process of generation and application of assertions is hidden from the end user, therefore they are named invisible assertions.

Model checking cannot deal directly with parameterized systems because the state space of a parameterized system is unbounded. A common approach is to abstract a parameterized system into a finite state system and apply model checking afterwards ([23]). The following must be true for both an original and an abstract system and the property we wish to prove:
CHAPTER 1. INTRODUCTION

If the property holds in the abstracted system then it also holds in the original system.

Obviously, the abstracted system should be simpler than the original one and suitable (i.e. finite-state) to be model checked w.r.t. the property. Such an approach is an instance of a more general abstract interpretation framework [27].

The methods of counter abstraction [59, 28, 17] and indexed predicates [49, 48, 50] are in the recent area of focus in what concerns verification of parameterized systems. They can be perceived as derivations of the predicate abstraction method introduced by Graf and Saidi [39, 60, 13], where the program states and variables are abstracted as a set predicates. The abstract system is model checked afterwards to prove a property. Together with automatic predicate discovery the method becomes fully automatized.

Indexed predicates method [49, 13, 50] is an extension to predicate abstraction schema. This approach can handle cases when predicates are quantified over parameters and component indexes. It introduces predicates with free index variables which facilitates automatic predicate discovery.

The name of counter abstraction was introduced by Pnueli et al. [59]. The main idea is based on introducing counters for the number of processes in each local state. The counters range over small finite domain to make the system finite-state (e.g. they can have three values: more than one, one and zero).

The environment abstraction [23, 64, 24] is a comparatively new abstraction technique which tries to follow the algorithm designer’s line of thoughts when constructing a system abstraction. This abstraction idea is probably the closest to the one presented in this thesis. One process in a parameterized system serves as a reference process (it can be any process in a uniform system), and the rest of processes is modeled as its environment. This approach became known as Ptolemaic system analysis. The use of environment predicates to describe the environment hides parameterization and makes the abstract system finite state.

Monotonic abstraction [1] is a model checking approach for which decidability theorems were presented in [4]. There is a number of verification methods for parameterized systems based on monotonic abstraction [6, 9, 5] where an approximate transition relation is used for reachability analysis instead of the one defined by the parameterized system. The approximate transition relation must be monotonic with respect to the sub-word relation so that a symbolic reachability algorithm can be applied for reachability analysis. In [6] the transition relation is over-approximated by elimination of the processes that violate the guarding condition. The applied symbolic backward reachability algorithm is much more efficient than the algorithms in regular model checking which use transducers and regular languages. An improved verification algorithm was proposed later in [9] where context-sensitive constraints were introduced which helped to verify a broader range of parameterized systems and verify successfully the systems, which returned
false positives with the previous method. The following improvement of the monotonic abstraction approach is represented by CEGAR (counterexample-guided abstraction refinement) framework [5]. A set of configurations called a "Safety Zone" is extracted by CEGAR algorithm after each iteration of reachability analysis. That set is based on the found counterexamples and is used to refine the approximate transition relation in the following iteration.

Thesis Outline

The thesis is organized as follows: Chapter 2 introduces models of systems, which will be considered in this thesis as well as a language for their specifications. Chapter 3 describes our approach of proving safety properties by considering multiple over-approximations. Chapter 4 introduces an intermediate representation for sets of global states and sets of transitions required to compute over-approximated sets of reachable states. The purpose of intermediate representation is to introduce a formal language describing abstracted systems. Chapter 5 shows how to encode the intermediate representation as Presburger formulas and how to perform effectively basic set and transition operations. An algorithm is presented at the end of that chapter describing how to compute a(n over-approximated) set of reachable states and prove a safety property.

Chapter 6 describes the experimental evaluation of few examples attached in the Appendix together with an overview of other work related to these examples. Related work is covered in Chapter 7 followed by conclusions and future work presented in Chapter 8.
Chapter 2

Parameterized Systems

In Section 2.1 we start with a general introduction to the systems we consider in this thesis. Sections 2.2–2.4 present a notation for describing such systems, while Section 2.5 sums up everything in definitions of uniform state transition systems and families of such systems, which will be used in the rest of the thesis. Section 2.6 introduces basic notions concerning reachability analysis while Section 2.7 formalizes the notion of safety properties considered in this thesis. The chapter ends with a short summary (Section 2.8).

2.1 State Transition Systems

A basic subject of consideration is a state transition system (later, for the sake of brevity, it will be referred to simply as a system) consisting of \( n \) components (or processes) indexed from 0 to \( n-1 \), which run concurrently. The number of processes in a system is stationary. Each component \( p \) has a fixed set \( \{K, \ldots, L\} \) of registers. The domain of a register may be finite or infinite. An assignment \( \sigma = \{K \mapsto k, \ldots, L \mapsto l\} \) of values to registers is called a local state of that component. The set of all possible assignments is denoted as \( \Sigma \) and any set of local states as \( \Sigma \). We consider uniform systems, where the number of processes is finite and all processes have uniform sets of registers and run the same algorithm (represented by a transition relation, described later) up to parameterization. Therefore, relevant registers of different processes can be arranged into arrays \( K, \ldots, L \), such that the position of a register in the array corresponds to its process index\(^1\). For example, the local state of the \( i \)-th component is represented by an assignment \( \{K[i] \mapsto k_i, \ldots, L[i] \mapsto l_i\} \). Sometimes instead of mentioning all registers \( K, \ldots, L \) of a component we use the notation \( X \) to represent one compound register which takes a vector \( (k, \ldots, l) \in K \times \ldots \times L \) of values. A particular

\(^1\)We assume that all the registers of different processes are local (i.e. disjoint).
compound value is referred to as \( a \in A \), where \( A \) is a uniform domain of registers \( L, \ldots, K \) of a component. Accordingly, the vector \( X \) is used to represent an array of registers \( X[0], \ldots, X[n-1] \) of every component in a system. The set \( \{K[0], \ldots, K[n-1], \ldots, L[0], \ldots, L[n-1]\} \) of all registers in a system is called the set of system variables or the system set.

A global state (or global configuration) \( s \) is an assignment of values to all registers from the system set, that is, the union \( \sigma_0 \cup \ldots \cup \sigma_{n-1} \) of local states of all \( n \) components. The set \( S \) of all possible assignments to the system set constitutes the system’s state space. The set of system’s initial states \( S_I \) is a given subset of its state space: \( S_I \subseteq S \).

### Transition Relation

The behavior of a system is described by the transition relation \( T \), which is a set of pairs of global states \( (s, s') \). Such a pair is called an atomic transition. We consider transition relations where each pair of global states differs in at most one local state. To put it simply, only one component in a system can change its local state at a time. We call by the moving component the one, whose local states are different in that pair of global states. The first global state \( s \) in a pair \( (s, s') \) is called the current state, and the second, \( s' \), the next state. The same applies to the local states of the moving component. The pair \( (\sigma, \sigma') \) of the current local state and the next local state of the moving component in an atomic transition is called its local transition. We call state \( s' \) a successor of state \( s \) and state \( s \) a predecessor of \( s' \) whenever there exists an atomic transition \( (s, s') \in T \). An atomic transition is idling iff both the current and the next states are equal. (In that case the notion of the moving component is undetermined). Any subset \( \tau \subseteq T \) of atomic transitions is called a meta-transition.

The constraint that only one process can change its local state at a time perfectly suites for modeling the real world of parameterized system such as mutual exclusion protocols and communication protocols which are considered in this thesis. Though it can be a limitation when it comes, for example, to hardware systems where processes can change their states synchronously.

### Process Indexes

As we have mentioned earlier, we assume that all the processes in a system of size \( n \) are indexed from 0 to \( n-1 \). In some algorithms which we will consider, the behavior of a process may depend on its index. For example, in the Lamport’s Bakery algorithm (Appendix B) a process compares its own index with the indexes of other processes in certain cases in order to decide whether it can go into the critical section or not. One can think of a process index as an additional local register with the constant value where the index of that process is kept. However, because of the special treatment of the process index in this work, we will not consider the assignment of such
registers to indexes as a part of a global state (or a local state, respectively), but rather talk about local states and indexes separately.

## 2.2 Symbolic Notation

In this section we introduce a basic notation of sets of global states and sets of transitions in the form of Presburger formulas.

Let $\mathbf{V}$ be the set of variables and $\mathbf{C}$ be the set of constants $\{0, 1\}$. The syntax of a *Presburger term* $E$ is defined by the following rules in BNF notation:

$$E := \mathbf{V} \mid \mathbf{C} \mid -E \mid E + E$$

A constant representing an integer $k$ can be represented as a term $(1+\ldots+1)$ repeated $k$ times. If $k$ is an integer constant and $V$ is a variable, then $kV$ is a syntactic sugar for the term $V + \ldots + V$ repeated $k$ times.

A *Presburger atom* is defined by the following rule:

$$A := E \leq E$$

Other atoms containing $=, <, >, \geq, \text{True}$ and $\text{False}$ can be derived based on $\leq$ and the connectives below.

A *Presburger formula* is defined by the following rules:

$$F := A \mid \lnot F \mid F \lor F \mid \exists V. F$$

The connectives $\land, \rightarrow$ and $\forall$ can be derived based on the existing ones.

A solution of a Presburger formula $F$ is an assignment of integer values to the free variables, such that $F$ is true.

### State Formulas

A *state formula* is a Presburger formula built on system variables. The semantics of a state formula $F$ is the set $\{F\}$ of global states corresponding to assignments of all system variables such that $F$ is true and any value assigned to a system variable is from the domain of the corresponding register.

**Example 2.2.1** Assume, that we have a system of size $n = 3$ with two local registers $L$ and $K$ per process. The state formula

$$L[0] = 0 \land 0 < K[0] \leq 3 \land (L[1] = 1 \lor L[1] = 2) \land K[1] = 1 \land L[2] = 5$$

encodes the set of all global states $s$, where $s(L[0]) = 0$, $s(K[0])$ is more than 0 but less than or equal to 3, $s(L[1]) = 1$ or $s(L[1]) = 2$, $s(K[1]) = 1$ and $s(L[2]) = 5$, while $s(K[2])$ can take any value from the domain of register $K[2]$. $\Box$
CHAPTER 2. PARAMETERIZED SYSTEMS

Transition Formulas

To describe a meta-transition we use transition formulas of Presburger arithmetic. The syntax is the same as for state formulas, with the only difference that we use two copies of system variables – one unprimed copy to describe the current state and one primed copy for the next state. The solution of a transition formula $\rho$ is an assignment of values to both copies of the system variables such that $\rho$ is true. The semantics of a transition formula $\rho$ is the meta-transition $\tau = \llbracket \rho \rrbracket$ such that for every atomic transition $(s, s') \in \tau$ the assignment $s \cup s'|_{V \rightarrow V'}$ is a solution of $\rho$, where $s'|_{V \rightarrow V'}$ is $s'$ with all the variables renamed to their primed versions, and where all values assigned to primed or unprimed versions of the system variables fit into the domain of the corresponding register.

Example 2.2.2 Assume, that in a system of size $n = 2$ we have one register $L$ per process. The semantics of the transition formula $L[0] = 0 \land L'[0] = 1 \land (L[1] = 2 \lor L[1] = 3) \land L'[1] = L[1]$ is the following meta-transition consisting of two atomic transitions:

\[
\{(L[0] \mapsto 0, L[1] \mapsto 2), (L[0] \mapsto 1, L[1] \mapsto 2)\},
\]
\[
\{(L[0] \mapsto 0, L[1] \mapsto 3), (L[0] \mapsto 1, L[1] \mapsto 3)\}\]

\[\square\]

2.3 State and Transition Schemas

We introduce here schema parameters – a syntactic sugar, which facilitates writing some long formulas with a common pattern in a more concise way. The main advantage of schema parameter is that after the parameterization is introduced, we will be able to write formulas, where the number of system variables depends on a (possibly unbounded) parameter.

If we have an array $X = (X[0], \ldots, X[n-1])$ of variables, then we can introduce into the syntax a schema variable $X[I]$ which is a syntactic sugar, where $I$ is a schema parameter ranging over a subset of indexes $0, \ldots, n - 1$. In our case, a schema variable ranges over an array of registers of the same type, e.g. a schema variable $L[I]$ will range over $L[0], \ldots, L[n-1]$ depending on the value of the schema parameter $I$.

Let $F(I)$ be a system (or transition) formula, where the set of the variables is extended with a set of schema variables with schema parameter $I$. State Schemas (or transition schemas) $SF$ have the following syntax:

\[SF ::= F | \bigvee_{\text{range}(I)} F(I) | \bigwedge_{\text{range}(I)} F(I) | \neg SF | SF \lor SF\]

where $\text{range}(I)$ is a Presburger formula of $I$ describing its range. De-sugaring of constructs $\bigvee_{\text{range}(I)} F(I)$ and $\bigwedge_{\text{range}(I)} F(I)$ is done according to the
following rules:

\[ \bigvee_{\text{range}(I)} F(I) \text{ denotes } F(i_1) \lor \ldots \lor F(i_n) \]

and

\[ \bigwedge_{\text{range}(I)} F(I) \text{ denotes } F(i_1) \land \ldots \land F(i_n) \]

where \( \{i_1, \ldots, i_n\} \) is a set of solutions of the formula \( \text{range}(I) \).

**Example 2.3.1** Assume, that we have a system of size \( n = 4 \) (therefore \( I = \{0, 1, 2, 3\} \)) with two registers \( L \) and \( K \) per process. The state schema

\[
L[0] = 0 \land 0 < K[0] \leq 3 \land \bigwedge_{1 \leq I \leq 3} [(I = K[0]) \rightarrow (L[I] = 1)]
\]

means the set of all global states \( s \), where \( s(L[0]) = 0 \), \( s(K[0]) \in \{1, 2, 3\} \) and \( s(L[I]) = 1 \) whenever \( I = s(K[0]) \).

Let us now introduce restrictions on the syntax of state and transition schemas which is required to work with such formulas later.

The *normal form*\(^2\) of a state schema has the following syntax

\[
\bigwedge_I f(I, X[I])
\]

where \( f(I, X[I]) \) is a quantifier-free Presburger formula with the only variables \( I \) and \( X[I] \). The function \( f(I, X[I]) \) is called a *specification* of the set of global states.

**Example 2.3.2** Assume that we have a set of global states in the system of size \( n = 5 \) described by the following state formula

\[
X[0] = 2 \land X[1] = 3 \land 0 \leq X[2] \leq 3 \land 0 \leq X[3] \leq 3 \land 0 \leq X[4] \leq 3.
\]

This set can also be described by the following state schema in the normal form:

\[
\bigwedge_{0 \leq I \leq 4} \begin{bmatrix}
 (I = 0 \land X[I] = 2) \\
 \lor \\
 (I = 1 \land X[I] = 3) \\
 \lor \\
 (2 \leq I \leq 4 \land 0 \leq X[I] \leq 3) 
\end{bmatrix}
\]

\( ^2 \text{In the strict sense, it is not a normal form of a state schema because not every state schema can be expressed in that form. It is rather a normal form of a state schema describing a disjunctive set of states, which will be introduced in Section 4.1.} \)
In what follows we will often use the shorthand $\mathcal{J}(i)$ which stands for the conjunct

$$
\bigwedge_{0 \leq J < n \land J \neq i} X[J] = X'[J]
$$

where $n$ is the system size and $i$ is the index of a component. This conjunct is called the idling condition and is used in transition schemas to encode the condition, that primed and unprimed local states of all the processes but the one with index $i$ (the moving one) remain the same in the transition(s). It can be also written as $\mathcal{J}(I)$ whenever it is a part of a bigger transition schema where the index of the moving component $I$ is not a constant.

Let us now introduce two basic types of meta-transitions which we will use to model systems. The first type describes a situation when a component always can change its local state irrespectively of all other components in the systems. That is, the same change can happen for any configuration of the rest of the system. For example, a process in a computer system sets a certain variable to zero at some point of its execution regardless of the states of other processes in the system. However, the change might depend on its own local state and the index of that component. We can describe such a situation by a transition formula of the equivalent form

$$
u(X[i], X'[i]) \land \mathcal{J}(i)$$

The part $\nu(X[i], X'[i])$ represents dependency between the current local state $X[i]$ of the component $i$ and its next local state $X'[i]$. The idling condition states, that the rest of the system outside of the moving component does not change its state. As we can see, such a formula does not put any other restriction on the components other than $i$. Note, that it is possible to describe any (possibly unbounded) set of $k$ such transitions

$$
\left[ \nu_1(X[i], X'[i]) \land \mathcal{J}(i) \right] \lor \ldots \lor \left[ \nu_k(X[i], X'[i]) \land \mathcal{J}(i) \right]
$$

by one formula which can be schematically presented as

$$
u(X[i], X'[i]) \land \mathcal{J}(i).$$

In order to model transitions of such type for all components (or any subset of all components) we can take the disjunction of particular formulas for each component, as we did for unsynchronized transitions:

$$
u_0(X[0], X'[0]) \land \mathcal{J}(0)
\lor \ldots \lor
\nu_{n-1}(X[n-1], X'[n-1]) \land \mathcal{J}(n-1)$$
which presented in the equivalent form as a transition schema
\[ \bigvee_{0 \leq I < n} (u(I, X[I], X'[I]) \land J(I)) \]
where \( u(I, X[I], X'[I]) \) is a quantifier-free Presburger formula built on variables \( I, X[I] \) and \( X'[I] \). A meta-transition which can be presented this way is called an unsynchronized transition, and the transition schema
\[ \bigvee_{0 \leq I < n} (u(I, X[I], X'[I]) \land J(I)) \]
is called the normal form of an unsynchronized transition. The function \( u(I, X[I], X'[I]) \) is called the specification of an unsynchronized transition.

The second type of meta-transitions relates to the situation when a component can perform a local transition only if some other components are in certain states. For example, process 3 can set its variable \( X \) to zero only if the variable \( Y \) of process 1 has a value in the range 5...7. A meta-transition, where local transitions of component \( i \) are dependent on the local state of component \( j \) (called guard in this context), can be described by a transition formula in the equivalent form
\[ \text{syn}_{i}(X[i], X'[i], X[j]) \land J(i) \]
which can be generalized as dependency on any other component in the system and presented schematically in the equivalent form as a transition schema
\[ \bigwedge_{0 \leq G < n \land G \neq i} \text{syn}_{i}(X[i], X'[i], G, X[G]) \land J(i) \]
That is, the generalized transition schema above can express the dependency of a local transition (or even some sets of local transitions, though not all possible) on local states of any subset of the processes in the system.

In order to model transitions of such type for all components (or any subset of all components) we can take the disjunction of particular formulas for each component, as we did for unsynchronized transitions:
\[ \bigvee_{0 \leq G < n \land G \neq 0} \text{syn}_{g}(X[0], X'[0], G, X[G]) \land J(0) \]
\[ \lor \ldots \lor \]

14
CHAPTER 2. PARAMETERIZED SYSTEMS

\[ \bigwedge_{0 \leq G < n \land G \neq n-1} \text{syn}_{n-1}(X[n-1], X'[n-1], G, X[G]) \land J(n-1) \]

which presented in the equivalent form as a transition schema

\[ \bigvee_{0 \leq I < n} \left( \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right) \]

where \( \text{syn}(I, X[I], X'[I], G, X[G]) \) is a quantifier-free Presburger formula built on variables \( I, G, X[I], X'[I] \) and \( X[G] \). A meta-transition which is not an unsynchronized transition and can be presented this way is called a synchronized transition, and the transition schema

\[ \bigvee_{0 \leq I < n} \left( \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right) \]

is called the normal form of a synchronized transition. The function \( \text{syn}(I, X[I], X'[I], G, X[G]) \) is called the specification of a synchronized transition.

**Example 2.3.3** Assume, that we have a system of size \( n = 4 \) with one register \( L \) per process. The transition schema

\[ \bigvee_{0 \leq I < 3} \ L[I] = 1 \land L'[I] = 2 \land 3 \leq L[0] \leq 7 \land J(I) \]

encodes a set of atomic transitions such that any component \( I \in 1, \ldots, 3 \) can make a move from the state 1 to state 2 under condition that the component 0 is in one of the states 3, \ldots, 7. A normal form of this transition can be the following:

\[ \bigvee_{0 \leq I < 4} \left( \bigwedge_{0 \leq G < 4 \land G \neq I} \left( 0 < I < 4 \land L[I] = 1 \land L'[I] = 2 \land (G = 0 \rightarrow 3 \leq L[G] \leq 7) \right) \right) \land J(I) \]

The situation, when a transition of a component depends on local states of all other processes, is called atomicity assumption. For many real systems models with atomicity assumption are unrealistic. For example, a process in a distributed environment cannot check the states of all other processes in one atomic step. Nevertheless, the use of atomicity assumption is a recognized abstraction technique which facilitates verification of many parameterized systems ([64], [48], [62], [15]). For instance, there is no currently described method of automatic verification of Szymanski algorithm without atomicity assumption. Models of well-known algorithms with atomicity assumptions can be found in the appendix. For example, synchronized
transitions $\rho_1$ and $\rho_2$ of Simplified Bakery algorithm in Appendix A are modeled with atomicity assumption.

Note, that there are meta-transitions which are neither synchronized nor unsynchronized transitions, although any meta-transition can be expressed as a (possibly unbounded) union of the presented two types (indeed, any atomic transition can be expressed as a synchronized meta-transition and a meta-transition by definition is a set of atomic transitions).

### 2.4 Parameterized formulas

Here we introduce parameterization of state and transition formulas. Its purpose is to describe all instances of a family of uniform systems or their behavior by one formula.

A state (or transition) formula (as well as a state schema or a transition schema) can be extended with a finite set of family parameters (referred to later simply as parameters). The assignment of values to all parameters in such a parameterized state (or transition) formula is called significant if the formula characterizes a non-empty set of global states (or transitions) under this assignment. The semantics of a parameterized state (or transition) formula is the family of state (or transition) formulas for all significant assignments to the parameters. Consequently, a parameterized state (or transition) formula characterizes the family of sets of global states (or transitions) represented by each state (or transition) formula in the family.

In particular, introduction of the family parameter $N$ representing the size of a system facilitates describing a family of sets of global states by one formula $F(N)$.

**Example 2.4.1** Assume, that we have two systems of size $n = 2$ and $n = 3$ with one register $L$ per process. The parameterized state formula $F(N) = (L[0] = 0 \land 0 < L[1] \leq 3 \land N = 2) \lor (L[0] = 0 \land L[1] = 1 \land L[2] = 1 \land N = 3)$ encodes the family of two sets of global states: one set consisting of states $[F(2)]$:

- $\{L[0] \mapsto 0, L[1] \mapsto 1\}$
- $\{L[0] \mapsto 0, L[1] \mapsto 2\}$
- $\{L[0] \mapsto 0, L[1] \mapsto 3\}$

for the system of size $n = 2$, and another set consisting of just one global state $[F(3)] = \{\{L[0] \mapsto 1, L[1] \mapsto 1, L[2] \mapsto 1\}\}$ for the system of size $n = 3$.

Family parameters together with schema parameters provide a formalism to describe a family of sets of global states (or sets of transitions) for an unbounded family of uniform systems.
Example 2.4.2 Assume, that we have a family of uniform systems with \( n \geq 2 \) processes and one register \( L \) per process. The parameterized state schema

\[
F(N) \equiv N \geq 2 \land L[0] = 1 \land \bigwedge_{1 \leq j < N} L[j] = 0
\]

describes an infinite family \( \{F(2), F(3), \ldots\} \) of state formulas where

\[
\llbracket F(2) \rrbracket = \{ \{ L[0] \mapsto 1, L[1] \mapsto 0 \} \}
\]
\[
\llbracket F(3) \rrbracket = \{ \{ L[0] \mapsto 1, L[1] \mapsto 0, L[2] \mapsto 0 \} \}
\]
\[
\llbracket F(4) \rrbracket = \{ \{ L[0] \mapsto 1, L[1] \mapsto 0, L[2] \mapsto 0, L[3] \mapsto 0 \} \}
\]
\[
\ldots
\]

A parameterized transition formula (or transition schema) encodes a possibly infinite family of transition formulas (or transition schemas) for each significant assignment of values to the family parameters. In what follows, we will often use the idling condition \( J(i, N) \) parameterized by size \( N \) which is the shorthand for the expression

\[
\bigwedge_{0 \leq j < N \land j \neq i} X[j] = X'[j]
\]

where \( i \) assumed to be the moving component. Whenever the idling condition is a part of a bigger transition schema where the index of the moving component \( I \) is not a constant, then we will write the idling condition as \( J(I, N) \).

Example 2.4.3 The following parameterized transition formula \( F(N) \) represents an infinite family of meta-transitions, one per system size \( N \geq 2 \) (assuming that we have one register \( L \) per process):

\[
F(N) \equiv N \geq 2 \land L[1] = 1 \land L'[1] = 2 \land L[0] \neq 7 \land J(1, N).
\]

We can see it as the following family of transition formulas for different values of parameter \( N \):

\[
\{ L[1] = 1 \land L'[1] = 2 \land L[0] \neq 7 \land J(1, N) \}_{N \geq 2}.
\]

Its meaning is that for each system of size \( N \geq 2 \), whenever process 0 is in a local state other than 7, then process 1 can make a move from state 1 to state 2. \( \square \)
 CHAPTER 2. PARAMETERIZED SYSTEMS

The definitions of the normal forms of state and transition schemas introduced in Section 2.3 can be extended to reflect parameterization. Assume, for example, that \( N \) is a parameter for the system size, then

\[
\bigwedge_{0 \leq I < N} f(I, X[I], N)
\]

is the normal form of a parameterized state schema, while

\[
\bigvee_{0 \leq I < N} (u(I, X[I], X'[I], N) \land J(I, N)),
\]

and

\[
\bigvee_{0 \leq I < N} \left[ \bigwedge_{0 \leq G < N \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G], N) \land J(I, N) \right]
\]

are the normal forms of parameterized transition schemas. A family of synchronized (or unsynchronized) transitions is called a parameterized synchronized (or unsynchronized) transition.

Example 2.4.4 The following parameterized transition schema

\[
\bigvee_{0 \leq I < N} [N \geq 2 \land X[I] = 1 \land X'[I] = 2 \land J(I, N)]
\]

is in a normal form and encodes a family of unsynchronized transitions, one per system size \( N \geq 2 \), such that any process \( 0 \leq i < N \) in the system of size \( N \) can make a move from state 1 to state 2.

2.5 System Definition

Let us now provide a formal definition of a system and a family of systems, summarizing all we have presented so far. These definitions will be used throughout the thesis.

A state transition system is a tuple \((X, S, F, \rho)\) where

- \( X \) is the system set as defined in Section 2.1,
- \( S \) is the set of all possible assignments to the system set as defined in Section 2.1,
- \( F \) is a state formula describing the set \( S_I = \llbracket F \rrbracket \) of initial states, and
- \( \rho \) is a transition formula describing the transition relation \( T = \llbracket \rho \rrbracket \) of the state transition system.
CHAPTER 2. PARAMETERIZED SYSTEMS

A uniform state transition system \((X, S, F, \rho)\) (or simply a system) is a state transition system where \(F\) is a state schema in the normal form and \(\rho\) is a disjunction of transition schemas for synchronized and unsynchronized transitions. For the sake of convenience, we will often provide a set of transition schemas describing synchronized and unsynchronized transitions separately, instead of writing \(\rho\) as a long disjunctive formula directly.

In this thesis we consider possibly infinite families of uniform state transition systems (often referred to simply as families of systems for the sake of brevity), which are families \(\{(X_n, S_n, F(n), \rho(n))\}_{n \in \mathbb{M}}\) of tuples \((X_n, S_n, F(n), \rho(n))\) such that

- \(\mathbb{M}\) is a set of system sizes;
- \(\{X_n\}_{n \in \mathbb{M}}\) is a family of system sets – one for every system size \(n \in \mathbb{M}\);
- \(\{S_n\}_{n \in \mathbb{M}}\) is a family of all possible assignments \(S_n\) to the system set \(X_n\);
- \(F(N)\) is a parameterized state schema in the normal form describing a possibly infinite family \(\{F(n)\}_{n \in \mathbb{M}}\) of state formulas encoding sets of initial states for each system in the family of size \(n \in \mathbb{M}\);
- \(\rho(N)\) is a finite disjunction of parameterized transition schemas in their normal forms and describes a family \(\{\rho(n)\}_{n \in \mathbb{M}}\) of transition schemas which encode the transition relations for each system of size \(n \in \mathbb{M}\) in the family.

An example of a family of uniform state transition systems representing a family of systems running Szymanski algorithm with different numbers of processes in a system is presented in Appendix C.

As it was mentioned earlier, the proposed notation has a limited expressive power, that is, it is not possible to express arbitrary sets of states of an arbitrary family of systems. The situation can be improved to some extent by introducing additional (non-size) parameters. For example, imagine a system with \(n\) numbered processes, where all processes run a uniform algorithm but one has a completely different behavior. We may wish to define a family of system where in one system the special process has index \(P = 0\), in another system it has index \(P = 1\), etc. Therefore, the index \(P\) of a special process is a parameter for such a family of systems. To make things more complicated we may consider even a family of systems with a size parameter and non-size parameters.

A family of uniform state transition systems with a parameter (or a set of parameters) is a family \(\{(X_n, S_n, F(n, p), \rho(n, p))\}_{n \in \mathbb{M}, p \in \text{DOM}(P)}\) of tuples such that

- \(\{X_n\}_{n \in \mathbb{M}}\) and \(\{S_n\}_{n \in \mathbb{M}}\) have the same meaning as in the previous definition;
• $F(N,P)$ is a parameterized state schema in the normal form describing an infinite family $\{F(n,p)\}_{n\in\mathbb{M},p\in\text{DOM}(P)}$ of state schemas encoding sets of initial states for each system in the family of size $n \in \mathbb{M}$ and each value $p$ of the parameter $P$;

• $\rho(N,P)$ is a finite disjunction of parameterized transition schemas in their normal forms describing a family $\{\rho(n,p)\}_{n\in\mathbb{M},p\in\text{DOM}(P)}$ of transition schemas which encode the transition relations for each system of size $n \in \mathbb{M}$ and each value $p$ of the parameter $P$ in the family.

We may also consider a family of uniform state transition systems without a size parameter, in which case it is a family $\{(X,S,F(p),\rho(p))\}_{p\in\text{DOM}(P)}$ of tuples. Every member of such a family is a state transition system and has the same set of registers and the same state space, but the set of initial states and the transition relation depend on the value of the parameter $P$.

### 2.6 Reachability

Safety properties of the form

certain system configurations never occur

are understood as

certain system configurations are never reached from some given set of initial configurations

and their verification is reduced to reachability problem. Let us provide a formal definition for the set of reachable states in a system.

**Definition 2.6.1** The set of reachable states of a system $(X,S,F,\rho)$ is the least set $S$ of global states such that

- if $s \in [F]$ then $s \in S$, where $[F]$ is the set of initial states, and
- if $s \in S$ and $(s,s') \in [\rho]$ , then $s' \in S$

The same idea applies to families of systems. The family of sets of reachable states of a family $\{\{X_n,S_n,F(n),\rho(n)\}\}_{n\in\mathbb{M}}$ of uniform state transition systems is the family $\{S_n\}_{n\in\mathbb{M}}$ of sets of global states such that each $S_n$ is the set of reachable states of the system $(X_n,S_n,F(n),\rho(n))$.

In the same manner we define the families $\{S_{n,p}\}_{n\in\mathbb{M},p\in\text{DOM}(P)}$ of sets of reachable states of uniform state transition systems $\{(X_n,S_n,F(n,p),\rho(n,p))\}_{n\in\mathbb{M},p\in\text{DOM}(P)}$ parameterized by size parameter $N$ and non-size parameter $P$, so that $S_{n,p}$ is the set of reachable states of $(X_n,S_n,F(n,p),\rho(n,p))$.

We say, that a meta-transition $\tau$ is applicable to a global state $s$ iff there exists an atomic transition $(s_1,s_2) \in \tau$ such that $s = s_1$. The result of
application of a transition \( \tau \) to a global state \( s \) is the set of states defined in the following way: \( \tau(s) = \{ s' \mid (s, s') \in \tau \} \). Applicability can be extended to sets of states. The result of application of a meta-transition \( \tau \) to a set of global states \( S \) is denoted as \( \tau(S) \) and results in a set of states \( S' = \bigcup_{s \in S} \tau(s) \).

Again, the same idea is extended to cover families of systems. A family of meta-transitions \( \{ \tau_n \}_{n \in \mathbb{M}} \) is applicable to a family of sets of global states \( \{ S_n \}_{n \in \mathbb{M}} \) iff there exists a global state \( s \) in a set of global states \( S_n \) for some system size \( n \in \mathbb{M} \) such that \((s, s') \in \tau_n \) for some \( s' \). The result of application of a family of meta-transitions \( \{ \tau_n \}_{n \in \mathbb{M}} \) to a family \( \{ S_n \}_{n \in \mathbb{M}} \) of sets of global states results in a family \( \{ \tau_n(S_n) \}_{n \in \mathbb{M}} \) of sets of global states. Of course, the definitions for families of meta-transitions is valid for parameters other than size as well.

In the remainder of this section we introduce two reachability operations on a set of global states using a meta-transition. The reason for introducing those operations is that they can be performed efficiently in the notation presented later (Section 5.4). Afterwards we establish that the application of those operations in a fixed-point manner on a set of initial states will result in the set of reachable states. We present all the definitions and results only for a single system for the sake of brevity, but they can be easily extended to families of uniform state transition systems in the same manner as we did previously in this section.

A parallel application of a meta-transition \( \tau \) on a set \( S \) of global states is denoted \( \tau^\parallel(S) \) and results in the set \( S' \) of global states \( s_m \) such that there exists a non-empty sequence \((s_1, s_2), (s_2, s_3), \ldots, (s_{m-1}, s_m)\) of atomic transitions where

- \( s_1 \in S \), and
- \((s_i, s_{i+1}) \in \tau \) for \( 1 \leq i < m \), and
- every transition \((s, s_i)\) in the sequence is either idle \( (s_i = s_{i+1}) \), or \( s_{i+1} \in \tau(s_i) \) and its moving component is different from the moving components of all previous non-idle transitions in the sequence.

An aggregate parallel application of a meta-transition \( \tau \) onto a set \( S \) of global states is defined as \( \tau^\#(S) = S \cup \tau^\parallel(S) \).

**Proposition 2.6.2** Assume that \((X, S, F, \rho)\) is a uniform state transition system and \( T \) is a set of transition schemas such that \( \rho \) is a disjunction of all formulas described by transition schemas in \( T \). Let \( S \) be the least set of global states such that

- \( s \in S \) whenever \( s \in [F] \), and
- \( \tau^\parallel(S) \subseteq S \) whenever \( s \in S \) and there is a transition schema in \( T \) describing meta-transition \( \tau \),
then $S$ is the set of reachable states.

\[ \text{Proof: } \] For any transition $(s_1, s_2) \in [\rho]$ there must be a corresponding transition schema in $T$ describing a meta-transition $\tau$ such that $(s_1, s_2) \in \tau$. Therefore, according to the definition of the parallel application, $s_2 \in S$ whenever $s_1 \in S$. \[ \square \]

Proposition 2.6.2 will be useful when we specify $\rho$ as a set (disjunction) of separate transition schemas and apply them individually.

### 2.7 Safety Property

Formally, a safety property is a statement that the set $S_R$ of reachable states of a system is safe with respect to a given set of safe states $P$, that is,

$$ S_R \subseteq P. $$

We often refer to $P$ simply as a safety property.

We consider safety properties which can be expressed as a state schema in the following normal form:

$$ P \equiv \bigwedge_{0 \leq I_1 < N} \ldots \bigwedge_{0 \leq I_k < N} \bigwedge_{I_1 \neq I_2 \land \ldots \land I_k \neq I_{k-1}} f(I_1, X[I_1], \ldots, I_k, X[I_k]) $$

where $f(I_1, X[I_1], \ldots, I_k, X[I_k])$ is a quantifier-free Presburger formula built only with the variables $I_1, \ldots, I_k$ and $X[I_1], \ldots, X[I_k]$. Obviously, the number $k$ of schema parameters must be fixed.

A property can also be parameterized, for example, by the system size $N$, in which case its normal form will be

$$ P(N) \equiv \bigwedge_{0 \leq I_1 < N} \ldots \bigwedge_{0 \leq I_k < N} \bigwedge_{I_1 \neq I_2 \land \ldots \land I_k \neq I_{k-1}} f(I_1, X[I_1], \ldots, I_k, X[I_k], N) $$

**Example 2.7.1** Assume that we have a family of systems with sizes $N \geq 2$ where the local state $\{X \mapsto cs\}$ of a process represents execution of the critical code by that process. The family of sets of systems configurations where at most one process can execute the critical code at a time can be expresses by the following state schema:

$$ \bigwedge_{0 \leq I_1 < N} \bigwedge_{0 \leq I_2 < N} \bigwedge_{I_1 \neq I_2} X[I_1] \neq cs \lor X[I_2] \neq cs. $$

\[ \square \]
2.8 Summary

In this section we have defined systems and families of systems in a formal way, introduced a basic notation to describe them as well as (some) safety properties, and shown how reachability operations could be expressed with the help of this notation. The thesis will consider all systems which can be modeled in the presented way. Despite all the limitations we have introduced, a broad range of parameterized systems can be modeled this way (look in the appendix for some examples). However, a straightforward reachability analysis of an unbounded family of systems will be also “unbounded”, as we will operate on unbounded state and transition formulas. Starting from the next chapter we present our approach of transforming parameterized systems into abstract ones where an effective (over-approximating) reachability analysis is possible and potentially can prove some safety properties.
Chapter 3

Conceptual Approach

3.1 Introduction

Automatic verification of parameterized systems is a non-trivial task, because the state space in general is unbounded as is the number of systems we have to consider. Before a model of a system reaches the verification stage, it probably has been already verified to some extent in human mind by its designer(s) during the design stage. Indeed, the publications of the most well-known algorithms and protocols include some reasoning about their correctness (For example, [51] and [63] – publications of Lamport’s Bakery algorithm and Szymanski Mutual Exclusion algorithm). Therefore, it looks tempting to create a verification framework formalizing the way the system designer would think in order to check a property.

We consider such safety properties of parameterized systems, where we are interested first of all in a mutual relationship between a fixed number of arbitrary processes and the rest of the system. In particular, the word “fixed” means that it is independent of the system size and is the same for every instance of the family.

A classical example is the mutual exclusion property in a parameterized system of size $n \geq 2$:

\textit{no two processes can be in the critical section at the same time.}

That is, we have a relation between a fixed number of processes (two in our example) which ranges over the set of all $n^2$ pairs in the system, and the size $n$ of the system is unbounded. The question now arises of how the designer reasons about the correctness. The answer, for example, may be received from studying the correctness proofs submitted along with algorithms. From there we can learn, that algorithm designers usually consider the system from the point of view of few reference processes. Many proofs start with sentences like “Let us consider any two process $p_i$ and $p_j$” and continue with an analysis of what can happen with these processes and the rest of
the system. Such a method of analysis is basically a transformation of a system with \( n \) processes (where \( n \) may be unbounded) into an abstract finite state transition system of \( k + 1 \) components with \( k \) reference processes and one component representing the rest of the systems (called sometimes the environment of the reference processes). Although the number of processes in the environment may be unbounded, we treat it as a single unit (or, alternatively, a fixed number of single units). By this means we abstract a system with unbounded number of process into one with a bounded number of components.

The idea behind our framework follows the same way of thinking and is based on the abstraction of a parameterized system to a finite or infinite state transition system but with a fixed number of components, so the problem is reduced to model-checking of the abstracted system. Although the abstracted system usually has an infinite state space the problem is still easier than the original one.

The usual algorithms to prove a safety property are

- to perform forward reachability analysis by construction of the set of reachable states starting from the set of initial states and prove that the set of reachable states is safe (is contained in the property), or
- perform backward reachability analysis by construction of the set of all states from which unsafe states can be reached and prove that there is no overlap with the set of initial states.

Although in this thesis we will concentrate on the forward reachability analysis, both forward and backward reachability analyzes are possible to apply with our method.

The component which represents the environment is formed from an unbounded number of processes and therefore has an infinite state space. We use over-approximated transition relation in order to provide for termination. Over-approximation in our work is achieved by removing some synchronization between processes and its basic principles are explained in Section 3.4. The proposed over-approximation schema is parameterized by the reference processes, therefore the transition relation for an abstracted system and, obviously, the (over-approximated) set of reachable states will be parameterized by the choice of reference processes. The encoding schema proposed in Chapters 4 and 5 allow to perform (over-approximating) reachability analysis when the choice of reference processes is a parameter (as well as the system size), therefore we can talk about parallel over-approximations defined by our approach.

In this section we present how we prove safety properties in terms of reachability analysis with many parallel over-approximations. In particular, Section 3.2 explains the idea of partitioning the property and then constructing and verifying separate over-approximations for each partition. In Sections 3.3-3.4 we concentrate on how we define over-approximated sets
CHAPTER 3. CONCEPTUAL APPROACH

w.r.t. the transition relation and parameterize them by each partition of the property. Section 3.5 makes a short overview of the whole approach.

As before, for the sake of simplicity, we run the whole presentation for a single system of fixed size $n$. At a later stage we will show how to parameterize the system size $n$, while implementing the ideas contained in this section, and reason about (possibly) infinite families of uniform systems.

3.2 Top-level Approach

To prove a safety property we must show that every reachable state is safe, that is, $[F] \subseteq [P]$ where $F$ is a state formula describing the set of reachable states and $P$ is a state formula describing safe states. As we know from Section 2.7 the normal form of a set of safe states is $P \equiv \bigwedge_{0 \leq I_1 < n} \cdots \bigwedge_{0 \leq I_k < n, I_k \neq I_1 \land \ldots \land I_k \neq I_{k-1}} p(I_1, X[I_1], \ldots, I_k, X[I_k])$ which is a conjunction with $C^n_k$ conjuncts $p(0, X[0], \ldots, k-1, X[k-1]) \land \cdots \land p(n-k, X[n-k], \ldots, n-1, X[n-1])$. Hence, in terms of state formulas we have to prove that $F \rightarrow P$ is valid, or, alternatively, that $F \rightarrow p(i_1, X[i_1], \ldots, i_k, X[i_k])$ is valid for any $i_1, \ldots, i_k \in I$.

Any conjunct $p(i_1, X[i_1], \ldots, i_k, X[i_k])$ can be interpreted as some property $p$ which holds for $k$ particular processes $i_1, \ldots, i_k$. If we parameterize the tuple of processes for which the property $p$ holds, then the safety property can be interpreted as the requirement that $p$ holds for all $C^n_k$ combinations (concrete values of the parameter $I_1, \ldots, I_k$) of $k$ processes in a system of size $n$.

In order to make the reachability analysis of a parameterized system computable, we will introduce over-approximation of the original transition relation (Section 3.3) parameterized by the choice of reference processes. Our idea is to consider every conjunct $p(i_1, X[i_1], \ldots, i_k, X[i_k])$ individually and treat processes $i_1, \ldots, i_k$ as parameters in our over-approximation schema. As the result we deal not with one but with $C^n_k$ over-approximations $S_{i_1, \ldots, i_k}$ of the set of reachable states, one for each conjunct $p(i_1, X[i_1], \ldots, i_k, X[i_k])$ where $i_1, \ldots, i_k \in I$. In other words, there are $k$ parameters $I_1, \ldots, I_k$ and $C^n_k$ different assignments, such that for each assignment $\{I_1 \mapsto i_1, \ldots, I_k \mapsto i_k\}$ we have

- an instance $p(i_1, X[i_1], \ldots, i_k, X[i_k])$ of the predicate $p(I_1, X[I_1], \ldots, I_k, X[I_k])$,
- an over-approximated transition relation $T_{i_1, i_2, \ldots, i_k}$ parameterized by the assignment (the algorithm will be presented later), and, accordingly,
CHAPTER 3. CONCEPTUAL APPROACH

Figure 3.1: An example of the relationship between conjuncts $P_1$ and $P_2$ describing property $P$ and over-approximations $S_1$ and $S_2$ of the set $S$ of reachable states.

- the over-approximated set $S_{i_1,\ldots,i_k}$ of reachable states defined by $T_{i_1,\ldots,i_k}$.

If the set of global states represented by each instance $p(i_1, X[i_1], \ldots, i_k, X[i_k])$ of the predicate $p(I_1, X[I_1], \ldots, I_k, X[I_k])$ includes the corresponding over-approximated set $S_{i_1,\ldots,i_k}$ then, obviously, it includes every reachable state as well. Consequently,

$$\bigwedge_{0 \leq I_1 < N} \ldots \bigwedge_{0 \leq I_k < N} \bigwedge_{I_1 \neq I_2 \land \ldots \land I_k \neq I_{k-1}} p(I_1, X[I_1], \ldots, I_k, X[I_k])$$

includes the set of all reachable states.

The idea is illustrated in Figure 3.1. For the sake of simplicity we assume that a safety property $P$ consists of only two conjuncts $P_1$ and $P_2$, therefore it is pictured in the figure as the intersection of ovals $P_1$ and $P_2$. For every conjunct $P_1$ and $P_2$ we have corresponding over-approximations $S_1$ and $S_2$ of the set $S$ of reachable states. (Correspondence between a predicate and an over-approximation is determined by the same value of the parameter.) If $S_1$ is included in $P_1$ and $S_2$ is included in $P_2$ then the set $S$ of reachable states is included in the property. In reality the number $C_k^n$ of over-approximations is unbounded because of $n$, but the encoding proposed in Chapters 4 and 5 enables computing the whole family of over-approximations at a time because of the parameterization.

In the rest of Section 3 we explain step-by-step how to construct over-approximations of the set of reachable states for different instances of the predicate obtained by assigning values to $I_1, \ldots, I_k$. Recall, that the presentation in the whole Section 3 is given for a single system of fixed size $n$, and the parameterization of the system size which allows to reason about infinite families of uniform systems will be done at a later stage.

3.3 Relaxed Transition Relations

An over-approximation in this work is achieved by using a relaxed transition relation instead of the original one. A relaxed transition relation $T'$ is a
modification of an original transition relation $T$, such that $T \subseteq T'$. Sometimes we refer to the transition relation $T$ of a system as to the *original* transition relation to distinguish it from its modified (relaxed) versions. If we change the transition relation $T$ of a system for its relaxed version $T'$ in the Definition 2.6.1 of the set of reachable states, we obtain a relaxed set of reachable states — an over-approximation of the set of reachable states which is the set of reachable states of the relaxed system.

**Proposition 3.3.1**. Any relaxed set $O$ of reachable states is an over-approximation of the set of reachable states $S$: $S \subseteq O$. \qed

We will achieve over-approximation by removing synchronization between processes. By synchronization between processes (on the intuitive level) we mean the situation when a process is allowed to make a local transition depending on the local states of some other processes. For example, in many realistic models a process synchronizes with at most one other process at a time, while in abstract models simultaneous synchronization with all other processes is often possible as well. In this section we formalize only how to relax the transition relation by removing synchronization between some components. The details when to apply relaxation and between which processes are discussed in Section 3.4.

Synchronized transitions serve as a base point for relaxation. Recall from Section 2.3, that a transition schema of a synchronized transition has the following normal form:

$$\bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \wedge J(I) \right].$$

If we get rid of $\bigwedge_{0 \leq G < n \land G \neq I}$ by de-sugaring then we obtain

$$\bigvee_{0 \leq I < n} \left[ (I \neq 0 \rightarrow \text{syn}(I, X[I], X'[I], 0, X[0])) \wedge \ldots \wedge (I \neq n - 1 \rightarrow \text{syn}(I, X[I], X'[I], n - 1, X[n - 1])) \wedge J(I) \right].$$

which can alternatively be presented as

$$\bigvee_{0 \leq I < n} \left[ \begin{array}{l}
syn(I, X[I], X'[I], 0, X[0]) \\
\wedge \ldots \wedge \\
syn(I, X[I], X'[I], I - 1, X[I - 1]) \\
\wedge \\
syn(I, X[I], X'[I], I + 1, X[I + 1]) \\
\wedge \ldots \wedge \\
syn(I, X[I], X'[I], n - 1, X[n - 1]) \\
\wedge J(I) \end{array} \right].$$
A formula in such a form expresses the most general case when the moving component synchronizes with every other component in the system. In particular, the predicate \( \text{syn}(I, X[I], X'[I], G, X[G]) \) describes local transitions of the moving component \( I \) which may be taken in synchronization with the local state of component \( G \). Namely, if \( \text{syn}(i, x_i, x'_i, g, x_g) \) holds then it expresses the condition that the moving component \( i \) can make a transition from \( \{ X[i] \mapsto x_i \} \) into \( \{ X[i] \mapsto x'_i \} \) whenever component with index \( g \) is in local state \( \{ X[g] \mapsto x_g \} \). Relaxation proposed here is based on ignoring all the conjuncts \( \text{syn}(I, X[I], X'[I], g, X[g]) \) except of those where the index of the guarding component belongs to a chosen subset \( \{ g_1, \ldots, g_k \} \subseteq \{ g \mid 0 \leq g < n \land g \neq I \} \) so that the transition formula will look like the following:

\[
\bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \right] \wedge \mathcal{J}(I)
\]

Formally, transformation of a synchronized transition \( \tau = [\rho] \) expressed in the normal form as

\[
\bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \right] \wedge \mathcal{J}(I)
\]

to its relaxed version \( \tau|_\epsilon \), results in a meta-transition with the following transition schema

\[
\bigvee_{0 \leq I < n} \left[ \bigwedge_{G=\epsilon} \text{syn}(I, X[I], X'[I], G, X[G]) \right] \wedge \mathcal{J}(I)
\]

where the set of components \( \epsilon = \{ g_1, \ldots, g_k \} \) is the parameter of the relaxation. Alternatively, we may write \( \bigwedge_{G \in \{ g_1, \ldots, g_k \}} \) instead of \( \bigwedge_{G=g_1 \lor \ldots \lor G=g_k} \).

The consequence of transforming a number of synchronized transitions to their relaxed versions is a number of additional atomic transitions (since \( [\tau] \subseteq [\tau|_\epsilon] \)), which (potentially) will lead to additional states during reachability analysis, therefore we get an over-approximated set of reachable states as a result (Proposition 3.3.1).

### 3.4 Environmental Relaxation

Quantification of all the system variables (except those describing the moving component) in a synchronized transition and the replacement of all synchronized transitions with their relaxed versions seem to be to no avail because the resulting system will probably lose its important properties.
Therefore we will use an approach of relaxing synchronized transitions conditionally, based on the indexes of the moving component and any other component.

Recall from Section 3.2 that we decided to compute an over-approximation of the set of reachable states for every assignment \( \{ I_1 \mapsto i_1, \ldots, I_k \mapsto i_k \} \) to \( k \) parameters \( I_1, \ldots, I_k \) ranging over the set \( I \) of all processes in the system. The same values instantiate a partition of the property, in which we will check the inclusion of the obtained over-approximation. Therefore, it is reasonable that the processes, whose indexes are chosen for the assignment, should be treated as first-class citizens by the respective over-approximating reachability algorithm. We mark the set \( \epsilon = \{ i_1, \ldots, i_k \} \) of such processes as being reference processes and all other as environment processes (sometimes denoted as \( \bar{\epsilon} \)) and in what follows we show how to relax synchronized transitions based on that division.

Indeed, when an algorithm designer performs algorithm analysis by considering a few reference processes and the rest, the reference processes are paid much more attention than the others. In particular, reference processes follow all the algorithmic steps precisely, while we do not care about all the details about mutual relationship between processes in the environment. This is the main idea which we formalize in this section.

Let us partition a synchronized transition based on the type of the moving component. For that we de-sugar the \( \bigvee_{0 \leq I < n} \) construct in the transition’s normal form:

\[
\bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{sym}(I, X[I], X'[I], G, X[G]) \land J(I) \right] = \\
\bigwedge_{0 \leq G < n \land G \neq 0} \text{sym}(0, X[0], X'[0], G, X[G]) \land J(0) \\
\bigvee \ldots \bigvee \\
\bigwedge_{0 \leq G < n \land G \neq n-1} \text{sym}(n-1, X[n-1], X'[n-1], G, X[G]) \land J(n-1)
\]

and consider partitions represented by disjuncts separately. Since reference processes are first class citizens and are very important for us, they will always synchronize with any other process in the system. Therefore, the partitions where the moving component is a reference process will remain untouched.

The environment processes are not so interesting for us, therefore we will neglect any dependency between them. That is, if the disjunct

\[
\bigwedge_{0 \leq G < n \land G \neq i} \text{sym}(i, X[i], X'[i], G, X[G]) \land J(i)
\]
describes a partition with the moving environment component \( i \), we will remove any synchronization between it and any other environment component in the way presented in Section 3.3. Given a set \( \epsilon \) of reference processes the disjunct above will look as following after environmental relaxation:

\[
\bigwedge_{G \in \epsilon} \text{syn}(i, X[i], X'[i], G, X[G]) \land J(i).
\]

After gathering together such conjuncts for all environment processes \( \bar{\epsilon} \) we get the following formula:

\[
\bigvee_{I \in \bar{\epsilon}} \bigwedge_{G \in \epsilon} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I).
\]

The part of the synchronized transition where the moving components belong to the reference set is untouched and can be expressed as transition schema

\[
\bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right].
\]

Let us now summarize what we have discussed about the relaxation of synchronized transitions. The relaxed version of a synchronized transition \( \tau \) described by the transition schema \( \rho \)

\[
\bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right] 
\]

in the normal form w.r.t. a set \( \epsilon \) of reference processes is a meta-transition \( \tau|_{\epsilon} \) which can be described by the transition schema \( \rho|_{\epsilon} \)

\[
\bigvee_{I \in \epsilon} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right] 
\]

\[
\bigvee_{I \in \epsilon} \left[ \bigwedge_{G \in \epsilon} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right].
\]

The first part of the disjunction represents all the partitions with reference moving components, while the second one — all the partitions with environment moving components. As explained in Section 3.3, the result of such a transformation will be an over-approximated set of atomic transitions.

So far we considered relaxation based on a fixed set of reference processes. As will be shown later the choice of reference processes depends on a safety property. However, safety properties usually concern a fixed size combination of arbitrary processes, for example:

\textit{no two processes can be in the critical section at the same time.}
Therefore, we introduce a parameterization of processes composing the reference set by using a set $I_R = \{I_1, \ldots, I_k\}$ of $k$ reference parameters. Furthermore, we need a function $\text{range}(I_R)$ to restrict all possible reference sets only to those we are interested in:

$$\mathcal{E} = \{i_1, \ldots, i_k \mid (i_1, \ldots, i_k) \in [\text{range}(I_R)]\}.$$ 

In particular, to model $k$ arbitrary reference processes we can define $\text{range}(I_R)$ as $0 \leq I_1, \ldots, I_k < n \land \text{distinct}(I_1, \ldots, I_k)$. In what follows we assume this function to be a default one whenever not otherwise specified.

We define the environmental relaxation $\omega$ as the tuple $(k, I_R, \text{range}(I_R), \mathcal{E})$ where

- $k \geq 2$ is the rank of the environmental relaxation,
- $I_R$ is a set $\{I_1, \ldots, I_k\}$ of $k$ reference parameters,
- $\text{range}(I_R)$ is some function on $I_R$, and
- $\mathcal{E}$ is a set of sets of reference processes defined by $\text{range}(I_R)$.

The application of an environmental relaxation $\omega = (k, I_R, \text{range}(I_R), \mathcal{E})$ on a synchronized transition $\tau$ is denoted $\omega(\tau)$ and defined as the family $\{\tau_{\epsilon} \mid \epsilon \in E\}$ of relaxed transitions $\tau_{\epsilon}$, such that $\tau_{\epsilon}$ is the synchronized transition $\tau$ relaxed w.r.t. $\epsilon \in \mathcal{E}$. Note that the unsynchronized transitions may be considered as already relaxed to the greatest possible extent because the moving components make the transition regardless of the state of other components. Therefore, we may write $\omega(\tau) = \tau$ if the meta-transition $\tau$ is the unsynchronized one.

### 3.5 Summary of the approach

Let us summarize the approach presented in this section. If we have a property expressed in the normal form as

$$P \equiv \bigwedge_{0 \leq i_1 < N} \ldots \bigwedge_{0 \leq i_k < N \land i_k \neq i_1 \land \ldots \land i_k \neq i_{k-1}} p(I_1, \ldots, I_k, X[I_1], \ldots, X[I_k])$$

then we perform the environmental relaxation of rank $k$ of the transition relation $\mathcal{T}$ to obtain the family $\{\mathcal{T}_{\epsilon} \mid \epsilon \in \mathcal{E}\}$ of $C^n_k$ over-approximated transition relations where $\mathcal{E}$ is a set of all possible combinations of $k$ processes in the system. The property $P$ is proved if every over-approximated set $S_\epsilon$ of reachable states defined by the relaxed transition relation $\mathcal{T}_{\epsilon}$ from the family $\{\mathcal{T}_{\epsilon} \mid \epsilon \in \mathcal{E}\}$ is included in the set of states described by the corresponding partition $p(i_1, \ldots, i_k, X[i_1], \ldots, X[i_k])$ of the property where $\epsilon = \{i_1, \ldots, i_k\}$.

The advantage of this approach is that with the encoding provided in Chapters 4 and 5 it is possible to compute a family of $C^n_k$ over-approximations even when $n$ is unbounded such that each over-approximation is tailored for a particular partition of the property.
Chapter 4

Intermediate Representation

In this chapter we introduce disjunctive relations - a new intermediate representation of sets of global states. The formalism presented in Sections 2.2-2.4 in general is not suitable for computer encoding and automatic verification of infinite families of systems because the number of variables in state and transition formulas is proportional to the number of processes in a system, therefore we cannot reason directly about families of systems where the number of processes is unbounded.

The main advantage of disjunctive relations is that they may represent a set of globals states and have a finite encoding as Presburger formulas even for a system consisting of an unbounded number of processes. Next, parameterization of disjunctive relations enables finite representation of sets of global states even for infinite families of systems (although it is not possible for all cases).

Besides just representing sets of global states we will show how to perform the usual set operations – intersection, union, inclusion and emptiness test of sets of global states, operating on the correspondent disjunctive relations. More than that, it is also possible to perform application of synchronized and unsynchronized transitions with environmental relaxation applied (which was introduced in Section 3.4). Again, parameterization of disjunctive relations allows to perform all the discussed operations on all instances of a family of sets of global states at the same time. Therefore, we have all the range of operations required to perform the reachability analysis of (possibly infinite) families of uniform state transition systems and proving safety properties automatically (although the termination is not guaranteed for the general case which makes the approach incomplete).

Note that disjunctive sets are not closed under some set operations, like the union of two sets. Therefore, we provide methods which do not guarantee that the result will be exact but they are applicable for the general case in
return. The application of such methods results in an additional source of over-approximations besides the one described in Chapter 3. Nevertheless, the use of over-approximating techniques on the level of the intermediate representation is very often crucial for the performance and/or termination.

The plan of this chapter is as follows. First, in Section 4.1 we propose an encoding scheme to represent certain sets of global states (called disjunctive sets) of a uniform state transition system as a special class of relations, called disjunctive relations. Section 4.2 introduces an index-free disjunctive relation, which is a simplified version of the disjunctive relation and can be used in some systems providing for more compact representation.

Section 4.3 introduces parameterization as a technique which allows to represent any set of global states as a collection of disjunctive relations, while Section 4.4 introduces a disjunctive meta-relation whose purpose is to represent families of sets of global states for different values of some parameters such as the system size. Finally, in Section 4.5, we join all the techniques we have discussed in Sections 4.1-4.4 into the definition of an environment disjunctive relation which is a representation of a family of sets of global states specially suited for the reachability analysis with environmental relaxation.

Sections 4.6-4.10 discuss set operations of global sets of states performed on the intermediate representation, while Section 4.11 covers the issue of the parallel application of synchronized and unsynchronized transitions to the sets of states represented by disjunctive relations. Although our final target is the operations on environment disjunctive relations, we often start discussing each operation on more simple representations, like disjunctive relations, to explain complex issues gradually to the reader.

Afterwards, in Chapter 5, we show how disjunctive relations and operations on them can be mapped to the domain of Presburger formulas with standard operations.

### 4.1 Disjunctive Relations and Disjunctive Sets

A disjunctive relation (DR) w.r.t. a uniform state transition system of size $n$ is a binary relation $R \subseteq \mathbb{I} \times A$ on the set $\mathbb{I} = \{0, \ldots, n-1\}$ of $n$ process indexes and the set $A$ of local state values of a component (since any considered system is uniform, we can afford to talk about a component). Note, that the set $A$ is system specific, and, in particular, may also depend on $n$ or be infinite. The semantics of a disjunctive relation $R$ is the set $[R]$ of all global states $\{X[0] \rightarrow a_0, \ldots, X[n-1] \rightarrow a_{n-1}\}$ such that $\{(0, a_0), \ldots, (n-1, a_{n-1})\} \subseteq R$. The set of global states which can be represented by a disjunctive relation is called a disjunctive set.

We will visualize a disjunctive relation on a two-dimensional graph, like the one in Figure 4.1, where process indexes are on the X-axis and the domain of uniform registers is on the Y-axis. Accordingly, the point $(i, x)$ is drawn iff $R(i, x)$. A global state can be pictured as a line which starts from some point with index 0, goes through exactly one point with every
consequent index and finishes at a point with index \( n - 1 \). In Figure 4.1 the line connecting the points \((0,1), (1,1), (2,0), (3,3)\) represents the global state \( \{X[0] \mapsto 1, X[1] \mapsto 1, X[2] \mapsto 0, X[3] \mapsto 3\} \). Hence, the set of global states represented by a disjunctive relation can be visualized by all the lines which can be drawn that way.

**Example 4.1.1** In a system of size \( n = 4 \) the set consisting of the following global states

\[
s_1 = \{X[0] \mapsto 1, X[1] \mapsto 0, X[2] \mapsto 0, X[3] \mapsto 3\}
\]

\[
s_2 = \{X[0] \mapsto 1, X[1] \mapsto 1, X[2] \mapsto 0, X[3] \mapsto 3\}
\]

\[
s_3 = \{X[0] \mapsto 1, X[1] \mapsto 0, X[2] \mapsto 2, X[3] \mapsto 3\}
\]

\[
s_4 = \{X[0] \mapsto 1, X[1] \mapsto 1, X[2] \mapsto 2, X[3] \mapsto 3\}
\]

is represented by the disjunctive relation

\[
R = \{(0,1), (1,0), (1,1), (2,0), (2,2), (3,3)\}.
\]

A visualization of \( R \) is illustrated in Figure 4.1.

For the rest of the thesis we introduce the following duality of the notation \( R(I, X) \):

- \( R(I, X) \) means \( (I, X) \in R \), or

- \( R(I, X) \) means the relation \( R \) itself. The reason to prefer notation \( R(I, X) \) over \( R \) is that we wish lay emphasis on the signature of the relation. For example, after introducing parameters to disjunctive relations, writing \( R_1(I, X, G_1) \) and \( R_2(I, X, G_2) \) means that although \( R_1 \) and \( R_2 \) have the same arity, the type of the third position is different.
Figure 4.2: A visualization of the disjunctive relation from Example 4.1.2. The disjunctive relation is non-filtered because it is non-empty and there are no points for indexes $i = 0$ and $i = 1$.

It should be clear from the context which meaning of the notation is suitable for a given situation.

A disjunctive relation $R(I, X)$ is called filtered iff

$$\forall I \in I \exists X \in A R(I, X)$$

or

$$\forall I, X \neg R(I, X).$$

In other words, a filtered disjunctive relation must be empty whenever it represents the empty set of global states. Otherwise, for every index there must be at least one point belonging to the relation.

**Example 4.1.2** The relation $R(I, X) = \{(2, 0), (3, 1)\}$ represents the empty set of global states in a system of size $n = 4$ because there does not exist a subset $\{(0, a_0), \ldots, (3, a_3)\} \subseteq R(I, X)$ for some $a_0, \ldots, a_3$. However, this relation is not filtered because it is non-empty. A visualization is illustrated in Figure 4.2.

The requirement of being filtered simplifies the definition of certain operations on disjunctive relations, like the inclusion test introduced later in this chapter.

### 4.2 Indexes-Free Disjunctive Relation

Sometimes there are situations where the set of initial states and the transition relation can be expressed without referring to the process index. Accordingly, there is no point of keeping track of process indexes. Then we can
CHAPTER 4. INTERMEDIATE REPRESENTATION

Figure 4.3: A visualization of the disjunctive relation $R(I,X)$ from Example 4.2.1 which can be presented as an indexes-free disjunctive relation $R'(X)$.

use an index-free disjunctive relation (IFDR) $R \subseteq A$ w.r.t. a state transition system of size $n$ and a set of indexes $I$, where $A$ is a(n indexes-free) uniform domain of a component. The semantics of an indexes-free disjunctive relation $R(X)$ is a set $[R(X)]$ of global states $\{X[0] \mapsto a_0, \ldots, X[n-1] \mapsto a_{n-1}\}$ such that $R(a_i)$ for all $0 \leq i < n$.

Example 4.2.1 The disjunctive relation $R(I,X) = \{(0,0), (0,2), (1,0), (1,2), (2,0), (2,2), (3,0), (3,2)\}$ in a system of size $n = 4$ describes the situation, when a value contained in register $X[i]$ is equal 0 or 2 independently of the index $i$ of the component. Such a set of global states can be presented with an indexes-free disjunctive relation $R'(X) = \{0,2\}$. A visualization of $R(I,X)$ and $R'(X)$ is illustrated in Figure 4.3.

The application of IFDRs instead of DRs tremendously simplifies calculations. See Appendix A for an example.

4.3 Parameterized Disjunctive Relations

Not all sets of global states can be represented as a disjunctive relation, which is illustrated by the following example.

Example 4.3.1 Let us show that the set $S = \{\{X[0] \mapsto 0, X[1] \mapsto 1\}$, $\{X[0] \mapsto 1, X[1] \mapsto 0\}\}$ in a system of size $n = 2$ is not disjunctive. Assume that it is disjunctive so there exists a disjunctive relation $R(I,X)$ such that

- $[R(I,X)] = S$, and
- $\{(0,0), (1,1)\} \subseteq R(I,X)$ because $\{X[0] \mapsto 0, X[1] \mapsto 1\} \in S$, and
Chapter 4. Intermediate Representation

- \{0, 1\}, \{(0, 1), (1, 0)\} \subseteq R(I, X) because \{X[0] \mapsto 1, X[1] \mapsto 0\} \subseteq S.

Since \{(0, 1), (1, 1)\} \subseteq R(I, X) then, by the semantics of the disjunctive relation, \{X[0] \mapsto 1, X[1] \mapsto 1\} \in [R(I, X)]. This means that [R(I, X)] \neq S because \{X[0] \mapsto 1, X[1] \mapsto 1\} \not\subseteq S. We came to contradiction, hence our assumption is wrong and S is not disjunctive.

However, any set S of global states of a finite uniform system can be described (non-uniquely) by a collection of disjunctive relations \{R_1, \ldots, R_k\} such that

\[ S = \bigcup_{i=1}^{k} [R_i] \]

A parameterized disjunctive relation (pDR) \( R \subseteq \mathbb{I} \times A \times G_1 \times \ldots \times G_k \) is a disjunctive relation extended with a set of parameters \( G_1, \ldots, G_k \) ranging over respective domains \( G_1, \ldots, G_k \). Such parameters will be called collection parameters. Often, for the sake of brevity, we will consider parameterized disjunctive relations \( R \subseteq \mathbb{I} \times A \times G \) with only one collection parameter \( G \), because, in general, a collection parameter can be composite and consist of any (finite) vector of parameters \( (G_1, \ldots, G_k) \):

\[ R \subseteq \mathbb{I} \times A \times (G_1 \times \ldots \times G_k). \]

The semantics of a disjunctive relation \( R(I, X, G) \) parameterized by a collection parameter \( G \) is the set \( [R(I, X, G)] \) of all global states such that

\[ [R(I, X, G)] = \bigcup_{g \in G} [R_g(I, X)] \text{ where } R_g(I, X, g). \]

We will use pDRs to represent any set of global states of any uniform system.

Example 4.3.2 In a system of size \( n = 4 \) the set consisting of the following global states

\[ s_1 = \{X[0] \mapsto 1, X[1] \mapsto 0, X[2] \mapsto 0, X[3] \mapsto 4\} \]
\[ s_2 = \{X[0] \mapsto 1, X[1] \mapsto 0, X[2] \mapsto 2, X[3] \mapsto 4\} \]
\[ s_3 = \{X[0] \mapsto 1, X[1] \mapsto 0, X[2] \mapsto 3, X[3] \mapsto 4\} \]
\[ s_4 = \{X[0] \mapsto 4, X[1] \mapsto 1, X[2] \mapsto 3, X[3] \mapsto 4\} \]

is not disjunctive. However, \( \{s_1, s_2, s_3\} \) is disjunctive as well as \( \{s_1\} \), therefore the set \( \{s_1, s_2, s_3, s_4\} \) can be thought of as a union \( \{s_1, s_2, s_3\} \cup \{s_4\} \) of two disjunctive sets and be represented by the collection of disjunctive relations

\[ \{\{(0, 1), (1, 0), (2, 0), (2, 2), (2, 3), (3, 4)\}, \{(0, 4), (1, 1), (2, 3), (3, 4)\}\}. \]

A visualization of these two disjunctive relations is presented in Figure 4.4. This collection can be presented as one relation parameterized by a collection parameter with two different values, for example, 1 and 2: \( R(I, X, G) = \{(0, 1, 1), (1, 0, 1), (2, 0, 1), (2, 2, 1), (2, 3, 1), (3, 4, 1), (0, 4, 2), (1, 1, 2), (2, 3, 2), (3, 4, 2)\} \). The parameterized disjunctive relation is visualized in Figure 4.5.

\[ \square \]
A parameterized disjunctive relation $R(I, X, G)$ can be thought of as a collection of disjunctive relations $\{R_{g_1}(I, X), \ldots, R_{g_k}(I, X)\}$ for different values of the parameter $G$, where $R_g = \{(I, X)|R(I, X, g)\}$ and $g \in G$. A pDR $R(I, X, G)$ is called filtered iff every disjunctive relation $R_g = \{(I, X)|R(I, X, g)\}$, such that $g \in G$, is filtered.

### 4.4 Encoding Families of Systems

In this section we present techniques how to represent and reason about possibly infinite families of uniform systems in terms of the intermediate representation.

#### Size Parameter

Families of uniform systems as they are introduced at the end of Section 2.4 are often defined with the size parameter which determines the number of components in each system of the family. To reason about such a family we have to be able to keep track of sets $S_1, S_2, \ldots, S_n$ of global states for each member of the family where $n$ means the number of components in the system and can be unbounded. We denote such a family of sets of global states as $\{S_n\}_{n \in M}$ or even shorter as $\{S_n\}$ under implicit assumption that $n$ ranges over $M$, where $M$ is a subset of the positive integers.

When we represent a set of states for each system in the family as a pDR, then we have to consider a family $\{R_n\}_{n \in M}$ of parameterized disjunctive relations to represent sets of states for the whole family.
A disjunctive meta-relation (DMR) with size parameter is a relation $R \subseteq \mathbb{N} \times A \times G \times \mathbb{N}$ where the first three dimensions are the same as for the parameterized disjunctive relation — process index, local state and collection parameter, while the new last dimension is used for the system size. Since dimensions $A$ and $G$ can be composite, a DMR in its most general form is a relation

$$R \subseteq \mathbb{N} \times \underbrace{\mathbb{N} \times \mathbb{N} \times \ldots \times \mathbb{N}}_{A} \times \underbrace{\mathbb{N} \times \ldots \times \mathbb{N}}_{G} \times \mathbb{N}$$

(with a fixed arity, obviously).

The semantics of a possibly infinite disjunctive meta-relation $R(I,X,G,N)$ with size parameter $N$ and collection parameter $G$ is the family $[[R(I,X,G,N)]] = \{S_n\}_{n \in \mathbb{M}}$ of sets of global states such that

$$S_n = \left\{(I,X,G) \mid R(I,X,G,n) \land n \in \mathbb{M} \land 0 \leq I < n \land X \in A_n \land G \in G_n \right\}$$

where $A_n$ and $G_n$ are the domains of the uniform register and the collection parameter, respectively, corresponding to the system of size $n$. That is, a disjunctive meta-relation with instantiated size parameter becomes a parameterized disjunctive relation with its usual semantics.
CHAPTEER 4. INTERMEDIATE REPRESENTATION

Example 4.4.1  In Example 2.4.2 we presented a parameterized state schema $F(N)$

$$N \geq 2 \land L[0] = 1 \land \bigwedge_{1 \leq i < N} L[i] = 0$$

which describes the following family of sets of global states for each system of size $N \geq 2$:

$$\{ L[0] \mapsto 1, L[1] \mapsto 0 \}$$

$$\{ L[0] \mapsto 1, L[1] \mapsto 0, L[2] \mapsto 0 \}$$

$$\{ L[0] \mapsto 1, L[1] \mapsto 0, L[2] \mapsto 0, L[3] \mapsto 0 \}$$

$$\ldots$$

That family of sets of global states can be represented by a disjunctive meta-relation $R(I, X, N)$ with the size parameter $N$:

$$R(I, X, N) = \begin{cases} 
(0, 1, 2), (1, 0, 2), \\
(0, 1, 3), (1, 0, 3), (2, 0, 3), \\
(0, 1, 4), \ldots 
\end{cases}$$

Each set in the family is disjunctive, therefore we can omit the collection parameter.

The semantics of $R(I, X, N)$ is the following infinite set of disjunctive relations:

$$\begin{cases} 
\{(0, 1), (1, 0)\}, \\
\{(0, 1), (1, 0), (2, 0)\}, \\
\{(0, 1), (1, 0), (2, 0), (3, 0)\}, \\
\ldots 
\end{cases}$$

The DMR $R(I, X, N)$ is visualized in Figure 4.6. □
Family Parameter

It is possible that a family of systems introduced in Section 2.4 has other parameters than the number of components in a system. For example, we may want to describe a family of uniform systems consisting of \( n \) processes but with different sets of initial states. To reason about such a family of systems we have to be able to keep track of a family \( \{ S_p \}_{p \in P} \) of sets of global states for each value \( p \) of the parameter where \( P \) is the domain of the parameter \( p \). Sometimes we will write \( \{ S_p \} \) instead of \( \{ S_p \}_{p \in P} \) when the domain \( P \) is clear from the context.

When a set of global states for one system is represented as a pDR, then in order to represent a family of sets of global states for every value \( p \) of the parameter we have to consider a family \( \{ R_p \}_{p \in P} \) of parameterized disjunctive relations. A family parameter \( P \) is used to represent a family \( \{ R_p \}_{p \in P} \) of parameterized disjunctive relations \( R_p \subseteq I \times A \times G \) with collection parameter \( G \) as one disjunctive meta-relation (DMR) \( R \subseteq I \times A \times G \times P \) with family parameter \( P \) (and collection parameter \( G \)).

The semantics of a disjunctive meta-relation \( R(I, X, G, P) \) with family parameter \( P \) and collection parameter \( G \) is the family \( [R(I, X, G, P)] = \{ S_p \}_{p \in P} \) of sets of global states such that

\[
S_p = \{ (I, X) | R(I, X, G, p) \land p \in P \}.
\]

That is, a disjunctive meta-relation with instantiated family parameter becomes a parameterized disjunctive relation with its usual semantics.

**Example 4.4.2** Assume, that there is a family of systems of size \( n = 3 \) whose sets of global states can be described by the following parameterized state schema \( F(P) \):

\[
0 \leq P < 3 \land L[P] = 1 \land \bigwedge_{0 \leq I \leq 2 \land I \neq P} L[I] = 0
\]

That is, we have a family of three systems and there are different sets of global states described by \( F(0) \), \( F(1) \) and \( F(2) \) for each system in the family depending on values of the parameter \( P \). Such a family \( \{ S_p \}_{p \in \{0, 1, 2\}} \) of sets of global states can be represented by the following disjunctive meta-relation \( R(I, X, P) \) with family parameter \( P \):

\[
\begin{align*}
\{ (0, 1, 0), (1, 0, 0), (2, 0, 0), \\
(0, 0, 1), (1, 1, 1), (2, 0, 1), \\
(0, 0, 2), (1, 0, 2), (2, 1, 2)
\}
\end{align*}
\]

The semantics of \( R(I, X, P) \) is the following collection of disjunctive relations:

\[
\begin{align*}
\{ \{(0, 1), (1, 0), (2, 0)\} & , \{(0, 0), (1, 1), (2, 0)\} , \{(0, 0), (1, 0), (2, 1)\} \}
\end{align*}
\]

\( \square \)
CHAPTER 4. INTERMEDIATE REPRESENTATION

Size and Family Parameters Together

Parameterized formulas introduced in Section 2.4 can involve both size and family parameters. That is, it might be necessary to represent a family \( \{ S_{n,p} \}_{n \in M, p \in P} \) of sets of global states parameterized by both the system size \( n \) and a family parameter \( p \). To represent a family \( \{ R_{n,p} \}_{n \in M, p \in P} \) of pDRs we will use disjunctive meta-relation (DMR) \( R \subseteq N \times A \times G \times N \times P \) with size and family parameters with dimensions for both size and family parameter. Since dimensions \( A, G \) and \( P \) can be composite, a DMR with size and family parameters is generally a relation

\[
R \subseteq N \times N \times \ldots \times N \times A \times N \times \ldots \times N \times G \times \ldots \times N
\]

(with a fixed arity, obviously). For the sake of brevity, we will write simply \( \{ S_{n,p} \} \) or \( \{ R_{n,p} \} \) when the domains of parameters are clear from the context and omit mentioning types of parameters of a DMR if they are obvious.

The semantics of a DMR \( R(I, X, G, N, P) \) with size parameter \( N \) and family parameter \( P \) is the family \( \{ R(I, X, G, N, P) \} = \{ S_{n,p} \} \) of sets of global states such that

\[
S_{n,p} = \left\{ (I, X, G) \mid R(I, X, G, n, p) \land n \in M \land p \in P_n \land 0 \leq I < n \land X \in A_n \land G \in G_n \right\}
\]

where \( A_n, G_n \) and \( P_n \) are the domains of the uniform register, the collection parameter and the family parameter, respectively, corresponding to the system of size \( n \).

Example 4.4.3 Let us extend Examples 4.4.2 and 4.4.1 so that we will have a family of systems, whose sets of global states can be described by the following parameterized state schema \( F(N, P) \):

\[
N \geq 2 \land 0 \leq P < N \land L[P] = 1 \land \bigwedge_{0 \leq I < N \land I \neq P} L[I] = 0
\]

That is, we have an infinite family of uniform systems with different sizes \( N \geq 2 \) and there are different sets of global states for different values of parameter \( P \). Such a family of sets of global states can be represented by a disjunctive meta-relation with size and a family parameters \( N \) and \( P \):

\[
R(I, X, N, P) = \left\{ (0, 1, 2, 0), (1, 0, 2, 0), (0, 0, 2, 1), (1, 1, 2, 1), (0, 1, 3, 0), (1, 0, 3, 0), (2, 0, 3, 0), (0, 0, 3, 1), (1, 1, 3, 1), (2, 0, 3, 1), (0, 0, 3, 2), (1, 0, 3, 2), (2, 1, 3, 2), (0, 1, 4, 0), (1, 0, 4, 0), (2, 0, 4, 0), (3, 0, 4, 0), \ldots \right\}
\]

43
The semantics of $R$ is the following collection of disjunctive relations:

$$\begin{align*}
\{& (0, 1), (1, 0) \\ & (0, 0), (1, 1) \\ & (0, 1), (1, 0), (2, 0) \\ & (0, 0), (1, 1), (2, 0) \\ & (0, 0), (1, 0), (2, 1) \\ & \ldots \end{align*}$$

4.5 Environment Disjunctive Relation

In Section 3.5 we set a goal to compute $C_n^k$ over-approximations for each combinations of $k$ processes marked as reference in a system of size $n$. That is, according to Section 4.4, we require $k$ family parameters $I_1, \ldots, I_k$ to represent a family $\{S_{i_1, \ldots, i_k}\}_{i_1, \ldots, i_k \in I}$ of disjunctive sets of global states as a disjunctive-meta relation. In this section we introduce a new most general representation for families of sets of global states which will be used in the remaining of the thesis in the reachability analysis with environmental relaxation.

An environment disjunctive relation (EDR) $R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \subseteq \mathbb{N} \times A \times \ldots \times \mathbb{N} \times A \times \mathbb{N} \times G$ for $k \geq 1$ is a disjunctive meta-relation where

- $I_1, \ldots, I_k$ are family parameters representing reference components,
- $X_1, \ldots, X_k$ are collection parameters representing local states of reference components (parameters $X_1, \ldots, X_k$ are sometimes called reference collection parameters to distinct them from optional collection parameter $G$),
- $I$ and $X$ represent the index and the local state of environment components,
- $N$ is the size parameter, and
- $G$ is an optional collection parameter.

The variables $I_1, X_1, \ldots, I_k, X_k$ are also called reference parameters or reference variables because they describe reference processes.

The semantics of an environment disjunctive relation $R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ is the family

$$[R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] = \{S_{i_1, \ldots, i_k, n}\}_{n \in \mathbb{M}, 0 \leq i_1, \ldots, i_k < n}$$

of sets $S_{i_1, \ldots, i_k, n}$ of global states $\{X[i_1] \mapsto a_1, \ldots, X[i_k] \mapsto a_k, X[j_1] \mapsto b_1, \ldots, X[j_m] \mapsto b_m\}$ such that
CHAPTER 4. INTERMEDIATE REPRESENTATION

Figure 4.7: A visualization of the environment disjunctive relation $R(I_1, X_1, I_2, X_2, I, X)$ from Example 4.5.1 as a collection of two disjunctive relations.

- $n \in \mathbb{M}$ is a system size,
- $\{i_1, \ldots, i_k\} \subseteq \mathbb{I}$ is the reference set,
- $\{j_1, \ldots, j_m\} \subseteq \mathbb{I}$ where $k + m = n$ is the set of environment processes disjoint from the reference set, and
- $\{(i_1, a_1, \ldots, i_k, a_k, j_1, b_1, n, g), \ldots, (i_1, a_1, \ldots, i_k, a_k, j_m, b_m, n, g)\} \subseteq R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ for some $g \in G_n$ where $a_1, \ldots, a_k$, $b_1, \ldots, b_m \in A_n$.

The intuition behind the environment disjunctive relation is that we represent all the information about reference processes by the means of the respective reference variables, whereas the information about the states of all environment processes is represented as a respective disjunctive relation (or a collection of disjunctive relations whenever parameter $G$ is used) for the given values of reference parameters.

Example 4.5.1 Assume that we have a system of size $n = 5$ where processes with indexes $I_1 = 1$ and $I_2 = 3$ are chosen to serve as reference processes. Assume further, that we wish to describe a set of states where

- the reference process with index 1 is in state $\{X \mapsto 2\}$,
- the reference process with index 3 is in state $\{X \mapsto 2\}$ or $\{X \mapsto 3\}$, and
- the value of the local state of any environment process can be more than or equal to zero, but not more than the value of the state of the reference process with index 3.
Such a set of states can be described by the following environment disjunctive relation $R(I_1, X_1, I_2, X_2, I, X)$:

$$
\begin{align*}
(1, 2, 3, 2, 0, 0), & (1, 2, 3, 2, 2, 0), (1, 2, 3, 2, 4, 0), \\
(1, 2, 3, 2, 0, 1), & (1, 2, 3, 2, 2, 1), (1, 2, 3, 2, 4, 1), \\
(1, 2, 3, 2, 0, 2), & (1, 2, 3, 2, 2, 2), (1, 2, 3, 2, 4, 2), \\
(1, 2, 3, 3, 0, 0), & (1, 2, 3, 3, 2, 0), (1, 2, 3, 3, 4, 0), \\
(1, 2, 3, 3, 0, 1), & (1, 2, 3, 3, 2, 1), (1, 2, 3, 3, 4, 1), \\
(1, 2, 3, 3, 0, 2), & (1, 2, 3, 3, 2, 2), (1, 2, 3, 3, 4, 2), \\
(1, 2, 3, 3, 0, 3), & (1, 2, 3, 3, 2, 3), (1, 2, 3, 3, 4, 3)
\end{align*}
$$

The relation $R(I_1, X_1, I_2, X_2, I, X)$ is visualized in Figure 4.7.

We called parameters $X_1, \ldots, X_k$ reference collection parameters. Indeed, if we look at Figure 4.7 visualizing the EDR from Example 4.5.1 we can notice that it is represented as a collection of two disjunctive relations corresponding to two different instantiation of the reference collection parameters: \{\(X_1 \mapsto 2, X_2 \mapsto 2\)\} and \{\(X_1 \mapsto 2, X_2 \mapsto 3\)\}. According to Section 4.3 a parameterized disjunctive relation with a collection parameter can be considered as a collection of disjunctive relations with instantiated parameter. In EDR the local states of reference processes play the role of the collection parameter. That is why in each chart there is exactly one point corresponding to the local state of each reference process from the reference set \{1, 3\} (points (1, 2) and (3, 2) in the left chart and points (1, 3) and (3, 2) in the right chart). The values of family parameters $I_1$ and $I_2$ are constant in the EDR, therefore, there is only one system in the family.

As we will see later in Chapter 5, even an infinite environment disjunctive relation may have finite encoding with Presburger formulas for many important problems.

**Example 4.5.2** The set of initial states of the Simplified Bakery Algorithm (see Appendix A for details) consists of one global state where the values of local registers $Q$ and $L$ of all processes are set to zero. The family of systems parameterized by size $N \geq 3$ and the reference set ranging over all pairs of processes can be represented by the following infinite environment disjunctive relation $R(I_1, Q_1, I_2, Q_2, I_2, I, Q, L, N)$:

$$
\begin{align*}
(0, 0, 0, 1, 0, 0, 2, 0, 3), & (0, 0, 0, 2, 0, 0, 1, 0, 3), \\
(1, 0, 0, 0, 0, 0, 2, 0, 3), & (1, 0, 0, 2, 0, 0, 0, 0, 3), \\
(2, 0, 0, 0, 0, 0, 1, 0, 3), & (2, 0, 0, 1, 0, 0, 0, 0, 3), \\
(0, 0, 0, 1, 0, 0, 2, 0, 4), & (0, 0, 0, 1, 0, 0, 3, 0, 4), \\
\ldots
\end{align*}
$$

where variables $I_1$ and $I_2$ are used to encode reference processes.

An environment disjunctive relation is called *valid* if it does not contain points where the value of any index variable is outside the range of indexes.
for the given system size, which can be expressed formally as

\[
\forall I_1, X_1, \ldots, I_k, X_k, I, X, N, G \left( R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \rightarrow (0 \leq I_1, \ldots, I_k, I < N) \right).
\]

An environment disjunctive relation \( R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) is called filtered if it is valid and every disjunctive relation, which can be obtained by the instantiation of all family and collection parameters \( I_1, X_1, \ldots, I_k, X_k, N, G \), is filtered. This property can be formalized as follows:

\[
\forall I_1, X_1, \ldots, I_k, X_k, I, X, N, G \left[ \left( \forall 0 \leq I < N \land I \neq I_1 \land \ldots \land I \neq I_k \right) \land \left( \exists I, X . R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \right) \right] \vee \left( \neg \exists I, X . R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \right).
\]

We will require this property to test for the inclusion and the set emptiness (Sections 4.9 and 4.10). How to filter an EDR is explained in Section 4.9.

### 4.6 Set Intersection

Here we consider the intersection of sets of global states in terms of operations on their intermediate representations. The problem can be defined as follows. Given two environment disjunctive relations \( R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) and \( R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) representing sets \([R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)]\) and \([R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)]\) of global states we wish to obtain a disjunctive relation \( R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) such that

\[
[R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] \cap [R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] = [R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)].
\]

First, we examine the simple case of the intersection of two disjunctive sets.

**Proposition 4.6.1** Let \( R_1(I, X) \) and \( R_2(I, X) \) be disjunctive relations and \([R_1(I, X)]\) and \([R_2(I, X)]\) be disjunctive sets of global states represented by these relations. Then

\[
[R_1(I, X) \cap R_2(I, X)] = [R_1(I, X)] \cap [R_2(I, X)].
\]

**Proof:** (\(\subseteq\)) Assume, that \( s \in [R_1(I, X) \cap R_2(I, X)] \) and \( s = \{ X[0] \mapsto a_0, \ldots, X[n - 1] \mapsto a_{n-1} \} \). Then, according to the semantics of the disjunctive relation, \( \{ (0, a_0), \ldots, (n-1, a_{n-1}) \} \subseteq (R_1(I, X) \cap R_2(I, X)) \), hence \( \{ (0, a_0), \ldots, (n-1, a_{n-1}) \} \subseteq [R_1(I, X)] \cap [R_2(I, X)] \).
(n-1,a_{n-1}) \subseteq R_1(I,X) \text{ and } \{(0,a_0),\ldots,(n-1,a_{n-1})\} \subseteq R_2(I,X). Therefore, from the definition of the disjunctive relation, \(s \in [R_1(I,X)] \text{ and } s \in [R_2(I,X)]\), which implies that \(s \in ([R_1(I,X)] \cap [R_2(I,X)])\) and, consequently, if \(s \in [R_1(I,X) \cap R_2(I,X)]\) then \(s \in ([R_1(I,X)] \cap [R_2(I,X)])\).

(\supseteq) Assume now, that \(s \in ([R_1(I,X)] \cap [R_2(I,X)])\) and \(s = \{X[0] \mapsto a_0, \ldots, X[n-1] \mapsto a_{n-1}\}\). Obviously, \(s \in [R_1(I,X)] \text{ and } s \in [R_2(I,X)]\) (by the definition of set intersection), which implies that \(\{(0,a_0),\ldots,(n-1,a_{n-1})\} \subseteq R_1(I,X) \text{ and } \{(0,a_0),\ldots,(n-1,a_{n-1})\} \subseteq R_2(I,X)\), therefore \(\{(0,a_0),\ldots,(n-1,a_{n-1})\} \subseteq (R_1(I,X) \cap R_2(I,X)) \text{ and } s \in [R_1(I,X) \cap R_2(I,X)]\).

\[ \square \]

**Example 4.6.2** Assume that we have two disjunctive sets \(S_1 = \{s_1, s_2, s_3, s_4, s_5, s_6\}\) and \(S_1 = \{s_3, s_4, s_7\}\) of global states a system of size \(n = 3\) where

\[
\begin{align*}
  s_1 &= \{X[0] \mapsto 0, X[1] \mapsto 1, X[2] \mapsto 0\}, \\
  s_2 &= \{X[0] \mapsto 0, X[1] \mapsto 1, X[2] \mapsto 1\}, \\
  s_3 &= \{X[0] \mapsto 1, X[1] \mapsto 1, X[2] \mapsto 0\}, \\
  s_4 &= \{X[0] \mapsto 1, X[1] \mapsto 1, X[2] \mapsto 1\}, \\
  s_5 &= \{X[0] \mapsto 2, X[1] \mapsto 1, X[2] \mapsto 0\}, \\
  s_6 &= \{X[0] \mapsto 2, X[1] \mapsto 1, X[2] \mapsto 1\}, \\
  s_7 &= \{X[0] \mapsto 1, X[1] \mapsto 1, X[2] \mapsto 2\}.
\end{align*}
\]

Sets \(S_1\) and \(S_2\) can be represented as disjunctive relations relation \(R_1(I,X) = \{(0,0), (0,1), (0,2), (1,1), (2,0), (2,1)\}\) and \(R_2(I,X) = \{(0,1), (1,1), (2,0), (2,1), (2,2)\}\), respectively.

The intersection \(S = S_1 \cap S_2 = \{s_3, s_4\}\) can be represented as a disjunctive relation \(R(I,X) = \{(0,1), (1,1), (2,0), (2,1)\}\) such that \(R(I,X) = R_1(I,X) \cap R_2(I,X)\), which is illustrated in Figure 4.8 \(\square\)

If parameterized disjunctive relations with collection parameters are taken into consideration, then the intersection of two sets of global states can be expressed via a relation with a composite collection parameter.
Proposition 4.6.3 Let \( R_1(I, X, G) \) and \( R_2(I, X, G) \) be parameterized disjunctive relations with collection parameters and \([R_1(I, X, G)]\) and \([R_2(I, X, G)]\) be disjunctive sets of global states represented by these relations. Then
\[
\left[ \{(I, X, G_1, G_2) \mid R_1(I, X, G_1) \land R_2(I, X, G_2)\} \right] = [R_1(I, X, G)] \cap [R_2(I, X, G)].
\]

**Proof:** Assume, that the disjunctive relation \( R(I, X, G_1, G_2) \) is defined as
\[
\left[ \{(I, X, G_1, G_2) \mid R_1(I, X, G_1) \land R_2(I, X, G_2)\} \right].
\]
Let us show that \( s \in [R(I, X, G_1, G_2)] \) if \( s \in ([R_1(I, X, G)] \cap [R_2(I, X, G)]) \) for any global state \( s \).

(\( \subseteq \)) Assume that \( s \in [R(I, X, G_1, G_2)] \) and \( s = \{X[0] \mapsto a_0, \ldots, X[n-1] \mapsto a_{n-1}\} \). Then, according to the semantics of the disjunctive relation, \( \{(0, a_0, g_1, g_2), \ldots, (n-1, a_{n-1}, g_1, g_2)\} \subseteq R(I, X, G_1, G_2) \) for some \( g_1 \) and \( g_2 \), hence \( \{(0, a_0, g_1), \ldots, (n-1, a_{n-1}, g_1)\} \subseteq R_1(I, X, G) \) and \( \{(0, a_0, g_2), \ldots, (n-1, a_{n-1}, g_2)\} \subseteq R_2(I, X, G) \). Therefore, by the semantics of the disjunctive relation, \( s \in [R_1(I, X, G)] \) and \( s \in [R_2(I, X, G)] \), which implies that \( s \in ([R_1(I, X, G)] \cap [R_2(I, X, G)]) \). Hence if \( s \in [R(I, X, G_1, G_2)] \) then \( s \in ([R_1(I, X, G)] \cap [R_2(I, X, G)]) \).

(\( \supseteq \)) Assume now, that \( s \in ([R_1(I, X, G)] \cap [R_2(I, X, G)]) \) and \( s = \{X[0] \mapsto a_0, \ldots, X[n-1] \mapsto a_{n-1}\} \). Obviously, \( s \in [R_1(I, X, G)] \) and \( s \in [R_2(I, X, G)] \) (by the definition of set intersection), which implies that \( \{(0, a_0, g_1), \ldots, (n-1, a_{n-1}, g_1)\} \subseteq R_1(I, X, G) \) for some \( g_1 \) and \( \{(0, a_0, g_2), \ldots, (n-1, a_{n-1}, g_2)\} \subseteq R_2(I, X, G) \) for some \( g_2 \), therefore \( \{(0, a_0, g_1, g_2), \ldots, (n-1, a_{n-1}, g_1, g_2)\} \subseteq R(I, X, G_1, G_2) \) by the definition of \( R(I, X, G_1, G_2) \) and \( s \in [R(I, X, G_1, G_2)] \).

As we can see from Proposition 4.6.3, parameterized disjunctive relations with collection parameters will grow in dimension with every intersection. However, this drawback does not apply to family or size parameters as we can see from the following proposition.

Proposition 4.6.4 Let \( R_1(I, X, P) \) and \( R_2(I, X, P) \) be disjunctive meta-relations with family or size parameter \( P \) such that \([R_1(I, X, P)] = \{S_p\}\) and \([R_2(I, X, P)] = \{S'_p\}\) are families of sets of global states for some family of uniform systems. Then
\[
\left[ \{(I, X, P) \mid R_1(I, X, P) \land R_2(I, X, P)\} \right] = \{S_p \cap S'_p\}
\]
Proof: The proof is the same as in Proposition 4.6.1 if we consider a pair of disjunctive relations $R'_i(I, X) = \{(I, X) \mid R_i(I, X, p)\}$ and $R'_2(I, X) = \{(I, X) \mid R_2(I, X, p)\}$ representing the sets of global states $S_p$ and $S'_p$ for any $p \in P$.

In the following proposition we provide for a special case of handling reference collection parameters in terms of the intersection of two relations, in contrast to the general case for collection parameters covered in Proposition 4.6.3.

**Proposition 4.6.5** Let $R_1(I_1, X_1, I, X)$ and $R_2(I_1, X_1, I, X)$ be two environment disjunctive relations with one reference process such that $[R_1(I_1, X_1, I, X)] = \{S_i\}$ and $[R_2(I_1, X_1, I, X)] = \{S'_i\}$ are families of sets of global states of some uniform systems where the index of the reference process (encoded as $I_1$) is a parameter. Then

$$
\left[ \{(I_1, X_1, I, X) \mid R_1(I_1, X_1, I, X) \land R_2(I_1, X_1, I, X)\} \right] = \{S_i \cap S'_i\}
$$

Proof: In the same manner as in the proof of Proposition 4.6.4 we will consider relations $R_{1,i}(X_1, I, X) = \{(X_1, I, X) \mid R_{1,i}(i, X_1, I, X)\}$ and $R_{2,i}(X_1, I, X) = \{(X_1, I, X) \mid R_{2,i}(i, X_1, I, X)\}$ pairwise for any $i \in \mathbb{I}$. As was demonstrated in Section 4.5, EDRs $R_{1,i}(X_1, I, X)$ and $R_{2,i}(X_1, I, X)$ can be considered as collections $\{R_{1,i,a}\}_{a \in A} = \{\{(I, X) \mid R_{1,i}(a, I, X)\}\}_{a \in A}$ and $\{R_{2,i,a}\}_{a \in A} = \{\{(I, X) \mid R_{2,i}(a, I, X)\}\}_{a \in A}$ of disjunctive relations. Hence, their intersection can be expressed as the collection $\{(a_1, a_2, I, X) \mid R_{1,i}(a_1, I, X) \land R_{2,i}(a_2, I, X)\}\}_{a_1, a_2 \in A}$ of parameterized disjunctive relations (according to Proposition 4.6.3). However, the intersection $R_{1,i,a_1}(I, X)$ and $R_{2,i,a_2}(I, X)$ will result in the empty set of global states for $a_1 \neq a_2$ because the reference process $i$ cannot be in two different local states at the same time. Therefore, we can take pairwise intersections of families $\{R_{1,i,a}(I, X)\}_{a \in A}$ and $\{R_{1,i,a}(I, X)\}_{a \in A}$ of disjunctive relations only when the local state $\{X_1 \rightarrow a\}$ of the respective reference process is the same.

In the same way we can prove that Proposition 4.6.5 holds for any fixed number of reference processes.

Let us now generalize intersection to families of sets of global states represented by environment disjunctive relations.

**Proposition 4.6.6** Let $R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ and $R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ be environment disjunctive relations such that $[R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] = \{S_{i_1, \ldots, i_k, n}\}$ and $[R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] = \{S'_{i_1, \ldots, i_k, n}\}$ are families of sets of global states for some family of uniform
systems. Then
\[
\{ (I_1, X_1, \ldots, I_k, X_k, I, X, N, G_1, G_2) \mid R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G_1) \\
\wedge R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G_2) \}\}
= \{ S_{i_1, \ldots, i_k, n} \cap S'_{i_1, \ldots, i_k, n} \}
\]

We omit the proof for Proposition 4.6.6, because it is enough to follow the ideas from Propositions 4.6.1-4.6.5 and integrate their proofs together.

### 4.7 Set Union

Disjunctive sets are not closed under set union operation which is illustrated with the following example.

**Example 4.7.1** The sets \( S_1 = \{ \{X[0] \mapsto 0, X[1] \mapsto 1\} \} \) and \( S_2 = \{ \{X[0] \mapsto 1, X[1] \mapsto 0\} \} \) are both disjunctive because there exist disjunctive relations \( R_1(I, X) = \{(0,0), (1,1)\} \) and \( R_2 = \{(0,1), (1,0)\} \) such that \( \{R_1(I, X)\} = S_1 \) and \( \{R_2(I, X)\} = S_2 \). However, the set \( S_1 \cup S_2 \) is not disjunctive, which was explained in Example 4.3.1.

However, it is possible to perform the set union operation on two disjunctive sets represented as disjunctive relations by the introduction of a collection parameter. The idea of treating any set of global states as a collection of disjunctive states was already discussed in Section 4.3.

**Proposition 4.7.2** Let \( R_1(I, X) \) and \( R_2(I, X) \) be two disjunctive relations and \( \{R_1(I, X)\} \) and \( \{R_2(I, X)\} \) be two disjunctive sets represented by these relations. Then
\[
\{ (I, X, G) \mid (R_1(I, X) \wedge G = 1) \lor (R_2(I, X) \wedge G = 2) \}
= \{R_1(I, X)\} \cup \{R_2(I, X)\}.
\]

**Proof:** Assume, that the disjunctive relation \( R(I, X, G) \) is defined as
\[
\{ (I, X, G) \mid (R_1(I, X) \wedge G = 1) \lor (R_2(I, X) \wedge G = 2) \}.
\]
Let us show that \( s \in [R(I, X, G)] \) iff \( s \in ([R_1(I, X)] \cup [R_2(I, X)]) \) for any global state \( s \).

(\( \subseteq \)) Assume that \( s \in [R(I, X, G)] \) and \( s = \{X[0] \mapsto a_0, \ldots, X[n-1] \mapsto a_{n-1}\} \). Then, according to the semantics of the disjunctive
relation, \( \{(0, a_0), \ldots, (n-1, a_{n-1}), g\} \subseteq R(I, X, G) \) for \( g = 1 \) or \( g = 2 \), hence, \( \{(0, a_0), \ldots, (n-1, a_{n-1})\} \subseteq R_1(I, X) \) or \( \{(0, a_0), \ldots, (n-1, a_{n-1})\} \subseteq R_2(I, X) \). Therefore, by the semantics of the disjunctive relation, \( s \in [R_1(I, X)] \) or \( s \in [R_2(I, X)] \), which implies that \( s \in ([R_1(I, X)] \cup [R_2(I, X)]) \). Hence if \( s \in [R(I, X, G)] \) then \( s \in ([R_1(I, X)] \cap [R_2(I, X)]) \).

(\(\geq\)) Assume now, that \( s \in ([R_1(I, X)] \cup [R_2(I, X)]) \) and \( s = \{X[0] \mapsto a_0, \ldots, X[n-1] \mapsto a_{n-1}\} \). Obviously, \( s \in [R_1(I, X)] \) or \( s \in [R_2(I, X)] \) (by the definition of set union), which implies that \( \{(0, a_0), \ldots, (n-1, a_{n-1})\} \subseteq R_1(I, X) \) or \( \{(0, a_0), \ldots, (n-1, a_{n-1})\} \subseteq R_2(I, X) \), therefore \( \{(0, a_0, 1), \ldots, (n-1, a_{n-1}, 1)\} \subseteq R(I, X, G) \) or \( \{(0, a_0, 2), \ldots, (n-1, a_{n-1}, 2)\} \subseteq R(I, X, G) \) (by the definition of \( R(I, X, G) \)) in which case \( s \in [R(I, X, G)] \).

Of course, instead of \( G = 1 \) and \( G = 2 \) one can take any other different values.

The same idea is used for environment disjunctive relations.

**Proposition 4.7.3** Let \( R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) and \( R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) be two environment disjunctive relations such that
\[
[R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] = \{S_{i_1, \ldots, i_k, n}\}
\]
and
\[
[R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] = \{S'_{i_1, \ldots, i_k, n}\}
\]
are families of sets of global states for some family of uniform systems. Then
\[
\begin{align*}
\{ & (I_1, X_1, \ldots, I_k, X_k, I, X, N, G, G') \mid \\
& (R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \land G' = 1) \\
& \lor \\
& (R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \land G' = 2) \} \\
= \\
\{ & S_{i_1, \ldots, i_k, n} \cup S'_{i_1, \ldots, i_k, n} \}.
\end{align*}
\]

A disadvantage of the method above is that the number of collection parameters in general is increasing each time we perform the operation. Since the set union is a very important and usually frequently repeated operation during reachability analysis, we may use two other methods instead, which do not affect the arity of the resulting relation.

The first idea of the set union operation presented below is that we can take the union of two sets of global states represented by parameterized disjunctive relations with collection parameters, if the values of the collection parameters are disjoint for those relations, that is, the relations do not contain points with the same values of the collection parameter as in points of another relation.

52
Proposition 4.7.4 Let $R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ and $R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ be environment disjunctive relations such that $[R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] = \{S_{i_1, \ldots, i_k, n}\}$ and $[R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] = \{S'_{i_1, \ldots, i_k, n}\}$ are families of sets of global states for some family of uniform systems. If the sets of values of the collection parameter $G$ appearing in tuples of $R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ and $R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ are disjoint, then

$$\lbrack \{ (I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \mid R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \lor R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \rbrack = \{S_{i_1, \ldots, i_k, n} \cup S'_{i_1, \ldots, i_k, n}\}.$$  

Note, that we can always transform any two relations $R_1$ and $R_2$ with collection parameter into $R'_1$ and $R'_2$ such that they will satisfy the condition stated in Proposition 4.7.4. For example, we can apply the following transformations:

$$R'_1 = \{ (I, X, G') \mid R_1(I, X, G) \land G' = 2G \}$$

$$R'_2 = \{ (I, X, G') \mid R_2(I, X, G) \land G' = 2G + 1 \}$$

Such transformations do not change the semantics of the resulting relations, that is, $[R'_1] = [R_1]$ and $[R'_2] = [R_2]$.

Sometimes we may be interested neither to increase the number of parameters in a formula nor to change parameters' domain since those methods introduce additional complexity. Then we may use the following proposition at the cost of additional over-approximation.

Proposition 4.7.5 Let $R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ and $R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ be environment disjunctive relations such that $[R_1] = \{S_{i_1, \ldots, i_k, n}\}$ and $[R_2] = \{S'_{i_1, \ldots, i_k, n}\}$ are families of sets of global states for some family of uniform systems. Then

$$\lbrack \{ (I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \mid R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \lor R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \rbrack \supseteq \{S_{i_1, \ldots, i_k, n} \cup S'_{i_1, \ldots, i_k, n}\}.$$  

\[53\]
4.8 Set Difference and Set Complement

In this section we discuss obtaining set difference (relative complement) of two sets of global states expressed in the intermediate representation. Note, that this topic covers also the complement (sometimes called absolute complement) of a set of global states which can be considered as a special case of set difference.

Let us consider first the situation when both sets of global states are represented as disjunctive relations.

**Proposition 4.8.1** Let \( R_1(I,X) \) and \( R_2(I,X) \) be disjunctive relations and \([R_1(I,X)]\) and \([R_2(I,X)]\) be two disjunctive sets of global states represented be these relations. Then

\[
\begin{align*}
\left[ \{ (I,X,G) \mid G \in I \land R_1(I,X) \land (R_2(I,X) \rightarrow I \neq G) \} \right] \\
= \left[ R_1(I,X) \right] \setminus \left[ R_2(I,X) \right]
\end{align*}
\]

where \( G \) is a collection parameter, and \( I = \{0,\ldots,n-1\} \) is the set of components.

**Proof:** (\( \subseteq \)) Assume, that
\[
s \in \left[ \{ (I,X,G) \mid G \in I \land R_1(I,X) \land (R_2(I,X) \rightarrow I \neq G) \} \right]
\]
and \( s = \{ X[0] \mapsto a_0, \ldots, X[n-1] \mapsto a_{n-1} \} \). From the semantics of the parameterized disjunctive relation we know that

\[
\left[ \{ (I,X,G) \mid G \in I \land R_1(I,X) \land (R_2(I,X) \rightarrow I \neq G) \} \right] \\
= \bigcup_{g \in I} \left[ \{ (I,X) \mid R_1(I,X) \land (R_2(I,X) \rightarrow I \neq g) \} \right].
\]

If \( s \in (\bigcup_{g \in I} \left[ \{ (I,X) \mid R_1(I,X) \land (R_2(I,X) \rightarrow I \neq g) \} \right]) \) then there exists some \( g \in I \) such that \( s \in \left[ \{ (I,X) \mid R_1(I,X) \land (R_2(I,X) \rightarrow I \neq g) \} \right] \). Consequently, by the semantics of the disjunctive relation, there exists \( g \in I \) such that for all \( (i,a_i) \in \{ (0,a_0),\ldots,(n-1,a_{n-1}) \} \) we have

\[
R_1(i,a_i) \land (R_2(i,a_i) \rightarrow i \neq g),
\]
which implies that \( \{ (0,a_0),\ldots,(n-1,a_{n-1}) \} \subseteq R_1(I,X) \) and \( \{ (0,a_0),\ldots,(n-1,a_{n-1}) \} \not\subseteq R_2(I,X) \). This means that \( s \in [R_1(I,X)] \) and \( s \not\in [R_2(I,X)] \).

(\( \supseteq \)) Assume now, that there is a state \( s = \{ X[0] \mapsto a_0, \ldots, X[n-1] \mapsto a_{n-1} \} \) such that \( s \in ([R_1(I,X)] \setminus [R_2(I,X)]) \) and \( s \not\in \left[ \{ (I,X,G) \mid G \in I \land R_1(I,X) \land (R_2(I,X) \rightarrow I \neq G) \} \right] \). Then, according to the semantics of the
disjunctive relation, \(\{(0, a_0), \ldots, (n-1, a_{n-1})\} \subseteq R_1(I, X)\) and \(\{(0, a_0), \ldots, (n-1, a_{n-1})\} \not\subseteq R_2(I, X)\). This means, that there exists at least one point \((i, a_i) \in \{(0, a_0), \ldots, (n-1, a_{n-1})\}\) for some \(i \in I\) such that \((i, a_i) \in R_1(I, X)\) and \((i, a_i) \not\in R_2(I, X)\). Therefore, \(s \in \{\{I, X, G\} \mid G \in I \land R_2(I, X) \land (R_2(I, X) \rightarrow I \neq G)\}\) by the semantics of the parameterized disjunctive relation with collection parameter, because \(\{(0, a_0), \ldots, (i, a_i), i, \ldots, (n-1, a_{n-1})\} \subseteq \{\{I, X, G\} \mid G \in I \land R_1(I, X) \land (R_2(I, X) \rightarrow I \neq G)\}\), which contradicts our initial assumption. Hence, \(s \in \{\{I, X, G\} \mid G \in I \land R_1(I, X) \land (R_2(I, X) \rightarrow I \neq G)\}\) whenever \(s \in ([R_1(I, X)] \setminus [R_2(I, X)])\). 

Let us consider environment disjunctive relations.

**Proposition 4.8.2** Let \(R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N)\) and \(R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N)\) be environment disjunctive relations without optional collection parameters and \([R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N)] = \{S_{i_1, \ldots, i_k, n}\}\) and \([R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N)] = \{S'_{i_1, \ldots, i_k, n}\}\) be two families of sets of global states represented by these relations. Then

\[
\begin{align*}
\{\{I_1, X_1, \ldots, I_k, X_k, I, X, N, G\} \
&\text{ whenever } 0 \leq G < N \land R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N) \land \\
&\quad (R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N) \rightarrow I \neq G)\} \\
&= \{S_{i_1, \ldots, i_k, n} \setminus S'_{i_1, \ldots, i_k, n}\}.
\end{align*}
\]

Let us finally introduce set difference on parameterized disjunctive relations. From the set theory we know that

\[
(S_1 \cup \ldots \cup S_k) \setminus (S'_1 \cup \ldots \cup S'_m) = ([S_1 \cup \ldots \cup S_k] \setminus S'_1] \cap \ldots \cap ([S_1 \cup \ldots \cup S_k] \setminus S'_m] = ((S_1 \setminus S'_1) \cup \ldots \cup (S_1 \setminus S'_m)) \cap \ldots \cap ((S_k \setminus S'_1) \cup \ldots \cup (S_k \setminus S'_m)) = \bigcap_{i=1}^k \bigcup_{j=1}^m (S_i \setminus S_j).
\]

Therefore, when it comes to the set difference of two parameterized disjunctive relations \(R_1(I, X, G)\) and \(R_2(I, X, G)\), where the collection parameter
CHAPTER 4. INTERMEDIATE REPRESENTATION

$G$ ranges over $G_1$ and $G_2$, respectively, we can apply the above rule to compute $[R_1(I, X, G)] \setminus [R_2(I, X, G)]$ as

$$\bigcap_{i \in G_1} \bigcup_{j \in G_2} ([R_1,i(I, X)] \setminus [R_2,j(I, X)])$$

where $R_{1,i}(I, X)$ iff $R_1(I, X, i)$ and $R_{1,j}(I, X)$ iff $R_2(I, X, j)$.

Obviously, it is possible to compute the set difference this way only if $G_2$ is finite and bounded, otherwise, the technique presented here does not help.

This idea can be applied to environment disjunctive relations, but, unfortunately, in many cases optional parameters range over unbounded domains, like in case of the optional parameter introduced in Proposition 4.8.2.

Proposition 4.8.3 Let $R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ and $R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ be environment disjunctive relations such that in the second EDR the optional parameter $G$ ranges over finite and bounded domain \{g_1, \ldots, g_m\}, while \[[R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)]\] = \{S_{i_1}^{1}, \ldots, S_{i_k}^{1}\} and \[[R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)]\] = \{S_{i_1}^{2}, \ldots, S_{i_k}^{2}\} be two families of sets of global states represented be these relations. Then

\[
\begin{align*}
&\{ (I_1, X_1, \ldots, I_k, X_k, I, X, N, G_1, G_1', \ldots, G_m, G_m') \mid \\
&0 \leq G_1' < N \land R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G_1) \\
&(R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, g_1) \rightarrow I \neq G_1') \\
&\ldots \\
&0 \leq G_m' < N \land R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G_m) \\
&(R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, g_m) \rightarrow I \neq G_m') \} \]
\end{align*}
\]

= \{S_{i_1}^{1}, \ldots, S_{i_k}^{1} \setminus S_{i_1}^{2}, \ldots, S_{i_k}^{2}\}.

\[\square\]

The operation of set (absolute) complement for environment disjunctive relations is easily derived from Proposition 4.8.3. In that case the first set of global states represents all possible states in the system, therefore, its intermediate representation includes all possible points.

Proposition 4.8.4 Let $R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ be environment disjunctive relation such that the optional parameter $G$ ranges over finite and bounded domain \{g_1, \ldots, g_m\}, and \[[R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)]\] = \{S_{i_1}^{1}, \ldots, S_{i_k}^{1}\} is a family of sets of global states represented be this relations. Then

\[
\begin{align*}
&\{ (I_1, X_1, \ldots, I_k, X_k, I, X, N, G_1, \ldots, G_m) \mid \\
&\}\]
\]

56
0 ≤ G_1 < N \land (R(I_1, X_1, \ldots, I_k, X_k, I, X, N, g_1) \rightarrow I \neq G_1) \\
\wedge \ldots \wedge \\
0 ≤ G_m < N \land (R(I_1, X_1, \ldots, I_k, X_k, I, X, N, g_m) \rightarrow I \neq G_m))

= \\
\{ \bar{S}_{i_1, \ldots, i_k, n} \}.

### 4.9 Set Emptiness

An important operation is to test if a set of global states expressed in the intermediate representation is empty. In general, it is not enough just to examine, whether a disjunctive relation is empty or not, because even a non-empty relation may denote the empty set of states. However, the task is trivial if the intermediate representation is filtered.

**Proposition 4.9.1** Let $R(I, X)$ be a filtered disjunctive relation. A set $[R(I, X)]$ of global states is empty iff $R(I, X) = \emptyset$.

**Proof**: ($\Leftarrow$) If $R(I, X) = \emptyset$, then there cannot be any set $\{(0, a_0), \ldots, (n-1, a_{n-1})\} \subseteq R(I, X)$, therefore $[R(I, X)] = \emptyset$.

($\Rightarrow$) Let $[R(I, X)] = \emptyset$ where $R(I, X)$ is a filtered disjunctive relation. Assume that $R(I, X) \neq \emptyset$. Then, by the definition of the filtered disjunctive relation, $\forall I \exists X \in A R(I, X)$. In other words, there exists a set $\{(0, a_0), \ldots, (n-1, a_{n-1})\} \subseteq R(I, X)$, hence, by the definition of the semantics of the disjunctive relation, $\{X[0] \mapsto a_0, \ldots, X[n-1] \mapsto a_{n-1}\} \in [R(I, X)]$, which contradicts our assumption. Thus, $R(I, X) = \emptyset$ whenever $R(I, X)$ is filtered and $[R(I, X)] = \emptyset$.

The same idea applies for parameterized disjunctive relations with collection parameter.

**Proposition 4.9.2** Let $R(I, X, G)$ be a filtered parameterized disjunctive relation with a collection parameter $G$. Then $[R(I, X, G)] = \emptyset$ iff $R(I, X, G) = \emptyset$.

In the rest of this section we concentrate on a relationship between any intermediate representation and its filtered version. This relationship will be used later in Section 5.3 to filter system descriptive formulas introduced in the next chapter.

**Proposition 4.9.3** Let $R(I, X, G)$ be a parameterized disjunctive relation with a collection parameter $G$. Then the parameterized disjunctive relation $R'(I, X, G)$ defined as

$$\{(I, X, G) \mid R(I, X, G) \land \forall I \exists X R(I, X, G)\}$$

is filtered and $[R'(I, X, G)] = [R(I, X, G)]$.
For environment disjunctive relation we use the same idea as above.

**Proposition 4.9.4** Let \( R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) be a environment disjunctive relation with parameter \( G \). Then the environment disjunctive relation \( R'(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) defined as

\[
\{ (I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \mid R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \land \forall_{0 \leq i < N} \exists X R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \}
\]

is filtered and

\[
[R'(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] = [R(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)].
\]


**4.10 Inclusion Test**

Inclusion test is a very important operation usually performed at every iteration of the reachability analysis to check if the termination condition is reached. In general, we have \( S_1 \subseteq S_2 \) iff \( S_1 \cap \bar{S}_2 = \emptyset \). If \( S_2 \) is represented as a parameterized disjunctive relation with a collection parameter, then, as we explained in Section 4.8, its complement can be represented in the general case as a pDR only if the collection parameter ranges over finite and bounded domain. Therefore, the inclusion test is not applicable with our method if the collection parameter of a would-be superset is unbounded. However, we provide some alternative algorithms which may help to test the inclusion.

For two disjunctive relations the problem is trivial.

**Proposition 4.10.1** Let \( R_1(I, X) \) and \( R_2(I, X) \) be two disjunctive relations representing sets \([R_1(I, X)]\) and \([R_2(I, X)]\) of global states such that \( R_1(I, X) \) is filtered. Then

\[
[R_1(I, X)] \subseteq [R_2(I, X)]
\]

iff

\[
R_1(I, X) \subseteq R_2(I, X)
\]

**Proof:** \((\Rightarrow)\) Assume that \([R_1(I, X)] \subseteq [R_2(I, X)]\) and there is a point \( R_1(i, a_i) \). Since \( R_1(I, X) \) is filtered and \( R_1(i, a_i) \) holds, there exists a set \( \{ (0, a_0), \ldots, (i, a_i), \ldots, (n-1, a_{n-1}) \} \subseteq R_1(I, X) \). Hence, \( \{ X[0] \mapsto a_0, \ldots, X[i] \mapsto a_i, \ldots, X[n-1] \mapsto a_{n-1} \} \in [R_1(I, X)] \) by the semantics of the disjunctive relation. Since \([R_1(I, X)] \subseteq [R_2(I, X)]\), we have \( \{ X[0] \mapsto a_0, \ldots, \)

58
\[ X[i] \mapsto a_1, \ldots, X[n-1] \mapsto a_{n-1} \in [R_2(I,X)]. \] From the semantics of the disjunctive relation follows that \( \{(0,a_0), \ldots, (i,a_i), \ldots, (n-1,a_{n-1})\} \subseteq R_2(I,X) \) and, in particular, \( R_2(i,a_i) \) holds.

In view of the fact, that for any point \((i,a_i)\) we have \( R_2(i,a_i) \) whenever \( R_1(i,a_i) \) and \([R_1(I,X)] \subseteq [R_2(I,X)]\), we can conclude that \( R_1(I,X) \subseteq R_2(I,X) \) whenever \([R_1(I,X)] \subseteq [R_2(I,X)]\).

(\(\Rightarrow\)) Assume that \( R_1(I,X) \subseteq R_2(I,X) \) and there exists a global state \( s \in [R_1(I,X)] \) such that \( s = \{X[0] \mapsto a_0, \ldots, X[i] \mapsto a_i, \ldots, X[n-1] \mapsto a_{n-1}\} \). By the semantics of the disjunctive relation, \( \{(0,a_0), \ldots, (i,a_i), \ldots, (n-1,a_{n-1})\} \subseteq R_1(I,X) \). On the other side, we have \( \{(0,a_0), \ldots, (i,a_i), \ldots, (n-1,a_{n-1})\} \subseteq R_2(I,X) \) because \( R_1(I,X) \subseteq R_2(I,X) \). Applying the semantics of the disjunctive relation again, we obtain that \( s \in [R_2(I,X)] \).

\(\square\)

The same idea applies to the environment disjunctive relations if they do not contain optional collection parameters.

**Proposition 4.10.2** Let \( R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N) \) and \( R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N) \) be two filtered environment disjunctive relations without optional collection parameter representing families \([R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N)] = \{S_{i_1, \ldots, i_k, n}\}\) and \([R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N)] = \{S'_{i_1, \ldots, i_k, n}\}\) of sets of global states. Then

\[
\{S_{i_1, \ldots, i_k, n} \subseteq S'_{i_1, \ldots, i_k, n}\}
\]

if

\[
R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N) \subseteq R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N).
\]

\(\square\)

According to what we have mentioned in the beginning of this section, we can test inclusion if the representation of a would-be subset has an optional collection parameter but not the representation of a would-be superset.

**Proposition 4.10.3** Let \( R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) and \( R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N) \) be two filtered environment disjunctive relations representing families \([R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] = \{S'_{i_1, \ldots, i_k, n}\}\) and \([R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N)] = \{S_{i_1, \ldots, i_k, n}\}\) of sets of global states, where EDR \( R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N) \) does not have an optional collection parameter. Then

\[
\{S'_{i_1, \ldots, i_k, n} \subseteq S_{i_1, \ldots, i_k, n}\}
\]

whenever

\[
R'_1(I_1, X_1, \ldots, I_k, X_k, I, X, N) \subseteq R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N)
\]

where \( R'_1(I_1, X_1, \ldots, I_k, X_k, I, X, N) = \{(I_1, X_1, \ldots, I_k, X_k, I, X, N) | \exists G.R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)\}\).

\(\square\)
CHAPTER 4. INTERMEDIATE REPRESENTATION

Proof: Let \([R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N)]\) be a family \(\{S_{i_1, \ldots, i_n}^{\prime \prime}\}\) of sets of global states and \(R_1'(I_1, X_1, \ldots, I_k, X_k, I, X, N) \subseteq R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N)\) for the rest of the proof. Then, according to Proposition 4.10.2, we have \(\{S_{i_1, \ldots, i_n}^{\prime \prime} \subseteq S_{i_1, \ldots, i_n}\}\). In what follows we will prove that \(\{S_{i_1, \ldots, i_n}^{\prime \prime} \subseteq S_{i_1, \ldots, i_n}\}\). This obviously results in \(\{S_{i_1, \ldots, i_n}^{\prime \prime} \subseteq S_{i_1, \ldots, i_n}\}\) (assuming \(\{S_{i_1, \ldots, i_n}^{\prime \prime} \subseteq S_{i_1, \ldots, i_n}\}\)).

Assume that \(S_{i_1, \ldots, i_n}^{\prime \prime} \not\subseteq S_{i_1, \ldots, i_n}\) for some \(i_1, \ldots, i_n \in \mathbb{I}\) and \(n \in M\). It means that there exists a global state \(s = [X[i_1] \mapsto a_1, \ldots, X[i_k] \mapsto a_k, X[j_1] \mapsto b_1, \ldots, X[j_m] \mapsto b_m]\) where \(\{j_1, \ldots, j_m\} = \mathbb{I}\{i_1, \ldots, i_k\}\) and \(a_1, \ldots, a_k, b_1, \ldots, b_m \in A\), such that \(s \in S_{i_1, \ldots, i_n}^{\prime \prime}\) but \(s \not\in S_{i_1, \ldots, i_n}\). By the EDR semantics, we have \(\{\langle i_1, a_1, \ldots, i_k, a_k, j_1, b_1, n, g \rangle, \ldots, \langle i_1, a_1, \ldots, i_k, a_k, j_m, b_m, n, g \rangle\} \subseteq R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)\) for some \(g \in G_n\). Therefore, \(\{\langle i_1, a_1, \ldots, i_k, a_k, j_1, b_1, n, g \rangle, \ldots, \langle i_1, a_1, \ldots, i_k, a_k, j_m, b_m, n, g \rangle\} \subseteq R_1''(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)\) by the definition of \(R_1''(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)\), which results in \(s \in S_{i_1, \ldots, i_n}^{\prime \prime}\) by the semantics of EDR. The last conclusion contradicts our assumption, hence it was wrong and we have \(\{S_{i_1, \ldots, i_n} \subseteq S_{i_1, \ldots, i_n}^{\prime \prime}\}\).

If both relations contain optional collection parameters then the following sufficient condition may help to test set inclusion.

**Proposition 4.10.4** Let \(R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)\) and \(R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)\) be two filtered environment disjunctive relations with optional collection parameters representing families \([R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] = \{S_{i_1, \ldots, i_n}^{\prime \prime}\}\) and \([R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] = \{S_{i_1, \ldots, i_n}\}\) of sets of global states. Then

\[
\{S_{i_1, \ldots, i_n} \subseteq S_{i_1, \ldots, i_n}^{\prime \prime}\}\]

whenever

\[
R_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \subseteq R_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G).
\]

\(\square\)

To prove it we can use the same idea as in Propositions 4.10.1 and 4.10.2.

It is also possible to check inclusion if the domain of the optional parameter in a representation of the would-be superset is finite. Then we can obtain the set difference (Section 4.8) between the would-be subset and would-be superset and check if the result is empty (Section 4.9).

### 4.11 Parallel Application

Here we introduce parallel application and aggregate parallel application (first introduced in Section 2.6) — basic operations on disjunctive relations
required to perform reachability analysis. For simplicity reasons, in this section we consider these operations only on disjunctive relations, but it should still be enough to understand fundamentals. The operations on environment disjunctive relations will be considered in Chapter 5 in terms of the respective operations on symbolic encodings of environment disjunctive relations—so-called system descriptive formulas.

First we present the aggregate parallel application of an unsynchronized transition, then the parallel application of a synchronized transition.

**Unsynchronized transitions**

**Proposition 4.11.1** Assume that \( \tau \) is an unsynchronized transition described by the normal form transition schema

\[
\bigvee_{0 \leq i < n} (u(I, X[I], X'[I]) \land J(I))
\]

and \( R(I, X) \) is a disjunctive relation describing the set \([R(I, X)]\) of global states. Let \( R'(I, X) \) be the least disjunctive relation such that

- \( R'(i, a) \) whenever \( R(i, a) \)
- \( R'(i, a') \) whenever \( R(i, a) \) and \( u(i, a, a') \) holds for some \( a \) and \( a' \).

Then \([R'(I, X)] = \tau^#([R(I, X)])\). \(\square\)

**Proof:** (\(\subseteq\)) Let us first prove that \([R'(I, X)] \subseteq \tau^#([R(I, X)])\). For any \( s' \in [R'(I, X)] \) such that \( s' = \{X[0] \mapsto a'_0, \ldots, X[n-1] \mapsto a'_{n-1}\} \) we have \( \{(0, a'_0), \ldots, (n-1, a'_{n-1})\} \subseteq R'(I, X) \) by the semantics of the disjunctive relation. From the definition of \( R'(I, X) \) it follows that for every \( a'_i \) where \( 0 \leq i < n \) there exists \( a_i \) such that \( R(i, a_i) \) and either \( a_i = a'_i \) or \( u(i, a_i, a'_i) \) holds. Since \( R(i, a_i) \) for \( 0 \leq i < n \), we have \( s \in [R(I, X)] \) where \( s = \{X[0] \mapsto a_0, \ldots, X[n-1] \mapsto a_{n-1}\} \) by the semantics of the disjunctive relation. Then, by the definition of the aggregate parallel application, \( s' \in \tau^#([R(I, X)]) \) because there exists a sequence of states \((s, s_1), \ldots, (s_{n-1}, s')\) where each transition \((s_i, s_{i+1})\) for \( 0 \leq i < n \) is either idle if \( a_i = a'_i \) or \( u(i, a_i, a'_i) \) holds. Hence, \([R'(I, X)] \subseteq \tau^#([R(I, X)])\) because for any global states \( s \) we have \( s \in [R'(I, X)] \) whenever \( s \in [R'(I, X)] \).

(\(\supseteq\)) Let us then prove that \([R'(I, X)] \supseteq \tau^#([R(I, X)])\). If \( s' \in \tau^#([R(I, X)]) \) where \( s' = \{X[0] \mapsto a'_0, \ldots, X[n-1] \mapsto a'_{n-1}\} \) then by the definition of the aggregate parallel application either (1) \( s' \in [R(I, X)] \) or (2) there exists a state \( s = \{X[0] \mapsto a_0, \ldots, X[n-1] \mapsto a_{n-1}\} \) such that \( s \in [R(I, X)] \) and a non-empty sequence \((s, s_1), \ldots, (s_{m-1}, s')\) of atomic transitions such that

- \((s_i, s_{i+1}) \in \tau\) for \( 1 \leq i < m \), and
CHAPTER 4. INTERMEDIATE REPRESENTATION

Figure 4.9: A visualization of a disjunctive relations $R(I, X)$ and $R'(I, X)$ from Example 4.11.2 such that $R'(I, X)$ is the representation of the result of aggregate parallel application of $\tau$ the the set of states represented by $R(I, X)$.

- every transition $(s, s_i)$ in the sequence is either idle ($s_i = s_{i+1}$), or $s_{i+1} \in \tau(s_i)$ and its moving component is different from the moving components of all previous non-idle transitions in the sequence, and
- at least one transition in the sequence is non-idle.

In the first case, $s' \in [R'(I, X)]$ because $\{(0, a'_0), \ldots, (n-1, a'_{n-1})\} \subseteq R(I, X)$ (since $s' \in [R(I, X)]$) and $R'(i, a)$ whenever $R(i, x)$ by the definition of $R'(I, X)$.

In the second case, $\{(0, a_0), \ldots, (n-1, a_{n-1})\} \subseteq R(I, X)$ since $s \in [R(I, X)]$ and, consequently, by the definition of $R'(I, X)$, $\{(0, a_0), \ldots, (n-1, a_{n-1})\} \subseteq R'(I, X)$. From the definition of the aggregate parallel application it follows that for each component it could be at most one atomic transition in the sequence such that that component is the moving one. If there is such an atomic transition for a component $0 \leq i < n$ then $u(i, a_i, a'_i)$ holds, or $a_i = a'_i$ otherwise. Hence, $\{(0, a'_0), \ldots, (n-1, a'_{n-1})\} \subseteq R'(I, X)$ by the definition of $R'(I, X)$ and $s' \in [R'(I, X)]$ by the semantics of the disjunctive relation. Therefore, $[R'(I, X)] \supseteq \tau#([R(I, X)])$.

Example 4.11.2 Assume that we have a disjunctive set $S$ of global states in a system of size $n = 5$ which is represented by the disjunctive relation $R(I, X)$ visualized in Figure 4.9 (on the left). Assume further that we have an unsynchronized transition $\tau$ described by the following normalized transition schema:

$$\bigvee_{0 \leq i < 5} \left( X[i] = 1 \land X'[i] = 2 \land \mathcal{J}(I) \right).$$

Then the set $S' = \tau#(S)$ is represented by the disjunctive relation $R'(I, X)$ shown on the right in Figure 4.9.
There is no easy way to perform the parallel application of an unsynchronized transition. One may try to perform the aggregate parallel application and then use set difference but the set difference operation has its limitations as explained above.

**Synchronized transitions**

The result of the parallel application or the aggregate parallel application of a synchronized transition on a disjunctive set of global states is in general not a disjunctive set.

**Example 4.11.3** Assume that we have a disjunctive set $S$ of global states in a system of size $n = 3$ which is represented by the disjunctive relation $R(I, X)$ visualized in Figure 4.10 (on the left). Assume further that we have a synchronized transition $\tau$ described by the following normalized transition schema:

$$\bigvee_{I=1} \left[ \bigwedge_{0 \leq G < I, G \neq I} \left( 0 \leq X[I] \leq 1 \land X'[I] = X[I] + 1 \land (G < I \rightarrow X[G] = X[I]) \right) \right].$$

This meta-transition describes a set of atomic transitions, where process with index $I = 1$ can make a move from any local state $X[I]$ to $X[I] + 1$ under condition that all processes with smaller indexes $G$ (in our case it is process with index $G = 0$) are in the same local state as the current local state of the moving component ($X[G] = X[I]$). In particular, if process 1 makes a move from $\{X \mapsto 0\}$ to $\{X \mapsto 1\}$, then process 0 must be (only) in local state $\{X \mapsto 0\}$, while if process 1 moves from $\{X \mapsto 1\}$ to $\{X \mapsto 2\}$, then process 0 must be only in local state $\{X \mapsto 1\}$.

The non-disjunctive set $S' = \tau(S)$ is represented by the collection of two disjunctive relations $R_1(I, X)$ and $R_2(I, X)$ visualized in Figure 4.10. □
CHAPTER 4. INTERMEDIATE REPRESENTATION

Figure 4.11: A visualization of the disjunctive relation \( R(I, X) \) and the parameterized disjunctive relation \( R'(I, X, J, X', X'') \) from Example 4.11.4 such that \( [R'(I, X, J, X', X'')] = \tau^s([R(I, X, J, X', X'')]) \). A triple at the left from points are the values of the collection parameter.

As we can notice in Example 4.11.3, the resulting disjunctive relations \( R_1(I, X) \) and \( R_2(I, X) \) can be considered as the results of the application of particular atomic transitions from \( \tau \) to individual global states. The following example utilizes this idea and provides the basis for Proposition 4.11.5 presented thereafter.

Example 4.11.4 Let us re-consider the disjunctive set \( S \) of global states represented by the disjunctive relation \( R(I, X) \) and synchronized transition \( \tau \) from Example 4.11.3. Meta-transition \( \tau \) contains only two atomic transitions which are applicable to \( S \):

\[
(\{X[0] \mapsto 0, X[1] \mapsto 0, X[2] \mapsto 2\}, \{X[0] \mapsto 0, X[1] \mapsto 1, X[2] \mapsto 2\})
\]

and

\[
(\{X[0] \mapsto 1, X[1] \mapsto 1, X[2] \mapsto 2\}, \{X[0] \mapsto 1, X[1] \mapsto 2, X[2] \mapsto 2\}).
\]

Application of atomic transitions w.r.t. a non-moving component is uniquely identified by

- the index of the moving component,
- the current local state of the moving component, and
- the next local state of the moving component.

Therefore, a stamp of those values can serve as the collection parameter. In Figure 4.11 we can see a visualization of the parameterized disjunctive relation \( R'(I, X, J, X', X'') \) such that \( [R'(I, X, J, X', X'')] = \tau^s(S) \) which is created in this way.

Although we can safely drop the index of the moving components from the stamp in this example (because it is the same for all atomic transitions), we decided to keep it for the sake of generality. □
The idea from Example 4.11.4 is formalized in the following proposition.

**Proposition 4.11.5** Assume that \( \tau \) is a synchronized transition described by the following transition schema in the normal form

\[
\bigvee_{0 \leq I < n} \bigwedge_{0 \leq G < n} G \neq I \ (\text{syn}(I, X[I], X'[I], G, X[G]) \wedge J(I))
\]

and \( R(I, X) \) is a disjunctive relation describing a set \([R(I, X)]\) of global states. Let \( R'(I, X, J, X', X'') \) be the least parameterized disjunctive relation with collection parameters \( J, X' \) and \( X'' \) such that

1. \( R'(i, b, j, a, a') \) whenever \( R(i, b) \) and \( \text{syn}(i, b, j, a, a') \) holds for some \( j, a, a', i, b \) where \( j \neq i \), and
2. \( R'(j, a', j, a, a') \) whenever \( R(j, a) \) and \( \text{syn}(j, a, j, a, a') \) holds for some \( j, a \) and \( a' \).

Then \([R'(I, X, J, X', X'')]] = \tau[\llbracket R(I, X) \rrbracket] \). \( \square \)

What we get in the result in Proposition 4.11.5 is a collection of disjunctive relation, where the collection parameter consists of the index of the moving component and its local transition. For each value of the collection parameter there is a disjunctive relation such that it contains all the points representing local states of non-moving components which satisfy the guard (case (1)) and the point corresponding to the new local state of the moving component (case (2)). Proposition 4.11.5 can be easily proved if one understands the idea in Example 4.11.4.
Chapter 5

System Descriptive Formulas

In this chapter we show how the intermediate representation and operations on it can be mapped to the domain of Presburger formulas with standard operations.

First, in Section 5.1 we introduce system descriptive formulas and explain their relation with state schemas and disjunctive relations. Section 5.2 extends system descriptive formulas with parameters, which, in particular, allows us to encode environment disjunctive relations. In Section 5.3 we present an overview of how to perform basic set operations on sets of global states encoded as system descriptive formulas. Section 5.4 deals with operations on the system descriptive formulas required to perform reachability analysis. The property of a set (or a family of sets) of global states to be encoded as a system descriptive formula is called the definability property and its preservation under the introduced operations is discussed in Section 5.6.

5.1 System Descriptive Formulas and State Schemas

To encode a disjunctive relation, we use a system descriptive formula (SDF) $f$, which is a Presburger formula built upon a variable $I$ ranging over indexes and a variable $X$ ranging over values of local states of a component. The semantics $[[f(I,X)]]$ of an SDF $f(I,X)$ is the disjunctive relation $R(I,X)$, such that $R(i,a)$ holds iff $f(i,a)$ is true. Since the semantics of a disjunctive relation $R(I,X)$, in turn, is a set of global states $[R(I,X)]$, we will use the notation $[[[f(I,X)]]] = [R(I,X)]$ if $[f(I,X)] = R(I,X)$, and, for the sake of brevity, we often say directly that a system descriptive formula $f(I,X)$ encodes a set of global states $[[f(I,X)]]$. 
Example 5.1.1 If we have the following SDF $f(I, X)$ in the system of size $n = 4$:

$$f : (I = 0 \land X = 1) \lor (1 \leq I \leq 2 \land (X = 0 \lor X = 4)) \lor (I = 3 \land X = 5)$$

then the corresponding disjunctive relation is

$$R(I, X) = \{(0, 1), (1, 0), (1, 4), (2, 0), (2, 4), (3, 5)\}$$

A system descriptive formula $f(I, X)$ is called filtered iff $\llbracket f(I, X) \rrbracket$ is filtered. How to filter a formula is discussed in Section 5.3.

In Chapter 2 we have already introduced one formalism to model uniform finite state transition systems whereas system descriptive formulas described in this chapter will be used in the reachability analysis with over-approximation. The following theorem explains a correspondence between the system descriptive formulas and the state schemas and provides a practical guide for the syntactical transformation from one type of formula to the other.

Theorem 5.1.2 Let $f(I, X)$ be a quantifier-free Presburger formula such that $I \in I$ and $X \in A$, and let $f(I, X[I])$ be $f(I, X)$ with all occurrences of $X$ replaced by $X[I]$. Then

$$\llbracket \llbracket I \rrbracket \rrbracket = \llbracket \bigwedge_{0 \leq I < n} f(I, X[I]) \rrbracket$$

where $\llbracket f(I, X) \rrbracket$ is a disjunctive relation, and

$$\bigwedge_{0 \leq I < n} f(I, X[I])$$

is a state schema in the normal form.

Proof: Let us prove that

$$\llbracket \llbracket f(I, X) \rrbracket \rrbracket = \llbracket \bigwedge_{0 \leq I < n} f(I, X[I]) \rrbracket$$

($\supseteq$) Let us assume that there exists a global state $s = \{X[0] \mapsto x_0, \ldots, X[n - 1] \mapsto x_{n - 1}\}$ such that $s \in \bigwedge_{0 \leq I < n} f(I, X[I])$ but $s \notin \llbracket f(I, X) \rrbracket$. After we de-sugar the state schema

$$\bigwedge_{0 \leq I < n} f(I, X[I]) = f(0, X[0]) \land \ldots \land f(n - 1, X[n - 1])$$

and apply the assignment $s$, the resulting formula

$$f(0, x_0) \land \ldots \land f(n - 1, x_{n - 1})$$
must evaluate to true. Obviously, every conjunct \( f(i, x_i) \) must be true.

On the other hand, if \( s \not\in [[f(I, X)]] \) then by definition there must exist at least one pair \((i, x_i)\) such that \( f(i, x_i) \) does not hold, contradicting the condition above, therefore \( s \in [[f(I, X)]] \).

(⊆) Assume that there exists a global state \( s = \{X[0] \mapsto x_0, \ldots, X[n-1] \mapsto x_{n-1}\} \) such that \( s \not\in [[f(I, X)]] \) but \( s \not\in [[\bigwedge_{0 \leq I < n} f(I, X[I])] \]. By definition, \( \{(0, x_0), \ldots, (n-1, x_{n-1})\} \subseteq [[f(I, X)]] \), therefore \( f(i, x_i) \) holds for every \((i, x_i)\) where \( 0 \leq i < n \) and

\[
  f(0, x_0) \land \ldots \land f(n-1, x_{n-1})
\]
is true.

On the other hand, if \( s \not\in [[\bigwedge_{0 \leq I < n} f(I, X[I])] \), then under assignment \( s \) the desugared state schema

\[
  f(0, x_0) \land \ldots \land f(n-1, x_{n-1})
\]
is false, contradicting our previous conclusion. Therefore \( s \in [[\bigwedge_{0 \leq I < n} f(I, X[I])]] \) whenever \( s \in [[f(I, X)]] \).

We will use the sign “\( \models \)” whenever a formula on one side is a state schema and a formula on the other side is a system descriptive formula but both represent the same set of global states, that is, \( F \models f \) iff \( [F] = [[f]] \), where \( F \) is a state schema and \( f \) is a system descriptive formula.

**Example 5.1.3** Assume that we have \( N = 20 \) processes and wish to describe the set \( S = \{\{X[0] \mapsto 0, \ldots, X[9] \mapsto 0, X[10] \mapsto 2, \ldots, X[19] \mapsto 2\}\} \) of one global state as a system descriptive formula. We may first construct the following state schema

\[
  \bigwedge_{0 \leq I < 10} X[I] = 0 \land \bigwedge_{10 \leq I < 20} X[I] = 2
\]
but it is not suitable for the application of Theorem 5.1.2, therefore we should reconstruct it to the applicable form to apply the transformation:

\[
  \bigwedge_{0 \leq I < 20} ((0 \leq I < 10) \rightarrow (X[I] = 0)) \land ((10 \leq I < 20) \rightarrow (X[I] = 2))
\]

\[
  \models ((0 \leq I < 10) \rightarrow (X = 0)) \land ((10 \leq I < 20) \rightarrow (X = 2))
\]
The semantics of the resulting system descriptive formula is the disjunctive relation \( R = \{(0, 0), \ldots, (9, 0), (10, 2), \ldots, (19, 2)\} \) such that \([R] = S\).

Note, that in the system descriptive formula it is implicitly assumed that \( I \) ranges over the set \( I = \{0, \ldots, 19\} \) of indexes. If the domain of \( I \) is neither clear nor follows from the context, we have to enforce it properly in the formula:

\[
  0 \leq I < 20 \land ((0 \leq I < 10) \rightarrow (X = 0)) \land ((10 \leq I < 20) \rightarrow (X = 2))
\]
CHAPTER 5. SYSTEM DESCRIPTIVE FORMULAS

5.2 System Descriptive Formulas with Parameters

A system descriptive formula \( f(I, X, P_1, \ldots, P_k) \) with parameters for some \( k \geq 1 \) is a system descriptive formula where the set of variables is extended with variables representing collection, size and family parameters of a disjunctive meta-relation introduced in Section 4.4. For the sake of brevity, we may omit mentioning of parameters and call it simply a system descriptive formula, when the presence of parameters is clear from the context. Note that a parameter \( P \) in the notation \( f(I, X, P) \) may be composite and represent a set \( P_1, \ldots, P_k \) of particular parameters.

The semantics \( [[[f(I, X, P)]]] \) of a SDF \( f(I, X, P) \) with parameters is a disjunctive meta-relation or a parameterized disjunctive relation depending on what kinds of parameters it involves. In particular,

- when at least one size or family parameter is present, then the meaning of an SDF \( f(I, X, P) \) with (a composite) parameter \( P \) involving at least one size or family parameter is a disjunctive meta-relation \( R(I, X, P) \), such that \( R(i, a, p) \) holds iff \( f(i, a, p) \) is true. For this case the notation \( [[[f(I, X, P)]]] \) stands for a family of sets of global states.

- If a parameterized system descriptive formula \( f(I, X, G) \) has only a (possibly composite) collection parameter \( G \), then its semantics is a parameterized disjunctive relation \( R(I, X, G) \) such that \( R(i, a, g) \) holds iff \( f(i, a, g) \) is true. The notation \( [[[f(I, X, G)]]] \) denotes a set of global states in that case.

Clearly, the types of parameters used in a system descriptive formula must be always known from the context or should be stated explicitly in order to determine its semantics.

A parameterized system descriptive formula \( f(I, X, P, N) \) with size parameter \( N \) and possibly some other parameter(s) \( P \) is called filtered iff \( [[[f(I, X, P, N)]]] \) is filtered.

In particular, we will use system descriptive formulas with parameters to encode environment disjunctive relations (introduced in Section 4.5). The semantics of a system descriptive formula \( f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) with parameters \( I_1, X_1, \ldots, I_k, X_k, N, G \) is the environment disjunctive relation \( [[[f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)]]] \) which represents the family \( [[[f(i_1, X_1, \ldots, i_k, X_k, I, X, n, G)]]] \) of sets of global states for \( n \in N \) and \( i_1, \ldots, i_k \in I \).

5.3 Set Operations on SDFs

We next present the basic set operations discussed in Sections 4.6-10 as operations on system descriptive formulas. All the propositions below follow directly from the respective definitions for the intermediate representation.
CHAPTER 5. SYSTEM DESCRIPTIVE FORMULAS

Set Intersection

The following proposition for set intersection follows directly from Proposition 4.6.6.

Proposition 5.3.1 Let \( f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G_1) \) and \( f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G_2) \) be two system descriptive formulas. Then

\[
\llbracket [f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G_1)] \cap [f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G_2)] \rrbracket = \llbracket [f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G_1)] \land [f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G_2)] \rrbracket.
\]

\( \Box \)

Set Union

First we present a way of performing set union based on Proposition 4.7.3 by adding an additional collection parameter to the formula.

Proposition 5.3.2 Let \( f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) and \( f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) be two system descriptive formulas. Then

\[
\llbracket [f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] \cup [f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] \rrbracket = \llbracket (G' = 1 \land f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)) \lor (G' = 2 \land f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)) \rrbracket.
\]

\( \Box \)

In order to limit the growth of the number of parameters, we can use the following proposition by the cost of additional over-approximation which is based on Proposition 4.7.5:

Proposition 5.3.3 Let \( f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) and \( f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) be two system descriptive formulas. Then

\[
\llbracket [f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] \cup [f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] \rrbracket \subseteq \llbracket [f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] \lor [f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)] \rrbracket.
\]

\( \Box \)
CHAPTER 5. SYSTEM DESCRIPTIVE FORMULAS

Set Difference

Set difference is probably the most difficult set operation on the intermediate representation and, in consequence, on SDFs. As it is explained in Section 4.8, it is decidable only if the collection parameter $G$ has finite domain. The following Proposition is based on Proposition 4.8.3:

**Proposition 5.3.4** Let $f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ and $f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ be two system descriptive formulas and $\{g_1, \ldots, g_m\}$ be the finite domain of the parameter $G$ in $f_2$. Then

$$\left[\left[ f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \right] \right] \setminus \left[\left[ f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \right] \right] = \left[\left[ \begin{array}{l} 0 \leq G'_1 < N \land f_1(I_1, X_1, \ldots, I_k, X_k, I, X, G_1) \\
\vdots \\
0 \leq G'_m < N \land f_1(I_1, X_1, \ldots, I_k, X_k, I, X, G_m) \\
\end{array} \right] \land \left[ f_2(I_1, X_1, \ldots, I_k, X_k, I, X, G_1) \Rightarrow (I = G'_1) \right] \\
\vdots \\
\left[ f_2(I_1, X_1, \ldots, I_k, X_k, I, X, G_m) \Rightarrow (I = G'_m) \right] \right]$$

Set Complement

Set complement is considered as a special case of set difference and is based on Proposition 4.8.4.

**Proposition 5.3.5** Let $f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)$ be a system descriptive formula and $\{g_1, \ldots, g_m\}$ be the finite domain of the parameter $G$. Then

$$\left[\left[ f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \right] \right] = \left[\left[ \begin{array}{l} 0 \leq G'_1 < N \land [f(I_1, X_1, \ldots, I_k, X_k, I, X, g_1) \Rightarrow (I = G'_1)] \\
\vdots \\
0 \leq G'_m < N \land [f(I_1, X_1, \ldots, I_k, X_k, I, X, g_m) \Rightarrow (I = G'_m)] \\
\end{array} \right] \land \left[ \text{complement of } \left[\left[ f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \right] \right] \right] \right]$$

where $\left[\left[ f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \right] \right]$ is the complement of $\left[\left[ f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \right] \right]$. \qed
CHAPTER 5. SYSTEM DESCRIPTIVE FORMULAS

Filtering

In order to apply some set operations like the test for the set emptiness or entailment we need to make sure that the input formulas are filtered. The following proposition follows directly from Proposition 4.9.4 for the intermediate representation.

**Proposition 5.3.6** Let \( f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) be a system descriptive formula. Then the system descriptive formula

\[
f'(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \equiv f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \land \forall 0 \leq I < N \exists X f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)
\]

is filtered and

\[
[[f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)]] = [[[f'(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)]]].
\]

\[\square\]

Emptiness Test

The following propositions is based on Proposition 4.9.3.

**Proposition 5.3.7** Let \( f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) be a filtered system descriptive formula. Then it encodes the family \( \{S_i_1, \ldots, i_k, n\} = \{\emptyset\} \) of empty sets for every value of the parameters \( I_1, \ldots, I_k, N \) and \( G \) iff

\[
\neg f(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)
\]

is valid.

\[\square\]

Inclusion Test

As it was discussed in Section 4.10 we do not have a technique to check for inclusion for the general case. In this work we use results from Proposition 4.10.4 to test for set inclusion.

**Proposition 5.3.8** Let \( f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) and \( f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) be two filtered system descriptive formulas. Then

\[
[[f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)]] \subseteq [[[f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G)]]]
\]

if \( f_1(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \rightarrow f_2(I_1, X_1, \ldots, I_k, X_k, I, X, N, G) \) is valid.

\[\square\]

Note, that Proposition 5.3.8 describes just a sufficient condition for set inclusion.
5.4 Reachability Operations on SDFs

In this section we concentrate on the reachability operations on system descriptive formulas — parallel application and aggregate parallel application which are used to perform reachability analysis for systems with environmental relaxation presented in Chapter 3.

For the sake of simplicity, let us consider only a single system in this section. The transition from the reasoning about one system into the reasoning about a possibly infinite family of systems can be done by the addition of the size parameter $N$ to the respective system descriptive formulas without alteration of the main concepts.

First we start with fundamentals and describe how to perform aggregate parallel application of an unsynchronized transition on a disjunctive set of global states encoded by an SDF. Although we will not work directly on SDFs describing disjunctive relations but rather environment disjunctive relations, it may be easier for the reader to start with simpler theorems and proofs before proceeding with more complex notions — reachability operations on system descriptive formulas with parameters encoding environment disjunctive relations.

Afterwards we consider the parallel or aggregate parallel application of synchronized and unsynchronized transitions on sets of global states represented by environment disjunctive relations and encoded as system descriptive formulas.

Unsynchronized Transitions and Disjunctive Sets

If we have a normalized transition schema $\rho$ 

$\bigvee_{0 \leq I < n} [ u(I, X[I], X'[I]) \wedge J(I) ]$

describing an unsynchronized transition and a system descriptive formula $f(I, X)$ describing a set of global states then the aggregate parallel application of the unsynchronized transition in the domain of system descriptive formulas is an SDF denoted $apaUnsynch(u(I, X, X'), f(I, X))$ \footnote{The acronym $apa$ stands for the aggregate parallel application in the symbolic name of reachability operation.} defined as follows: $f(I, X) \vee \exists X'. [f(I, X') \wedge (u(I, X', X))]$. As we can see, the formula is quite simple which makes the application efficient in the SDF domain.

Let us now introduce two propositions which help later to prove that the application of a transition in the SDF domain is equivalent to its application in the real system.

Proposition 5.4.1 Assume, that we have a normalized transition schema $\rho$ 

$\bigvee_{0 \leq I < n} [ u(I, X[I], X'[I]) \wedge J(I) ]$
describing an unsynchronized transition \([\rho]\). If there is an atomic non-idling transition \((\{X[0] \mapsto x_0, \ldots, X[m] \mapsto x_m, \ldots, X[n-1] \mapsto x_{n-1}\}, \{X[0] \mapsto x_0, \ldots, X[m] \mapsto x_m, \ldots, X[n-1] \mapsto x_{n-1}\}) \in [\rho]\) where \(m\) is the moving component, then \(u(m, x_m, x'_m)\) must hold.

**Proposition 5.4.2** Assume, that we have a normalized transition schema \(\rho\)
\[
\bigvee_{0 \leq i < n} [u(I, X[I], X'[I]) \land J(I)]
\]
describing an unsynchronized transition \([\rho]\). If \(u(m, x_m, x'_m)\) holds for some \(m, x_m, x'_m\), then there exists an unsynchronized transition \(\tau \subseteq [\rho]\) which can be described by the transition formula \(\rho'\)
\[
X[m] = x_m \land X'[m] = x'_m \land J(i)
\]
where \(m\) is the moving component.

**Theorem 5.4.3** Assume, that we have a normalized transition schema \(\rho\)
\[
\bigvee_{0 \leq i < n} [u(I, X[I], X'[I]) \land J(I)]
\]
describing an unsynchronized transition \(\tau = [\rho]\), and that \(f(I, X)\) is a system descriptive formula describing some disjunctive set \([f(I, X)]\) of global states. Then
\[
[\text{apaUnsynch}(u(I, X, X'), f(I, X))] = \tau^\#([f(I, X)]).
\]

\[
\begin{proof}
(\supseteq) Let us assume that there is a global state \(s' \in \tau^\#([f(I, X)])\) where \(s' = (X[0] \mapsto x'_0, \ldots, X[n-1] \mapsto x'_{n-1})\). By definition, \(\tau^\#([f(I, X)]) = [f(I, X)] \cup \tau([f(I, X)])\) therefore either \(s'' \in [f(I, X)]\) or \(s'' \in [\tau([f(I, X)])]\). Let us consider these two cases:
\begin{itemize}
    \item If \(s'' \in [f(I, X)]\) then \((0, x'_0, \ldots, n-1, x'_{n-1})\) \(\subseteq [f(I, X)]\). Then \(\{0, x'_0, \ldots, n-1, x'_{n-1}\} \subseteq \text{apaUnsynch}(u(I, X, X'), f(I, X))\) and \(s' \in [\tau([f(I, X)])]\) because by the definition of \(\text{apaUnsynch}(u(I, X, X'), f(I, X))\) all the solutions of \(f(I, X)\) are the solution of \(\text{apaUnsynch}(u(I, X, X'), f(I, X))\).
    \item If \(s'' \in [\tau([f(I, X)])]\) then there exists a global state \(s_1 \in [f(I, X)]\), where \(s_1 = \{X[0] \mapsto x_{1,0}, \ldots, X[n-1] \mapsto x_{1,n-1}\}\), such that there exists a non-empty sequence of transitions \((s_1, s_2), \ldots, (s_m, s')\) where
\[
\begin{align*}
(s_1, s_2) &= ((X[0] \mapsto x_{1,0}, \ldots, X[n-1] \mapsto x_{1,n-1}), \\
(X[0] \mapsto x_{2,0}, \ldots, X[n-1] \mapsto x_{2,n-1}))
\end{align*}
\]
\end{itemize}
\end{proof}

74
CHAPTER 5. SYSTEM DESCRIPTIVE FORMULAS

\[
(s_m, s') = (\{X[0] \mapsto x_{m, 0}, \ldots, X[n - 1] \mapsto x_{m, n - 1}\},
\{X[0] \mapsto x_0', \ldots, X[n - 1] \mapsto x_{n - 1}'\})
\]

and \((s_i, s_{i+1}) \in \tau\) for any \(1 \leq i \leq m\). Since \(\tau = [\rho]\), the transition schema \(\rho\) has the following solutions:

\[
\{X[0] \mapsto x_{1, 0}, \ldots, X[n - 1] \mapsto x_{1, n - 1}, X'[0] \mapsto x_{2, 0}, \ldots, X'[n - 1] \mapsto x_{2, n - 1}\},
\]

\[
\{X[0] \mapsto x_{m, 0}, \ldots, X[n - 1] \mapsto x_{m, n - 1}, X'[0] \mapsto x_0', \ldots, X'[n - 1] \mapsto x_{n - 1}'\}.
\]

The definition also states, that the moving components of all non-idling transitions in the sequence are different. Therefore, there is a set \(I_m = \{i_1, \ldots, i_k\}\) of the components which make exactly one local transition \((X[i], X'[i])\), \(i \in I_m\) during the sequence of atomic transitions. Thus, for each \(0 \leq i < n\) in the states \(s_1 = \{X[0] \mapsto x_{1, 0}, \ldots, X[n - 1] \mapsto x_{1, n - 1}\}\) and \(s' = \{X[0] \mapsto x_{0}', \ldots, X[n - 1] \mapsto x_{n - 1}'\}\) we have

- \(x_{1,i} = x_i'\) whenever \(i \notin I_m\), or
- \(i \in I_m\) and there exists non-idling transition \((s_i, s_{i+1}) \in \tau\) in the sequence where the moving component makes a local transition \(((X[i] \mapsto x_{1, i}), (X'[i] \mapsto x_i'))\).

From the latter fact and Proposition 5.4.1 follows, that \(u(i, x_{1,i}, x_i')\) must hold for every \(i \in I_m\).

Since \((\{0, x_{1,0}, \ldots, (n-1), x_{1,n-1}\}) \subseteq [f(I, X)]\) (because \(s_1 \in [\tau](f(I, X)]\)) and for each \(0 \leq i < n\) either \(x_{1,i} = x_i'\) or \(u(i, x_{1,i}, x_i')\) holds, by the definition of \(\text{apaUnsynch}(u(I, X, X'), f(I, X))\) we have \(\{(0, x_0', \ldots, n-1, x_{n-1}')\} \subseteq [f'(I, X)]\), therefore \(s' \in [\tau''](f(I, X)]\).

(C) Let us assume that there exists a global state \(s'\) belonging to the set \([\text{apaUnsynch}(u(I, X, X'), f(I, X)]\) such that \(s' \notin \tau''([f(I, X)]\)). If \(s' \in [\text{apaUnsynch}(u(I, X, X'), f(I, X))]\), where \(s' = \{X[0] \mapsto x_0', \ldots, X[n - 1] \mapsto x_{n - 1}'\}\), then, by definition of the disjunctive relation, we have \(\{(0, x_0'), \ldots, (n - 1, x_{n - 1}')\} \subseteq [\text{apaUnsynch}(u(I, X, X'), f(I, X))\]. Then, from the definition of \(\text{apaUnsynch}(u(I, X, X'), f(I, X))\) there exists a subset \(\{(0, x_0, \ldots, (n - 1, x_{n - 1})\} \subseteq [f(I, X)]\) such that for any \(0 \leq i < n\) either \(x_{1,i} = x_i'\) or \(u(i, x_{1,i}, x_i')\) holds. In other words, there exists a global state \(s_1 = \{X[0] \mapsto x_{1,0}, \ldots, X[n - 1] \mapsto x_{1,n - 1}\}\). Let \(I_m\) be a subset of all components \(0 \leq i < n\) such that their local states are different in \(s'\) and \(s_1\): \(x_{1,i} \neq x_i'\). Then, \(u(i, x_{1,i}, x_i')\) must hold for any \(i \in I_m\) and, according to Proposition 5.4.2, there exists a set of unsynchronized transitions \(\tau_1 \subseteq \tau, \ldots, \tau_m \subseteq \tau\) which can be described by transition formulas of the form

\[
X[i] = x_{1,i} \land X'[i] = x_i' \land J(i)
\]

75
where $i \in I_m$. Let $(s_1, s_2, \ldots, (s_m, s'')$ be a sequence of atomic transitions such that $s_2 = \tau_1(s_1), \ldots, s'' = \tau_m(s_m)$. Obviously, the moving components of all transitions in the sequence will be different, therefore, by the definition of parallel and aggregate parallel application, $s'' \in \tau^\parallel([[f(I, X)])$ and $s'' \in \tau^\#([[f(I, X)])$. Since $s'' = \tau_m(\ldots(\tau_1(s_1))\ldots), we have $s'' = \{X'[0] \mapsto x''_0, \ldots, X'[n - 1] \mapsto x''_{n-1}\}$ and therefore $s' = s''$ and $s' \in \tau^\#([[f(I, X)])]$.  

**Example 5.4.4** Assume that $\rho$ is transition formula of the form

$$X[0] = 1 \land X'[0] = 3 \land X'[1] = X[1] \land \ldots \land X[7] = X'[7]$$

describing a meta-transition $\rho$ in a system of size $N = 8$ and that $f(I, X)$ is a system descriptive formula

$$((0 \leq I \leq 3) \rightarrow (X = 1 \lor X = 2)) \land ((4 \leq I \leq 7) \rightarrow X = 5)$$

describing some set of states $[[f(I, X)])]$. Let us express $[[\rho]$ as a transition schema in the normalized form first:

$$\bigvee_{I=0}^{I=3} \big[ X[I] = 1 \land X'[I] = 3 \land \mathcal{J}(i) \big]$$

Now we can apply Theorem 5.4.3 to get a system descriptive formula describing $\tau^\#([[f(I, X)])]$:

$$f'(I, X) = \exists X':((0 \leq I \leq 3) \rightarrow (X' = 1 \lor X' = 2)) \land ((4 \leq I \leq 7) \rightarrow X' = 5))$$

$$\land ((I = 0 \land X' = 1 \land X = 3) \lor (X = X'))$$

After quantifier elimination (which is always possible for Presburger arithmetic) the formula $f'(I, X)$ can be expressed as

$$(I = 0 \land (X = 1 \lor X = 2 \lor X = 3)) \lor ((1 \leq I \leq 3) \land (X = 1 \lor X = 2))$$

$$\lor (4 \leq I < 8 \land X = 5).$$

**Synchronized Transitions and Disjunctive Sets**

The problem with parallel application of synchronized transitions is that the resulting set of global states in many cases is not disjunctive, which is illustrated by the following example.
Example 5.4.5 Assume that we have a system of two processes, whose current set of states is described by the following state formula $F$

$$1 \leq X[0] \leq 2 \land 1 \leq X[1] \leq 2.$$ 

Let $\rho$ be a transition schema describing a synchronized transition $\tau = [\rho]$

$$\bigvee_{0 \leq I \leq 1} (X[I] = 1 \land X'[I] = 3 \land \bigwedge_{0 \leq G \leq 1 \land G \neq I} X[G] = 2 \land J(I)),$$

then the result $S = \tau^\parallel([F])$ of the parallel application can be described by the state formula

$$(X[0] = 3 \land X[1] = 2) \lor (X[0] = 2 \land X[1] = 3)$$

and the result $S = \tau^\#([F])$ of the aggregate parallel application can be described by the state formula

$$(X[0] = 3 \land X[1] = 2) \lor (X[0] = 2 \land X[1] = 3) \lor (1 \leq X[0] \leq 2 \land 1 \leq X[1] \leq 2).$$

Neither of the sets of global states described by these two state formulas is disjunctive.

In the remainder of the section we consider parallel or aggregate parallel applications of synchronized and unsynchronized transitions on system descriptive formulas parameterized by the reference processes for different types of transitions, so that it is possible to perform reachability analysis of a system with environmental relaxation.

Synchronized Transitions under Environmental Relaxation

As we presented in Section 3.4, the idea of environmental relaxation is the transformation of every original synchronized transitions $\tau$ of the transition relation into its relaxed versions $\tau|_\epsilon$ w.r.t. reference processes $\epsilon$. Recall from Section 3.4 that given a synchronized transition $\tau$ described by the normal form transition schema $\rho$

$$\bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right],$$

and the set $\epsilon$ of reference processes, the relaxed version $\tau|_\epsilon$ of $\tau$ can be described by the following transition schema $\rho|_\epsilon$:

$$\bigvee_{I \in \epsilon} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right].$$
CHAPTER 5. SYSTEM DESCRIPTIVE FORMULAS

\[ \bigvee_{I \in \epsilon} \left[ \bigwedge_{G \in \epsilon} \text{syn}(I, X[I], X'[I], G, X[G]) \land \mathcal{J}(I) \right]. \]

Two parts of the disjunction above correspond to two different cases based on the type of the moving component — whether it is a reference (case 1: \( I \in \epsilon \)) or an environment one (case 2: \( I \in \bar{\epsilon} \)). We say that a meta-transition is applied on reference or environment processes if all the moving components in non-idling atomic transitions of that meta-transition are reference or environment, respectively. In what follows we consider these two cases separately, because it will require different techniques.

Recall from Section 3.4 that in order to perform the environmental relaxation \( \omega = (k, \mathbf{I}_R, \text{range}(\mathbf{I}_R), \mathcal{E}) \) we need to relax synchronized transitions for many different reference sets \( \epsilon \in \mathcal{E} \) where the set \( \mathcal{E} \) of reference sets is defined in the following way:

\[ \mathcal{E} = \{i_1, \ldots, i_k \mid (i_1, \ldots, i_k) \in \text{range}(\mathbf{I}_R)\} \]

where \( \text{range}(\mathbf{I}_R) \) is some function on the set \( \mathbf{I}_R = \{I_1, \ldots, I_k\} \) of \( k \) reference parameters and \( k \) is the rank of environmental relaxation. For example, for a typical safety property about two arbitrary processes the set \( \mathcal{E} \) of reference sets might be defined as \( \{I_1, I_2 \mid 0 \leq I_1 < I_2 < n\} \) where \( n \) is the system size and \( \text{range}(I_1, I_2) = 0 \leq I_1 < I_2 < n \).

The system size \( n \) is a possibly unbounded parameter so it would be impossible to make relaxation for every reference set \( \epsilon \in \mathcal{E} \) one by one in the general case because the size of \( \mathcal{E} \) might be unbounded, too. Therefore, we use parameterization again. Having an environmental relaxation \( \omega = (k, \mathbf{I}_R, \text{range}(\mathbf{I}_R), \mathcal{E}) \) where the set \( \mathcal{E} \) of reference sets is defined by some function \( \text{range}(\mathbf{I}_R) \), and a synchronized transition \( \tau \) represented by a synchronized transition schema

\[ \rho = \bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land \mathcal{J}(I) \right], \]

the family \( \omega(\tau) = \{\tau_\epsilon\}_{\epsilon \in \mathcal{E}} \) of relaxed synchronized transitions parameterized by the choice of reference components can be expressed as the following transition schema:

\[ \text{range}(\mathbf{I}_R) \land \left( \bigvee_{I \in \mathbf{I}_R} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land \mathcal{J}(I) \right] \right) \]

\[ \bigvee_{0 \leq I < n \land I \notin \mathbf{I}_R} \left[ \bigwedge_{G \in \mathbf{I}_R} \text{syn}(I, X[I], X'[I], G, X[G]) \land \mathcal{J}(I) \right] \]
where $I \in I_R$ and $I \not\in I_R$ are shortcuts for $I = I_1 \lor \ldots \lor I = I_k$ and $I \neq I_1 \land \ldots \land I \neq I_k$, respectively. The transition schema above is denoted $\omega(\rho)$ and represents a family $\{\rho'_e\}_{e \in E}$ of relaxed synchronized transitions schemas after the environmental relaxation.

Let $\omega(\rho)$ be a transition schema

$$
\text{range}(I_R) \land \left( \bigvee_{I \in I_R} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right] \right)
$$

where $\omega = (k, I_R, \text{range}(I_R), E)$ is an environmental relaxation and $\rho$ is a synchronized transition schema. Then we define functions $\text{onRef}$ and $\text{onEnv}$ on $\rho$ in the following way:

$$
\text{onRef}(\omega, \rho) = \text{range}(I_R) \land
\left( \bigvee_{I \in I_R} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right] \right)
$$

and

$$
\text{onEnv}(\omega, \rho) = \text{range}(I_R) \land
\left( \bigvee_{0 \leq I < n \land I \not\in I_R} \left[ \bigwedge_{G \in I_R} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right] \right).
$$

The functions take a transition schema of a relaxed synchronized transition as the input and return such a part of the transition schema which is applicable to reference or environment components, respectively.

**Case 1: Application on Reference Processes**

Let us now show how to apply a synchronized transition on reference processes. We start with a simpler version ($\text{paSynchR}_{1,2}$) and then proceed to a more complex one ($\text{paSynchR}_{2,4}$) with a lower degree of over-approximation. The aim is to demonstrate that the framework is not rigid and allows for trade-offs between efficiency and completeness. That is, there is a choice to get the answer quicker though for the price of higher risk that the answer is “Do not know”.

2 The acronym $\text{pa}$ stands for the parallel application in the symbolic name of reachability operation, $\text{Synch}$ stands for the synchronized transition and $R$ stands for the reference set.
Let \( \tau \) be a synchronized transition represented by the synchronized transition schema

\[
\rho = \bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land \mathcal{J}(I) \right],
\]

\( \omega = (k, \mathbf{I}_R, \text{range}(\mathbf{I}_R), \mathcal{E}) \) be an environmental relaxation, and \( f(I_1, X_1, \ldots, I_k, X_k, I, X) \) be a system descriptive formula describing a set of global states. Then the parallel application of the synchronized transition in the domain of system descriptive formulas is a pSDF denoted \( \text{paSynch}_1(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \) and defined as follows:

\[
\exists X'_1 [ f(I_1, X'_1, \ldots, I_k, X'_k, I, X) \land \text{syn}(I_1, X'_1, I, X) ] \\
\lor \ldots \lor \\
\exists X'_k [ f(I_1, X_1, \ldots, I_k, X'_k, I, X) \land \text{syn}(I_k, X'_k, I, X) ]
\]

**Theorem 5.4.6** Assume, that \( \tau \) is a synchronized transition represented by the synchronized transition schema

\[
\rho = \bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land \mathcal{J}(I) \right],
\]

\( \omega = (k, \mathbf{I}_R, \text{range}(\mathbf{I}_R), \mathcal{E}) \) is an environmental relaxation, and \( f(I_1, X_1, \ldots, I_k, X_k, I, X) \) is a system descriptive formula describing a set of global states. Then

\[
\{ \tau | \epsilon \}_{\epsilon \in \mathcal{E}}(\mathcal{I}([f(I_1, X_1, \ldots, I_k, X_k, I, X)])) \\
\subseteq \\
\mathcal{I}(\text{paSynch}_1(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))).
\]

\( \Box \)

**Proof:** Assume to the contrary that

\[
\{ \tau | \epsilon \}_{\epsilon \in \mathcal{E}}(\mathcal{I}([f(I_1, X_1, \ldots, I_k, X_k, I, X)])) \\
\not\subseteq \\
\mathcal{I}(\text{paSynch}_1(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))).
\]

Then for some set \( \epsilon = \{ i_1, \ldots, i_k \} \in \mathcal{E} \) of reference processes there exists an atomic transition \( (s, s') \in \tau | \epsilon \) such that \( s \in \mathcal{I}([f(I_1, X_1, \ldots, I_k, X_k, I, X)]) \) and \( s' \in \{ \tau | \epsilon \}_{\epsilon \in \mathcal{E}}(\mathcal{I}([f(I_1, X_1, \ldots, I_k, X_k, I, X)])) \) but \( s' \not\in \mathcal{I}(\text{paSynch}_1(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))). \)

Let \( m \) be the moving component in \( (s, s') \) which changes its local state from \( x_m \) to \( x'_m \). Note, that \( m \) is one of the reference components \( i_1, \ldots, i_k \).
which follows from the definition of relaxation. From $s \in [[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]$ it follows that there exist $j_1, \ldots, j_m \in I$ and $x_{i_1}, \ldots, x_{i_k}$, $x_{j_1}, \ldots, x_{j_m} \in X$ such that

$$
X[i_1] \mapsto x_{i_1} \in s,
\ldots
X[m] \mapsto x_m \in s,
\ldots
X[i_k] \mapsto x_{i_k} \in s,
X[j_1] \mapsto x_{j_1} \in s,
\ldots
X[j_m] \mapsto x_{j_m} \in s
$$

and

$$f(i_1, x_{i_1}, \ldots, m, x_m, \ldots, i_k, x_{i_k}, j_1, x_{j_1}) \text{ is true}
\quad \text{and}
\ldots
\quad \text{and}
\quad \text{and}
$$

$$f(i_1, x_{i_1}, \ldots, m, x_m, \ldots, i_k, x_{i_k}, j_m, x_{j_m}) \text{ is true}$$

On the other side, from $(s, s') \in \tau_\epsilon$ it follows that

$$\text{syn}(m, x_m, x'_m, i_1, x_{i_1}) \text{ is true}
\quad \text{and}
\ldots
\quad \text{and}
\quad \text{and}
$$

$$\text{syn}(m, x_m, x'_m, i_k, x_{i_k}) \text{ is true}.$$

Based on these facts and on the definition of $\text{paSynchR}_1(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))$ we can conclude that

$$(i_1, x_{i_1}, \ldots, m, x_m, \ldots, i_k, x_{i_k}, j_1, x_{j_1}) \in [[\text{paSynchR}_1(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))]]$$

and

$$\ldots$$

and

$$(i_1, x_{i_1}, \ldots, m, x_m, \ldots, i_k, x_{i_k}, j_m, x_{j_m}) \in [[\text{paSynchR}_1(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))]].$$

Since $s' = \{X[i_1] \mapsto x_{i_1}, \ldots, X[m] \mapsto x'_m, \ldots, X[i_k] \mapsto x_{i_k}, X[j_1] \mapsto x_{j_1}, \ldots, X[j_m] \mapsto x_{j_m}\}$ (because it is the same as $s$ with the exception of the local state of the moving component), then $s' \in [[\text{paSynchR}_1(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))]]$ which contradicts the initial assumption. 

Let us now provide an improved definition of the parallel application of synchronized transition. Let $\tau$ be a synchronized transition represented by the synchronized transition schema

$$\rho = \bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right],$$
\[ \omega = (k, I_R, range(I_R), \mathcal{E}) \] be an environmental relaxation, and \( f(I_1, X_1, \ldots, I_k, X_k, I, X) \) be a system descriptive formula describing a set of global states. Then an improved version of parallel application of the synchronized transition in the domain of system descriptive formulas is a pSDF denoted \( \text{paSynchR}_2(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \) and defined as follows:

\[
\exists_{X'_1} \left[ f(I_1, X'_1, \ldots, I_k, X_k, I, X) \land \right.
\begin{align*}
syn(I_1, X'_1, X_1, I, X) \land \\
syn(I_1, X'_1, X_1, X_2, I) \land \ldots \land syn(I_1, X'_1, X_1, I_k, X_k) \right].
\]

The following theorem is an immediate consequence of Theorem 5.4.6 and Proposition 5.3.3.

**Theorem 5.4.7** Assume, that \( \tau \) is a synchronized transition represented by the synchronized transition schema

\[
\rho = \bigvee_{0 \leq i < n} \left( \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right),
\]

\( \omega = (k, I_R, range(I_R), \mathcal{E}) \) is an environmental relaxation, and \( f(I_1, X_1, \ldots, I_k, X_k, I, X) \) is a system descriptive formula describing a set of global states. Then

\[
\{ \tau \mid \epsilon \in \mathcal{E} \} \subseteq \left[ \left[ \text{paSynchR}_2(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \right] \right].
\]

Note, that the degree of the over-approximation achieved in Theorem 5.4.7 is generally lower than in Theorem 5.4.6 because of additional constraints in every disjunct in the definition of \( \text{paSynchR}_2 \).

At the end we show how to make the aggregate parallel application to be used in algorithms which based on the parallel applications introduced above. It is denoted as \( \text{apaSynchR}(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \) and defined as follows

\[
f(I_1, X_1, \ldots, I_k, X_k, I, X) \lor \text{paSynchR}(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)).
\]
Case 2: Application on Environment Processes

Let $\tau$ be a synchronized transition represented by the synchronized transition schema

$$\rho = \bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right],$$

$\omega = (k, I_R, \text{range}(I_R), E)$ be an environmental relaxation, and $f(I_1, X_1, \ldots, I_k, X_k, I, X)$ be a system descriptive formula describing a set of global states. Then the aggregate parallel application $\text{apaSynchE}(onEnv(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))$ of the synchronized transition in the domain of system descriptive formulas is an SDF defined as follows:

$$f(I_1, X_1, \ldots, I_k, X_k, I, X) \lor \exists X': \left( \left( f(I_1, X_1, \ldots, I_k, X_k, I, X') \land \text{syn}(I, X'[I], X[I], I_k, X_k) \right) \land \ldots \land \text{syn}(I, X'[I], X[I], I_k, X_k) \right).$$

In the symbolic name $\text{apaSynchE}$ the acronym $\text{apa}$ stands for the aggregate parallel application, $\text{Synch}$ stands for the synchronized transition and $E$ stands for the environment processes which are the target of the application.

**Theorem 5.4.8** Let $\tau$ be a synchronized transition represented by the synchronized transition schema

$$\rho = \bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land J(I) \right],$$

and $\omega = (k, I_R, \text{range}(I_R), E)$ be an environmental relaxation. Assume further that a system descriptive formula $f(I_1, X_1, \ldots, I_k, X_k, I, X)$ encodes an environment disjunctive relation describing a set of global states $[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]$. Then

$$\{\tau|_{e \in E}([[f(I_1, X_1, \ldots, I_k, X_k, I, X)]])) \subseteq [[\text{apaSynchE}(onEnv(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))]]$$

**Proof:** Assume to the contrary that

$$[[\text{apaSynchE}(onEnv(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))]] \nsubseteq \{\tau|_{e \in E}([[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]).$$

83
Then for some set \( \epsilon = \{ i_1, \ldots, i_k \} \in \mathcal{E} \) of reference processes there exists an atomic transition \((s, s') \in \tau|_r\) such that \( s \in [[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]] \) and \( s' \in [[\{ \tau|_r \} s \in \mathcal{E} [[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]]] \) but \( s' \not\in [[[apaSynchE(onEnv(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))]]]] \).

Let \( m \) be the moving component in \((s, s')\) which changes its local state from \( x_m \) to \( x'_m \). Note that \( m \) is one of the environment components \( j_1, \ldots, j_p \in \bar{\epsilon} \) which follows from the definition of relaxation. From \( s \in [[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]] \) it follows that there exist \( j_1, \ldots, j_p \in I \) and \( x_{i_1}, \ldots, x_{i_k}, x_{j_1}, \ldots, x_{j_p} \in X \) such that

\[
\begin{align*}
X[i_1] &\rightarrow x_{i_1} \in s, \\
\vdots & \\
X[i_k] &\rightarrow x_{i_k} \in s, \\
X[j_1] &\rightarrow x_{j_1} \in s, \\
\vdots & \\
X[m] &\rightarrow x_m \in s, \\
\vdots & \\
X[j_p] &\rightarrow x_{j_p} \in s
\end{align*}
\]

and

\[
\begin{align*}
f(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, j_1, x_{j_1}) & \text{ is true} \\
\text{and} & \\
\vdots & \\
f(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, m, x_m) & \text{ is true} \\
\text{and} & \\
\vdots & \\
f(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, j_p, x_{j_p}) & \text{ is true}
\end{align*}
\]

On the other side, from \((s, s') \in \tau|_r\) it follows that

\[
\begin{align*}
syn(m, x_m, x'_m, i_1, x_{i_1}) & \text{ is true} \\
\text{and} & \\
\vdots & \\
syn(m, x_m, x'_m, i_k, x_{i_k}) & \text{ is true}
\end{align*}
\]

Based on these facts and on the definition of \( apaSynchE(onEnv(\omega, \rho)) \),
f(I_1, X_1, \ldots, I_k, X_k, I, X)) we can conclude that

\[(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, j_1, x_{j_1}) \in [apaSynchE(onEnv(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))]\]

\[\ldots\]

\[(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, m, x'_m) \in [apaSynchE(onEnv(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))]\]

\[\ldots\]

\[(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, j_p, x_{j_p}) \in [apaSynchE(onEnv(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))].\]

Since \(s' = \{X[i_1] \mapsto x_{i_1}, \ldots, X[i_k] \mapsto x_{i_k}, X[j_1] \mapsto x_{j_1}, \ldots, X[m] \mapsto x'_m, \ldots, X[j_p] \mapsto x_{j_p}\}\) (because it is the same as \(s\) with the exception of the local state of the moving component), then \(s' \in [\{apaSynchE(onEnv(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))\}]\) which contradicts the initial assumption.

**Unsynchronized Transitions under Environmental Relaxation**

Although unsynchronized transitions are not changed during environmental relaxation, we still consider their application on reference and environment processes separately, because it requires different techniques as well. The difference is that it is possible to do perform a parallel application when only reference components are considered.

If an unsynchronized transition \(\tau\) is described by a transition schema

\[\rho \equiv \bigvee_{0 \leq I < n} [u(I, X[I], X'[I]) \land J(I)]\]

then for the purpose of environmental relaxation \(\omega = (k, I_R, range(I_R), E)\) we consider it as consisting of two parts:

\[range(I_R) \land \left( \bigvee_{I \in I_R} [u(I, X[I], X'[I]) \land J(I)] \right) \lor \left( \bigvee_{I \not\in I_R} [u(I, X[I], X'[I]) \land J(I)] \right)\]

depending whether the moving component is a reference one \((I \in I_R)\) or not \((I \not\in I_R)\) and apply them separately. Note, as was explained in Section 3.4, the unsynchronized transition is not changed by the environmental relaxation and therefore \(\omega(\rho) = \rho\) because \(\omega(\tau) = \tau\).
In the same way as we did with synchronized transition schemas we define functions $onRef$ and $onEnv$ on those transition schemas which describe unsynchronized transitions:

$$onRef(\omega, \rho) = \text{range}(I_R) \land \left( \bigvee_{I \in I_n} [u(I, X[I], X'[I]) \land J(I)] \right)$$

and

$$onEnv(\omega, \rho) = \text{range}(I_R) \land \left( \bigvee_{I \not\in I_R} [u(I, X[I], X'[I]) \land J(I)] \right)$$

where $\rho$ is a transition schema $\rho \equiv \bigvee_{0 \leq I < n} [u(I, X[I], X'[I]) \land J(I)]$ describing an unsynchronized transition and $\omega = (k, I_R, \text{range}(I_R), E)$ is an environmental relaxation. The functions return such a part of the transition schema under environmental relaxation which is applicable to reference or environment components, respectively.

Let us consider separately these two cases of applying unsynchronized transitions on environment and reference processes.

**Case 1: Application on Reference Processes**

Let $\tau$ be an unsynchronized transition represented by the unsynchronized transition schema $\rho \equiv \bigvee_{0 \leq I < n} [u(I, X[I], X'[I]) \land J(I)]$, $\omega = (k, I_R, \text{range}(I_R), E)$ be an environmental relaxation and $f(I_1, X_1, \ldots, I_k, X_k, I, X)$ be a system descriptive formula which describes a set of global states then the parallel application $paUnsynchR(onRef(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))$ of the unsynchronized transition in the domain of system descriptive formulas is an SDF defined as follows:

$$\exists X_1 \ldots \exists X_k \ [u(I_1, X_1', X_1) \land f(I_1, X_1', \ldots, I_k, X_k, I, X)]$$

The aggregate parallel application $apaUnsynchR(onRef(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))$ is defined as

$$f(I_1, X_1, \ldots, I_k, X_k, I, X) \lor paUnsynchR(onRef(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)).$$
Theorem 5.4.9 Let $\tau$ be an unsynchronized transition represented by the unsynchronized transition schema
\[
\rho \equiv \bigvee_{0 \leq I < n} \left[ u(I, X[I], X'[I]) \land J(I) \right],
\]
$\omega = (k, I_R, \text{range}(I_R), E)$ be an environmental relaxation and $f(I_1, X_1, \ldots, I_k, X_k, I, X)$ be a system descriptive formula which describes a set of global states $[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]$. Then
\[
\{ \tau \} \parallel \{ \omega \} \subseteq \left[ \text{paUnsynchR(onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \right].
\]

Proof: Assume to the contrary that $\{ \tau \} \parallel \{ \omega \} \not\subseteq \left[ \text{paUnsynchR(onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \right]$. Then for some set $\epsilon = \{ i_1, \ldots, i_k \} \in E$ of reference processes there exists an atomic transition $(s, s') \in \tau$ such that $s \in \left[ [f(I_1, X_1, \ldots, I_k, X_k, I, X)] \right]$ and $s' \not\in \left[ \text{paUnsynchR(onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \right]$. Let $m$ be the moving component in $(s, s')$ which changes its local state from $x_m$ to $x'_m$. Note, that $m$ is one of the reference components $i_1, \ldots, i_k$ which follows from the definition of relaxation. From $s \in \left[ [f(I_1, X_1, \ldots, I_k, X_k, I, X)] \right]$ it follows that there exist $j_1, \ldots, j_m \in \Gamma$ and $x_{i_1}, \ldots, x_{i_k}, x_{j_1}, \ldots, x_{j_m} \in X$ such that
\[
X[i_1] \mapsto x_{i_1} \in s,
\]
\[
\ldots
\]
\[
X[i_k] \mapsto x_{i_k} \in s,
\]
\[
X[j_1] \mapsto x_{j_1} \in s,
\]
\[
\ldots
\]
\[
X[j_m] \mapsto x_{j_m} \in s
\]
and
\[
f(i_1, x_{i_1}, \ldots, m, x_m, \ldots, i_k, x_{i_k}, j_1, x_{j_1}) \text{ is true}
\]
\[
\ldots
\]
\[
\text{and}
\]
\[
f(i_1, x_{i_1}, \ldots, m, x_m, \ldots, i_k, x_{i_k}, j_m, x_{j_m}) \text{ is true}
\]
On the other side, from \((s, s') \in \tau|_{\epsilon}\) it follows that \(u(m, x_m, x'_m, i_1, x_{i_1})\) is true.

Based on these facts and on the definition of \(\text{paUnsynchR(onRef(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))}\) we can conclude that

\[
(i_1, x_{i_1}, \ldots, m, x'_m, \ldots, i_k, x_{i_k}, j_1, x_{j_1}) \in \\
[\text{paUnsynchR(onRef(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))}] \\
\text{and} \\
\ldots \text{and} \\
(i_1, x_{i_1}, \ldots, m, x'_m, \ldots, i_k, x_{i_k}, j_m, x_{j_m}) \in \\
[\text{paUnsynchR(onRef(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))}].
\]

Since \(s' = \{X[i_1] \mapsto x_{i_1}, \ldots, X[m] \mapsto x'_m, \ldots, X[i_k] \mapsto x_{i_k}, X[j_1] \mapsto x_{j_1}, \ldots, X[j_m] \mapsto x_{j_m}\}\) (because it is the same as \(s\) with the exception of the local state of the moving component), then \(s' \in [[\text{paUnsynchR(onRef(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))}]])\) which contradicts the initial assumption. \(\square\)

The following theorem is the immediate consequence of Theorem 5.4.9 and Proposition 5.3.3.

**Theorem 5.4.10** Let \(\tau\) be an unsynchronized transition represented by the unsynchronized transition schema

\[
\rho \equiv \bigvee_{0 \leq I < n} [u(I, X[I], X'[I]) \wedge J(I)],
\]

\(\omega = (k, I_R, \text{range}(I_R), \mathcal{E})\) be an environmental relaxation and \(f(I_1, X_1, \ldots, I_k, X_k, I, X)\) be a system descriptive formula which describes a set of global states \([[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]])\]. Then

\[
\{\tau|_{\epsilon}\}_{\epsilon \in \mathcal{E}}([[f(I_1, X_1, \ldots, I_k, X_k, I, X)])] \\
\subseteq [[\text{apaUnsynchR(onRef(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))}]].
\]

The proof of Theorem 5.4.10 is trivial when taking Theorem 5.4.9 into account.

Note, that the degree of the over-approximation achieved in Theorem 5.4.10 is generally higher than in Theorem 5.4.9 and it is achieved because of an additional union of two sets of global states expressed as SDFs.

**Case 2: Application on Environment Processes**

Let \(\tau\) be an unsynchronized transition represented by the unsynchronized transition schema

\[
\rho \equiv \bigvee_{0 \leq I < n} [u(I, X[I], X'[I]) \wedge J(I)],
\]
\[ \omega = (k, I_R, \text{range}(I_R), \mathcal{E}) \] be an environmental relaxation and \( f(I_1, X_1, \ldots, I_k, X_k, I, X) \) be a system descriptive formula which describes a set of global states \([[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]] \). Then the parallel application \( \text{apaUnsynchE}(\text{onEnv}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \) of the unsynchronized transition in the domain of system descriptive formulas is an SDF defined as follows:

\[
f(I_1, X_1, \ldots, I_k, X_k, I, X) \ \cup \ \exists X' \ [ f(I_1, X_1, \ldots, I_k, X_k, I, X') \wedge u(I, X', X) ] .
\]

**Theorem 5.4.11** Let \( \tau \) be an unsynchronized transition represented by the unsynchronized transition schema

\[
\rho \equiv \bigvee_{0 \leq l < n} [ u(I, X[I], X'[I]) \wedge J(l) ] ,
\]

\( \omega = (k, I_R, \text{range}(I_R), \mathcal{E}) \) be an environmental relaxation and \( f(I_1, X_1, \ldots, I_k, X_k, I, X) \) be a system descriptive formula which describes a set of global states \([[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]] \). Then

\[
\{ \tau \mid \epsilon \in \mathcal{E} \} \# [[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]] = [[[\text{apaUnsynchE}(\text{onEnv}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))]]] .
\]

**Proof:** (\( \subseteq \)) Assume to the contrary that

\[
\{ \tau \mid \epsilon \in \mathcal{E} \} \# [[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]] \nsubseteq [[[\text{apaUnsynchE}(\text{onEnv}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))]]].
\]

Then for some set \( \epsilon = \{ i_1, \ldots, i_k \} \in \mathcal{E} \) of reference processes there exists an atomic transition \( \tau \mid _{\epsilon} \in \tau \mid _{\epsilon} \) such that \( s \in [[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]] \) and \( s' \in \{ \tau \mid _{\epsilon} \} \# [[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]] \) but \( s' \nsubseteq [[[\text{apaUnsynchE}(\text{onEnv}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))]]] \).

Let \( m \) be the moving component in \((s, s')\) which changes its local state from \( x_m \) to \( x'_m \). Note, that \( m \) is one of the environment components \( j_1, \ldots, j_p \in \epsilon \) which follows from the definition of relaxation. From \( s \in [[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]] \) it follows that there exist \( j_1, \ldots, j_p \in I \) and \( x_{i_1}, \ldots, x_{i_k}, x_{j_1}, \ldots, x_{j_p} \in X \) such that

\[
\begin{align*}
X[i_1] & \Rightarrow x_{i_1} \in s, \\
\cdots \\
X[i_k] & \Rightarrow x_{i_k} \in s, \\
X[j_1] & \Rightarrow x_{j_1} \in s, \\
\cdots \\
X[m] & \Rightarrow x_m \in s, \\
\cdots \\
X[j_p] & \Rightarrow x_{j_p} \in s
\end{align*}
\]
and
\[ f(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, j_1, x_{j_1}) \text{ is true} \]
and
\[ \ldots \]
and
\[ f(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, m, x_m) \text{ is true} \]
and
\[ \ldots \]
and
\[ f(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, j_p, x_{j_p}) \text{ is true} \]

On the other side, from \( (s, s') \in \tau_0 \) it follows that \( u(m, x_m, x_m') \) is true.

Based on these facts and on the definition of \( \text{apaUnsynchE}(\text{onEnv}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \) we can conclude that
\[
(\begin{array}{c}
(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, j_1, x_{j_1}) \\
\text{[apaUnsynchE(\text{onEnv}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)])}
\end{array})
\]
CHAPTER 5. SYSTEM DESCRIPTIVE FORMULAS

\[ s \in [[[\text{apaUnsynchE}(\text{onEnv}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)))]]] \]

it follows that

\[
(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, j_1, x_{j_1}) \in \\
[[[\text{apaUnsynchE}(\text{onEnv}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)))]]
\]

and

\[
(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, j_p, x_{j_p}) \in \\
[[[\text{apaUnsynchE}(\text{onEnv}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)))].
\]

Based on that fact and on the definition of \( \text{apaUnsynchE}(\text{onEnv}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \) we have that for any \( j \in j_1, \ldots, j_p \) and \( x_j \in x_{j_1}, \ldots, x_{j_p} \)

- either \( f(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, j, x_j) \) is true,
- or there exists \( x'_j \in X \) such that
  - \( f(i_1, x_{i_1}, \ldots, i_k, x_{i_k}, j, x'_j) \) is true and
  - \( u(j, x'_j, x) \) is true.

In other words, there exists \( s' \in [[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]] \) such that

\[
X[i_1] \mapsto x_{i_1} \in s', \\
\ldots \\
X[i_k] \mapsto x_{i_k} \in s', \\
X[j_1] \mapsto x'_{j_1} \in s' \text{ and either } x'_{j_1} = x_{j_1} \text{ or } u(j_1, x'_{j_1}, x_{j_1}), \\
\ldots \\
X[j_p] \mapsto x'_{j_p} \in s' \text{ and either } x'_{j_p} = x_{j_p} \text{ or } u(j_p, x'_{j_p}, x_{j_p})
\]

for some \( x'_{j_1}, \ldots, x'_{j_p} \in X \). Consequently,

- either \( s' = s \) when \( x'_j = x_j \) for any \( j \in j_1, \ldots, j_p \),
- or it is possible to construct a non-empty sequence \( (s', s_2), \ldots, (s_{m-1}, s) \) of atomic transitions where the local transition \( (X[j] \mapsto x'_j, X'[j] \mapsto x_j) \) of the moving environment component can be described by formula \( u(j_1, x'_{j_1}, x_{j_1}) \). Obviously, \( (s_i, s_{i+1}) \in \tau \) for every atomic transition \( (s_i, s_{i+1}) \) from such a sequence.

Based on the definition of the aggregate parallel application, we can conclude that \( s \in \{\tau \} \in E \left( [[[f(I_1, X_1, \ldots, I_k, X_k, I, X)]]] \right) \) which contradicts the initial assumption. \( \square \)

5.5 General Algorithm

In this section we present a general reachability algorithm for an over-approximating reachability analysis with environmental relaxation which is used in the thesis. First, in Section 5.5.1, we start with an algorithm which makes use of Theorems 5.4.7, 5.4.8, 5.4.10 and 5.4.11. An improved version is presented in Section 5.5.2.
Algorithm REACHABLE

input:
\( f_0 \): an SDF describing the set of initial states of the systems
\( T_o \): a finite set of transition schemas describing (all) synchronized and unsynchronized transitions of the system
\( \omega \): environmental relaxation

local:
\( f, f' \): SDFs
\( T, T' \): sets of transition schemas
\( \rho \): a transition schema

1: \( f = f_0 \);
2: \( T = \{ \rho \mid \rho' \in T_o \land (\rho = \text{onRef}(\omega, \rho') \lor \rho = \text{onEnv}(\omega, \rho')) \} \);
3: \( T' = T \);

4: while \( (T' \neq \emptyset) \) do
5: \hspace{1em} select \( \rho \) in \( T' \);
6: \hspace{1em} \( f' = \text{apa}(\rho, f) \);
7: \hspace{1em} if \( \lfloor f' \rfloor \subseteq \lfloor f \rfloor \) then \( T' = T' \setminus \{ \rho \} \) else \( T' = T \);
8: \hspace{1em} \( f = f' \);
9: return \( f \);

Figure 5.1: General reachability algorithm

5.5.1 First Approach

The first general reachability algorithm is illustrated in Figure 5.1. The input required by the algorithm is an SDF \( f_0 \) describing the set of initial states, a set \( T_o \) of all original synchronized and unsynchronized transitions of the system and an environmental relaxation \( \omega \).

The local variables \( f \) and \( f' \) are used to keep an SDF describing the currently explored set of global states. Initially, \( f \) equals the SDF describing the set of initial states (Line 1). The set \( T \) is used to keep all transition schemas after environment relaxation and every schema is split into two parts depending whether it is applied to reference or environment processes (Line 2). Variable \( T' \) is used to keep track of which transitions from sets \( T \) have not been yet applied in the while loop (Lines 4-8).

In the while loop (Lines 4-8) a transition \( \rho \) is picked up from \( T' \) (Line 5) and the aggregate parallel application (acronymed as \( \text{apa} \) in the algorithm) is performed on the currently explored set of global states, which is expressed by the SDF \( f \) (Line 6). How the application is performed depends on the type of the transition \( \rho \). If the transition applies to reference processes, then the parallel application \( \text{paSynchR}_2 \) (Theorem 5.4.7) or \( \text{paUnsynchR} \) (Theorem 5.4.10) is used with the following set union operations on SDFs (Proposition 5.3.3) to obtain the aggregate parallel application. If the transition
applies to environment processes, then the aggregate parallel application \( \text{apaSynchE} \) (Theorem 5.4.8) or \( \text{apaUnsynchE} \) (Theorem 5.4.11) is used. If the application of \( \rho \) does not result in new explored states, then we proceed with other transition schemas in \( T' \) (Line 7). Otherwise, if the set of currently explored states is changed, we reset \( T' \) to its initial value stored in \( T \). The computation is finished when there is no transition schemas left, the application of which can add anything new to the result.

Note, that the algorithm is presented in the declarative manner and is non-deterministic in few places (Lines 2 and 5), which gives rise to many strategies within it. However, the discussion of particular strategies is out of the scope of this thesis.

5.5.2 Improved Algorithm

The next version of the algorithm employs the idea of applying all transitions at once. The theoretical foundation is simple and makes use of Proposition 5.3.3. For example,

\[
\begin{align*}
&\left[ \text{apaSynchR}_1(\text{onRef}(\omega, \rho_1), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \right] \\
&\cup \\
&\left[ \text{apaUnsynchR}(\text{onRef}(\omega, \rho_2), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \right] \\
&\subseteq \\
&\left[ \text{apaSynchR}_1(\text{onRef}(\omega, \rho_1), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \right] \\
&\lor \\
&\left[ \text{apaUnsynchR}(\text{onRef}(\omega, \rho_2), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \right]
\end{align*}
\]

is the immediate consequence of this theorem. Thus, we can make a single disjunction of all transition applications and apply such a disjunction in one step, which appears to be more efficient than applying each transition separately.

Assume, that \( f \) is a state schema describing a set of global states, \( T = \{\rho_1, \ldots, \rho_k, \rho_{u1}, \ldots, \rho_m\} \) is a finite set of transition schemas describing \( k \) synchronized and \( m \) unsynchronized transitions and \( \omega \) is an environmental relaxation. Then we define the aggregate parallel application \( \text{apa}(T, \omega, f) \) as the following schema:

\[
\begin{align*}
&\text{apaSynchR}_1(\text{onRef}(\omega, \rho_1), f) \lor \text{apaSynchE}(\text{onEnv}(\omega, \rho_1), f) \\
&\lor \ldots \lor \\
&\text{apaSynchR}_1(\text{onRef}(\omega, \rho_k), f) \lor \text{apaSynchE}(\text{onEnv}(\omega, \rho_k), f) \\
&\lor \\
&\text{apaUnsynchR}(\text{onRef}(\omega, \rho_{u1}), f) \lor \text{apaUnsynchE}(\text{onEnv}(\omega, \rho_{u1}), f) \\
&\lor \ldots \lor \\
&\text{apaUnsynchR}(\text{onRef}(\omega, \rho_{um}), f) \lor \text{apaUnsynchE}(\text{onEnv}(\omega, \rho_{um}), f).
\end{align*}
\]

Note that we use \( \text{apaSynchR}_1 \) and not \( \text{apaSynchR}_2 \) in the definition above because that version was used in the examples provided in the appendix.
CHAPTER 5. SYSTEM DESCRIPTIVE FORMULAS

Algorithm **reachable2**

**input**:  
$f_0$: an SDF describing the set of initial states of the systems  
$T$: a finite set of transition schemas describing (all) synchronized and unsynchronized transitions of the system  
$\omega$: environmental relaxation  

**local**:  
$f, f'$: SDFs

1: $f' = f_0$;  
2: do  
3: $f = f'$;  
4: $f' = apa(T, \omega, f)$  
5: while ($f \neq f'$);  
6: return $f$;  

Figure 5.2: Improved reachability algorithm

Obviously, $apaSynchR_2$ can be used whenever the lower degree of overapproximation is desired.

The improved version of reachability algorithm is presented in Figure 5.2. In this approach we apply all of the transition schemas at once in every loop cycle until the fixed-point is reached.

In the remainder of this section we show how to perform a syntactic transformation of the disjunction of different applications so that the formula $f(I_1, X_1, \ldots, I_k, X_k, I, X)$ describing the set of currently explored states appears as a single parameter. Using this form will simplify calculation because the set of reachable states is changing all the time during reachability analysis while the parts of the formula related to transitions are constant.

Recall the definition of $apaSynchR_1(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X))$:

$$f(I_1, X_1, \ldots, I_k, X_k, I, X)$$

$$\exists X'[f(I_1, X'_1, \ldots, I_k, X_k, I, X) \land \text{syn}(I_1, X'_1, X_1, I, X)]$$

$$\exists X_k'[f(I_1, X_1, \ldots, I_k, X'_k, I, X) \land \text{syn}(I_k, X'_k, X_k, I, X)]$$

where the synchronized transition schema

$$\rho = \bigvee_{0 \leq I < n} \left[ \bigwedge_{0 \leq G < n \land G \neq I} \text{syn}(I, X[I], X'[I], G, X[G]) \land \mathcal{J}(I) \right]$$

describes a synchronized transition, $\omega = (k, I_R, \text{range}(I_R), \mathcal{E})$ is an envi-
CHAPTER 5. SYSTEM DESCRIPTIVE FORMULAS

environmental relaxation, and \( f(I_1, X_1, \ldots, I_k, X_k, I, X) \) is a system descriptive formula describing a set of global states. The definition can be syntactically transformed to the following form:

\[
\exists x_1 \left[ f(I_1, X_1', \ldots, I_k, X_k, I, X, N, P) \land (X_1 = X_1' \lor \text{syn}(I_1, X_1', X_1, I, X, N)) \lor \ldots \lor \right.
\]

\[
\exists x_k' \left[ f(I_1, X_1, \ldots, I_k, X_k', I, X, N, P) \land (X_k = X_k' \lor \text{syn}(I_k, X_k', X_k, I, X, N)) \right]
\]

This expression can be re-written as follows by moving the existential quantifiers to the front of the formula and quantifying variable \( X' \) as well even when it is not used in this particular application:

\[
\exists x_1', \ldots, x_k', x' \left[ f(I_1, X_1', \ldots, I_k, X_k', I, X', N, P) \land (X_1 = X_1' \land \ldots \land X_k = X_k' \land X = X') \lor \right.
\]

\[
\text{syn}(I_1, X_1', X_1, I, X, N) \lor \ldots \lor \text{syn}(I_k, X_k', X_k, I, X, N)
\]

\[
X_1 = X_1' \land \ldots \land X_{k-1} = X_{k-1}' \land X_k = X_k'
\]

In the same way we re-write other applications: \( \text{apaSynchE(onEnv(\omega, \rho)), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \) can be represented as

\[
\exists x_1', \ldots, x_k', x' \left[ f(I_1, X_1', \ldots, I_k, X_k', I, X', N, P) \land (X_1 = X_1' \land \ldots \land X_k = X_k' \land X = X') \lor \right.
\]

\[
X_1 = X_1' \land \ldots \land X_k = X_k'
\]

\[
G = I_1 \land \text{syn}(I, X', X, I_1, X_1)) \lor \ldots \lor \text{syn}(I, X', X, I_k, X_k)) \lor \right.
\]

\[
G \in I \land G \neq I_1 \land \ldots \land G \neq I_k \land \text{syn}(I, X', X, G, X_g, N)
\]

95
CHAPTER 5. SYSTEM DESCRIPTIVE FORMULAS

\[ \text{\textit{apaUnsychR}}(\text{onRef}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \text{ can be represented as} \]
\[
\exists X'_1, \ldots, X'_k, X'.
\begin{bmatrix}
  f(I_1, X'_1, \ldots, I_k, X'_k, I, X', N, P) \\
  (X_1 = X'_1 \wedge \ldots \wedge X_k = X'_k \wedge X = X') \\
  u(I_1, X'_1, X_1, N) \\
  \vdots \\
  u(I_k, X'_k, X_k, N) \\
  X_1 = X'_1 \wedge \ldots \wedge X_{k-1} = X'_{k-1} \wedge X = X'
\end{bmatrix}
\]

and \( \text{\textit{apaUnsychE}}(\text{onEnv}(\omega, \rho), f(I_1, X_1, \ldots, I_k, X_k, I, X)) \) can be represented as

\[
\exists X'_1, \ldots, X'_k, X'.
\begin{bmatrix}
  f(I_1, X'_1, \ldots, I_k, X'_k, I, X', N, P) \\
  (X_1 = X'_1 \wedge \ldots \wedge X_k = X'_k \wedge X = X') \\
  u(I, X', X, N) \\
  X_1 = X'_1 \wedge \ldots \wedge X_k = X_k
\end{bmatrix}
\]

Therefore, any definition of the aggregate parallel application can be presented in a schematic form

\[
\exists X'_1, \ldots, X'_k, X'.
\begin{bmatrix}
  f(I_1, X'_1, \ldots, I_k, X'_k, I, X', N, P) \\
  (X_1 = X'_1 \wedge \ldots \wedge X_k = X'_k \wedge X = X') \\
  g(I_1, X_1, X'_1, \ldots, I_k, X_k, X'_k, I, X, X', N)
\end{bmatrix}
\]

where \( g(I_1, X_1, X'_1, \ldots, I_k, X_k, X'_k, I, X, X', N) \) is the part specific to a particular expression. Consequently, the operations on Line 5 in Figure 5.2 can be expressed as follows:

\[
\exists X'_1, \ldots, X'_k, X'.
\begin{bmatrix}
  f(I_1, X'_1, \ldots, I_k, X'_k, I, X', N, P) \\
  (X_1 = X'_1 \wedge \ldots \wedge X_k = X'_k \wedge X = X') \\
  g_1(I_1, X_1, X'_1, \ldots, I_k, X_k, X'_k, I, X, X', N) \\
  \vdots \\
  g_m(I_1, X_1, X'_1, \ldots, I_k, X_k, X'_k, I, X, X', N)
\end{bmatrix}
\]

where formulas \( g_1(I_1, X_1, X'_1, \ldots, I_k, X_k, X'_k, I, X, X', N), \ldots, g_m(I_1, X_1, X'_1, \ldots, I_k, X_k, X'_k, I, X, X', N) \) depend on the corresponding transition schemas from the set \( T \) and \( f(I_1, X'_1, \ldots, I_k, X'_k, I, X', N, P) \) appears only once.
5.6 Definability

In this section we discuss definability issues — the ability to express sets of global states in the given encoding (disjunctive relations and SDFs) and how this property is preserved during operations on sets of global states.

Theorem 5.1.2 provides a mean to encode a set of global states as a disjunctive relation if we express it first with a state schema in the normal form. However, we know that we can express only disjunctive sets as a disjunctive relation while state schemas can encode any sets of states (in finite systems). The following theorem addresses that issue.

**Theorem 5.6.1** A set $S$ of global states is disjunctive if and only if there exists a state schema $F$ in the normal form:

$$\bigwedge_{0 \leq I < n} f(I, X[I])$$

such that $S = [F].$

**Proof:** A set of global states $S$ is a disjunctive set if and only if there exists a disjunctive relation $R(I, X)$ with corresponding system descriptive formula $f(I, X)$ such that $[[f(I, X)]] = S$ (the existence of an SDF for any disjunctive relation follows from the fact that we consider finite state-transition systems and, accordingly, finite disjunctive relations). Then, by Theorem 5.1.2 we can construct a state schema of the form

$$\bigwedge_{0 \leq I < n} f(I, X[I])$$

and

$$[[\bigwedge_{0 \leq I < n} f(I, X[I])]] = S.$$ 

In other words, it is impossible to express a non-disjunctive set of global states as a state schema in the normal form required for Theorem 5.1.2, and it is always possible to express in that form and apply the theorem whenever a set of states is disjunctive.

**Example 5.6.2** Assume that we wish to describe a set $S$ of global states of a system of size $N = 10$ such that at least one process of those with indexes 4...9 is in state 5. A state formula for $S$ can be

$$X[4] = 5 \lor \ldots \lor X[9] = 5$$

but it is impossible to express $S$ by a state schema in the normal form required for Theorem 5.1.2, because $S$ is not disjunctive. 

---

97
We introduced parameterization in Sections 4.3-4.4 and 5.2 to be able to encode even non-disjunctive sets of global states as parameterized DRs and SDFs. However, not every parameterized disjunctive relation can be represented as a finite Presburger formula if it has parameters with unbounded domains. We call a family of sets of global states of a family of uniform systems *Presburger definable* if it can be represented by a parameterized disjunctive relation expressible via (finite) Presburger formula. A family of sets of unit transitions are Presburger definable, if they can be represented by transition descriptive formulas.

**Theorem 5.6.3** All set operations on sets of global states defined in Section 5.3, except set difference for the general case, performed on corresponding parameterized disjunctive relations are closed under Presburger definability property.

Parallel application and aggregate application defined in Section 5.4, are closed under Presburger definability property under condition, that respective family of synchronized or unsynchronized transitions are Presburger definable.

**Proof:** The closure property follows directly from the corresponding operations on formulas – whenever we have a finite formula in the input, we get a finite formula as the output.
Chapter 6
Experimental Evaluation

The method of formal verification of parameterized systems presented in the thesis has been successfully applied to a number of examples presented in the appendix. The examples include the verification of safety properties of a simplified and a full version of Lamport’s Bakery algorithm (Appendix A and Appendix B, respectively) as well as a simplified version of Szymanski algorithm (Appendix C).

The main simplifying assumption in the version of Bakery algorithm presented in Appendix A is that a process can read the values of tickets of all other processes as well as change its own state in one atomic step.

The next version of Bakery algorithm in Appendix B is more realistic as it considers a situation when two processes can generate the same ticket number as a result of their concurrent execution.

Lamport’s Bakery algorithm has been verified previously by other methods [12, 59]. For example, the method of invisible variants [12] verifies a version of Bakery algorithm with atomicity assumption and bounded tickets (which are unbounded in the original algorithm). The method of counter abstraction [59] could handle a version of Bakery algorithm with atomicity assumption after the introduction of auxiliary variables. Regular model checking [8] successfully verified automatically Lamport’s Bakery algorithm with atomicity assumption as well as its improved version with non-atomic computation of tickets. Environment abstraction method [23] can handle a version of Bakery algorithm with atomicity assumption, while an improved version of the method [64] can handle the full version of the algorithm without atomicity assumption.

Appendix C contains a version of Szymanski algorithm with added atomicity assumption such that a process can read the state of all other processes in the system and modify its own state in one atomic step. Safety and liveness properties of Szymanski algorithm with atomicity assumption have been automatically verified by other researches [3, 16, 12, 23]. However, to the best of the author’s knowledge, there is no fully automatic approach capable
to verify safety property of the full version of Szymanski algorithm (without atomicity assumption). Clark, Talupur and Veith [23] claimed that they had developed a natural extension to their method of environment abstraction which eliminates the atomicity assumption. They promised to describe it in future work, however, Talupur [64] reported later a failure with the full version of algorithm as they could not model it correctly with their approach.

The choice of examples indicates that the proposed approach is limited to the cases where the atomicity assumption can be applied. Although it is possible to model systems without the atomicity assumption, the degree of over-approximation is too high to achieve meaningful results. For example, we tried to model and verify the full version of Szymanski algorithm without atomicity assumption in our framework but failed to prove the safety property because non-safe system states were included in the over-approximation of the set of reachable states.

The tests were run on an Opteron 250 processor with 4 GB RAM available. Table 6.1 presents the running times of the scripts for Omega Calculator presented in the appendices.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time (mm:ss)</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplified Bakery</td>
<td>0:12</td>
<td>4</td>
</tr>
<tr>
<td>Bakery</td>
<td>5:27</td>
<td>12</td>
</tr>
<tr>
<td>Szymanski</td>
<td>11:00</td>
<td>16</td>
</tr>
</tbody>
</table>

Every example contains a script for Omega calculator which computes a set of reachable states and checks the safety property against it. The number of iterations required to compute the set of reachable states has been determined empirically just because Omega calculator lacks the ability to express conditional loops (for example, a *while* loop would perfectly solve the problem).

1Note that the number of iterations provided here is the one used to calculate the set of reachable states only. In practise, there is always at least one iteration which does not produce any new system states but nevertheless required to establish that the fixed-point is reached.
Chapter 7

Related Work

The use of rich assertional languages in verification of parameterized systems is not new. In particular, it includes first order logic (e.g. Pnueli et al. [57], Ghilardi and Ranise [38]), subsets of second order logic (e.g. Jensen et al. [42], Pnueli and Shahar [58]) and Presburger constraints (Bultan et al. [21]). A general framework for rich assertional languages has been formulated by Kesten et al. [46].

The use of Presburger constraints has been widely adopted in verification of counter systems, for example in works of Fribourg et al. [36], Bultan et al. [21], Bardin et al. [14], Demri et al. [30]. Presburger arithmetic on integers seems to be suitable for describing systems with an infinite state space because of having counters with unbounded integer domains. The presence of a well developed tool such as Omega library [45] allows for efficient constraint manipulation. In some of these works parameterized systems are used as benchmarks, however, with a fixed number of processes. For example, an instance of Bakery algorithm with two processes and infinite ticket domains is used by Bultan et al. [21]. There were also attempts (see Bultan et al. [18, 21], Comon and Cortier [26], Schude and Schneider [61], Demri [29]) to define temporal logic with Presburger constraints and use it for the verification of infinite space systems. Presburger arithmetic is also used as a part of symbolic composite representation in Action Language Verifier [65] to verify properties of reactive parameterized systems.

State and transition schemas of our modeling language can be related to some assertive languages in deductive verification in what concerns the use of index variables in quantified formulas. For example, the array reference parameterized by the array index in work of Arons et al. [12] closely resembles the schema variable used by our modeling language.

A generic framework introduced by Bouajjani et al. [19] uses Petri Nets to model parameterized system. A new logic called Colored Markings Logic (CML) has been proposed to specify system configurations. It allows nested quantification and can be parameterized by an underlying first order logic.
used to express constraints on transitions, for example, by Presburger arithmetic on natural numbers. The work also presents a fragment of CML which is decidable.

The environment abstraction in works of Clarke, Talupur and Veith [23, 64, 24] has probably the closest idea to consider the system as an interaction between a chosen reference process and its environment. The constructed abstraction is finite and therefore the method is complete. Our approach uses a fixed set of reference processes and one environment process abstracts their environment. Our approach does not guarantee termination because the state space of abstracted processes can be infinite or unbounded. The same version of Szymanski algorithm is considered as a benchmark in both approaches.

One of the recent trends in the verification of parameterized systems is finding a cut-off (see e.g. Hanna et al. [40], Kaiser et al. [44], Abdulla et al. [10]) A cut-off is the smallest instance of a parameterized system such that the property is satisfied in it if and only if it is satisfied for all other instances larger than this one. Thus, the verification reduces to proving the property in systems of usually small sizes until the cut-off is reached.

A cut-off technique which is somewhat close to the one proposed in this thesis is view abstraction technique by Abdulla et al. [10]. The idea is to inspect a small number of processes to show the correctness of a whole parameterized system. As in this work, the system is considered from the perspective of a fixed number of processes. However during the verification the number of the processes grows until a cut-off point is found which is a fixed and usually small number of processes which is enough to verify a safety property. The concept of a cut-off point whose size defines the precision of the abstraction can be related to the number of reference processes. In our case two reference processes was either enough to verify the considered systems or it did not seem helpful to increase their number to get rid of false property violations.

The abstraction function in view abstraction approach corresponds to the way we encode a set of system states as a system descriptive formula with the help of Theorem 5.1.2 while the semantics of an SDF plays the role of the concretization function. View abstraction can handle universal global and existential transitions in the system model. Our approach can handle global transitions while the existential transitions in our case are simply reduced to the unsynchronized transitions which can explain why our method cannot verify as many systems as view abstraction.
The main contribution of the thesis is a new approach in automatic verification of safety properties of parameterized systems consisting of concurrent processes. It is an instantiation of an abstraction approach when a large finite state system or a system with an infinite state space is abstracted to a simpler state system which can be further model checked w.r.t. a given safety property.

The abstraction to a finite state transition system is one of the mainstream approaches in automatic verification of the parameterized systems ([37, 59, 34, 28, 48]). It is an instance of the Abstract Interpretation framework [27] where the verification is performed in an abstract domain instead of the actual one. The abstraction of the system must be sound w.r.t. the considered property, so the result of the verification on the abstracted domain also holds in the actual system.

In an attempt to classify the related works according to how parameterized systems are abstracted and what is behind the choice of abstraction techniques we can see that it is a combination of a domain specific knowledge and the available mathematical tools that decides about the choice. For example, regular languages is an example of such a tool, and once started being used it became a base for many approaches where it plays the role of an efficient symbolic representation of infinite state spaces [2]. On the other side, abstraction techniques in different approaches use domain specific knowledge which comes from certain predetermined classes of parameterized systems they intend to handle.

The idea of the method presented in the thesis was to simulate and follow an abstraction of the system designer’s mind. The method presented in the thesis makes an attempt to use such a domain-specific knowledge that a protocol designer often thinks about a limited number of reference processes and the rest of the system. The over-approximation schema proposed in the
thesis abstracts the original system consisting of a parameterized number of processes into a system consisting of a fixed number of processes: a small fixed number of reference processes (there are only two reference processes in all examples presented in the appendix) and one process representing the rest of the system. The approach can deal with parameterized networks of both finite and infinite state processes.

Although the abstract system contains the fixed number of processes, we allow each process to have an infinite state space. This means, that the abstracted system can still have an infinite state space (and it can be verified anyway because symbolic model checking is used). As a consequence, the reachability algorithm may never terminate. This is not surprising if we take into account that the nature of the problem – the fact that the verification of parameterized systems is undecidable [11]. Research of the sufficient conditions under which the state space exploration is guaranteed to terminate may be a goal of the future work.

The work on environment abstraction [23, 64, 24] uses a similar idea to follow human way of thinking when designing a parameterized system. However, environment abstraction uses only a single reference process and its environment (according to so called “Ptolemaic” perspective) while the abstraction approach used in this thesis uses at least two reference processes and concentrates on the mutual relation between them and the rest of the system. The environment abstraction results in a finite state abstracted system, while our approach allows the abstracted system to have infinite state space in case the local state space of a process is unbounded.

Another idea used in this thesis is that the complexity of representing the abstracted systems and performing reachability operations does not need to be very high because usually the human mind cannot proceed that many things concurrently and tends to simplify (approximate) problem complexity in an easy way. For example, a system designer can think in terms of usual relations in a form such as

\[ \text{a process } i \text{ is in state } x_i, \text{ a process } j \text{ is in state } x_j, \text{ and the all other processes are in states } x_1, x_2 \text{ or } x_k. \]

That is, the underlying representation and computation logic does not need to be more complex than that. Therefore, a choice made in this work of using relations on local states of processes and their ids as a symbolic representation of sets of states seems to be quite reasonable.

The input system is described by state and transition schemas which is a means of describing infinite families of Presburger formulas. Although good for description, such formulas are not suitable for model checking. After abstraction is performed\(^1\), another representation is used for a family of parameterized systems – disjunctive relations encoded by Presburger formulas. The thesis describes how to perform reachability and set operations

\(^1\text{It is not possible to describe the original system with disjunctive relations. Therefore, we can use them only after abstraction is performed.}\)

104
on this symbolic representation required for model checking. Presburger
arithmetics is decidable, and examples in the appendix show how symbolic
model checking can be performed with the help of available tools such as
Omega Calculator.

The abstraction algorithm is limited to systems having only certain
types of synchronized transitions. If a system has synchronized transitions
which do not comply with the algorithm requirements, we can still over-
approximate such transitions, for example, to unsynchronized ones, and ab-
stract the system afterwards, as we did in Appendix C. Despite the fact,
that the system in Appendix C was verified successfully, most other sys-
tems get over-approximated too much if we solve the limitation this way.
Therefore, handling of other types of synchronized transition is the major
direction of the future work.
Bibliography


Appendix A

Simplified Bakery Algorithm

In this section we consider a simplified version of Lamport’s Bakery algorithm which solves the critical section problem for an arbitrary finite number of concurrent processes. The original version of the algorithm was first published in [51]. The version presented here sometimes is referred to in the literature as Simplified Bakery algorithm or as Bakery algorithm with atomicity conditions.

A.1 System Definition

Let us first define a family of uniform finite state transition systems \((\{X_n\}_{n \in \mathbb{N}}, \{S_n\}_{n \in \mathbb{N}}, F(N), \rho(N))\). We assume that there is at least three processes in the family. Each process has two registers: \(Q\) and \(T\):

\[
\{X_n\}_{n \geq 3} = \{Q[0], T[0], \ldots, Q[n - 1], T[n - 1]\}_{n \geq 3}.
\]

Register \(Q\) ranges over three control states:

- 0 – *idle*: the process is in the non-critical section and has not requested the access to the critical section;

- 1 – *wait*: the process has requested the access to the critical section and waits for it;

- 2 – *use*: the process is in the critical section.

Register \(T\) ranges over the set natural numbers with zero and is used to store a ticket. Therefore, \(\{S_n\}_{n \geq 3}\) is the family of sets of all possible assignments \(s \in S_n\) for each \(n \geq 3\) such that \(\{s(Q[i]) \in \{0, 1, 2\}\}_{n \geq 3}\) and \(\{s(T[i]) \in \mathbb{N}\}_{n \geq 3}\) where \(0 \leq i < n\).
APPENDIX A. SIMPLIFIED BAKERY ALGORITHM

The set of initial states is described by the following parameterized state schema $F(N)$:

$$N \geq 3 \land \bigwedge_{0 \leq I < N} Q[I] = 0 \land T[I] = 0$$

which states that initially all processes are in the idle state and have their tickets set to zero.

The transition relation described by transition schema $\rho(N) = \rho_1(N) \lor \rho_2(N) \lor \rho_3(N)$ consists of the following three meta-transitions.

- Synchronized transition $\rho_1(N)$:

$$\bigvee_{0 \leq I < N} \left[ \bigwedge_{0 \leq G < N \land G \neq I} (Q[I] = 0 \land Q'[I] = 1 \land T'[I] > T[G]) \land J(I, N) \right].$$

Whenever a process $0 \leq I < N$ requests the access to the critical section, it changes its control state from idle ($\{Q[I] = 0\}$) to wait ($\{Q'[I] = 1\}$) and obtains a ticket which is greater than the tickets of all other processes in the system ($\bigwedge_{0 \leq G < N \land G \neq I} (T'[I] > T[G])$). As it is a simplified version of the Lamport’s Bakery algorithm, we do not bother with details how the value of the ticket is obtained and assume this step to be atomic.

- Synchronized transition $\rho_2(N)$:

$$\bigvee_{0 \leq I < N} \left[ \bigwedge_{0 \leq G < N \land G \neq I} (Q[I] = 1 \land Q'[I] = 2 \land T[I] = T'[I]) \right].$$

A process can enter the critical section by changing its state from wait ($\{Q[I] = 1\}$) to use ($\{Q'[I] = 2\}$) if it has the smallest ticket number (ignoring the processes with the ticket equal to zero).

- Unsyncronized transition $\rho_3(N)$:

$$\bigvee_{0 \leq I < N} [Q[I] = 2 \land Q'[I] = 0 \land T'[I] = 0 \land J(I, N)].$$

A process leaves the critical section and resets its ticket to zero.

A.2 Safety Property

We are interested to prove the property that no two processes can be in the critical section at the same time. The corresponding family of sets of safe states where the system size $N$ is a parameter can be described by the following parameterized state schema:

$$\bigwedge_{0 \leq I_1 < I_2 < N} (Q[I_1] \neq 2 \lor Q[I_2] \neq 2).$$
The set of safe states is expressed as the following parameterized system
descriptive formula
\[ p(I_1, Q_1, T_1, I_2, Q_2, T_2, I, T, K) = 0 \leq I_1 < I_2 < N \land \neg (Q_1 = 2 \land Q_2 = 2). \]

### A.3 Environmental Relaxation

Let us perform the environmental relaxation of rank two because we used two parameters \( I_1 \) and \( I_2 \) to express the safety property. In the following we demonstrate how to relax transitions \( \rho_1, \rho_2 \) and \( \rho_3 \). The relaxation is parameterized by the reference set \( \{ I_1, I_2 \} \) and the resulting formulas divided into environment and reference parts for convenience based on the type of the moving component.

- \( \rho'_1(I_1, I_2, N) \) for environment processes:
  \[
  \bigvee_{0 \leq I < N \land \neg I = I_1 \land \neg I = I_2} \left( Q[I] = 0 \land Q'[I] = 1 \land \bigwedge_{0 \leq G < N \land G \neq I} \left( (G = I_1 \rightarrow T'[I] > T[I_1]) \land (G = I_2 \rightarrow T'[I] > T[I_2]) \right) \right) \]

- \( \rho'_1(I_1, I_2, N) \) for reference processes:
  \[
  \bigwedge_{0 \leq G < N \land G \neq I_1} (Q[I_1] = 0 \land Q'[I_1] = 1 \land T'[I_1] > T[G]) \land J(I_1, N)
  \]
  and
  \[
  \bigwedge_{0 \leq G < N \land G \neq I_2} (Q[I_2] = 0 \land Q'[I_2] = 1 \land T'[I_2] > T[G]) \land J(I_2, N)
  \]

- \( \rho'_2(I_1, I_2, N) \) for environment processes:
  \[
  \bigvee_{0 \leq I < N \land \neg I = I_1 \land \neg I = I_2} \left( Q[I] = 1 \land Q'[I] = 2 \land T[I] = T'[I] \land \bigwedge_{0 \leq G < N \land G \neq I} \left( (G = I_1 \rightarrow (T[I] < T[I_1] \lor T[I_1] = 0)) \land (G = I_2 \rightarrow (T[I] < T[I_2] \lor T[I_2] = 0)) \right) \right) \]

114
• $\rho'_2(I_1, I_2, N)$ for reference processes:

$$\bigwedge_{0 \leq G < N \land G \neq I_1} \left( Q[I_1] = 1 \land Q'[I_1] = 2 \land T[I_1] = T'[I_1] \land (T[I_1] < T[G] \lor T[G] = 0) \right) \land J(I_1, N)$$

and

$$\bigwedge_{0 \leq G < N \land G \neq I_2} \left( Q[I_2] = 1 \land Q'[I_2] = 2 \land T[I_2] = T'[I_2] \land (T[I_2] < T[G] \lor T[G] = 0) \right) \land J(I_2, N)$$

• $\rho'_3(I_1, I_2, N)$ for environment processes:

$$\bigvee_{0 \leq I < N \land I \neq I_1 \land I \neq I_2} [Q[I] = 2 \land Q'[I] = 0 \land T'[I] = 0 \land J(I, N)]$$

• $\rho'_3(I_1, I_2, N)$ for reference processes:

$$Q[I_1] = 2 \land Q'[I_1] = 0 \land T'[I_1] = 0 \land J(I_1, N)$$

and

$$Q[I_2] = 2 \land Q'[I_2] = 0 \land T'[I_2] = 0 \land J(I_2, N).$$

We also need to transform state formula $F(N)$ into a corresponding pSDF $f_0(I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T)$:

$$F(N) \doteq I_1 < I_2 \land 0 \leq I_1 < N \land Q_1 = 0 \land T_1 = 0$$

$$\land$$

$$0 \leq I_2 < N \land Q_2 = 0 \land T_2 = 0$$

$$\land$$

$$0 \leq I < N \land I \neq I_1 \land I \neq I_2 \land Q = 0 \land T = 0.$$ 

### A.4 Reachability Analysis

#### A.4.1 First Approach

Reachability analysis according to the algorithm \texttt{REACHABLE} (Figure 5.1) consists of

• parallel or aggregate parallel application of meta-transitions described by the transition schemas above to the the currently explored set of reachable states described by a pSDF $f(I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T)$ (which initially is equal to $f_0(I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T)$), and
• merging the result with the current value of \( f(I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T) \)

until we reach a fixed-point, that is, when no application of any meta-transition can produce anything new. All the necessary operations on pSDFs are described in Section 5.

For example, the result of aggregate parallel application of \( \tau = [\rho'_1(I_1, I_2, N)] \) to the set of global states described by a pSDF \( f(I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T) \) applied only to environment processes according to Theorem 5.4.8 is expressed by the following pSDF \( f'(I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T) \):

\[
\exists_{Q', T'} \forall_G
\begin{align*}
& f(I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T) \\
& \quad \land \\
& \quad 0 \leq I < N \land I \neq I_1 \land I \neq I_2 \\
& \quad \land \\
& \quad Q = 0 \land Q' = 1 \\
& \quad \land \\
& \quad G = I_1 \\
& \quad \land \\
& \quad 0 \leq I_1 < N \\
& \quad \land \\
& \quad I \neq I_1 \\
& \quad \rightarrow \\
& \quad T' > T_1 \\
& \quad \land \\
& \quad G = I_2 \\
& \quad \land \\
& \quad 0 \leq I_2 < N \\
& \quad \land \\
& \quad I \neq I_2 \\
& \quad \rightarrow \\
& \quad T' > T_2 \\
& \quad \land \\
& \quad (Q = Q' \land T = T')
\end{align*}
\]

\[Q \mapsto Q' \]
\[T \mapsto T' \]
\[Q' \mapsto Q \]
\[T' \mapsto T \]

A.4.2 Second Approach

To apply the algorithm presented in Section 5.5.2 we construct the transition relation with environmental relaxation for the aggregate parallel application of all transitions in the system. The Omega Calculator [45] script illustrated below represents such a verification of Simplified Bakery Algorithm.

The property \( P \) (Lines 4-6) and the set of initial states \( R_i \) (Lines 9-11) are expressed as Presburger relations on variables \( I_1, Q_1, T_1, I_2, Q_2, T_2, I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T, I_1, Q_1, T_1, I_2, Q_2, T_2, \) where variables \( I_1 \) and \( I_2 \) represent indexes of reference components while \( Q_1, T_1 \) and \( Q_2, T_2 \) their respective local states. Variable \( I \) is used to represent environment components’ indexes while variables \( Q \) and \( T \) - their local states.
APPENDIX A. SIMPLIFIED BAKERY ALGORITHM

The over-approximated transition relation $Tr$ (Lines 13-54) is a union of all encoded meta-transitions. In what follows, we iteratively apply the transition relation $Tr$ to the set relation $R$ describing the current set of explored states (Line 58 in Iteration 1) and filter the result afterwards (Lines 60-69 in Iteration 1). We check if the fixed-point is reached after each iteration (Line 70 in Iteration 1). It is required 5 iterations to reach the fixed-point, which was determined empirically because of the limited script language. Obviously, it can be automated as well.

After the fixed point is reached, we make the intersection of the (over-approximated) set of reachable states with the set of bad states (Lines 140-141) and print the result (Line 144).

```plaintext
SimplifiedBakery
1 symbolic N;
3  # The set of bad states (the negation of the property)
4  Bad :=
5  { [I1,2,T1,I2,2,T2,1,Q,T] :
6      0<=I1,I2<N && I1<I2 && !(I1=1) && !(I2=1) };
8  # The set of initial states
9  Ri :=
10  { [I1,0,0,I2,0,0,1,0,0] :
11      0<=I1,I2,N && I1<I2 && !(I1=1) && !(I2=1) };
13  Tr :=
14  # rho1 for environment
15  { [I1,Q1,T1,I2,Q2,T2,1,0,T] -> [I1,Q1,T1,I2,Q2,T2,1,1,T'] :
16      T<T' && T1<T' && T2<T' }
17  union
18  # rho1 for I1
19  { [I1,0,T1,I2,Q2,T2,I,Q,T] -> [I1,1,T1',I2,Q2,T2,I,Q,T] :
20      T1<T1' && T2<T1' && T<T1' }
21  union
23  # rho1 for I2
24  { [I1,Q1,T1,I2,0,T2,1,Q,T] -> [I1,Q1,T1,I2,1,T2',1,Q,T] :
25      T,T1,T2<T2' }
27  union
29  # rho2 for environment
30  { [I1,Q1,T1,I2,Q2,T2,1,1,T] -> [I1,Q1,T1,I2,Q2,T2,1,2,T] :
31      (T<T1 || T1=0) && (T<T2 || T2=0) }
32  union
33  # rho2 for I1
34  # rho2 for I2
117
```
APPENDIX A. SIMPLIFIED BAKERY ALGORITHM

{ [I_1, I_2, Q_1, T_1, I_2, Q_2, T_2, I, Q, T] } → { [I_1, I_2, Q_1, T_1, I_2, Q_2, T_2, I, Q, T] :
(T_1 < T_2 || T_2 = 0) && (T_1 < T || T = 0) }

union

# rho2 for I_2

{ [I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T] } → { [I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T] :
(T_2 < T_1 || T_1 = 0) && (T_2 < T || T = 0) }

union

# rho3 for environment

{ [I_1, Q_1, T_1, I_2, Q_2, T_2, I, 2, T] } → { [I_1, Q_1, T_1, I_2, Q_2, T_2, I, 0, 0] }

union

# rho3 for I_2

{ [I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T] } → { [I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T] }

union

# the idling condition

{ [I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T] } → { [I_1, Q_1, T_1, I_2, Q_2, T_2, I, Q, T] };

# # # # # # # # # # # # # # # # # # # # # # # iteration 1

R := Ri;
R’ := range (Tr \ R);
Rf :=

R’ intersection complement range

{ [I_1, Q_1, T_1, I_2, Q_2, T_2, I] } → { [I_1, Q_1, T_1, I_2, Q_2, T_2, I’, Q’, T’] :
0 < I < N && !(I = I_1) && !(I = I_2) }

\ complement range

({ [I_1, Q_1, T_1, I_2, Q_2, T_2, 1, Q, T] } \ R’) →
[I_1, Q_1, T_1, I_2, Q_2, T_2, 1] \ R’
Rf subset R;

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # iteration 2

R := Rf;
R’ := range (Tr \ R);
Rf :=

R’ intersection complement range

{ [I_1, Q_1, T_1, I_2, Q_2, T_2, 1] } → { [I_1, Q_1, T_1, I_2, Q_2, T_2, I’, Q’, T’] :
0 < I < N && !(I = I_1) && !(I = I_2) }

\ complement range

({ [I_1, Q_1, T_1, I_2, Q_2, T_2, 1, Q, T] } \ R’) →
APPENDIX A. SIMPLIFIED BAKERY ALGORITHM

[11, Q1, T1, I2, Q2, T2, I] \ R')

Rf subset R;

# # # # # # # # # # # # # # # # # # # # # # # iteration 3
R := Rf;
R' := range (Tr \ R);
Rf :=
R' intersection complement range
\{{[I1, Q1, T1, I2, Q2, T2, I] ->
[I1, Q1, T1, I2, Q2, T2, I', Q', T'] :
0<=I<N & & !(I=I1) & & !(I=I2)
}\)
complement range
\{{[I1, Q1, T1, I2, Q2, T2, I, Q, T] ->
[I1, Q1, T1, I2, Q2, T2, I] \ R')
);
Rf subset R;

# # # # # # # # # # # # # # # # # # # # # # # iteration 4
R := Rf;
R' := range (Tr \ R);
Rf :=
R' intersection complement range
\{{[I1, Q1, T1, I2, Q2, T2, I] ->
[I1, Q1, T1, I2, Q2, T2, I', Q', T'] :
0<=I<N & & !(I=I1) & & !(I=I2)
}\)
complement range
\{{[I1, Q1, T1, I2, Q2, T2, I, Q, T] ->
[I1, Q1, T1, I2, Q2, T2, I] \ R')
);
Rf subset R;

# # # # # # # # # # # # # # # # # # # # # # # iteration 5
R := Rf;
R' := range (Tr \ R);
Rf :=
R' intersection complement range
\{{[I1, Q1, T1, I2, Q2, T2, I] ->
[I1, Q1, T1, I2, Q2, T2, I', Q', T'] :
0<=I<N & & !(I=I1) & & !(I=I2)
}\)
complement range
\{{[I1, Q1, T1, I2, Q2, T2, I, Q, T] ->
[I1, Q1, T1, I2, Q2, T2, I] \ R')
}
133 );
134 Rf subset R;

136 # The fixed-point is reached
137 # (determined empirically)
138 R;

140 ReachedBadStates :=
141 R intersection Bad;

143 # print relation describing reached bad states
144 ReachedBadStates;
Appendix B

Lamport’s Bakery Algorithm

Here we consider a more complicated version of Lamport’s Bakery algorithm than in Appendix A which is closer to the reality, because we take into account that any two processes can potentially generate the same ticket number while working concurrently. This version of the algorithm is taken from [7].

B.1 System Definition

Let us first define a family of uniform finite state transition systems \(\{X_n\}_{n \in \mathbb{N}}, \{S_n\}_{n \in \mathbb{N}}, F(N), \rho(N)\). We assume that there is at least three processes in the family. Each process has three registers: \(Q\), \(T\) and \(A\):

\[
\{X_n\}_{n \geq 3} = \{Q[0], T[0], A[0], \ldots, Q[n - 1], T[n - 1], A[n - 1]\}_{n \geq 3}.
\]

Register \(Q\) ranges over four control states:

- 0 – idle: the process is in the non-critical section and has not requested the access to the critical section;
- 1 – choose: the process has requested the access to the critical section and is generating the ticket value;
- 2 – wait: the process has generated the ticket and is waiting for the access to the critical section;
- 3 – use: the process is in the critical section.

Register \(T\) ranges over the set of natural numbers with zero and is used to store a ticket. Register \(A\) is an auxiliary register used to generate a ticket. Therefore, \(\{S_n\}_{n \geq 3}\) is the family of sets of all possible assignments \(s \in S_n\).
for each \( n \geq 3 \) such that \( \{s(Q[i]) \in \{0, 1, 2, 3\}\}_{n \geq 3}, \{s(T[i]) \in \mathbb{N}\}_{n \geq 3} \) and \( \{s(A[i]) \in \mathbb{N}\}_{n \geq 3} \) where \( 0 \leq i < n \).

The set of initial states is described by the following parameterized state schema \( F(N) \):

\[
N \geq 3 \land \bigwedge_{0 \leq i < N} Q[I] = 0 \land T[I] = 0 \land A[I] = 0
\]

which states that initially all processes are in the idle state and have their tickets and auxiliary variables set to zero.

The transition relation described by transition schema \( \rho(N) = \rho_1(N) \lor \rho_2(N) \lor \rho_3(N) \lor \rho_4(N) \) consists of the following four meta-transitions.

- **Synchronized transition** \( \rho_1(N) \):

\[
\forall_{0 \leq I < N} \left[ \bigwedge_{0 \leq G < N \land G \neq I} \left( Q[I] = 0 \land Q'[I] = 1 \land \bigwedge \left( A'[I] > T[G] \land T'[I] = T[I] \right) \right] \land J(I, N) \right].
\]

Whenever a process \( 0 \leq I < N \) requests the access to the critical section, it changes its control state from idle (\( \{Q[I] = 0\} \)) to choose (\( \{Q[I] = 1\} \)) and generates a ticket stored in the auxiliary register which after the transition becomes greater than the tickets of all other processes in the system (\( \bigwedge_{0 \leq G < N \land G \neq I} (A'[I] > T[G]) \)).

- **Unsynchronized transition** \( \rho_2(N) \):

\[
\forall_{0 \leq I < N} \left[ (Q[I] = 1 \land Q'[I] = 2 \land T'[I] = A[I] \land A'[I] = A[I] \land J(I, N)) \right].
\]

A process \( 0 \leq I < N \) changes its state from choose (\( Q[I] = 1 \)) to wait (\( Q[I] = 2 \)) along with setting register \( T[I] \) to the value of the ticket generated in the choosing state and stored in the auxiliary register \( A[I] \). From that moment the new value of the process’ ticket becomes available for reading for other processes.

- **Synchronized transition** \( \rho_3(N) \):

\[
\forall_{0 \leq I < N} \left[ \bigwedge_{0 \leq G < N \land G \neq I} \left( Q[I] = 2 \land Q'[I] = 3 \land T[I] = T'[I] \land A'[I] = A[I] \land Q[G] \neq 1 \land \bigwedge \left( T[G] = 0 \lor T[I] < T[G] \land (T[I] = T[G] \land I < G) \right) \right] \land J(I, N) \right].
\]
A process can enter the critical section by changing its state from wait (\{Q[I] = 2\}) to use (\{Q'[I] = 3\}) if it has the smallest ticket number \(T[I]\) (ignoring the processes with the ticket equal to zero) or its index \(I\) is smaller than of other processes in case they have the same tickets. At the same time no process can be at the choose state during this transition (\(Q[G] \neq 1\)).

- Unsynchronized transition \(\rho_4(N)\):

\[
\bigvee_{0 \leq I < N} \left[ Q[I] = 3 \land Q'[I] = 0 \land T'[I] = 0 \land J(I, N) \right].
\]

A process leaves the critical section and resets its ticket to zero.

### B.2 Safety Property

We are interested to prove the property that no two processes can be in the critical section at the same time. The corresponding family of sets of safe states where the system size \(N\) is a parameter can be described by the following parameterized state schema:

\[
\bigwedge_{0 \leq I_1 < I_2 < N} (Q[I_1] \neq 3 \lor Q[I_2] \neq 3).
\]

The set of safe states is expressed as the following parameterized system descriptive formula \(p(I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A)\):

\[
0 \leq I_1 < I_2 < N \land \neg(Q_1 = 3 \land Q_2 = 3).
\]

### B.3 Environmental Relaxation

Let us perform the environmental relaxation of rank two because we used two parameters \(I_1\) and \(I_2\) to express the safety property. In the following we demonstrate how to relax transitions \(\rho_1, \rho_2, \rho_3\) and \(\rho_4\). The relaxation is parameterized by the reference set \(\{I_1, I_2\}\) and the resulting formulas divided into environment and reference parts for convenience based on the type of the moving component.

- \(\rho'_1(I_1, I_2, N)\) for environment processes:

\[
\bigvee_{0 \leq I < N \land I \neq I_1 \land I \neq I_2}
\]
APPENDIX B. LAMPORT'S BAKERY ALGORITHM

\[
\begin{align*}
\bigwedge_{0 \leq G < N \land G \neq I} & \quad \left( Q[I] = 0 \land Q'[I] = 1 \land T'[I] = T[I] \right) \\
& \quad \land (G = I_1 \rightarrow A'[I] > T[I_1]) \\
& \quad \lor (G = I_2 \rightarrow A'[I] > T[I_2]) \\
& \quad \lor (G \neq I_1 \land G \neq I_2) \\
& \quad \land \mathcal{J}(I, N)
\end{align*}
\]

- \( \rho'_1(I_1, I_2, N) \) for reference processes:

\[
\bigwedge_{0 \leq G < N \land G \neq I_1} \left( \begin{array}{c}
Q[I_1] = 0 \land Q'[I_1] = 1 \\
T'[I_1] = T[I_1] \land A'[I_1] > T[G]
\end{array} \right) \land \mathcal{J}(I_1, N)
\]

and

\[
\bigwedge_{0 \leq G < N \land G \neq I_2} \left( \begin{array}{c}
Q[I_2] = 0 \land Q'[I_2] = 1 \\
T'[I_2] = T[I_2] \land A'[I_2] > T[G]
\end{array} \right) \land \mathcal{J}(I_2, N)
\]

- \( \rho'_2(I_1, I_2, N) \) for environment processes:

\[
\bigvee_{0 \leq I < N \land I \neq I_1 \land I \neq I_2} \left[ \begin{array}{c}
Q[I] = 1 \land Q'[I] = 2 \land T'[I] = A[I] \\
A'[I] = A[I] \land \mathcal{J}(I, N)
\end{array} \right]
\]

- \( \rho'_2(I_1, I_2, N) \) for reference processes:

\[
Q[I_1] = 1 \land Q'[I_1] = 2 \land T'[I_1] = A[I_1] \land A'[I_1] = A[I_1] \land \mathcal{J}(I_1, N)
\]

and

\[
Q[I_2] = 1 \land Q'[I_2] = 2 \land T'[I_2] = A[I_2] \land A'[I_2] = A[I_2] \land \mathcal{J}(I_2, N)
\]

- \( \rho'_3(I_1, I_2, N) \) for environment processes:

\[
\bigvee_{0 \leq I < N \land I \neq I_1 \land I \neq I_2}
\]

124
\[ Q[I] = 2 \land Q'[I] = 3 \]
\[ G = I_1 \rightarrow (Q[I_1] \neq 1 \land T[I_1] = 0 \lor T[I] < T[I_1] \land T[I] = T[I_1] \land I < I_1 \lor G = I_2 \rightarrow (Q[I_2] \neq 1 \land T[I_2] = 0 \lor T[I] < T[I_2] \land T[I] = T[I_2] \land I < I_2 \lor (G \neq I_1 \land G \neq I_2) \land J(I, N)) \]
\[ J(I, N) \]

- \( \rho'_3(I_1, I_2, N) \) for reference processes:

\[ \bigwedge_{0 \leq G < N \land G \neq I_1} \left[ Q[I_1] = 2 \land Q'[I_1] = 3 \land T'[I_1] = T[I_1] \right] \land A'[I_1] = A[I_1] \land Q[G] \neq 1 \land (T[G] = 0 \lor T[I_1] < T[G] \lor (T[I_1] = T[G] \land I_1 < G)) \land J(I_1, N) \]
APPENDIX B. LAMPORT’S BAKERY ALGORITHM

and

$$\bigwedge_{0 \leq G < N \land G \neq I_2} \left[ Q[I_2] = 2 \land Q'[I_2] = 3 \land T'[I_2] = T[I_2] \land A'[I_2] = A[I_2] \land Q[G] \neq 1 \land (T[G] = 0 \lor T[I_2] < T[G] \lor (T[I_2] = T[G] \land I_2 < G)) \right] \land \mathcal{J}(I_2, N)$$

- \(\rho'_4(I_1, I_2, N)\) for environment processes:

$$\bigvee_{0 \leq I < N \land I \neq I_1 \land I \neq I_2} [ Q[I] = 2 \land Q'[I] = 0 \land T'[I] = 0 \land A'[I] = A[I] \land \mathcal{J}(I, N)]$$

- \(\rho'_4(I_1, I_2, N)\) for reference processes:

$$Q[I_1] = 2 \land Q'[I_1] = 0 \land T'[I_1] = 0 \land A'[I_1] = A[I_1] \land \mathcal{J}(I_1, N)$$

and

$$Q[I_2] = 2 \land Q'[I_2] = 0 \land T'[I_2] = 0 \land A'[I_2] = A[I_2] \land \mathcal{J}(I_2, N)$$

We also need to transform state formula \(F(N)\) into a corresponding pSDF \(f_0(I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A)\):

$$F(N) = I_1 < I_2 \land 0 \leq I_1 < N \land Q_1 = 0 \land T_1 = 0 \land A_1 = 0$$

$$\land$$

$$0 \leq I_2 < N \land Q_2 = 0 \land T_2 = 0 \land A_2 = 0$$

$$\land$$

$$0 \leq I < N \land I \neq I_1 \land I \neq I_2 \land Q = 0 \land T = 0 \land A = 0.$$ 

B.4 Reachability Analysis

B.4.1 First Approach

Reachability analysis according to the algorithm \(\text{REACHABLE}\) (Figure 5.1) consists of

- parallel or aggregate parallel application of meta-transitions described by the transition schemas above to the the currently explored set of reachable states described by a pSDF \(f(I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A)\) (which initially is equal to \(f_0(I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A)\)), and
• merging the result with the current value of \( f(I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A) \)

until we reach a fixed-point, that is, when no application of any meta-transition can produce anything new. All the necessary operations on pSDFs are described in Section 5.

For example, the result of aggregate parallel application of \( \tau = [\rho'_1(I_1, I_2, N)] \) to the set of global states described by a pSDF \( f(I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A) \) applied only to reference processes according to Theorem 5.4.7 is expressed by the following pSDF \( f'(I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A) \):

\[
\exists Q', T', A' \left[ f(I_1, Q_1', T_1', A_1', I_2, Q_2, T_2, A_2, I, Q, T, A) \land \left( (Q_1 = Q_1' \land T_1 = T_1' \land A_1 = A_1') \lor (Q_1' = 0 \land Q_1 = 1 \land T_1 = T_1' \land A_2 > A_1 \land A_1 > A) \right) \right] \\
\lor \exists Q', T', A' \left[ f(I_1, Q_1, T_1, A_1, I_2, Q_2', T_2', A_2', I, Q, T, A) \land \left( (Q_2 = Q_2' \land T_2 = T_2' \land A_2 = A_2') \lor (Q_2' = 0 \land Q_2 = 1 \land T_2 = T_2' \land A_1 > A_2 \land A_2 > A) \right) \right].
\]

### B.4.2 Second Approach

To apply the algorithm presented in Section 5.5.2 we construct the transition relation with environmental relaxation for the aggregate parallel application of all transitions in the system. The Omega Calculator [45] script illustrated below represents such a verification of Simplified Bakery Algorithm.

The set of bad states \( P \) (Lines 21-23) and the set of initial states \( R_i \) (Lines 26-28) are expressed as Presburger relations on variables \( I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A \), where variables \( I_1 \) and \( I_2 \) represent indexes of reference components while \( Q_1, T_1, A_1 \) and \( Q_2, T_2, A_2 \) their respective local states. Variable \( I \) is used to represent environment components' indexes while variables \( Q, T, A \) - their local states.

The over-approximated transition relation \( Tr \) (Lines 32-104) is a union of all encoded meta-transitions. In what follows, we iteratively apply the transition relation \( Tr \) to the set relation \( R \) describing the current set of explored states (Line 128 in Iteration 1) and filter the result afterwards (Lines 115-123 in Iteration 1). We check if the fixed-point is reached after each iteration (Line 124 in Iteration 1). It is required 13 iterations to reach the fixed-point, which was determined empirically because of the limited script language. Obviously, it can be automated as well.
After the fixed point is reached, we make the intersection of the (over-approximated) set of reachable states with the set of bad states (Lines 310-311) and print the result (Line 314).
Bakery

# Lamport's Bakery Algorithm

1 # Described in "Parameterized Verification of Infinite-state
2 # Processes with Global Conditions"
3 # by Parosh, Delzanno, Rezine, B.2, Figure 3

6 # the system size
7 symbolic N;

9 # Registers: I,Q,T,A:
10 #
11 # I: indexes, 0<=I<N
12 # Q: control state:
13 # 0 - idle
14 # 1 - choose
15 # 2 - wait
16 # 3 - use (critical version)
17 # T: ticket number, integer
18 # A: auxiliary variable, integer

20 # The set of bad states (the negation of the property)
21 Bad :=
22 {{I1,3,T1,A1, I2,3,T2,A2, I,Q,T,A} : 
23 0<=I1,I2<N && I1<I2 && !(I=I2)};

25 # The set of initial states
26 Ri :=
27 {{I1,0,0,0, I2,0,0,0, I1,0,0,0} :
28 0<=I1,I2<N && I1<I2 && !(I=I2)};

31 # The transition relation
32 Tr :=
33 # t1 for environment
34 {{I1, Q1,T1,A1, I2, Q2,T2,A2, I,Q,T,A} ->
35 [I1, Q1,T1,A1, I2, Q2,T2,A2, I,1,T,A']:
36 T1,T2<A'}
37 union
38 # t1 for I1
39 {{I1,0,T1,A1, I2, Q2,T2,A2, I,Q,T,A} ->
40 [I1,1,T1,A', I2, Q2,T2,A2, I,Q,T,A]:
41 T,T2<A1'}
42 union
43 # t1 for I2
44 {{I1, Q1,T1,A1, I2,0,T2,A2, I,Q,T,A} ->
45 [I1, Q1,T1,A1, I2,1,T2,A2', I,Q,T,A]:
46 T1,T<A2'}
47 union
48 # t2 for environment
APPENDIX B. LAMPORT’S BAKERY ALGORITHM

49 \{ [I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, 1, T, A] \} \rightarrow
50 \{ [I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, 2, A, A] \}
51 \text{union}
52 \text{# t2 for I1}
53 \{ [I_1, 1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A] \} \rightarrow
54 \{ [I_1, 2, A_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A] \}
55 \text{union}
56 \text{# t2 for I2}
57 \{ [I_1, Q_1, T_1, A_1, I_2, 1, T_2, A_2, I, Q, T, A] \} \rightarrow
58 \{ [I_1, Q_1, T_1, A_1, I_2, 2, A_2, A_2, I, Q, T, A] \}
59 \text{union}
60 \text{# t3 for environment}
61 \{ [I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, 3, T, A] \} :
62 \quad !(Q_1=1) \land \lnot !(Q_2=1)
63 \quad \land \quad (T_1=0 \lor T < T_1 \lor (T=T_1 \land I < I_1))
64 \quad \land \quad (T_2=0 \lor T < T_2 \lor (T=T_2 \land I < I_2))
65 \text{union}
66 \text{# t3 for I1}
67 \{ [I_1, 1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A] \} \rightarrow
68 \{ [I_1, 3, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A] \} :
69 \quad !(Q_1=1) \land \lnot !(Q=1)
70 \quad \land \quad (T=0 \lor T < T_1 \lor (T=T_1 \land I < I_1))
71 \quad \land \quad (T_2=0 \lor T < T_2 \lor (T=T_2 \land I < I_2))
72 \text{union}
73 \text{# t3 for I2}
74 \{ [I_1, Q_1, T_1, A_1, I_2, 3, T_2, A_2, I, Q, T, A] \} :
75 \quad !(Q=1) \land \lnot !(Q_1=1)
76 \quad \land \quad (T_1=0 \lor T < T_2 \lor (T=T_2 \land I < I_2))
77 \quad \land \quad (T_2=0 \lor T < T_1 \lor (T=T_1 \land I < I_1))
78 \text{union}
79 \text{# t4 for environment}
80 \{ [I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, 3, T, A] \} \rightarrow
81 \{ [I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, 0, 0, A] \}
82 \text{union}
83 \text{# t4 for I1}
84 \{ [I_1, 3, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A] \} \rightarrow
85 \{ [I_1, 0, 0, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A] \}
86 \text{union}

130
\textbf{APPENDIX B. LAMPORT'S BAKERY ALGORITHM}

\begin{verbatim}
98  \# t4 for I2
99  \{ [I1, Q1, T1, A1, I2, 3, T2, A2, I, Q, T, A] \rightarrow
100     [I1, Q1, T1, A1, I2, 0, 0, A2, I, Q, T, A] \}
101 union
102 \# the idling condition
103 \{ [I1, Q1, T1, A1, I2, Q2, T2, A2, I, Q, T, A] \rightarrow
104     [I1, Q1, T1, A1, I2, Q2, T2, A2, I, Q, T, A] \};

107nst
108 \# Fixed-point computation
109nst

111nst \textit{iteration 1}
112 R := R;
113 R' := range (Tr \ R);
114 Rf := R' \ complement range (\{ [I1, Q1, T1, A1, I2, Q2, T2, A2, I] \rightarrow
115     [I1, Q1, T1, A1, I2, Q2, T2, A2, I', Q', T', A'] : 0<=I<N && !(I=I1) && !(I=I2) \}
116 \}
117 \) ;
118 \textit{iteration 2}
119 R := Rf;
120 R' := range (Tr \ R);
121 Rf := R' \ complement range (\{ [I1, Q1, T1, A1, I2, Q2, T2, A2, I] \rightarrow
122     [I1, Q1, T1, A1, I2, Q2, T2, A2, I] \} \ R')
123 ) ;
124 \textit{iteration 3}
125 R := Rf;
126 R' := range (Tr \ R);
127 Rf := R' \ complement range (\{ [I1, Q1, T1, A1, I2, Q2, T2, A2, I] \rightarrow
128     [I1, Q1, T1, A1, I2, Q2, T2, A2, I] \} \ R')
129 ) ;
130 \end{verbatim}
APPENDIX B. LAMPORT’S BAKERY ALGORITHM

\[ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I', Q', T', A' \] :
\[ 0 \leq I < N \land (I=I_1) \land (I=I_2) \]
\[ \text{complement range} \left\{ \left\{ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I', Q', T', A' \right\} -> \right. \]
\[ \left. \left\{ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I \right\} \setminus R' \right) ; \]
\[ R_f \subset R; \]

# # # # # # # # # # # # # # # # # # # # # # # iteration 4
\[ R := R_f; \]
\[ R' := \text{range} \ (\text{Tr} \ \setminus \ R); \]
\[ R_f := \text{intersection complement range} \left( \right. \]
\[ \left\{ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I \right\} \rightarrow \]
\[ \left\{ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I', Q', T', A' \right\} : \]
\[ 0 \leq I < N \land (I=I_1) \land (I=I_2) \]
\[ \text{complement range} \left\{ \left\{ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A \right\} \setminus \right. \]
\[ \left. \left\{ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I \right\} \right) ; \]
\[ R_f \subset R; \]

# # # # # # # # # # # # # # # # # # # # # # # iteration 5
\[ R := R_f; \]
\[ R' := \text{range} \ (\text{Tr} \ \setminus \ R); \]
\[ R_f := \text{intersection complement range} \left( \right. \]
\[ \left\{ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I \right\} \rightarrow \]
\[ \left\{ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I', Q', T', A' \right\} : \]
\[ 0 \leq I < N \land (I=I_1) \land (I=I_2) \]
\[ \text{complement range} \left\{ \left\{ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A \right\} \setminus \right. \]
\[ \left. \left\{ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I \right\} \right) ; \]
\[ R_f \subset R; \]

# # # # # # # # # # # # # # # # # # # # # # # iteration 6
\[ R := R_f; \]
\[ R' := \text{range} \ (\text{Tr} \ \setminus \ R); \]
\[ R_f := \text{intersection complement range} \left( \right. \]
\[ \left\{ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I \right\} \rightarrow \]
\[ \left\{ I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I', Q', T', A' \right\} : \]
\[ 0 \leq I < N \land (I=I_1) \land (I=I_2) \]
\[ \text{complement} \]
APPENDIX B. LAMPORT'S BAKERY ALGORITHM

range (\{[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2,A_2, I_3,Q_3,T_3,A_3] \rightarrow
[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2,A_2, I_3] \} \setminus R')

R_f \subset R;

# # # # # # # # # # # # # # # # # # # # # # # iteration 7
R := R_f;
R' := range (Tr \ R);
R_f :=
R' \ intersection complement range (\{[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2, A_2, I_3] \rightarrow
[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2, A_2, I_3', Q_3', T_3', A_3'] : 0 \leq I < N \&\& !(I=I_1) \&\& !(I=I_2)}
\setminus complement
range (\{[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2, A_2, I_3] \rightarrow
[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2, A_2, I_3] \} \setminus R')
R_f \subset R;

# # # # # # # # # # # # # # # # # # # # # # # iteration 8
R := R_f;
R' := range (Tr \ R);
R_f :=
R' \ intersection complement range (\{[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2, A_2, I_3] \rightarrow
[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2, A_2, I_3', Q_3', T_3', A_3'] : 0 \leq I < N \&\& !(I=I_1) \&\& !(I=I_2)}
\setminus complement
range (\{[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2, A_2, I_3] \rightarrow
[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2, A_2, I_3] \} \setminus R')
R_f \subset R;

# # # # # # # # # # # # # # # # # # # # # # # iteration 9
R := R_f;
R' := range (Tr \ R);
R_f :=
R' \ intersection complement range (\{[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2, A_2, I_3] \rightarrow
[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2, A_2, I_3', Q_3', T_3', A_3'] : 0 \leq I < N \&\& !(I=I_1) \&\& !(I=I_2)}
\setminus complement
range (\{[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2, A_2, I_3] \rightarrow
[I_1,Q_1,T_1,A_1, I_2,Q_2,T_2, A_2, I_3] \} \setminus R')
R_f \subset R;
APPENDIX B. LAMPORT’S BAKERY ALGORITHM

# iteration 10
R := Rf;
R’ := range (Tr \ R);
Rf :=

R’ intersection complement range(

{ [I1, Q1, T1, A1, I2, Q2, T2, A2, I] \ ->
  [I1, Q1, T1, A1, I2, Q2, T2, A2, I’, Q’, T’, A’] :
    0<=I<N && !(I=I1) && !(I=I2)}
\ complement
range ( { [I1, Q1, T1, A1, I2, Q2, T2, A2, I, Q, T, A] ->
  [I1, Q1, T1, A1, I2, Q2, T2, A2, I]} \ R’)
);
Rf subset R;

# iteration 11
R := Rf;
R’ := range (Tr \ R);
Rf :=

R’ intersection complement range(

{ [I1, Q1, T1, A1, I2, Q2, T2, A2, I] \ ->
  [I1, Q1, T1, A1, I2, Q2, T2, A2, I’, Q’, T’, A’] :
    0<=I<N && !(I=I1) && !(I=I2)}
\ complement
range ( { [I1, Q1, T1, A1, I2, Q2, T2, A2, I, Q, T, A] ->
  [I1, Q1, T1, A1, I2, Q2, T2, A2, I]} \ R’)
);
Rf subset R;

# iteration 12
R := Rf;
R’ := range (Tr \ R);
Rf :=

R’ intersection complement range(

{ [I1, Q1, T1, A1, I2, Q2, T2, A2, I] \ ->
  [I1, Q1, T1, A1, I2, Q2, T2, A2, I’, Q’, T’, A’] :
    0<=I<N && !(I=I1) && !(I=I2)}
\ complement
range ( { [I1, Q1, T1, A1, I2, Q2, T2, A2, I, Q, T, A] ->
  [I1, Q1, T1, A1, I2, Q2, T2, A2, I]} \ R’)
);
Rf subset R;

# iteration 13
R := Rf;
R’ := range (Tr \ R);
APPENDIX B. LAMPORT'S BAKERY ALGORITHM

294 \text{R}_f :=
295 \text{R}' \text{ intersection complement range (}
296 \quad \{ [I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I] \rightarrow
297 \quad [I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I', Q', T', A'] \}:
298 \quad 0 < I < N && \neg (I = I_1) && \neg (I = I_2)
299 \quad \text{complement range} \left( \{ [I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I, Q, T, A] \rightarrow
300 \quad [I_1, Q_1, T_1, A_1, I_2, Q_2, T_2, A_2, I] \} \setminus \text{R}' \right)
301 \};
302 \text{R}_f \subset \text{R};
303
304 \# The fixed-point is reached
305 \# \text{(determined empirically)}
306 \text{ReachedBadStates} :=
307 \text{R intersection Bad;}
308
309 \# print relation describing reached bad states
310 \text{ReachedBadStates};
Appendix C

Simplified Szymanski Mutual Exclusion Algorithm

This section considers a simplified version of Szymanski mutual exclusion algorithm originally presented in [63] and modified in [23] by adding atomicity assumptions.

C.1 System Definition

Let us first define a family of uniform finite state transition systems \( \{X_n\}_{n \in \mathbb{N}}, \{S_n\}_{n \in \mathbb{N}}, F(N), \rho(N) \). We assume that there is at least three processes in the family. Each process has only one register \( Q \):

\[
\{X_n\}_{n \geq 3} = \{Q[0], \ldots, Q[n - 1]\}_{n \geq 3}
\]

which \( Q \) ranges over seven control states:

- \( 0 - idle \): the process is in the non-critical section and has not requested the access to the critical section;
- \( 1, \ldots, 6 \): the process has requested the access to the critical section and is in the course of obtaining it.
- \( 7 - use \): the process is in the critical section.

Therefore, \( \{S_n\}_{n \geq 3} \) is the family of sets of all possible assignments \( s \in S_n \) for each \( n \geq 3 \) such that \( \{s(Q[i]) \in \{0, 1, 2, 3, 4, 5, 6, 7\} \}_{n \geq 3} \) where \( 0 \leq i < n \).

The set of initial states is described by the following parameterized state schema \( F(N) \):

\[
N \geq 3 \land \bigwedge_{0 \leq I < N} Q[I] = 0
\]

136
which states that initially all processes are in the idle state.

The transition relation described by transition schema \( \rho(N) = \rho_1(N) \lor \ldots \lor \rho_9(N) \) consists of the following nine meta-transitions.

- **Unsynchronized transition \( \rho_1(N) \):**
  \[
  \bigvee_{0 \leq I < N} (Q[I] = 0 \land Q'[I] = 1 \land J(I, N)).
  \]

  Whenever a process \( 0 \leq I < N \) wish to obtain the access to the critical section, it changes first its control state from idle \((\{Q[I] = 0\})\) to trying \((\{Q'[I] = 1\})\).

- **Synchronized transition \( \rho_2(N) \):**
  \[
  \bigvee_{0 \leq I < N} \left[ \bigwedge_{0 \leq G < N \land G \neq I} \left( \begin{array}{c}
  Q[I] = 1 \land Q'[I] = 2 \\
  (0 \leq Q[G] \leq 2 \lor Q[G] = 4)
  \end{array} \right) \land J(I, N) \right].
  \]

- **Unsynchronized transition \( \rho_3(N) \):**
  \[
  \bigvee_{0 \leq I < N} (Q[I] = 2 \land Q'[I] = 3 \land J(I, N)).
  \]

- **Synchronized transition \( \rho_4(N) \):**
  \[
  \bigvee_{0 \leq I < N} \left[ \bigwedge_{0 \leq G < N \land G \neq I} \left( \begin{array}{c}
  Q[I] = 3 \land Q'[I] = 5 \\
  Q[G] \neq 1 \land Q[G] \neq 2
  \end{array} \right) \land J(I, N) \right].
  \]

- **Synchronized transition \( \rho_5(N) \):**
  \[
  \bigvee_{0 \leq I < N} \left[ \bigwedge_{0 \leq G < N \land G \neq I} \left( \begin{array}{c}
  Q[I] = 5 \land Q'[I] = 6 \\
  Q[G] \neq 2 \land Q[G] \neq 3 \land Q[G] \neq 4
  \end{array} \right) \land J(I, N) \right].
  \]

- **Synchronized transition \( \rho_6(N) \):**
  \[
  \bigvee_{0 \leq I < N} \left[ \bigwedge_{0 \leq G < N \land G \neq I} \left( \begin{array}{c}
  Q[I] = 6 \land Q'[I] = 7 \\
  ((G < I) \rightarrow 0 \leq Q[G] \leq 2)
  \end{array} \right) \land J(I, N) \right].
  \]

- **Unsynchronized transition \( \rho_7(N) \):**
  \[
  \bigvee_{0 \leq I < N} (Q[I] = 7 \land Q'[I] = 0 \land J(I, N)).
  \]

137
• A transition $\rho_8(N)$:

$$\bigvee_{0 \leq I,J < N \land J \neq I} \left( \begin{array}{c}
Q[I] = 3 \land Q'[I] = 4 \\
\land \\
(Q[J] = 1 \lor Q[J] = 2) \\
\land \\
J(I,N)
\end{array} \right).$$

A process $I$ can make transition from the state $Q[I] = 3$ to $Q[I] = 4$ if there is another process $J \neq I$ such that $Q[J] = 1$ or $Q[J] = 2$.

• A transition $\rho_9(N)$:

$$\bigvee_{0 \leq I,J < N \land J \neq I} \left( \begin{array}{c}
Q[I] = 4 \land Q'[I] = 5 \\
\land \\
5 \leq Q[J] \leq 7 \land J(I,N)
\end{array} \right).$$

A process $I$ can make transition from the state $Q[I] = 4$ to $Q[I] = 5$ if there is another process $J \neq I$ such that $Q[J] = 5$ or $Q[J] = 6$ or $Q[J] = 7$.

Transitions $\rho_8$ and $\rho_9$ express the existential quantification over all processes ("a process $I$ can make a move if there exists a process $J$ such that ...") in contrast to synchronized transitions expressing the universal quantification ("a process $I$ can make a move if for all processes $G$ ... "). The method represented in the thesis does explain how to deal with such transitions, therefore in what follows we introduce additional relaxation of $\rho_8$ and $\rho_9$ and make them unsynchronized:

• Unsynchronized $\rho_8(N)$:

$$\bigvee_{0 \leq I,J < N \land J \neq I} \left( Q[I] = 3 \land Q'[I] = 4 \land J(I,N) \right).$$

• Unsynchronized $\rho_9(N)$:

$$\bigvee_{0 \leq I,J < N \land J \neq I} \left( Q[I] = 4 \land Q'[I] = 5 \land J(I,N) \right).$$

### C.2 Safety Property

We are interested to prove the property that no two processes can be in the critical section at the same time. The corresponding family of sets of safe states where the system size $N$ is a parameter can be described by the following parameterized state schema:

$$\bigwedge_{0 \leq I_1 < I_2 < N} (Q[I_1] \neq 7 \lor Q[I_2] \neq 7).$$
The set of safe states is expressed as the following parameterized system descriptive formula \( p(I_1, Q_1, I_2, Q_2, I, Q) \):

\[
0 \leq I_1 < I_2 < N \land \neg (Q_1 = 7 \land Q_2 = 7).
\]

### C.3 Environmental Relaxation

Let us perform the environmental relaxation of rank 2 because we used two parameters \( I_1 \) and \( I_2 \) to express the safety property. In the following we demonstrate how to relax transitions \( \rho_1, \rho_2, \rho_3 \) and \( \rho_4 \). The relaxation is parameterized by the reference set \( \{I_1, I_2\} \) and the resulting formulas divided into environment and reference parts for convenience based on the type of the moving component.

- \( \rho'_1(I_1, I_2, N) \) for environment processes:

\[
\bigvee_{0 \leq I < N \land I \neq I_1 \land I \neq I_2} (Q[I] = 0 \land Q'[I] = 1 \land \mathcal{J}(I, N))
\]

- \( \rho'_1(I_1, I_2, N) \) for reference processes:

\[
Q[I_1] = 0 \land Q'[I_1] = 1 \land \mathcal{J}(I_1, N)
\]

and

\[
Q[I_2] = 0 \land Q'[I_2] = 1 \land \mathcal{J}(I_2, N)
\]

- \( \rho'_2(I_1, I_2, N) \) for environment processes:

\[
\bigvee_{0 \leq I < N \land I \neq I_1 \land I \neq I_2} \bigwedge_{0 \leq G < N \land G \neq I} \left( \begin{array}{c}
Q[I] = 1 \land Q'[I] = 2 \\
\land \\
G = I_1 \\
\rightarrow \\
0 \leq Q[I_1] \leq 2 \\
\lor \\
Q[I_1] = 4 \\
\land \\
G = I_2 \\
\rightarrow \\
0 \leq Q[I_2] \leq 2 \\
\lor \\
Q[I_2] = 4 \\
\land \\
G \neq I_1 \land G \neq I_2 \\
\land \\
\mathcal{J}(I, N)
\end{array} \right)
\]
APPENDIX C. SIMPLIFIED SZYMANSKI MUTUAL EXCLUSION ALGORITHM

• \( \rho'_2(I_1, I_2, N) \) for reference processes:
  \[
  \bigwedge_{0 \leq G < N \land G \neq I_1} \left( Q[I_1] = 1 \land Q'[I_1] = 2 \land (0 \leq Q[G] \leq 2 \lor Q[G] = 4) \right) \land J(I_1, N)
  \]
  and
  \[
  \bigwedge_{0 \leq G < N \land G \neq I_2} \left( Q[I_2] = 1 \land Q'[I_2] = 2 \land (0 \leq Q[G] \leq 2 \lor Q[G] = 4) \right) \land J(I_2, N)
  \]

• \( \rho'_3(I_1, I_2, N) \) for environment processes:
  \[
  \bigvee_{0 \leq I < N \land I \neq I_1, I \neq I_2} (Q[I] = 2 \land Q'[I] = 3 \land J(I, N))
  \]

• \( \rho'_4(I_1, I_2, N) \) for reference processes:
  \[
  Q[I_1] = 2 \land Q'[I_1] = 3 \land J(I_1, N)
  \]
  and
  \[
  Q[I_2] = 2 \land Q'[I_2] = 3 \land J(I_2, N)
  \]

• \( \rho'_4(I_1, I_2, N) \) for environment processes:
  \[
  \bigwedge_{0 \leq G < N \land G \neq I} \left( Q[G] = 3 \land Q'[G] = 5 \land \begin{cases} G = I_1 \\ G \neq I_1, G \neq I_2 \end{cases} \right) \land J(I, N)
  \]
and
\[ \bigwedge_{0 \leq G < N \land G \neq I_0} \left( Q[I_2] = 3 \land Q'[I_2] = 5 \land Q[G] \neq 1 \land Q[G] \neq 2 \right) \land J(I_2, N) \]

- \( \rho'_5(I_1, I_2, N) \) for environment processes:

\[ \bigvee_{0 \leq I < N \land I \neq I_1 \land I \neq I_2} \bigwedge_{0 \leq G < N \land G \neq I} \left( Q[I] = 5 \land Q'[I] = 6 \land G = I_1 \land Q[I] \neq 2 \land Q[I] \neq 3 \land Q[I] \neq 4 \right) \land J(I, N) \]

- \( \rho'_5(I_1, I_2, N) \) for reference processes:

\[ \bigwedge_{0 \leq G < N \land G \neq I_1} \left( Q[I_1] = 5 \land Q'[I_1] = 6 \land Q[G] \neq 2 \land Q[G] \neq 3 \land Q[G] \neq 4 \right) \land J(I_1, N) \]

and
\[ \bigwedge_{0 \leq G < N \land G \neq I_2} \left( Q[I_2] = 5 \land Q'[I_2] = 6 \land Q[G] \neq 2 \land Q[G] \neq 3 \land Q[G] \neq 4 \right) \land J(I_2, N) \]
APPENDIX C. SIMPLIFIED SZYMANSKI MUTUAL EXCLUSION
ALGORITHM

- $\rho'_6(I_1, I_2, N)$ for environment processes:
  \[ \bigvee_{0 \leq I < N \land I \neq I_1 \land I \neq I_2} \left( Q[I] = 6 \land Q'[I] = 7 \land \left( \begin{array}{l} (G = I_1 \land I < I) \rightarrow (0 \leq Q[I] \leq 2) \\ (G = I_2 \land I < I) \rightarrow (0 \leq Q[I] \leq 2) \\ G \neq I_1 \land G \neq I_2 \end{array} \right) \right) \land J(I, N) \]

- $\rho'_6(I_1, I_2, N)$ for reference processes:
  \[ \bigwedge_{0 \leq G < N \land G \neq I_1} \left( Q[I_1] = 6 \land Q'[I_1] = 7 \land (G < I_1 \rightarrow 0 \leq Q[G] \leq 2) \right) \land J(I_1, N) \]
  and
  \[ \bigwedge_{0 \leq G < N \land G \neq I_2} \left( Q[I_2] = 6 \land Q'[I_2] = 7 \land (G < I_2 \rightarrow 0 \leq Q[G] \leq 2) \right) \land J(I_2, N) \]

- $\rho'_7(I_1, I_2, N)$ for environment processes:
  \[ \bigvee_{0 \leq I < N \land I \neq I_1 \land I \neq I_2} \left( Q[I] = 7 \land Q'[I] = 0 \land J(I, N) \right) \]

- $\rho'_7(I_1, I_2, N)$ for reference processes:
  \[ Q[I_1] = 7 \land Q'[I_1] = 0 \land J(I_1, N) \]
  and
  \[ Q[I_2] = 7 \land Q'[I_2] = 0 \land J(I_2, N) \]

- $\rho'_8(I_1, I_2, N)$ for environment processes:
  \[ \bigvee_{0 \leq J < N \land J \neq I_1 \land J \neq I_2} \left( Q[I] = 3 \land Q'[I] = 4 \land J(I, N) \right) \]

- $\rho'_8(I_1, I_2, N)$ for reference processes:
  \[ Q[I_1] = 3 \land Q'[I_1] = 4 \land J(I_1, N) \]
  and
  \[ Q[I_2] = 3 \land Q'[I_2] = 4 \land J(I_2, N) \]
and
\[ \bigvee_{0 \leq J < N \land J \neq I_2} (Q[I_2] = 3 \land Q'[I_2] = 4 \land \mathcal{J}(I_2, N)) \]

- \( \rho'_9(I_1, I_2, N) \) for environment processes:
\[ \bigvee_{0 \leq I < N \land I \neq I_1 \land I \neq I_2} (Q[I] = 4 \land Q'[I] = 5 \land \mathcal{J}(I, N)) \]

- \( \rho'_9(I_1, I_2, N) \) for reference processes:
\[ \bigvee_{0 \leq J < N \land J \neq I_1} (Q[I_1] = 4 \land Q'[I_1] = 5 \land \mathcal{J}(I_1, N)) \]
and
\[ \bigvee_{0 \leq J < N \land J \neq I_2} (Q[I_2] = 4 \land Q'[I_2] = 5 \land \mathcal{J}(I_2, N)) \]

We also need to transform state formula \( F(N) \) into a corresponding pSDF \( f_0(I_1, Q_1, I_2, Q_2, I, Q) \):

\[ F(N) \equiv I_1 < I_2 \land 0 \leq I_1 < N \land Q_1 = 0 \]
\[ \land \]
\[ 0 \leq I_2 < N \land Q_2 = 0 \]
\[ \land \]
\[ 0 \leq I < N \land I \neq I_1 \land I \neq I_2 \land Q = 0. \]

### C.4 Reachability Analysis

#### C.4.1 First Approach

Reachability analysis according to the algorithm \textsc{reachable} (Figure 5.1) consists of

- parallel or aggregate parallel application of meta-transitions described by the transition schemas above to the the currently explored set of reachable states described by a pSDF \( f(I_1, Q_1, I_2, Q_2, I, Q) \) (which initially is equal to \( f_0(I_1, Q_1, I_2, Q_2, I, Q) \)), and

- merging the result with the current value of \( f(I_1, Q_1, I_2, Q_2, I, Q) \) until we reach a fixed-point, that is, when no application of any meta-transition can produce anything new. All the necessary operations on pSDFs are described in Section 5.
APPENDIX C. SIMPLIFIED SZYMANSKI MUTUAL EXCLUSION ALGORITHM

For example, the result of aggregate parallel application of \( \tau = [\rho_2'(I_1, I_2, N)] \) to the set of global states described by a pSDF \( f(I_1, Q_1, I_2, Q_2, I, Q) \) applied only to reference processes according to Theorem 5.4.7 is expressed by the following pSDF \( f'(I_1, Q_1, I_2, Q_2, I, Q) \):

\[
\exists Q_1' \left[ \begin{array}{c}
  f(I_1, Q_1', I_2, Q_2, I, Q) \\
  (Q_1 = Q_1') \\
  (Q_1 = 1 \land Q_1' = 2 \land 0 \leq Q_2 \leq 2 \land 0 \leq Q \leq 2)
\end{array} \right] \\
\lor \\
\exists Q_2' \left[ \begin{array}{c}
  f(I_1, Q_1, I_2, Q_2', I, Q) \\
  (Q_2 = Q_2') \\
  (Q_2 = 1 \land Q_2' = 2 \land 0 \leq Q_1 \leq 2 \land 0 \leq Q \leq 2)
\end{array} \right].
\]

C.4.2 Second Approach

To apply the algorithm presented in Section 5.5.2 we construct the transition relation with environmental relaxation for the aggregate parallel application of all transitions in the system. The Omega Calculator [45] script illustrated below represents such a verification of Simplified Bakery Algorithm.

The set of bad states \( \text{Bad} \) (Lines 9-15) and the set of initial states \( \text{R}_i \) (Lines 4-6) are expressed as Presburger relations on variables \( I_1, Q_1, I_2, Q_2, I, Q \), where variables \( I_1 \) and \( I_2 \) represent indexes of reference components while \( Q_1 \) and \( Q_2 \) their respective local states. Variable \( I \) is used to represent environment components’ indexes while the variable \( Q \) - their local states.

The over-approximated transition relation \( \text{Tr} \) (Lines 19-207) is a union of all encoded meta-transitions. In what follows, we iteratively apply the transition relation \( \text{Tr} \) to the set relation \( \text{R} \) describing the current set of explored states (for example, Line 219 in Iteration 1) and filter the result afterwards (Lines 222-225 in Iteration 1). We check if the fixed-point is reached after each iteration (Line 228 in Iteration 1). It is required 17 iterations to reach the fixed-point, which was determined empirically because of the limited script language. Obviously, it can be automated as well.

After the fixed point is reached, we make the intersection of the (over-approximated) set of reachable states with the set of bad states (Lines 511-512) and print the result (Line 516). If there is no intersection (in which case the relation \( \text{ReachedBadStates} \) is empty), then the property holds.
APPENDIX C. SIMPLIFIED SYMANSKI MUTUAL EXCLUSION
ALGORITHM

SzymanskiSimplified

1 symbolic I1, I2, N;

3 # The set of initial states
4 Ri :=
5 { [0, 0, I, 0] :
6 0 <= I1, I1, I2 < N && I1 < I2 && !(I1=I) && !(I2=I) };

8 # The set of the unsafe states
9 Bad :=
10 { [Q1, Q2, I, Q] :
11 Q1=7 && Q2=7
12 &&
13 0 <= I1 < I2 < N
14 &&
15 0 <= I < N && !(I=I1) && !(I=I2) };

17 # big transition relation
18 Tr :=
19 # rho1 for environment
20 { [Q1, Q2, I, Q] ->
21 [Q1, Q2, I, Q'] :
22 Q=0 && Q'=1
23 union
24 # rho2 for environment
25 { [Q1, Q2, I, Q] ->
26 [Q1, Q2, I, Q'] :
27 Q=1 && Q'=2
28 &&
29 (0 <= Q1 <= 2 || Q1=4)
30 &&
31 (0 <= Q2 <= 2 || Q2=4)
32 union
33 # rho3 for environment
34 { [Q1, Q2, I, Q] ->
35 [Q1, Q2, I, Q'] :
36 Q=2 && Q'=3
37 union
38 # rho4 for environment
39 { [Q1, Q2, I, Q] ->
40 [Q1, Q2, I, Q'] :
41 Q=3 && Q'=5
42 &&
43 !(Q1=1) && !(Q1=2)
44 &&
45 !(Q2=1) && !(Q2=2)
46 union
47 # rho5 for environment

145
APPENDIX C. SIMPLIFIED SZYMANSKI MUTUAL EXCLUSION ALGORITHM

\{ [Q_1, Q_2, I, Q] \rightarrow \\
[Q_1, Q_2, I, Q'] : \\
Q = 5 \&\& Q' = 6 \\
\&\& \\
!(Q_1 = 2) \&\& !(Q_1 = 3) \&\& !(Q_1 = 4) \\
\&\& \\
!(Q_2 = 2) \&\& !(Q_2 = 3) \&\& !(Q_2 = 4) \\
\}
union

# rho6 for environment
\{ [Q_1, Q_2, I, Q] \rightarrow \\
[Q_1, Q_2, I, Q'] : \\
Q = 6 \&\& Q' = 7 \\
\&\& \\
(1 < I_1 | | 0 < = Q_1 = 2) \\
\&\& \\
(1 < I_2 | | 0 < = Q_2 = 2) \\
\}
union

# rho7 for environment
\{ [Q_1, Q_2, I, Q] \rightarrow \\
[Q_1, Q_2, I, Q'] : \\
Q = 7 \&\& Q' = 0 \\
\}
union

# rho8 for environment
\{ [Q_1, Q_2, I, Q] \rightarrow \\
[Q_1, Q_2, I, Q'] : \\
Q = 3 \&\& Q' = 4 \\
\}
union

# rho9 for environment
\{ [Q_1, Q_2, I, Q] \rightarrow \\
[Q_1, Q_2, I, Q'] : \\
Q = 4 \&\& Q' = 5 \\
\}
union

# the idling condition
# (because we have AGGREGATE parallel application for everything)
\{ [Q_1, Q_2, I, Q] \rightarrow \\
[Q_1, Q_2, I, Q] \\
\}
union

# rho1 for I1
\{ [Q_1, Q_2, I, Q] \rightarrow \\
[Q_1', Q_2, I, Q] : \\
Q_1 = 0 \&\& Q_1' = 1 \\
\}
union

# rho1 for I2
\{ [Q_1, Q_2, I, Q] \rightarrow \\
[Q_1, Q_2', I, Q] : \\
Q_2 = 0 \&\& Q_2' = 1 \\
\}
union

# rho2 for I1


APPENDIX C. SIMPLIFIED SZYMANSKI MUTUAL EXCLUSION ALGORITHM

98 \{ \{Q1, Q2, I, Q\} ->
99 \{Q1', Q2, I, Q\} : 
100 Q1=1 && Q1'=2
101 &&
102 (0<=Q<=2 || Q=4)
103 &&
104 (0<=Q2<=2 || Q2=4)
105 union
106 # rho2 for I2
107 \{ \{Q1, Q2, I, Q\} ->
108 \{Q1, Q2', I, Q\} : 
109 Q2=1 && Q2'=2
110 &&
111 (0<=Q<=2 || Q=4)
112 &&
113 (0<=Q1<=2 || Q1=4)
114 union
115 # rho3 for I2
116 \{ \{Q1, Q2, I, Q\} ->
117 \{Q1', Q2, I, Q\} : 
118 Q1=2 && Q1'=3
119 &&
120 \{ \{Q1, Q2, I, Q\} ->
121 \{Q1, Q2', I, Q\} : 
122 Q2=3 && Q2'=5
123 &&
124 # rho4 for I2
125 \{ \{Q1, Q2, I, Q\} ->
126 \{Q1', Q2, I, Q\} : 
127 Q1=3 && Q1'=5
128 &&
129 ! (Q=1) && ! (Q=2)
130 &&
131 ! (Q2=1) && ! (Q2=2)
132 union
133 # rho4 for I1
134 \{ \{Q1, Q2, I, Q\} ->
135 \{Q1', Q2, I, Q\} : 
136 Q2=3 && Q2'=5
137 &&
138 ! (Q=1) && ! (Q=2)
139 &&
140 ! (Q1=1) && ! (Q1=2)
141 union
142 # rho5 for I1
143 \{ \{Q1, Q2, I, Q\} ->
144 \{Q1', Q2, I, Q\} : 
145 Q1=5 && Q1'=6

147
APPENDIX C. SIMPLIFIED SYMANSKI MUTUAL EXCLUSION ALGORITHM

147    &&
148    !(Q=2) && !(Q=3) && !(Q=4)
149    &&
150    !(Q2=2) && !(Q2=3) && !(Q2=4)
151 union
152    # rho5 for I2
153   { [Q1, Q2, I, Q] \to
154      [Q1, Q2', I, Q] :
155      Q2=5 && Q2'=6
156      &&
157      !(Q=2) && !(Q=3) && !(Q=4)
158      &&
159      !(Q1=2) && !(Q1=3) && !(Q1=4)
160 union
161    # rho6 for I1
162   { [Q1, Q2, I, Q] \to
163      [Q1', Q2, I, Q] :
164      Q1=6 && Q1'=7
165      &&
166      0<=Q<=2
167      &&
168      0<=Q2<=2
169 union
170    # rho6 for I2
171   { [Q1, Q2, I, Q] \to
172      [Q1', Q2, I, Q] :
173      Q2=6 && Q2'=7
174      &&
175      0<=Q<=2
176      &&
177      0<=Q1<=2
178 union
179    # rho7 for I1
180   { [Q1, Q2, I, Q] \to
181      [Q1', Q2, I, Q] :
182      Q1=7 && Q1'=0
183 union
184    # rho7 for I2
185   { [Q1, Q2, I, Q] \to
186      [Q1, Q2', I, Q] :
187      Q2=7 && Q2'=0
188 union
189    # rho8 for I1
190   { [Q1, Q2, I, Q] \to
191      [Q1', Q2, I, Q] :
192      Q1=3 && Q1'=4
193 union
194    # rho8 for I2
195   { [Q1, Q2, I, Q] \to

148
[Q1, Q2', I, Q] :
Q2=3 && Q2'=4
union
# rho9 for I1
{{Q1, Q2, I, Q} ->
|Q1', Q2, I, Q| :
Q1=4 && Q1'=5
union
# rho9 for I2
{{Q1, Q2, I, Q} ->
|Q1, Q2', I, Q| :
Q2=4 && Q2'=5};

# computation
# = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
# iteration 1
R := Ri;
# apply APA of the whole transition relation
# P is an auxiliary variable, needed to filter the result
P := Tr(R);
# Let us filter the result
P := P intersection complement
{{Q1, Q2, I} -> [Q1, Q2, I, Q] :
0 <= I < N && !(I=I1) && !(I=I2)}
complement {{Q1, Q2, I} -> [Q1, Q2, I]}(P));
# Let's check if we have reached the fixed-point
P subset R;

# iteration 2
R := P;
# apply APA of the whole transition relation
# P is an auxiliary variable, needed to filter the result
P := Tr(R);
# Let us filter the result
P := P intersection complement
{{Q1, Q2, I} -> [Q1, Q2, I, Q] :
0 <= I < N && !(I=I1) && !(I=I2)}
complement {{Q1, Q2, I} -> [Q1, Q2, I]}(P));
# Let's check if we have reached the fixed-point
APPENDIX C. SIMPLIFIED SZYMANSKI MUTUAL EXCLUSION
ALGORITHM

245  P subset R;
247  #----------------------------------------
248  # iteration 3
249  R := P;
251  # apply APA of the whole transition relation
252  # P is an auxiliary variable, needed to filter the result
253  P := Tr(R);
255  # Let us filter the result
256  P := P intersection complement
257  { [Q1, Q2, I] -> [Q1, Q2, I, Q] : 
258    0 <= I < N && !(I=I1) && !(I=I2) } ( 
259    complement { [Q1, Q2, I, Q] -> [Q1, Q2, I] } (P) );
261  # Let's check if we have reached the fixed-point
262  P subset R;
264  #----------------------------------------
265  # iteration 4
266  R := P;
268  # apply APA of the whole transition relation
269  # P is an auxiliary variable, needed to filter the result
270  P := Tr(R);
272  # Let us filter the result
273  P := P intersection complement
274  { [Q1, Q2, I] -> [Q1, Q2, I, Q] : 
275    0 <= I < N && !(I=I1) && !(I=I2) } ( 
276    complement { [Q1, Q2, I, Q] -> [Q1, Q2, I] } (P) );
278  # Let's check if we have reached the fixed-point
279  P subset R;
281  #----------------------------------------
282  # iteration 5
283  R := P;
285  # apply APA of the whole transition relation
286  # P is an auxiliary variable, needed to filter the result
287  P := Tr(R);
289  # Let us filter the result
290  P := P intersection complement
291  { [Q1, Q2, I] -> [Q1, Q2, I, Q] : 
292    0 <= I < N && !(I=I1) && !(I=I2) } ( 
293    complement { [Q1, Q2, I, Q] -> [Q1, Q2, I] } (P) );
APPENDIX C. SIMPLIFIED SZYMANSKI MUTUAL EXCLUSION ALGORITHM

295  # Let's check if we have reached the fixed-point
296  P subset R;
298  # -----------------------------------------------
299  # iteration 6
300  R := P;
302  # apply APA of the whole transition relation
303  # P is an auxiliary variable, needed to filter the result
304  P := Tr(R);
306  # Let us filter the result
307  P := P intersection complement
308  { [Q1, Q2, I] -> [Q1, Q2, I,Q] :
309    0=<I=<N & & !(I=11) & & !(I=12) }
310  complement ( [Q1, Q2, I,Q] -> [Q1, Q2, I,Q] ) (P);
312  # Let's check if we have reached the fixed-point
313  P subset R;
315  # -----------------------------------------------
316  # iteration 7
317  R := P;
319  # apply APA of the whole transition relation
320  # P is an auxiliary variable, needed to filter the result
321  P := Tr(R);
323  # Let us filter the result
324  P := P intersection complement
325  { [Q1, Q2, I] -> [Q1, Q2, I,Q] :
326    0=<I=<N & & !(I=11) & & !(I=12) }
327  complement ( [Q1, Q2, I,Q] -> [Q1, Q2, I,Q] ) (P);
329  # Let's check if we have reached the fixed-point
330  P subset R;
332  # -----------------------------------------------
333  # iteration 8
334  R := P;
336  # apply APA of the whole transition relation
337  # P is an auxiliary variable, needed to filter the result
338  P := Tr(R);
340  # Let us filter the result
341  P := P intersection complement
342  { [Q1, Q2, I] -> [Q1, Q2, I,Q] :
343    0=<I=<N & & !(I=11) & & !(I=12) }
344  complement ( [Q1, Q2, I,Q] -> [Q1, Q2, I,Q] ) (P);
APPENDIX C. SIMPLIFIED SZYMANSKI MUTUAL EXCLUSION ALGORITHM

\begin{verbatim}
343 0 \leq i < N \&\& ![i=I1] \&\& ![i=I2]} \{
344 \text{complement} \{[Q1, Q2, I, Q] \rightarrow [Q1, Q2, I]}(P)\};
346 \# \text{Let's check if we have reached the fixed-point}
347 P \text{ subset } R;
348
349 \# \text{iteration 9}
350 R := P;
351
353 \# \text{apply APA of the whole transition relation}
354 \# P is an auxiliary variable, needed to filter the result
355 P := Tr(R);
356
357 \# Let us filter the result
358 P := P \text{ intersection complement}
359 \{[Q1, Q2, I] \rightarrow [Q1, Q2, I]} \{
360 0 \leq i < N \&\& ![i=I1] \&\& ![i=I2]} \{
361 \text{complement} \{[Q1, Q2, I, Q] \rightarrow [Q1, Q2, I]}(P)\};
363 \# \text{Let's check if we have reached the fixed-point}
364 P \text{ subset } R;
366
367 \# \text{iteration 10}
368 R := P;
370 \# \text{apply APA of the whole transition relation}
371 \# P is an auxiliary variable, needed to filter the result
372 P := Tr(R);
374 \# Let us filter the result
375 P := P \text{ intersection complement}
376 \{[Q1, Q2, I] \rightarrow [Q1, Q2, I]} \{
377 0 \leq i < N \&\& ![i=I1] \&\& ![i=I2]} \{
378 \text{complement} \{[Q1, Q2, I, Q] \rightarrow [Q1, Q2, I]}(P)\};
380 \# \text{Let's check if we have reached the fixed-point}
381 P \text{ subset } R;
383
384 \# \text{iteration 11}
385 R := P;
387 \# \text{apply APA of the whole transition relation}
388 \# P is an auxiliary variable, needed to filter the result
389 P := Tr(R);
391 \# Let us filter the result
\end{verbatim}
APPENDIX C. SIMPLIFIED SZYMANSKI MUTUAL EXCLUSION
ALGORITHM

392 \( P := P \text{ intersection complement} \)
393 \( \{ [Q_1, Q_2, I] \rightarrow [Q_1, Q_2, I].Q \} : \)
394 \( 0 \leq I < N \land \neg(I = I_1) \land \neg(I = I_2) \} \)
395 \( \text{complement} \ \{ [Q_1, Q_2, I,Q] \rightarrow [Q_1, Q_2, I] \}(P) \); \)
397 \# Let's check if we have reached the fixed-point
398 \( P \text{ subset } R; \)
400 # ------------------------------------------
401 # iteration 12
402 \( R := P; \)
404 # apply APA of the whole transition relation
405 # \( P \) is an auxiliary variable, needed to filter the result
406 \( P := \text{Tr}(R); \)
408 # Let us filter the result
409 \( P := P \text{ intersection complement} \)
410 \( \{ [Q_1, Q_2, I] \rightarrow [Q_1, Q_2, I].Q \} : \)
411 \( 0 \leq I < N \land \neg(I = I_1) \land \neg(I = I_2) \} \)
412 \( \text{complement} \ \{ [Q_1, Q_2, I,Q] \rightarrow [Q_1, Q_2, I] \}(P) \); \)
414 \# Let's check if we have reached the fixed-point
415 \( P \text{ subset } R; \)
417 # ------------------------------------------
418 # iteration 13
419 \( R := P; \)
421 # apply APA of the whole transition relation
422 # \( P \) is an auxiliary variable, needed to filter the result
423 \( P := \text{Tr}(R); \)
425 # Let us filter the result
426 \( P := P \text{ intersection complement} \)
427 \( \{ [Q_1, Q_2, I] \rightarrow [Q_1, Q_2, I].Q \} : \)
428 \( 0 \leq I < N \land \neg(I = I_1) \land \neg(I = I_2) \} \)
429 \( \text{complement} \ \{ [Q_1, Q_2, I,Q] \rightarrow [Q_1, Q_2, I] \}(P) \); \)
431 \# Let's check if we have reached the fixed-point
432 \( P \text{ subset } R; \)
434 # ------------------------------------------
435 # iteration 14
436 \( R := P; \)
438 # apply APA of the whole transition relation
439 # \( P \) is an auxiliary variable, needed to filter the result
440 \( P := \text{Tr}(R); \)
442 \# \textit{Let us filter the result}
443 P := P \cap \left( \left\{ [Q_1, Q_2, I] \rightarrow [Q_1, Q_2, I, Q] : 0 < I < N \land \neg (I=I_1) \land \neg (I=I_2) \right\} \right) \cap \left( \left\{ [Q_1, Q_2, I, Q] \rightarrow [Q_1, Q_2, I] \right\} (P) \right);
444
445 \# \textit{Let's check if we have reached the fixed-point}
446 P \subseteq R;
447
448 \# \textit{Iteration 15}
449 R := P;
450
451 \# \textit{Apply APA of the whole transition relation}
452 \# \textit{P is an auxiliary variable, needed to filter the result}
453 P := Tr(R);
454
455 \# \textit{Let us filter the result}
456 P := P \cap \left( \left\{ [Q_1, Q_2, I] \rightarrow [Q_1, Q_2, I, Q] : 0 < I < N \land \neg (I=I_1) \land \neg (I=I_2) \right\} \right) \cap \left( \left\{ [Q_1, Q_2, I, Q] \rightarrow [Q_1, Q_2, I] \right\} (P) \right);
456
457 \# \textit{Let's check if we have reached the fixed-point}
458 P \subseteq R;
459
460 \# \textit{Iteration 16}
461 R := P;
462
463 \# \textit{Apply APA of the whole transition relation}
464 \# \textit{P is an auxiliary variable, needed to filter the result}
465 P := Tr(R);
466
467 \# \textit{Let us filter the result}
468 P := P \cap \left( \left\{ [Q_1, Q_2, I] \rightarrow [Q_1, Q_2, I, Q] : 0 < I < N \land \neg (I=I_1) \land \neg (I=I_2) \right\} \right) \cap \left( \left\{ [Q_1, Q_2, I, Q] \rightarrow [Q_1, Q_2, I] \right\} (P) \right);
468
469 \# \textit{Let's check if we have reached the fixed-point}
470 P \subseteq R;
471
472 \# \textit{Iteration 17}
473 R := P;
474
475 \# \textit{Apply APA of the whole transition relation}
476 \# \textit{P is an auxiliary variable, needed to filter the result}
477 P := Tr(R);
478
479 \# \textit{Let us filter the result}
480 P := P \cap \left( \left\{ [Q_1, Q_2, I] \rightarrow [Q_1, Q_2, I, Q] : 0 < I < N \land \neg (I=I_1) \land \neg (I=I_2) \right\} \right) \cap \left( \left\{ [Q_1, Q_2, I, Q] \rightarrow [Q_1, Q_2, I] \right\} (P) \right);
480
481 \# \textit{Let's check if we have reached the fixed-point}
482 P \subseteq R;
483
484 \# \textit{Iteration 18}
485 R := P;
486
487 \# \textit{Apply APA of the whole transition relation}
# P is an auxiliary variable, needed to filter the result
P := Tr(R);

# Let us filter the result
P := P intersection complement
{[Q1, Q2, I] -> [Q1, Q2, I, Q] :
 0 <= I < N & & !(I=I1) & & !(I=I2)}
complement {[Q1, Q2, I, Q] -> [Q1, Q2, I]}(P);

# Let's check if we have reached the fixed-point
P subset R;

# Now the fixed-point is reached
# (the number of iterations is
determined empirically)
#
#
#
# let us check if the property holds
# (There should be no intersection
# with the set of unsafe states)
ReachedBadStates :=
R intersection Bad;

# print relation describing reached bad states
# (Should be false)
ReachedBadStates;


| No 380 | Johan Ringström | Compiler Generation for Parallel Languages from Denotational Specifications, 1993. |
| No 441 | Lars Poignant | Informationsteknologi och företagsätt - Effekter på produktivitet och region, 1994. |
| No 476 | Bo Lagerström | Successiv resultatavräkning av pågående arbeten. - Fallstudier i tre byggföretag, 1995. |
| No 488 | Eva Toller | Contributions to Parallel Multiparadigm Languages: Combining Object-Oriented and Rule-Based Programming, 1995. |
| FHS 8/95 | Dan Fristedt | Metoder i användning - mot förbättring av systemutveckling genom situationell metodkunskap och metodanalys, 1995. |
| No 510 | Magnus Werner | Multidatabase Integration using Polymorphic Queries and Views, 1996. |
| No 538 | Staffan Flodin | Efficient Management of Object-Oriented Queries with Late Binding, 1996. |
| No 546 | Magnus Werner | Multidatabase Integration using Polymorphic Queries and Views, 1996. |
| No 550 | Kristina Larsen | Förutsättningar och begränsningar för arbeta på distans - erfarenheter från fyra svenska företag, 1996. |
| No 557 | Mikael Johansson | Quality Functions for Requirements Engineering Methods, 1996. |
| No 558 | Patrik Nordling | The Simulation of Rolling Bearing Dynamics on Parallel Computers, 1996. |
| No 561 | Anders Ekman | Exploration of Polygonal Environments, 1996. |
No 563 Niclas Andersson: Compilation of Mathematical Models to Parallel Code, 1996.
No 587 Jörgen Lindström: Chefs användning av kommunikationsteknik, 1996.
No 589 Esa Falkenroth: Data Management in Control Applications - A Proposal Based on Active Database Systems, 1996.
FiF-a 4 Carita Åhomb: Videomötesteknik i olika affärs situationer - möjligheter och hinder, 1997.
No 627 Fredrik Eklund: Declerative Error Diagnosis of GAPLog Programs, 1997.
No 653 Mats Gustafsson: Bringing Role-Based Access Control to Distributed Systems, 1997.
No 668 Per-Ove Zetterlund: Normering av svensk redovisning - En studie av tillkomsten av Redovisningsrådets rekommendation om koncernredovisning (RR01:91), 1998.
FiF-a 16 Marie-Therese Christiansson: Inter-organisatorisk verksamhetsutveckling - metoder som stöd vid utveckling av partnerskap och informationssystem, 1998.
No 742 Pawel Pietrzak: Static Incorrectness Diagnosis of CLP (FD), 1999.
No 766  Martin V. Howard: Designing dynamic visualizations of temporal data, 1999.
No 787  Charlotte Bjärkegren: Learning for the next project - Bearers and barriers in knowledge transfer within an organisation, 1999.
No 807  Svein Bergum: Managerial communication in telework, 2000.
FiF-a 32  Karin Hedström: Kunskapsanvändning och kunskapsutveckling hos verksamhetskonsulter - Erfarenheter från ett FOU-samarbete, 2000.
No 820  Jean Paul Meynard: Control of industrial robots through high-level task programming, 2000.
FiF-a 34  Göran Hultgren: Nätverksinriktad Förändringsanalys - perspektiv och metoder som stöd för förståelse och utveckling av affärsrelationer och informationssystem, 2000.
No 842  Magnus Kähl: The role of management control systems in strategic business units, 2000.
FiF-a 41  Benneth Christiansson: Att komponentbasera informationssystem - Vad säger teori och praktik?, 2000.
FiF-a 49  Stefan Holgersson: IT-system och filtrering av verksamhetskunskap - kvalitetsproblem vid analyser och beslutsfattande som bygger på uppgifter hämtade från polisens IT-system, 2001.
FiF-a 51  Per Oscarsson: Informationssäkerhet i verksamheter - begrepp och modeller som stöd för förståelse av informationssäkerhet och dess hantering, 2001.
No 931  Fredrik Elg: Ett dynamiskt perspektiv på individuella skillnader av heuristisk kompetens, intelligens, mentala modeller, mål och konfidenst i kontroll av mikrovärlden Moro, 2002.
No 938  Bourhane Kadmiry: Fuzzy Control of Unmanned Helicopter, 2002.
No 964  Peter Bunus: Debugging and Structural Analysis of Declarative Equation-Based Languages, 2002.
No 991 Almut Herzog: Secure Execution Environment for Java Electronic Services, 2002.
No 1084 Charlotte Stoltz: Calling for Call Centre Locations in a Swedish Rural Region, 2004.
FiF-a 74 Björn Johansson: Deciding on Using Application Service Provision in SMEs, 2004.
No 1095 Ulf Johansson: Rule Extraction - the Key to Accurate and Comprehensible Data Mining Models, 2004.
FiF-a 77 Ulf Larsson: Designarbete i dialog - karaktärisering av interaktionen mellan användare och utvecklare i en systemutvecklingsprocess, 2004.
No 1127 Per-Ola Kristensson: Large Vocabulary Shorthand Writing on Stylus Keyboard, 2004.
No 1149 Vaida Jakonienė: A Study in Integrating Multiple Biological Data Sources, 2005.
No 1156 Abid Rashid Mohamed: High-Level Techniques for Built-In Self-Test Resources Optimization, 2005.
No 1160 Adrian Pop: Contributions to Meta-Modeling Tools and Methods, 2005.
FiF-a 84 Jenny Lagteng: Verksamhetsutvecklande utvärdering i informationssystemprojekt, 2005.
No 1167 Christina Keller: Virtual Learning Environments in higher education. A study of students' acceptance of educational technology, 2005.
FiF-a 86 Jan Olausson: Att modellera uppslag - grunder för förståelse av processinriktade informationssystem i transaktionsintensiva verksamheter, 2005.
David Dinka: Role and Identity - Experience of technology in professional settings, 2005.
Andreas Hansson: Increasing the Storage Capacity of Recursive Auto-associative Memory by Segmenting Data, 2005.
Andreas Käll: Oversättningar av en managementmodell - En studie av införandet av Balanced Scorecard i ett landsting, 2005.
He Tan: Aligning and Merging Biomedical Ontologies, 2006.
Fif-a 91: Ett praktikperspektiv på hantering av mjukvarukomponenter, 2006.
Marten Karresand: Completing the Picture - Fragments and Back Again, 2008.
Miro Thorstensson: Using Observers for Model Based Data Collection in Distributed Tactical Operations, 2008.
Mohammad Saifullah: Exploring Biologically Inspired Interactive Networks for Object Recognition, 2011.
Anna Vapen: Contributions to Web Authentication for Untrusted Computers, 2011.
Kristian Staväker: Contributions to Parallel Simulation of Equation-Based Models on Graphics Processing Units, 2011.
Dag Sonntag: A Study of Chain Graph Interpretations, 2014.
Zlatan Dragicic: Completing the Is-a Structure in Description Logics Ontologies, 2014.