DoCS 90/21

Parosh Aziz Abdulla

Decision Problems in Systolic Circuit Verification

Ph. D. Thesis
Department of Computer Systems
Uppsala University
Uppsala, Sweden

ISSN 0283-0574

DoCS 90/21

Parosh Aziz Abdulla

Decision Problems in Systolic Circuit Verification

Ph. D. Thesis
Department of Computer Systems
Uppsala University
Uppsala, Sweden

ISSN 0283-0574

Doctoral Dissertation at Uppsala University 1990

Abstract

Parosh Aziz Abdulla, 1990, Decision Problems in Systolic Circuit Verification, DoCS 90/21, Uppsala University, ISSN 0283-0574

We present methods for automatic verification of several classes of systolic circuits.

A model for the formal description of implementations and specifications of systolic circuits is provided. The implementation description of a circuit reflects the computations performed inside the cells, the topology of the circuit, and the interconnection pattern among the cells of the circuit. The specification expresses the desired relation between the input and the output data of the circuit.

The mathematical operations defining the cell computations are interpreted as the operations of an algebra A. The ordered ring I of integers is used to define the notion of discrete time, and to describe the topology and the interconnection pattern of the circuit. By circuit verification over a class $\mathcal{K}$ of algebras we mean that we check whether or not the circuit implementation is correct with respect to the specification over I and each algebra A in $\mathcal{K}$.

Methods for automatic verification of three classes of circuits are described. Each class is characterized by the interpretation of the cell operations in the circuits of the class. The three classes considered have cell computations which are defined as the operations of a commutative ring, uninterpreted function symbols, and the operations of a boolean algebra. For each class, a nontrivial subclass is defined by imposing restrictions on the forms of the cell computations and circuit architectures. The verification problem is then reduced, using the properties of I and the class $\mathcal{K}$ of algebras on which the circuits operate, to a set of decidable problems over I and $\mathcal{K}$. We illustrate the verification methods by applying them to non-trivial circuits in the respective classes.

Examples of circuits which can be verified automatically by our methods include circuits for: convolution algorithms, matrix operations (such as matrix multiplication and transposition), string comparisons (such as substring detection, approximate string matching, and palindrome recognition), and implementation of digital filters.

Parosh Aziz Abdulla, Department of Computer Systems, Uppsala University, P.O.Box 520, S-751 20 Uppsala, Sweden, E-mail: parosh@docs.uu.se

Printed in Sweden by
Graphic Systems, Malmö, 1990

ISSN 0283-0574

بۆ یادگـاری دایکم و شـاهۆی پورزام

بۆ کوردسـتان

To the memory of my mother and Shaho

To my country Kurdistan

Acknowledgements

It has been a pleasure to be a graduate student under the competent and inspiring supervision of my advisor Stefan Arnborg.

I wish to thank my friend and teacher Bengt Jonsson for generously sharing his knowledge, and providing valuable advice on both personal and professional matters.

I have enjoyed discussing my work with John Tucker during his visits at our department. I am very grateful for his advice, and helpful comments.

I am much in debt to Björn Pehrson for giving me the opportunity to start my graduate studies. Special thanks to Steffen Weckner for his early support and encouragement.

Many thanks to Linda Arnold Christoff for reading and commenting earlier versions of this work, and to my graduate student colleague Ivan Christoff who has always been an inspiring discussion-partner.

I am grateful to all the people at the department of Computer Systems at Uppsala University, and in particular the chairman Hans Flack, for providing a creative environment in which it has been a pleasure to work.

Finally, my warmest thanks to Monica Wallenquist for putting up with my complaints, being cheerful, and helping me with various practical issues during my years at the department.

This research was partially supported by the Swedish National Board for Technical Development.

Contents

Chapter 1

Introduction

1.1 Systolic Circuits

The advent of VLSI has led to an increased interest in designing highly-parallel computing architectures in order to maximize performance and minimize cost. One type of such architecture is the systolic circuit [Kun82], which consists of a large number of processors (computation cells), each of which performs a limited amount of computation and contains very little storage. These processors are interconnected in a simple and regular pattern allowing communication between near neighbors. Such architectures allow us to minimize circuit parameters such as circuit area and power consumption. In addition, the regularity of the system and the simplicity of the processors make systematic design, analysis, and reasoning possible. Applications of systolic circuits can be found in digital signal processing, image processing, pattern matching, linear algebra, graph theory, etc. Prototype systolic circuits are available and are used especially in digital signal processing systems [FW87, ABC*87, AAG*86].

1.2 Verification

A rigorous method for establishing correctness of a system is to provide a formal model for describing the system structure and intended behaviour. In general when dealing with system correctness two descriptions of the system are involved: a *specification* and an *implementation*. The specification of the system describes how we expect the system to behave, while the implementation describes how the system is built. In the case of systolic circuits the specification is often a statement of how the output data of the circuit is related to the input data, and the implementation is a description of the behaviours of the processors and the interconnections among the processors inside the circuit.

Two important and related approaches for proving system correctness are *synthesis* and *verification*. In synthesis, a specification is given, and the synthesis system generates an implementation which is correct with respect to the specification. In verification the descriptions of the specification and the implementation are given. The task of verification is to check whether or not the implementation is correct with respect to the specification. The notions of system synthesis and verification are often related. Synthesis is usually carried out by applying a sequence of *transformations* to derive the implementation from the specification. These transformations must be verified in the sense that they must be

proved to preserve the correctness of the specification within the formal model. When an implementation is derived from a specification by synthesis then the implementation is verified by construction, i.e. the derived implementation is guaranteed to be correct with respect to the specification as the transformations preserve the correctness of the specification.

Formal synthesis and verification proofs can be accomplished by *hand*, be supported by *proof development systems*, or be completely mechanized by *decision methods*. By *hand methods* we mean proofs based on standard mathematical reasoning. Examples of hand synthesis and verification methods for systolic circuits include fixed point theory, process algebra, term rewriting techniques, etc. Recently *proof checkers* and *theorem provers* have been used to carry out systolic circuit verification. These are proof-supporting computer systems each of which is based on some established formal logic. A proof checker inputs a logical formula and a proof of validity of the formula, and checks whether the proof is correct or not. A theorem prover inputs a logical formula and tries to return a proof of validity or falsity of the formula. Examples of such systems are the Boyer-Moore theorem prover [BM79], the Nuprl proof development system (due to R. Constable) based on Martin-Löf type theory, and the HOL system (due to M. Gordon) [Gor87] based on the Church type theory. A *decision method* is given a statement in a theory, and checks whether or not the statement is valid in the theory. While for most theories there are no decision methods, sometimes nontrivial classes of problems can be found for which decision methods can be constructed. This is the main concern of this thesis. We will try to find nontrivial classes of systolic circuits for which we define decision methods to perform automatic verification.

1.3 Results

In this thesis we will provide:

- a formal model for systolic circuits.
- a formal definition of systolic circuit verification.
- methods for automatic verification of several nontrivial classes of systolic circuits.

A formal model for systolic circuits is a means of providing formal descriptions of implementations and specifications of these.

The implementation of a systolic circuit consists of a description of i) the computations performed inside the cells, which is given by a description of the relation of the output and input data of the cells, ii) the topology of the circuit, i.e. the manner in which the cells are placed inside the circuit, and iii) the pattern of interconnection inside the circuit, i.e. the manner in which the cells communicate and interchange computation results among each other. In this thesis we will consider synchronous circuits. We assume that we have a global clock which synchronizes the flow of data inside the circuit. Each cell has a number of inputs, local variables (registers), and outputs. At each clock cycle a number of computations are performed inside each cell on the data received from the inputs of the cell and the data stored inside the local variables of the cell. The results of computations are partially sent out via the outputs of the cell to the other cells, and partially stored inside the local variables of the cell.

We will use the notion of an *algebra* to interpret the cell computations inside the circuit. An *algebra* consists of a set of *domains* which defines the sets of data on which the cells operate, and a set of *operations* which define the meanings of the computations (mathematical operations) inside the cells. When we say that a circuit operates on e.g. a ring, then we mean that the data on which the cells compute are the elements of the ring, and the computations performed inside the cells are defined by means of the ordinary ring operators, $+$, $\cdot$, $-$, and 0. The regularity of the topology of systolic circuits makes it possible (as we will show in the description of our model) to use the integers to define the cell positions. This is achieved by implementing an *indexing* which assigns to each cell (in an n-dimensional circuit) a unique n-tuple of integers which defines the position of the cell. We will also use the integers to define the notion of time, where each clock cycle corresponds to one time step. Thus, when dealing with a circuit, we work with two algebras: i) the algebra on which the circuit operates, and ii) the integers which are used to define the notions of time and cell position. The implementation of a systolic circuit can be described by a set of recursive equations over the algebra on which the circuit operates and the integers. The equations arise from the descriptions of the cell computations, and the interconnection pattern of the cells inside the circuit. The boundary conditions of the equation system are the inputs to the circuit, i.e. the inputs to those cells which are not connected to outputs of other cells, and the initial values, i.e. the values stored in the circuit wires and cell registers at the start of circuit computations.

The specification of a circuit tells us which values certain wires and cell registers should have at certain time instants. The values demanded by the specification are also described as recursive equations over the algebra on which the circuit operates and the integers.

In order to perform formal verification of a circuit we must be able to compare the equation systems describing the implementation and specification of the circuit. We will use elementary fixed point theory to define a formal semantics for the equation systems which we get from the descriptions of the implementations and specifications of the circuits in our model. We define formally the notion of equality between two equation systems. Thus we can define the formal meaning of the verification of a circuit over the algebra on which it operates.

Automatic verification can be achieved by constructing decision methods to compare the implementation and specification equations of circuits. This is to be regarded as the main concern of this thesis. We will define several nontrivial classes of systolic circuits, and construct automatic verification methods for them. For each class we will give an example of how the automatic verification method can be applied to a nontrivial circuit in the class. In each case the automatic verification is accomplished by starting with the verification condition (i.e. the statement of correctness of the implementation with respect to the specification), and applying a sequence of rewriting rules (based on the axioms of the algebra of the circuit and the integers) to transform the verification condition into a set of formulas with the two properties that: i) checking the validity of each formula in the set is straightforward, and ii) the verification condition is valid iff each formula in the set is valid.

We will study the automatic verification of a class of circuits which operate over commutative rings. Examples of such circuits are found especially in the field of digital signal processing systems. The cell computations in the circuits of the class are defined by

the ring operators $+$, $\cdot$, $-$, and 0. We will investigate the problem of whether a circuit implementation is correct with respect to a specification for the class of all commutative rings. If a circuit is verified in this manner then the circuit is correctly implemented for any arbitrary interpretation of the ring operators in a commutative ring. We will also study the verification of circuits which operate on some particularly interesting commutative rings, such as the ring of integers, and the ring of natural numbers modulo m, for some fixed natural number m. The latter is particularly interesting in systolic circuit implementations since modular arithmetics allow bounding the sizes of the cell registers in the circuit.

Another problem which we will study is that of verification of circuits with uninterpreted function symbols. In this case we regard the cell operations as uninterpreted function symbols. We will perform the verification without taking into consideration the particular properties of the cell operations. Many times the particular properties of the cell operations do not affect the outcome of the verification. The verification problem nevertheless is still nontrivial. This is due to the fact that the verification involves dealing with the axioms of the integers which are used to model time and cell position, and because the recursive patterns of the equation systems defining the specification and implementation of the circuit are often quite complicated and deciding equivalence is difficult. If a circuit is verified in this manner then the circuit is correctly implemented with respect to the specification regardless of the particular interpretation of the cell operations. On the other hand, if the verification condition is not valid, then there may still be interpretations of the cell operations which make the implementation correct with respect to the specification.

Lastly we will study the verification problem of a class of circuits which compute over boolean algebras. The cell operations are defined by the boolean operators $\wedge$, $\vee$, $\neg$, true, and false. We will also consider a class of circuits where the cells operate on two disjoint sets of function symbols. The elements of the first set are interpreted as the operations of a boolean algebra, while the elements of the second set are left uninterpreted.

1.4 Related Work

Several techniques for systematic design and synthesis of systolic circuits are known. One method is that of manipulating data-dependency graphs. The main idea is to obtain a data-dependency graph of the specification (which is of the form of a system of recurrence equations) and transform it into an equivalent graph satisfying certain conditions which make it possible to provide a time-space mapping and hence a systolic implementation. Examples of this type of method are the works of Lisper [Lis88, Lis89], Moldovan [Mol82, Mol87], Quinton [Qui84, Qui86], and Rajopadhye [RF87, Raj89]. Huang and Lengauar [HL87, HL89] present a method where the specification is a sequential program. The method is based on deriving a sequential trace of the program and then transforming the sequential trace into a semantically equivalent parallel trace. The idea of the transformation is that semantically independent basic program statements can be executed in parallel. Finally a time-space mapping is provided from the parallel trace into a systolic architecture. Sheeran [She85, She86] presents a VLSI design language μFp in which higher order functions are used to give both graphic and semantic interpretations of the circuit. The language obeys a set of algebraic laws which are used to transform a

correct (but inefficient) specification into a systolic circuit implementation which is correct with respect to the specification. Further developments of the method can be found in [LJ88, Luk89].

Various approaches for the hand verification of systolic circuits have appeared in the literature. For example, Chen [Che83] uses systems of recursive equations to describe systolic circuits, and fixed point induction to show that an implementation meets a certain specification. Hennessy [Hen86] uses a language, derived from CCS [Mil89], to describe systolic circuits. Verification is done by applying a sequence of semantics-preserving transformations and fixed-point induction to derive the implementation from the specification. Gribomont [Gri88] uses a small language, derived from CSP [Hoa85], to describe systolic circuits. The semantics of the language allows the transformation of each circuit into an equivalent sequential program. The circuit correctness is then proved using invariant methods for sequential programs. Rem [Rem87] presents a methodology based on trace theory (which is related to CSP) for the design and verification of systolic circuits. Melhem and Rheinbold [MR84] suggest a mathematical model in which systolic circuits are described as systems of difference equations. The verification process is based on providing a solution to the system of equations, and then using induction to show that it realizes a certain specification. Tucker and Thompson [TT88] have defined a framework for synchronous concurrent algorithms, in which systems are modelled as networks composed of modules computing and communicating in parallel. Systolic circuits are a special case of these algorithms. Formalization of synchronous concurrent algorithms is done by simultaneous primitive and course of values recursive functions over abstract algebras. Case studies on hand verification of systolic circuits have been done within the framework [HTT88, MT87, ET89].

Of course, any systematic method should have machine support, either customized to the method, or based on general proof development systems for logics. Purushothaman and Subrahmanyam [PS88, PS89] give a methodology to verify systolic circuits based on solving systems of uniform recurrence equations, and then show how their methodology allows mechanical circuit verification, using the Boyer-Moore theorem prover system [BM79]. In [DLT89] primitive recursion is represented in the Martin-Löf type theory underlying the Nuprl proof development system, and a proof of correctness of a convolution circuit is carried out in the system. We have experimented with the verification of a number of systolic circuits in the Church type theory underlying the HOL system of M. Gordon (see [Gor86] and [Gor87] for a description of the HOL system).

None of the above methods are decision methods, and thus are not *fully* automatic. All of them require the user to submit "extra information" that is not in the specification or the implementation. Examples of such extra information are lemmas which are first proven and then used in the final verification proof, or induction hypotheses which are used when performing inductive proofs. It is often not obvious which lemmas or induction hypotheses are needed.

One method for automatic verification of systems composed of many identical processes is presented by Clarke et al in [CGB86]. The verification of a system with an arbitrary number of processes is reduced to the verification of a system with a fixed number of processes. The method of Clarke is used to prove synchronization properties of systems, but can not be applied to prove assertions about the data computations performed by a system, and thus can not be applied to the class of circuits which we will

treat.

1.5 Outline

In chapter 2 we introduce a formal model of systolic circuits. In chapter 3 we give some examples of systolic circuits. In chapter 4 we use basic ideas from the theories of abstract data types and fixed points to define a formal semantics of our model, and then define formally what we mean by systolic circuit verification. In chapter 5 we describe a method for automatic verification of a class of circuits which operate on commutative rings. In chapter 6 we describe a method for automatic verification of a class of circuits which operate on uninterpreted function symbols. In chapter 7 we describe a method for automatic verification of a class of circuits which operate on boolean algebras.

Chapter 2

The Model

In this chapter we will give a formal model for describing implementations and specifications of systolic circuits. A *systolic circuit* is composed of a number of *computation cells*, which are placed inside the circuit in a regular manner. Each cell has a number of inputs, a number of outputs, and a number of local storage variables (see figure 2.1). We consider synchronous circuits, thus we assume the presence of a global clock. Each cycle of the clock corresponds to a time step. An input of a computation cell is connected to an output of another cell. These connections follow a certain pattern which will be described later. At each clock cycle, each computation cell performs a number of computations on the data received from the inputs of the cell and the data stored in the local variables of the cell. Some of the computation results are sent out from the outputs of the cell, while some are stored in the local variables. The inputs of the computation cells on the "boundary" of the circuit which are not connected to the outputs of any other cells receive their data from outside the circuit. These constitute the inputs of the entire circuit. Before the first computations are performed the outputs and the local variables of the cells have initial values.

We will describe a very simple summation circuit which we will use as an illustration throughout the chapter. The circuit receives the elements of a matrix with infinitely many rows and ℓ columns, and outputs the result of summation of the elements of each row in the matrix. The summation circuit consists of an array of ℓ computation cells where each cell has two inputs and two outputs (see figures 2.2 and 2.3). The element $a(i,j)$ of the matrix is received via the second input of the cell at position j at time $i+j-2$. At each clock cycle each cell adds the values received from its two inputs and

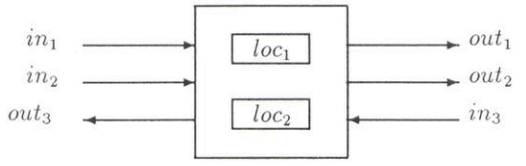

Figure 2.1: A cell with three inputs, three outputs, and two local variables

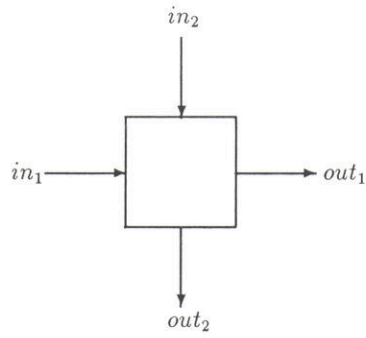

Figure 2.2: A cell in the summation circuit

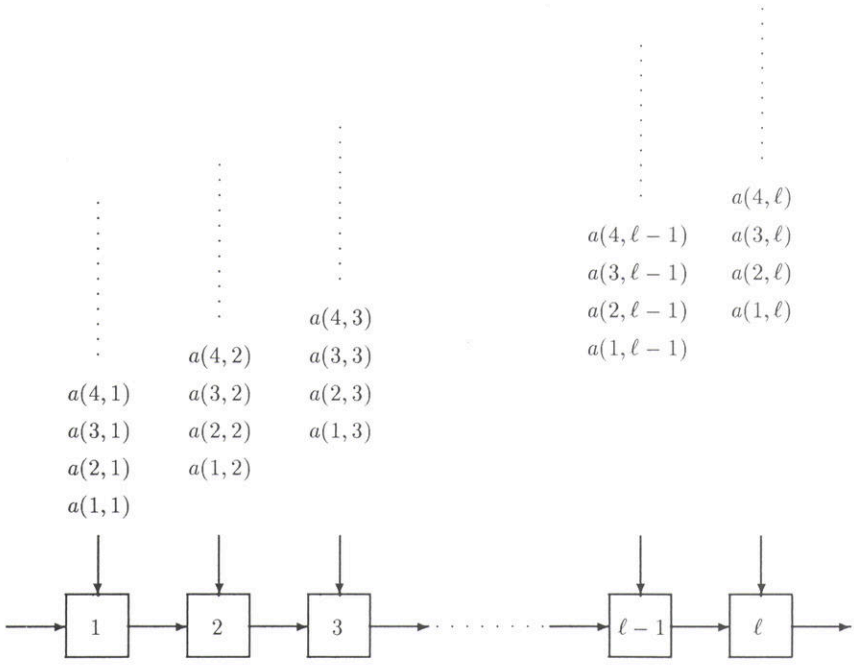

Figure 2.3: The summation circuit

outputs the result, via the first output of the cell, to the next cell. The first input of the first cell has always a value of 0. In this manner the result of summation of row i is obtained from the first ouput of the last cell at time $\ell + i - 1$. Note that the input to the circuit is in the form of streams of elements of a.

2.1 Preliminaries

Let $I =< I, +, -, \cdot, \leq, 0, 1 >$ be the ordered ring of integers, and $Q =< Q, +, -, \cdot, 0, 1 >$ the ring of rationals. We will use x and y (possibly with subscripts) to denote variables which range over I. We will use $\alpha, \beta, \gamma, \delta$ to range over I, and ρ to range over Q. By a *QI-polynomial* we mean a polynomial whose coefficients are rational numbers and whose variables range over the integers. By an *integer polynomial* we mean a polynomial whose coefficients are integers and whose variables range over the integers. We will use q and ℓ to range over integer and QI-polynomials.

When we write $q(x_1, \ldots, x_n)$ we mean that the set of free variables of q is a subset of $\{x_1, \ldots, x_n\}$. We use vector notation whenever possible. Thus we write e.g. $q(\overline{x})$ instead of $q(x_1, \ldots, x_n)$, $q(\overline{x} + \overline{y})$ instead of $q(x_1 + y_1, \ldots, x_n + y_n)$, and $\overline{\alpha}\,\overline{x} + \beta$ instead of $\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta$, if n is known or irrelevant in the context.

We assume that each circuit operates on a set $\mathcal{S}$ of data domains which are called the *sorts* of the circuit. Also we assume that we have two sets $\mathcal{A}$ and $\mathcal{G}$ of function symbols which are associated with each circuit. Each $a \in \mathcal{A}$ is called a *stream variable* and has an *arity* n, where n is a non-negative integer, and a *sort* S, where $S \in \mathcal{S}$. Each element $g \in \mathcal{G}$ is called an *operation symbol* and has a *domain sort* $w \in \mathcal{S}^*$ and a *range sort* $S \in \mathcal{S}$. A *constant operation symbol* is an operation symbol whose domain sort is empty. We denote the set of constant operation symbols in $\mathcal{G}$ by $\mathcal{K}$. The pair $< \mathcal{S}, \mathcal{G} >$ is called the *signature* of the circuit. We will use the stream variables to describe the inputs for our circuits which will be of the form of streams over the data domains on which the circuit operates. The operation symbols are used to describe the computations on data which are performed inside the cells of our circuits.

We also need to define a class of recursive functions which we will use to give formal descriptions of the implementations and specifications of our circuits. To do this we introduce the notion of *systems of recurrence equations* over a signature $\mathcal{SIG}$ as follows: We assume that we have a set of *function variables*, where each function variable f has an *arity* n where n is a nonnegative integer, and a *sort* $S \in \mathcal{S}$. The class of *systolic terms* over $\mathcal{SIG} =< \mathcal{S}, \mathcal{G} >$ and their *sorts* are defined as follows:

- If $a \in \mathcal{A}$ is a stream variable of arity n and sort S and $q_1(\overline{x}), \ldots, q_n(\overline{x})$ are QI-polynomials, then $a(q_1(\overline{x}), \ldots, q_n(\overline{x}))$ (called a *stream expression*) is a term over $\mathcal{SIG}$ of sort S.

- If f is a function variable of arity n and sort S, and $q_1(\overline{x}), \ldots, q_n(\overline{x})$ are integer polynomials, then $f(q_1(\overline{x}), \ldots, q_n(\overline{x}))$ (called a *function variable expression*) is a term over $\mathcal{SIG}$ of sort S.

- If $g \in \mathcal{G}$ is an operation symbol of domain sort $S_1 \cdot \cdots \cdot S_n$ and range sort S, and $t_1(\overline{x}), \ldots, t_n(\overline{x})$ are terms over $\mathcal{SIG}$, of sorts $S_1, \ldots, S_n$ respectively, then $g(t_1(\overline{x}), \ldots, t_n(\overline{x}))$ is a term over $\mathcal{SIG}$ of sort S.

A stream expression $a(q_1(\overline{x}), \ldots, q_n(\overline{x}))$ is said to be *well-defined* for a value $\overline{\alpha}$ of $\overline{x}$ if $q_i(\overline{\alpha})$ is an integer for $1 \leq i \leq n$. A systolic term $t(\overline{x})$ is said to be *well-defined* for a value $\overline{\alpha}$ of $\overline{x}$ if each stream expression in $t(\overline{x})$ is well-defined for $\overline{\alpha}$. A systolic term $t(\overline{x})$ is said to be *well-defined* under a predicate $p(\overline{x})$ if for each value $\overline{\alpha}$ of $\overline{x}$ if $p(\overline{\alpha})$ is true then $t(\overline{x})$ is well-defined for $\overline{\alpha}$.

A *functional* over $\mathcal{SIG}$ is of the form:

$$\begin{array}{l}\textbf{case}\\ \quad \begin{array}{lcl} p_1(\overline{x}) & \Longrightarrow & t_1(\overline{x}) \\ & \vdots & \\ p_n(\overline{x}) & \Longrightarrow & t_n(\overline{x}) \end{array}\\ \textbf{endcase}\end{array}$$

where $p_i(\overline{x})$ is a predicate over I, and $t_i(\overline{x})$ is a term over $\mathcal{SIG}$. We assume that $t_1(\overline{x}), \ldots, t_n(\overline{x})$ all have the same sort S, and that $t_i(\overline{x})$ is well-defined under $p_i(\overline{x})$. Furthermore, we assume that for each value $\overline{\alpha}$ of $\overline{x}$, $p_i(\overline{\alpha}) \wedge p_j(\overline{\alpha})$ is false if $j \neq i$. Each $p_i(\overline{x})$ is called a *guard* of the functional, and each $t_i(\overline{x})$ is called a *result* of the functional, while every $p_i(\overline{x}) \Rightarrow t_i(\overline{x})$ is called a *case* of the functional. We call S the *sort* of the functional.

A *system of recurrence equations* over $\mathcal{SIG}$ is of the form:

$$\begin{array}{lcl} f_1(\overline{x}) & = & \mathcal{F}_1(\overline{x}) \\ & \vdots & \\ f_n(\overline{x}) & = & \mathcal{F}_n(\overline{x}) \end{array}$$

where f_i is a function variable, and $\mathcal{F}_i$ is a functional over $\mathcal{SIG}$ whose sort is the same as the sort of f_i. In addition, we assume that the only function variables which occur in $\mathcal{F}_1(\overline{x}), \ldots, \mathcal{F}_n(\overline{x})$ are $f_1, \ldots, f_n$. We call $\mathcal{SIG}$ the *signature* of the system of recurrence equations.

2.2 Implementation of Systolic Circuits

The formal description of the implementation of a systolic circuit consists of three parts:

- The *topology* of the circuit, i.e. how the cells are placed inside the circuit.
- The *interconnections*, i.e. how inputs and outputs of the cells are connected.
- The *cell computations*, i.e. the manner in which values of outputs and local variables in a cell at some time instant depend on values of inputs and local variables in the cell at previous time instants.

A. Topology Generally the topology of an n-dimensional circuit can be described as a predicate $top(\overline{x}, \overline{\ell})$ where $\overline{x}$ is an n-tuple, and $\overline{\ell}$ is called a *circuit parameter*. The predicate *top* represents a family of circuits. For every integer value of $\overline{\ell}$ we get a member of the family of circuits whose topology is described as follows: there is a computation cell at each integer point $\overline{x}$ satisfying $top(\overline{x}, \overline{\ell})$. The circuit parameter usually describes

the size of the circuit. For example in the case of the summation circuit, the topology of the circuit is described by the predicate *sumtop*, where:

$$sumtop(x, \ell) \;=\; 1 \leq x \leq \ell \tag{2.1}$$

which means that there is a computation cell at each integer point between and including 1 and ℓ. We observe that ℓ (which is a circuit parameter) decides the length of the array, and that *sumtop* defines the topology of a family of summation circuits. For each value of ℓ we get the topology of one circuit in the family. For example if $\ell = 3$ then we get the topology of a summation circuit which has a computation cell at the points 1, 2, and 3, and if $\ell = 5$ then we get a circuit which has a cell at the points 1, 2, 3, 4, and 5, etc.

In the following we denote the inputs, outputs, and local variables of the cells by $in_1, \ldots, in_{m_1}$, $out_1, \ldots, out_{m_2}$, and $loc_1, \ldots, loc_{m_3}$ respectively. We will use the same notation in_i to denote the wire in_i, and the function in_i which gives the value $in_i(\overline{x}, \overline{\ell}, t)$ of the i^{th} input of the cell placed at $\overline{x}$ at time instant t. The same applies to out_i and loc_i. By a *signal* we mean any in_i, out_i, or loc_i. We use s to range over signals.

B. Interconnections In the class of circuits we consider, we allow a pattern of interconnections which can be defined as follows: For each input in_i there is an integer polynomial $\Delta_i(\overline{x}, \overline{\ell})$ (called the *connection function* of in_i), such that the input in_i of a cell at $\overline{x}$ is connected to the corresponding output out_i of the cell at $\overline{x} + \Delta_i(\overline{x}, \overline{\ell})$. Observe that this means that the number of inputs of each cell is equal to the number of outputs of the cell. The cells on the "boundary" of the circuit whose i^{th} inputs are not connected to the outputs of any other cells receive their inputs from outside the circuit. Let $top_i'(\overline{x}, \overline{\ell})$ denote the position of those cells whose i^{th} inputs are connected to outputs of other cells, and $top_i''(\overline{x}, \overline{\ell})$ the position of those cells whose i^{th} inputs are not connected to outputs of other cells, then we get:

$$top_i'(\overline{x}, \overline{\ell}) \;\longrightarrow\; in_i(\overline{x}, \overline{\ell}, t) \;=\; out_i(\overline{x} + \Delta_i(\overline{x}, \overline{\ell}), \overline{\ell}, t) \tag{2.2}$$

$$top_i''(\overline{x}, \overline{\ell}) \;\longrightarrow\; in_i(\overline{x}, \overline{\ell}, t) = Input_i(\overline{x}, \overline{\ell}, t) \tag{2.3}$$

Here $Input_i$ is called an *input function* and defines the value of the i^{th} inputs of the cells which are on the "boundary" of the circuit. These may be considered as inputs to the entire circuit. Observe that:

$$top_i'(\overline{x}, \overline{\ell}) \;=\; top_i(\overline{x}, \overline{\ell}) \;\wedge\; top_i(\overline{x} + \Delta_i(\overline{x}, \overline{\ell}), \overline{\ell}) \tag{2.4}$$

$$top_i''(\overline{x}, \overline{\ell}) \;=\; top_i(\overline{x}, \overline{\ell}) \;\wedge\; \neg top_i(\overline{x} + \Delta_i(\overline{x}, \overline{\ell}), \overline{\ell}) \tag{2.5}$$

We assume that the input functions are described as *input expressions*. An *input expression* is a functional over the signature of the circuit of the form:

$$\begin{array}{lll} \textbf{case} & & \\ \quad p_1(\overline{x}, \overline{\ell}, t) & \Longrightarrow & it_1(\overline{x}, \overline{\ell}, t) \\ & \vdots & \\ \quad p_n(\overline{x}, \overline{\ell}, t) & \Longrightarrow & it_n(\overline{x}, \overline{\ell}, t) \\ \textbf{endcase} & & \end{array} \tag{2.6}$$

where each $it_i(\overline{x}, \overline{\ell}, t)$ is an *input term*. An *input term* is either a stream expression, or a constant operation symbol.

In the case of the summation circuit the connection functions are defined by the following:

$$\Delta_1(x, \ell) = -1 \tag{2.7}$$
$$\Delta_2(x, \ell) = -x \tag{2.8}$$

The interconnections are defined by:

$$(2 \leq x \leq \ell) \longrightarrow in_1(x, \ell, t) = out_1(x - 1, \ell, t) \tag{2.9}$$
$$(x = 1) \longrightarrow in_1(x, \ell, t) = Input_1(x, \ell, t) \tag{2.10}$$
$$(1 \leq x \leq \ell) \longrightarrow in_2(x, \ell, t) = Input_2(x, \ell, t) \tag{2.11}$$

These equations mean that the first input of each cell (except the first cell) is connected to the first output of the previous cell, while the first input of the first cell, which is not connected to the outputs of other cells, receives its values from the environment of the circuit as described by the input function $Input_1$ which is considered as an input to the entire circuit. No cell has a second input which is connected to the outputs of other cells. The value of each in_2 is received from the environment of the circuit as defined by the input function $Input_2$. The input functions $Input_1$ and $Input_2$ are defined as follows:

$$Input_1(x, \ell, t) = 0 \tag{2.12}$$
$$Input_2(x, \ell, t) = \textbf{case} \quad \{\, x \leq t + 1 \,\} \implies a(t - x + 2, x) \quad \textbf{endcase} \tag{2.13}$$

Note that $Input_2$ declares how the elements of a actually are input to the circuit. This mapping of the input sequences into the clock cycles of the circuit is called the *input scheduling* of the circuit [HTT88].

C. Cell Computations As mentioned earlier, at each clock cycle, a cell performs a number of computations on the data received from the inputs of the cell and the data stored in the local variables of the cell. Some of the computation results are sent out from the outputs of the cell, while some are stored in the local variables. Thus the value of an output or a local variable at a certain time step depends on the values of the inputs and local variables at some previous time. The computations performed in a cell are dependent on the position of the cell and the time instant. Thus, although all the cells in a circuit are similar, they may perform different computations, since their positions may differ. The same cell may perform varying computations, depending on the current time.

Let $\mathcal{SIG} =< \mathcal{S}, \mathcal{G} >$ be the signature of the circuit. Each output or local variable s in the circuit has a *sort* $S \in \mathcal{S}$. Formally a *cell computation term* its *sort*, and its *delay* are defined by the following:

- If s is an input or a local variable of sort S, then $s(\overline{x}, \overline{\ell}, t - \tau)$, where τ is a positive integer, is a cell computation term with sort S and delay τ.

- If $ct_1(\overline{x},\overline{\ell},t),\ldots,ct_n(\overline{x},\overline{\ell},t)$ are cell computation terms with sorts $S_1,\ldots,S_n$, and delays $\tau_1,\ldots,\tau_n$ respectively, and if $g \in \mathcal{G}$ is an operation symbol with a domain sort $S_1 \cdot \cdots \cdot S_n$ and a range sort S, then $g(ct_1(\overline{x},\overline{\ell},t),\ldots,ct_n(\overline{x},\overline{\ell},t))$ is a cell computation term with a sort S and a delay $max(\tau_1,\ldots,\tau_n)$.

A *cell computation* is a functional of the form:

$$\begin{array}{l}\textbf{case}\\ \quad p_1(\overline{x},\overline{\ell},t) \implies ct_1(\overline{x},\overline{\ell},t)\\ \qquad\vdots\\ \quad p_m(\overline{x},\overline{\ell},t) \implies ct_m(\overline{x},\overline{\ell},t)\\ \textbf{endcase}\end{array} \tag{2.14}$$

where each $ct_i(\overline{x},\overline{\ell},t)$ is a cell computation term. The *delay* of the cell computation above is defined to be $max(\tau_1,\ldots,\tau_m)$, where τ_i is the delay of $ct_i(\overline{x},\overline{\ell},t)$.

For each output or local variable s, there is a computation $\mathcal{F}$ associated with s such that the sort of s is the same as the sort of $\mathcal{F}$. The *delay* τ of s is defined to be equal to the delay τ of $\mathcal{F}$. We assume that the first computation result appears on s first when $t \geq \tau$. When $t < \tau$ the value of s will be equal to an initial value.

In general the value of s can be described as follows: After the period of initialization the value of s is described by an equation of the form:

$$(t \geq \tau) \wedge top(\overline{x},\overline{\ell}) \longrightarrow s(\overline{x},\overline{\ell},t) = \mathcal{F}(\overline{x},\overline{\ell},t) \tag{2.15}$$

Let $\mathcal{F}(\overline{x},\overline{\ell},t)$ be of the form of (2.14), then equation (2.15) means that the value of s at each cell at $t \geq \tau$ is decided by checking the predicates $p_1,\ldots,p_m$ (i.e. by checking the cell position and time instant). If $p_i(\overline{x},\overline{\ell},t)$ is true then the value of $s(\overline{x},\overline{\ell},t)$ will be equal to $ct_i(\overline{x},\overline{\ell},t)$. If no $p_i(\overline{x},\overline{\ell},t)$ is true then the value of $s(\overline{x},\overline{\ell},t)$ is undefined. Note that by the help of the predicates p_i we can code such statements as: "the cells at the odd positions perform a certain operation while the cells at the even positions perform another operation", or "a cell performs a certain operation (or is idle) up to a certain time instant (which may depend on the cell position) after which it performs another operation", etc. We will see examples of such statements in describing the computations of the summation circuit, and also in the circuits described in chapter 3.

The period of initialization is described by the equation:

$$(0 \leq t < \tau) \wedge top(\overline{x},\overline{\ell}) \longrightarrow s(\overline{x},\overline{\ell},t) = Init_s(\overline{x},\overline{\ell},t) \tag{2.16}$$

Equation (2.16) means that when $t < \tau$, the value of $s(\overline{x},\overline{\ell},t)$ is equal to an initial value $Init_s(\overline{x},\overline{\ell},t)$. For each output or local variable s, there is an *initial function* $Init_s$ which defines the value of s when $t < \tau$, where τ is the delay of s. We assume that the initial functions are described as *initial expressions*. An *initial expression* is of the same form as an input expression (see (2.6)).

In the case of the summation circuit the cell computations are described as follows: After the initialization period the computations are defined by:

$$\begin{array}{l}(t \geq 1) \wedge (1 \leq x \leq \ell) \longrightarrow\\ \quad out_1(x,\ell,t) = \textbf{case}\\ \qquad\qquad \{\ x \leq t\ \} \implies in_1(x,\ell,t-1) + in_2(x,\ell,t-1)\\ \qquad\qquad \textbf{endcase}\end{array} \tag{2.17}$$

$$(t \geq 1) \wedge (1 \leq x \leq \ell) \longrightarrow out_2(x,\ell,t) = in_2(x,\ell,t-1) \tag{2.18}$$

The initialization period is defined:

$$(0 \leq t < 1) \wedge (1 \leq x \leq \ell) \longrightarrow out_1(x,\ell,t) = Init_1(x,\ell,t) \tag{2.19}$$

and:

$$(0 \leq t < 1) \wedge (1 \leq x \leq \ell) \longrightarrow out_2(x,\ell,t) = Init_2(x,\ell,t) \tag{2.20}$$

The initial functions are defined by:

$$Init_1(x,\ell,t) = \text{undefined} \tag{2.21}$$

and:

$$Init_2(x,\ell,t) = \text{undefined} \tag{2.22}$$

Equations (2.21) and (2.22) indicate that no initial values are preloaded into the circuit.

2.3 Specification of Systolic Circuits

A *specification* of a system describes how we expect the system to behave. A specification of a systolic circuit states which values we expect certain wires (i.e. cell inputs and outputs) and certain local variables of the circuit to have at certain time instants. This can be expressed by the general form:

$$spec(\overline{x},\overline{\ell},t) = \tag{2.23}$$
$$\Big(p_1(\overline{x},\overline{\ell},t) \longrightarrow (s_1(\overline{x},\overline{\ell},t) = f_1(\overline{x},\overline{\ell},t))\Big) \wedge \cdots \wedge \Big(p_m(\overline{x},\overline{\ell},t) \longrightarrow (s_m(\overline{x},\overline{\ell},t) = f_m(\overline{x},\overline{\ell},t))\Big)$$

where p_i is a predicate, f_i is a function variable in a system of recurrence equations, and s_i is a signal in the circuit. Intuitively the specification formula means that whenever the predicate $p_i(\overline{x},\overline{\ell},t)$ is true then the value of the signal $s_i(\overline{x},\overline{\ell},t)$ should be equal to $f_i(\overline{x},\overline{\ell},t)$.

In the case of the summation circuit the specification formula is given by:

$$sumspec(x,\ell,t) = (x = \ell) \wedge (\ell \leq t) \longrightarrow out_1(x,\ell,t) = f(t-\ell+1,\ell)$$

where f is defined by:

$$f(y_1,y_2) = \begin{array}{l}\textbf{case} \\ \qquad \left\{ y_2 = 1 \right\} \implies a(y_1,y_2) \\ \qquad \left\{ 1 < y_2 \leq \ell \right\} \implies f(y_1,y_2-1) + a(y_1,y_2) \\ \textbf{endcase}\end{array}$$

Note that the specification not only tells us that we get the expected output of the circuit, but also the time instants at which to expect the elements of the output. This mapping from circuit output to clock cycles is called *output scheduling* [HTT88].

Chapter 3

Examples

In this chapter we introduce three systolic circuits: a convolution circuit, a string matching circuit, and a substring detecting circuit. We will use these circuits as examples to illustrate the application of the automatic verification methods introduced in chapters 5, 6, and 7. For each circuit, we give the formal descriptions of the implementation and the specification according to the model of chapter 2.

3.1 A Convolution Circuit

The convolution of two sequences $a(0), a(1), \ldots, a(\ell)$ and $b(0), b(1), \ldots, b(\ell)$ of ring elements is a sequence $c(0), c(1), \ldots, c(2\ell)$ where[1]:

$$c(j) = \begin{cases} \sum_{i=0}^{j} a(i) \cdot b(j-i) & \text{if } 0 \leq j \leq \ell \\ \sum_{i=0}^{2\ell-j} a(j-\ell+i) \cdot b(\ell-i) & \text{if } \ell < j \leq 2\ell \end{cases} \tag{3.1}$$

The convolution algorithm has many important applications in digital signal processing, and is also related to integer multiplication and polynomial multiplication. In this section we introduce a convolution circuit, which is a version of that given in [Ull84].

The circuit consists of an array of $\ell+1$ cells. Each cell has three inputs, three outputs, and a local variable (see figures 3.1 and 3.2). The input sequences a and b are fed into the circuit from the left, and propagate to the right. The sequence a is input from in_1 of the first cell (i.e. that at position 0), while the sequence b is input from in_2 of the same cell. The elements of b move twice as fast as the elements of a. When $a(0)$ enters a cell at time t, the value of the local variable of the cell is updated to the value of the element of b entering the cell at time $t-2$. The elements of c propagate from right to left. As

[1]Notice that $\sum_{i=0}^{\ell(\overline{x})} r(\overline{x}, i)$ can be defined by $sum(\overline{x}, 0, \ell(\overline{x}))$, where $sum(\overline{x}, i, n)$ is given by the recurrence equation:

$$sum(\overline{x}, i, n) = \begin{array}{l} \textbf{case} \\ \quad \{\ i = n\ \} \implies r(\overline{x}, i) \\ \quad \{\ i < n\ \} \implies r(\overline{x}, i) + sum(\overline{x}, i+1, n) \\ \textbf{endcase} \end{array}$$

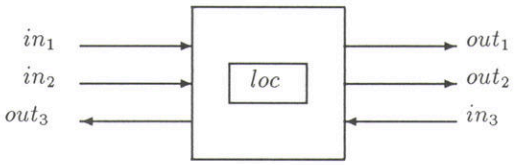

Figure 3.1: A cell in the convolution circuit

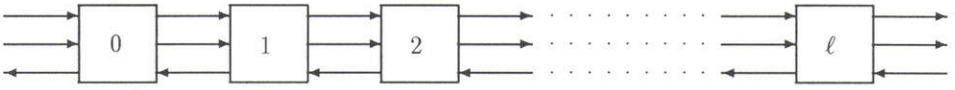

Figure 3.2: The convolution circuit

an element of c moves, it meets elements of a propagating to the right, and elements of b stored in the local variables of the cells. As an element $c(i)$ meets a pair of elements from a and b, the product of the pair is added to the sum stored in $c(i)$. Observe that an element $c(i)$ should meet a pair of elements from a and b iff they are part of the sum for $c(j)$, i.e. of the form $a(i), b(j-i)$, or $a(j-\ell+i), b(\ell-i)$. Also observe that the elements of a and c should propagate in alternate processors, otherwise each element c would miss half of the elements of a which are part of its sum.

In section 3.1.1 we will give a formal description of the implementation of the convolution circuit, and show that it falls into the category of circuits we described in section 2.2. In section 3.1.2 we will give a formal specification of the circuit.

3.1.1 Implementation of the Convolution Circuit

We will structure the formal description of the implementation of the convolution circuit according to the scheme presented in section 2.2.

A. Topology The convolution circuit consists of an array of $\ell + 1$ cells. The topology of the circuit can be described as:

$$top(x, \ell) \quad = \quad 0 \leq x \leq \ell \tag{3.2}$$

This means that we place a cell at each integer point between (and including) 0 and ℓ. Observe that ℓ is a circuit parameter and decides the length of the array. The predicate *top* describes the topology of a family of convolution circuits; one for each value of the length of the input sequences a and b.

B. Interconnections The connection functions are defined by $\delta_1 =< -1 >$, $\delta_2 =< -1 >$, and $\delta_3 =< 1 >$. Thus we get the following equations:

$$(1 \leq x \leq \ell) \longrightarrow in_1(x,\ell,t) = out_1(x-1,\ell,t) \tag{3.3}$$

$$(1 \leq x \leq \ell) \longrightarrow in_2(x,\ell,t) = out_2(x-1,\ell,t) \tag{3.4}$$

$$(0 \leq x \leq \ell-1) \longrightarrow in_3(x,\ell,t) = out_3(x+1,\ell,t) \tag{3.5}$$

These equations mean that in_1 and in_2 of each cell (except the first cell i.e. that at position 0) is connected to out_1 and out_2 respectively of the previous cell, while in_3 of each cell (except the last cell i.e. that at position ℓ) is connected to out_3 of the next cell. The inputs $Input_1$ and $Input_2$ are fed into the circuit via in_1 and in_2 respectively of the first cell, while $Input_3$ is fed into the circuit via in_3 of the last cell.

$$(x = 0) \longrightarrow in_1(x,\ell,t) = Input_1(x,\ell,t) \tag{3.6}$$

$$(x = 0) \longrightarrow in_2(x,\ell,t) = Input_2(x,\ell,t) \tag{3.7}$$

$$(x = \ell) \longrightarrow in_3(x,\ell,t) = Input_3(x,\ell,t) \tag{3.8}$$

The input functions are given by:

$$\begin{aligned} Input_1(x,\ell,t) \;=\; & \textbf{case} \\ & \left\{ \begin{array}{c} (t+2)\ mod\ 4 = 0 \\ 2 \leq t \leq 4\ell+2 \end{array} \right\} \implies a\left(\tfrac{t-2}{4}\right) \\ & \left\{ \begin{array}{c} (t+2)\ mod\ 4 = 0 \\ 4\ell+2 < t \leq 8\ell+2 \end{array} \right\} \implies 0 \\ & \textbf{endcase} \end{aligned} \tag{3.9}$$

$$\begin{aligned} Input_2(x,\ell,t) \;=\; & \textbf{case} \\ & \left\{ 0 \leq t \leq \ell \right\} \implies b(t) \\ & \textbf{endcase} \end{aligned} \tag{3.10}$$

$$\begin{aligned} Input_3(x,\ell,t) \;=\; & \textbf{case} \\ & \left\{ \begin{array}{c} 2\ell+6 \leq t \leq 6\ell+2 \\ (t+2\ell+2)\ mod\ 4 = 0 \end{array} \right\} \implies 0 \\ & \textbf{endcase} \end{aligned} \tag{3.11}$$

Note that $Input_1$ and $Input_2$ define the input scheduling of the circuit, i.e. how the sequences a and b actually are input to the circuit (see figure 3.3).

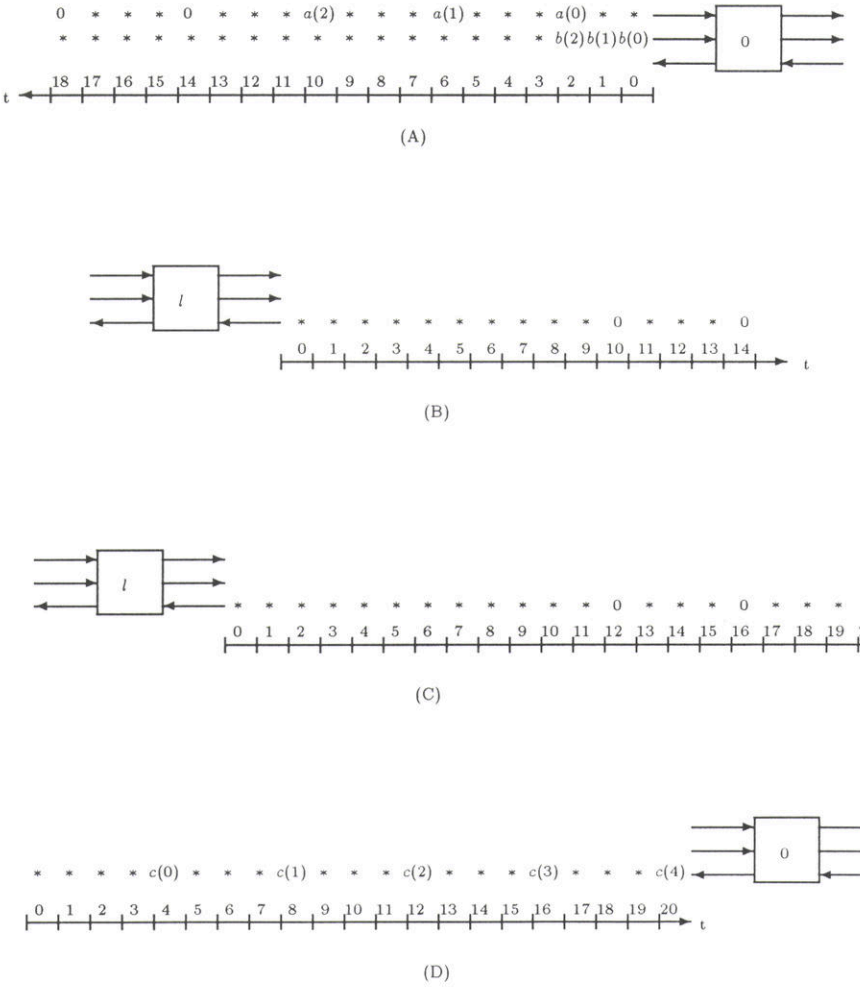

Figure 3.3: (A) $Input_1$ and $Input_2$ when $\ell = 2$. (B) $Input_3$ when $\ell = 2$. (C) $Input_3$ when $\ell = 3$. (D) Value of $out_3(0, \ell, t)$ when $\ell = 2$. The "*"s represent undefined values.

C. Cell Computations The computations which take place in the cells can be described by the following:

$$\begin{array}{l}(t \geq 2) \wedge (0 \leq x \leq \ell) \longrightarrow \\ out_1(x,\ell,t) = \textbf{case} \\ \qquad \left\{ t \; mod \; 2 = 0 \right\} \implies in_1(x,\ell,t-2) \\ \textbf{endcase}\end{array} \tag{3.12}$$

$$(t \geq 1) \wedge (0 \leq x \leq \ell) \longrightarrow out_2(x,\ell,t) = in_2(x,\ell,t-1) \tag{3.13}$$

$$\begin{array}{l}(t \geq 2) \wedge (0 \leq x \leq \ell) \longrightarrow \\ loc(x,\ell,t) = \textbf{case} \\ \qquad \left\{ t = 2x+2 \right\} \implies in_2(x,\ell,t-2) \\ \qquad \left\{ \begin{array}{l} t \; mod \; 2 = 0 \\ 2x+2 < t \end{array} \right\} \implies loc(x,\ell,t-2) \\ \textbf{endcase}\end{array} \tag{3.14}$$

$$\begin{array}{l}(t \geq 2) \wedge (0 \leq x \leq \ell) \longrightarrow \\ out_3(x,\ell,t) = \\ \textbf{case} \\ \quad \left\{ t = 2x+4 \right\} \implies in_1(x,\ell,t-2) \cdot loc(x,\ell,t-2) \\ \quad \left\{ \begin{array}{l} t \; mod \; 2 = 0 \\ 2x+4 < t \end{array} \right\} \implies in_3(x,\ell,t-2) + in_1(x,\ell,t-2) \cdot loc(x,\ell,t-2) \\ \textbf{endcase}\end{array} \tag{3.15}$$

From (3.12), (3.13), (3.14), and (3.15) we conclude that the delays of out_1, out_2, loc, and out_3 are 2, 1, 2, and 2 respectively.

Consider a cell at position x. The value of out_1 at the cell at an even clock cycle $t \geq 2$ is equal to the value of in_1 at $t-2$. The value of out_2 at any clock cycle $t \geq 1$ is equal to the value of in_2 at $t-1$. The value of loc at any even clock cycle $t \geq 2$ is decided by the following: if $t = 2x+2$ then the value of in_2 at $t-2$ is "stored" in the local variable, otherwise if $t > 2x+2$ then the value of loc will not be changed from its value at $t-2$. The value of out_3 at any even clock cycle $t \geq 2$ is decided by the following: if $t = 2x+4$ then it is equal to the multiplication of the values of in_1 and loc at $t-2$, while if $2x+4 < t$ then it is equal to the result of adding the value of in_3 to the multiplication of the values of in_1 and loc at $t-2$. Observe that out_1, out_3, and loc are "idle" at odd clock cycles. This means that their values are "undefined" at these clock cycles. In sections 4.3 and 5.3 we will define formally what we mean by "undefined". In practice it means that they have "don't care" values. Note that out_2 is "active" at each clock cycle.

The period of initialization is defined by:

$$(0 \leq t < 2) \wedge (0 \leq x \leq \ell) \longrightarrow out_1(x,\ell,t) = OutInit_1(x,\ell,t) = \text{undefined} \tag{3.16}$$

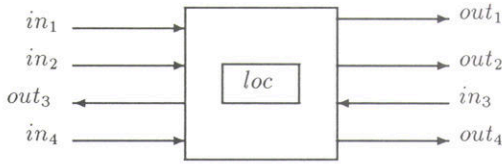

Figure 3.4: A cell of the modified convolution circuit

$$(0 \leq t < 1) \wedge (0 \leq x \leq \ell) \longrightarrow out_2(x,\ell,t) = OutInit_2(x,\ell,t) = \text{undefined} \quad (3.17)$$

$$(0 \leq t < 2) \wedge (0 \leq x \leq \ell) \longrightarrow out_3(x,\ell,t) = OutInit_3(x,\ell,t) = \text{undefined} \quad (3.18)$$

$$(0 \leq t < 2) \wedge (0 \leq x \leq \ell) \longrightarrow loc(x,\ell,t) = LocInit(x,\ell,t) = \text{undefined} \quad (3.19)$$

The above equations indicate that no initial values are preloaded to the circuit. The initial values do not affect the results of the computations of the circuit, and hence their values are undefined.

Control Signals We observe that the computations in the cells of the convolution circuit depend on the time and the position of the cell. For example if we consider equation (3.14) we observe that the value stored in the local variable is different at different clock cycles, depending on whether $t = 2x + 2$, or $t > 2x + 2$. Similarly the value sent out via out_3, as described by equation (3.15), is dependent on whether $t = 2x + 4$, or $t > 2x + 4$.

In practice there are different methods of implementation to enable the cells to check the relative relation of time and position.

One method is to preload the position into the cell and then use hardware circuitry to compare the cell position with the number of clock cycles.

Another method is to use *control signals*. A *control signal* is a sequence of constants, which is fed into the circuit, and then propagated to the different cells. The constants are coded in such a way as to reflect the time-position relation of each cell.

The convolution circuit can, for example, be modified and supplied with a control signal. A fourth input and output is added to each cell (see figure 3.4). These are used to propagate the control signal throughout the circuit, as described by the following equations:

$$\begin{array}{l} (t \geq 2) \wedge (0 \leq x \leq \ell) \longrightarrow \\ out_4(x,\ell,t) = \textbf{case} \\ \qquad \left\{ \; t \; mod \; 2 = 0 \; \right\} \implies in_4(x,\ell,t-2) \\ \qquad \textbf{endcase} \end{array} \quad (3.20)$$

$$OutInit_4(x,\ell,t) = \text{undefined} \quad (3.21)$$

$$(1 \leq x \leq \ell) \longrightarrow in_4(x,\ell,t) = out_4(x-1,\ell,t) \tag{3.22}$$

$$\begin{aligned} Input_4(x,\ell,t) \;=\; & \textbf{case} \\ & \left\{\; t = 0 \;\right\} \implies 0 \\ & \left\{\begin{array}{c} t \bmod 2 = 0 \\ 2 \leq t \leq 8\ell + 2 \end{array}\right\} \implies 1 \\ & \textbf{endcase} \end{aligned} \tag{3.23}$$

We observe that if $in_4(x,\ell,t-2) = 0$ then $t = 2x+2$, if $in_4(x,\ell,t-2) = 1$ then $t > 2x+2$, if $in_4(x,\ell,t-4) = 0$ then $t = 2x+4$, and if $in_4(x,\ell,t-4) = 1$ then $t > 2x+4$. This means that: the equality $t = 2x+2$ in equation (3.14), the inequality $t > 2x+2$ in equation (3.14), the equality $t = 2x+4$ in equation (3.15), and the inequality $t > 2x+4$ in equation (3.15) can be replaced by $in_4(x,\ell,t-2) = 0$, $in_4(x,\ell,t-2) = 1$, $in_4(x,\ell,t-4) = 0$, and $in_4(x,\ell,t-4) = 1$ respectively.

The control signals can be treated in much the same way as other signals in the circuit, thus allowing us to analyze circuits containing control signals. This analysis is not included here.

3.1.2 Specification of the Convolution Circuit

The specification of the circuit tells us that we get the convolution sequence c from out_3 of the first cell. Furthermore it reveals that we get one element of c at each fourth clock cycle, starting at $t = 4$ and finishing at $t = 8\ell + 4$.

$$\begin{aligned} & spec(x,\ell,t) = \\ & \quad (x = 0) \wedge (4 \leq t \leq 8\ell + 4) \wedge (t \bmod 4 = 0) \longrightarrow \left(out_3(x,\ell,t) \;=\; c\left(\frac{t-4}{4}\right)\right) \end{aligned}$$

From equation (3.1):

$$\begin{aligned} & spec(x,\ell,t) \;= \\ & \quad (x = 0) \wedge (4 \leq t \leq 4\ell + 4) \wedge (t \bmod 4 = 0) \longrightarrow \\ & \qquad \left(out_3(x,\ell,t) \;=\; \textstyle\sum_{i=0}^{\frac{t-4}{4}} a\,(i) \cdot b\left(\frac{t-4i-4}{4}\right)\right) \\ & \qquad\qquad \wedge \\ & \quad (x = 0) \wedge (4\ell + 4 < t \leq 8\ell + 4) \wedge (t \bmod 4 = 0) \longrightarrow \\ & \qquad \left(out_3(x,\ell,t) \;=\; \textstyle\sum_{i=0}^{\frac{8\ell-t+4}{4}} a\left(\frac{t-4\ell+4i-4}{4}\right) \cdot b\,(\ell - i)\right) \end{aligned} \tag{3.24}$$

Note that the specification defines the output scheduling of the circuit, i.e. the time instants at which to expect the elements of the output.

3.2 A String Matching Circuit

A simple measure of similarity between two strings is the distance between them. The *distance* between two strings is defined to be the smallest number of characters (in the two

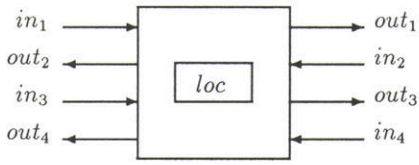

Figure 3.5: A cell in the string matching circuit

strings) which should be removed to make the strings equal [Sel80, Ukk85]. For example the distance between IRAQ and IRAN is 2, and between FRANCE and FLORENCE is 4.

String distance algorithms have applications in molecular biology, signal processing, spelling correction, speech understanding, and fault tolerant systems.

Formally the distance between the two strings:

$$a(1), a(1), \ldots, a(\ell_1)$$

and

$$b(1), b(2), \ldots, b(\ell_2)$$

is defined by $d(\ell_1, \ell_2)$, where:

$$d(i,j) = \begin{cases} g_1 & \text{if } i = 0 \text{ and } j = 0 \\ g_2(d(i-1,j)) & \text{if } 0 < i \leq \ell_1 \text{ and } j = 0 \\ g_2(d(i,j-1)) & \text{if } i = 0 \text{ and } 0 < j \leq \ell_2 \\ g_3(a(i), b(j), d(i,j-1), & \text{if } 0 < i \leq \ell_1 \text{ and } 0 < j \leq \ell_2 \\ \quad d(i-1,j), d(i-1,j-1)) & \end{cases} \tag{3.25}$$

where g_1, g_2, and g_3 are interpreted as:

$$\begin{aligned} g_1 &= 0 \\ g_2(x) &= 1 + x \\ g_3(x_1, x_2, x_3, x_4, x_5) &= \text{if } (x_1 = x_2) \text{ then } x_5 \text{ else } 1 + min(x_3, x_4) \end{aligned}$$

We will describe a string matching circuit which is a version of that given in [LL85]. It consists of an array of cells. Each cell has four inputs, four outputs, and a local variable (see figures 3.5 and 3.6).

The sequence a is preloaded into the left half of the array. One element of a is stored at each other cell starting at the first cell from the left. The sequence b is preloaded into the right half of the array. One element of b is stored at each other cell starting at the first cell from the right. At each clock cycle the elements of a move one step to the right, while the elements of b move one step to the left. When an element $a(i)$ meets an element

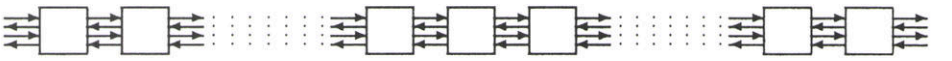

Figure 3.6: The string matching circuit

$b(j)$ in a cell, the element $d(i,j)$ is calculated. The result is both stored inside the cell, and sent to the cells immediately to the left and the right. If $a(i) = b(j)$ then the value of $d(i,j)$ will be equal to the value of $d(i-1,j-1)$ which is stored inside the cell. If $a(i) \neq b(j)$ then the value of $d(i,j)$ will be equal to $1 + min(d(i,j-1), d(i-1,j))$. The elements $d(i,j-1)$ and $d(i-1,j)$ are received from the cells immediately to the left and the right respectively. The distance between the two strings, as defined by the element $d(\ell_1,\ell_2)$, is fetched from the cell at $\ell_2 - \ell_1$, at time $\ell_1 + \ell_2 + 1$.

In section 3.2.1 we will give a formal description of the implementation of the string matching circuit. In section 3.2.2 we will give a formal specification of the circuit.

3.2.1 Implementation of the String Matching Circuit

We will structure the formal description of the implementation of the string matching circuit according to the scheme presented in section 2.2.

A. Topology The string matching circuit consists of an array of $2(\ell_1 + \ell_2) + 3$ cells. The topology of the circuit can be described as:

$$top(x,\ell_1,\ell_2) \;=\; -2\ell_1 - 1 \leq x \leq 2\ell_2 + 1 \tag{3.26}$$

This means that a cell is placed on each integer point between (and including) $-2\ell_1 - 1$ and $2\ell_2 + 1$. Note that ℓ_1 and ℓ_2 are circuit parameters and decide the length of the array from left and right. The predicate *top* describes the topology of a family of string matching circuits; one for each value of the length of the input sequences a and b.

B. Interconnections The connection vectors are defined by $\delta_1 =< -1 >$, $\delta_2 =< 1 >$, $\delta_3 =< -1 >$, and $\delta_4 =< 1 >$. Thus we get the following equations:

$$(-2\ell_1 \leq x \leq 2\ell_2 + 1) \;\longrightarrow\; in_1(x,\ell_1,\ell_2,t) = out_1(x-1,\ell_1,\ell_2,t) \tag{3.27}$$

$$(-2\ell_1 - 1 \leq x \leq 2\ell_2) \;\longrightarrow\; in_2(x,\ell_1,\ell_2,t) = out_2(x+1,\ell_1,\ell_2,t) \tag{3.28}$$

$$(-2\ell_1 \leq x \leq 2\ell_2 + 1) \;\longrightarrow\; in_3(x,\ell_1,\ell_2,t) = out_3(x-1,\ell_1,\ell_2,t) \tag{3.29}$$

$$(-2\ell_1 - 1 \leq x \leq 2\ell_2) \;\longrightarrow\; in_4(x,\ell_1,\ell_2,t) = out_4(x+1,\ell_1,\ell_2,t) \tag{3.30}$$

These equations mean that in_1 and in_3 of each cell (except the first cell i.e. that at position $-2\ell_1 - 1$) is connected to out_1 and out_3 respectively of the previous cell, while

in_2 and in_4 of each cell (except the last cell i.e. that at position $2\ell_2 + 1$) is connected to out_2 and out_4 respectively of the next cell. The inputs to the circuit are defined by:

$$(x = -2\ell_1 - 1) \longrightarrow in_1(x, \ell_1, \ell_2, t) = Input_1(x, \ell_1, \ell_2, t) = \text{undefined} \quad (3.31)$$

$$(x = 2\ell_2 + 1) \longrightarrow in_2(x, \ell_1, \ell_2, t) = Input_2(x, \ell_1, \ell_2, t) = \text{undefined} \quad (3.32)$$

$$(x = -2\ell_1 - 1) \longrightarrow in_3(x, \ell_1, \ell_2, t) = Input_3(x, \ell_1, \ell_2, t) = \text{undefined} \quad (3.33)$$

$$(x = 2\ell_2 + 1) \longrightarrow in_4(x, \ell_1, \ell_2, t) = Input_4(x, \ell_1, \ell_2, t) = \text{undefined} \quad (3.34)$$

The above equations indicate that no input is fed into the circuit.

C. Cell Computations The computations which take place in the cells can be described by the following:

$$\begin{array}{l} (t \geq 1) \wedge (-2\ell_1 - 1 \leq x \leq 2\ell_2 + 1) \longrightarrow \\ out_1(x, \ell_1, \ell_2, t) = \textbf{case} \\ \qquad \left\{ (t + x + 1)\ mod\ 2 = 0 \right\} \implies in_1(x, \ell_1, \ell_2, t - 1) \\ \textbf{endcase} \end{array} \quad (3.35)$$

$$\begin{array}{l} (t \geq 1) \wedge (-2\ell_1 - 1 \leq x \leq 2\ell_2 + 1) \longrightarrow \\ out_2(x, \ell_1, \ell_2, t) = \textbf{case} \\ \qquad \left\{ (t + x + 1)\ mod\ 2 = 0 \right\} \implies in_2(x, \ell_1, \ell_2, t - 1) \\ \textbf{endcase} \end{array} \quad (3.36)$$

$$\begin{array}{l} (t \geq 2) \wedge (-2\ell_1 - 1 \leq x \leq 2\ell_2 + 1) \longrightarrow \\ loc(x, \ell_1, \ell_2, t) = \\ \textbf{case} \\ \qquad \left\{ \begin{array}{c} x < 0 \\ t = -x + 1 \end{array} \right\} \implies g_2(in_4(x, \ell_1, \ell_2, t - 1)) \\ \qquad \left\{ \begin{array}{c} 0 < x \\ t = x + 1 \end{array} \right\} \implies g_2(in_3(x, \ell_1, \ell_2, t - 1)) \\ \qquad \left\{ \begin{array}{c} -x + 1 < t \\ x + 1 < t \\ (t + x + 1)\ mod\ 2 = 0 \end{array} \right\} \implies \begin{array}{l} g_3(in_1(x, \ell_1, \ell_2, t - 1), in_2(x, \ell_1, \ell_2, t - 1), \\ \quad in_3(x, \ell_1, \ell_2, t - 1), in_4(x, \ell_1, \ell_2, t - 1), \\ \quad loc(x, \ell_1, \ell_2, t - 2)) \end{array} \\ \textbf{endcase} \end{array} \quad (3.37)$$

$$out_3(x, \ell_1, \ell_2, t) = loc(x, \ell_1, \ell_2, t) \quad (3.38)$$

$$out_4(x,\ell_1,\ell_2,t) \;=\; loc(x,\ell_1,\ell_2,t) \tag{3.39}$$

From (3.35), (3.36), and (3.37) we conclude that the delays of out_1, out_2, and loc are 1,1, and 2 respectively. Let $Init_1$, $Init_2$, and $LocInit$ be the initial functions corresponding to out_1, out_2, and loc respectively, then:

$$\begin{array}{l}(0 \le t < 1) \wedge (-2\ell_1 - 1 \le x \le 2\ell_2 + 1) \longrightarrow \\ \quad out_1(x,\ell_1,\ell_2,t) = Init_1(x,\ell_1,\ell_2,t)\end{array} \tag{3.40}$$

$$\begin{array}{l}(0 \le t < 1) \wedge (-2\ell_1 - 1 \le x \le 2\ell_2 + 1) \longrightarrow \\ \quad out_2(x,\ell_1,\ell_2,t) = Init_2(x,\ell_1,\ell_2,t)\end{array} \tag{3.41}$$

$$\begin{array}{l}(0 \le t < 2) \wedge (-2\ell_1 - 1 \le x \le 2\ell_2 + 1) \longrightarrow \\ \quad loc(x,\ell_1,\ell_2,t) = LocInit(x,\ell_1,\ell_2,t)\end{array} \tag{3.42}$$

The initial functions are defined by:

$$\begin{array}{rcl} Init_1(x,\ell_1,\ell_2,t) & = & \textbf{case} \\ & & \left\{ \begin{array}{c} -2\ell_1 - 1 \le x \le -3 \\ (x+1)\ mod\ 2 = 0 \end{array} \right\} \implies a\left(-\frac{x+1}{2}\right) \\ & & \textbf{endcase} \end{array} \tag{3.43}$$

$$\begin{array}{rcl} Init_2(x,\ell_1,\ell_2,t) & = & \textbf{case} \\ & & \left\{ \begin{array}{c} 3 \le x \le 2\ell_2 + 1 \\ (x+1)\ mod\ 2 = 0 \end{array} \right\} \implies b\left(\frac{x-1}{2}\right) \\ & & \textbf{endcase} \end{array} \tag{3.44}$$

$$\begin{array}{rcl} LocInit(x,\ell_1,\ell_2,t) & = & \textbf{case} \\ & & \left\{ x = 0 \right\} \implies g_1 \\ & & \textbf{endcase} \end{array} \tag{3.45}$$

Equation (3.43) means that an element of a is preloaded into out_1 of each other cell starting at $x = -3$ and finishing at $x = -2\ell_1 - 1$. Equation (3.44) means that an element of b is preloaded into out_2 of each cell starting at $x = 3$ and finishing at $x = 2\ell_2 + 1$. Equation (3.45) indicates that g_1 is preloaded into loc of the cell in the middle (that at position 0), while no initial values are preloaded into the local variables of the other cells.

3.2.2 Specification of the String Matching Circuit

The specification of the circuit tells us that we get the difference between the sequences a and b from loc of the cell at position $\ell_2 - \ell_1$ at time $\ell_1 + \ell_2 + 1$. Formally:

$$\begin{array}{l}spec(x,\ell_1,\ell_2,t) = \\ \quad (0 < \ell_1) \wedge (0 < \ell_2) \wedge (x = \ell_2 - \ell_1) \wedge (t = \ell_1 + \ell_2 + 1) \longrightarrow \\ \qquad (loc(x,\ell_1,\ell_2,t) \;=\; d(\ell_1,\ell_2,\ell_1,\ell_2))\end{array} \tag{3.46}$$

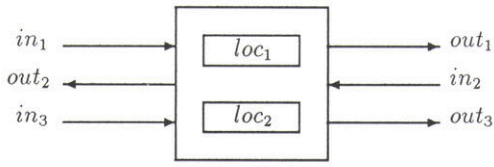

Figure 3.7: A cell in the substring detecting circuit

where:

$$
\begin{array}{l}
d(y_1, y_2, \ell_1, \ell_2) = \\
\textbf{case} \\
\qquad \left\{ \begin{array}{l} y_1 = 0 \\ y_2 = 0 \end{array} \right\} \implies g_1 \\
\qquad \left\{ \begin{array}{l} 0 < y_1 \leq \ell_1 \\ y_2 = 0 \end{array} \right\} \implies g_2(d(y_1 - 1, y_2, \ell_1, \ell_2)) \\
\qquad \left\{ \begin{array}{l} y_1 = 0 \\ 0 < y_2 \leq \ell_2 \end{array} \right\} \implies g_2(d(y_1, y_2 - 1, \ell_1, \ell_2)) \\
\qquad \left\{ \begin{array}{l} 0 < y_1 \leq \ell_1 \\ 0 < y_2 \leq \ell_2 \end{array} \right\} \implies g_3(\ a(y_1), b(y_2), \\
\qquad\qquad\qquad d(y_1, y_2 - 1, \ell_1, \ell_2), d(y_1 - 1, y_2, \ell_1, \ell_2), \\
\qquad\qquad\qquad d(y_1 - 1, y_2 - 1, \ell_1, \ell_2)) \\
\textbf{endcase}
\end{array}
\tag{3.47}
$$

3.3 A Substring Detecting Circuit

In this section we introduce a substring detecting circuit, which given two strings $a(1)$, $\ldots, a(\ell_1)$ and $b(1), \ldots, b(\ell_2)$, checks whether one of the strings is a (contiguous) substring of the other. The substring relation can be defined as $substr(\ell_1, \ell_2)$, where[2]:

$$
substr(y_1, y_2) = \left\{ \begin{array}{ll} \bigvee_{j=0}^{y_1 - y_2} \bigwedge_{i=0}^{y_2 - 1} (a(i + j + 1) = b(i + 1)) & \text{if } y_1 \geq y_2 \\ \bigvee_{j=0}^{y_2 - y_1} \bigwedge_{i=0}^{y_1 - 1} (a(i + 1) = b(i + j + 1)) & \text{if } y_2 > y_1 \end{array} \right. \tag{3.48}
$$

The circuit consists of an array of ℓ_1 cells where each cell has three inputs, three outputs, and two local variables (see figures 3.7 and 3.8).

The elements of the string a are stored inside the cells of the circuit, while the elements of the string b are input from the left. The element $a(i)$ is stored inside the first local

[2]Notice that $\bigvee_{i=0}^{\ell(\overline{x})} b(\overline{x}, i)$ and $\bigwedge_{i=0}^{\ell(\overline{x})} b(\overline{x}, i)$ can be described by recurrence equations in a similar manner to $\sum_{i=0}^{\ell(\overline{x})} r(\overline{x}, i)$ (see section 3.1).

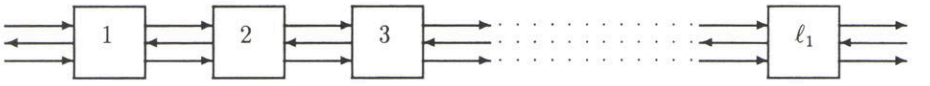

Figure 3.8: The substring detecting circuit

variable of the i^{th} cell (the cell at position i). The elements of b are input, one element each other clock cycle, via the first input of the first cell.

The elements of b move one step to the right at each clock cycle. In addition to the stream of elements of b moving to the right, two other streams flow inside the circuit. One stream (which we call stream 2) moves through the second inputs and outputs of the cells, one step at each other clock cycle, from the right to the left. Another stream (which we call stream 3) moves through the third inputs and outputs of the cells, one step at each clock cycle, from the left to the right.

When an element $b(j)$ enters a cell i the elements $a(i)$ and $b(j)$ are compared for equality. The element of stream 2 entering the cell from the right at the same time as $b(j)$ indicates the equality of the two substrings:

$$a(i+1)\, a(i+2)\, \cdots\, a(\ell_1)$$

and:

$$b(j+1)\, b(j+2)\, \cdots\, b(j+\ell_1-i)$$

if $\ell_1 - i \leq \ell_2 - j$, and the equality of the two substrings:

$$a(i+1)\, a(i+2)\, \cdots\, a(i+\ell_2-j)$$

and:

$$b(j+1)\, b(j+2)\, \cdots\, b(\ell_2)$$

if $\ell_2 - j \leq \ell_1 - i$. The value of the element of stream 2 entering the cell is "and"ed with the result of comparison of $a(i)$ and $b(j)$. Thus the new value of the element of stream 2, leaving the cell at the next clock cycle, indicates the equality of the two substrings:

$$a(i)\, a(i+1)\, a(i+2)\, \cdots\, a(\ell_1)$$

and:

$$b(j)\, b(j+1)\, b(j+2)\, \cdots\, b(j+\ell_1-i)$$

if $\ell_1 - i \leq \ell_2 - j$, and the equality of the two substrings:

$$a(i)\, a(i+1)\, a(i+2)\, \cdots\, a(i+\ell_2-j)$$

and:

$$b(j)\, b(j+1)\, b(j+2)\, \cdots\, b(\ell_2)$$

if $\ell_2 - j \leq \ell_1 - i$.

Also, when $b(j)$ enters the cell i, the element of stream 3, entering the cell at the same time from the left, indicates whether the string:

$$b(j)\ b(j+1)\ \cdots\ b(\ell_2)$$

is equal to one of the strings:

$$\begin{array}{cccc} a(1) & a(2) & \cdots & a(\ell_2 - j + 1) \\ a(2) & a(3) & \cdots & a(\ell_2 - j + 2) \\ & & \vdots & \\ a(i-1) & a(i) & \cdots & a(\ell_2 - j + i - 1) \end{array}$$

The value of this element of stream 3 is "or"ed with the value of the element of stream 2 entering the cell at the same time. Thus the new value of the element of stream 3, leaving the cell at the next clock cycle, indicates whether the string:

$$b(j)\ b(j+1)\ \cdots\ b(\ell_2)$$

is equal to one of the strings:

$$\begin{array}{cccc} a(1) & a(2) & \cdots & a(\ell_2 - j + 1) \\ a(2) & a(3) & \cdots & a(\ell_2 - j + 2) \\ & & \vdots & \\ a(i-1) & a(i) & \cdots & a(\ell_2 - j + i - 1) \\ a(i) & a(i+1) & \cdots & a(\ell_2 - j + i) \end{array}$$

The value of $substr(\ell_1, \ell_2)$ can be picked as follows:

- If $\ell_1 \geq \ell_2$. When the element $b(\ell_2)$ enters the cell $\ell_1 - \ell_2 + 1$, the value of the element of stream 3 leaving the cell at the next clock cycle is equal to $substr(\ell_1, \ell_2)$. This value is copied to the second local variable of the cell, from which it can be fetched.

- If $\ell_2 \geq \ell_1$. When the element $b(j)$, where $j \geq \ell_1$ enters the first cell of the circuit, the value of the element of stream 2 entering the first cell at the same time indicates the equality of the two substrings:

  $$a(2)\ a(3)\ \cdots a(\ell_1)$$

  and:

  $$b(j+1)\ b(j+2)\ \cdots\ b(j+\ell_1-1)$$

  This value is "and"ed with the result of comparison of $b(j)$ and $a(1)$, the result indicating the equality of the string a to the substring of b defined by:

  $$b(j)\ b(j+1)\ b(j+2)\ \cdots\ b(j+\ell_1-1)$$

  This last value is "or"ed with the contents of the second local variable of the first cell. The value $substr(\ell_1, \ell_2)$ can be fetched from the second local variable of the first cell one clock cycle after that $b(1)$ has entered the cell.

In section 3.3.1 we will give a formal description of the implementation of the substring detecting circuit. In section 3.3.2 we will give a formal specification of the circuit.

3.3.1 Implementation of the Substring Detecting Circuit

We will structure the formal description of the implementation of the substring detecting circuit according to the scheme presented in section 2.2.

A. Topology The substring detecting circuit consists of an array of ℓ_1 cells. The topology of the circuit can be described as:

$$top(x, \ell_1, \ell_2) \;=\; 1 \leq x \leq \ell_1$$

This means that we place a cell at each integer point between (and including) 1 and ℓ_1. There are two circuit parameters ℓ_1 and ℓ_2. The circuit parameter ℓ_1 decides the length of the array. The predicate *top* describes the topology of a family of substring detecting circuits; one for each value of the lengths ℓ_1 and ℓ_2 of the sequences a and b.

B. Interconnections The connection vectors are defined by $\delta_1 =< -1 >$, $\delta_2 =< 1 >$, and $\delta_3 =< -1 >$. Thus we get the following equations:

$$\begin{aligned}
(2 \leq x \leq \ell_1) &\longrightarrow in_1(x, \ell_1, \ell_2, t) = out_1(x-1, \ell_1, \ell_2, t) \\
(1 \leq x \leq \ell_1 - 1) &\longrightarrow in_2(x, \ell_1, \ell_2, t) = out_2(x+1, \ell_1, \ell_2, t) \\
(2 \leq x \leq \ell_1) &\longrightarrow in_3(x, \ell_1, \ell_2, t) = out_3(x-1, \ell_1, \ell_2, t)
\end{aligned}$$

These equations mean that in_1 and in_3 of each cell (except the first cell i.e. that at position 1) is connected to out_1 and out_3 respectively of the previous cell, while in_2 of each cell (except the last cell i.e. that at position ℓ_1) is connected to out_3 of the next cell. The inputs $Input_1$ and $Input_3$ are fed into the circuit via in_1 and in_3 respectively of the first cell, while $Input_2$ is fed into the circuit via in_2 of the last cell.

$$\begin{aligned}
(x = 1) &\longrightarrow in_1(x, \ell_1, \ell_2, t) = Input_1(x, \ell_1, \ell_2, t) \\
(x = \ell_1) &\longrightarrow in_2(x, \ell_1, \ell_2, t) = Input_2(x, \ell_1, \ell_2, t) \\
(x = 1) &\longrightarrow in_3(x, \ell_1, \ell_2, t) = Input_3(x, \ell_1, \ell_2, t)
\end{aligned}$$

The input functions are given by:

$$\begin{aligned}
Input_1(x, \ell_1, \ell_2, t) \;=\; & \textbf{case} \\
& \left\{ \begin{array}{l} t \bmod 2 = 0 \\ t \leq 2\ell_2 - 2 \end{array} \right\} \;\Longrightarrow\; b\left(\tfrac{2\ell_2 - t}{2}\right) \\
& \textbf{endcase} \\
Input_2(x, \ell_1, \ell_2, t) \;=\; & \textbf{case} \\
& \left\{ \ell_1 + 1 \leq t \right\} \;\Longrightarrow\; tt \\
& \textbf{endcase} \\
Input_3(x, \ell_1, \ell_2, t) \;=\; & ff
\end{aligned}$$

where tt stands for the value *true* and ff stands for the value *false*. Note that $Input_1$ defines the input scheduling of the circuit, i.e. how the sequence b actually is input to the circuit.

C. Cell Computations The computations which take place in the cells can be described by the following:

$$(t \geq 1) \wedge (1 \leq x \leq \ell_1) \longrightarrow out_1(x, \ell_1, \ell_2, t) = in_1(x, \ell_1, \ell_2, t-1)$$

$$\begin{array}{l} (t \geq 1) \wedge (1 \leq x \leq \ell_1) \longrightarrow \\ out_2(x, \ell_1, \ell_2, t) = \\ \textbf{case} \\ \quad \left\{ \begin{array}{c} (t+x)\ mod\ 2 = 0 \\ t = x \end{array} \right\} \Longrightarrow (loc_1(x, \ell_1, \ell_2, t-1) = in_1(x, \ell_1, \ell_2, t-1)) \\ \quad \left\{ \begin{array}{c} (t+x)\ mod\ 2 = 0 \\ t > x \end{array} \right\} \Longrightarrow \left(\begin{array}{c} in_2(x, \ell_1, \ell_2, t-1) \\ \wedge \\ (loc_1(x, \ell_1, \ell_2, t-1) = in_1(x, \ell_1, \ell_2, t-1)) \end{array} \right) \\ \textbf{endcase} \end{array}$$

$$\begin{array}{l} (t \geq 1) \wedge (1 \leq x \leq \ell_1) \longrightarrow \\ out_3(x, \ell_1, \ell_2, t) = \\ \textbf{case} \\ \quad \left\{ \begin{array}{c} (t+x)\ mod\ 2 = 0 \\ t = x \end{array} \right\} \Longrightarrow \left(\begin{array}{c} in_3(x, \ell_1, \ell_2, t-1) \\ \vee \\ \left(\begin{array}{c} loc_1(x, \ell_1, \ell_2, t-1) \\ = \\ in_1(x, \ell_1, \ell_2, t-1) \end{array} \right) \end{array} \right) \\ \quad \left\{ \begin{array}{c} (t+x)\ mod\ 2 = 0 \\ t > x \\ t + x \leq 2\ell_1 \end{array} \right\} \Longrightarrow \left(\begin{array}{c} in_3(x, \ell_1, \ell_2, t-1) \\ \vee \\ \left(\begin{array}{c} in_2(x, \ell_1, \ell_2, t-1) \\ \wedge \\ \left(\begin{array}{c} loc_1(x, \ell_1, \ell_2, t-1) \\ = \\ in_1(x, \ell_1, \ell_2, t-1) \end{array} \right) \end{array} \right) \end{array} \right) \\ \textbf{endcase} \end{array}$$

$$(t \geq 1) \wedge (1 \leq x \leq \ell_1) \longrightarrow loc_1(x, \ell_1, \ell_2, t) = loc_1(x, \ell_1, \ell_2, t-1)$$

$$(t \geq 1) \wedge (1 \leq x \leq \ell_1) \longrightarrow$$
$$loc_2(x, \ell_1, \ell_2, t) =$$
case

$$\left\{ \begin{array}{c} t = 2\ell_1 - x \\ x = \ell_1 \end{array} \right\} \Longrightarrow \left(\begin{array}{c} in_3(x, \ell_1, \ell_2, t-1) \\ \vee \\ \left(\begin{array}{c} loc_1(x, \ell_1, \ell_2, t-1) \\ = \\ in_1(x, \ell_1, \ell_2, t-1) \end{array} \right) \end{array} \right)$$

$$\left\{ \begin{array}{c} t = 2\ell_1 - x \\ x < \ell_1 \end{array} \right\} \Longrightarrow \left(\begin{array}{c} in_3(x, \ell_1, \ell_2, t-1) \\ \vee \\ \left(\begin{array}{c} in_2(x, \ell_1, \ell_2, t-1) \\ \wedge \\ \left(\begin{array}{c} loc_1(x, \ell_1, \ell_2, t-1) \\ = \\ in_1(x, \ell_1, \ell_2, t-1) \end{array} \right) \end{array} \right) \end{array} \right)$$

$$\left\{ \begin{array}{c} t > 2\ell_1 - x \\ x = 1 \\ (t+1)\ mod\ 2 = 0 \end{array} \right\} \Longrightarrow \left(\begin{array}{c} loc_2(x, \ell_1, \ell_2, t-1) \\ \vee \\ \left(\begin{array}{c} in_2(x, \ell_1, \ell_2, t-1) \\ \wedge \\ \left(\begin{array}{c} loc_1(x, \ell_1, \ell_2, t-1) \\ = \\ in_1(x, \ell_1, \ell_2, t-1) \end{array} \right) \end{array} \right) \end{array} \right)$$

$$\left\{ \begin{array}{c} t > 2\ell_1 - x \\ x = 1 \\ t\ mod\ 2 = 0 \end{array} \right\} \Longrightarrow loc_2(x, \ell_1, \ell_2, t-1)$$

endcase

From the above equations we conclude that the delay of each of out_1, out_2, out_3, loc_1, and loc_2 is 1. The period of initialization is defined by:

$$(0 \leq t < 1) \wedge (1 \leq x \leq \ell_1) \longrightarrow out_1(x, \ell_1, \ell_2, t) = OutInit_1(x, \ell_1, \ell_2, t) = \text{undefined}$$
$$(0 \leq t < 1) \wedge (1 \leq x \leq \ell_1) \longrightarrow out_2(x, \ell_1, \ell_2, t) = OutInit_2(x, \ell_1, \ell_2, t) = \text{undefined}$$
$$(0 \leq t < 1) \wedge (1 \leq x \leq \ell_1) \longrightarrow out_3(x, \ell_1, \ell_2, t) = OutInit_3(x, \ell_1, \ell_2, t) = \text{undefined}$$
$$(0 \leq t < 1) \wedge (1 \leq x \leq \ell_1) \longrightarrow loc_1(x, \ell_1, \ell_2, t) = LocInit(x, \ell_1, \ell_2, t) = a(x)$$
$$(0 \leq t < 1) \wedge (1 \leq x \leq \ell_1) \longrightarrow loc_2(x, \ell_1, \ell_2, t) = LocInit(x, \ell_1, \ell_2, t) = \text{undefined}$$

The above equations indicate that the element $a(i)$ is preloaded to loc_1 of cell i, while no initial values are preloaded to the other wires and local variables of the circuit.

3.3.2 Specification of the Substring Detecting Circuit

The specification of the substring detecting circuit is of the following form:

$$\begin{array}{c} spec(x,\ell_1,\ell_2,t) = \\ (1 \le \ell_2) \wedge (\ell_2 \le \ell_1) \wedge (t = \ell_1 + \ell_2 - 1) \wedge (x = \ell_1 - \ell_2 + 1) \longrightarrow \\ (loc_2(x,\ell_1,\ell_2,t) = substr(\ell_1,\ell_2)) \\ \wedge \\ (1 \le \ell_1) \wedge (\ell_1 < \ell_2) \wedge (x = 1) \wedge (t = 2\ell_2) \longrightarrow \\ (loc_2(x,\ell_1,\ell_2,t) = substr(\ell_1,\ell_2)) \end{array}$$

From equation (3.48) it follows that:

$$\begin{array}{c} spec(x,\ell_1,\ell_2,t) = \\ (1 \le \ell_2) \wedge (\ell_2 \le \ell_1) \wedge (t = \ell_1 + \ell_2 - 1) \wedge (x = \ell_1 - \ell_2 + 1) \longrightarrow \\ \left(loc_2(x,\ell_1,\ell_2,t) = \bigvee_{j=0}^{\ell_1-\ell_2} \bigwedge_{i=0}^{\ell_2-1} (a(i+j+1) = b(i+1))\right) \\ \wedge \\ (1 \le \ell_1) \wedge (\ell_1 < \ell_2) \wedge (x = 1) \wedge (t = 2\ell_2) \longrightarrow \\ \left(loc_2(x,\ell_1,\ell_2,t) = \bigvee_{j=0}^{\ell_2-\ell_1} \bigwedge_{i=0}^{\ell_1-1} (a(i+1) = b(i+j+1))\right) \end{array} \tag{3.49}$$

Notice that the specification defines the output scheduling of the circuit, i.e. the time instants at which to expect the elements of the output.

Chapter 4

Verification of Systolic Circuits

In this chapter we define formally the notion of verification of systolic circuits. In sections 4.1 and 4.2 we will summarize some basic ideas of the theories of abstract data types and fixed points. In section 4.3 we use the ideas of sections 4.1 and 4.2 to define a formal semantics for systems of recurrence equations. Based on the semantics definition we will give the formal meaning of equality among systems of recurrence equations. In section 4.4 we will show, based on the results of section 4.3, how implementations and specifications of systolic circuits can be compared for equality and hence how the notion of formal verification of these circuits can be defined. In section 4.5 we will describe how decision methods can be found to carry out automatic verification of different classes of systolic circuits.

4.1 Signatures, Algebras, and Algebraic Specifications

In this section we describe some basic ideas concerning the theory of abstract data types. For more details see e.g. [MG85], [EM85], or [GTW78].

A *signature* $\mathcal{SIG}$ is a pair $< \mathcal{S}, \mathcal{G} >$, where $\mathcal{S}$ is the set of *sorts* of $\mathcal{SIG}$, and $\mathcal{G}$ is the set of *operation symbols* of $\mathcal{SIG}$. Each $g \in \mathcal{G}$ has a *domain sort* $w \in \mathcal{S}^*$, and a *range sort* $S \in \mathcal{S}$. The set $\mathcal{K} \subseteq \mathcal{G}$ such that $g \in \mathcal{K}$ iff g has an empty domain sort is called the set of *constant operation symbols* of $\mathcal{SIG}$.

A *$\mathcal{SIG}$-algebra* A, where $\mathcal{SIG} =< \mathcal{S}, \mathcal{G} >$, is a a pair $< \mathcal{S}^A, \mathcal{G}^A >$, where $\mathcal{S}^A$ is the set of *domains* of A, and $\mathcal{G}^A$ is the set of *operations* of A. The set of domains $\mathcal{S}^A$ is of the form $\{S^A;\ S \in \mathcal{S}\}$, where each S^A is a set, and is called a *domain* of A. The set of operations $\mathcal{G}^A$ is of the form $\{g^A;\ g \in \mathcal{G}\}$, where each g^A is a function of type:

$$g^A : S_1^A \times \cdots \times S_n^A \longrightarrow S^A$$

if g has a domain sort $S_1 \cdot \cdots \cdot S_n$, and a range sort S. Each g^A is called an *operation* of the algebra.

For each $S \in \mathcal{S}$ we assume that we have an infinite set of variables X_S which we call the set of variables of sort S. We call $X = \bigcup_{S \in \mathcal{S}} X_S$ the set of variables of $\mathcal{SIG}$.

The set T_S of *terms* of sort S are defined as follows:

- If $g \in \mathcal{K}$ has a range sort S then $g \in T_S$.

- If $x \in X_S$ then $x \in T_S$.

- If $t_1, \ldots, t_n$ are terms of sorts $S_1, \ldots, S_n$ respectively, and $g \in \mathcal{G}$ has a domain sort $S_1 \cdot \cdots \cdot S_n$ and a range sort S, then $g(t_1, \ldots, t_n) \in T_S$.

The set $T_{\mathcal{SIG}}$ of terms of $\mathcal{SIG}$ is defined by $\bigcup_{S \in \mathcal{S}} T_S$.

Let A be a $\mathcal{SIG}$-algebra. An *assignment* θ is a function which maps the elements of X_S into S^A, for each $S \in \mathcal{S}$. That is if θ is an assignment and $x \in X_S$ then $\theta(x) \in S^A$.

Given a $\mathcal{SIG}$-algebra A and an assignment θ, the *evaluation* θ' of the elements of $T_{\mathcal{SIG}}$ is defined as follows:

- If $g \in \mathcal{K}$ then $\theta'(g) = g^A$.

- if $x \in X$ then $\theta'(x) = \theta(x)$.

- If $g(t_1, \ldots, t_n) \in T_{\mathcal{SIG}}$ then $\theta'(g(t_1, \ldots, t_n)) = g^A(\theta'(t_1), \ldots, \theta'(t_n))$

A $\mathcal{SIG}$-equation is a pair $< L, R >$, where $L, R \in T_{\mathcal{SIG}}$. We write an equation in the usual notation $L = R$.

A $\mathcal{SIG}$-algebra A is said to satisfy a $\mathcal{SIG}$-equation $L = R$, iff for all assignments θ in A, we have $\theta'(L) = \theta'(R)$.

An *algebraic specification* $\mathcal{SPEC}$ is a triple $< \mathcal{S}, \mathcal{G}, \mathcal{E} >$, where $\mathcal{SIG} =< \mathcal{S}, \mathcal{G} >$ is a signature, and $\mathcal{E}$ is a set of $\mathcal{SIG}$-equations.

A $\mathcal{SIG}$-algebra A is said to be a $\mathcal{SPEC}$-algebra, where $\mathcal{SPEC} =< \mathcal{SIG}, \mathcal{E} >$, iff A satisfies each equation in $\mathcal{E}$.

4.2 Functionals and Fixed Points

In this section we give some basic results of fixed point theory. The results are simple modifications of the theory found in e.g. [Man74].

Assume that we have a set D of sets, and an element $\perp$ (representing the value *undefined*), such that $\perp \notin d$, for each $d \in D$. We define:

$$D^+ = \{d \cup \{\perp\} \; ; \; d \in D\}$$

and:

$$D^* = D \cup D^+$$

A *partial function* is a function of type $d_1^* \times \cdots \times d_n^* \rightarrow d_{n+1}^*$, where $d_i^* \in D^*$, for $1 \leq i \leq n+1$.

We define a partial order $\sqsubseteq$ on the elements of each $d^* \in D^*$ in the following way:

- $z \sqsubseteq z$, for each $z \in d^*$.

- $\perp \sqsubseteq z$, for each $z \in d^*$, if $\perp \in d^*$.

The relation $\sqsubseteq$ can be extended to n-tuples in the following way:

$$< z_1, \ldots, z_n > \sqsubseteq < w_1, \ldots, w_n >$$

if $z_i \sqsubseteq w_i$, for $1 \leq i \leq n$, where $w_i, z_i \in d_i^*$, for some $d_i^* \in D^*$.

A partial function h is said to be *monotonic* if:

$$(\overline{z}_1 \sqsubseteq \overline{z}_2) \longrightarrow (h(\overline{z}_1) \sqsubseteq h(\overline{z}_2))$$

for each $\overline{z}_1$ and $\overline{z}_2$.

A partial function h is said to be *strict* if $h(z_1, \ldots, z_n) = \perp$ whenever $z_i = \perp$ for some $1 \leq i \leq n$.

Proposition 1 *Each partial function with type* $d_1 \times \cdots \times d_n \rightarrow d^*$, *where* $d_i \in D$, *for* $1 \leq i \leq n$, *is strict.*

Proof: The proof follows immediately from the definition of strictness, and from the fact that $\perp \notin d_i$. □

Given a partial function h of type $d_1 \times \cdots \times d_n \rightarrow d_{n+1}$, where $d_i \in D$, for $1 \leq i \leq n+1$, the *natural extension* h^+ of h is a partial function of type $d_1^+ \times \cdots \times d_n^+ \rightarrow d_{n+1}^+$, where $d_i^+ = d_i \cup \{\perp\}$, for $1 \leq i \leq n+1$, given by:

$$h^+(z_1, \ldots, z_n) = \begin{cases} h(\overline{z}) & \text{if } z_i \neq \perp \text{ for } 1 \leq i \leq n \\ \perp & \text{if } z_i = \perp \text{ for some } 1 \leq i \leq n \end{cases}$$

It is clear that the natural extension of each function is strict.

Lemma 2 *Every strict partial function is monotonic.*

Proof: Suppose that h is strict but not monotonic. Then there are $\overline{z}_1$ and $\overline{z}_2$ such that $\overline{z}_1 \sqsubseteq \overline{z}_2$ and $h(\overline{z}_1) \not\sqsubseteq h(\overline{z}_2)$. It can be shown that the strictness of h implies that $h(\overline{z}_1) = \perp$, and consequently $h(\overline{z}_1) \sqsubseteq h(\overline{z}_2)$, which is a contadiction. □

Lemma 3 *The composition of monotonic partial functions is monotonic.*

Proof: Let $h(\overline{z})$ be of the form:

$$h'(h_1(\overline{z}_1), \ldots, h_n(\overline{z}_n))$$

where $h', h_1, \ldots, h_n$ are monotonic. The claim follows from:

$$(\overline{w} \sqsubseteq \overline{z}) \longrightarrow (\forall (1 \leq i \leq n).\ (\overline{w}_i \sqsubseteq \overline{z}_i)) \longrightarrow$$
$$(\forall (1 \leq i \leq n).\ (h_i(\overline{w}_i) \sqsubseteq h_i(\overline{z}_i))) \longrightarrow (h(\overline{w}) \sqsubseteq h(\overline{z}))$$

□

Let h_1 and h_2 be partial functions. The relation $h_1 \sqsubseteq h_2$ (read h_1 *is less defined than or equal to* h_2) is defined as:

$$(h_1 \sqsubseteq h_2) \quad \text{iff} \quad \forall \overline{z}.\ (h_1(\overline{z}) = h_2(\overline{z}))$$

The relation $h_1 = h_2$ (read h_1 *is equal to* h_2) is defined by:

$$(h_1 = h_2) \quad \text{iff} \quad (h_1 \sqsubseteq h_2) \wedge (h_2 \sqsubseteq h_1)$$

Let $h_0, h_1, h_2, \ldots$ be a sequence of monotonic partial functions, denoted $\{h_i\}$, then $\{h_i\}$ is called a *chain* if:

$$h_0 \sqsubseteq h_1 \sqsubseteq h_2 \sqsubseteq \cdots$$

We say that h is an *upper bound* of the chain if:

$$h_i \sqsubseteq h$$

for $i \geq 0$. We say that h is a *least upper bound* of the chain, denoted $\sqcup\{h_i\}$, if h is an upper bound, and for each other upper bound h', $h \sqsubseteq h'$.

Lemma 4 *Each chain has a least upper bound.*

Proof: We define $\sqcup\{h_i\}$ as follows:

$$\sqcup\{h_i\}(\overline{z}) \quad = \quad \begin{cases} w & \text{if } h_i(\overline{z}) = w \text{ for some } i \geq 0 \\ \perp & \text{if } h_i(\overline{z}) = \perp \text{ for each } i \geq 0 \end{cases}$$

It can be easily checked that $\sqcup\{h_i\}$ exists and that it is the least upper bound of $\{h_i\}$. □

A *basic functional*[1] $\mathcal{F}$, of type $(d_1^* \times \cdots \times d_n^* \rightarrow d^+) \rightarrow (d_1^* \times \cdots \times d_n^* \rightarrow d^+)$, where $d_1^*, \ldots, d_n^* \in D^*$ and $d^+ \in D^+$, maps the set of partial functions of type $d_1^* \times \cdots \times d_n^* \rightarrow d^+$ into itself. That is, if $\mathcal{F}(f)$ is a functional of type $(d_1^* \times \cdots \times d_n^* \rightarrow d^+) \rightarrow (d_1^* \times \cdots \times d_n^* \rightarrow d^+)$, and h is partial function of type $d_1^* \times \cdots \times d_n^* \rightarrow d^+$ then $\mathcal{F}(h)$ is a partial function of type $d_1^* \times \cdots \times d_n^* \rightarrow d^+$.

A basic functional $\mathcal{F}(f)$ is said to be *monotonic* if:

$$(h_1 \sqsubseteq h_2) \quad \longrightarrow \quad (\mathcal{F}(h_1) \sqsubseteq \mathcal{F}(h_2))$$

for each two partial functions h_1 and h_2.

A basic functional is said to be *continuous* if:

$$\mathcal{F}(\sqcup\{h_i\}) \quad = \quad \sqcup\{\mathcal{F}(h_i)\}$$

for each chain $\{h_i\}$.

Lemma 5 *Any basic functional $\mathcal{F}(f)$ defined by the composition of monotonic partial functions and the function variable f is continuous.*

Proof: The claim can be shown by induction on the structure of $\mathcal{F}$. □

A partial function h is said to be a *fixed point* of a basic functional $\mathcal{F}(f)$ if:

$$\mathcal{F}(h) \quad = \quad h$$

A partial function h is said to be the *least fixed point* of a basic functional $\mathcal{F}(f)$ if h is a fixed point and for each other fixed point h', $h \sqsubseteq h'$.

[1] In the literature $\mathcal{F}$ is simply called a *functional*. We reserve the term *functional* to the class we introduced in section 2.1, and which we will use throughout the later chapters

Lemma 6 *Every continuous basic functional has a least fixed point. Furthermore the least fixed point $f_{\mathcal{F}}$ of $\mathcal{F}(f)$ is defined by:*

$$f_{\mathcal{F}} = \sqcup\{\mathcal{F}^i(\Omega)\}$$

where $\Omega(\overline{z}) = \perp$ for every $\overline{z}$.

Proof: That $f_{\mathcal{F}}$ is a fixed point of $\mathcal{F}(f)$ follows from:

$$\mathcal{F}(f_{\mathcal{F}}) = \mathcal{F}(\sqcup\{\mathcal{F}^i(\Omega)\}) = \sqcup\{\mathcal{F}^{i+1}(\Omega)\} = \sqcup\{\mathcal{F}^i(\Omega)\} = f_{\mathcal{F}}$$

Now suppose that h is a fixed point of $\mathcal{F}$. It can be shown by induction on i that:

$$\mathcal{F}^i(\Omega) \sqsubseteq h$$

for each $i \geq 0$. Since $f_{\mathcal{F}}$ is the least upper bound of $\{\mathcal{F}^i(\Omega)\}$, then $f_{\mathcal{F}} \sqsubseteq g$. □

A *recursive program* is of the form:

$$h(\overline{z}) = \mathcal{F}(h)(\overline{z})$$

Where $\mathcal{F}(f)(\overline{z})$ is a basic functional defined by the composition of monotonic partial functions and the function variable f applied to $\overline{z}$. The *function defined by the recursive program* above is the least fixed point $f_{\mathcal{F}}$ of $\mathcal{F}$, and we denote it by fp. From lemma 5 it follows that $\mathcal{F}$ is continuous. From lemma 6 it follows that fp exists.

A *system of recursive programs* is of the form:

$$\begin{aligned} f_1(\overline{z}) &= \mathcal{F}_1(f_1, \ldots, f_n)(\overline{z}) \\ f_2(\overline{z}) &= \mathcal{F}_2(f_1, \ldots, f_n)(\overline{z}) \\ &\vdots \\ f_n(\overline{z}) &= \mathcal{F}_n(f_1, \ldots, f_n)(\overline{z}) \end{aligned}$$

where each $\mathcal{F}_i$ is a functional defined by the composition of monotonic partial functions and the function variables $f_1, \ldots, f_n$ applied to $\overline{z}$.

In order to interpret systems of recursive programs, we need to define the notions of monotonicity and continuity for tuples of partial functions and basic functionals.

A tuple $< h_1, \ldots, h_n >$ of partial functions is said to be *monotonic* if h_i is monotonic for $1 \leq i \leq n$. We say that:

$$< h_{11}, \ldots, h_{1n} > \sqsubseteq < h_{21}, \ldots, h_{2n} >$$

if $h_{1i} \sqsubseteq h_{2i}$, for $1 \leq i \leq n$.

Consider a tuple $\mathcal{F}(f_1, \ldots, f_n) =< \mathcal{F}_1(f_1, \ldots, f_n), \ldots, \mathcal{F}_n(f_1, \ldots, f_n) >$ of functionals, where each $\mathcal{F}_i$ maps an n-tuple of partial functions of types $ty_1, \ldots, ty_n$ respectively into a partial function of type ty_i, for $1 \leq i \leq n$. We say that $\mathcal{F}$ is *continuous* if $\mathcal{F}_i$ is continuous, for $1 \leq i \leq n$. We say that $h =< h_1, \ldots, h_n >$ is a *simultaneous fixed point* of $\mathcal{F}(f_1, \ldots, f_n)$ if:

$$h_i = \mathcal{F}_i(h_1, \ldots, h_n)$$

for $1 \leq i \leq n$. We say that h is the *simultaneous least fixed point* of $\mathcal{F}$ if h is a simultaneous fixed point and for each other simultaneous fixed point h', $h \sqsubseteq h'$.

The results of lemmas 5 and 6 still hold, namely:

Lemma 7 *Any basic functional $\mathcal{F}(f_1, \ldots, f_n)$ defined by the composition of monotonic partial functions and the function variables $f_1, \ldots, f_n$ is continuous.*

Proof: The proof is similar to that of lemma 5. □

Lemma 8 *Each continuous tuple of basic functionals:*

$$\mathcal{F}(f_1, \ldots, f_n) =< \mathcal{F}_1(f_1, \ldots, f_n), \ldots, \mathcal{F}(f_1, \ldots, f_n) >$$

has a simultaneous least fixed point.

Proof: The proof is similar to that of lemma 6. □

The *n-tuple of functions defined by the system of recursive programs* above is the simultaneous least fixed point $< f_{\mathcal{F}_1}, \ldots, f_{\mathcal{F}_n} >$ of $\mathcal{F}(f_1, \ldots, f_n)$, which we denote by $< fp_1, \ldots, fp_n >$. From lemma 7 it follows that $\mathcal{F}(f_1, \ldots, f_n)$ is continuous. From lemma 8 it follows that $\mathcal{F}(f_1, \ldots, f_n)$ has a simultaneous least fixed point.

4.3 Semantics

In this section we use the ideas of sections 4.1 and 4.2 to explain how a system of recurrence equations with a signature $\mathcal{SIG}$ can be interpreted over a $\mathcal{SIG}$-algebra A. Then we define formally the notion of equality among systems of recurrence equations.

A *stream interpretation* $\mathcal{I}$ over a $\mathcal{SIG}$-algebra A, where $\mathcal{SIG} =< \mathcal{S}, \mathcal{G} >$, is a total function which maps each stream variable into a function from tuples of integers to the domains of A. Thus if $\mathcal{I}$ is a stream interpretation then, for each $a \in \mathcal{A}$ of arity n and sort S, $\mathcal{I}(a)$ is a function of type $I^n \rightarrow S^A$. We denote $\mathcal{I}(a)$ by $a^{\mathcal{I}}$.

Suppose that E is a system of recurrence equations over a signature $\mathcal{SIG} =< \mathcal{S}, \mathcal{G} >$. We will define a formal semantics, which, given a $\mathcal{SIG}$-algebra A, and a stream interpretation $\mathcal{I}$ over A, transforms E into a system of recursive programs.

Let E be of the form[2]:

$$\begin{aligned} f_1(\overline{x}) &= \mathcal{F}_1(\overline{x}) \\ &\vdots \\ f_n(\overline{x}) &= \mathcal{F}_n(\overline{x}) \end{aligned}$$

The *interpretation* $[\![E]\!]_{A,\mathcal{I}}$ of E, under the algebra A and the stream interpretation $\mathcal{I}$ over A, is defined to be the following equation system:

$$\begin{aligned} f_1(\overline{x}) &= [\![\mathcal{F}_1(\overline{x})]\!]_{A,\mathcal{I}} \\ &\vdots \\ f_n(\overline{x}) &= [\![\mathcal{F}_n(\overline{x})]\!]_{A,\mathcal{I}} \end{aligned}$$

[2]Notice that according to the notation of section 4.2 we should write $\mathcal{F}_i(f_1, \ldots, f_n)(\overline{x})$. Nevertheless, for notation convenience, we write $\mathcal{F}_i(\overline{x})$ whenever the tuple $< f_1, \ldots, f_n >$ is known or irrelevant in the context.

where $[\![\mathcal{F}_i(\overline{x})]\!]_{A,\mathcal{I}}$ is the *interpretation* of $\mathcal{F}_i(\overline{x})$ under the algebra A and stream interpretation $\mathcal{I}$. The *interpretation* $[\![\mathcal{F}(\overline{x})]\!]_{A,\mathcal{I}}$ of a functional $\mathcal{F}(\overline{x})$ of the form:

$$\begin{array}{l}\textbf{case}\\ \quad p_1(\overline{x}) \Longrightarrow t_1(\overline{x})\\ \qquad\vdots\\ \quad p_m(\overline{x}) \Longrightarrow t_m(\overline{x})\\ \textbf{endcase}\end{array}$$

is defined by:

$$[\![\mathcal{F}(\overline{x})]\!]_{A,\mathcal{I}} \;=\; \begin{cases} [\![t_i(\overline{x})]\!]_{A,\mathcal{I}} & \text{if } p_i(\overline{x}) \text{ is true} \\ \\ \bot & \text{if } p_i(\overline{x}) \text{ is false for } 1 \le i \le m \end{cases}$$

where $[\![t_i(\overline{x})]\!]_{A,\mathcal{I}}$ is the *interpretation* of $t_i(\overline{x})$ under the algebra A and stream interpretation $\mathcal{I}$. The *interpretation* $[\![t(\overline{x})]\!]_{A,\mathcal{I}}$, of a term $t(\overline{x})$ is defined by the following:

- If $t(\overline{x})$ is of the form $a(q_1(\overline{x}),\ldots,q_k(\overline{x}))$ then:

$$[\![t(\overline{x})]\!]_{A,\mathcal{I}} \;=\; a^{\mathcal{I}+}(q_1(\overline{x}),\ldots,q_k(\overline{x}))$$

  where $a^{\mathcal{I}+}$ is defined by the following:

$$a^{\mathcal{I}+}(x_1,\ldots,x_k) \;=\; \begin{cases} a^{\mathcal{I}}(x_1,\ldots,x_k) & \text{if } x_i \text{ is an integer for } 1 \le i \le k \\ \\ \bot & \text{if } x_i \text{ is not an integer for some } 1 \le i \le k \end{cases}$$

- If $t(\overline{x})$ is of the form $f(q_1(\overline{x}),\ldots,q_k(\overline{x}))$ then:

$$[\![t(\overline{x})]\!]_{A,\mathcal{I}} \;=\; f(q_1(\overline{x}),\ldots,q_k(\overline{x}))$$

- If $t(\overline{x})$ is of the form $g(t_1(\overline{x}),\ldots,t_k(\overline{x})$ then:

$$[\![t(\overline{x})]\!]_{A,\mathcal{I}} \;=\; g^{A+}([\![t_1(\overline{x})]\!]_{A,\mathcal{I}},\ldots,[\![t_k(\overline{x})]\!]_{A,\mathcal{I}})$$

  where g^{A+} is the natural extension of g^A.

It can easily be checked that $[\![E]\!]_{A,\mathcal{I}}$ is a system of recursive programs. Now we will show that our semantics is well-defined in the sense that the basic functionals $[\![\mathcal{F}_i(\overline{x})]\!]_{A,\mathcal{I}}$ in $[\![E]\!]_{A,\mathcal{I}}$ are continuous, and consequently $[\![E]\!]_{A,\mathcal{I}}$ has a simultaneous least fixed point.

We note that the basic functionals in $[\![E]\!]_{A,\mathcal{I}}$ are built by the composition of the function variables $f_1,\ldots,f_n$ and the following partial functions:

- $g^{A+} : S_1^{A+} \times \cdots \times S_m^{A+} \to S^{A+}$, for each $g \in \mathcal{G}$, where g^{A+} is the natural extension of g^A, g has a domain sort $S_1 \cdot \cdots \cdot S_m$, and range sort S.
- $q : I^n \to Q$, where $q(x_1,\ldots,x_n)$ is a QI-polynomial.
- $q : I^n \to I$, where $q(x_1,\ldots,x_n)$ is an integer polynomial.

- $a^{\mathcal{I}+} : Q^n \rightarrow S^{A+}$, for each stream variable $a \in \mathcal{A}$, where a has an arity n and a sort S.
- **case-endcase** : $B^n \times (S^{A+})^n \rightarrow S^{A+}$, where B denotes the booleans and $S \in \mathcal{S}$.
- $p : I^n \rightarrow B$, where $p(\overline{x})$ is a predicate over the integers.

We will show that the above partial functions, from which the basic functionals in $[\![E]\!]_{A,\mathcal{I}}$ are built, all are monotonic. This is achieved by the following lemma:

Lemma 9 *Let E be a system of recurrence equations over a signature $\mathcal{SIG}$. Then for each $\mathcal{SIG}$-algebra A and stream interpretation $\mathcal{I}$ over A, the basic functionals in $[\![E]\!]_{A,\mathcal{I}}$ are built from the composition of function variables and monotonic partial functions.*

Proof: The strictness of q, $a^{\mathcal{I}+}$, and p follows from proposition 1. Also g^{A+} is strict by definition. The monotonicity of q, $a^{\mathcal{I}+}$, p, and g^{A+} follows from lemma 2.

We will show that **case-endcase** is also monotonic. We will first show that **if-then-else** of type $B^2 \times (d^+)^2 \rightarrow d^+$ is monotonic. Suppose that $< z_1, z_2, z_3 > \sqsubseteq < w_1, w_2, w_3 >$. Two cases are possible:

1. $w_1 = z_1 =$ true. In such a case we have:

$$\textbf{if-then-else}(z_1, z_2, z_3) = z_2 \sqsubseteq w_2 = \textbf{if-then-else}(w_1, w_2, w_3)$$

2. $w_1 = z_1 =$ false. In such a case we have:

$$\textbf{if-then-else}(z_1, z_2, z_3) = z_3 \sqsubseteq w_3 = \textbf{if-then-else}(w_1, w_2, w_3)$$

It can easily be verified that **case-endcase** is built by the composition of **if-then-else**, id, and Ω, where $id(z) = z$, for each z, and $\Omega(\overline{z}) = \perp$, for each $\overline{z}$. The monotonicity of **case-endcase** follows from lemma 3 and the monotonicity of **if-then-else**, id, and Ω. □

Lemma 10 *Let E be a system of recurrence equations over a signature $\mathcal{SIG}$. Then for each $\mathcal{SIG}$-algebra A and stream interpretation $\mathcal{I}$ over A, the basic functionals in $[\![E]\!]_{A,\mathcal{I}}$ are continuous.*

Proof: The proof follows from lemmas 9 and 7. □

Theorem 11 *Let E be a system of recurrence equations over a signature $\mathcal{SIG}$. Then for each $\mathcal{SIG}$-algebra A and stream interpretation $\mathcal{I}$ over A, the tuple of basic functionals in $[\![E]\!]_{A,\mathcal{I}}$ has a simultaneous least fixed point.*

Proof: The proof follows from lemmas 10 and 8. □

Now, given a system of recurrence equations E over a signature $\mathcal{SIG}$, a $\mathcal{SIG}$-algebra A, and a stream interpretation $\mathcal{I}$ over A, we agree that the tuple of functions defined by E, under A and $\mathcal{I}$, is equal to the tuple of functions defined by the interpretation $[\![E]\!]_{A,\mathcal{I}}$ of E, under A and $\mathcal{I}$ (which by definition is equal to the the simultaneous least fixed point $fp =< fp_1, \ldots, fp_n >$ of $[\![E]\!]_{A,\mathcal{I}}$). By theorem 11 it follows that fp exists.

Let E_1 and E_2 be systems of recurrence equations over a signature $\mathcal{SIG}$ of the forms:

$$\begin{aligned} f_{11}(\overline{x}) &= \mathcal{F}_{11}(\overline{x}) \\ &\vdots \\ f_{1n_1}(\overline{x}) &= \mathcal{F}_{1n_1}(\overline{x}) \end{aligned}$$

and:

$$\begin{aligned} f_{21}(\overline{x}) &= \mathcal{F}_{21}(\overline{x}) \\ &\vdots \\ f_{2n_2}(\overline{x}) &= \mathcal{F}_{2n_2}(\overline{x}) \end{aligned}$$

respectively. Given a $\mathcal{SIG}$-algebra A and a stream interpretation $\mathcal{I}$ over A, we say that:

$$f_{1i}(\overline{\alpha}_1) = f_{2j}(\overline{\alpha}_2)$$

for some $1 \leq i \leq n_1$ and $1 \leq j \leq n_2$ if:

$$fp_{1i}(\overline{\alpha}_1) = fp_{2j}(\overline{\alpha}_2)$$

where $< fp_{11}, \ldots, fp_{1n_1} >$ and $< fp_{21}, \ldots, fp_{2n_2} >$ are the functions defined by $[\![E_1]\!]_{A,\mathcal{I}}$ and $[\![E_2]\!]_{A,\mathcal{I}}$ respectively.

We say that $f_{1i}(\overline{\alpha}_1) = f_{2j}(\overline{\alpha}_2)$ over an algebra A, written:

$$A \models f_{i1}(\overline{\alpha}_1) = f_{2j}(\overline{\alpha}_2) \tag{4.1}$$

iff:

$$f_{1i}(\overline{\alpha}_1) = f_{2j}(\overline{\alpha}_2)$$

for each stream interpretation over A.

We say that $f_{1i}(\overline{\alpha}_1) = f_{2j}(\overline{\alpha}_2)$ over an algebraic specification $\mathcal{SPEC}$, written:

$$\mathcal{SPEC} \models f_{1i}(\overline{\alpha}) = f_{2j}(\overline{\alpha})$$

iff (4.1) is valid for each $\mathcal{SPEC}$-algebra A.

Notice that the validity of :

$$A \models f_{1i}(\overline{x}_1) = f_{2j}(\overline{x}_2)$$

can be interpreted as a predicate $P(\overline{x}_1, \overline{x}_2)$ over the integers where for each value $\overline{\alpha}_1$ and $\overline{\alpha}_2$ of $\overline{x}_1$ and $\overline{x}_2$, $P(\overline{\alpha}_1, \overline{\alpha}_2)$ is true iff (4.1) is valid. The same applies to:

$$\mathcal{SPEC} \models f_{1i}(\overline{x}_1) = f_{2j}(\overline{x}_2)$$

4.4 Formal Verification

In this section we define formally what we mean by the verification of a systolic circuit.

As mentioned earlier, a specification of a system is a statement of how we expect the output of the system to be related to the input. The verification problem is to check whether the system implementation fulfills the requirements stated by the specification. In order to give a formal definition of systolic circuit verification we need the following proposition.

Proposition 12 *For each systolic circuit, the value of the signals of the circuit can be described as a system of recurrence equations, where the signature of the equation system is the same as the signature of the circuit and the function variables of the equation system are the signals of the circuit.*

Proof: The proof follows easily form (2.2), (2.3), (2.4), (2.5), (2.6), (2.14), (2.15), and (2.16). □

By the *verification* of a systolic circuit, with a signature $\mathcal{SIG}$, over a $\mathcal{SIG}$-algebra A, we mean that we check whether or not the specification formula $spec(\overline{x})$ of the circuit is valid over A; in symbols:

$$A \models spec(\overline{x})$$

We know that the specification formula of a circuit is of the form:

$$spec(\overline{x}) = (p_1(\overline{x}) \longrightarrow (s_1(\overline{x}) = f_1(\overline{x}))) \wedge \cdots \wedge (p_m(\overline{x}) \longrightarrow (s_m(\overline{x}) = f_m(\overline{x})))$$

where $p_i(\overline{x})$ is a predicate over the integers, s_i is a signal in the circuit, and $f_i(\overline{x})$ is a function variable in a system of recurrence equations. Thus the verification amounts to checking the validity of:

$$A \models (p_1(\overline{x}) \longrightarrow (s_1(\overline{x}) = f_1(\overline{x}))) \wedge \cdots \wedge (p_m(\overline{x}) \longrightarrow (s_m(\overline{x}) = f_m(\overline{x}))) \quad (4.2)$$

The validity of the formula in (4.2) is interpreted as follows: for each value $\overline{\alpha}$ of $\overline{x}$ if $p(\overline{\alpha})$ is true then:

$$A \models s_i(\overline{\alpha}) = f_i(\overline{\alpha}) \quad (4.3)$$

is valid, for $1 \leq i \leq m$. From proposition 12 and the definition of specification formulas we know that $s_i(\overline{x})$ and $f_i(\overline{x})$ are function variables in systems of recurrence equations. Thus from section 4.3 we know the formal meaning of the validity of (4.3).

By the *verification* of a circuit with a specification formula $spec(\overline{x})$ and a signature $\mathcal{SIG}$ over an algebraic specification $\mathcal{SPEC} = (\mathcal{SIG}, \mathcal{E})$, written:

$$\mathcal{SPEC} \models spec(\overline{x})$$

we mean that we check whether or not:

$$A \models spec(\overline{x})$$

is valid for each $\mathcal{SPEC}$-algebra A.

In chapter 5 we will study verification of systolic circuits over rings. This means that the operation symbols of the circuit will be $\{+, \cdot, -, 0\}$, and the verification will be performed over the algebraic specification of rings (see appendix A).

In chapter 7 we will study circuit verification over boolean algebras. Thus the operation symbols will be $\{\vee, \wedge, \neg, false, true\}$, and the verification is performed over the algebraic specification of boolean algebras (see appendix B).

In chapter 6 we will consider circuit verification over algebraic specifications with empty equation sets. That is, given a systolic circuit with signature $\mathcal{SIG}$ and specification formula $spec(\overline{x})$ we consider the validity of:

$$\mathcal{SPEC} \models spec(\overline{x}) \tag{4.4}$$

where $\mathcal{SPEC} = (\mathcal{SIG}, \emptyset)$. Clearly if (4.4) holds then:

$$A \models spec(\overline{x})$$

holds for each $\mathcal{SIG}$-algebra A.

4.5 Automatic Verification

By the automatic verification of a class of circuits, each with a signature $\mathcal{SIG}$, over a $\mathcal{SIG}$-algebra A, we mean that we have a decision method which, given the specification formula $spec(\overline{x})$ of a circuit in the class, the method checks automatically whether:

$$A \models spec(\overline{x}) \tag{4.5}$$

By automatic verification of a class of circuits over an algebraic specification we mean that we have a decision method which checks automatically the validity of (4.5) for all $\mathcal{SPEC}$-algebras A. In chapters 5, 6, and 7 we give decision methods to perform automatic verification of certain classes of systolic circuits over the algebraic specification of rings, algebraic specifications with empty equation sets, and the algebraic specification of boolean algebras respectively. We derive our classes of circuits, on which we perform automatic verification, by imposing suitable restrictions on the general model of circuits we presented in chapter 2. The restrictions are made by demanding that the circuit topologies, interconnections, cell computations, and specification formulas should be of certain restricted forms which are special cases of the general model. The restrictions are made in a such a way that we can define decision methods while still maintaining nontrivial classes of circuits. The reason for considering restricted classes of circuits is that the general model often leads to undecidable verification problems.

Chapter 5

Class of Rings

In this chapter we study verification of systolic circuits which operate over commutative rings. We will give a decision method for automatic verification of a class of these circuits. Such circuits are found in the field of digital signal processing. Examples of circuits which can be automatically verified by the methods of this chapter include the convolution circuit in [Ull84] (described in section 3.1), the matrix multiplication circuits in [MC80] and [Ull84], the systolic realization of linear phase FIR digital filters in [Kwa87], the systolic arrays for Viterbi Processing in [PG88], and the systolic array for 2-D spatial filtering in [AS88]. The cell computations in the circuits of this class are defined by the ring operators $+$, $\cdot$, $-$, and 0. We will investigate the problem of deciding whether a circuit implementation is correct with respect to a specification for the class of all commutative rings. If a circuit is verified in this manner then the circuit is correctly implemented for any arbitrary interpretation of the ring operators in a commutative ring. We will also study the verification of circuits which operate on some particularly interesting rings, such as the ring of integers, and the ring of natural numbers modulo m, for some fixed natural number m. The latter is particularly interesting in systolic circuit implementations since modular arithmetics allow bounding the sizes of the cell registers in the circuit. We will illustrate the ideas of the chapter by sketching how the verification method can be applied to the convolution circuit introduced in section 3.1.

In section 5.1 we give some preliminaries on rings. In section 5.2 we define two classes of systolic circuits, the class of *linear systolic circuits*, and the class of *tail-recursive ring systolic circuits*. These two classes of circuits are defined by imposing restrictions on topologies, interconnections, cell computations, and specification formulas of the circuits of the general model in chapter 2. The class of linear circuits will be used even in chapters 6 and 7 to define new classes of circuits. The class of tail-recursive ring systolic circuits is a subclass of linear systolic circuits. In this chapter we construct a method for automatic verification of tail-recursive ring systolic circuits. In section 5.3 we study what the general definition of semantics amounts to when restricting ourselves to the class of commutative rings. In section 5.4 we give an overview of a decision method for automatic verification of tail-recursive ring circuits. In section 5.5 we show that the values of signals in the implementation and specification of a tail-recursive ring systolic circuit can be described as a class of guarded expressions over rings. In section 5.6 we show that the problem of deciding equality between two guarded ring expressions can be reduced to the integer linear programming problem which is decidable [BT76], and hence the verification

problem is decidable.

5.1 Preliminaries

We will work with an arbitrary commutative ring[1] $R =< R, +, \cdot, -, 0 >$ (see appendix A). As in chapter 2, let $I =< I, +, -, \cdot, \leq, 0, 1 >$ be the ordered ring of integers, and let $Q =< Q, +, -, \cdot, 0, 1 >$ be the ring of rationals. We will use x and y (possibly with subscripts) to denote variables which range over I. We will use $\alpha, \beta, \gamma, \delta$ to range over I, and ρ to range over Q.

A linear QI-polynomial is of the form:

$$\rho_1 x_1 + \ldots + \rho_n x_n + \rho_{n+1}$$

We will use ℓ and q to range over linear QI-polynomials.

A *linear inequality* is of the form[2]:

$$\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta \leq 0$$

A *linear equality* is of the form:

$$\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta = 0$$

A *linear modulo predicate* is of the form

$$(\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta) \; mod \; \gamma = 0$$

where $0 \leq \beta < \gamma$. Here γ is called the *modulus* of the linear modulo predicate.

By a *linear predicate* we mean a linear inequality, a linear equality or a linear modulo predicate. We use p to range over conjunctions of linear predicates.

When working with a ring R, the set $\mathcal{A}$ of stream variables is such that each $a \in \mathcal{A}$ has a sort R.

We consider a class of expressions which expand the class of polynomials over R. The class of *ring expressions* is defined by the following:

- 0 is a ring expression.
- If $a \in \mathcal{A}$ is a stream variable with arity n, and $q_1(\overline{x}), \ldots, q_n(\overline{x})$ are linear QI-polynomials then $a(q_1(\overline{x}), \ldots, q_n(\overline{x}))$ (called a *linear stream expression*) is a ring expression.
- If $r_1(\overline{x})$ and $r_2(\overline{x})$ are ring expressions, then $r_1(\overline{x}) + r_2(\overline{x})$, $-r_1(\overline{x})$, and $r_1(\overline{x}) \cdot r_2(\overline{x})$ are ring expressions.
- If $r(\overline{x}, i)$ is a ring expression, and $\ell(\overline{x})$ is a linear QI-polynomial then $\sum_{i=0}^{\ell(\overline{x})} r(\overline{x}, i)$ is a ring expression.

[1] In the rest of the chapter we say only *ring* instead of *commutative ring*.

[2] Observe that for each inequality of the form $\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta \leq \alpha'_1 x_1 + \cdots + \alpha'_n x_n + \beta'$, or $\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta \geq 0$, or $\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta > 0$, or $\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta < 0$, there is an equivalent inequality of the form we have defined above.

We will denote ring expressions by r.

The class of *ring polynomials* over a ring R is defined by the following:

- 0 is a ring polynomial over R.
- Each variable over R is a ring polynomial over R.
- If r_1 and r_2 are ring polynomials over R, then $r_1 + r_2$, $-r_1$, and $r_1 \cdot r_2$ are ring polynomials over R.

A *linear guarded ring expression* is of the form:

$$\begin{array}{ll} \textbf{case} & \\ \quad p_1(\overline{x}) & \Longrightarrow \; r_1(\overline{x}) \\ \quad \vdots & \\ \quad p_n(\overline{x}) & \Longrightarrow \; r_n(\overline{x}) \\ \textbf{endcase} & \end{array}$$

where $r_i(\overline{x})$ is a ring expression, and $p_i(\overline{x})$ is a conjunction of linear predicates. In addition, we assume that $p_i(\overline{\alpha}) \wedge p_j(\overline{\alpha})$ is false for each $\overline{\alpha}$ if $j \neq i$, and that $r_i(\overline{x})$ is well-defined under $p_i(\overline{x})$ (Informally we say that a ring expression $r(\overline{x})$ is well-defined under a predicate $p(\overline{x})$ iff for each value $\overline{\alpha}$ of $\overline{x}$ if $p(\overline{\alpha})$ is true then the upper indexes of the $\sum$ operators in $r(\overline{\alpha})$ evaluate to nonnegative integers, and the arguments of the stream variables evaluate to integers. For a formal definition of well-definedness see section 5.3). Each $p_i(\overline{x})$ is called a *guard* of the expression, and each $r_i(\overline{x})$ is called a *result* of the expression, while every $p_i(\overline{x}) \Rightarrow r_i(\overline{x})$ is called a *case* of the expression. It can easily be shown that each linear guarded ring expression can be defined by a system of recurrence equations over the signature of rings. We use $e, e_1, e_2, e_3, \ldots$ to range over linear guarded ring expressions.

5.2 Special Classes of Systolic Circuits

In this section we will introduce classes of circuits which are special cases of the general model described in chapter 2. These special classes of circuits are defined by imposing restrictions on how the circuit topologies, interconnections, cell computations, and specifications may look like. The automatic verification methods will be applied to the restricted classes of circuits. In section 5.2.1 we will define the class of *linear systolic circuits*. The notion of linear systolic circuits will also be used in chapters 6 and 7. In section 5.2.2 we will describe *tail-recursive ring systolic circuits*. In the rest of this chapter we will give a method for automatic verification of tail-recursive ring circuits.

5.2.1 Linear Systolic Circuits

We will use linear systolic circuits to define the class of circuits on which we perform automatic verification in this chapter and in chapters 6 and 7.

Topology The topology of a linear systolic circuit is of the form:

$$top(\overline{x},\overline{\ell}) \;=\; top_1(\overline{x},\overline{\ell}) \wedge \ldots \wedge top_m(\overline{x},\overline{\ell})$$

where each $top_i(\overline{x},\overline{\ell})$ is a linear inequality. Observe that $top(\overline{x},\overline{\ell})$ defines a polytope.

Interconnections In a linear systolic circuit, the connection function Δ_i corresponding to an input in_i is a tuple of integers, i.e. :

$$\Delta_i \;=\; \overline{\delta}_i \tag{5.1}$$

where $\overline{\delta}_i$ is an n-tuple of integers in an n-dimensional circuit[3]. We call $\overline{\delta}_i$ the *connection vector* of in_i. Also an input expression is of the form:

$$\begin{array}{l}\textbf{case}\\ \quad p_1(\overline{x},\overline{\ell},t) \;\Longrightarrow\; it_1(\overline{x},\overline{\ell},t)\\ \qquad\qquad\vdots\\ \quad p_n(\overline{x},\overline{\ell},t) \;\Longrightarrow\; it_n(\overline{x},\overline{\ell},t)\\ \textbf{endcase}\end{array} \tag{5.2}$$

where $p_i(\overline{x},\overline{\ell},t)$ is a conjunction of linear predicates.

Cell Computations In linear systolic circuits, the cell computation associated with each output or local variable is of the form:

$$\begin{array}{l}\textbf{case}\\ \quad p_1(\overline{x},\overline{\ell},t) \;\Longrightarrow\; ct_1(\overline{x},\overline{\ell},t)\\ \qquad\qquad\vdots\\ \quad p_m(\overline{x},\overline{\ell},t) \;\Longrightarrow\; ct_m(\overline{x},\overline{\ell},t)\\ \textbf{endcase}\end{array} \tag{5.3}$$

where $p_i(\overline{x},\overline{\ell},t)$ is a conjunction of linear predicates. An initial expression is of the same form as an input expression (see (5.2)).

Specification The specification formula $spec(\overline{x},\overline{\ell},t)$ of a linear circuit is of the form:

$$\begin{array}{l}spec(\overline{x},\overline{\ell},t) \;=\\ (p_1(\overline{x},\overline{\ell},t) \longrightarrow (s_1(\overline{x},\overline{\ell},t) \;=\; f_1(\overline{x},\overline{\ell},t))) \wedge \cdots \wedge (p_m(\overline{x},\overline{\ell},t) \longrightarrow (s_m(\overline{x},\overline{\ell},t) \;=\; f_m(\overline{x},\overline{\ell},t)))\end{array}$$

where $p_i(\overline{x})$ is a conjunction of linear predicates.

In sections 5.2.2, 6.2, and 7.2 we define new classes of circuits which are special cases of linear circuits, by considering special forms of cell computations and specification formulas.

[3]Notice that although the interconnections of the summation circuit (section 2.2) do not follow the form given here, the circuit can be considered as a two-dimensional linear circuit, where the topology of the circuit is defined by $sumtop(x,y,\ell) = (1 \leq x \leq \ell) \wedge (y = 1)$, and the interconnection vectors are $\overline{\delta}_1 =< -1, 0 >$ and $\overline{\delta}_2 =< 0, -1 >$.

5.2.2 Tail-recursive Ring Circuits

A *tail-recursive ring systolic circuit* is a linear systolic circuit, with the signature of rings, where certain restrictions are imposed on the cell computations and the specification formula of the circuit.

Cell Computations in Tail-recursive Ring Systolic Circuits A cell computation is of the form:

$$
\begin{array}{l}
s(\overline{x},\overline{\ell},t) \;=\\
\textbf{case}\\
\quad \begin{array}{rcl}
p_1(\overline{x},\overline{\ell},t) & \Longrightarrow & Q_1(s_1(\overline{x},\overline{\ell},t-\tau),\ldots,s_k(\overline{x},\overline{\ell},t-\tau))\\
\vdots & & \\
p_m(\overline{x},\overline{\ell},t) & \Longrightarrow & Q_m(s_1(\overline{x},\overline{\ell},t-\tau),\ldots,s_k(\overline{x},\overline{\ell},t-\tau))\\
p_{m+1}(\overline{x},\overline{\ell},t) & \Longrightarrow & s'(\overline{x},\overline{\ell},t-\tau)+Q_{m+1}(s_1(\overline{x},\overline{\ell},t-\tau),\ldots,s_k(\overline{x},\overline{\ell},t-\tau))\\
\vdots & & \\
p_n(\overline{x},\overline{\ell},t) & \Longrightarrow & s'(\overline{x},\overline{\ell},t-\tau)+Q_n(s_1(\overline{x},\overline{\ell},t-\tau),\ldots,s_k(\overline{x},\overline{\ell},t-\tau))
\end{array}\\
\textbf{endcase}
\end{array}
\qquad (5.4)
$$

where $Q_i(\overline{z})$ is a ring polynomial over R, and s' is defined by the following: if s is a local variable then s' is the same as s, while if s is an output out_i then s' is the corresponding input in_i.

Ther *dependency relation* $\prec_D$ ($s_2 \prec_D s_1$ is read s_1 is dependent on s_2) among the signals of a tail-recursive ring circuit is defined as the smallest relation containing the following elements:

- Let s be any output or local variable, and let the computation of s be of the form of equation (5.4), then $s_i \prec_D s$, for $1 \leq i \leq k$.
- Let s be any signal and out_i any output, then if $s \prec_D out_i$ then $s \prec_D in_i$.
- let s be any signal and in_i any input, then if $in_i \prec_D s$ then $out_i \prec_D s$.

In the class of tail-recursive ring circuits, the dependency relation is acyclic, i.e. for each signal s, we have $s \not\prec_D^+ s$, where s is the transitive closure of $\prec_D$. This implies that $\prec_D$ is a well-founded relation. The dependency relation of the convolution circuit in section 3.1 is given by the set $\{< loc, in_2 >, < loc, out_2 >, < out_3, loc >, < in_3, loc >, < out_3, in_1 >, < out_3, out_1 >, < in_3, in_1 >, < in_3, out_1 >\}$. Observe that the relation is acyclic.

Specification Formulas of Tail-recursive Ring Systolic Circuits The specification formula of a tail-recursive ring circuit is of the form:

$$
\begin{array}{l}
spec(\overline{x},\overline{\ell},t) \;=\\
(p_1(\overline{x},\overline{\ell},t) \longrightarrow (s_1(\overline{x},\overline{\ell},t) = r_1(\overline{x},\overline{\ell},t))) \wedge \cdots \wedge (p_m(\overline{x},\overline{\ell},t) \longrightarrow (s_m(\overline{x},\overline{\ell},t) = r_m(\overline{x},\overline{\ell},t)))
\end{array}
$$

where $r_i(\overline{x},\overline{\ell},t)$ is a ring expression which is well-defined under $p_i(\overline{x},\overline{\ell},t)$.

The reason for the name *tail-recursive* is that the computations of a tail-recursive ring circuit can be described (section 5.5.2) by a class of functions which we will introduce in section 5.5.1, and which we call *tail-recursive ring functions.*

It can easily be checked that the convolution circuit described in section 3.1 is a tail-recursive ring circuit.

5.3 Semantics

Let R be a ring. In a similar manner to the general case in section 4.3, given a stream interpretation $\mathcal{I}$ over R, we give a formal semantics for our ring expressions and linear guarded ring expressions. A stream interpretation $\mathcal{I}$ over R maps each stream variable into a function from tuples of integers into R. Thus if $\mathcal{I}$ is a stream interpretation then, for each $a \in \mathcal{A}$ of arity n, $\mathcal{I}(a)$ is a function of type $I^n \to R$.

The semantics definition in the case of rings is a special case of defining semantics for arbitrary algebras as described in section 4.3. Nevertheless we think it is interesting to investigate what this amounts to in the special case of rings, and when dealing with the special class of tail-recursive ring circuits.

Let $\perp$ be an element such that $\perp \notin R$. By $[\![r(\overline{x})]\!]_{R,\mathcal{I}}$ and $[\![e(\overline{x})]\!]_{R,\mathcal{I}}$ we mean the *interpretation* of the ring expression $r(\overline{x})$ and the linear guarded ring expression $e(\overline{x})$ respectively under the ring R and the stream interpretation $\mathcal{I}$ over R. We have:

$$[\![0]\!]_{R,\mathcal{I}} = 0$$

$$[\![a(\overline{q}(\overline{x}))]\!]_{R,\mathcal{I}} = \begin{cases} a^{\mathcal{I}}(\overline{q}(\overline{x})) & \text{if } \overline{q}(\overline{x}) \text{ is a tuple of integers} \\ \perp & \text{otherwise} \end{cases}$$

$$[\![r_1(\overline{x}) + r_2(\overline{x})]\!]_{R,\mathcal{I}} = \begin{cases} [\![r_1(\overline{x})]\!]_{R,\mathcal{I}} + [\![r_2(\overline{x})]\!]_{R,\mathcal{I}} & \text{if } [\![r_i(\overline{x})]\!]_{R,\mathcal{I}} \neq \perp \text{ for } i = 1,2 \\ \perp & \text{otherwise} \end{cases}$$

$$[\![-r(\overline{x})]\!]_{R,\mathcal{I}} = \begin{cases} -[\![r(\overline{x})]\!]_{R,\mathcal{I}} & \text{if } [\![r(\overline{x})]\!]_{R,\mathcal{I}} \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

$$[\![r_1(\overline{x}) \cdot r_2(\overline{x})]\!]_{R,\mathcal{I}} = \begin{cases} [\![r_1(\overline{x})]\!]_{R,\mathcal{I}} \cdot [\![r_2(\overline{x})]\!]_{R,\mathcal{I}} & \text{if } [\![r_i(\overline{x})]\!]_{R,\mathcal{I}} \neq \perp \text{ for } i = 1,2 \\ \perp & \text{otherwise} \end{cases}$$

$$\left[\!\!\left[\sum_{i=0}^{\ell(\overline{x})} r(\overline{x}, i)\right]\!\!\right]_{R,\mathcal{I}} = \begin{cases} \sum_{i=0}^{\ell(\overline{x})} [\![r(\overline{x}, i)]\!]_{R,\mathcal{I}} & \text{if } \ell(\overline{x}) \text{ is a nonnegative integer} \\ \perp & \text{otherwise} \end{cases}$$

$$\left[\!\!\left[\begin{array}{l} \textbf{case} \\ \quad p_1(\overline{x}) \Rightarrow r_1(\overline{x}); \\ \quad\quad \vdots \\ \quad p_m(\overline{x}) \Rightarrow r_m(\overline{x}) \\ \textbf{endcase} \end{array}\right]\!\!\right]_{R,\mathcal{I}} = \begin{cases} [\![r_i(\overline{x})]\!]_{R,\mathcal{I}} & \text{if } p_i(\overline{x}) \text{ is true} \\ \perp & \text{if } p_i(\overline{x}) \text{ is false for } 1 \leq i \leq m \end{cases}$$

We say that a ring expression $r(\overline{x})$ is *well-defined* for a certain value $\overline{\alpha}$ if, for each stream interpretation $\mathcal{I}$ over R, $[\![r(\overline{\alpha})]\!]_{R,\mathcal{I}} \neq \perp$. Otherwise we say that $r(\overline{\alpha})$ is *undefined* for $\overline{\alpha}$. Observe that the well-definedness property does not depend on the particular stream interpretation $\mathcal{I}$, nor on the particular ring R. We say that a ring expression $r(\overline{x})$ is well-defined under a conjunction of linear predicates $p(\overline{x})$ iff for each value $\overline{\alpha}$ of $\overline{x}$, if

$p(\overline{\alpha})$ is true then $r(\overline{\alpha})$ is well-defined.

We say that two ring expressions $r_1(\overline{x})$ and $r_2(\overline{x})$ are *equal*, over R, for a value $\overline{\alpha}$ of $\overline{x}$ (denoted $r_1(\overline{\alpha}) = r_2(\overline{\alpha})$) iff for each stream interpretation $\mathcal{I}$ over R, $[\![r_1(\overline{\alpha})]\!]_{R,\mathcal{I}} = [\![r_2(\overline{\alpha})]\!]_{R,\mathcal{I}}$.

Observe that a linear guarded ring expression is well-defined for a value $\overline{\alpha}$ of $\overline{x}$ iff one of its guards is true for $\overline{\alpha}$.

5.4 Overview of The Verification Decision Method

We will give a sketch of an automatic decision method for tail-recursive ring circuits. In section 5.2.2 we mentioned that a circuit specification was of the form:

$$spec(\overline{x}) = \\ (p_1(\overline{x}) \longrightarrow (s_1(\overline{x}) = r_1(\overline{x}))) \wedge \cdots \wedge (p_m(\overline{x}) \longrightarrow (s_m(\overline{x}) = r_m(\overline{x}))) \tag{5.5}$$

where $p_1(\overline{x}), \ldots, p_m(\overline{x})$ are conjunctions of linear predicates, $s_1, \ldots, s_m$ are signals in the circuit, and $r_1(\overline{x}), \ldots, r_m(\overline{x})$ are ring expressions such that $r_i(\overline{x})$ is well-defined under $p_i(\overline{x})$.

In section 5.3 we defined formally the notion of equality for linear guarded ring expressions with respect to any ring R.

Let $\mathcal{K}$ be a class of rings. By the *circuit verification* for the class of rings $\mathcal{K}$, we mean that we check whether the specification formula in (5.5) is valid when interpreted in every ring $R \in \mathcal{K}$; in symbols:

$$\mathcal{K} \models spec(\overline{x})$$

Notice that $spec(\overline{x})$ contains stream variables (which occur in $r_1, \ldots, r_m$) and integer variables ($\overline{x}$, $\overline{\ell}$, and t). Thus to interpret $spec(\overline{x})$, the stream variables are interpreted in the rings of $\mathcal{K}$, while the integer variables are interpreted in the standard model of integers.

The verification process is carried out in the following two steps, each of which is carried out automatically:

1. In section 5.5 we show that, for each tail-recursive ring systolic circuit, the value of each signal in the circuit can be described by a class of functions which we will introduce in section 5.5.1, and which we call *tail-recursive ring functions*. Furthermore we will show that for each tail-recursive ring function, there is a linear guarded ring expression, which is equal to it over each ring R. This means that, considering the specification formula in (5.5), there are linear guarded ring expressions $e_1(\overline{x}), \ldots, e_m(\overline{x})$ such that:

$$s_1(\overline{x}) = e_1(\overline{x}) \quad , \quad \ldots \quad , \quad s_m(\overline{x}) = e_m(\overline{x})$$

over each ring R, so that the specification formula can be rewritten as:

$$spec(\overline{x}) = \\ (p_1(\overline{x}) \longrightarrow (e_1(\overline{x}) = r_1(\overline{x}))) \wedge \cdots \wedge (p_m(\overline{x}) \longrightarrow (e_m(\overline{x}) = r_m(\overline{x}))) \tag{5.6}$$

which is equivalent over $\mathcal{K}$.

2. In section 5.6 we will study the decidability of the validity of formulas of the general form:

$$p(\overline{x}) \longrightarrow (e(\overline{x}) = r(\overline{x})) \tag{5.7}$$

where $p(\overline{x})$ is a conjunction of linear predicates, $e(\overline{x})$ is a linear guarded ring expression, and $r(\overline{x})$ is a ring expression which is well-defined under $p(\overline{x})$. From the formula in (5.6) we conclude that if the validity, over $\mathcal{K}$, of the formula in (5.7) is decidable, then the validity of the specification formula is decidable.

We know that $e(\overline{x})$ in (5.7) is of the form:

$$\textbf{case} \;\; p'_1(\overline{x}) \Rightarrow r'_1(\overline{x}) \; ; \; \ldots \; ; \; p'_n(\overline{x}) \Rightarrow r'_n(\overline{x}) \;\; \textbf{endcase}$$

Thus , for $\mathcal{K}$, the validity of the formula in (5.7) is equivalent to the validity of the two formulas:

$$p(\overline{x}) \longrightarrow p'_1(\overline{x}) \vee \cdots \vee p'_n(\overline{x}) \tag{5.8}$$

over I, and:

$$p(\overline{x}) \wedge p'_i(\overline{x}) \longrightarrow (r(\overline{x}) = r'_i(\overline{x})) \tag{5.9}$$

over $\mathcal{K}$, for $1 \leq i \leq n$.

The validity, over I, of the formula in (5.8) can be shown (lemma 25) to be reducible to the integer linear programming problem which is decidable [BT76] (integer linear programming is described in section 5.6.4).

To decide the validity, over $\mathcal{K}$, of the formula in (5.9), we need to decide the validity, over $\mathcal{K}$, of formulas of the form:

$$p(\overline{x}) \longrightarrow (r(\overline{x}) = 0) \tag{5.10}$$

where $p(\overline{x})$ is a conjunction of linear predicates and $r(\overline{x})$ is a ring expression well-defined under $p(\overline{x})$. We will consider a class of formulas which we call *bilinear formulas* (see section 5.6.2 for the definition of bilinearity). We define a measure of complexity for formulas of the form of (5.10), and then show (in lemma 22) that if the formula in (5.10) is bilinear then the formula can be rewritten into the conjunction of a set of "less complex" formulas, which is equivalent over each ring R. The rewriting can be repeated recursively (lemma 24) until we obtain a conjunction of "simple" formulas. A "simple formula" is of the form:

$$p'(\overline{x}) \longrightarrow (r = 0) \tag{5.11}$$

where $p'(\overline{x})$ is a conjunction of linear predicates, and r is a ring expression which does not contain any free integer variables. Obviously the validity, over $\mathcal{K}$, of the simple formula in (5.11) is equivalent to the unsatisfiability, over I, of $p'(\overline{x})$, or the validity, over $\mathcal{K}$, of $r = 0$. The unsatisfiability of $p'(\overline{x})$ can be easily shown to be reducible to the integer linear programming problem which is decidable [BT76]. Also the validity, over $\mathcal{K}$, of $r = 0$ can be shown (lemma 23) to be reducible to the zero-equivalence of a ring polynomial over $\mathcal{K}$ (see section 5.1 for the definition of ring polynomials).

Thus deciding the validity of the specification formula for the class of rings $\mathcal{K}$ is reduced to deciding the zero-equivalence of polynomials over $\mathcal{K}$; i.e.:

$$\mathcal{K} \models (Q = 0) \tag{5.12}$$

where Q is a ring polynomial.

The problem in (5.12) is decidable if $\mathcal{K}$ is the class of all rings. It is also decidable for some interesting rings, for example, if $\mathcal{K}$ is the ring of integers, the ring of reals, or the ring of natural numbers modulo m (for some fixed natural number m).

5.5 Ring Circuits as Guarded Ring Expressions

In this section we carry out the first step of the automatic verification of tail-recursive ring circuits. We show that, for a tail-recursive ring systolic circuit, the values of outputs, inputs, and local variables of the different cells at different time instants can be described as linear guarded ring expressions. This is done in two steps: i) we introduce a class of functions which we call *tail-recursive ring functions*, and show that for each tail-recursive ring function, there is a linear guarded ring expression which is equal to it over each ring R. ii) we show that the values of cell outputs, inputs, and local variables can be described as tail-recursive ring functions, and hence as linear guarded ring expressions. We apply the results obtained in the section to describe the signals of the convolution circuit (described in section 3.1) as linear guarded ring expressions.

5.5.1 Tail-Recursive Ring Functions

In this section we introduce *tail-recursive ring functions*, and show that for each tail-recursive ring function there is a linear guarded ring expression which is equal to it over each ring R.

Definitions

A *tail-recursive ring function* f over a tuple $\overline{\delta}$ of integers, where at least one element δ_i of $\overline{\delta}$ is not equal to 0, is of the following form:

$$\begin{aligned} f(\overline{x}) \;=\; & \textbf{case} \\ & \quad p_1(\overline{x}) \Rightarrow r_1(\overline{x}) \;;\; \ldots \;;\; p_m(\overline{x}) \Rightarrow r_m(\overline{x}) \;; \\ & \quad p_{m+1}(\overline{x}) \Rightarrow f(\overline{x} + \overline{\delta}) + r_{m+1}(\overline{x}) \;;\; \ldots \;;\; p_n(\overline{x}) \Rightarrow f(\overline{x} + \overline{\delta}) + r_n(\overline{x}) \\ & \textbf{endcase} \end{aligned} \tag{5.13}$$

where each $r_i(\overline{x})$ is well-defined under $p_i(\overline{x})$, and $p_i(\overline{\alpha}) \wedge p_j(\overline{\alpha})$ is false for each $\overline{\alpha}$ if $j \neq i$. Note that each tail-recursive ring function is defined by a recurrence equation over the signature of rings.

Given a ring R and a stream interpretation $\mathcal{I}$ over R, the function defined by $f(\overline{x})$ in

(5.13) is given by the least fixed point of the following:

$$[\![f(\overline{x})]\!]_{R,\mathcal{I}} = \begin{cases} [\![r_i(\overline{x})]\!]_{R,\mathcal{I}} & \text{if } p_i(\overline{x}) \text{ is true and } 1 \le i \le m \\ \left[\!\left[f(\overline{x}+\overline{\delta})\right]\!\right]_{R,\mathcal{I}} + [\![r_i(\overline{x})]\!]_{R,\mathcal{I}} & \text{if } p_i(\overline{x}) \text{ is true and } m+1 \le i \le n \\ \perp & \text{if } p_i(\overline{x}) \text{ is false for } 1 \le i \le n \end{cases}$$

Note that if the recursive calls of $[\![f(\overline{x})]\!]_{R,\mathcal{I}}$ continue infinitely (i.e. for each $j \ge 0$ there is a p_{i_j} such that $p_{i_j}(\overline{x} + \overline{\delta} \cdot j)$ is true, where $m+1 \le i_j \le n$), then $[\![f(\overline{x})]\!]_{R,\mathcal{I}} = \perp$. For each value $\overline{\alpha}$ of $\overline{x}$, we say that a tail-recursive ring function $f(\overline{x})$ is *equal* to a linear guarded ring expression $e(\overline{x})$, over a ring R, iff for each stream interpretation $\mathcal{I}$ over R, $[\![f(\overline{\alpha})]\!]_{R,\mathcal{I}} = [\![e(\overline{\alpha})]\!]_{R,\mathcal{I}}$. What the relation $f_1(\overline{\alpha}) = f_2(\overline{\alpha})$ means should be clear.

A tail-recursive ring function $f(\overline{x})$ is said to be *elementary* if it is of the form:

$$\begin{array}{l} f(\overline{x}) = \\ \textbf{case} \\ \qquad\qquad\qquad p_1(\overline{x}) \implies r_1(\overline{x}) \\ \qquad\qquad\qquad\qquad \vdots \\ \qquad\qquad\qquad p_m(\overline{x}) \implies r_m(\overline{x}) \\ \qquad p(\overline{x}) \wedge p_{m+1}(\overline{x}) \implies f(\overline{x}+\overline{\delta}) + r_{m+1}(\overline{x}) \\ \qquad\qquad\qquad\qquad \vdots \\ \qquad\qquad p(\overline{x}) \wedge p_n(\overline{x}) \implies f(\overline{x}+\overline{\delta}) + r_n(\overline{x}) \\ \textbf{endcase} \end{array} \tag{5.14}$$

where $p(\overline{x})$ is a conjunction of linear inequalities and equalities, $p_1(\overline{x}), \ldots, p_m(\overline{x})$ are conjunctions of linear predicates, and $p_{m+1}(\overline{x}), \ldots, p_n(\overline{x})$ are conjunctions of linear modulo predicates.

Elementary Tail-Recursive Functions as Linear Guarded Ring Expressions

We will show that for each elementary tail-recursive ring function there is a linear guarded ring expression which is equal to it over each ring R. This is achieved in lemma 14. For the proof of the lemma we need some auxiliary definitions and lemmas.

Two linear modulo predicates $p_1(\overline{x})$ and $p_2(\overline{x})$ are said to be *similar* if $p_1(\overline{x})$ is of the form $(\overline{\alpha_1}\,\overline{x} + \beta_1) \bmod \gamma_1 = 0$ and $p_2(\overline{x})$ is of the form $(\overline{\alpha_2}\,\overline{x} + \beta_2) \bmod \gamma_2 = 0$, where $\overline{\alpha_1} = \overline{\alpha_2}$ and $\gamma_1 = \gamma_2$. Two conjunctions of linear modulo predicates $p_1(\overline{x})$ and $p_2(\overline{x})$ are said to be *similar* if $p_1(\overline{x})$ is of the form $p_{11}(\overline{x}) \wedge \ldots \wedge p_{1m}(\overline{x})$ and $p_2(\overline{x})$ is of the form $p_{21}(\overline{x}) \wedge \ldots \wedge p_{2m}(\overline{x})$, and $p_{1i}(\overline{x})$ and $p_{2i}(\overline{x})$ are similar for $1 \le i \le m$.

We call the elementary tail-recursive function $f(\overline{x})$ in equation (5.14) *complete* if $p_{m+1}(\overline{x}), \ldots, p_n(\overline{x})$ all are similar.

Lemma 13 *For each elementary tail-recursive function f_1, there is a complete elementary tail-recusive function f_2 such that $f_1(\overline{x}) = f_2(\overline{x})$ over each ring R.*

Proof: The details of the proof are not included. The main idea behind the proof is the following: Let $p_1(\overline{x})$ and $p_2(\overline{x})$ be two linear modulo predicates, which are not similar.

Let $p_i(\overline{x})$ be of the form $(\overline{\alpha_i}\ \overline{x} + \beta_i)\ mod\ \gamma_i = 0$, for $i = 1, 2$. Then:

$$p_1(\overline{x}) \ = \ p_1^{(1)}(\overline{x}) \ \vee \ \cdots \ \vee \ p_{\gamma_2}^{(1)}(\overline{x})$$

where:

$$p_j^{(1)}(\overline{x}) \ = \ ((\overline{\alpha_1}\ \overline{x} + \beta_1)\ mod\ \gamma_1 = 0) \ \wedge \ ((\overline{\alpha_2}\ \overline{x} + \beta_2 + j - 1)\ mod\ \gamma_2 = 0)$$

and:

$$p_2(\overline{x}) \ = \ p_1^{(2)}(\overline{x}) \ \vee \ \cdots \ \vee \ p_{\gamma_1}^{(2)}(\overline{x})$$

where:

$$p_j^{(2)}(\overline{x}) \ = \ ((\overline{\alpha_1}\ \overline{x} + \beta_1 + j - 1)\ mod\ \gamma_1 = 0) \ \wedge \ ((\overline{\alpha_2}\ \overline{x} + \beta_2)\ mod\ \gamma_2 = 0)$$

Observe that $p_1^{(1)}(\overline{x}), \ldots, p_{\gamma_2}^{(1)}(\overline{x}), p_1^{(2)}(\overline{x}), \ldots, p_{\gamma_1}^{(2)}(\overline{x})$ all are similar.

An elementary tail-recursive function $f_1(\overline{x})$ can thus be rewritten into a complete elementary tail-recursive function $f_2(\overline{x})$, such that each pair of guards in $f_1(\overline{x})$, which contain linear modulo predicates which are not similar, are relpaced in $f_2(\overline{x})$ by a set of guards whose modulo predicates are similar. □

Consider a set $M = \{p_1(\overline{x}), \ldots, p_n(\overline{x})\}$ of similar conjunctions of linear modulo predicates. Let each $p_i(\overline{x})$ be of the form:

$$(\overline{\alpha_1}\ \overline{x} + \beta_{i1})\ mod\ \gamma_1 = 0 \ \wedge \ \cdots \ \wedge \ (\overline{\alpha_m}\ \overline{x} + \beta_{im})\ mod\ \gamma_m = 0$$

Let $\overline{\delta}$ be a tuple of integers. We define a relation $\mathcal{M}(\overline{\delta})$ on M as follows:

$$(p_i(\overline{x}), p_j(\overline{x})) \in \mathcal{M}(\overline{\delta}) \quad \text{iff} \quad ((\overline{\alpha_k} \cdot \overline{\delta} + \beta_{ik})\ mod\ \gamma_k \ = \ \beta_{jk}) \qquad \text{for} \qquad 1 \leq k \leq m$$

Intuitively if $(p_i(\overline{x}), p_j(\overline{x})) \in \mathcal{M}(\overline{\delta})$, then for each $\overline{\beta}$, if $p_i(\overline{\beta})$ is true then $p_j(\overline{\beta} + \overline{\delta})$ will be true. We call the relation $\mathcal{M}(\overline{\delta})$ above the *successive modulo relation.*

Let $\mathcal{M}^*(\overline{\delta})$ be the transitive closure of $\mathcal{M}(\overline{\delta})$. For any $p_i(\overline{x}) \in \mathcal{M}(\overline{\delta})$ if $(p_i(\overline{x}), p_i(\overline{x})) \in \mathcal{M}^*(\overline{\delta})$, then $p_i(\overline{x})$ is called *closed*, while if $(p_i(\overline{x}), p_i(\overline{x})) \notin \mathcal{M}^*(\overline{\delta})$, then $p_i(\overline{x})$ is called *open*. The $\mathcal{M}(\overline{\delta})$*-sequence* of $p_i(\overline{x})$ is the sequence whose first element is $p_i(\overline{x})$, and in which any element $p_j(\overline{x})$ is followed by an element $p_k(\overline{x})$ iff $(p_j(\overline{x}), p_k(\overline{x})) \in \mathcal{M}(\overline{\delta})$. Consider $p_i(\overline{x}) \in M$, then:

- If $p_i(\overline{x})$ is closed then the $\mathcal{M}(\overline{\delta})$-sequence of $p_i(\overline{x})$ will be of the form:

  $$p_{i_1}(\overline{x}), \ldots, p_{i_\alpha}(\overline{x}), p_{i_1}(\overline{x}), \ldots, p_{i_\alpha}(\overline{x}), \ldots$$

  where $i_1 = i$, and $i_j \neq i_k$ for $1 \leq j \neq k \leq \alpha$. The sequence:

  $$p_{i_1}(\overline{x}), \ldots, p_{i_\alpha}(\overline{x})$$

  is called the *basic* $\mathcal{M}(\overline{\delta})$*-sequence* of $p_i(\overline{x})$.

- If $p_i(\overline{x})$ is open then the $\mathcal{M}(\overline{\delta})$-sequence of $p_i(\overline{x})$ will be of the form:

  $$p_{i_1}(\overline{x}), \ldots, p_{i_\alpha}(\overline{x})$$

  where $i_1 = i$, $i_j \neq i_k$ for $1 \leq j \neq k \leq \alpha$, and there is no $p_j(\overline{x}) \in M$ such that $(p_{i_\alpha}(\overline{x}), p_j(\overline{x})) \in \mathcal{M}(\overline{\delta})$.

Consider a linear inequality $p(\overline{x})$ of the form $\overline{\alpha}\ \overline{x} + \beta \leq 0$. Let $\overline{\delta}$ be a tuple of integers. We call the minimum nonnegative i such that $p(\overline{x} + i \cdot \overline{\delta})$ is false (i.e. such that $\overline{\alpha}\ (\overline{x} + i \cdot \overline{\delta}) + \beta > 0$) the *$\overline{\delta}$-range* of the inequality. Let $\mu(\overline{x})$ denote the $\overline{\delta}$-range of $p(\overline{x})$. Observe that if $p(\overline{x})$ is false then $\mu(\overline{x}) = 0$. If $p(\overline{x})$ is true, then $\mu(\overline{x})$ is finite if $\overline{\alpha} \cdot \overline{\delta} > 0$, otherwise $\mu(\overline{x}) = \infty$. If $p(\overline{x})$ is true and $\overline{\alpha} \cdot \overline{\delta} > 0$ then:

$$\mu(\overline{x}) = \left\lceil \frac{1 - \overline{\alpha}\ \overline{x} - \beta}{\overline{\alpha} \cdot \overline{\delta}} \right\rceil$$

The *range* of an equality $p(\overline{x})$ of the form $\overline{\alpha}\ \overline{x} + \beta = 0$ is defined in the same way. Observe that if $p(\overline{x})$ is true, then $\mu(\overline{x}) = 1$ if $\overline{\alpha} \cdot \overline{\delta} \neq 0$, otherwise $\mu(\overline{x}) = \infty$. Now given a conjunction $p_1(\overline{x}) \wedge \ldots \wedge p_m(\overline{x})$ of equalities and inequalities, we define the *$\overline{\delta}$-range* of the conjunction as the minimum of the $\overline{\delta}$-ranges of the elements of the conjunction. Clearly the range of the conjunction above is the minimum i such that any of $p_1(\overline{x} + i \cdot \overline{\delta}), \ldots, p_m(\overline{x} + i \cdot \overline{\delta})$ is false.

Consider a tail-recursive function f of the form:

$$\begin{array}{lll} f(\overline{x}) & = & \textbf{case} \\ & & \quad p_1(\overline{x}) \Rightarrow r_1(\overline{x})\ ;\ \ldots\ ;\ p_m(\overline{x}) \Rightarrow r_m(\overline{x})\ ; \\ & & \quad p_{m+1}(\overline{x}) \Rightarrow f(\overline{x} + \overline{\delta}) + r_{m+1}(\overline{x})\ ;\ \ldots\ ;\ p_n(\overline{x}) \Rightarrow f(\overline{x} + \overline{\delta}) + r_n(\overline{x}) \\ & & \textbf{endcase} \end{array}$$

Denote the cases of f by $c_1, \ldots, c_m, c_{m+1}, \ldots, c_n$. We call $c_1, \ldots, c_m$ the *non-recursive cases* of f, and $c_{m+1}, \ldots, c_n$ the *recursive cases* of f. For any value $\overline{\beta}$ of $\overline{x}$, the recursive sequence of $f(\overline{\beta})$ is of the form:

$$i_1, i_2, \ldots, i_\alpha$$

where $p_{i_j}(\overline{\beta} + (j-1) \cdot \overline{\delta})$ is true for $1 \leq j \leq \alpha$, c_{i_j} is recursive for $1 \leq j \leq \alpha - 1$, and either c_{i_α} is non-recursive, or $p_k(\overline{\beta} + \alpha \cdot \overline{\delta})$ is false for $1 \leq k \leq n$. Observe that the recursive sequence of $f(\overline{\beta})$ may be:

- empty i.e. $\alpha = 0$ (when $p_k(\overline{\beta})$ is false for $1 \leq k \leq n$). In this case $f(\overline{\beta}) = \perp$.
- infinite i.e. $\alpha = \infty$ (when for each $0 \leq j$, there is a k such that $m+1 \leq k \leq n$ and $p_k(\overline{\beta} + j \cdot \overline{\delta})$ is true). In this case $f(\overline{\beta}) = \perp$.
- finite and $1 \leq i_\alpha \leq m$ (when for each $0 \leq j < \alpha - 1$, there is a k such that $m+1 \leq k \leq n$ and $p_k(\overline{\beta} + j \cdot \overline{\delta})$ is true; and there is a k' such that $1 \leq k' \leq m$ and $p_{k'}(\overline{\beta} + (\alpha - 1) \cdot \overline{\delta})$ is true). In this case $f(\overline{\beta}) = \sum_{j=0}^{\alpha-1} r_{i_{j+1}}(\overline{\beta} + j \cdot \overline{\delta})$, over each ring R.
- finite and $m+1 \leq i_\alpha \leq n$ (when for each $0 \leq j \leq \alpha - 1$, there is a k such that $m+1 \leq k \leq n$ and $p_k(\overline{\beta} + j \cdot \overline{\delta})$ is true; and there is no k' such that $p_{k'}(\overline{\beta} + \alpha \cdot \overline{\delta})$ is true). In this case $f(\overline{\beta}) = \perp$.

Observe that in each case whether $f(\overline{\beta}) = \perp$ or not depends entirely on the guards of f, and is independent on any particular stream interpretation $\mathcal{I}$ or ring R.

Lemma 14 *For each elementary tail-recursive function f, there is a linear guarded ring expression e such that $f(\overline{x}) = e(\overline{x})$ over each ring R.*

Proof: Let R be an arbitrary ring. From lemma 13 we can assume without loss of generality that f is complete. Let $f(\overline{x})$ be of the form:

$f(\overline{x}) =$
case
$p_1(\overline{x}) \Rightarrow r_1(\overline{x})$; ... ; $p_m(\overline{x}) \Rightarrow r_m(\overline{x})$;
$p(\overline{x}) \wedge p_{m+1}(\overline{x}) \Rightarrow f(\overline{x}+\overline{\delta}) + r_{m+1}(\overline{x})$; ... ; $p(\overline{x}) \wedge p_n(\overline{x}) \Rightarrow f(\overline{x}+\overline{\delta}) + r_n(\overline{x})$
endcase

We define the successive modulo relation $\mathcal{M}(\overline{\delta})$ on the set $\{p_{m+1}(\overline{x}), \ldots, p_n(\overline{x})\}$. For any value $\overline{\beta}$ of $\overline{x}$ if $f(\overline{\beta})$ is defined, then a number of cases are possible:
1. If $p_i(\overline{\beta})$ is true, for some $1 \leq i \leq m$. In this case $f(\overline{\beta}) = r_i(\overline{\beta})$ over R.
2. If $p(\overline{\beta}) \wedge p_i(\overline{\beta})$ is true, for some $m+1 \leq i \leq n$, and $p_i(\overline{x})$ is closed. We observe that $f(\overline{\beta})$ is defined only if the $\overline{\delta}$-range $\mu(\overline{\beta})$ of $p(\overline{\beta})$ is finite, and $p_j(\overline{\beta} + \mu(\overline{\beta}) \cdot \overline{\delta})$ is true for some $1 \leq j \leq m$. In this case the recursive sequence of $f(\overline{\beta})$ will be of the form:

$$\underbrace{i_1, \ldots, i_\alpha}_{1}, \underbrace{i_1, \ldots, i_\alpha}_{2}, \cdots, \underbrace{i_1, \ldots, i_\alpha}_{k}, \underbrace{i_1, \ldots, i_\gamma}, i'$$

where:

$$p_{i_1}(\overline{x}), \ldots, p_{i_\alpha}(\overline{x})$$

is the basic $\mathcal{M}(\overline{\delta})$-sequence of $p_i(\overline{x})$, and:

$$k = \mu(\overline{\beta})\ div\ \alpha \qquad \gamma = \mu(\overline{\beta})\ mod\ \alpha$$
$$m+1 \leq i_j \leq n \text{ for } 1 \leq j \leq \alpha, \qquad 1 \leq i' \leq m$$

This means that:

$$f(\overline{\beta}) = \left(\sum_{j=0}^{k-1} r_{i_1}(\overline{\beta} + j \cdot \alpha \cdot \overline{\delta}) + \cdots + r_{i_\alpha}(\overline{\beta} + (j \cdot \alpha + \alpha - 1) \cdot \overline{\delta})\right) + r_{i_1}(\overline{\beta} + k \cdot \alpha \cdot \overline{\delta}) + \cdots + r_{i_\gamma}(\overline{\beta} + (k \cdot \alpha + \gamma - 1) \cdot \overline{\delta}) + r_{i'}(\overline{\beta} + (k \cdot \alpha + \gamma) \cdot \overline{\delta})$$

over R. Observe that if $k = 0$ then:

$$f(\overline{\beta}) = r_{i_1}(\overline{\beta}) + \cdots + r_{i_\gamma}(\overline{\beta} + (\gamma - 1) \cdot \overline{\delta}) + r_{i'}(\overline{\beta} + \gamma \cdot \overline{\delta})$$

3. If $p(\overline{\beta}) \wedge p_i(\overline{\beta})$ is true, for some $m+1 \leq i \leq n$, and $p_i(\overline{x})$ is open. Let $\mu(\overline{\beta})$ be the $\overline{\delta}$-range of $p(\overline{\beta})$, and let:

$$p_{i_1}(\overline{x}), \ldots, p_{i_\alpha}(\overline{x})$$

be the $\mathcal{M}(\overline{\delta})$-sequence of $p_i(\overline{x})$. We observe that if $f(\overline{\beta})$ is defined then the recursive sequence of $f(\overline{\beta})$ will be of the form:

$$i_1, \ldots, i_\gamma, i'$$

where $\gamma = min(\mu(\overline{\beta}), \alpha)$, $1 \leq i' \leq m$, and $p_{i'}(\overline{\beta} + \gamma \cdot \overline{\delta})$ is true. In this case:

$$f(\overline{\beta}) \;=\; r_{i_1}(\overline{\beta}) + \cdots + r_{i_\gamma}(\overline{\beta} + (\gamma - 1) \cdot \overline{\delta}) + r_{i'}(\overline{\beta} + \gamma \cdot \overline{\delta})$$

over R.

Now we will define a linear guarded ring expression $e(\overline{x})$, such that $f(\overline{x}) = e(\overline{x})$ over R. The set C of the cases of $e(\overline{x})$ reflects the above three possibilities. The set C is the smallest set containing the following elements:

1. For each $1 \leq i \leq m$, C contains the case:

$$\left\{\; p_i(\overline{x}) \;\right\} \Longrightarrow\; r_i(\overline{x})$$

2. For each $m+1 \leq i \leq n$, if $p_i(\overline{x})$ is closed, then let:

$$p_{i_1}(\overline{x}), \ldots, p_{i_\alpha}(\overline{x})$$

be the basic $\mathcal{M}(\overline{\delta})$-sequence of $p_i(\overline{x})$, and $\mu(\overline{x})$ the $\overline{\delta}$-range of $p(\overline{x})$. Let:

$$k(\overline{x}) = \mu(\overline{x})\ div\ \alpha \qquad \gamma(\overline{x}) = \mu(\overline{x})\ mod\ \alpha$$

For each $1 \leq i' \leq m$, C contains the cases:

$$\left\{ \begin{array}{c} p(\overline{x}) \\ p_i(\overline{x}) \\ p_{i'}(\overline{x} + \mu(\overline{x}) \cdot \overline{\delta}) \\ \alpha \leq \mu(\overline{x}) \end{array} \right\} \Longrightarrow \begin{array}{c} \left(\displaystyle\sum_{j=0}^{k(\overline{x})-1} r_{i_1}(\overline{x} + j \cdot \alpha \cdot \overline{\delta}) + \cdots + r_{i_\alpha}(\overline{x} + (j \cdot \alpha + \alpha - 1) \cdot \overline{\delta}) \right) \\ + \\ r_{i_1}(\overline{x} + k(\overline{x}) \cdot \alpha \cdot \overline{\delta}) + \;\cdots\; + r_{i_{\gamma(\overline{x})}}(\overline{x} + (k(\overline{x}) \cdot \alpha + \gamma(\overline{x}) - 1) \cdot \overline{\delta}) \\ + \\ r_{i'}(\overline{x} + (k(\overline{x}) \cdot \alpha + \gamma(\overline{x})) \cdot \overline{\delta}) \end{array}$$

and:

$$\left\{ \begin{array}{c} p(\overline{x}) \\ p_i(\overline{x}) \\ p_{i'}(\overline{x} + \mu(\overline{x}) \cdot \overline{\delta}) \\ \alpha > \mu(\overline{x}) \end{array} \right\} \Longrightarrow r_{i_1}(\overline{x}) + \;\cdots\; + r_{i_{\gamma(\overline{x})}}(\overline{x} + (\gamma(\overline{x}) - 1) \cdot \overline{\delta}) + r_{i'}(\overline{x} + \gamma(\overline{x}) \cdot \overline{\delta})$$

3. For each $m+1 \leq i \leq n$, if $p_i(\overline{x})$ is open, then let:

$$p_{i_1}(\overline{x}), \ldots, p_{i_\alpha}(\overline{x})$$

be the $\mathcal{M}(\overline{\delta})$-sequence of $p_i(\overline{x})$. Let $\mu(\overline{x})$ be the $\overline{\delta}$-range of $p(\overline{x})$, and let $\gamma(\overline{x}) = min(\mu(\overline{x}), \alpha)$. Then, for each $1 \leq i' \leq m$, C contains the case:

$$\left\{ \begin{array}{c} p(\overline{x}) \\ p_i(\overline{x}) \\ p_{i'}(\overline{x} + \gamma(\overline{x}) \cdot \overline{\delta}) \end{array} \right\} \Longrightarrow\; r_{i_1}(\overline{x}) + \;\cdots\; + r_{i_{\gamma(\overline{x})}}(\overline{x} + (\gamma(\overline{x}) - 1) \cdot \overline{\delta}) + r_{i'}(\overline{x} + \gamma(\overline{x}) \cdot \overline{\delta})$$

Now we observe that if $r(\overline{x})$ is a ring expression, then $r(\overline{x})[x_i \leftarrow (\overline{\alpha}\ \overline{x} + \beta)\ mod\ \gamma]$, where γ is a positive integer, can be written as:

case

$$\begin{array}{rcl} \left\{\; (\overline{\alpha}\ \overline{x} + \beta)\ mod\ \gamma = 0 \;\right\} & \Longrightarrow & r(\overline{x})[x_i \leftarrow 0] \\ & \vdots & \\ \left\{\; (\overline{\alpha}\ \overline{x} + \beta - \gamma + 1)\ mod\ \gamma = 0 \;\right\} & \Longrightarrow & r(\overline{x})[x_i \leftarrow \gamma - 1] \end{array}$$

endcase

and $r(\overline{x})[x_i \leftarrow (\overline{\alpha}\ \overline{x} + \beta)\ div\ \gamma]$, where γ is a positive integer, can be written as:

$$\begin{array}{l}\textbf{case}\\ \qquad \left\{\ (\overline{\alpha}\ \overline{x} + \beta)\ mod\ \gamma = 0\ \right\} \implies r(\overline{x})[x_i \leftarrow \frac{\overline{\alpha}\ \overline{x}+\beta}{\gamma}]\\ \qquad\qquad \vdots\\ \left\{\ (\overline{\alpha}\ \overline{x} + \beta - \gamma + 1)\ mod\ \gamma = 0\ \right\} \implies r(\overline{x})[x_i \leftarrow \frac{\overline{\alpha}\ \overline{x}+\beta-\gamma+1}{\gamma}]\\ \textbf{endcase}\end{array}$$

Thus each case in C, in which $\overline{\alpha}\ \overline{x} + \beta\ mod\ \gamma$ or $(\overline{\alpha}\ \overline{x} + \beta)\ div\ \gamma$ occurs in the result, can be replaced by a number of equivalent cases in which $\overline{\alpha}\ \overline{x} + \beta\ mod\ \gamma$ or $(\overline{\alpha}\ \overline{x} + \beta)\ div\ \gamma$ is replaced in the result by an integer or a linear QI-polynomial respectively, and such that the result of each new case is still well-defined under the guard of the case. Similarly all occurences of forms $\lceil \overline{\alpha}\ \overline{x} + \beta/\gamma \rceil$, $min(\overline{\alpha}\ \overline{x} + \beta, \gamma)$, or $min(\overline{\alpha}_1\ \overline{x}_1 + \beta_1, \overline{\alpha}_2\ \overline{x}_2 + \beta_2)$ in the results of C can be replaced by integers or linear polynomials. It follows that each element of C will be of the form $p(\overline{x}) \Rightarrow r(\overline{x})$, where $p(\overline{x})$ is a conjunction of linear predicates and $r(\overline{x})$ is a ring expression.

We observe that the construction of the cases of $e(\overline{x})$ does not depend on the particular ring R. It follows that $f(\overline{x}) = e(\overline{x})$ over each ring R. □

Tail-Recursive functions as Linear Guarded Ring Expressions

We will show that for each tail-recursive function, there is a linear guarded ring expression which is equal to it over each ring R.

Let $\mathcal{F}(\overline{x})$ be any functional of the form:

$$\begin{array}{l}\textbf{case}\\ \qquad p_1(\overline{x}) \implies t_1(\overline{x})\\ \qquad\qquad \vdots\\ \qquad p_n(\overline{x}) \implies t_n(\overline{x})\\ \textbf{endcase}\end{array}$$

Then the result of *deleting* the case $p_i(\overline{x}) \Rightarrow t_i(\overline{x})$ in $\mathcal{F}(\overline{x})$ is the functional:

$$\begin{array}{l}\textbf{case}\\ \qquad p_1(\overline{x}) \implies t_1(\overline{x})\\ \qquad\qquad \vdots\\ \qquad p_{i-1}(\overline{x}) \implies t_{i-1}(\overline{x})\\ \qquad p_{i+1}(\overline{x}) \implies t_{i+1}(\overline{x})\\ \qquad\qquad \vdots\\ \qquad p_n(\overline{x}) \implies t_n(\overline{x})\\ \textbf{endcase}\end{array}$$

Theorem 15 *For each tail-recursive function f, there is a linear guarded ring expression e such that $f(\overline{x}) = e(\overline{x})$ over each ring R.*

Proof: Let R be an arbitrary ring. Let $f(\overline{x})$ be a tail-recursive function. It is clear that $f(\overline{x})$ can be written as:

$$f(\overline{x}) = \tag{5.15}$$

case

$$p_1(\overline{x}) \wedge p_1^{(1)}(\overline{x}) \Rightarrow r_1^{(1)}(\overline{x}) \; ; \; \ldots \; ; p_1(\overline{x}) \wedge p_{m_1}^{(1)}(\overline{x}) \Rightarrow r_{m_1}^{(1)}(\overline{x}) \; ;$$
$$p_1(\overline{x}) \wedge p_{m_1+1}^{(1)}(\overline{x}) \Rightarrow f(\overline{x}+\overline{\delta}) + r_{m_1+1}^{(1)}(\overline{x}) \; ; \; \ldots \; ; p_1(\overline{x}) \wedge p_{n_1}^{(1)}(\overline{x}) \Rightarrow f(\overline{x}+\overline{\delta}) + r_{n_1}^{(1)}(\overline{x}) \; ;$$
$$\vdots$$
$$p_k(\overline{x}) \wedge p_1^{(k)}(\overline{x}) \Rightarrow r_1^{(k)}(\overline{x}) \; ; \; \ldots \; ; p_k(\overline{x}) \wedge p_{m_k}^{(k)}(\overline{x}) \Rightarrow r_{m_k}^{(k)}(\overline{x}) \; ;$$
$$p_k(\overline{x}) \wedge p_{m_k+1}^{(k)}(\overline{x}) \Rightarrow f(\overline{x}+\overline{\delta}) + r_{m_k+1}^{(k)}(\overline{x}) \; ; \; \ldots \; ; p_k(\overline{x}) \wedge p_{n_k}^{(k)}(\overline{x}) \Rightarrow f(\overline{x}+\overline{\delta}) + r_{n_k}^{(k)}(\overline{x})$$

endcase

where $1 \leq k$, $p_i(\overline{x})$ is a conjunction of linear inequalities and equalities, $p_i^{(j)}(\overline{x})$ is a conjunction of linear modulo predicates, $p_{i_1}(\overline{\alpha}) \wedge p_{i_2}(\overline{\alpha})$ is false for each $\overline{\alpha}$ if $i_2 \neq i_1$, and $p_{i_1}^{(j)}(\overline{\alpha}) \wedge p_{i_2}^{(j)}(\overline{\alpha})$ is false for each j and $\overline{\alpha}$ if $i_2 \neq i_1$. We will show the theorem by induction on k.

Base Case ($k = 1$): If $k = 1$ then $f(\overline{x})$ is an elementary tail-recursive function. The result follows from lemma 14.

Induction Step: We observe that for each value $\overline{\beta}$ of $\overline{x}$, $f(\overline{\beta})$ is defined only if $p_i(\overline{\beta})$ is true for some $1 \leq i \leq k$. Furthermore if $p_i(\overline{\beta})$ is true then p_i continues to be true in the subsequent recursive calls of f, until the number of recursive calls is equal to the $\overline{\delta}$-range $\mu_i(\overline{\beta})$ of $p_i(\overline{\beta})$ (in which case p_i becomes false). Also, once p_i has become false then it will not become true again at any later recursive call of f. Formally:

$$p_i(\overline{x}) \longrightarrow (\forall\, j < \mu_i(\overline{x}).\, p_i(\overline{x} + j \cdot \overline{\delta})) \wedge (\forall\, j \geq \mu_i(\overline{x}).\, \neg p_i(\overline{x} + j \cdot \overline{\delta})) \tag{5.16}$$

Let $h_i(\overline{x})$ be the result of deleting the cases:

$$p_i(\overline{x}) \wedge p_1^{(i)}(\overline{x}) \Rightarrow r_1^{(i)}(\overline{x}) \; , \; \ldots \; , p_i(\overline{x}) \wedge p_{m_i}^{(i)}(\overline{x}) \Rightarrow r_{m_i}^{(i)}(\overline{x}) \; ,$$
$$p_i(\overline{x}) \wedge p_{m_i+1}^{(i)}(\overline{x}) \Rightarrow f(\overline{x}+\overline{\delta}) + r_{m_i+1}^{(i)}(\overline{x}) \; , \; \ldots \; , p_i(\overline{x}) \wedge p_{n_i}^{(i)}(\overline{x}) \Rightarrow f(\overline{x}+\overline{\delta}) + r_{n_i}^{(i)}(\overline{x})$$

in $f(\overline{x})$. Then from (5.16) it follows that for R:

$$p_i(\overline{x}) \longrightarrow (f(\overline{x}) = f_i(\overline{x})) \tag{5.17}$$

where:

$$\begin{array}{lll} p_i(\overline{x}) \wedge p_j^{(i)}(\overline{x}) \longrightarrow (f_i(\overline{x}) = r_j^{(i)}(\overline{x})) & & \text{for } 1 \leq j \leq m_i \\ p_i(\overline{x}) \wedge p_j^{(i)}(\overline{x}) \longrightarrow (f_i(\overline{x}) = f_i(\overline{x}+\overline{\delta}) + r_j^{(i)}(\overline{x})) & & \text{for } m_i + 1 \leq j \leq n_i \\ \neg p_i(\overline{x}) \longrightarrow (f_i(\overline{x}) = h_i(\overline{x})) & & \end{array} \tag{5.18}$$

It can be shown that there are conjunctions of linear inequalities and equalities $P_1^{(i)}(\overline{x})$, $\ldots, P_{l_i}^{(i)}(\overline{x})$, such that for each $\overline{\alpha}$, $P_{j_1}^{(i)}(\overline{\alpha}) \wedge P_{j_2}^{(i)}(\overline{\alpha})$ is false if $j_2 \neq j_1$, and:

$$\neg p_i(\overline{x}) = P_1^{(i)}(\overline{x}) \vee \cdots \vee P_{l_i}^{(i)}(\overline{x}) \tag{5.19}$$

Also from the induction hypothesis it follows that there is a linear guarded ring expression $e^{(i)}$ such that:

$$h_i(\overline{x}) = e^{(i)}(\overline{x}) \tag{5.20}$$

over R. From (5.18), (5.19), and (5.20) it follows that:

$$
\begin{array}{l}
f_i(\overline{x}) \; = \\
\textbf{case} \\
\quad P_1^{(i)}(\overline{x}) \Rightarrow e^{(i)}(\overline{x}) \; ; \; \ldots \; ; \; P_{l_i}^{(i)}(\overline{x}) \Rightarrow e^{(i)}(\overline{x}) \; ; \\
\quad p_i(\overline{x}) \wedge p_1^{(i)}(\overline{x}) \Rightarrow r_1^{(i)}(\overline{x}) \; ; \; \ldots \; ; \; p_i(\overline{x}) \wedge p_{m_i}^{(i)}(\overline{x}) \Rightarrow r_{m_i}^{(i)}(\overline{x}) \; ; \\
\quad p_i(\overline{x}) \wedge p_{m_i+1}^{(i)}(\overline{x}) \Rightarrow f(\overline{x}+\overline{\delta}) + r_{m_i+1}^{(i)}(\overline{x}) \; ; \; \ldots \; ; \; p_i(\overline{x}) \wedge p_{n_i}^{(i)}(\overline{x}) \Rightarrow f(\overline{x}+\overline{\delta}) + r_{n_i}^{(i)}(\overline{x}) \\
\textbf{endcase}
\end{array}
\tag{5.21}
$$

From the above equation we can easily conclude that $f_i(\overline{x})$ is an elementary tail-recursive function. From lemma 14 it follows that there is a linear guarded ring expression e_i such that:

$$f_i(\overline{x}) \; = \; e_i(\overline{x}) \tag{5.22}$$

over R. Let:

$$e(\overline{x}) \; = \; \textbf{case} \;\; p_1(\overline{x}) \Rightarrow e_1(\overline{x}) \; ; \; \ldots \; ; \; p_k(\overline{x}) \Rightarrow e_k(\overline{x}) \;\; \textbf{endcase} \tag{5.23}$$

It follows easily that $e(\overline{x})$ can be written as a linear guarded ring expression. From (5.17), (5.22), and (5.23) we get:

$$f(\overline{x}) \; = \; e(\overline{x})$$

over R. We observe that the construction of $e(\overline{x})$ does not depend on the particular ring R. It follows that $f(\overline{x}) = e(\overline{x})$ over each ring R. □

5.5.2 Describing Tail-recursive Ring Systolic Circuits by Linear Guarded Ring Expressions

In this section we show that, for tail-recursive ring systolic circuits, the values of outputs, inputs, and local variables of the different cells at different time instants can be described as linear guarded ring expressions. This is achieved by the following theorem:

Theorem 16 *Consider a tail-recursive ring systolic circuit. Let s be any signal in the circuit. Then there is a linear guarded ring expression e such that:*

$$s(\overline{x}, \overline{\ell}, t) \; = \; e(\overline{x}, \overline{\ell}, t)$$

over each ring R.

Proof: We will prove the theorem by induction on the dependency relation (see section 5.2.2). We will show the claim when s is any out_i. The proof is similar when s is any in_i or loc_i. Let the computation equation of out_i be of the form of (5.4). Let the corresponding initial function, input function, and connection vector be $Init(\overline{x}, \overline{\ell}, t)$, $Input(\overline{x}, \overline{\ell}, t)$, and

$\overline{\delta}$ respectively. Recalling equations (2.2), (5.1), (2.3), (2.16), (5.3), and (2.15) we get:

$$
\begin{array}{l}
s(\overline{x},\overline{\ell},t) \;=\; \\
\textbf{case} \\
\qquad \left\{ \begin{array}{c} 0 \le t < \tau \\ top(\overline{x},\overline{\ell}) \end{array} \right\} \implies Init(\overline{x},\overline{\ell},t) \\
\qquad \left\{ \begin{array}{c} t \ge \tau \\ top(\overline{x},\overline{\ell}) \\ p_1(\overline{x},\overline{\ell},t) \end{array} \right\} \implies Q_1(s_1(\overline{x},\overline{\ell},t-\tau),\ldots,s_k(\overline{x},\overline{\ell},t-\tau)) \\
\qquad\qquad \vdots \\
\qquad \left\{ \begin{array}{c} t \ge \tau \\ top(\overline{x},\overline{\ell}) \\ p_m(\overline{x},\overline{\ell},t) \end{array} \right\} \implies Q_m(s_1(\overline{x},\overline{\ell},t-\tau),\ldots,s_k(\overline{x},\overline{\ell},t-\tau)) \\
\left\{ \begin{array}{c} t \ge \tau \\ top''(\overline{x},\overline{\ell}) \\ p_{m+1}(\overline{x},\overline{\ell},t) \end{array} \right\} \implies \left(\begin{array}{c} Input(\overline{x},\overline{\ell},t-\tau) \\ + \\ Q_{m+1}(s_1(\overline{x},\overline{\ell},t-\tau),\ldots,s_k(\overline{x},\overline{\ell},t-\tau)) \end{array} \right) \\
\qquad\qquad \vdots \\
\left\{ \begin{array}{c} t \ge \tau \\ top''(\overline{x},\overline{\ell}) \\ p_n(\overline{x},\overline{\ell},t) \end{array} \right\} \implies \left(\begin{array}{c} Input(\overline{x},\overline{\ell},t-\tau) \\ + \\ Q_n(s_1(\overline{x},\overline{\ell},t-\tau),\ldots,s_k(\overline{x},\overline{\ell},t-\tau)) \end{array} \right) \\
\left\{ \begin{array}{c} t \ge \tau \\ top'(\overline{x},\overline{\ell}) \\ p_{m+1}(\overline{x},\overline{\ell},t) \end{array} \right\} \implies \left(\begin{array}{c} s(\overline{x}+\overline{\delta},\overline{\ell},t-\tau) \\ + \\ Q_{m+1}(s_1(\overline{x},\overline{\ell},t-\tau),\ldots,s_k(\overline{x},\overline{\ell},t-\tau)) \end{array} \right) \\
\qquad\qquad \vdots \\
\left\{ \begin{array}{c} t \ge \tau \\ top'(\overline{x},\overline{\ell}) \\ p_n(\overline{x},\overline{\ell},t) \end{array} \right\} \implies \left(\begin{array}{c} s(\overline{x}+\overline{\delta},\overline{\ell},t-\tau) \\ + \\ Q_n(s_1(\overline{x},\overline{\ell},t-\tau),\ldots,s_k(\overline{x},\overline{\ell},t-\tau)) \end{array} \right) \\
\textbf{endcase}
\end{array}
\tag{5.24}
$$

From the definition of the dependency relation we know that $s_1,\ldots,s_k \prec_D s$. It follows from the induction hypothesis that there are linear guarded ring expressions $e_1,\ldots,e_k$ such that:

$$s_j(\overline{x},\overline{\ell},t) \;=\; e_j(\overline{x},\overline{\ell},t)$$

over each ring R. Furthermore we know that $Init(\overline{x},\overline{\ell},t)$ and $Input(\overline{x},\overline{\ell},t)$, in tail-recursive ring circuits, are described as linear guarded ring expressions. It follows easily that out_i is a tail-recursive function. From theorem 15 it follows that there is a linear guarded ring expression e such that:

$$s(\overline{x},\overline{\ell},t) \;=\; e(\overline{x},\overline{\ell},t)$$

over each ring R. □

5.5.3 First Step of Verification of The Convolution Circuit: Describing it by Linear Guarded Ring Expressions

Theorem 16 can be applied to the convolution circuit introduced in section 3.1, in order to describe the values of the wires and local variables as linear guarded ring expressions. The details of the application of the theorem are not given here.

From (3.12), (3.16), (3.3), and (3.6) we get:

$$
\begin{aligned}
out_1(x,\ell,t) \;=\; & \textbf{case} \\
& \left\{ \begin{array}{c} 0 \le t < 2 \\ 0 \le x \le \ell \end{array} \right\} \implies OutInit_1(x,\ell,t) \\
& \left\{ \begin{array}{c} t \ge 2 \\ x = 0 \\ t \bmod 2 = 0 \end{array} \right\} \implies Input_1(x,\ell,t-2) \\
& \left\{ \begin{array}{c} t \ge 2 \\ 1 \le x \le \ell \\ t \bmod 2 = 0 \end{array} \right\} \implies out_1(x-1,\ell,t-2) \\
& \textbf{endcase}
\end{aligned}
$$

From (3.16) and (3.9) we get:

$$
\begin{aligned}
out_1(x,\ell,t) \;=\; & \textbf{case} \\
& \left\{ \begin{array}{c} 4 \le t \le 4\ell+4 \\ x = 0 \\ t \bmod 4 = 0 \end{array} \right\} \implies a\left(\tfrac{t-4}{4}\right) \\
& \left\{ \begin{array}{c} 4\ell+4 < t \le 8\ell+4 \\ x = 0 \\ t \bmod 4 = 0 \end{array} \right\} \implies 0 \\
& \left\{ \begin{array}{c} t \ge 2 \\ 1 \le x \le \ell \\ t \bmod 2 = 0 \end{array} \right\} \implies out_1(x-1,\ell,t-2) \\
& \textbf{endcase}
\end{aligned}
$$

By applying theorem 15 we get:

$$
\begin{aligned}
out_1(x,\ell,t) \;=\; & \textbf{case} \\
& \left\{ \begin{array}{c} 2x+4 \le t \le 2x+4\ell+4 \\ 0 \le x \le \ell \\ (t-2x)\ mod\ 4 = 0 \end{array} \right\} \implies a\left(\tfrac{t-2x-4}{4}\right) \\
& \left\{ \begin{array}{c} 2x+4\ell+4 < t \le 2x+8\ell+4 \\ 0 \le x \le \ell \\ (t-2x)\ mod\ 4 = 0 \end{array} \right\} \implies 0 \\
& \textbf{endcase}
\end{aligned}
\tag{5.25}
$$

which is a linear guarded ring expression.

Similarly we can show that:

$$
\begin{aligned}
in_1(x,\ell,t) \;=\; & \textbf{case} \\
& \left\{ \begin{array}{c} 2x+2 \le t \le 2x+4\ell+2 \\ 0 \le x \le \ell \\ (t-2x+2)\ mod\ 4 = 0 \end{array} \right\} \implies a\left(\tfrac{t-2x-2}{4}\right) \\
& \left\{ \begin{array}{c} 2x+4\ell+2 < t \le 2x+8\ell+2 \\ 0 \le x \le \ell \\ (t-2x+2)\ mod\ 4 = 0 \end{array} \right\} \implies 0 \\
& \textbf{endcase}
\end{aligned}
\tag{5.26}
$$

and:

$$
\begin{aligned}
in_2(x,\ell,t) \;=\; & \textbf{case} \\
& \left\{ \begin{array}{c} x \le t \le x+\ell \\ 0 \le x \le \ell \end{array} \right\} \implies b(t-x) \\
& \textbf{endcase}
\end{aligned}
\tag{5.27}
$$

and:

$$
\begin{aligned}
loc(x,\ell,t) \;=\; & \textbf{case} \\
& \left\{ \begin{array}{c} 2x+2 \le t \\ 0 \le x \le \ell \\ t\ mod\ 2 = 0 \end{array} \right\} \implies b(x) \\
& \textbf{endcase}
\end{aligned}
\tag{5.28}
$$

From (3.15), (3.18), (3.5), and (3.8) we get:

$$
\begin{array}{l}
out_3(x,\ell,t) = \\
\textbf{case} \\
\quad \left\{ \begin{array}{c} 0 \le t < 2 \\ 0 \le x \le \ell \end{array} \right\} \implies OutInit_3(x,\ell,t) \\
\quad \left\{ \begin{array}{c} t \ge 2 \\ t = 2x+4 \\ 0 \le x \le \ell \end{array} \right\} \implies in_1(x,\ell,t-2) \cdot loc(x,\ell,t-2) \\
\quad \left\{ \begin{array}{c} t \ge 2 \\ 2x+4 < t \\ x = \ell \\ t \bmod 2 = 0 \end{array} \right\} \implies Input_3(x,\ell,t-2) + in_1(x,\ell,t-2) \cdot loc(x,\ell,t-2) \\
\quad \left\{ \begin{array}{c} t \ge 2 \\ 2x+4 < t \\ 0 \le x < \ell \\ t \bmod 2 = 0 \end{array} \right\} \implies out_3(x+1,\ell,t-2) + in_1(x,\ell,t-2) \cdot loc(x,\ell,t-2) \\
\textbf{endcase}
\end{array}
$$

From (3.18), (3.11), (5.26), and (5.28) we get:

$$
\begin{array}{l}
out_3(x,\ell,t) = \\
\textbf{case} \\
\quad \left\{ \begin{array}{c} t = 2x+4 \\ 0 \le x \le \ell \end{array} \right\} \implies a\left(\frac{t-2x-4}{4}\right) \cdot b(x) \\
\quad \left\{ \begin{array}{c} 2\ell+8 \le t \le 6\ell+4 \\ x = \ell \\ (t+2\ell) \bmod 4 = 0 \end{array} \right\} \implies a\left(\frac{t-2x-4}{4}\right) \cdot b(x) \\
\quad \left\{ \begin{array}{c} 2x+4 < t \le 2x+4\ell+4 \\ 0 \le x < \ell \\ (t-2x) \bmod 4 = 0 \end{array} \right\} \implies out_3(x+1,\ell,t-2) + a\left(\frac{t-2x-4}{4}\right) \cdot b(x) \\
\quad \left\{ \begin{array}{c} 2x+4\ell+4 < t \le 2x+8\ell+4 \\ 0 \le x < \ell \\ (t-2x) \bmod 4 = 0 \end{array} \right\} \implies out_3(x+1,\ell,t-2) \\
\textbf{endcase}
\end{array}
$$

By applying theorem 15 we get:

$$
\begin{array}{l}
out_3(x,\ell,t) = \\
\textbf{case} \\
\quad \left\{ \begin{array}{c} 2x+4 \le t \le 4\ell - 2x + 4 \\ 0 \le x \le \ell \\ (t-2x)\ mod\ 4 = 0 \end{array} \right\} \implies \sum_{i=0}^{\frac{t-2x-4}{4}} a\left(\frac{t-2x-4i-4}{4}\right) \cdot b(x+i) \\
\quad \left\{ \begin{array}{c} 4\ell - 2x + 8 \le t \le 4\ell + 2x + 4 \\ 0 \le x \le \ell \\ (t-2x)\ mod\ 4 = 0 \end{array} \right\} \implies \sum_{i=0}^{\ell - x} a\left(\frac{t-2x-4i-4}{4}\right) \cdot b(x+i) \qquad (5.29) \\
\quad \left\{ \begin{array}{c} 4\ell + 2x + 8 \le t \le 8\ell - 2x + 4 \\ 0 \le x \le \ell \\ (t-2x)\ mod\ 4 = 0 \end{array} \right\} \implies \sum_{i=0}^{\frac{8\ell-2x-t+4}{4}} a(\ell - i) \cdot b\left(\frac{t-4\ell+2x+4i-4}{4}\right) \\
\textbf{endcase}
\end{array}
$$

which is a linear guarded ring expression.

5.6 Equality Checking

In this section we will accomplish the second step of the verification of tail-recursive ring circuits; the first step being given in section 5.5. We will study the validity of formulas of the form:

$$p(\overline{x}) \longrightarrow (e(\overline{x}) = r(\overline{x})) \qquad (5.30)$$

where $p(\overline{x})$ is a conjunction of linear predicates, $e(\overline{x})$ is a linear guarded ring expression, and $r(\overline{x})$ is a ring expression which is well-defined under $p(\overline{x})$. We will consider a class of formulas which we call *bilinear* formulas, and show that if a formula of the form of (5.30) is bilinear, then it is decidable to check whether or not it is valid for the class of all rings. We will also show that the validity of bilinear formulas of the form of (5.30) is decidable for some interesting rings such as the ring of integers, the ring of reals, and the ring of natural numbers modulo m (for some fixed natural number m). Furthermore we use the results obtained in the section to carry out the second step of the verification of the convolution circuit.

We need the notion of normal forms for ring expressions.

5.6.1 Normal Forms for Ring Expressions

We introduce the notions of sums of products and normal expressions, and then show that for each ring expression there is a normal expression which is equal to it over each ring R.

A *sum of products* is a ring expression of the form:

$$\sum_{i_1=0}^{\ell_1(\overline{x})} \sum_{i_2=0}^{\ell_2(\overline{x},i_1)} \cdots \sum_{i_m=0}^{\ell_m(\overline{x},i_1,\ldots,i_{m-1})} a_1(\overline{q_1}(\overline{x},i_1,\ldots,i_m)) \cdot \cdots \cdot a_n(\overline{q_n}(\overline{x},i_1,\ldots,i_m))$$

Here m is called the *depth* of the sum of products, and $a_1(\overline{q_1}(\overline{x}, i_1, \ldots, i_m)) \cdot \cdots \cdot a_n(\overline{q_n}(\overline{x}, i_1, \ldots, i_m))$ the *stream expression part* of the sum of products.

A *normal ring expression* is a ring expression of the form $S_1(\overline{x}) + \cdots + S_k(\overline{x})$ where $S_1(\overline{x}), \ldots, S_k(\overline{x})$ are sums of products.

Lemma 17 *For each ring expression $r(\overline{x})$, there is a normal expression $N(\overline{x})$ such that $r(\overline{x}) = N(\overline{x})$ over each ring R.*

Proof: The proof can be carried out easily by structural induction on ring expressions, and rewriting by distributive laws as:

$$\begin{aligned}
r_1(\overline{x}) \cdot (r_2(\overline{x}) + r_3(\overline{x})) &= r_1(\overline{x}) \cdot r_2(\overline{x}) + r_1(\overline{x}) \cdot r_3(\overline{x}) \\
\left(\sum_{i=0}^{\ell(\overline{x})} r_1(\overline{x}, i)\right) \cdot r_2(\overline{x}) &= \sum_{i=0}^{\ell(\overline{x})} r_1(\overline{x}, i) \cdot r_2(\overline{x}) \\
\sum_{i=0}^{\ell(\overline{x})} r_1(\overline{x}, i) + r_2(\overline{x}, i) &= \sum_{i=0}^{\ell(\overline{x})} r_1(\overline{x}, i) + \sum_{i=0}^{\ell(\overline{x})} r_2(\overline{x}, i) \\
\left(\sum_{i=0}^{\ell_1(\overline{x})} r_1(\overline{x}, i)\right) \cdot \left(\sum_{j=0}^{\ell_2(\overline{x})} r_2(\overline{x}, j)\right) &= \sum_{i=0}^{\ell_1(\overline{x})} \sum_{j=0}^{\ell_2(\overline{x})} r_1(\overline{x}, i) \cdot r_2(\overline{x}, j)
\end{aligned}$$

□

From now on we consider only normal expressions.

5.6.2 Ring Conditional equalities

A sum of products is *bilinear* if its stream expression part is of the form:

$$a_1(q_{11}(\overline{x}, \overline{i}), \ldots, q_{1m_1}(\overline{x}, \overline{i})) \cdot \cdots \cdot a_n(q_{n1}(\overline{x}, \overline{i}), \ldots, q_{nm_n}(\overline{x}, \overline{i}))$$

and for each i and j where $1 \leq i \neq j \leq n$, either a_i is not the same stream variable as a_j, or the elements of $\overline{x}$ and $\overline{i}$ have the same coefficients in $q_{ik}(\overline{x}, \overline{i})$ as in $q_{jk}(\overline{x}, \overline{i})$, for $1 \leq k \leq m_i = m_j$, i.e. there are integers α_k such that:

$$q_{ik}(\overline{x}, \overline{i}) = q_{jk}(\overline{x}, \overline{i}) + \alpha_k$$

for $1 \leq k \leq m_i = m_j$. A normal expression is called *bilinear* if all its sums of products are bilinear. A linear guarded ring expression is called *bilinear* if all its results are bilinear.

A *zero-equivalence formula* is of the form $p(\overline{x}) \longrightarrow (r(\overline{x}) = 0)$, where $p(\overline{x})$ is a conjunction of linear predicates and $r(\overline{x})$ is a bilinear normal expression which is well-defined under $p(\overline{x})$.

A *ring conditional equality* is of the form $p(\overline{x}) \longrightarrow (e(\overline{x}) = r(\overline{x}))$, where $p(\overline{x})$ is a conjunction of linear predicates, $e(\overline{x})$ is a linear guarded ring expression which is bilinear, and $r(\overline{x})$ is a bilinear normal expression well-defined under $p(\overline{x})$.

In section 5.6.3 we define a measure of *complexity* on zero-equivalence formulas. In section 5.6.5 we show that for each zero-equivalence formula $P(\overline{x})$, there is a finite conjunction of "simpler" zero-equivalence formulas, which is equivalent to $P(\overline{x})$ over each ring R.

5.6.3 Complexity

We define a measure of *complexity* for zero-equivalence formulas. In order to do that we need the notions of *matching relation* and *block*.

Two sums of products are said to be *matching* if their stream expression parts are of the forms:

$$a_1(q_{11}(\overline{x},\overline{i}),\ldots,q_{1m_1}(\overline{x},\overline{i}))\cdot\cdots\cdot a_n(q_{n1}(\overline{x},\overline{i}),\ldots,q_{nm_n}(\overline{x},\overline{i}))$$

and:

$$a_1(q'_{11}(\overline{x},\overline{i}),\ldots,q'_{1m_1}(\overline{x},\overline{i}))\cdot\cdots\cdot a_n(q'_{n1}(\overline{x},\overline{i}),\ldots,q'_{nm_n}(\overline{x},\overline{i}))$$

respectively, and the elements of $\overline{x}$ have the same coefficients in $q_{jk}(\overline{x},\overline{i})$ as in $q'_{jk}(\overline{x},\overline{i})$, i.e. there are linear polynomials $q''_{jk}(\overline{i})$, for $1\leq j\leq n$ and $1\leq k\leq m_j$, such that:

$$q_{jk}(\overline{x},\overline{i}) \;=\; q'_{jk}(\overline{x},\overline{i})+q''_{jk}(\overline{i})$$

It is clear that the matching relation is an equivalence relation among sums of products. Two normal expressions, of the forms $S_{11}(\overline{x})+\cdots+S_{1m}(\overline{x})$ and $S_{21}(\overline{x})+\cdots+S_{2m}(\overline{x})$ respectively, are said to be matching if $S_{1i}(\overline{x})$ and $S_{2i}(\overline{x})$ are matching for $1\leq i\leq m$.

Let $N(\overline{x})$ be a normal expression of the form $S_1(\overline{x})+\cdots+S_m(\overline{x})$. Let the equivalence classes defined by the set $\{S_1(\overline{x}),\ldots,S_m(\overline{x})\}$ and the matching relation be $\{S_{11}(\overline{x}),\ldots,S_{1m_1}(\overline{x})\},\ldots,\{S_{n1}(\overline{x}),\ldots,S_{nm_n}(\overline{x})\}$. Let $B_i(\overline{x})$ be the normal expression $S_{i1}(\overline{x})+\cdots+S_{im_i}(\overline{x})$. Then $B_i(\overline{x})$ is called a *block* of $N(\overline{x})$, while $B_1(\overline{x})+\cdots+B_n(\overline{x})$ is called the *block form* of $N(\overline{x})$.

The *complexity* of a sum of products is defined to be the pair $<d,\#x>$, where d is the depth of the sum of products, and $\#x$ is the number of free variables in the sum of products. The *complexity* of a block in a normal expression is defined to be the maximum of the complexities of the sums of products in the block. The *complexity* of a normal expression is defined to be a pair $<c_1,c_2>$, where c_1 is the complexity of a block with maximum complexity in the normal expression, and c_2 is the number of blocks with maximum complexity in the normal expression. The *complexity* of a conjunction of linear predicates is defined to be the number of free variables in the conjunction of linear predicates. The *complexity* of a zero-equivalence formula of the form $p(\overline{x})\rightarrow(r(\overline{x})=0)$ is defined to be the pair $<c_1,c_2>$, where c_1 is the complexity of $r(\overline{x})$, and c_2 is the complexity of $p(\overline{x})$.

A sum of products is called *simple* if it has a complexity of $<0,0>$. Note that if a sum of products is simple then it does not contain any free integer variables. A normal expression is called *simple* if all the sums of products in it are simple, i.e. it has a complexity of the form $<<0,0>,c>$. A zero-equivalence formula is called *simple* if its normal expression is simple, i.e. it has a complexity of the form $<<<0,0>,c_1>,c_2>$.

The complexity of a normal ring expression $r(\overline{x})$ is denoted by $\mathcal{C}(r(\overline{x}))$. The same applies to linear predicates and zero-equivalence formulas.

5.6.4 Integer Linear Programming

The integer linear programming problem is the following: *Given a conjunction $p(\overline{x})$ of linear equalities, is there a nonnegative value $\overline{\alpha}$ of $\overline{x}$ such that $p(\overline{\alpha})$ is true ?*. In [BT76]

the integer linear programming problem is shown to be decidable. This is done by finding a bound $\overline{\gamma}$ which is derived from $p(\overline{x})$, such that if there is a nonnegative $\overline{\alpha}$ for which $p(\overline{\alpha})$ is true then there is a nonegative $\overline{\beta} \leq \overline{\gamma}$, such that $p(\overline{\beta})$ is true.

5.6.5 Deciding the Validity of Zero-equivalence Formulas

In this section we show that the validity of zero-equivalence formulas over a class of rings $\mathcal{K}$ can be reduced to the zero-equivalence of polynomials over $\mathcal{K}$. This is achieved by lemma 24. For the proof of the lemma we need the definition of the *shifting operation* and some auxiliary lemmas.

Let $r(\overline{x})$ be a ring expression. The result of *shifting* the stream variable a by $\overline{qs}(\overline{x})$ in $r(\overline{x})$ is the ring expression $rs(\overline{x})$ we get from $r(\overline{x})$ by replacing each stream expression (see section 5.1) of the form $a(\overline{q}(\overline{x}, \overline{i}))$, by the stream expression $a(\overline{q}(\overline{x}, \overline{i}) + \overline{qs}(\overline{x}))$. Here $rs(\overline{x})$ is called a *shift* of $r(\overline{x})$.

The result of shifting the variable x by α in $r(\overline{x})$ is the ring expression we get from $r(\overline{x})$ by replacing each occurence of x in each stream expression in $r(\overline{x})$ by $x + \alpha$.

Lemma 18 *Let $r(\overline{x})$ be a ring expression. Let $rs(\overline{x})$ be a shift of $r(\overline{x})$. Then for each value $\overline{\alpha}$ of $\overline{x}$, $r(\overline{\alpha}) = 0$ iff $rs(\overline{\alpha}) = 0$ over each ring R.*

Proof: Let R be an arbitrary ring. First we show that for each value $\overline{\alpha}$ of $\overline{x}$, it is true that for each stream interpretation $\mathcal{I}$ over R, there is another stream interpretation $\mathcal{I}'$ over R, such that $[\![r(\overline{\alpha})]\!]_{R,\mathcal{I}} = [\![rs(\overline{\alpha})]\!]_{R,\mathcal{I}'}$.

Let $rs(\overline{x})$ be the result of shifting a by $qs(\overline{x})$ in $r(\overline{x})$. For each stream interpretation $\mathcal{I}$ over R, we define the stream interpretation $\mathcal{I}'$ over R, such that for each $\overline{\beta}$:

$$a^{\mathcal{I}'}(\overline{\beta} + \overline{qs}(\overline{\alpha})) \;=\; a^{\mathcal{I}}(\overline{\beta})$$

and:

$$b^{\mathcal{I}'}(\overline{\beta}) \;=\; b^{\mathcal{I}}(\overline{\beta}) \quad \text{if } b \neq a$$

It is clear that $\mathcal{I}'$ exists and that $[\![r(\overline{\alpha})]\!]_{R,\mathcal{I}} = [\![rs(\overline{\alpha})]\!]_{R,\mathcal{I}'}$.

Now suppose that $rs(\overline{\alpha}) = 0$ over R. It follows that for each stream interpretation $\mathcal{I}'$ over R, $[\![rs(\overline{\alpha})]\!]_{R,\mathcal{I}'} = 0$. Suppose that $r(\overline{\alpha}) \neq 0$ over R. Then there is a stream interpretation $\mathcal{I}$ over R, such that $[\![r(\overline{\alpha})]\!]_{R,\mathcal{I}} \neq 0$. This implies that there is a stream interpretation $\mathcal{I}'$ over R such that $[\![rs(\overline{\alpha})]\!]_{R,\mathcal{I}'} \neq 0$, which is a contradiction. It follows that if $rs(\overline{\alpha}) = 0$ over R then $r(\overline{\alpha}) = 0$ over R.

In a similar manner we can show that if $r(\overline{\alpha}) = 0$ over R then $rs(\overline{\alpha}) = 0$ over R.

As R was chosen arbitrarily, the result holds for each ring R. □

Lemma 19 *Let:*

- *$S(\overline{x})$ be a bilinear sum of products, which is not simple.*
- *α be an integer.*
- *$S'(\overline{x}) = S(\overline{x})[x_i \leftarrow x_i + \alpha]$.*

Then there are bilinear sums of products $S^*(\overline{x}), S_1(\overline{x}), \ldots, S_m(\overline{x})$, such that:

- $S^*(\overline{x})$ *is the result of shifting* x_i *by* α *in* $S(\overline{x})$.
- $\mathcal{C}(S_i(\overline{x})) < \mathcal{C}(S(\overline{x}))$ *for* $1 \leq i \leq m$.
- *For each value* $\overline{\beta}$ *of* $\overline{x}$ *if both* $S'(\overline{\beta})$ *and* $S(\overline{\beta})$ *are well-defined then* $S'(\overline{\beta}) = S^*(\overline{\beta}) + S_1(\overline{\beta}) + \cdots + S_m(\overline{\beta})$ *over each ring* R.

Proof: The proof can be carried out using induction on the complexity of sums of products and expanding expressions of the form $\sum_{i=0}^{\ell(\overline{x})+\gamma} r(\overline{x}, i)$ to $\sum_{i=0}^{\ell(\overline{x})} r(\overline{x}, i) + r(\overline{x}, \ell(\overline{x}) + 1) + \cdots + r(\overline{x}, \ell(\overline{x}) + \gamma)$ if γ is a positive integer, and to $\sum_{i=0}^{\ell(\overline{x})} r(\overline{x}, i) - r(\overline{x}, \ell(\overline{x})) - r(\overline{x}, \ell(\overline{x}) - 1) - \cdots - r(\overline{x}, \ell(\overline{x}) + \gamma + 1)$ if γ is a negative integer. □

Remarks on lemma 19

1. Observe that $S(\overline{x})$ and $S^*(\overline{x})$ are matching, and that $\mathcal{C}(S^*(\overline{x})) = \mathcal{C}(S(\overline{x}))$.
2. Let $N(\overline{x}) = S_1(\overline{x}) + \cdots + S_m(\overline{x})$, then $S'(\overline{x}) = S^*(\overline{x}) + N(\overline{x})$. Observe that $N(\overline{x})$ is a normal expression and that $\mathcal{C}(N(\overline{x})) < \mathcal{C}(S(\overline{x}))$.
3. If the stream expression part of $S(\overline{x})$ does not contain x_i then $S^*(\overline{x}) = S(\overline{x})$.

Lemma 20 *Let:*

- $r(\overline{x})$ *be a bilinear normal expression, which is not simple.*
- α *be an integer.*
- $r'(\overline{x}) = r(\overline{x})[x_i \leftarrow x_i + \alpha]$.

then there is a bilinear normal expression $r^*(\overline{x})$ *such that:*

- $\mathcal{C}(r^*(\overline{x})) < \mathcal{C}(r(\overline{x}))$
- *for each value* $\overline{\beta}$ *of* $\overline{x}$ *if both* $r'(\overline{\beta})$ *and* $r(\overline{\beta})$ *are well-defined, then for each ring* R *if* $r(\overline{\beta}) = 0$, *then* $r'(\overline{\beta}) = 0$ *iff* $r^*(\overline{\beta}) = 0$.

We call the ring expression $r^*(\overline{x})$ the result of rewriting $r'(\overline{x})$ by $r(\overline{x}) = 0$.
Proof: Let R be an arbitrary ring. Let the block form of $r(\overline{x})$ be:

$$B_1(\overline{x}) + \cdots + B_m(\overline{x})$$

Without loss of generality let $B_1(\overline{x})$ be a block with highest complexity in $r(\overline{x})$.

Let $a_1, \ldots, a_m$ be the stream variables which occur in $B_1(\overline{x})$. We know that $B_1(\overline{x})$ is bilinear and that all the sums of products in $B_1(\overline{x})$ are matching. It follows that there are polynomials $\overline{q}_1(x_i), \ldots, \overline{q}_m(x_i)$, such that the result of shifting $a_1, \ldots, a_m$ by $\overline{q}_1(x_i), \ldots, \overline{q}_m(x_i)$ respectively in $B_1(\overline{x})$ yields a block $Bs_1(\overline{x})$ in which the following is true: x_i does not occur in the stream expression parts of the sums of products in $Bs_1(\overline{x})$.

Let $rs(\overline{x})$ be the result of shifting $a_1, \ldots, a_m$ by $\overline{q}_1(x_i), \ldots, \overline{q}_m(x_i)$ respectively in $r(\overline{x})$, then:

$$rs(\overline{x}) \quad = \quad Bs_1(\overline{x}) + \cdots + Bs_m(\overline{x})$$

where $Bs_j(\overline{x})$ is the result of shifting $a_1, \ldots, a_m$ by $\overline{q}_1(x_i), \ldots, \overline{q}_m(x_i)$ respectively in $B_j(\overline{x})$.

Also we note that:

$$r'(\overline{x}) = B'_1(\overline{x}) + \cdots + B'_m(\overline{x})$$

where $B'_j(\overline{x}) = B_j(\overline{x})[x_i \leftarrow x_i + \alpha]$. Let $rs'(\overline{x})$ be the result of shifting $a_1, \ldots, a_m$ by $\overline{q}_1(x_i), \ldots, \overline{q}_m(x_i)$ respectively in $r'(\overline{x})$ then:

$$rs'(\overline{x}) = Bs'_1(\overline{x}) + \cdots + Bs'_m(\overline{x}) \tag{5.31}$$

Observe that the stream expression parts of $Bs_1(\overline{x})$ and $Bs'_1(\overline{x})$ do not contain x_i.

From the remarks below lemma 19 it follows that:

$$Bs'_j(\overline{x}) = Bs^*_j(\overline{x}) + N_j(\overline{x}) \tag{5.32}$$

over R, for $1 \leq j \leq m$, where $Bs^*_j(\overline{x})$ is the result of shifting x_i in $Bs_j(\overline{x})$ by α. Also we know from the fact that the stream expression part of $Bs_1(\overline{x})$ does not contain x_i, that:

$$Bs^*_1(\overline{x}) = Bs_1(\overline{x}) \tag{5.33}$$

From (5.31), (5.32), and (5.33) it follows that:

$$rs'(\overline{x}) = Bs_1(\overline{x}) + N_1(\overline{x}) + Bs^*_2(\overline{x}) + N_2(\overline{x}) + \cdots + Bs^*_m(\overline{x}) + N_m(\overline{x}) \tag{5.34}$$

over R. Now suppose that $r(\overline{\beta}) = 0$ for a certain $\overline{\beta}$. We know by lemma 18 that $rs(\overline{\beta}) = 0$, so:

$$Bs_1(\overline{\beta}) + \cdots + Bs_m(\overline{\beta}) = 0$$

over R, and:

$$Bs_1(\overline{\beta}) = -Bs_2(\overline{\beta}) - \cdots - Bs_m(\overline{\beta}) \tag{5.35}$$

over R. From (5.34) and (5.35) we get:

$$\begin{aligned} rs'(\overline{\beta}) = \\ & Bs^*_2(\overline{\beta}) - Bs_2(\overline{\beta}) + \cdots + Bs^*_m(\overline{\beta}) - Bs_m(\overline{\beta}) + \\ & N_1(\overline{\beta}) + \cdots + N_m(\overline{\beta}) \end{aligned}$$

over R. Let $r^*(\overline{x})$ be:

$$\begin{aligned} & Bs^*_2(\overline{x}) - Bs_2(\overline{x}) + \cdots + Bs^*_m(\overline{x}) - Bs_m(\overline{x}) + \\ & N_1(\overline{x}) + \cdots + N_m(\overline{x}) \end{aligned}$$

It is clear that $r^*(\overline{x})$ is a bilinear normal expression. By lemma 18 it follows that $r'(\overline{\beta}) = 0$ iff $rs'(\overline{\beta}) = 0$ iff $r^*(\overline{\beta}) = 0$ over R, so $r^*(\overline{x})$ fulfills the second condition stated in the lemma. We must also show the first condition i.e. $\mathcal{C}(r^*(\overline{x})) < \mathcal{C}(r(\overline{x}))$. From the remarks below lemma 19 we know that $Bs_j(\overline{x})$ and $Bs^*_j(\overline{x})$ belong to the same block in $r(\overline{x})$, and that:

$$\mathcal{C}(Bs^*_j(\overline{x})) = \mathcal{C}(Bs_j(\overline{x}))$$

and:

$$\mathcal{C}(N_j(\overline{x})) \;<\; \mathcal{C}(Bs_j(\overline{x}))$$

As we have chosen $B_1(\overline{x})$ to be a block with highest complexity, it follows that:

$$\mathcal{C}(r^*(\overline{x})) \;<\; \mathcal{C}(r(\overline{x}))$$

As R was chosen arbitrarily, the result holds for each ring R. □

Corollary 21 *Consider a conjunction $p(\overline{x})$ of linear predicates, and a ring expression $r(\overline{x})$ which is well-defined under $p(\overline{x})$. Let α be an integer. Let $p'(\overline{x}) = p(\overline{x})[x_i \leftarrow x_i + \alpha]$, and $r'(\overline{x}) = r(\overline{x})[x_i \leftarrow x_i + \alpha]$. Let $r^*(\overline{x})$ be the result of rewriting $r'(\overline{x})$ by $r(\overline{x}) = 0$, then for each ring R:*

$$(\forall \overline{x}.\ p(\overline{x}) \longrightarrow (r(\overline{x}) = 0)) \;\longrightarrow\; (\forall \overline{x}.\ p(\overline{x}) \wedge p'(\overline{x}) \longrightarrow (r^*(\overline{x}) = 0))$$

Proof: Let R be an arbitrary ring. Suppose that $\forall \overline{x}.\ p(\overline{x}) \longrightarrow (r(\overline{x}) = 0)$ over R. Suppose that for a certain value $\overline{\beta}$ of $\overline{x}$ we have $p(\overline{\beta})$ and $p'(\overline{\beta})$ true. Then $r(\overline{\beta}) = 0$ and $r'(\overline{\beta}) = 0$ over R. By lemma 20 it follows that $r^*(\overline{\beta}) = 0$ over R. □

Lemma 22 *Suppose that $P(\overline{x})$ is a zero-equivalence formula, which is not simple. Then there is a finite set of zero-equivalence formulas such that each element of the set is less complex than $P(\overline{x})$, and such that $P(\overline{x})$ is valid over a ring R iff each of the elements of the set is valid over R.*

We call the set of less complex formulas the *decomposition set* of $P(\overline{x})$. The lemma can be applied recursively (as will be explained in lemma 24) to get a set of simple zero-equivalence formulas.

Proof: Let R be an arbitrary ring. The two cases A and B below are possible. In each case we will give a finite set Z of zero-equivalence formulas which are less complex than $P(\overline{x})$, and whose conjunction is equivalent to $P(\overline{x})$ over R.

A. If the conditional part of $P(\overline{x})$ contains an equality $leq(\overline{x})$. In this case $P(\overline{x})$ will be of the form:

$$leq(\overline{x}) \wedge p(\overline{x}) \;\longrightarrow\; (r(\overline{x}) = 0)$$

We will use $leq(\overline{x})$ to eliminate a free variable in $P(\overline{x})$, reducing it to a less complex zero-equivalence formula which is equivalent over R. Let x_i be a free variable in $leq(\overline{x})$ chosen in the following way:

- If all the free variables of $leq(\overline{x})$ are also free in $r(\overline{x})$, then let x_i be any free variable in $leq(\overline{x})$.
- If there is a variable which is free in $leq(\overline{x})$, but not in $r(\overline{x})$, then let x_i be any such a variable.

Let $leq(\overline{x})$ be of the form $\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta = 0$, let:

$$\begin{array}{rcl} \ell(\overline{x}) & = & \alpha_1 x_1 + \cdots + \alpha_{i-1} x_{i-1} + \alpha_{i+1} x_{i+1} + \ldots + \alpha_n x_n + \beta \\ p'(\overline{x}) & = & p(\overline{x}) \left[x_i \leftarrow -\frac{\ell(\overline{x})}{\alpha_i} \right] \quad r'(\overline{x}) = r(\overline{x}) \left[x_i \leftarrow -\frac{\ell(\overline{x})}{\alpha_i} \right] \end{array}$$

Let:

$$P_1(\overline{x}) \;=\; p'(\overline{x}) \wedge (\ell(\overline{x}) \; mod \; \alpha_i = 0) \longrightarrow (r'(\overline{x}) = 0)$$

We define Z to be the one element set $\{P_1(\overline{x})\}$. It is obvious that $P(\overline{x})$ iff $P_1(\overline{x})$ over R. Also the manner in which x_i is chosen above implies that either the number of free variables in $r(\overline{x})$ is reduced, or the number of free variables in $p(\overline{x})$ is reduced while the number of free variables in $r(\overline{x})$ is not changed. In both cases it follows that:

$$\mathcal{C}(P_1(\overline{x})) \;<\; \mathcal{C}(P(\overline{x}))$$

B. If the conditional part of $P(\overline{x})$ does not contain any equalities. Let $P(\overline{x})$ be of the form $p(\overline{x}) \longrightarrow (r(\overline{x}) = 0)$. Then $p(\overline{x})$ will be of the form $p_1(\overline{x}) \wedge p_2(\overline{x})$ where $p_1(\overline{x})$ is a conjunction of linear inequalities and $p_2(\overline{x})$ a conjunction of linear modulo predicates. Let x_i be any free variable in $P(\overline{x})$. Let α be the least common multiple of the moduli of the modulo predicates in $p_2(\overline{x})$. Observe that

$$p_2(\overline{x}) \;\text{ iff }\; p_2(\overline{x})[x_i \leftarrow x_i + \alpha] \;\text{ iff }\; p_2(\overline{x})[x_i \leftarrow x_i - \alpha]$$

By induction:

$$\forall \overline{x}.\ P(\overline{x})$$

iff

$$\begin{array}{c} \wedge_{j=0}^{\alpha-1} \forall \overline{x}.\ P(\overline{x})[x_i \leftarrow j] \\ \wedge \\ \forall \overline{x}.\ P(\overline{x}) \longrightarrow (P(\overline{x})[x_i \leftarrow x_i + \alpha]) \\ \wedge \\ \forall \overline{x}.\ P(\overline{x}) \longrightarrow (P(\overline{x})[x_i \leftarrow x_i - \alpha]) \end{array}$$

iff

$$\begin{array}{c} \underbrace{\wedge_{j=0}^{\alpha-1} \forall \overline{x}.\ P(\overline{x})[x_i \leftarrow j]}_{P_1(\overline{x})} \\ \wedge \\ \forall \overline{x}.\ \underbrace{\neg p_1(\overline{x}) \ \wedge\ P(\overline{x}) \longrightarrow (P(\overline{x})[x_i \leftarrow x_i + \alpha])}_{P_2(\overline{x})} \\ \wedge \\ \forall \overline{x}.\ \underbrace{p_1(\overline{x}) \ \wedge\ P(\overline{x}) \longrightarrow (P(\overline{x})[x_i \leftarrow x_i + \alpha])}_{P_3(\overline{x})} \\ \wedge \\ \forall \overline{x}.\ \underbrace{\neg p_1(\overline{x}) \ \wedge\ P(\overline{x}) \longrightarrow (P(\overline{x})[x_i \leftarrow x_i - \alpha])}_{P_4(\overline{x})} \\ \wedge \\ \forall \overline{x}.\ \underbrace{p_1(\overline{x}) \ \wedge\ P(\overline{x}) \longrightarrow (P(\overline{x})[x_i \leftarrow x_i - \alpha])}_{P_5(\overline{x})} \end{array} \tag{5.36}$$

We will give five sets Z_1, Z_2, Z_3, Z_4, and Z_5 of zero-equivalence formulas, such that each element in Z_i is less complex than $P(\overline{x})$, and such that $P_i(\overline{x})$ is equivalent to the conjunction of the elements of Z_i over R, for $i = 1, 2, 3, 4, 5$. The set Z will be equal to $Z_1 \cup Z_2 \cup Z_3 \cup Z_4 \cup Z_5$.

1. We define Z_1 to be the set $\{P(\overline{x})[x_i \leftarrow j];\ 0 \leq j \leq \alpha - 1\}$. We can prove that each element of Z_1 is less complex than $P(\overline{x})$ in a similar manner to that of case A above.

2. Let:

$$\begin{array}{lll} p_1'(\overline{x}) = p_1(\overline{x})[x_i \leftarrow x_i + \alpha] & & p_2'(\overline{x}) = p_2(\overline{x})[x_i \leftarrow x_i + \alpha] \\ r'(\overline{x}) = r(\overline{x})[x_i \leftarrow x_i + \alpha] & & \end{array}$$

We observe that:

$$\begin{array}{c} \forall \overline{x}.\ P_2(\overline{x}) \\ \text{iff} \\ \forall \overline{x}.\ \neg p_1(\overline{x}) \ \wedge\ (p_1(\overline{x}) \ \wedge\ p_2(\overline{x}) \longrightarrow (r(\overline{x}) = 0)) \ \wedge\ p_1'(\overline{x}) \ \wedge\ p_2'(\overline{x}) \longrightarrow (r'(\overline{x}) = 0) \\ \text{iff} \\ \forall \overline{x}.\ \neg p_1(\overline{x}) \ \wedge\ p_1'(\overline{x}) \ \wedge\ p_2'(\overline{x}) \longrightarrow (r'(\overline{x}) = 0) \end{array}$$

over R. Furthermore it can be shown that there are formulas $p_{11}(\overline{x}), \ldots, p_{1k}(\overline{x}), p_{21}(\overline{x}), \ldots, p_{2k}(\overline{x})$, where $p_{1j}(\overline{x})$ is a conjunction of linear inequalities and $p_{2j}(\overline{x})$ is a linear equality, such that:

$$\neg p_1(\overline{x}) \wedge p_1'(\overline{x}) = (p_{11}(\overline{x}) \wedge p_{21}(\overline{x})) \vee \cdots \vee (p_{1k}(\overline{x}) \wedge p_{2k}(\overline{x}))$$

which means that:

$$\forall \overline{x}.\ P_2(\overline{x}) \quad \text{iff} \quad \left(\begin{array}{c} \forall \overline{x}.\ p_{11}(\overline{x}) \wedge p_{21}(\overline{x}) \wedge p_2'(\overline{x}) \longrightarrow (r'(\overline{x}) = 0) \\ \wedge \\ \vdots \\ \wedge \\ \forall \overline{x}.\ p_{1k}(\overline{x}) \wedge p_{2k}(\overline{x}) \wedge p_2'(\overline{x}) \longrightarrow (r'(\overline{x}) = 0) \end{array} \right)$$

over R. Let:

$$P_{2j}(\overline{x}) = p_{1j}(\overline{x}) \wedge p_{2j}(\overline{x}) \wedge p_2'(\overline{x}) \longrightarrow (r'(\overline{x}) = 0)$$

The equality $p_{2j}(\overline{x})$ can be used, in a similar manner to that in case A, to generate a zero-equivalence formula $P_{2j}'(\overline{x})$ which is equivalent to and less complex than $P_{2j}(\overline{x})$. Now Z_2 can be defined to be $\{P_{2j}'(\overline{x}); 1 \leq j \leq k\}$.

3. Let:

$$\begin{array}{lll} p_1'(\overline{x}) = p_1(\overline{x})[x_i \leftarrow x_i + \alpha] & & p_2'(\overline{x}) = p_2(\overline{x})[x_i \leftarrow x_i + \alpha] \\ r'(\overline{x}) = r(\overline{x})[x_i \leftarrow x_i + \alpha] & & \end{array}$$

We observe that:

$$\begin{array}{c} \forall \overline{x}.\ P_3(\overline{x}) \\ \text{iff} \\ \forall \overline{x}.\ p_1(\overline{x}) \wedge (p_1(\overline{x}) \wedge p_2(\overline{x}) \longrightarrow (r(\overline{x}) = 0)) \wedge p_1'(\overline{x}) \wedge p_2'(\overline{x}) \longrightarrow (r'(\overline{x}) = 0) \\ \text{iff} \\ \forall \overline{x}.\ p_1(\overline{x}) \wedge p_1'(\overline{x}) \wedge p_2(\overline{x}) \wedge (r(\overline{x}) = 0)) \longrightarrow (r'(\overline{x}) = 0) \end{array}$$

over R. From lemma 20 it follows that:

$$\forall \overline{x}.\ P_3(\overline{x}) \quad \text{iff} \quad \forall \overline{x}.\ p_1(\overline{x}) \wedge p_1'(\overline{x}) \wedge p_2(\overline{x}) \wedge (r(\overline{x}) = 0) \longrightarrow (r^*(\overline{x}) = 0)$$

over R, where $r^*(\overline{x})$ is the result of rewriting $r'(\overline{x})$ by $r(\overline{x}) = 0$ (see lemma 20). From corollary 21 it follows that:

$$\forall \overline{x}.\ P(\overline{x}) \quad \text{iff} \quad (\forall \overline{x}.\ P(\overline{x})) \wedge (\forall \overline{x}.\ p_1(\overline{x}) \wedge p_1'(\overline{x}) \wedge p_2(\overline{x}) \longrightarrow (r^*(\overline{x}) = 0))$$

over R. This means that $P_3(\overline{x})$ in (5.36) can be replaced by $P_{31}(\overline{x})$ without changing the validity of the formula over R, where:

$$P_{31}(\overline{x}) = \forall \overline{x}.\ p_1(\overline{x}) \wedge p_1'(\overline{x}) \wedge p_2(\overline{x}) \longrightarrow (r^*(\overline{x}) = 0)$$

Let Z_3 be the set $\{P_{31}(\overline{x})\}$. By lemma 20 we know that:

$$\mathcal{C}(r^*(\overline{x})) < \mathcal{C}(r(\overline{x}))$$

Consequently:

$$\mathcal{C}(P_{31}(\overline{x})) \;<\; \mathcal{C}(P(\overline{x}))$$

4. Similar to 2.

5. Similar to 3.

We observe that the construction of the decomposition set is not dependent on the particular ring R which was chosen arbitrarily, and hence the result holds for each ring R. □

Lemma 23 *Let r be a ring expression which does not contain any free integer variables. Then there is a ring polynomial Q such that $r = 0$ iff $Q = 0$ over each ring R.*

Proof: Let R be an arbitrary ring. As r does not contain any free integer variables then it is of the form[4]:

$$\sum_{i=1}^{m} \gamma_i \cdot a_1^{\beta_{i1}}(\overline{\alpha}_1) \cdot \cdots \cdot a_n^{\beta_{in}}(\overline{\alpha}_n)$$

where m is a positive integer, γ_i is an integer, β_{ij} is a nonnegative integer, a_i is a stream variable, and $\overline{\alpha}_i$ is a tuple of integers. In addition, for each $1 \leq i \leq m$ there is a $1 \leq j \leq n$ such that $\beta_{ij} > 0$, and for each $1 \leq i \neq j \leq n$ either $a_i \neq a_j$ or $\overline{\alpha}_i \neq \overline{\alpha}_j$. We define a ring polynomial Q:

$$Q \;=\; \sum_{i=1}^{m} \gamma_i \cdot v_1^{\beta_{i1}} \cdot \cdots \cdot v_n^{\beta_{in}}$$

where v_i is a variable over R. It can easily be checked that $r = 0$ iff $Q = 0$ over R. As R was chosen arbitrarily, the result holds for each ring R. □

Lemma 24 *Let $\mathcal{K}$ be a class of rings. Suppose that the zero-equivalence of a ring polynomial over $\mathcal{K}$ is decidable[5]. Then the validity of zero-equivalence formulas over $\mathcal{K}$ is decidable.*

Proof: Let $P(\overline{x})$ be any zero-equivalence formula. We compute the decomposition set of $P(\overline{x})$ (see lemma 22), and repeat the procedure recursively on the elements of the set until all the elements of the set are simple. Thus we get a finite set $\{P_1(\overline{x}), \ldots, P_m(\overline{x})\}$ of simple zero-equivalence formulas such that:

$$P(\overline{x})$$
$$\text{iff}$$
$$P_1(\overline{x}) \wedge \cdots \wedge P_m(\overline{x})$$

[4]By $\gamma \cdot x$ we mean $\underbrace{x + \cdots + x}_{\gamma}$, and by x^{β} we mean $\underbrace{x \cdot \cdots \cdot x}_{\beta}$.

[5]We say that the zero-equivalence of a ring polynomial, over a class $\mathcal{K}$ of rings, is decidable if for each ring polynomial Q (see section 5.1) it is decidable whether $Q = 0$ for all rings $R \in \mathcal{K}$

over each ring R. Let $P_i(\overline{x})$ be of the form $p_i(\overline{x}) \rightarrow (r_i = 0)$ (see the definition of simple zero-equivalence formulas in section 5.6.3). This means that:

$$\begin{array}{c} P(\overline{x}) \\ \text{iff} \\ (p_1(\overline{x}) \longrightarrow (r_1 = 0)) \wedge \cdots \wedge (p_m(\overline{x}) \longrightarrow (r_m = 0)) \end{array} \tag{5.37}$$

over each ring R. As r_i does not contain any free integer variables then it follows from lemma 23 that there is a ring polynomial Q_i such that:

$$(r_i = 0) \quad \text{iff} \quad (Q_i = 0) \tag{5.38}$$

over each ring R. From (5.37) and (5.38) we get:

$$\begin{array}{c} P(\overline{x}) \\ \text{iff} \\ (p_1(\overline{x}) \longrightarrow (Q_1 = 0)) \wedge \cdots \wedge (p_m(\overline{x}) \longrightarrow (Q_m = 0)) \end{array} \tag{5.39}$$

over each ring R.

Now, let $\mathcal{K}$ be a class of rings. From (5.39) it follows that the validity of $P(\overline{x})$ over $\mathcal{K}$ is equivalent to the validity of $p_i(\overline{x}) \rightarrow (Q_i = 0)$ over $\mathcal{K}$, for $1 \leq i \leq m$. It is clear that the validity of $p_i(\overline{x}) \rightarrow (Q_i = 0)$ over $\mathcal{K}$ is equivalent to the unsatisfiability of $p_i(\overline{x})$ over I, or the validity of $Q_i = 0$ over $\mathcal{K}$. It can easily be shown that the unsatisfiability of $p_i(\overline{x})$ over I can be transformed to the integer linear programming problem which is decidable (see section 5.6.4). Also we have assumed that the validity of $Q_i = 0$ over $\mathcal{K}$ is decidable. It follows that the validity of $P(\overline{x})$ over $\mathcal{K}$ is decidable. □

5.6.6 Deciding the Validity of Ring Conditional Equalities

We will show that the problem of deciding the validity of ring conditional equalities is decidable over the class of all rings. We will also show that the problem is decidable over the ring of integers, the ring of reals, and the ring of natural numbers modulo m (for some fixed natural number m). This is achieved by theorem 26 and corollary 27. First we need the following lemma:

Lemma 25 *Let $p(\overline{x}), p_1(\overline{x}), \ldots, p_m(\overline{x})$ be conjunctions of linear predicates. Then it is decidable to check the validity of:*

$$p(\overline{x}) \longrightarrow p_1(\overline{x}) \vee \cdots \vee p_m(\overline{x})$$

Proof: First we will show that the negation of each linear predicate is equivalent to the disjunction of a finite number of linear predicates. This follows from:

$$\neg(\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta \leq 0) \quad \text{iff} \quad (\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta > 0)$$

and:

$$\begin{array}{c} \neg(\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta = 0) \\ \text{iff} \\ (\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta < 0) \vee (\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta > 0) \end{array}$$

and:

$$\neg((\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta)\ mod\ \gamma = 0)$$
$$\text{iff}$$
$$\bigvee_{i=1}^{\gamma-1}((\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta + i)\ mod\ \gamma = 0)$$

Now, we know that:

$$\forall \overline{x}.\ p(\overline{x}) \longrightarrow p_1(\overline{x}) \vee \cdots \vee p_m(\overline{x})$$
$$\text{iff}$$
$$\forall \overline{x}.\ \neg p(\overline{x}) \vee p_1(\overline{x}) \vee \cdots \vee p_m(\overline{x})$$
$$\text{iff}$$
$$\neg\exists \overline{x}.\ \underbrace{p(\overline{x}) \wedge \neg p_1(\overline{x}) \wedge \cdots \wedge \neg p_m(\overline{x})}_{P(\overline{x})}$$

Let us consider $P(\overline{x})$. Each $\neg p_i(\overline{x})$ can be written as a disjunction of negations of linear predicates. A negation of a linear predicate can be replaced by a disjunction of linear predicates as shown above. By taking the normal form it follows that $P(\overline{x})$ can be written as a disjunction of conjunctions of linear predicates, i.e. :

$$P(\overline{x})$$
$$\text{iff}$$
$$P_1(\overline{x}) \vee \cdots \vee P_n(\overline{x})$$

where $P_i(\overline{x})$ is a conjunction of linear predicates. It follows that:

$$\neg\exists \overline{x}.\ P(\overline{x})$$
$$\text{iff}$$
$$\neg(\exists \overline{x}.\ P_1(\overline{x}) \vee \cdots \vee \exists \overline{x}.\ P_n(\overline{x}))$$

It can be shown that the validity of $\exists \overline{x}.\ P_i(\overline{x})$ can be reduced to the integer linear programming problem which is decidable (see section 5.6.4). The result follows immediately. □

Theorem 26 *Let $\mathcal{K}$ be a class of rings. Suppose that the zero-equivalence of a ring polynomial over $\mathcal{K}$ is decidable. Then the validity of ring conditional equalities over $\mathcal{K}$ is decidable.*

Proof: Consider a ring conditional equality $P(\overline{x})$ of the form $p(\overline{x}) \rightarrow (e(\overline{x}) = r(\overline{x}))$. Let $e(\overline{x})$ be of the form:

$$\textbf{case}\ \ p_1(\overline{x}) \Rightarrow r_1(\overline{x})\ ;\ \ldots\ ;\ p_m(\overline{x}) \Rightarrow r_m(\overline{x})\ \ \textbf{endcase}$$

The validity of $P(\overline{x})$ over $\mathcal{K}$ can be checked in two steps. In the first step we check whether the following is true: whenever $p(\overline{x})$ is true then there is a guard in $e(\overline{x})$ which is true. This is equivalent to the validity of:

$$p(\overline{x}) \rightarrow \vee_{i=1}^{m} p_i(\overline{x})$$

over I. We know from lemma 25 that the validity of the above formula is decidable.

In the second step we check whether the following is true: whenever a guard in $e(\overline{x})$ is true then the corresponding result is equal to $r(\overline{x})$ over $\mathcal{K}$. This is equivalent to whether:

$$p(\overline{x}) \wedge p_i(\overline{x}) \rightarrow (r_i(\overline{x}) - r(\overline{x}) = 0)$$

is valid over $\mathcal{K}$, for $1 \leq i \leq m$. The above formula is a zero-equivalence formula, and its validity over $\mathcal{K}$ can be checked according to lemma 24. □

Corollary 27 *The validity of ring conditional equalities is decidable over:*

- *The class of all rings.*
- *The ring of integers, the ring of reals, and the ring of natural numbers modulo m (for some fixed natural number m).*

Proof: Let Q be a ring polynomial of the form:

$$\sum_{i=1}^{m} \gamma_i \cdot v_1^{\beta_{i1}} \cdot \ldots \cdot v_n^{\beta_{in}}$$

First we will prove the second part of the claim. We know that $Q = 0$ is valid over the integers (reals) iff $\gamma_i = 0$, for $1 \leq i \leq m$. Also the validity of $Q = 0$ over the ring Z_m of natural numbers modulo m is decidable since Z_m is finite.

Considering the first part of the claim we know that if $\gamma_i = 0$, for $1 \leq i \leq m$, then $Q = 0$ is valid over the class of all rings (i.e. $Q = 0$ is valid over each ring R). On the other hand if $\gamma_i \neq 0$ for some $1 \leq i \leq m$ then it is not the case that $Q = 0$ is valid over each ring R, as there is at least one ring (the ring of integers) over which $Q = 0$ is not valid. □

5.6.7 Second Step of Verification of The Convolution Circuit: Checking Ring Conditional Equalities

In section 5.5.3 we carried out the first of the two verification steps on the convolution circuit. We described $out_3(x, \ell, t)$ as a linear guarded ring expression (see equation (5.29)).

Here we perform the second step. We use the algorithm in theorem 26 to check whether the value of $out_3(x, \ell, t)$ described by (5.29) fulfills the specification of the circuit as stated in (3.24).

Recalling the specification formula as stated in (3.24):

$$\begin{array}{l} spec(x, \ell, t) \; = \\ \quad (x = 0) \wedge (4 \leq t \leq 4\ell + 4) \wedge (t \bmod 4 = 0) \longrightarrow \\ \qquad \left(out_3(x, \ell, t) \; = \; \sum_{i=0}^{\frac{t-4}{4}} a(i) \cdot b\left(\tfrac{t-4i-4}{4}\right) \right) \\ \qquad\qquad \wedge \\ (x = 0) \wedge (4\ell + 4 < t \leq 8\ell + 4) \wedge (t \bmod 4 = 0) \longrightarrow \\ \quad \left(out_3(x, \ell, t) \; = \; \sum_{i=0}^{\frac{8\ell - t + 4}{4}} a\left(\tfrac{t-4\ell+4i-4}{4}\right) \cdot b(\ell - i) \right) \end{array}$$

Here we will consider only the first element of the conjunction in the specification formula above. The second element can be treated similarly. Let:

$$p(x,\ell,t) = (x=0) \wedge (4 \leq t \leq 4\ell+4) \wedge (t \bmod 4 = 0)$$
$$r(x,\ell,t) = \sum_{i=0}^{\frac{t-4}{4}} a(i) \cdot b\left(\frac{t-4i-4}{4}\right)$$

From equation (5.29) we know that:

$out_3(x,\ell,t) =$
case

$$\left\{ \begin{array}{c} 2x+4 \leq t \leq 4\ell-2x+4 \\ 0 \leq x \leq \ell \\ (t-2x) \bmod 4 = 0 \end{array} \right\} \Longrightarrow \sum_{i=0}^{\frac{t-2x-4}{4}} a\left(\tfrac{t-2x-4i-4}{4}\right) \cdot b(x+i)$$

$$\left\{ \begin{array}{c} 4\ell-2x+8 \leq t \leq 4\ell+2x+4 \\ 0 \leq x \leq \ell \\ (t-2x) \bmod 4 = 0 \end{array} \right\} \Longrightarrow \sum_{i=0}^{\ell-x} a\left(\tfrac{t-2x-4i-4}{4}\right) \cdot b(x+i)$$

$$\left\{ \begin{array}{c} 4\ell+2x+8 \leq t \leq 8\ell-2x+4 \\ 0 \leq x \leq \ell \\ (t-2x) \bmod 4 = 0 \end{array} \right\} \Longrightarrow \sum_{i=0}^{\frac{8\ell-2x-t+4}{4}} a(\ell-i) \cdot b\left(\tfrac{t-4\ell+2x+4i-4}{4}\right)$$

endcase

Let:

$$p_1(x,\ell,t) = (2x+4 \leq t \leq 4\ell-2x+4) \wedge (0 \leq x \leq \ell) \wedge ((t-2x) \bmod 4 = 0)$$
$$p_2(x,\ell,t) = (4\ell-2x+8 \leq t \leq 4\ell+2x+4) \wedge (0 \leq x \leq \ell) \wedge ((t-2x) \bmod 4 = 0)$$
$$p_3(x,\ell,t) = (4\ell+2x+8 \leq t \leq 8\ell-2x+4) \wedge (0 \leq x \leq \ell) \wedge ((t-2x) \bmod 4 = 0)$$
$$r_1(x,\ell,t) = \sum_{i=0}^{\frac{t-2x-4}{4}} a\left(\frac{t-2x-4i-4}{4}\right) \cdot b(x+i)$$
$$r_2(x,\ell,t) = \sum_{i=0}^{\ell-x} a\left(\frac{t-2x-4i-4}{4}\right) \cdot b(x+i)$$
$$r_3(x,\ell,t) = \sum_{i=0}^{\frac{8\ell-2x-t+4}{4}} a(\ell-i) \cdot b\left(\frac{t-4\ell+2x+4i-4}{4}\right)$$

According to the algorithm given in the proof of theorem 26 we have first to check the validity of the formula:

$$p(x,\ell,t) \longrightarrow p_1(x,\ell,t) \vee p_2(x,\ell,t) \vee p_3(x,\ell,t)$$

We observe that $p(x,\ell,t) \longrightarrow p_1(x,\ell,t)$, so the above formula is valid.

Then we have to check the validity of the formulas:

$$p(x,\ell,t) \wedge p_1(x,\ell,t) \longrightarrow (r_1(x,\ell,t) = r(x,\ell,t))$$
$$p(x,\ell,t) \wedge p_2(x,\ell,t) \longrightarrow (r_2(x,\ell,t) = r(x,\ell,t))$$
$$p(x,\ell,t) \wedge p_3(x,\ell,t) \longrightarrow (r_3(x,\ell,t) = r(x,\ell,t))$$

Each of the formulas above is a zero-equivalence formula, and its validity can be checked as described by the algorithm in lemma 24.

Chapter 6

Class of all algebras

In this chapter we study verification of systolic circuits which operate on uninterpreted function symbols. We do not assume any special properties of the functions corresponding to the operations performed inside the computation cells of the circuit. We recall that in chapter 5 we assumed that the cell operators fulfilled the properties (axioms) of a commutative ring. Often, it turns out that a circuit can be verified independently of the particular properties of the cell operations. In such a case the verification amounts to checking the *time-space* properties of the implementation with respect to the specification. We check that the right computation is performed at the right place at the right time. As we will see in the case of the string matching circuit (introduced in section 3.2), the verification is nevertheless nontrivial. This is due to the fact that the verification still involves i) checking properties of the integers, which are still used for the description of the notions of time, cell position, and circuit parameter, and ii) comparing the "recursive patterns" of the recurrence equations describing the implementation and the specification. Proving a circuit correct with respect to a specification, for uninterpreted cell operations, implies its correctness for each interpretation of the cell operations. Clearly if the circuit is not correct with respect to the specification for all interpretations of the cell operations, there may still be interpretations of the cell operations over which the implementation is correct with respect to the specification.

Formally by the verification of a circuit with a signature $\mathcal{SIG}$, for uninterpreted function symbols, we mean that we check the validity of the specification formula of the circuit over the algebraic specification $\mathcal{SPEC}$, which has a signature $\mathcal{SIG}$, and an equation set which is empty. The validity of the specification formula over $\mathcal{SPEC}$ implies its validity over each $\mathcal{SIG}$-algebra. On the other hand, if the specification formula is not valid over $\mathcal{SPEC}$ there may still be a $\mathcal{SIG}$-algebra over which the specification formula is valid.

Examples of circuits which can be automatically verified by the methods of this chapter include the string matching circuit in section 3.2, the palindrome recognizer circuits in [Hen86] and [LS81], and the matrix transposition circuits in [Ull84] and [Ole87].

In section 6.1 we give some preliminaries. In section 6.2 we define a class of systolic circuits which we call *acyclic systolic circuits*. In section 6.3 we study what the general definition of semantics amounts to when working with uninterpreted function symbols. In section 6.4 we give an overview of a decision method for automatic verification of acyclic systolic circuits. In section 6.5 we will show that the implementations and specifications

of acyclic systolic circuits can be described by a class of recurrence equations which we call *acyclic recurrence equations*. In section 6.6 we give a method for comparing acyclic recurrence equations for equality over the class of all algebras (i.e. with uninterpreted function symbols), and thus carry out automatic verification of acyclic systolic circuits over the class of all algebras.

6.1 Preliminaries

We will use the same notation as in chapter 5. Linear QI-polynomials, linear inequalities, linear equalities, linear modulo predicates, and linear predicates have the same meanings as in section 5.1.

We will work with a set $\mathcal{G}$ of operation symbols, a set $\mathcal{A}$ of stream variables. A *linear function variable expression* is of the form $f(\overline{x}+\overline{\alpha})$. A *linear systolic term* is of the form:

$$g(t_1(\overline{x}),\ldots,t_n(\overline{x}))$$

where $g \in \mathcal{G}$, and each $t_i(\overline{x})$ is either a linear stream expression or a linear function variable expression. A functional is said to be *linear* if each of its results is a linear stream expression, a linear function variable expression, or a linear systolic term. A system of recurrence equations is said to be *linear* if all its functionals are linear.

A *(one-dimensional) integer vector* is of the form $< \delta_1,\ldots,\delta_n >$ where each δ_i is an integer. A *zero vector*, denoted $\overline{0}$, is an integer vector $< \delta_1,\ldots,\delta_n >$, where $\delta_i = 0$, for $1 \leq i \leq n$. A *unit vector* denoted, $\overline{u}_i$, is an integer vector $< \delta_1,\ldots,\delta_n >$, where $\delta_i = 1$, and $\delta_j = 0$, if $j \neq i$ for $1 \leq j \leq n$. A *positive vector* is an integer vector $< \delta_1,\ldots,\delta_n >$, where $0 \leq \delta_i$, for $1 \leq i \leq n$, and there is at least one $1 \leq j \leq n$ such that $0 < \delta_j$. An *m-dimensional integer vector* is of the form $< \overline{\delta}_1,\ldots,\overline{\delta}_n >$ where $\overline{\delta}_i$ is an $(m-1)$-dimensional integer vector.

Consider a function variable f and a linear systolic term $g(t_1(\overline{x}),\ldots,t_n(\overline{x}))$. Let $t_{i_1}(\overline{x}),\ldots,t_{i_m}(\overline{x})$ be such that $t_{i_j}(\overline{x})$ is of the form $f(\overline{x}+\overline{\delta}_j)$, and such that no $t(\overline{x}) \in \{t_1(\overline{x}),\ldots,t_n(\overline{x})\} \setminus \{t_{i_1}(\overline{x}),\ldots,t_{i_m}(\overline{x})\}$ is a function variable expression which has f as a function variable. Then $< \overline{\delta}_1,\ldots,\overline{\delta}_m >$ is called the *recursive vector* of f in $g(t_1(\overline{x}),\ldots,t_n(\overline{x}))$, and the elements of $\{t_1(\overline{x}),\ldots,t_n(\overline{x})\} \setminus \{t_{i_1}(\overline{x}),\ldots,t_{i_m}(\overline{x})\}$ are called the *non-recursive terms* of f in $g(t_1(\overline{x}),\ldots,t_n(\overline{x}))$. For example, let $t(x) = g(h(x+2), f(x+3), a(4x), f(x-2))$, then the recursive vector of f in $t(x)$ is $< 3, -2 >$, and the non-recursive terms of f in $t(x)$ are $h(x+2)$ and $a(4x)$. The *recursion vector* of f in the linear function variable expression $f(\overline{x}+\overline{\beta})$ is defined to be $< \overline{\beta} >$.

Consider a conjunction $p(\overline{x})$ of linear inequalities and equalities, and a two-dimensional integer vector $\overline{\delta}$. We say that $p(\overline{x})$ is *stable* with respect to $\overline{\delta}$ iff there is no value $\overline{\alpha}$ of $\overline{x}$, and positive vectors $\overline{\mu}_1$ and $\overline{\mu}_2$ such that[1]:

$$p(\overline{\alpha}) \ \wedge \ \neg p(\overline{\alpha}+\overline{\mu}_1\cdot\overline{\delta}) \ \wedge \ p(\overline{\alpha}+\overline{\mu}_1\cdot\overline{\delta}+\overline{\mu}_2\cdot\overline{\delta})$$

[1]Let $\overline{\alpha} =< \alpha_1,\ldots,\alpha_m >$, $\overline{\mu} =< \mu_1,\ldots,\mu_n >$, and $\overline{\delta} =<< \delta_{11},\ldots,\delta_{1m} >,\ldots,< \delta_{n1},\ldots,\delta_{nm} >>$. According to our vector notation (section 2.1), we interpret $\overline{\alpha}+\overline{\mu}\cdot\overline{\delta}$ as:

$$< \alpha_1+\mu_1\cdot\delta_{11}+\cdots+\mu_n\cdot\delta_{n1},\ \ldots\ ,\alpha_m+\mu_1\cdot\delta_{1m}+\cdots+\mu_n\cdot\delta_{nm} >$$

Proposition 28 *Given a conjunction $p(\overline{x})$ of linear inequalities and equalities, and a two-dimensional integer vector $\overline{\delta}$, it is decidable to check whether $p(\overline{x})$ is stable with respect to $\overline{\delta}$.*

Proof: From the definition of stability above, it can be easily shown that the problem is reducible to the integer linear programming problem which is decidable (see section 5.6.4). □

A conjunction of linear predicates $p_1(\overline{x}) \wedge p_2(\overline{x})$, where $p_1(\overline{x})$ is a conjunction of linear inequalies and equalities and $p_2(\overline{x})$ is a conjunction of linear modulo predicates, is said to be *stable* with respect to a two-dimensional integer vector $\overline{\delta}$ if $p_1(\overline{x})$ is stable with respect to $\overline{\delta}$.

Consider a tuple $< p_1(\overline{x}), \ldots, p_n(\overline{x}) >$, where each $p_i(\overline{x})$ is a conjunction of linear inequalities and equalities, and a tuple $< \overline{\delta}_1, \ldots, \overline{\delta}_n >$, where each $\overline{\delta}_i$ is a two-dimensional integer vector. Then we say that $< p_1(\overline{x}), \ldots, p_n(\overline{x}) >$ is *uniform* with respect to $< \overline{\delta}_1, \ldots, \overline{\delta}_n >$ iff there is no value $\overline{\alpha}$ of $\overline{x}$ and positive vectors $\overline{\mu}_1, \ldots, \overline{\mu}_k$, such that:

$$p_{i_0}(\overline{\alpha}) \wedge p_{i_1}(\overline{\alpha}+\overline{\mu}_1 \cdot \overline{\delta}_{i_1}) \wedge p_{i_2}(\overline{\alpha}+\overline{\mu}_1 \cdot \overline{\delta}_{i_1}+\overline{\mu}_2 \cdot \overline{\delta}_{i_2}) \wedge \cdots \wedge p_{i_k}(\overline{\alpha}+\overline{\mu}_1 \cdot \overline{\delta}_{i_1}+\overline{\mu}_2 \cdot \overline{\delta}_{i_2}+\cdots+\overline{\mu}_k \cdot \overline{\delta}_{i_k})$$

where $i_k = i_0$, and $1 \leq i_j \leq n$, for $0 \leq j \leq k$.

Proposition 29 *Consider a tuple $< p_1(\overline{x}), \ldots, p_n(\overline{x}) >$, where each $p_i(\overline{x})$ is a conjunction of linear inequalities and equalities, and a tuple $< \overline{\delta}_1, \ldots, \overline{\delta}_n >$, where each $\overline{\delta}_i$ is a two-dimensional integer vector. Then it is decidable to check whether $< p_1(\overline{x}), \ldots, p_n(\overline{x}) >$ is uniform with respect to $< \overline{\delta}_1, \ldots, \overline{\delta}_n >$*

Proof: For each k the problem of satisfiability of:

$$p_{i_0}(\overline{\alpha}) \wedge p_{i_1}(\overline{\alpha}+\overline{\mu}_1 \cdot \overline{\delta}_{i_1}) \wedge p_{i_2}(\overline{\alpha}+\overline{\mu}_1 \cdot \overline{\delta}_{i_1}+\overline{\mu}_2 \cdot \overline{\delta}_{i_2}) \wedge \cdots \wedge p_{i_k}(\overline{\alpha}+\overline{\mu}_1 \cdot \overline{\delta}_{i_1}+\overline{\mu}_2 \cdot \overline{\delta}_{i_2}+\cdots+\overline{\mu}_k \cdot \overline{\delta}_{i_k})$$

can be easily shown to be reducible to the integer linear programming problem which is decidable (see section 5.6.4). The result follows from the fact that $k \leq n$, which means that only finitely many combinations need to be considered. □

A tuple $< p_{11}(\overline{x}) \wedge p_{12}(\overline{x}), \ldots, p_{n1}(\overline{x}) \wedge p_{n2}(\overline{x}) >$, where each $p_{i1}(\overline{x})$ is a conjunction of linear inequalities and equalities, and each $p_{i2}(\overline{x})$ is a conjunction of linear modulo predicates, is said to be *uniform* with respect to $< \overline{\delta}_1, \ldots, \overline{\delta}_n >$, where each $\overline{\delta}_i$ is a two-dimensional integer vector if $< p_{11}(\overline{x}), \ldots, p_{n1}(\overline{x}) >$ is uniform with respect to $< \overline{\delta}_1, \ldots, \overline{\delta}_n >$.

A linear modulo predicate $p(\overline{x})$ of the form $(\overline{\alpha} \cdot \overline{x} + \beta) \ mod \ \gamma = 0$, is said to be *invariable* with respect to a two-dimensional integer vector $\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_n >$, if $(\overline{\alpha} \cdot \overline{\delta}_i) \ mod \ \gamma = 0$, for $1 \leq i \leq n$. Notice that if $p(\overline{x})$ is invariable with respect to $\overline{\delta}$ then, for each $\overline{\alpha}$, if $p(\overline{\alpha})$ is true then $p(\overline{\alpha} + \overline{\mu} \cdot \overline{\delta})$ will be true for each integer vector $\overline{\mu}$. A conjunction of linear predicates $p_1(\overline{x}) \wedge p_2(\overline{x})$, where $p_1(\overline{x})$ is a conjunction of linear inequalies and equalities and $p_2(\overline{x})$ is a conjunction of linear modulo predicates, is said to be *invariable* with respect to a two-dimensional integer vector $\overline{\delta}$ if each modulo predicate in $p_2(\overline{x})$ is invariable with respect to $\overline{\delta}$.

Let $\mathcal{F}$ be a linear functional of the form:

$$\begin{array}{l}\textbf{case}\\ \quad p_1(\overline{x}) \implies t_1(\overline{x})\\ \qquad\vdots\\ \quad p_n(\overline{x}) \implies t_n(\overline{x})\\ \textbf{endcase}\end{array}$$

Let $\{t_{i_1}(\overline{x}),\ldots,t_{i_m}(\overline{x})\} \subseteq \{t_1(\overline{x}),\ldots,t_n(\overline{x})\}$ be such that $t(\overline{x}) \in \{t_{i_1}(\overline{x}),\ldots,t_{i_m}(\overline{x})\}$ iff f occurs in $t(\overline{x})$. Let the recursive vectors of f in $t_{i_1}(\overline{x}),\ldots,t_{i_m}(\overline{x})$ be $\overline{\delta}_1,\ldots,\overline{\delta}_m$ respectively. Then $\mathcal{F}$ is said to be *acyclic* with respect to f if:

- $p_{i_j}(\overline{x})$ is stable with respect to $\overline{\delta}_j$, for $1 \leq j \leq m$.
- $p_{i_j}(\overline{x})$ is invariable with respect to $\overline{\delta}_j$, for $1 \leq j \leq m$.
- $< p_{i_1}(\overline{x}),\ldots,p_{i_m}(\overline{x}) >$ is uniform with respect to $< \overline{\delta}_1,\ldots,\overline{\delta}_m >$.

Consider a system of linear recurrence equations of the form:

$$\begin{array}{rcl} f_1(\overline{x}) & = & \mathcal{F}_1(\overline{x})\\ & \vdots & \\ f_n(\overline{x}) & = & \mathcal{F}_n(\overline{x}) \end{array} \tag{6.1}$$

We say that f_i is *dependent* on f_j (written $f_j \prec_D f_i$) iff f_j occurs in $\mathcal{F}_i$.

A linear system of recurrence equations of the form of (6.1) is called an *acyclic system of recurrence equations* iff:

1. Each $\mathcal{F}_i$ is acyclic with respect to f_i.
2. The dependency relation $\prec_D$ is acyclic, i.e. $f_i \not\prec_D^+ f_i$, for $1 \leq i \leq n$, where $\prec_D^+$ is the transitive closure of $\prec_D$.

We say that $\mathcal{SPEC}$ is the *algebraic specification of the class of all $\mathcal{SIG}$-algebras* if $\mathcal{SPEC} =< \mathcal{SIG}, \emptyset >$.

6.2 The Class of Acyclic Systolic Circuits

In this section we introduce the class of *acyclic* systolic circuits. An example of a circuit in this class is the string matching circuit we introduced in section 3.2. In section 6.5 we will show that the implementation and specification of an acyclic systolic circuit can be described by acyclic systems of recurrence equations (and hence the name *acyclic*).

An *acyclic systolic circuit* is a linear systolic circuit, in which certain restrictions have been imposed on the cell computations and the specification formula of the circuit.

Cell Computations in Acyclic Systolic Circuits A cell computation term is said to be *linear* if it is of the form:

$$g(s_1(\overline{x},\overline{\ell},t-\tau_1),\ldots,s_n(\overline{x},\overline{\ell},t-\tau_n))$$

where s_i is an input or a local variable. A cell computation is said to be *linear* if each result in the cell computation is a linear stream expression, a linear cell computation term, or of the form $s(\overline{x},\overline{\ell},t-\tau)$ where s is an input or a local variable.

Two types of cell computations are allowed in acyclic systolic circuits:

1. *Linear cell computations*, where the cell computation $\mathcal{F}(\overline{x},\overline{\ell},t)$, associated with an output or a local variable s, is linear. Let τ be the delay of $\mathcal{F}(\overline{x},\overline{\ell},t)$, then the value of s will be defined by the two equations:

$$(0 \leq t < \tau) \wedge top(\overline{x},\overline{\ell}) \longrightarrow s(\overline{x},\overline{\ell},t) = Init_s(\overline{x},\overline{\ell},t) \tag{6.2}$$

and:

$$(t \geq \tau) \wedge top(\overline{x},\overline{\ell}) \longrightarrow s(\overline{x},\overline{\ell},t) = \mathcal{F}(\overline{x},\overline{\ell},t) \tag{6.3}$$

where equation (6.2) describes the value of s during the initialization period, and equation (6.3) describes the value of s after the initialization period.

2. *Branching cell computations.* If a *branching cell computation* is associated with an output s, then the value of s is defined by:

$$s(\overline{x},\overline{\ell},t) = s_1(\overline{x},\overline{\ell},t) \tag{6.4}$$

where s_1 is an output or a local variable whose value is defined by a linear cell computation. Intuitively equation (6.4) means that, at each clock cycle, the value of s_1 inside a cell is *branched* to the value of s in the cell i.e. the value of s is equal to the value of s_1 at each clock cycle and cell in the circuit.

In the string matching circuit (section 3.2), the values of out_1, out_2, and loc are defined by linear cell computations, while the values of out_3 and out_4 are defined by branching cell computations.

Furthermore a number of additional restrictions are made on the cell computations in an acyclic systolic circuit. These restrictions enable us (theorem 30) to describe the implementations and the specifications of acyclic systolic circuits as acyclic systems of recurrence equations, which are decidable to compare for equality (corollary 51). The additional restrictions are defined by:

1. Let s be any output or local variable. Let the cell computation $\mathcal{F}(\overline{x},\overline{\ell},t)$ associated with s be linear. Let $\mathcal{F}'(\overline{x},\overline{\ell},t)$ be the functional we get from $\mathcal{F}(\overline{x},\overline{\ell},t)$ by performing the following two steps:

- If s is an output, let s' be the input corresponding to (connected to) s, and $\overline{\delta}$ the connection vector of s'. Then replace each occurrence of the form $s'(\overline{x},\overline{\ell},t-\tau)$ in $\mathcal{F}(\overline{x},\overline{\ell},t)$ by $s(\overline{x}+\overline{\delta},\overline{\ell},t-\tau)$.

- Let s_1 be any output whose value is defined by a branching of the form:

$$s_1(\overline{x}, \overline{\ell}, t) \;=\; s(\overline{x}, \overline{\ell}, t)$$

  Let s_1' be the input corresponding to (connected to) s_1, and $\overline{\delta}_1$ the connection vector of s_1'. Then replace each occurrence of the form $s_1'(\overline{x}, \overline{\ell}, t-\tau)$ in $\mathcal{F}(\overline{x}, \overline{\ell}, t)$ by $s(\overline{x} + \overline{\delta}, \overline{\ell}, t-\tau)$.

In an acyclic systolic circuit, $\mathcal{F}'(\overline{x}, \overline{\ell}, t)$ is acyclic with respect to s.

2. The *dependency relation* $\prec_D$ ($s_2 \prec_D s_1$ is read s_1 is dependent on s_2) among the signals of an acyclic systolic circuit is defined as the smallest relation containing the following elements:

- Let s be any output or local variable, and let the computation of s be of the form of equation (6.3). Let s_1 be any input or local variable which occurs in $\mathcal{F}(\overline{x}, \overline{\ell}, t)$. If s_1 is a local variable and $s_1 \neq s$ then $s_1 \prec_D s$. If s_1 is an input, let s_1' be the output corresponding to s_1. If $s_1' \neq s$ then $s_1' \prec_D s$.

- Let s be any output whose computation is of the form of equation (6.4), then for each s_2 if $s \prec_D s_2$ and $s_1 \neq s_2$ then $s_1 \prec_D s_2$.

In the class of acyclic systolic circuits, the dependency relation is acyclic, i.e. $s \not\prec_D^+ s$, for each signal s in the circuit, where $\prec_D^+$ is the transitive closure of $\prec_D$.

The dependency relation of the string matching circuit in section 3.2 is defined by $\{< loc, out_1 >, < loc, out_2 >, < loc, out_3 >, < loc, out_4 >\}$. Notice that the relation is acyclic.

Specification Formulas of Acyclic Systolic Circuits The specification formula of an acyclic systolic circuit with a signature $\mathcal{SIG}$ is of the form:

$$spec(\overline{x}, \overline{\ell}, t) \;=$$
$$(p_1(\overline{x}, \overline{\ell}, t) \longrightarrow (s_1(\overline{x}, \overline{\ell}, t) = f_1(\overline{x}, \overline{\ell}, t))) \wedge \cdots \wedge (p_m(\overline{x}, \overline{\ell}, t) \longrightarrow (s_m(\overline{x}, \overline{\ell}, t) = f_m(\overline{x}, \overline{\ell}, t)))$$

where f_i is a function variable in an acyclic system of recurrence equations over $\mathcal{SIG}$.

It can be easily checked that the string matching circuit described in section 3.2 is an acyclic systolic circuit.

6.3 Semantics

Suppose that E_1 and E_2 are acyclic systems of recurrence equations over a signature $\mathcal{SIG}$. Let f_1 and f_2 be function variables in E_1 and E_2 respectively. Then, given a $\mathcal{SIG}$-algebra A, the relation:

$$A \;\models\; f_1(\overline{\alpha}_1) = f_2(\overline{\alpha}_2) \tag{6.5}$$

can be interpreted according to the semantics definition in section 4.3.

In this chapter we study the problem of deciding the validity of (6.5) for each $\mathcal{SIG}$-algebra A. Alternatively, whether:

$$\mathcal{SPEC} \;\models\; f_1(\overline{\alpha}_1) = f_2(\overline{\alpha}_2) \tag{6.6}$$

where $\mathcal{SPEC}$ is the algebraic specification of all $\mathcal{SIG}$-algebras (i.e. $\mathcal{SPEC} =< \mathcal{SIG}, \emptyset >$). Note that if (6.6) is valid, then (6.5) will be valid for each $\mathcal{SIG}$-algebra A. If (6.6) is not valid, there may still be a $\mathcal{SIG}$-algebra A for which (6.5) is valid.

Informally, we check the the equality of systems of recurrence equations without assuming particular properties for the sorts and operation symbols of the signatures of the equation systems. Thus if the equation systems are equal then there will be equal for each interpretation of the sorts and operation symbols. If they are not equal then we may still find an interpretation of the sorts and operation symbols which makes them equal.

6.4 Overview of the Verification Decision Method

We will give an overview of an automatic method for the verification of acyclic systolic circuits over the class of all algebras.

In section 6.2 we mentioned that a circuit specification was of the form:

$$spec(\overline{x}) = \\ (p_1(\overline{x}) \longrightarrow (s_1(\overline{x}) = f_1(\overline{x}))) \wedge \cdots \wedge (p_m(\overline{x}) \longrightarrow (s_m(\overline{x}) = f_m(\overline{x}))) \tag{6.7}$$

where $p_1(\overline{x}), \ldots, p_m(\overline{x})$ are conjunctions of linear predicates, $s_1, \ldots, s_m$ are signals in the circuit, and $f_1, \ldots, f_m$ are function variables in acyclic systems of recurrence equations over the signature of the circuit.

Let $\mathcal{SPEC}$ be the algebraic specification of the class of all $\mathcal{SIG}$-algebras, i.e. :

$$\mathcal{SPEC} = < \mathcal{SIG}, \emptyset >$$

Consider a circuit with a signature $\mathcal{SIG}$ and a specification formula $spec(\overline{x})$. By the *circuit verification* for the class of all $\mathcal{SIG}$-algebras, we mean that we check whether or not:

$$A \models spec(\overline{x})$$

is valid for each $\mathcal{SIG}$-algebra A. This is equivalent to checking:

$$\mathcal{SPEC} \models spec(\overline{x}) \tag{6.8}$$

Notice that although the sorts and operation symbols of $\mathcal{SIG}$ are interpreted as domains and operations of the algebras of $\mathcal{SIG}$, the integer variables $\overline{x}$ which occur in $spec(\overline{x})$, and the integer operators $+$, $\cdot$, $-$, $<$, and 0 which occur in $p_1(\overline{x}), \ldots, p_m(\overline{x})$ are always interpreted in the standard model of integers. The verification process is carried out in the following two steps, each of which is carried out automatically:

1. In section 6.5 (theorem 30) we show that the signals of an acyclic systolic circuit can be described by an acyclic system of recurrence equations, where the signature of the equation system is the same as the signature of the circuit, and the function variables of the equation system are the signals of the circuit.

2. In section 6.6 we study validity of formulas of the general form:

$$\mathcal{SPEC} \models p(\overline{x}_1, \overline{x}_2) \longrightarrow (f_1(\overline{x}_1) = f_2(\overline{x}_2)) \tag{6.9}$$

where $p(\overline{x}_1, \overline{x}_2)$ is a conjunction of linear predicates, f_1 and f_2 are function variables in acyclic systems of recurrence equations with a signature $\mathcal{SIG}$.

From (6.7), (6.8), and the first verification step we conclude that if (6.9) is decidable then (6.7) will be decidable.

In section 4.3 we defined the formal meaning of:

$$\mathcal{SPEC} \models (f_1(\overline{\alpha}_1) = f_2(\overline{\alpha}_2))$$

for each value $\overline{\alpha}_1$ and $\overline{\alpha}_2$ of $\overline{x}_1$ and $\overline{x}_2$.

In section 6.6.1 we define a class of formulas called *Presburger formulas*, which are the class of first order formulas over the structure $< I, +, -, <, 0, 1 >$ of integers. In section 6.6.6 (theorem 50) we show that given $p(\overline{x}_1, \overline{x}_2)$, f_1 and f_2 above there is a Presburger formula $P(\overline{x}_1, \overline{x}_2)$ such that:

$$P(\overline{x}_1, \overline{x}_2) \quad \text{iff} \quad \mathcal{SPEC} \models p(\overline{x}_1, \overline{x}_2) \longrightarrow (f_1(\overline{x}_1) = f(\overline{x}_2)) \tag{6.10}$$

The validity of Presburger formulas has been shown to be decidable. It follows that the validity of formulas of the form of (6.9) is decidable.

Now we will sketch how $P(\overline{x}_1, \overline{x}_2)$ can be found such that (6.10) holds. In section 6.6.4 (lemma 46) we show that there are Presburger formulas $P_1(\overline{x})$ and $P_2(\overline{x})$ such that:

$$P_1(\overline{x}) \quad \text{iff} \quad f_1(\overline{x}) = \perp \tag{6.11}$$

for each $\mathcal{SIG}$-algebra A, and:

$$P_2(\overline{x}) \quad \text{iff} \quad f_2(\overline{x}) = \perp \tag{6.12}$$

for each $\mathcal{SIG}$-algebra A. In section 6.6.5 (lemma 49) we show that there is a Presburger formula $P_3(\overline{x}_1, \overline{x}_2)$ such that:

$$\begin{array}{l} (f_1(\overline{x}_1) \neq \perp) \wedge (f_2(\overline{x}_2) \neq \perp) \longrightarrow \\ \quad (P_3(\overline{x}_1, \overline{x}_2) \text{ iff } (\mathcal{SPEC} \models (f_1(\overline{x}_1) = f_2(\overline{x}_2)))) \end{array} \tag{6.13}$$

From (6.11), (6.12), and (6.13) it follows that $P(\overline{x}_1, \overline{x}_2)$ in (6.10) can be defined as:

$$P(\overline{x}_1, \overline{x}_2) = \left(p(\overline{x}_1, \overline{x}_2) \longrightarrow \left(\begin{array}{c} (P_1(\overline{x}_1) \wedge P_2(\overline{x}_2)) \\ \vee \\ (\neg P_1(\overline{x}_1) \wedge \neg P_2(\overline{x}_2) \wedge P_3(\overline{x}_1, \overline{x}_2)) \end{array} \right) \right)$$

As Presburger formulas are first order formulas, they are closed under the first order logical connectives. It follows that $P(\overline{x}_1, \overline{x}_2)$ is a Presburger formula.

6.5 Systolic Circuits as Acyclic Recurrence Equations

We will show that the signals of an acyclic systolic circuit can be described as an acyclic system of recurrence equations.

Consider an acyclic systolic circuit. Let $\mathcal{F}(\overline{x},\overline{\ell},t)$ be a linear cell computation (see section 6.2) in the circuit. By the *normalization* of $\mathcal{F}(\overline{x},\overline{\ell},t)$ we mean the functional $\mathcal{F}^*(\overline{x},\overline{\ell},t)$ we get from $\mathcal{F}(\overline{x},\overline{\ell},t)$ by applying the following two steps:

1. Let s be any input. Let the output corresponding to (connected to) s be s', the connection vector of s be $\overline{\delta}$, and the input function corresponding to s be of the form:

$$\begin{array}{l}\textbf{case}\\ \quad p_1(\overline{x},\overline{\ell},t) \implies it_1(\overline{x},\overline{\ell},t)\\ \qquad\qquad\vdots\\ \quad p_n(\overline{x},\overline{\ell},t) \implies it_n(\overline{x},\overline{\ell},t)\\ \textbf{endcase}\end{array}$$

Let:

$$p(\overline{x},\overline{\ell},t) \implies ct(\overline{x},\overline{\ell},t)$$

be any case in $\mathcal{F}(\overline{x},\overline{\ell},t)$. Then replace the case above by the following cases:

- The case:

$$p(\overline{x},\overline{\ell},t) \wedge top'(\overline{x},\overline{\ell}) \implies ct'(\overline{x},\overline{\ell},t)$$

  where $top'(\overline{x},\overline{\ell})$ denotes the position of the cells whose inputs s are connected to the outputs of other cells, and $ct'(\overline{x},\overline{\ell},t)$ is the result of replacing each occurence of the form $s(\overline{x},\overline{\ell},t-\tau)$ in $ct(\overline{x},\overline{\ell},t)$ by $s'(\overline{x}+\overline{\delta},\overline{\ell},t-\tau)$.

- The case:

$$p(\overline{x},\overline{\ell},t) \wedge top''(\overline{x},\overline{\ell}) \wedge p_i(\overline{x},\overline{\ell},t) \implies ct'_i(\overline{x},\overline{\ell},t)$$

  for each $1 \le i \le n$, where $top''(\overline{x},\overline{\ell})$ denotes the position of the cells whose input s are on the boundary of the circuit (i.e. not connected to the outputs of other cells), and $ct'_i(\overline{x},\overline{\ell},t)$ is the result of replacing each occurence of the form $s(\overline{x},\overline{\ell},t-\tau)$ in $ct(\overline{x},\overline{\ell},t)$ by $it_i(\overline{x},\overline{\ell},t-\tau)$.

  Repeat the above procedure until no inputs are left in $\mathcal{F}(\overline{x},\overline{\ell},t)$.

2. Let $\mathcal{F}'(\overline{x},\overline{\ell},t)$ be the result of applying step (1) above to $\mathcal{F}(\overline{x},\overline{\ell},t)$. We get $\mathcal{F}^*(\overline{x},\overline{\ell},t)$ by applying the second step to $\mathcal{F}'(\overline{x},\overline{\ell},t)$. Let s be any output whose value is defined by branching, i.e. whose computation is of the form:

$$s(\overline{x},\overline{\ell},t) = s'(\overline{x},\overline{\ell},t)$$

Let:

$$p(\overline{x},\overline{\ell},t) \implies ct(\overline{x},\overline{\ell},t)$$

be any case in $\mathcal{F}'(\overline{x},\overline{\ell},t)$. Then replace the case above by the case:

$$p(\overline{x},\overline{\ell},t) \implies ct'(\overline{x},\overline{\ell},t)$$

where $ct'(\overline{x},\overline{\ell},t)$ is the result of replacing each occurence of the form $s(\overline{x},\overline{\ell},t-\tau)$ in $ct(\overline{x},\overline{\ell},t)$ by $s'(\overline{x},\overline{\ell},t-\tau)$.

Repeat the above procedure until no signals, whose values are defined by branching, occur in $\mathcal{F}'(\overline{x},\overline{\ell},t)$.

Theorem 30 *Consider an acyclic systolic circuit. Then the values of the signals of the circuit can be described as an acyclic system of recurrence equations, where the signature of the equation system is the same as the signature of the circuit and the function variables of the equation system are the signals of the circuit.*

Proof: The acyclic system E of recurrence equations defining the value of the signals of the circuit is given by the smallest set containing the following elements:
1. If s is an output or a local variable whose value is defined by a linear cell computation, then let the result of normalization of the cell computation of s be of the form:

$$\begin{array}{l}\textbf{case}\\ \quad p_1(\overline{x},\overline{\ell},t) \implies ct_1(\overline{x},\overline{\ell},t)\\ \qquad\qquad \vdots\\ \quad p_n(\overline{x},\overline{\ell},t) \implies ct_n(\overline{x},\overline{\ell},t)\\ \textbf{endcase}\end{array}$$

Let the initial function corresponding to s be of the form:

$$\begin{array}{l}\textbf{case}\\ \quad p'_1(\overline{x},\overline{\ell},t) \implies it_1(\overline{x},\overline{\ell},t)\\ \qquad\qquad \vdots\\ \quad p'_m(\overline{x},\overline{\ell},t) \implies it_m(\overline{x},\overline{\ell},t)\\ \textbf{endcase}\end{array}$$

Let the delay of s be τ. Then E contains the equation:

$$s(\overline{x},\overline{\ell},t) = \mathcal{F}(\overline{x},\overline{\ell},t)$$

where the set of cases of $\mathcal{F}(\overline{x},\overline{\ell},t)$ is the smallest set containing:

- The case:

$$(0 \le t < \tau) \wedge top(\overline{x},\overline{\ell}) \wedge p'_i(\overline{x},\overline{\ell},t) \implies it_i(\overline{x},\overline{\ell},t)$$

  for each $1 \le i \le m$.

- The case:

$$(\tau \le t) \wedge top(\overline{x},\overline{\ell}) \wedge p_i(\overline{x},\overline{\ell},t) \implies ct_i(\overline{x},\overline{\ell},t)$$

  for each $1 \le i \le n$.

2. If s is an output or local variable whose value is defined by branching, then let the value of s be defined by an equation of the form:

$$s(\overline{x},\overline{\ell},t) = s'(\overline{x},\overline{\ell},t)$$

then E contains the above equation.

3. If s is an input, then let the connection vector corresponding to s be $\overline{\delta}$, the output connected to s be s', and the input function corresponding to s be of the form:

$$\begin{array}{l}\textbf{case}\\ \quad p_1(\overline{x},\overline{\ell},t) \implies it_1(\overline{x},\overline{\ell},t)\\ \qquad\vdots\\ \quad p_n(\overline{x},\overline{\ell},t) \implies it_n(\overline{x},\overline{\ell},t)\\ \textbf{endcase}\end{array}$$

then E contains the equation:

$$s(\overline{x},\overline{\ell},t) = \mathcal{F}(\overline{x},\overline{\ell},t)$$

where the set of cases of $\mathcal{F}(\overline{x},\overline{\ell},t)$ is given by the smallest set containing the following elements:

- The case:

$$top'(\overline{x},\overline{\ell}) \implies s'(\overline{x}+\overline{\delta},\overline{\ell},t)$$

  where $top'(\overline{x},\overline{\ell})$ denotes the position of the cells whose inputs s are connected to the outputs of other cells

- The case:

$$top''(\overline{x},\overline{\ell}) \wedge p_i(\overline{x},\overline{\ell},t) \implies it_i(\overline{x},\overline{\ell},t)$$

  for each $1 \leq i \leq n$, where $top''(\overline{x},\overline{\ell})$ denotes the position of the cells whose input s are on the boundary of the circuit (i.e. not connected to the outputs of other cells)

The correctness of the equations in E follows from (2.2) (2.3), (2.6), (2.15), (2.16), (5.1) (6.2), (6.3), and (6.4).

That E is acyclic follows easily from the restrictions on the cell computations in acyclic systolic circuits (see section 6.2). □

6.5.1 First Step of Verification of The String Matching Circuit: Describing it by Recurrence Equations

We will show how the implementation of the string matching circuit, introduced in section 3.2 can be described as an acyclic system of recurrence equations.

Let $\mathcal{F}_{out_1}$,$\mathcal{F}_{out_2}$, and $\mathcal{F}_{loc}$ be the cell computations defining the values of out_1, out_2, and loc, as described by equations (3.35), (3.36), and (3.37) respectively. Let $\mathcal{F}^*_{out_1}$,$\mathcal{F}^*_{out_2}$, and $\mathcal{F}^*_{loc}$ be the the results of normalization of $\mathcal{F}_{out_1}$,$\mathcal{F}_{out_2}$, and $\mathcal{F}_{loc}$. Then:

$$\begin{array}{l}\mathcal{F}^*_{out_1}(x,\ell_1,\ell_2,t) = \textbf{case}\\ \qquad\qquad \left\{\begin{array}{l} t - x \bmod 2 = 0 \\ -2\ell_1 \leq x \leq 2\ell_2 + 1 \end{array}\right\} \implies out_1(x-1,\ell_1,\ell_2,t-1)\\ \qquad\qquad \textbf{endcase}\end{array}$$

$$\mathcal{F}^*_{out_2}(x,\ell_1,\ell_2,t) = \textbf{case}\ \left\{\begin{array}{c} t - x\ mod\ 2 = 0 \\ -2\ell_1 - 1 \le x \le 2\ell_2 \end{array}\right\} \implies out_2(x+1,\ell_1,\ell_2,t-1)\ \textbf{endcase}$$

$$\mathcal{F}^*_{loc}(x,\ell_1,\ell_2,t) =$$
$$\textbf{case}$$
$$\left\{\begin{array}{c} x < 0 \\ t = -x+1 \\ -2\ell_1 - 1 \le x \le 2\ell_2 \end{array}\right\} \implies g_2(loc(x+1,\ell_1,\ell_2,t-1))$$
$$\left\{\begin{array}{c} 0 < x \\ t = x+1 \\ -2\ell_1 \le x \le 2\ell_2 + 1 \end{array}\right\} \implies g_2(loc(x-1,\ell_1,\ell_2,t-1))$$
$$\left\{\begin{array}{c} -x+1 < t \\ x+1 < t \\ t+x+1\ mod\ 2 = 0 \\ -2\ell_1 \le x \le 2\ell_2 \end{array}\right\} \implies g_3(\ out_1(x-1,\ell_1,\ell_2,t-1),\ out_2(x+1,\ell_1,\ell_2,t-1),\ loc(x-1,\ell_1,\ell_2,t-1),\ loc(x+1,\ell_1,\ell_2,t-1),\ loc(x,\ell_1,\ell_2,t-2))$$
$$\textbf{endcase}$$

The acyclic system of recurrence equations describing the implementation of the circuit is given by:

$$out_1(x,\ell_1,\ell_2,t) = \textbf{case}$$
$$\left\{\begin{array}{c} 0 \le t < 1 \\ -2\ell_1 - 1 \le x \le 2\ell_2 + 1 \\ -2\ell_1 - 1 \le x \le -3 \\ (x+1)\ mod\ 2 = 0 \end{array}\right\} \implies a\left(-\tfrac{x+1}{2}\right)$$
$$\left\{\begin{array}{c} 1 \le t \\ -2\ell_1 - 1 \le x \le 2\ell_2 + 1 \end{array}\right\} \implies F^*_{out_1}(x,\ell_1,\ell_2,t)$$
$$\textbf{endcase}$$
$$out_2(x,\ell_1,\ell_2,t) = \textbf{case}$$
$$\left\{\begin{array}{c} 0 \le t < 1 \\ -2\ell_1 - 1 \le x \le 2\ell_2 + 1 \\ 3 \le x \le 2\ell_2 + 1 \\ (x+1)\ mod\ 2 = 0 \end{array}\right\} \implies b\left(\tfrac{x-1}{2}\right)$$
$$\left\{\begin{array}{c} 1 \le t \\ -2\ell_1 - 1 \le x < 2\ell_2 + 1 \end{array}\right\} \implies \mathcal{F}^*_{out_2}(x,\ell_1,\ell_2,t)$$
$$\textbf{endcase}$$

$$
\begin{aligned}
loc(x,\ell_1,\ell_2,t) \;=\;& \textbf{case} \\
& \left\{ \begin{array}{c} 0 \le t < 2 \\ -2\ell_1 - 1 \le x \le 2\ell_2 + 1 \\ x = 0 \end{array} \right\} \implies g_1 \\
& \left\{ \begin{array}{c} 2 \le t \\ -2\ell_1 - 1 \le x \le 2\ell_2 + 1 \end{array} \right\} \implies F^*_{loc}(x,\ell_1,\ell_2,t) \\
& \textbf{endcase} \\
out_3(x,\ell_1,\ell_2,t) \;=\;& loc(x,\ell_1,\ell_2,t) \\
out_4(x,\ell_1,\ell_2,t) \;=\;& loc(x,\ell_1,\ell_2,t) \\
in_1(x,\ell_1,\ell_2,t) \;=\;& \textbf{case} \\
& \left\{ -2\ell_1 \le x \le 2\ell_2 + 1 \right\} \implies out_1(x-1,\ell_1,\ell_2,t) \\
& \textbf{endcase} \\
in_2(x,\ell_1,\ell_2,t) \;=\;& \textbf{case} \\
& \left\{ -2\ell_1 - 1 \le x \le 2\ell_2 \right\} \implies out_1(x+1,\ell_1,\ell_2,t) \\
& \textbf{endcase} \\
in_3(x,\ell_1,\ell_2,t) \;=\;& \textbf{case} \\
& \left\{ -2\ell_1 \le x \le 2\ell_2 + 1 \right\} \implies out_3(x-1,\ell_1,\ell_2,t) \\
& \textbf{endcase} \\
in_4(x,\ell_1,\ell_2,t) \;=\;& \textbf{case} \\
& \left\{ -2\ell_1 - 1 \le x \le 2\ell_2 \right\} \implies out_4(x+1,\ell_1,\ell_2,t) \\
& \textbf{endcase}
\end{aligned}
$$

6.6 Equality Checking

In this section we show that, given two acyclic systems E_1 and E_2 of recurrence equations with a signature $\mathcal{SIG}$, two function variables f_1 and f_2 in E_1 and E_2, and a conjunction $p(\overline{x}_1,\overline{x}_2)$ of linear predicates, it is decidable to check the validity of:

$$\mathcal{SPEC} \;\models\; p(\overline{x}_1,\overline{x}_2) \longrightarrow (f_1(\overline{x}_1) = f_2(\overline{x}_2)) \tag{6.14}$$

where $\mathcal{SPEC}$ is the algebraic specification of all $\mathcal{SIG}$-algebras. This is done in several steps.

In section 6.6.1 we introduce the class of Presburger formulas.

In section 6.6.4 (lemma 46) we show that, given an acyclic system of recurrence equations E over a signature $\mathcal{SIG}$ and a function variable f in E, there is a Presburger formula $P(\overline{x})$ such that for each value $\overline{\alpha}$ of $\overline{x}$:

$$P(\overline{\alpha}) \quad \text{iff} \quad f(\overline{\alpha}) = \perp$$

for each $\mathcal{SIG}$-algebra A.

In section 6.6.5 (lemma 49) we show that, given two acyclic systems E_1 and E_2 of recurrence equations with a signature $\mathcal{SIG}$, two function variables f_1 and f_2 in E_1 and E_2, there is a Presburger formula $P(\overline{x}_1,\overline{x}_2)$ such that for each value $\overline{\alpha}_1$ and $\overline{\alpha}_2$ of $\overline{x}_1$ and

$\overline{x}_2$, if $f_1(\overline{\alpha}_1) \neq \perp$ and $f_2(\overline{\alpha}_2) \neq \perp$, then:

$$P(\overline{\alpha}_1, \overline{\alpha}_2) \quad \text{iff} \quad \mathcal{SPEC} \models (f_1(\overline{\alpha}_1) = f_2(\overline{\alpha}_2))$$

In section 6.6.6 we combine the results of sections 6.6.4 and 6.6.5 to show that (theorem 50) there is a Presburger formula which is equivalent to the formula in (6.14). The decidability of the validity of formulas of the form of (6.14) follows (corollary 51) from the decidability of Presburger formulas. In sections 6.6.2 and 6.6.3 we introduce some auxiliary definitions and lemmas which we will use in the proofs of the other sections of this chapter.

6.6.1 Presburger's Arithmetic

The class of *Presburger formulas* is the class of first order formulas over the structure $< I, +, -, <, 0, 1 >$ of integers. The class of *Presburger terms* is given by the following:

- Each integer is a Presburger term.
- Each variable over the integers is a Presburger term.
- If $t_1(\overline{x})$ and $t_2(\overline{x})$ are Presburger terms then $t_1(\overline{x}) + t_2(\overline{x})$ and $-t_1(\overline{x})$ are Presburger terms.

The class of *Presburger formulas* is given by the following:

- If $t_1(\overline{x})$ and $t_2(\overline{x})$ are Presburger terms then $t_1(\overline{x}) = t_2(\overline{x})$ and $t_1(\overline{x}) < t_2(\overline{x})$ are Presburger formulas.
- If $P_1(\overline{x})$ and $P_2(\overline{x})$ are Presburger formulas then $\neg P_1(\overline{x})$, $P_1(\overline{x}) \wedge P_2(\overline{x})$, $P_1(\overline{x}) \vee P_2(\overline{x})$, and $P_1(\overline{x}) \longrightarrow P_2(\overline{x})$ are Presburger formulas.
- If $P(\overline{x}, y)$ is a Presburger formula then $\exists y.\ P(\overline{x}, y)$ and $\forall y.\ P(\overline{x}, y)$ are Presburger formulas.

The class of Presburger formulas was shown to be decidable by Presburger [End72, Rab77]. This means that given a Presburger formula $P(\overline{x})$ it is decidable to check whether $P(\overline{x})$ is valid or not.

6.6.2 Constrained Integer Lattices

In this section we introduce the notion of *integer lattices*, and then prove some properties about them. These will be used in the proofs of the later sections in the chapter.

A *(constrained integer) lattice* $L(\overline{x})$, with *origin* $\overline{x}$, *basis* $\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$, where $\overline{\delta}_i$ is a tuple of integers, and *constraint* $p(\overline{y})$, where $p(\overline{y})$ is a conjunction of linear inequalities and equalities is defined by:

$$L(\overline{x}) = \begin{cases} \neg p(\overline{x}) & \emptyset \\ p(\overline{x}) & \{\overline{\mu} + \overline{u}_i \,;\, \overline{\mu} \in L(\overline{x} + \overline{\delta}_i) \text{ for some } 1 \leq i \leq m\} \cup \{\overline{0}\} \end{cases} \tag{6.15}$$

where u_i is a unit vector. Intuitively if $\overline{\mu} \in L(\overline{x})$ then $\overline{x}+\overline{\mu}\cdot\overline{\delta}$ is an integer point inside the polytope defined by $p(\overline{y})$, which can be reached from $\overline{x}$ by a finite number of steps inside the polytope, moving in the direction of one of $\overline{\delta}_1, \ldots, \overline{\delta}_n$ at each step. The *boundary* $B(\overline{x})$ of the lattice is defined by:

$$B(\overline{x}) = \begin{cases} \neg p(\overline{x}) & \{\overline{0}\} \\ p(\overline{x}) & \{\overline{\mu}\ ;\ \overline{\mu} \notin L(\overline{x}) \text{ and there is a } \overline{\nu} \text{ such that } \overline{\mu} = \overline{\nu} + \overline{u}_i \text{ and } \\ & \quad \overline{\nu} \in L(\overline{x}), \text{ for some } 1 \leq i \leq m\} \end{cases} \tag{6.16}$$

Intuitively if $\overline{\mu} \in B(\overline{x})$ then $\overline{x} + \overline{\mu} \cdot \overline{\delta}$ is a point outside the polytope defined by $p(\overline{y})$, which can be reached from a point on the lattice by one step, moving in the direction of one of $\overline{\delta}_1, \ldots, \overline{\delta}_n$. The *closure* $C(\overline{x})$ of the lattice is defined by:

$$C(\overline{x}) = L(\overline{x}) \cup B(\overline{x}) \tag{6.17}$$

The *predecessor relation* $\prec_p$, on $C(\overline{x})$ is defined in the following way:

$$\overline{\mu}_1 \prec_p \overline{\mu}_2$$
$$\text{iff}$$
$$(\overline{\mu}_1 = \overline{\mu}_2 + \overline{\nu}\ , \text{ where } \overline{\nu} \text{ is a positive vector}) \wedge (\overline{\mu}_2 \in L(\overline{x}))$$

As an example, consider a lattice $L(x_1, x_2)$, with origin $< x_1, x_2 >$, basis $<< -1, -1 >, < 1, -1 >, < 0, -2 >>$, and constraint $(x_1 \leq x_2 + 2) \wedge (x_1 + x_2 \geq 10)$, illustrated in figure 6.1. The points marked by filled circles are of the form $< 9 - \mu_1 + \mu_2, 15 - \mu_1 - \mu_2 - 2\mu_3 >$, where $< \mu_1, \mu_2, \mu_3 > \in L(9, 15)$. The points marked with hollow circles are of the form $< 9 - \mu_1 + \mu_2, 15 - \mu_1 - \mu_2 - 2\mu_3 >$, where $< \mu_1, \mu_2, \mu_3 > \in B(9, 15)$.

Let $p(\overline{x})$ be a conjunction of linear inequalities and equalities of the form:

$$\bigwedge_{i=1}^{n_1} (\overline{\alpha}_i \cdot \overline{x} + \beta_i \leq 0) \wedge \bigwedge_{i=n_1+1}^{n_2} (\overline{\alpha}_i \cdot \overline{x} + \beta_i = 0)$$

Consider a two-dimensional integer vector $\overline{\delta}$. Then we say that $p(\overline{x})$ is *finite* with respect to $\overline{\delta}$ iff there is no positive vector $\overline{\mu}$ such that;

$$\left(\bigwedge_{i=1}^{n_1} \left(\overline{\alpha}_i \cdot \overline{\mu} \cdot \overline{\delta} \leq 0\right)\right) \wedge \left(\bigwedge_{i=n_1+1}^{n_2} \left(\overline{\alpha}_i \cdot \overline{\mu} \cdot \overline{\delta} = 0\right)\right)$$

A conjunction of linear predicates $p_1(\overline{x}) \wedge p_2(\overline{x})$, where $p_1(\overline{x})$ is a conjunction of linear inequalies and equalities and $p_2(\overline{x})$ is a conjunction of linear modulo predicates, is said to be *finite* with respect to a two-dimensional integer vector $\overline{\delta}$ if $p_1(\overline{x})$ is finite with respect to $\overline{\delta}$.

In the following proofs we denote unit vectors by $\overline{u}_i$, $\overline{u}_j$, $\overline{u}_k$, etc.

Proposition 31 *Let $L(\overline{x})$ be a lattice with origin $\overline{x}$, basis $\overline{\delta}$, and constraint $p(\overline{y})$. Then:*

$$\overline{\mu} \in L(\overline{x}) \longrightarrow p(\overline{x} + \overline{\mu} \cdot \overline{\delta})$$

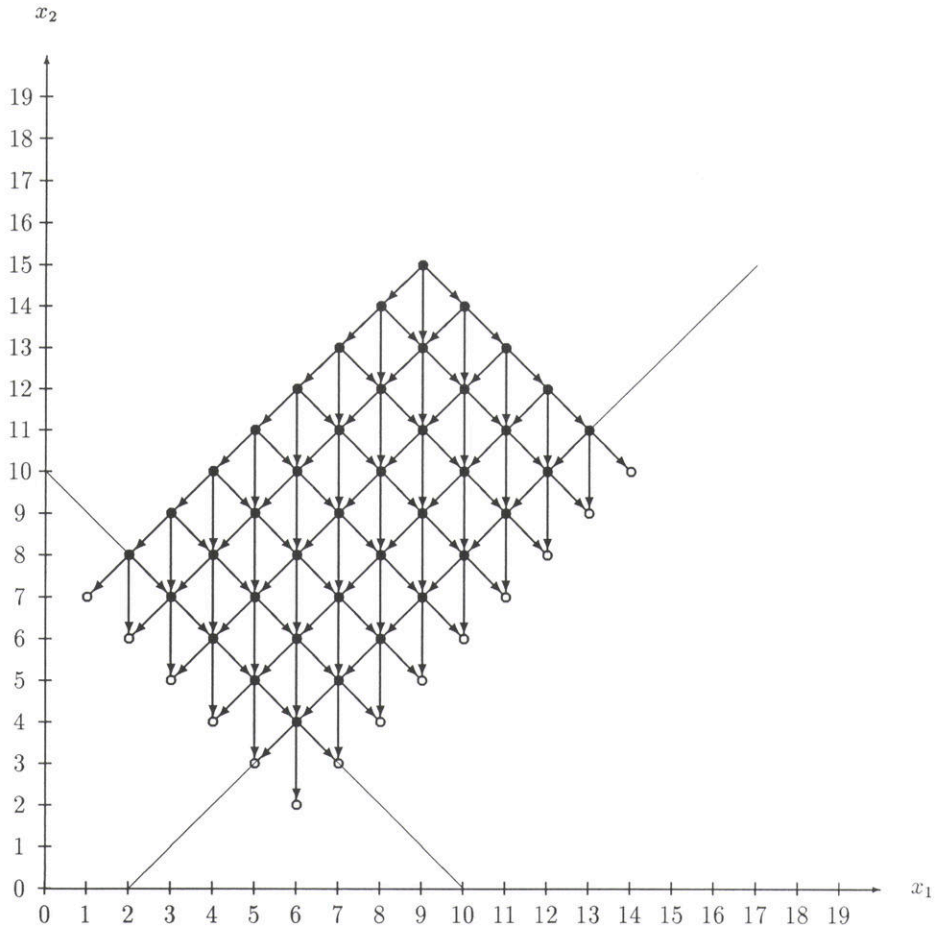

Figure 6.1: The lattice and the boundary of the lattice with origin $< 9, 15 >$, basis $<< -1, -1 >, < 1, -1 >, < 0, -2 >>$, and constraint $(x_1 \leq x_2 + 2) \wedge (x_1 + x_2 \geq 10)$.

Proof: By induction on $\overline{\mu}$:
Base Case: If $\overline{\mu} = \overline{0}$. From (6.15):

$$\overline{0} \in L(\overline{x}) \longrightarrow p(\overline{x}) \longrightarrow p(\overline{x} + \overline{\mu} \cdot \overline{\delta})$$

Induction Step: If $\overline{\mu} > \overline{0}$. Let $\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$. Suppose that $\overline{\mu} \in L(\overline{x})$. From (6.15) it follows that there is a $\overline{\nu}$ such that $\overline{\mu} = \overline{\nu} + \overline{u}_j$, for some $1 \leq j \leq m$ and such that $\overline{\nu} \in L(\overline{x} + \overline{\delta}_j)$. By the induction hypothesis it follows that $p(\overline{x} + \overline{\delta}_j + \overline{\nu} \cdot \overline{\delta})$ is true. It follows that:

$$p(\overline{x} + \overline{\mu} \cdot \overline{\delta}) = p(\overline{x} + (\overline{\nu} + \overline{u}_j) \cdot \overline{\delta}) = p(\overline{x} + \overline{\delta}_j + \overline{\nu} \cdot \overline{\delta}) = \text{true}$$

□

Proposition 32 *Let $B(\overline{x})$ be the boundary of a lattice $L(\overline{x})$ with origin $\overline{x}$, basis $\overline{\delta}$, and constraint $p(\overline{y})$. Then:*

$$\overline{\mu} \in B(\overline{x}) \longrightarrow \neg p(\overline{x} + \overline{\mu} \cdot \overline{\delta})$$

Proof: By induction on $\overline{\mu}$:
Base Case: If $\overline{\mu} = \overline{0}$. From (6.16):

$$\overline{0} \in B(\overline{x}) \longrightarrow \neg p(\overline{x}) \longrightarrow \neg p(\overline{x} + \overline{\mu} \cdot \overline{\delta})$$

Induction Step: Let $\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$. There are two cases:
1. If $\overline{\mu} = \overline{u}_j$, for some $1 \leq j \leq m$. From (6.16):

$$\overline{\mu} \notin L(\overline{x}) \tag{6.18}$$

and:

$$\overline{0} \in L(\overline{x})$$

From (6.15):

$$p(\overline{x}) = \text{true} \tag{6.19}$$

From (6.18), (6.19), and (6.15):

$$\overline{0} \notin L(\overline{x} + \overline{\delta}_j)$$

From (6.16):

$$p(\overline{x} + \overline{\mu} \cdot \overline{\delta}) = p(\overline{x} + \overline{\delta}_j) = \text{false}$$

2. $\overline{\mu} > \overline{u}_l$, for some $1 \leq l \leq m$. Suppose that:

$$\overline{\mu} \in B(\overline{x})$$

From (6.16) it follows that:

$$\overline{\mu} \notin L(\overline{x}) \tag{6.20}$$

and that:

$$\overline{\mu} - \overline{u}_j \in L(\overline{x}) \tag{6.21}$$

for some $1 \leq j \leq m$. From (6.15):

$$p(\overline{x}) = \text{true} \tag{6.22}$$

From (6.20), (6.22), and (6.15):

$$\overline{\mu} - \overline{u}_k \notin L(\overline{x} + \overline{\delta}_k) \tag{6.23}$$

for each $1 \leq k \leq m$. As $\overline{\mu} > \overline{u}_l$ we know that $\overline{\mu} - \overline{u}_j > 0$. From (6.21), (6.22), and (6.15), it follows that there is a $\overline{u}_k$, where $1 \leq k \leq m$, such that:

$$\overline{\mu} - \overline{u}_j - \overline{u}_k \in L(\overline{x} + \overline{\delta}_k) \tag{6.24}$$

From (6.24), and (6.15):

$$p(\overline{x} + \overline{\delta}_k) = \text{true} \tag{6.25}$$

From (6.23), (6.24), (6.25), and (6.15):

$$\overline{\mu} - \overline{u}_k \in B(\overline{x} + \overline{\delta}_k)$$

By the induction hypothesis:

$$B(\overline{x} + \overline{\mu} \cdot \overline{\delta}) = B(\overline{x} + \overline{\delta}_k + (\overline{\mu} - \overline{u}_k) \cdot \overline{\delta}) = \text{false}$$

□

Proposition 33 *Let $L(\overline{x})$ be a lattice with origin $\overline{x}$, basis $\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$, and constraint $p(\overline{y})$. Let $B(\overline{x})$ be the boundary of the lattice. Then:*

- $p(\overline{x}) \longrightarrow \left(\overline{\mu} \in L(\overline{x}) \text{ iff } \left((\overline{\mu} = \overline{0}) \vee \left(\bigvee_{i=1}^{m} \left((\overline{\mu} - \overline{u}_i) \in L(\overline{x} + \overline{\delta}_i)\right)\right)\right)\right)$
- $p(\overline{x}) \longrightarrow \left(\overline{\mu} \in B(\overline{x}) \text{ iff } \left(\bigvee_{i=1}^{m} \left((\overline{\mu} - \overline{u}_i) \in B(\overline{x} + \overline{\delta}_i)\right)\right)\right)$

Proof:

- The proof of the first claim follows directly from the definition of a lattice (see (6.15)).
- We will prove the second claim. Suppose that:

$$p(\overline{x}) = \text{true} \tag{6.26}$$

We will show the equivalence in both directions.
(⟶): Suppose that:

$$\overline{\mu} \in B(\overline{x}) \tag{6.27}$$

There are three cases:
1. If $\overline{\mu} = \overline{0}$. From (6.16) it follows that $p(\overline{x})$ is false, which is a contradiction.

2. If $\overline{\mu} = \overline{u}_j$, for some $1 \leq j \leq m$. From (6.26), (6.27), and (6.16):

$$\overline{\mu}_j \notin L(\overline{x})$$

From (6.15):

$$\overline{0} \notin L(\overline{x} + \overline{\delta}_j)$$

and:

$$p(\overline{x} + \overline{\delta}_j) = \text{false}$$

From (6.16):

$$\overline{0} \in B(\overline{x} + \overline{\delta}_j)$$

so:

$$\overline{\mu} - \overline{u}_j \in B(\overline{x} + \overline{\delta}_j)$$

and:

$$\bigvee_{i=1}^{m} \left((\overline{\mu} - \overline{u}_i) \in B(\overline{x} + \overline{\delta}_i) \right)$$

3. If $\overline{\mu} > \overline{u}_l$, for some $1 \leq l \leq m$. From (6.27) and (6.16) it follows that:

$$\overline{\mu} \notin L(\overline{x}) \tag{6.28}$$

and that there is a $\overline{\nu}_1$ such that:

$$\overline{\mu} = \overline{\nu}_1 + \overline{u}_j \tag{6.29}$$

for some $1 \leq j \leq m$ and:

$$\overline{\nu}_1 \in L(\overline{x}) \tag{6.30}$$

From the fact that $\overline{\mu} > \overline{u}_l$ it follows that $\overline{\mu} > \overline{u}_j$, and:

$$\overline{\nu}_1 > \overline{0} \tag{6.31}$$

From (6.26), (6.30), (6.31), and (6.15) it follows that there is a $\overline{\nu}_2$ such that:

$$\overline{\nu}_1 = \overline{\nu}_2 + \overline{u}_k \tag{6.32}$$

for some $1 \leq k \leq m$, and such that:

$$\overline{\nu}_2 \in L(\overline{x} + \overline{\delta}_k) \tag{6.33}$$

As $\overline{\mu} > \overline{\nu}_1 > \overline{u}_k$ it is clear that there is a $\overline{\nu}_3$ such that:

$$\overline{\mu} = \overline{\nu}_3 + \overline{u}_k \tag{6.34}$$

From (6.29), (6.32), and (6.34):

$$\overline{\nu}_3 \;=\; \overline{\nu}_2 + \overline{u}_j \tag{6.35}$$

From (6.26), (6.28), and (6.34) and (6.15):

$$\overline{\nu}_3 \;\notin\; L(\overline{x} + \overline{\delta}_k) \tag{6.36}$$

From (6.33) and (6.15):

$$p(\overline{x} + \overline{\delta}_k) \;=\; \text{true} \tag{6.37}$$

From (6.33), (6.35), (6.36), and (6.16):

$$\overline{\nu}_3 \;\in\; B(\overline{x} + \overline{\delta}_k) \tag{6.38}$$

From (6.34) and (6.38):

$$\overline{\mu} - \overline{u}_k \;\in\; B(\overline{x} + \overline{\delta}_k) \tag{6.39}$$

So:

$$\bigvee_{i=1}^{m} \Big((\overline{\mu} - \overline{u}_i) \in B(\overline{x} + \overline{\delta}_i) \Big)$$

(⟵): Suppose that:

$$\overline{\mu} - \overline{u}_j \;\in B(\overline{x} + \overline{\delta}_j) \tag{6.40}$$

for some $1 \leq j \leq m$. There are two cases:
1. If $\overline{\mu} = \overline{u}_j$. We have:

$$\overline{0} \;\in B(\overline{x} + \overline{\delta}_j) \tag{6.41}$$

From (6.16):

$$p(\overline{x} + \overline{\delta}_j) \;=\; \text{false}$$

From (6.15):

$$\overline{0} \;\notin\; L(\overline{x} + \overline{\delta}_j)$$

From (6.26) and (6.15):

$$\overline{u}_j \;\notin\; L(\overline{x}) \tag{6.42}$$

From (6.26) and (6.15):

$$\overline{0} \;\in\; L(\overline{x}) \tag{6.43}$$

From (6.26), (6.42), (6.43), and (6.16):

$$\overline{u}_j \;\in\; B(\overline{x})$$

So:

$$\overline{\mu} \in B(\overline{x})$$

2. If $\overline{\mu} > \overline{u}_j$. Let $\overline{\nu}_1$ be such that:

$$\overline{\mu} = \overline{\nu}_1 + \overline{u}_j \tag{6.44}$$

From (6.40):

$$\overline{\nu}_1 \in B(\overline{x} + \overline{\delta}_j) \tag{6.45}$$

From the fact that $\overline{\mu} > \overline{u}_j$ it follows that:

$$\overline{\nu}_1 > \overline{0} \tag{6.46}$$

From (6.45), (6.46), and (6.16):

$$p(\overline{x} + \overline{\delta}_j) = \text{true} \tag{6.47}$$

From (6.45), (6.47), and (6.16) it follows that:

$$\overline{\nu}_1 \notin L(\overline{x} + \overline{\delta}_j) \tag{6.48}$$

and that there is a $\overline{\nu}_2$ such that:

$$\overline{\nu}_1 = \overline{\nu}_2 + \overline{u}_k \tag{6.49}$$

for some $1 \leq k \leq m$, and:

$$\overline{\nu}_2 \in L(\overline{x} + \overline{\delta}_j) \tag{6.50}$$

Let:

$$\overline{\nu}_3 = \overline{\nu}_2 + \overline{u}_j \tag{6.51}$$

From (6.50), (6.51) and (6.15):

$$\overline{\nu}_3 \in L(\overline{x}) \tag{6.52}$$

From propositions 31 and 32, and equations (6.44) and (6.45):

$$\overline{\mu} \notin L(\overline{x}) \tag{6.53}$$

From (6.44), (6.49), and (6.51):

$$\overline{\mu} = \overline{\nu}_3 + \overline{u}_k \tag{6.54}$$

From (6.26), (6.52), (6.53), (6.54), and (6.16):

$$\overline{\mu} \in B(\overline{x})$$

□

Corollary 34 *Let $L(\overline{x})$ be a lattice with origin $\overline{x}$, basis $\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$, and constraint $p(\overline{y})$. Let $B(\overline{x})$ be the boundary of the lattice. Let $P(\overline{x})$ be any predicate. Then for each value $\overline{\alpha}$ of $\overline{x}$:*

- *If $p(\overline{\alpha})$ then:*

$$\forall \overline{\mu} \in L(\overline{\alpha}).\ P(\overline{\mu})$$

iff:

$$P(\overline{0})$$

and:

$$\forall \overline{\mu} \in L(\overline{\alpha} + \overline{\delta}_i).\ P(\overline{\mu} + \overline{u}_i)$$

for $1 \leq i \leq m$.

- *If $p(\overline{\alpha})$ then:*

$$\forall \overline{\mu} \in B(\overline{\alpha}).\ P(\overline{\mu})$$

iff:

$$\forall \overline{\mu} \in B(\overline{\alpha} + \overline{\delta}_i).\ P(\overline{\mu} + \overline{u}_i)$$

for $1 \leq i \leq m$.

- *If $p(\overline{\alpha})$ then:*

$$\exists \overline{\mu} \in L(\overline{\alpha}).\ P(\overline{\mu})$$

iff:

$$P(\overline{0})$$

or:

$$\exists \overline{\mu} \in L(\overline{\alpha} + \overline{\delta}_i).\ P(\overline{\mu} + \overline{u}_i)$$

for some $1 \leq i \leq m$.

- *If $p(\overline{\alpha})$ then:*

$$\exists \overline{\mu} \in B(\overline{\alpha}).\ P(\overline{\mu})$$

iff:

$$\exists \overline{\mu} \in B(\overline{\alpha} + \overline{\delta}_i).\ P(\overline{\mu} + \overline{u}_i)$$

for some $1 \leq i \leq m$.

Proof: The proof follows immediately from proposition 33.

Proposition 35 *Consider a lattice $L(\overline{x})$ with origin $\overline{x}$, basis $\overline{\delta}$, and constraint $p(\overline{y})$. Let $B(\overline{x})$ and $C(\overline{x})$ be the boundary and the closure of the lattice, and let $\prec_p$ be the predecessor relation on $C(\overline{\gamma})$, for some value $\overline{\gamma}$ of $\overline{x}$. Then $\prec_p$ is a partial order, and the bottom elements of $\prec_p$ are the elements of $B(\overline{\gamma})$.*

Proof: The irreflexitivity and transitivity of $\prec_p$ is clear from its definition. Let $\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$. Suppose that $\overline{\mu}_1 \in B(\overline{\gamma})$. From the definition of $\prec_p$ it is clear that there is no $\overline{\mu}_2$ such that $\overline{\mu}_2 \prec_p \overline{\mu}_1$. Suppose that $\overline{\mu} \in L(\overline{\gamma})$. From (6.15) it follows that $p(\overline{\gamma})$ is true. Consider $\overline{\mu} + \overline{u}_j$, for any $1 \leq j \leq m$. There are two cases: The first case is when $\overline{\mu} + \overline{u}_j \notin L(\overline{\gamma})$. From (6.16) it follows that $\overline{\mu} + \overline{u}_j \in B(\overline{\gamma})$ which means that $\overline{\mu} + \overline{u}_j \in C(\overline{\gamma})$ and $\overline{\mu} + \overline{u}_j \prec_p \overline{\mu}$. The second case is when $\overline{\mu} + \overline{u}_j \in L(\overline{\gamma})$ which means that $\overline{\mu} + \overline{u}_j \in C(\overline{\gamma})$ and $\overline{\mu} + \overline{u}_j \prec_p \overline{\mu}$. □

Proposition 36 *Consider a lattice $L(\overline{x})$ with origin $\overline{x}$, basis $\overline{\delta}$, and constraint $p(\overline{y})$, such that $p(\overline{y})$ is finite with respect to $\overline{\delta}$. Let $C(\overline{x})$ be the closure of the lattice. Then for each value $\overline{\gamma}$ of $\overline{x}$, if $\prec_p$ is the the predecessor relation on $C(\overline{\gamma})$, then $\prec_p$ is well-founded.*

Proof: Let $\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$. Let $p(\overline{x})$ be of the form:

$$p(\overline{x}) \;=\; p_1(\overline{x}) \wedge p_2(\overline{x})$$

where:

$$p_1(\overline{x}) \;=\; \bigwedge_{i=1}^{n_1} (\overline{\alpha}_i \cdot \overline{x} + \beta_i \leq 0)$$

and:

$$p_2(\overline{x}) \;=\; \bigwedge_{i=n_1+1}^{n_2} (\overline{\alpha}_i \cdot \overline{x} + \beta_i = 0)$$

That $p(\overline{x})$ is finite with respect to $\overline{\delta}$ means that we have no positive $\overline{\mu}$ such that:

$$\left(\bigwedge_{i=1}^{n_1} \left(\overline{\alpha}_i \cdot \overline{\mu} \cdot \overline{\delta} \leq 0\right)\right) \wedge \left(\bigwedge_{i=n_1+1}^{n_2} \left(\overline{\alpha}_i \cdot \overline{\mu} \cdot \overline{\delta} = 0\right)\right) \tag{6.55}$$

Let $\prec_p$ be the predecessor relation on $C(\overline{\gamma})$, for some value $\overline{\gamma}$ of $\overline{x}$. Suppose that $\prec_p$ is not well-founded. It follows that there is an infinite decreasing chain:

$$\overline{\mu}_1 \succ_p \overline{\mu}_2 \succ_p \overline{\mu}_3 \succ \cdots \tag{6.56}$$

where $\overline{\mu}_{j+1} = \overline{\mu}_j + \overline{u}_k$, for some $1 \leq k \leq m$, and $\overline{\mu}_j \in C(\overline{\gamma})$, for $j \geq 0$. From proposition 35 it follows that $\overline{\mu}_j \in L(\overline{\gamma})$, and from proposition 31 it follows that $p(\overline{\gamma} + \overline{\mu}_j \cdot \overline{\delta})$ is true, for $j \geq 0$.

It is clear that $p_2(\overline{\gamma} + \overline{\mu}_j \cdot \overline{\delta})$ will be true for $j \geq 0$. This means that:

$$\bigwedge_{i=n_1+1}^{n_2} \left(\overline{\alpha}_i \cdot \overline{\mu}_j \cdot \overline{\delta} = 0\right) \tag{6.57}$$

for $j \geq 0$.

Now consider any $\overline{\mu}_j$ and $\overline{\mu}_k$, where $j < k$. Then it is true that:

$$\bigvee_{i=1}^{n_1} \left(\overline{\alpha}_i \cdot \overline{\mu}_j \cdot \overline{\delta} < \overline{\alpha}_i \cdot \overline{\mu}_k \cdot \overline{\delta}\right) \tag{6.58}$$

because otherwise, we define the positive vector $\overline{\mu}$:

$$\overline{\mu} \;=\; \overline{\mu}_k - \overline{\mu}_j$$

From (6.57) it follows that:

$$\bigwedge_{i=n_1+1}^{n_2} \left(\overline{\alpha}_i \cdot \overline{\mu} \cdot \overline{\delta} = 0\right) \tag{6.59}$$

If the formula in (6.58) is not valid then:

$$\bigwedge_{i=1}^{n_1} \left(\overline{\alpha}_i \cdot \overline{\mu} \cdot \overline{\delta} \leq 0\right) \tag{6.60}$$

The formulas in (6.59) and (6.60) together are contradictory with the finiteness condition in (6.55). It follows that (6.58) is true. Consequently there is at least an $1 \leq i \leq n_1$, such that for each $j \geq 0$ there is a $k > j$ where:

$$\overline{\alpha}_i \cdot \overline{\mu}_j \cdot \overline{\delta} \;<\; \overline{\alpha}_i \cdot \overline{\mu}_k \cdot \overline{\delta}$$

It follows that for each $j \geq 0$ there is a $k > j$ such that:

$$\overline{\alpha}_i \cdot \overline{\gamma} + \beta_i + \overline{\alpha}_i \cdot \overline{\mu}_j \cdot \overline{\delta} \;<\; \overline{\alpha}_i \cdot \overline{\gamma} + \beta_i + \overline{\alpha}_i \cdot \overline{\mu}_k \cdot \overline{\delta} \;\leq\; 0$$

But this is impossible because $\overline{\alpha}_i \cdot \overline{\gamma} + \beta_i$ is finitely negative

This implies that there is no inifinte decreasing chain of the form of (6.56), which means that $\prec_p$ is a well-founded relation. □

Proposition 37 *Consider a lattice $L(\overline{x})$ with origin $\overline{x}$, basis $\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$, and constraint $p(\overline{y})$. Let $B(\overline{x})$ and $C(\overline{x})$ be the boundary and the closure of the lattice, and $\prec_p$ the predecessor relation on $C(\overline{\alpha})$ for some value $\overline{\alpha}$ of $\overline{x}$. Suppose that $p(\overline{x})$ is stable and not finite with respect to $\overline{\delta}$. Then if $p(\overline{\alpha})$ is true then there is an infinite decreasing chain:*

$$\overline{\mu}_1 \succ_p \overline{\mu}_2 \succ_p \overline{\mu}_3 \succ_p \cdots$$

where $\overline{\mu}_1 = \overline{0}$, $\overline{\mu}_{j+1} = \overline{\mu}_j + \overline{u}_k$, for some $1 \leq k \leq m$, and $p(\overline{\alpha} + \overline{\mu}_j \cdot \overline{\delta})$ is true for $j \geq 0$.

Proof: Let $p(\overline{x})$ be of the form:

$$p(\overline{x}) \;=\; \left(\bigwedge_{i=1}^{n_1} (\overline{\alpha}_i \cdot \overline{x} + \beta_i \leq 0)\right) \wedge \left(\bigwedge_{i=n_1+1}^{n_2} (\overline{\alpha}_i \cdot \overline{x} + \beta_i = 0)\right) \tag{6.61}$$

That $p(\overline{x})$ is not finite with respect to $\overline{\delta}$ means that there is a positive $\overline{\mu}$ such that:

$$\left(\bigwedge_{i=1}^{n_1} \left(\overline{\alpha}_i \cdot \overline{\mu} \cdot \overline{\delta} \leq 0\right)\right) \wedge \left(\bigwedge_{i=n_1+1}^{n_2} \left(\overline{\alpha}_i \cdot \overline{\mu} \cdot \overline{\delta} = 0\right)\right) \tag{6.62}$$

is true. From (6.61) and (6.62), it follows that, for each $\overline{x}$, if $p(\overline{x})$ is true then:

$$\left(\bigwedge_{i=1}^{n_1}\left(\overline{\alpha}_i\cdot\overline{x}+\beta_i+\overline{\alpha}_i\cdot\overline{\mu}\cdot\overline{\delta}\leq 0\right)\right)\wedge\left(\bigwedge_{i=n_1+1}^{n_2}\left(\overline{\alpha}_i\cdot\overline{x}+\beta_i+\overline{\alpha}_i\cdot\overline{\mu}\cdot\overline{\delta}=0\right)\right)$$
$$=\ p(\overline{x}+\overline{\mu}\cdot\overline{\delta})\ =\ \text{true}$$

It follows that for each $\overline{x}$, if $p(\overline{x})$ is true then:

$$p(\overline{x})\ \longrightarrow\ p(\overline{x}+\overline{\mu}\cdot\overline{\delta}) \tag{6.63}$$

We consider the sequence:

$$\overline{\mu}_0\succ_p\overline{\mu}_1\succ_p\overline{\mu}_2\succ_p\cdots\succ_p\overline{\mu}_n\succ_p\overline{\mu}_{n+1}\succ_p\overline{\mu}_{n+2}\succ_p\cdots\succ_p\overline{\mu}_{2n}\succ_p\overline{\mu}_{2n+1}\succ_p\overline{\mu}_{2n+2}\succ_p\cdots$$

where $\overline{\mu}_0=\overline{0}$, $\overline{\mu}_{j+1}=\overline{\mu}_j+\overline{u}_k$, $1\leq k\leq m$ for $0\leq j<n$, and $\overline{\mu}_{(l+1)\cdot n+j}=\overline{\mu}_{l\cdot n+j}+\overline{\mu}$, for $0\leq j<n$ and $0\leq l$.

Suppose that $p(\overline{\gamma})$ is true. We will use induction on l to show that $p(\overline{\gamma}+\overline{\mu}_{l\cdot n}\cdot\overline{\delta})$ is true for $l\geq 0$:

Base Case: If $l=0$ then:

$$p(\overline{\gamma}+\overline{\mu}_{l\cdot n}\cdot\overline{\delta})\ =\ p(\overline{\gamma}+\overline{\mu}_0)\ =\ p(\overline{\gamma})\ =\ \text{true}$$

Induction Step: Suppose that $p(\overline{\gamma}+\overline{\mu}_{l\cdot n})$ is true. From (6.63) it follows that:

$$p(\overline{\gamma}+\overline{\mu}_{(l+1)\cdot n})\ =\ p(\overline{\gamma}+\overline{\mu}_{l\cdot n}+\overline{\mu})\ =\ \text{true}$$

Now we will show that $p(\overline{\gamma}+\overline{\mu}_{l\cdot n+j}\cdot\overline{\delta})$ is true for $l\geq 0$ and $0<j<n$. Suppose that $p(\overline{\gamma}+\overline{\mu}_{l\cdot n+j}\cdot\overline{\delta})$ is false. We know from the induction proof above that $p(\overline{\gamma}+\overline{\mu}_{l\cdot n}\cdot\overline{\delta})$ and $p(\overline{\gamma}+\overline{\mu}_{l\cdot n+n}\cdot\overline{\delta})$ are true. We define:

$$\overline{\nu}_1\ =\ \overline{\mu}_{l\cdot n+j}-\overline{\mu}_{l\cdot n}$$

and:

$$\overline{\nu}_2\ =\ \overline{\mu}_{l\cdot n+n}-\overline{\mu}_{l\cdot n+j}$$

It is clear that $\overline{\nu}_1$ and $\overline{\nu}_2$ are positive vectors and that:

$$p(\overline{\gamma}+\overline{\mu}_{l\cdot n}\cdot\overline{\delta})\ \wedge\ \neg p(\overline{\gamma}+\overline{\mu}_{l\cdot n}+\overline{\nu}_1\cdot\overline{\delta})\ \wedge\ p(\overline{\gamma}+\overline{\mu}_{l\cdot n}+\overline{\nu}_1\cdot\overline{\delta}+\overline{\nu}_2\cdot\overline{\delta})$$

which is contradictory to the stability assumption. □

Proposition 38 *Let $L(\overline{x})$ be a lattice with origin $\overline{x}$, basis $\overline{\delta}=<\overline{\delta}_1,\ldots,\overline{\delta}_m>$, and constraint $p(\overline{y})$. Let $p(\overline{x})$ be stable with respect to $\overline{\delta}$. Then:*

$$p(\overline{x})\ \wedge\ p(\overline{x}+\overline{\mu}\cdot\overline{\delta})\ \longrightarrow\ \overline{\mu}\in L(\overline{x})$$

Proof: By induction on $\overline{\mu}$.
Base Case: If $\overline{\mu} = \overline{0}$. Then from (6.15):

$$p(\overline{x}) \longrightarrow \overline{\mu} \in L(\overline{x})$$

Induction Step: If $\overline{\mu} > \overline{0}$. Suppose that:

$$p(\overline{x}) \wedge p(\overline{x} + \overline{\mu} \cdot \overline{\delta}) = \text{true} \tag{6.64}$$

and:

$$\overline{\mu} \notin L(\overline{x}) \tag{6.65}$$

By (6.65) and (6.15) it follows that for each $1 \leq j \leq m$:

$$\overline{\mu} - \overline{u}_j \notin L(\overline{x} + \overline{\delta}_j)$$

From the induction hypothesis it follows that:

$$p(\overline{x} + \overline{\delta}_j) \wedge p(\overline{x} + \overline{\delta}_j + (\overline{\mu} - \overline{u}_j) \cdot \overline{\delta}) = \text{false} \tag{6.66}$$

Suppose that $p(\overline{x} + \overline{\delta}_j)$ is false. Then:

$$p(\overline{x}) \wedge \neg p(\overline{x} + \overline{\delta}_j) \wedge p(\overline{x} + \overline{\delta}_j + (\overline{\mu} - \overline{u}_j) \cdot \overline{\delta}) = \text{true}$$

which is contradictory to the stability assumption. It follows that:

$$p(\overline{x} + \overline{\delta}_j) = \text{true} \tag{6.67}$$

From (6.66) and (6.67) it follows that:

$$p(\overline{x} + \overline{\delta}_j + (\overline{\mu} - \overline{u}_j) \cdot \overline{\delta}) = p(\overline{x} + \overline{\mu} \cdot \overline{\delta}) = \text{false}$$

which is contradictory to (6.64). □

Proposition 39 *Let $L(\overline{x})$ be a lattice with origin $\overline{x}$, basis $\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$, and constraint $p(\overline{y})$. Let $p(\overline{x})$ be stable with respect to $\overline{\delta}$. Then:*

$$\overline{\mu} \in L(\overline{x}) \quad \textit{iff} \quad p(\overline{x}) \wedge p(\overline{x} + \overline{\mu} \cdot \overline{\delta})$$

Proof:
(⟶): Suppose that $\overline{\mu} \in L(\overline{x})$. From (6.15) it follows that $p(\overline{x})$ is true. By proposition 31 it follows that $p(\overline{x} + \overline{\mu} \cdot \overline{\delta})$ is true.
(⟵): Follows from proposition 38. □

Corollary 40 *Let $L(\overline{x})$ be a lattice with origin $\overline{x}$, basis $\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$, and constraint $p(\overline{y})$. Let $p(\overline{x})$ be stable with respect to $\overline{\delta}$. Then there are Presburger formulas $lf(\overline{x}, \overline{\mu})$ and $bf(\overline{x}, \overline{\mu})$ such that:*

$$\overline{\mu} \in L(\overline{x}) \quad \textit{iff} \quad lf(\overline{x}, \overline{\mu})$$

$$\overline{\mu} \in B(\overline{x}) \quad \textit{iff} \quad bf(\overline{x}, \overline{\mu})$$

We call $lf(\overline{x},\overline{\mu})$ and $bf(\overline{x},\overline{\mu})$ the *lattice formula* and the *boundary formula* respectively of $L(\overline{x})$.

Proof: From proposition 39 it follows that $lf(\overline{x},\overline{\mu})$ and $bf(\overline{x},\overline{\mu})$ can be defined as:

$$lf(\overline{x},\overline{\mu}) \;=\; p(\overline{x}) \;\wedge\; p(\overline{x}+\overline{\nu}\cdot\overline{\delta})$$

and:

$$bf(\overline{x},\overline{\mu}) \;=\; \left(\begin{array}{c} \neg p(\overline{x}) \;\wedge\; (\overline{\mu}=\overline{0}) \\ \vee \\ p(\overline{x}) \;\wedge\; \neg lf(\overline{x},\overline{\mu}) \;\wedge\; (lf(\overline{x},\overline{\mu}-\overline{u}_1) \;\vee\; \cdots \;\vee\; lf(\overline{x}+\overline{\mu}-\overline{u}_m)) \end{array}\right)$$

respectively. Both of these formulas are obviously Presburger formulas. □

Proposition 41 *Given a conjunction $p(\overline{x})$ of linear inequalities and equalities, and a two-dimensional integer vector $\overline{\delta}$, it is decidable to check whether $p(\overline{x})$ is finite with respect to $\overline{\delta}$.*

Proof: From the definition of finiteness, it can be easily shown that the probelm is reducible to the integer linear programming problem which is decidable (see section 5.6.4). □

6.6.3 Complexity of Systems of Recurrence Equations

We define a measure of *complexity* of systems of recurrence equations which we will use in the proofs of section 6.6.4. Let E be an acyclic system of recurrence equations. Let:

$$f(\overline{x}) \;=\; \mathcal{F}(\overline{x})$$

be an equation in E. Then the complexity of f in E, denoted $\mathcal{C}(E,f)$ is equal to the pair $< e,c >$, where e is the number of equations in E, and c is the number of cases in $\mathcal{F}(\overline{x})$.

6.6.4 Definedness of Systems of Recurrence Equations

In this section we show that, given an acyclic system of recurrence equations over a signature $\mathcal{SIG}$ and a function variable f in E, then there is a Presburger formula $P(\overline{x})$ such that for each value $\overline{\alpha}$ of $\overline{x}$, $P(\overline{\alpha})$ is true iff $f(\overline{\alpha}) = \perp$. The constuction of the formula $P(\overline{x})$ will be independent of the particular $\mathcal{SIG}$-algebra in which E is interpreted.

Let E be an acyclic system of recurrence equations. Let:

$$f(\overline{x}) \;=\; \mathcal{F}(\overline{x})$$

be an equation in E. Let $\mathcal{F}(\overline{x})$ be of the form:

$$\begin{array}{ll} \textbf{case} & \\ \quad p_1(\overline{x}) & \Longrightarrow\; t_1(\overline{x}) \\ \quad\quad\vdots & \\ \quad p_n(\overline{x}) & \Longrightarrow\; t_n(\overline{x}) \\ \textbf{endcase} & \end{array}$$

Suppose that $p_i(\overline{\alpha})$ is true, then *the guard defined by f and $\overline{\alpha}$* is $p_i(\overline{x})$ and *the result defined by f and $\overline{\alpha}$ is* $t_i(\overline{x})$.

Lemma 42 *Let:*

- *f be a function variable in an acyclic system of recurrence equations over a signature $\mathcal{SIG}$.*
- *$\overline{\alpha}$ be a tuple of integers.*
- *$p(\overline{x})$ and $t(\overline{x})$ be the guard and the result respectively defined by f and $\overline{\alpha}$, where $t(\overline{x})$ is a linear systolic term in which f occurs.*
- *$\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$ and $t_1(\overline{x}), \ldots, t_n(\overline{x})$ be the recursive vector and non-recursive terms respectively of f in $t(\overline{x})$.*
- *$L(\overline{x})$, $B(\overline{x})$, and $C(\overline{x})$, be the lattice, the boundary of the lattice, and the closure of the lattice with origin $\overline{x}$, basis $\overline{\delta}$, and constraint $p(\overline{y})$.*
- *$p(\overline{x})$ be finite with respect to $\overline{\delta}$.*

then:

$$f(\overline{\alpha}) = \perp$$

iff:

$$\exists \overline{\mu} \in B(\overline{\alpha}).\ f(\overline{\alpha} + \overline{\mu} \cdot \overline{\delta}) = \perp$$

or:

$$\exists \overline{\mu} \in L(\overline{\alpha}).\ t_i(\overline{\alpha} + \overline{\mu} \cdot \overline{\delta}) = \perp$$

for some $1 \leq i \leq n$.

Observe that the claim of the lemma is independent of the particular $\mathcal{SIG}$-algebra in which the equation system is interpreted.

Proof: Let $\prec_p$ be the predecessor relation on $C(\overline{\alpha})$. As $p(\overline{x})$ is finite with respect to $\overline{\delta}$, it follows by proposition 36 that $\prec_p$ is well-founded. We will use induction on $\prec_p$ to prove the stronger claim that: for each $\overline{\nu} \in C(\overline{\alpha})$:

$$f(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}) \ = \ \perp \tag{6.68}$$

iff:

$$\exists \overline{\mu} \in B(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}).\ f(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \perp \tag{6.69}$$

or:

$$\exists \overline{\mu} \in L(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}).\ t_i(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \perp \tag{6.70}$$

for some $1 \leq i \leq n$. The result follows immediately from the special case $\overline{\nu} = \overline{0}$.

Base Case: By proposition 35, the bottom elements are those of boundary of the lattice. If $\overline{\nu} \in B(\overline{\alpha})$ then by proposition 32 we know that $p(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta})$ is false. From (6.15) and (6.16) it follows that:

$$L(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}) \ = \ \emptyset$$

and:

$$B(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}) \;=\; \{\overline{0}\}$$

This means that the formula in (6.70) is false, while the formula in (6.69) is equivalent to:

$$f(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta} + \overline{0} \cdot \overline{\delta}) = \bot$$

which is equivalent to the formula in (6.68).

Induction Step: Suppose that the claim is true for each $\overline{\nu'} \prec_p \overline{\nu}$. We will show the claim for $\overline{\nu}$. If $\overline{\nu} \in B(\overline{\alpha})$ then the proof is reduced to that of the base case above. If $\overline{\nu} \in L(\overline{\alpha})$ then we know from proposition 31 that $p(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta})$ is true and consequently:

$$f(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}) \;=\; t(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}) \tag{6.71}$$

We will prove the equivalence in both directions.

($\longrightarrow$): Suppose that:

$$f(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}) \;=\; \bot \tag{6.72}$$

We will show that:

$$\exists \overline{\mu} \in B(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}).\ f(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \bot \tag{6.73}$$

or:

$$\exists \overline{\mu} \in L(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}).\ t_i(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \bot \tag{6.74}$$

for some $1 \leq i \leq n$.

From (6.72) and (6.71):

$$t(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}) \;=\; \bot \tag{6.75}$$

From (6.75) and the definition of semantics of recurrence equations (section 4.3) it follows that:

$$f(\overline{\alpha} + \overline{\nu}_j \cdot \overline{\delta}) \;=\; \bot \tag{6.76}$$

for some $1 \leq j \leq m$, where $\overline{\nu}_j = \overline{\nu} + \overline{u}_j$, or:

$$t_i(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}) \;=\; \bot \tag{6.77}$$

for some $1 \leq i \leq n$.

We consider the formula in (6.76). It is clear that $\overline{\nu}_j \prec_p \overline{\nu}$. From the induction hypothesis it follows that:

$$\exists \overline{\mu} \in B(\overline{\alpha} + \overline{\nu}_j \cdot \overline{\delta}).\ f(\overline{\alpha} + \overline{\nu}_j \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \bot \tag{6.78}$$

for some $1 \leq j \leq m$ or:

$$\exists \overline{\mu} \in L(\overline{\alpha} + \overline{\nu}_j \cdot \overline{\delta}).\ t_i(\overline{\alpha} + \overline{\nu}_j \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \bot \tag{6.79}$$

for some $1 \leq j \leq m$ and $1 \leq i \leq n$.

By applying corollary 34 on (6.78), where $P(\overline{\mu})$ is $f(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \perp$, we get (6.73).

By applying corollary 34 on (6.77) and (6.79), where $P(\overline{\mu})$ is $t_i(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \perp$, for some $1 \leq i \leq n$, we get (6.74).

(⟵): Suppose that:

$$\exists \overline{\mu} \in B(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}).\ f(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \perp \tag{6.80}$$

or:

$$\exists \overline{\mu} \in L(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}).\ t_i(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \perp \tag{6.81}$$

for some $1 \leq i \leq n$. We will show that:

$$f(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}) \ = \ \perp \tag{6.82}$$

Let $\overline{\nu}_j = \overline{\nu} + \overline{u}_j$, for $1 \leq j \leq m$.

By applying corollary 34 on (6.80), where $P(\overline{\mu})$ is $f(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \perp$, we get:

$$\exists \overline{\mu} \in B(\overline{\alpha} + \overline{\nu}_j \cdot \overline{\delta}).\ f(\overline{\alpha} + \overline{\nu}_j \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \perp \tag{6.83}$$

for some $1 \leq j \leq m$.

By applying corollary 34 on (6.81), where $P(\overline{\mu})$ is $t_i(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \perp$, for some $1 \leq i \leq n$, we get:

$$t_i(\overline{\alpha} + \overline{\nu} \cdot \overline{\delta}) \ = \ \perp \tag{6.84}$$

for some $1 \leq i \leq n$, or:

$$\exists \overline{\mu} \in L(\overline{\alpha} + \overline{\nu}_j \cdot \overline{\delta}).\ t_i(\overline{\alpha} + \overline{\nu}_j \cdot \overline{\delta} + \overline{\mu} \cdot \overline{\delta}) = \perp \tag{6.85}$$

for some $1 \leq j \leq m$ and $1 \leq i \leq n$.

We know that each $\overline{\nu}_j \prec_p \overline{\nu}$. From (6.83), (6.85), and the induction hypothesis it follows that:

$$f(\overline{\alpha} + \overline{\nu}_j \cdot \overline{\delta}) \ = \ \perp \tag{6.86}$$

for some $1 \leq j \leq m$.

From (6.86), (6.84), and the definition of semantics of recurrence equations (section 4.3):

$$t(\overline{x} + \overline{\nu} \cdot \overline{\delta}) \ = \ \perp \tag{6.87}$$

From (6.71) and (6.87) we get (6.82). □

Lemma 43 *Let:*

- *f be a function variable in an acyclic system of recurrence equations over a signature $\mathcal{SIG}$.*

- $\overline{\alpha}$ *be a tuple of integers.*
- $p(\overline{x})$ *and* $t(\overline{x})$ *be the guard and the result respectively defined by* f *and* $\overline{\alpha}$*, where* $t(\overline{x})$ *is of the form* $f(\overline{x} + \overline{\beta})$*.*
- $L(\overline{x})$*,* $B(\overline{x})$*, and* $C(\overline{x})$*, be the lattice, the boundary of the lattice, and the closure of the lattice with origin* $\overline{x}$*, basis* $\overline{\beta}$*, and constraint* $p(\overline{y})$*.*
- $p(\overline{x})$ *be finite with respect to* $\overline{\beta}$*.*

then:

$$f(\overline{\alpha}) = \bot$$

iff:

$$\exists \mu \in B(\overline{\alpha}).\ f(\overline{\alpha} + \mu \cdot \overline{\beta}) = \bot$$

Observe that the claim of the lemma is independent of the particular $\mathcal{SIG}$-algebra in which the equation system is interpreted.

Proof: Let $\prec_p$ be the predecessor relation on $C(\overline{\alpha})$. As $p(\overline{x})$ is finite with respect to $\overline{\beta}$, it follows by proposition 36 that $\prec_p$ is well-founded. We will use induction on $\prec_p$ to prove the stronger claim that: for each $\nu \in C(\overline{\alpha})$:

$$f(\overline{\alpha} + \nu \cdot \overline{\beta}) \;=\; \bot \tag{6.88}$$

iff:

$$\exists \mu \in B(\overline{\alpha} + \nu \cdot \overline{\beta}).\ f(\overline{\alpha} + \nu \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = \bot \tag{6.89}$$

The result follows immediately from the special case $\nu = 0$.

Base Case: By proposition 35, the bottom elements are those of boundary of the lattice. If $\nu \in B(\overline{\alpha})$, then by proposition 32 we know that $p(\overline{\alpha} + \nu \cdot \overline{\beta})$ is false. From (6.16) it follows that:

$$B(\overline{\alpha} + \nu \cdot \overline{\beta}) \;=\; \{0\}$$

This means that:

$$\exists \mu \in B(\overline{\alpha} + \nu \cdot \overline{\beta}).\ f(\overline{\alpha} + \nu \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = \bot \text{ iff } f(\overline{\alpha} + \nu \cdot \overline{\beta} + 0 \cdot \overline{\beta}) = \bot \text{ iff } f(\overline{\alpha} + \nu \cdot \overline{\beta}) = \bot$$

Induction Step: Suppose that the claim is true for each $\nu' \prec_p \nu$. We will show the claim for ν. If $\nu \in B(\overline{\alpha})$ then the proof is reduced to that of the base case above. If $\nu \in L(\overline{\alpha})$ then we know from proposition 31 that $p(\overline{\alpha} + \nu \cdot \overline{\beta})$ is true and consequently:

$$f(\overline{\alpha} + \nu \cdot \overline{\beta}) \;=\; f(\overline{\alpha} + \nu \cdot \overline{\beta} + \overline{\beta}) \;=\; f(\overline{\alpha} + (\nu + 1) \cdot \overline{\beta}) \tag{6.90}$$

We will prove the equivalence in both directions.

($\longrightarrow$): Suppose that:

$$f(\overline{\alpha} + \nu \cdot \overline{\beta}) \;=\; \bot \tag{6.91}$$

We will show that:

$$\exists \mu \in B(\overline{\alpha} + \nu \cdot \overline{\beta}).\ f(\overline{\alpha} + \nu \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = \bot \tag{6.92}$$

From (6.91) and (6.90):

$$f(\overline{\alpha} + (\nu + 1) \cdot \overline{\beta}) \;=\; \bot \tag{6.93}$$

It is clear that $\nu + 1 \prec_p \nu$. From the induction hypothesis it follows that:

$$\exists \mu \in B(\overline{\alpha} + (\nu + 1) \cdot \overline{\beta}).\ f(\overline{\alpha} + (\nu + 1) \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = \bot \tag{6.94}$$

By applying corollary 34 on (6.94), where $P(\mu)$ is $f(\overline{\alpha} + \nu \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = \bot$, we get (6.92).
(⟵): Suppose that:

$$\exists \mu \in B(\overline{\alpha} + \nu \cdot \overline{\beta}).\ f(\overline{\alpha} + \nu \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = \bot \tag{6.95}$$

We will show that:

$$f(\overline{\alpha} + \nu \cdot \overline{\beta}) \;=\; \bot \tag{6.96}$$

By applying corollary 34 on (6.95), where $P(\mu)$ is $f(\overline{\alpha} + \nu \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = \bot$, we get:

$$\exists \mu \in B(\overline{\alpha} + (\nu + 1) \cdot \overline{\beta}).\ f(\overline{\alpha} + (\nu + 1) \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = \bot \tag{6.97}$$

It is clear that $\nu + 1 \prec_p \nu$. From (6.97) and the induction hypothesis it follows that:

$$f(\overline{\alpha} + (\nu + 1) \cdot \overline{\beta}) \;=\; \bot \tag{6.98}$$

From (6.90) and (6.98) we get (6.96). □

Lemma 44 *Let:*

- *f be a function variable in an acyclic system of recurrence equations over a signature $\mathcal{SIG}$.*
- *$\overline{\alpha}$ be a tuple of integers.*
- *$p(\overline{x})$ and $t(\overline{x})$ be the guard and the result respectively defined by f and $\overline{\alpha}$, where $t(\overline{x})$ is a linear systolic term in which f occurs.*
- *$\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$ be the recursive vector of f in $t(\overline{x})$.*
- *$p(\overline{x})$ be not finite with respect to $\overline{\delta}$.*

then:

$$f(\overline{\alpha}) = \bot$$

Observe that the claim of the lemma is independent of the particular $\mathcal{SIG}$-algebra in which the equation system is interpreted.
Proof: Let $C(\overline{x})$ be the closure of the lattice with origin $\overline{x}$, basis $\overline{\delta}$, and constraint $p(\overline{y})$. By proposition 37 there is an infinite decreasing chain:

$$\overline{\mu}_1 \succ_p \overline{\mu}_2 \succ_p \overline{\mu}_3 \succ_p \cdots$$

where $\prec_p$ is the predecessor relation on $C(\overline{\alpha})$, $\overline{\mu}_1 = 0$, $\overline{\mu}_{j+1} = \overline{\mu}_j + \overline{u}_k$, for some $1 \leq k \leq m$, and $p(\overline{\alpha} + \overline{\mu}_j \cdot \overline{\delta})$ is true for $j \geq 0$. this means that:

$$
\begin{aligned}
& f(\overline{\alpha}) = \\
& \quad t(\ldots, f(\overline{\alpha} + \overline{\mu}_1), \ldots) = \\
& \quad t(\ldots, t(\ldots, f(\overline{\alpha} + \overline{\mu}_2), \ldots), \ldots) = \\
& \quad t(\ldots, t(\ldots, t(\ldots, f(\overline{\alpha} + \overline{\mu}_3), \ldots), \ldots), \ldots) = \\
& \quad t(\ldots, t(\ldots, t(\ldots, t(\ldots, f(\overline{\alpha} + \overline{\mu}_4), \ldots), \ldots), \ldots), \ldots) = \\
& \quad \ldots
\end{aligned}
$$

From the semantics of recurrence equations (section 4.3) we get:

$$f(\overline{\alpha}) \quad = \quad \bot$$

□

Lemma 45 *Let:*

- *f be a function variable in an acyclic system of recurrence equations over a signature $\mathcal{SIG}$.*
- *$\overline{\alpha}$ be a tuple of integers.*
- *$p(\overline{x})$ and $t(\overline{x})$ be the guard and the result respectively defined by f and $\overline{\alpha}$, where $t(\overline{x})$ is of the form $f(\overline{x} + \overline{\beta})$.*
- *$p(\overline{x})$ be infinite with respect to $\overline{\beta}$.*

then:

$$f(\overline{\alpha}) = \bot$$

Observe that the claim of the lemma is independent of the particular $\mathcal{SIG}$-algebra in which the equation system is interpreted.
Proof: Let $C(\overline{x})$ be the closure of the lattice with origin $\overline{x}$, basis $\overline{\beta}$, and constraint $p(\overline{y})$. By proposition 37 there is an infinite decreasing chain:

$$\mu_1 \succ_p \mu_2 \succ_p \mu_3 \succ_p \cdots$$

where $\prec_p$ is the predecessor relation on $C(\overline{\alpha})$, $\mu_1 = 0$, $\mu_{j+1} = \mu_j + 1$, and $p(\overline{\alpha} + \mu_j \cdot \overline{\beta})$ is true for $j \geq 0$. This means that:

$$f(\overline{\alpha}) = f(\overline{\alpha} + \overline{\beta}) = f(\overline{\alpha} + 2\overline{\beta}) = f(\overline{\alpha} + 3\overline{\beta}) = f(\overline{\alpha} + 4\overline{\beta}) = \cdots$$

From the semantics of recurrence equations (section 4.3) we get:

$$f(\overline{\alpha}) \quad = \quad \bot$$

□

Lemma 46 *Let f be a function variable in an acyclic system of recurrence equations over a signature $\mathcal{SIG}$. Then there is a Presburger formula $P(\overline{x})$, such that for each $\overline{\alpha}$:*

$$P(\overline{\alpha}) \quad \textit{iff} \quad (f(\overline{\alpha}) = \bot)$$

Observe that the equivalence is not dependent on the particular $\mathcal{SIG}$-algebra in which the equation system is interpreted.

Proof: Let E be an acyclic system of recurrence equations over a signature $\mathcal{SIG}$. Let the definition of f in E be of the form:

$$f(\overline{x}) \;=\; \mathcal{F}(\overline{x})$$

where $\mathcal{F}(\overline{x})$ is of the form:

$$\begin{array}{l}\textbf{case}\\ \quad p_1(\overline{x}) \;\Longrightarrow\; t_1(\overline{x})\\ \qquad\vdots\\ \quad p_k(\overline{x}) \;\Longrightarrow\; t_k(\overline{x})\\ \textbf{endcase}\end{array}$$

For each $\overline{\alpha}$, if $p_i(\overline{\alpha})$ is false for $1 \leq i \leq k$, then $f(\overline{\alpha}) = \bot$. If there is a $p_i(\overline{x})$ such that $p_i(\overline{\alpha})$ is true then $f(\overline{\alpha}) = t_i(\overline{\alpha})$. We will use induction on the complexity $\mathcal{C}(E, f)$ of f in E to give a Presburger formula $P_i(\overline{x})$ such that for each $\overline{\alpha}$, if $p_i(\overline{\alpha})$ is true then $P_i(\overline{\alpha})$ iff $f(\overline{\alpha}) = \bot$.

Base Case: If $\mathcal{C}(E, f)$ is of the form $< e, 0 >$, i.e. $k = 0$, then $f(\overline{x}) = \bot$, for all $\overline{x}$, and:

$$P_i(\overline{x}) \;=\; \text{true}$$

which is a Presburger formula.

Induction Step: If $k = 0$ then the proof is reduced to that of the base case above. Suppose that $k > 0$. A number of cases are possible. In each case we will write $P_i(\overline{x})$ as a Presburger formula.

1. If $t_i(\overline{x})$ is a stream expression of the form $a(\overline{q}(\overline{x}))$. As $t_i(\overline{x})$ is well-defined under $p_i(\overline{x})$, it follows from the definition of semantics in section 4.3 that:

$$P_i(\overline{x}) \;=\; \text{false}$$

which is a Presburger formula.

2. If $t_i(\overline{x})$ is a function variable expression of the form $f'(\overline{x} + \overline{\beta})$, where $f' \neq f$. Let E' be the system of recurrence equations we get from E by deleting the equation:

$$f(\overline{x}) \;=\; \mathcal{F}(\overline{x})$$

We note that f' is a function variable in E'. From the acyclicity of $\mathcal{F}(\overline{x})$ it follows that:

$$f(\overline{x}) \;=\; f'(\overline{x} + \overline{\beta})$$

where f' is considered as a function variable in E'. Also we know that:

$$\mathcal{C}(E', f') \;<\; \mathcal{C}(E, f)$$

From the induction hypothesis it follows that there is a Presburger formula $P_i'(\overline{x})$ such that:

$$P_i'(\overline{x}) \quad \text{iff} \quad (f'(\overline{x}) = \bot)$$

Now $P_i(\overline{x})$ can be defined to be:

$$P_i(\overline{x}) \quad = \quad P_i'(\overline{x} + \overline{\beta})$$

which is a Presburger formula.

3. If $t_i(\overline{x})$ is a function variable expression of the form $f(\overline{x} + \overline{\beta})$. By proposition 41 we can check whether $p_i(\overline{x})$ is finite with respect to $\overline{\beta}$ or not. If $p_i(\overline{x})$ is not finite with respect to $\overline{\beta}$ then it follows by lemma 45 that:

$$P_i(\overline{x}) \quad = \quad \text{true}$$

which is a Presburger formula. If $p_i(\overline{x})$ is finite with respect to $\overline{\beta}$ then let $B(\overline{x})$ be the boundary of the lattice with origin $\overline{x}$, basis $\overline{\beta}$ and constraint $p_i(\overline{y})$. By lemma 43 it follows that:

$$(f(\overline{x}) = \bot) \quad \text{iff} \quad \left(\exists \mu \in B(\overline{x}).\ f(\overline{x} + \mu \cdot \overline{\beta}) = \bot\right) \tag{6.99}$$

let $bf(\overline{x})$ be the boundary formula corresponding to $B(\overline{x})$, then:

$$bf(\overline{x}, \mu) \quad = \quad (\mu \in B(\overline{x})) \tag{6.100}$$

Let E^* be the equation system we get from E by replacing the equation:

$$f(\overline{x}) \quad = \quad \mathcal{F}(\overline{x})$$

in E by the equation:

$$f^*(\overline{x}) \quad = \quad \mathcal{F}^*(\overline{x})$$

where $\mathcal{F}^*(\overline{x})$ is the result of deleting the case $p_i(\overline{x}) \Rightarrow t_i(\overline{x})$ in $\mathcal{F}(\overline{x})$. By proposition 32 we know that for each $\mu \in B(\overline{x})$, $p(\overline{x} + \mu \cdot \overline{\beta})$ is false. By the acyclicity of $\mathcal{F}(\overline{x})$ it follows that:

$$\forall \mu \in B(\overline{x}).\ (f(\overline{x} + \mu \cdot \overline{\beta}) = f^*(\overline{x} + \mu \cdot \overline{\beta})) \tag{6.101}$$

We know that $\mathcal{C}(E^*, f^*) < \mathcal{C}(E, f)$. By the induction hypothesis it follows that there is a Presburger formula $P_i^*(\overline{x})$ such that:

$$P_i^*(\overline{x}) \quad \text{iff} \quad (f^*(\overline{x}) = \bot) \tag{6.102}$$

From (6.99), (6.100), (6.101), and (6.102) it follows that $P_i(\overline{x})$ can be defined as:

$$P_i(\overline{x}) \quad = \quad \exists \mu.\ bf(\overline{x}, \mu) \ \wedge \ P_i^*(\overline{x} + \mu \cdot \overline{\beta})$$

which is a Presburger formula.

4. If $t_i(\overline{x})$ is a linear systolic term in which f does not occur. Let $t_i(\overline{x})$ be of the form $g(t'_1(\overline{x}), \ldots, t'_n(\overline{x}))$. From the definition of semantics in section 4.3 it follows that:

$$(f(\overline{x}) = \bot) \quad \text{iff} \quad \bigvee_{l=1}^{n} (t'_l(\overline{x}) = \bot)$$

We will show that for each $1 \leq l \leq n$ there is a Presburger formula $P_i^{(l)}(\overline{x})$ such that:

$$P_i^{(l)}(\overline{x}) \quad \text{iff} \quad (t'_l(\overline{x}) = \bot)$$

There are a number of cases. In each case we will write $P_i^{(l)}(\overline{x})$ as a Presburger formula.

- If $t_l(\overline{x})$ is a stream expression. This case is similar to case (1) above.
- If $t_l(\overline{x})$ is a function variable expression, whose function variable is not f. This case is similar to case (2) above.

It follows that $P_i(\overline{x})$ can be defined as:

$$P_i(\overline{x}) \quad = \quad \bigvee_{l=1}^{n} P_i^{(l)}(\overline{x})$$

which is a Presburger formula.

5. If $t_i(\overline{x})$ is a linear systolic term in which f occurs. Let $\overline{\delta} =< \overline{\delta}_1, \ldots, \overline{\delta}_m >$ be the recursive vector of f in $t_i(\overline{x})$. Let $t'_1(\overline{x}), \ldots, t'_n(\overline{x})$ be the non-recursive terms of f in $t_i(\overline{x})$.

By proposition 41 we can check whether $p_i(\overline{x})$ is finite with respect to $\overline{\delta}$ or not. If $p_i(\overline{x})$ is not finite with respect to $\overline{\delta}$ then it follows by lemma 44 that:

$$P_i(\overline{x}) \quad = \quad \text{true}$$

which is a Presburger formula. If $p_i(\overline{x})$ is finite with respect to $\overline{\alpha}$ then let $L(\overline{x})$ and $B(\overline{x})$ be the lattice and the boundary of the lattice with origin $\overline{x}$, basis $\overline{\delta}$ and constraint $p_i(\overline{y})$. From lemma 42 it follows that:

$$\begin{array}{c} f(\overline{x}) = \bot \\ \text{iff} \\ \exists \overline{\mu} \in B(\overline{x}).\ f(\overline{x} + \overline{\mu} \cdot \overline{\delta}) = \bot \\ \vee \\ \bigvee_{l=1}^{n} \left(\exists \overline{\mu} \in L(\overline{x}).\ t'_l(\overline{x} + \overline{\mu} \cdot \overline{\delta}) = \bot \right) \end{array} \tag{6.103}$$

Let $lf(\overline{x}, \overline{\mu})$ and $bf(\overline{x}, \overline{\mu})$ be the lattice and the boundary formulas respectively of $L(\overline{x})$. This means that:

$$lf(\overline{x}, \overline{\mu}) \quad = \quad (\overline{\mu} \in L(\overline{x})) \tag{6.104}$$

and:

$$bf(\overline{x}, \overline{\mu}) \quad = \quad (\overline{\mu} \in B(\overline{x})) \tag{6.105}$$

In a similar manner to case (3) above we can show that there is a Presburger formula $P_i^*(\overline{x})$ such that:

$$\forall \overline{\mu} \in B(\overline{x}).\ P_i^*(\overline{x} + \overline{\mu} \cdot \overline{\delta}) \quad \text{iff} \quad (f(\overline{x} + \overline{\mu} \cdot \overline{\delta}) = \bot) \tag{6.106}$$

In a similar manner to case (4) above we can show that for each $1 \leq l \leq n$ there is a Presburger formula $P_i^{(l)}(\overline{x})$ such that:

$$P_i^{(l)}(\overline{x}) \quad \text{iff} \quad (t_l'(\overline{x}) = \bot) \tag{6.107}$$

From (6.103), (6.104), (6.105), (6.106), and (6.107), it follows that $P_i(\overline{x})$ can be defined as:

$$P_i(\overline{x}) \quad = \quad \left(\begin{array}{c} \left(\exists \overline{\mu}.\ bf(\overline{x}, \overline{\mu}) \ \wedge \ P_i^*(\overline{x} + \overline{\mu} \cdot \overline{\delta})\right) \\ \vee \\ \bigvee_{l=1}^{n} \left(\exists \overline{\mu}.\ lf(\overline{x}, \overline{\mu}) \ \wedge \ P_i^{(l)}(\overline{x} + \overline{\mu} \cdot \overline{\delta})\right) \end{array} \right)$$

which is a Presburger formula.

Now the formula $P(\overline{x})$ in the claim of the lemma can be defined as:

$$P(\overline{x}) \quad = \quad \left(\bigwedge_{i=1}^{k} \neg p_i(\overline{x})\right) \ \vee \ \left(\bigvee_{i=1}^{k} (p_i(\overline{x}) \ \wedge \ P_i(\overline{x}))\right)$$

which is a Presburger formula.

Observe that the construction of $P(\overline{x})$ in the lemma was independent of the particular $\mathcal{SIG}$-algebra in which E is interpreted. □

6.6.5 Equality of Defined Systems of Recurrence Equations

In this section we show that, given two systems E_1 and E_2 of acyclic recurrence equations over a signature $\mathcal{SIG}$, and two function variables f_1 and f_2 in E_1 and E_2, then there is a Presburger formula $P(\overline{x}_1, \overline{x}_2)$ such that for each value $\overline{\alpha}_1$ and $\overline{\alpha}_2$ of $\overline{x}_1$ and $\overline{x}_2$, if $f_1(\overline{\alpha}_1) \neq \bot$ and $f_2(\overline{\alpha}_2) \neq \bot$, then $P(\overline{\alpha}_1, \overline{\alpha}_2)$ is true iff $f_1(\overline{\alpha}_1) = f_2(\overline{\alpha}_2)$ for every $\mathcal{SIG}$-algebra.

By a *common lattice* $L^c(\overline{x}_1, \overline{x}_2)$ with origins $\overline{x}_1$ and $\overline{x}_2$, bases $\overline{\delta}_1 =< \overline{\delta}_{11}, \ldots, \overline{\delta}_{1m} >$ and $\overline{\delta}_2 =< \overline{\delta}_{21}, \ldots, \overline{\delta}_{2m} >$, and constraints $p_1(\overline{x})$ and $p_2(\overline{x})$, we mean a lattice with origin $< \overline{x}_1, \overline{x}_2 >$, basis $<< \overline{\delta}_{11}, \overline{\delta}_{21} >, \ldots, < \overline{\delta}_{1m}, \overline{\delta}_{2m} >>$, and constraint $p_1(\overline{x}_1) \wedge p_2(\overline{x}_2)$. Intuitively if $\overline{\mu} \in L^c(\overline{x}_1, \overline{x}_2)$ then $\overline{x}_1 + \overline{\mu} \cdot \overline{\delta}_1$ and $\overline{x}_2 + \overline{\mu} \cdot \overline{\delta}_2$ are two integer points inside the polytopes defined by $p_1(\overline{x}_1)$ and $p_2(\overline{x}_2)$ which can be reached from $\overline{x}_1$ and $\overline{x}_2$ by a finite number of steps inside the two polytopes, moving in the direction of $\overline{\delta}_{1i}$ and $\overline{\delta}_{2i}$ respectively, for some $1 \leq i \leq m$, at each step.

As an example, consider a common lattice $L^c(x_1, x_2, y_1, y_2)$, with origins $< x_1, x_2 >$ and $< y_1, y_2 >$, bases $<< -1, -1 >, < 1, -1 >, < 0, -2 >>$ and $<< 0, -1 >, < 1, 0 >, < 1, -1 >>$, and constraints $(x_1 \leq x_2 + 2) \wedge (x_1 + x_2 \geq 10)$ and $(y_1 \leq 13) \wedge (y_2 \geq -8)$, illustrated in figure 6.2. The points marked by filled circles in the upper half of the plane are of the form $< 9 - \mu_1 + \mu_2, 15 - \mu_1 - \mu_2 - 2\mu_3 >$, where $< \mu_1, \mu_2, \mu_3 > \in L^c(9, 15, 4, -3)$. The points marked by filled circles in the lower half of the plane are of the form $< 4 + \mu_2 + \mu_3, -3 - \mu_1 - \mu_3 >$, where $< \mu_1, \mu_2, \mu_3 > \in L^c(9, 15, 4, -3)$.

The points marked by hollow circles in the upper half of the plane are of the form $< 9-\mu_1+\mu_2, 15-\mu_1-\mu_2-2\mu_3 >$, where $< \mu_1, \mu_2, \mu_3 > \in B^c(9, 15, 4, -3)$. The points marked by hollow circles in the lower half of the plane are of the form $< 4+\mu_2+\mu_3, -3-\mu_1-\mu_3 >$, where $< \mu_1, \mu_2, \mu_3 > \in B^c(9, 15, 4, -3)$.

Consider two function variables f_1 and f_2 and two linear systolic terms $g(t_{11}(\overline{x}), \ldots, t_{1n}(\overline{x}))$ and $g(t_{21}(\overline{x}), \ldots, t_{2n}(\overline{x}))$. Let $t_{1i_1}(\overline{x}), \ldots, t_{1i_m}(\overline{x})$ and $t_{2i_1}(\overline{x}), \ldots, t_{2i_m}(\overline{x})$ be such that $t_{1i_j}(\overline{x})$ and $t_{2i_j}(\overline{x})$ are of the forms $f_1(\overline{x} + \overline{\delta}_{1j})$ and $f_2(\overline{x} + \overline{\delta}_{2j})$ respectively, and such that there are no $t_1(\overline{x})$ and $t_2(\overline{x})$ where $t_1(\overline{x}) \in \{t_{11}(\overline{x}), \ldots, t_{1n}(\overline{x})\} \setminus \{t_{1i_1}(\overline{x}), \ldots, t_{1i_m}(\overline{x})\}$, $t_2(\overline{x}) \in \{t_{21}(\overline{x}), \ldots, t_{2n}(\overline{x})\} \setminus \{t_{2i_1}(\overline{x}), \ldots, t_{2i_m}(\overline{x})\}$, $t_1(\overline{x})$ is a function variable expression which has f_1 as a function variable, and $t_2(\overline{x})$ is a function variable expression which has f_2 as a function variable. Then $< \overline{\delta}_{11}, \ldots, \overline{\delta}_{1m} >$ and $< \overline{\delta}_{21}, \ldots, \overline{\delta}_{2m} >$ are called the *common recursive vectors* of f_1 and f_2 in $g(t_{11}(\overline{x}), \ldots, t_{1n}(\overline{x}))$ and $g(t_{21}(\overline{x}), \ldots, t_{2n}(\overline{x}))$, and the elements of $\{t_{11}(\overline{x}), \ldots, t_{1n}(\overline{x})\} \setminus \{t_{1i_1}(\overline{x}), \ldots, t_{1i_m}(\overline{x})\}$ and $\{t_{21}(\overline{x}), \ldots, t_{2n}(\overline{x})\} \setminus \{t_{2i_1}(\overline{x}), \ldots, t_{2i_m}(\overline{x})\}$, are called the *common non-recursive terms* of f_1 and f_2 in $g(t_{11}(\overline{x}), \ldots, t_{1n}(\overline{x}))$ and $g(t_{21}(\overline{x}), \ldots, t_{2n}(\overline{x}))$. For example, let:

$$t_1(x) \;=\; g(f_1(x+4), a(3x), h(x), f_1(x-7), f_1(x+3))$$

and:

$$t_2(x) \;=\; g(f_2(x+7), h(x-4), b(2x), f_2(x), a(2x))$$

then the common recursive vectors of f_1 and f_2 in $t_1(x)$ and $t_2(x)$ are $< 4, -7 >$ and $< 7, 0 >$, and the common non-recursive terms are $a(3x)$, $h(x)$, $f_1(x+3)$, and $h(x-4)$, $b(2x)$, $a(2x)$.

Lemma 47 *Let:*

- *f_1 and f_2 be function variables in two acyclic systems of recurrence equations over a signature $\mathcal{SIG}$.*
- *$\overline{\alpha}_1$ and $\overline{\alpha}_2$ be tuples of integers.*
- *$p_i(\overline{x})$ and $t_i(\overline{x})$ be the guard and result respectively defined by f_i and $\overline{\alpha}_i$, where $t_i(\overline{x})$ is a linear systolic term in which f_i occurs, for $i = 1, 2$, and such that $t_1(\overline{x})$ and $t_2(\overline{x})$ have the same operation symbol.*
- *$\overline{\delta}_1 =< \overline{\delta}_{11}, \ldots, \overline{\delta}_{1m} >$ and $\overline{\delta}_2 =< \overline{\delta}_{21}, \ldots, \overline{\delta}_{2m} >$ be the common recursive vectors of f_1 and f_2 in $t_1(\overline{x})$ and $t_2(\overline{x})$.*
- *$t_{11}(\overline{x}), \ldots, t_{1n}(\overline{x})$ and $t_{21}(\overline{x}), \ldots, t_{2n}(\overline{x})$ be the common non-recursive terms of f_1 and f_2 in $t_1(\overline{x})$ and $t_2(\overline{x})$.*
- *$L^c(\overline{x}_1, \overline{x}_2)$, $B^c(\overline{x}_1, \overline{x}_2)$, and $C^c(\overline{x}_1, \overline{x}_2)$, be the common lattice, the boundary of the common lattice, and the closure of the common lattice with origins $\overline{x}_1, \overline{x}_2$, bases $\overline{\delta}_1, \overline{\delta}_2$, and constraints $p_1(\overline{x}), p_2(\overline{x})$.*
- *$f_1(\overline{\alpha}_1) \neq \perp$, and $f_2(\overline{\alpha}_2) \neq \perp$.*
- *$\mathcal{SPEC}$ be the algebraic specification of the class of all $\mathcal{SIG}$-algebras.*

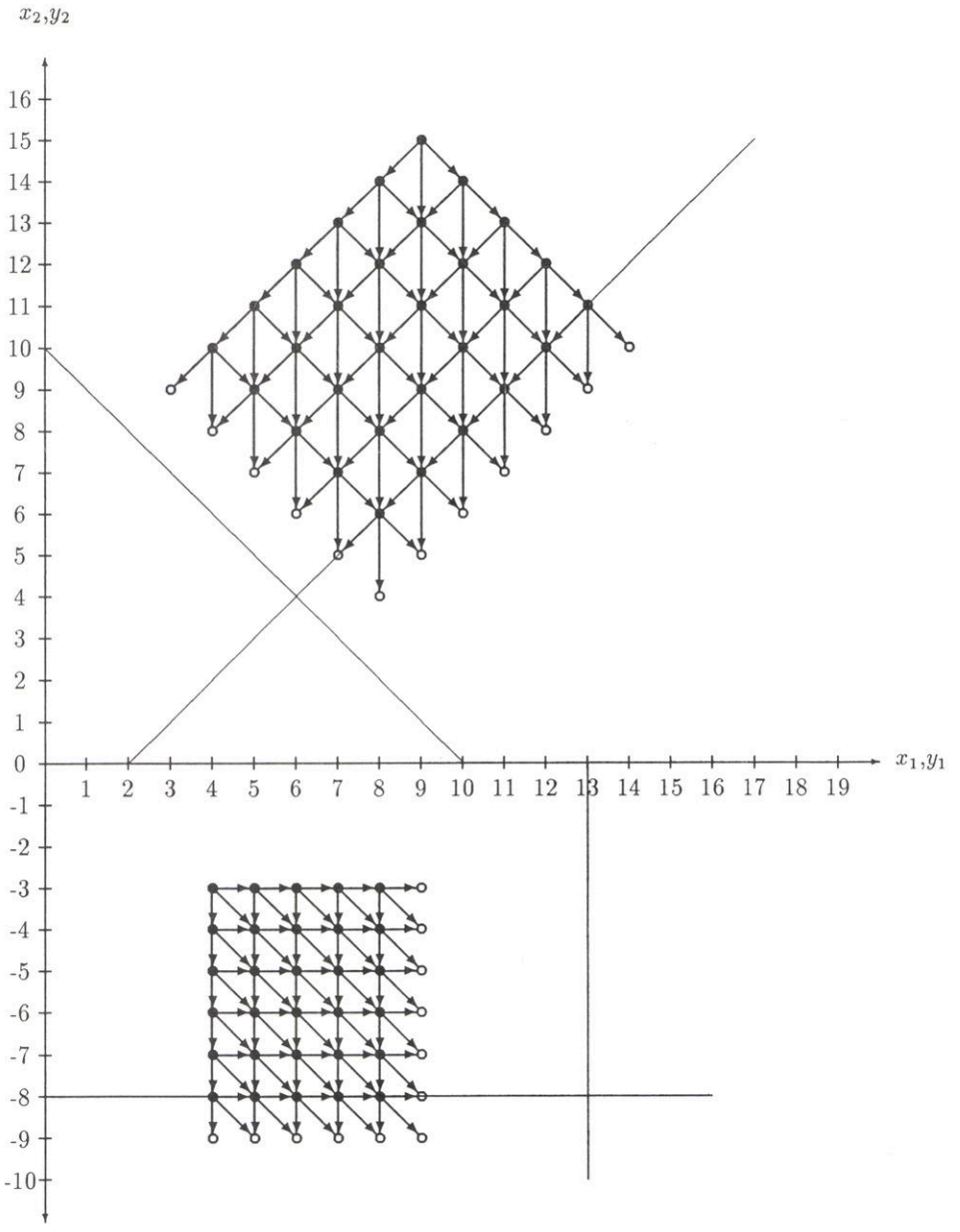

Figure 6.2: The common lattice and the boundary of the common lattice with origins $< 9, 15 >$ and $< 4, -3 >$, bases $<< -1, -1 >, < 1, -1 >, < 0, -2 >>$ and $<< 0, -1 >, < 1, 0 >, < 1, -1 >>$ and constraints $(x_1 \leq x_2{+}2) \wedge (x_1{+}x_2 \geq 10)$ and $(y_1 \leq 13) \wedge (y_2 \geq -8)$.

then

$$\mathcal{SPEC} \models f_1(\overline{\alpha}_1) = f_2(\overline{\alpha}_2)$$

iff:

$$\forall \overline{\mu} \in B^c(\overline{\alpha}_1, \overline{\alpha}_2).\ \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\mu} \cdot \overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\mu} \cdot \overline{\delta}_2)$$

and:

$$\forall \overline{\mu} \in L^c(\overline{\alpha}_1, \overline{\alpha}_2).\ \mathcal{SPEC} \models t_{1i}(\overline{\alpha}_1 + \overline{\mu} \cdot \overline{\delta}_1) = t_{2i}(\overline{\alpha}_2 + \overline{\mu} \cdot \overline{\delta}_2)$$

for $1 \leq i \leq n$.

Proof: Let $\prec_p$ be the predecessor relation on $C^c(\overline{\alpha}_1, \overline{\alpha}_2)$. As $f_i(\overline{\alpha}_1) \neq \bot$, it follows by lemma 44 that $p_i(\overline{x})$ is finite with respect to $\overline{\delta}_i$ for $i = 1, 2$. It follows that $p_1(\overline{x}) \wedge p_2(\overline{x})$ is finite with respect to $< \overline{\delta}_1, \overline{\delta}_2 >$. By proposition 36 it follows that $\prec_p$ is well-founded. We will use induction on $\prec_p$ to prove the stronger claim that: for each $\overline{\nu} \in C^c(\overline{\alpha}_1, \overline{\alpha}_2)$:

$$\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2) \tag{6.108}$$

iff:

$$\begin{aligned} &\forall \overline{\mu} \in B^c(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1, \overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2). \\ &\quad \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1 + \overline{\mu} \cdot \overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2 + \overline{\mu} \cdot \overline{\delta}_2) \end{aligned} \tag{6.109}$$

and:

$$\begin{aligned} &\forall \overline{\mu} \in L^c(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1, \overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2). \\ &\quad \mathcal{SPEC} \models t_{1i}(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1 + \overline{\mu} \cdot \overline{\delta}_1) = t_{2i}(\overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2 + \overline{\mu} \cdot \overline{\delta}_2) \end{aligned} \tag{6.110}$$

for $1 \leq i \leq n$. The result follows immediately from the special case $\overline{\nu} = \overline{0}$.
Base Case: By proposition 35, the bottom elements are those of the boundary of the common lattice. If $\overline{\nu} \in B^c(\overline{\alpha}_1, \overline{\alpha}_2)$, then by proposition 32 we know that $p_1(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1) \wedge p_2(\overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2)$ is false. From (6.15) and (6.16) it follows that:

$$L^c(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1, \overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2) = \emptyset$$

and:

$$B^c(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1, \overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2) = \{\overline{0}\}$$

This means that the formula in (6.110) is true, while the formula in (6.109) is equivalent to:

$$\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1 + \overline{0} \cdot \overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2 + \overline{0} \cdot \overline{\delta}_2)$$

which is equivalent to the formula in (6.108).
Induction Step: Suppose that the claim is true for each $\overline{\nu'} \prec_p \overline{\nu}$. We will show the claim for $\overline{\nu}$. If $\overline{\nu} \in B^c(\overline{\alpha}_1, \overline{\alpha}_2)$ then the proof is reduced to that of the base case above.

If $\overline{\nu} \in L^c(\overline{\alpha}_1, \overline{\alpha}_2)$ then we know from proposition 31 that $p_i(\overline{\alpha}_i + \overline{\nu} \cdot \overline{\delta}_i)$ is true and consequently:

$$\mathcal{SPEC} \models f_i(\overline{\alpha}_i + \overline{\nu} \cdot \overline{\delta}_i) = t_i(\overline{\alpha}_i + \overline{\nu} \cdot \overline{\delta}_i) \tag{6.111}$$

for $i = 1, 2$. We will prove the equivalence in both directions.
($\longrightarrow$): Suppose that:

$$\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2) \tag{6.112}$$

We will show that:

$$\begin{aligned} &\forall \overline{\mu} \in B^c(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1, \overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2). \\ &\quad \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1 + \overline{\mu} \cdot \overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2 + \overline{\mu} \cdot \overline{\delta}_2) \end{aligned} \tag{6.113}$$

and:

$$\begin{aligned} &\forall \overline{\mu} \in L^c(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1, \overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2). \\ &\quad \mathcal{SPEC} \models t_{1i}(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1 + \overline{\mu} \cdot \overline{\delta}_1) = t_{2i}(\overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2 + \overline{\mu} \cdot \overline{\delta}_2) \end{aligned} \tag{6.114}$$

for $1 \leq i \leq n$.

From (6.112) and (6.111):

$$\mathcal{SPEC} \models t_1(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1) = t_2(\overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2) \tag{6.115}$$

From (6.115) and the definition of semantics of recurrence equations (section 4.3) it can be shown that:

$$\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu}_j \cdot \overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu}_j \cdot \overline{\delta}_2) \tag{6.116}$$

for $1 \leq j \leq m$, where $\overline{\nu}_j = \overline{\nu} + \overline{u}_j$, and:

$$\mathcal{SPEC} \models t_{1i}(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1) = t_{2i}(\overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2) \tag{6.117}$$

for $1 \leq i \leq n$.

We consider the formula in (6.116). It is clear that $\overline{\nu}_j \prec_p \overline{\nu}$. From the induction hypothesis it follows that:

$$\begin{aligned} &\forall \overline{\mu} \in B^c(\overline{\alpha}_1 + \overline{\nu}_j \cdot \overline{\delta}_1, \overline{\alpha}_2 + \overline{\nu}_j \cdot \overline{\delta}_2). \\ &\quad \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu}_j \cdot \overline{\delta}_1 + \overline{\mu} \cdot \overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu}_j \cdot \overline{\delta}_2 + \overline{\mu} \cdot \overline{\delta}_2) \end{aligned} \tag{6.118}$$

for $1 \leq j \leq m$ and:

$$\begin{aligned} &\forall \overline{\mu} \in L^c(\overline{\alpha}_1 + \overline{\nu}_j \cdot \overline{\delta}_1, \overline{\alpha}_2 + \overline{\nu}_j \cdot \overline{\delta}_2). \\ &\quad \mathcal{SPEC} \models t_{1i}(\overline{\alpha}_1 + \overline{\nu}_j \cdot \overline{\delta}_1 + \overline{\mu} \cdot \overline{\delta}_1) = t_{2i}(\overline{\alpha}_2 + \overline{\nu}_j \cdot \overline{\delta}_2 + \overline{\mu} \cdot \overline{\delta}_2) \end{aligned} \tag{6.119}$$

for $1 \leq j \leq m$, and $1 \leq i \leq n$.

By applying proposition 34 on (6.118), where:

$$P(\overline{\mu}) = \left(\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu} \cdot \overline{\delta}_1 + \overline{\mu} \cdot \overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu} \cdot \overline{\delta}_2 + \overline{\mu} \cdot \overline{\delta}_2)\right)$$

we get (6.113).

By applying proposition 34 on (6.117), and (6.119), where:

$$P(\overline{\mu}) \;=\; \left(\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu}\cdot\overline{\delta}_1 + \overline{\mu}\cdot\overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu}\cdot\overline{\delta}_2 + \overline{\mu}\cdot\overline{\delta}_2)\right)$$

we get (6.114).

(⟵): Suppose that:

$$\begin{aligned}&\forall\overline{\mu} \in B^c(\overline{\alpha}_1 + \overline{\nu}\cdot\overline{\delta}_1, \overline{\alpha}_2 + \overline{\nu}\cdot\overline{\delta}_2). \\ &\quad \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu}\cdot\overline{\delta}_1 + \overline{\mu}\cdot\overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu}\cdot\overline{\delta}_2 + \overline{\mu}\cdot\overline{\delta}_2)\end{aligned} \tag{6.120}$$

and:

$$\begin{aligned}&\forall\overline{\mu} \in L^c(\overline{\alpha}_1 + \overline{\nu}\cdot\overline{\delta}_1, \overline{\alpha}_2 + \overline{\nu}\cdot\overline{\delta}_2). \\ &\quad \mathcal{SPEC} \models t_{1i}(\overline{\alpha}_1 + \overline{\nu}\cdot\overline{\delta}_1 + \overline{\mu}\cdot\overline{\delta}_1) = t_{2i}(\overline{\alpha}_2 + \overline{\nu}\cdot\overline{\delta}_2 + \overline{\mu}\cdot\overline{\delta}_2)\end{aligned} \tag{6.121}$$

for $1 \leq i \leq n$. We will show that:

$$\mathcal{SPEC} \;\models\; f_1(\overline{\alpha}_1 + \overline{\nu}\cdot\overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu}\cdot\overline{\delta}_2) \tag{6.122}$$

Let $\overline{\nu}_j = \overline{\nu} + \overline{u}_j$, for $1 \leq j \leq m$.

By applying proposition 34 on (6.120), where:

$$P(\overline{\mu}) \;=\; \left(\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu}\cdot\overline{\delta}_1 + \overline{\mu}\cdot\overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu}\cdot\overline{\delta}_2 + \overline{\mu}\cdot\overline{\delta}_2)\right)$$

we get:

$$\begin{aligned}&\forall\overline{\mu} \in B^c(\overline{\alpha}_1 + \overline{\nu}_j\cdot\overline{\delta}_1, \overline{\alpha}_2 + \overline{\nu}_j\cdot\overline{\delta}_2). \\ &\quad \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu}_j\cdot\overline{\delta}_1 + \overline{\mu}\cdot\overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu}_j\cdot\overline{\delta}_2 + \overline{\mu}\cdot\overline{\delta}_2)\end{aligned} \tag{6.123}$$

for $1 \leq j \leq m$.

By applying proposition 34 on (6.121), where:

$$P(\overline{\mu}) \;=\; \left(\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \overline{\nu}\cdot\overline{\delta}_1 + \overline{\mu}\cdot\overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu}\cdot\overline{\delta}_2 + \overline{\mu}\cdot\overline{\delta}_2)\right)$$

we get:

$$t_{1i}(\overline{\alpha}_1 + \overline{\nu}\cdot\overline{\delta}_1) \;=\; t_{2i}(\overline{\alpha}_2 + \overline{\nu}\cdot\overline{\delta}_2) \tag{6.124}$$

for $1 \leq i \leq n$, and:

$$\begin{aligned}&\forall\overline{\mu} \in L^c(\overline{\alpha}_1 + \overline{\nu}_j\cdot\overline{\delta}_1, \overline{\alpha}_2 + \overline{\nu}_j\cdot\overline{\delta}_2). \\ &\quad \mathcal{SPEC} \models t_{1i}(\overline{\alpha}_1 + \overline{\nu}_j\cdot\overline{\delta}_1 + \overline{\mu}\cdot\overline{\delta}_1) = t_{2i}(\overline{\alpha}_2 + \overline{\nu}_j\cdot\overline{\delta}_2 + \overline{\mu}\cdot\overline{\delta}_2)\end{aligned} \tag{6.125}$$

for $1 \leq j \leq m$ and $1 \leq i \leq n$.

We know that each $\overline{\nu}_j \prec_p \overline{\nu}$. From (6.123), (6.125), and the induction hypothesis it follows that:

$$\mathcal{SPEC} \;\models\; f_1(\overline{\alpha}_1 + \overline{\nu}_j\cdot\overline{\delta}_1) = f_2(\overline{\alpha}_2 + \overline{\nu}_j\cdot\overline{\delta}_2) \tag{6.126}$$

for $1 \leq j \leq m$.

From the definition of semantics, (6.126), and (6.124) we get:

$$\mathcal{SPEC} \;\models\; t_1(\overline{\alpha}_1 + \overline{\nu}\cdot\overline{\delta}_1) = t_2(\overline{\alpha}_2 + \overline{\nu}\cdot\overline{\delta}_2) \tag{6.127}$$

From (6.111) and (6.127) we get (6.122). □

Lemma 48 *Let:*

- f_1 *and* f_2 *be function variables in two acyclic systems of recurrence equations over a signature* $\mathcal{SIG}$.
- $\overline{\alpha}_1$ *and* $\overline{\alpha}_2$ *be tuples of integers.*
- $p(\overline{x})$ *and* $t(\overline{x})$ *be the guard and the result respectively defined by* f_1 *and* $\overline{\alpha}_1$, *where* $t(\overline{x})$ *is of the form* $f_1(\overline{x} + \overline{\beta})$.
- $L(\overline{x})$, $B(\overline{x})$, *and* $C(\overline{x})$, *be the lattice, the boundary of the lattice, and the closure of the lattice with origin* $\overline{x}$, *basis* $\overline{\beta}$, *and constraint* $p(\overline{y})$.
- $p(\overline{x})$ *be infinite with respect to* $\overline{\beta}$.
- $\mathcal{SPEC}$ *be the algebraic specification of the class of all* $\mathcal{SIG}$*-algebras.*

then:

$$\mathcal{SPEC} \models f_1(\overline{\alpha}_1) = f_2(\overline{\alpha}_2)$$

iff:

$$\forall \mu \in B(\overline{\alpha}_1).\ \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \mu \cdot \overline{\delta}_1) = f_2(\overline{\alpha}_2)$$

Proof: Let $\prec_p$ be the predecessor relation on $C(\overline{\alpha})$. As $p(\overline{x})$ is finite with respect to $\overline{\beta}$, it follows by proposition 36 that $\prec_p$ is well-founded. We will use induction on $\prec_p$ to prove the stronger claim that: for each $\nu \in C(\overline{\alpha})$:

$$\mathcal{SPEC} \models f_1(\overline{\alpha} + \nu \cdot \overline{\beta}) = f_2(\overline{\alpha}_2) \tag{6.128}$$

iff:

$$\forall \mu \in B(\overline{\alpha}_1 + \nu \cdot \overline{\beta}).\ \mathcal{SPEC} \models f(\overline{\alpha}_1 + \nu \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = f_2(\overline{\alpha}_2) \tag{6.129}$$

The result follows immediately from the special case $\nu = 0$.
Base Case: By proposition 35, the bottom elements are those of boundary of the lattice. If $\nu \in B(\overline{\alpha}_1)$, then by proposition 32 we know that $p(\overline{\alpha}_1 + \nu \cdot \overline{\beta})$ is false. From (6.16) it follows that:

$$B(\overline{\alpha}_1 + \nu \cdot \overline{\beta}) = \{0\}$$

This means that:

$$\begin{gathered}\forall \mu \in B(\overline{\alpha}_1 + \nu \cdot \overline{\beta}).\ \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \nu \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = f_2(\overline{\alpha}_2) \\ \text{iff} \\ \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \nu \cdot \overline{\beta} + 0 \cdot \overline{\beta}) = f_2(\overline{\alpha}_2) \\ \text{iff} \\ \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \nu \cdot \overline{\beta}) = f_2(\overline{\alpha}_2)\end{gathered}$$

Induction Step: Suppose that the claim is true for each $\nu' \prec_p \nu$. We will show the claim for ν. If $\nu \in B(\overline{\alpha}_1)$ then the proof is reduced to that of the base case above. If $\nu \in L(\overline{\alpha}_1)$, we know from proposition 31 that $p(\overline{\alpha}_1 + \nu \cdot \overline{\beta})$ is true and consequently:

$$\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \nu \cdot \overline{\beta}) = f_1(\overline{\alpha}_1 + \nu \cdot \overline{\beta} + \overline{\beta}) = f_1(\overline{\alpha}_1 + (\nu + 1) \cdot \overline{\beta}) \quad (6.130)$$

We will prove the equivalence in both directions.
(⟶): Suppose that:

$$\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \nu \cdot \overline{\beta}) = f_2(\overline{\alpha}_2) \quad (6.131)$$

We will show that:

$$\forall \mu \in B(\overline{\alpha}_1 + \nu \cdot \overline{\beta}).\ \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \nu \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = f_2(\overline{\alpha}_2) \quad (6.132)$$

From (6.131) and (6.130):

$$\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + (\nu + 1) \cdot \overline{\beta}) = f_2(\overline{\alpha}_2) \quad (6.133)$$

It is clear that $\nu + 1 \prec_p \nu$. From the induction hypothesis it follows that:

$$\forall \mu \in B(\overline{\alpha}_1 + (\nu + 1) \cdot \overline{\beta}).\ \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + (\nu + 1) \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = f_2(\overline{\alpha}_2) \quad (6.134)$$

By applying corollary 34 on (6.134), where:

$$P(\mu) = \left(\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \nu \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = f_2(\overline{\alpha}_2)\right)$$

we get (6.132).
(⟵): Suppose that:

$$\forall \mu \in B(\overline{\alpha}_1 + \nu \cdot \overline{\beta}).\ \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \nu \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = f_2(\overline{\alpha}_2) \quad (6.135)$$

We will show that:

$$\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \nu \cdot \overline{\beta}) = f_2(\overline{\alpha}_2) \quad (6.136)$$

By applying corollary 34 on (6.135), where:

$$P(\mu) = \left(\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + \nu \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = f_2(\overline{\alpha}_2)\right)$$

we get:

$$\forall \mu \in B(\overline{\alpha}_1 + (\nu + 1) \cdot \overline{\beta}).\ \mathcal{SPEC} \models f_1(\overline{\alpha}_1 + (\nu + 1) \cdot \overline{\beta} + \mu \cdot \overline{\beta}) = f_2(\overline{\alpha}_2) \quad (6.137)$$

It is clear that $\nu + 1 \prec_p \nu$. From (6.137) and the induction hypothesis it follows that:

$$\mathcal{SPEC} \models f_1(\overline{\alpha}_1 + (\nu + 1) \cdot \overline{\beta}) = f_2(\overline{\alpha}_2) \quad (6.138)$$

From (6.130) and (6.138) we get (6.136). □

Lemma 49 *Let f_1 and f_2 be function variables in acyclic systems of recurrence equations with a signature $\mathcal{SIG}$. Then there is a Presburger formula $P(\overline{x}_1, \overline{x}_2)$, such that for each $\overline{\alpha}_1$ and $\overline{\alpha}_2$, if $f_1(\overline{\alpha}_1) \neq \perp$ and $f_2(\overline{\alpha}_2) \neq \perp$, then:*

$$P(\overline{\alpha}_1, \overline{\alpha}_2) \quad \textit{iff} \quad (\mathcal{SPEC} \models f_1(\overline{\alpha}_1) = f_2(\overline{\alpha}_2))$$

where $\mathcal{SPEC}$ is the algebraic specification of the class of all $\mathcal{SIG}$-algebras.

Proof: Let the definitions of f_1 in E_1 and f_2 in E_2 be of the forms:

$$f_1(\overline{x}) \;=\; \mathcal{F}_1(\overline{x})$$

and:

$$f_2(\overline{x}) \;=\; \mathcal{F}_2(\overline{x})$$

respectively, where $\mathcal{F}_1(\overline{x})$ is of the form:

case
$p_{11}(\overline{x}) \implies t_{11}(\overline{x})$
$\vdots$
$p_{1k_1}(\overline{x}) \implies t_{1k_1}(\overline{x})$
endcase

and $\mathcal{F}_2(\overline{x})$ is of the form:

case
$p_{21}(\overline{x}) \implies t_{21}(\overline{x})$
$\vdots$
$p_{2k_2}(\overline{x}) \implies t_{2k_2}(\overline{x})$
endcase

For each $\overline{\alpha}_1$ and $\overline{\alpha}_2$, if $f_1(\overline{\alpha}_1) \neq \perp$ and $f_2(\overline{\alpha}_2) \neq \perp$, then there are $p_{1i}(\overline{x})$ and $p_{2j}(\overline{x})$, where $1 \leq i \leq k_1$ and $1 \leq j \leq k_2$, such that $p_{1i}(\overline{\alpha}_1)$ and $p_{2j}(\overline{\alpha}_2)$ are true.

We will use induction on $\mathcal{C}(E_1, f_1, E_2, f_2) = (\mathcal{C}(E_1, f_1), \mathcal{C}(E_2, f_2))$ to give a Presburger formula $P_{ij}(\overline{x}_1, \overline{x}_2)$ such that for each $\overline{\alpha}_1$ and $\overline{\alpha}_2$, if $p_{1i}(\overline{\alpha}_1)$ and $p_{2j}(\overline{\alpha}_2)$ are true, then $P_{ij}(\overline{\alpha}_1, \overline{\alpha}_2)$ iff $f_1(\overline{\alpha}_1) = f_2(\overline{\alpha}_2)$ for each $\mathcal{SIG}$-algebra A.

Base Case: If $\mathcal{C}(E_1, f_1, E_2, f_2) =< e_1, c_1, e_2, c_2 >$, where $c_1 = 0$ or $c_2 = 0$, i.e. $k_1 = 0$ or $k_2 = 0$, then $f_1(\overline{x}) = \perp$ or $f_2(\overline{x}) = \perp$, for all $\overline{x}$, and the proof is trivial.

Induction Step: Suppose that $\mathcal{C}(E_1, f_1, E_2, f_2) =< e_1, c_1, e_2, c_2 >$, where $c_1 > 0$ and $c_2 > 0$, i.e. $k_1 > 0$ and $k_2 > 0$. A number of cases are possible. In each case we will write $P_{ij}(\overline{x}_1, \overline{x}_2)$ as a Presburger formula.

1. If both $t_{1i}(\overline{x})$ and $t_{2j}(\overline{x})$ are stream expressions. Let $t_{1i}(\overline{x})$ and $t_{2j}(\overline{x})$ be of the forms $a_1(\overline{\ell}_1(\overline{x}))$ and $a_2(\overline{\ell}_2(\overline{x}))$ respectively. From the definition of semantics it follows that if $a_1 \neq a_2$ then:

$$P_{ij}(\overline{x}_1, \overline{x}_2) \;=\; \text{false}$$

which is a Presburger formula. If $a_1 = a_2$ then:

$$P_{ij}(\overline{x}_1, \overline{x}_2) \;=\; (\overline{\ell}_1(\overline{x}_1) = \overline{\ell}_2(\overline{x}_2))$$

which is a Presburger formula.
2. If $t_{1i}(\overline{x})$ is a stream expression and $t_{2j}(\overline{x})$ is a linear systolic term. Then from the definition of semantics it follows that:

$$P_{ij}(\overline{x}_1, \overline{x}_2) \quad = \quad \text{false}$$

which is a Presburger formula.
3. If $t_{2j}(\overline{x})$ is a stream expression and $t_{1i}(\overline{x})$ is a linear systolic term. This case is similar to case (2) above.
4. If $t_{1i}(\overline{x})$ is a function variable expression of the form $f_1'(\overline{x} + \overline{\beta})$, where $f_1' \neq f_1$. Let E_1' be the system of recurrence equations we get from E_1 by deleting the equation:

$$f_1(\overline{x}) \quad = \quad \mathcal{F}_1(\overline{x})$$

We note that f_1' is a function variable in E_1'. From the acyclicity of $\mathcal{F}_1(\overline{x})$ it follows that:

$$f_1(\overline{x}) \quad = \quad f_1'(\overline{x} + \overline{\beta})$$

where f_1' is considered as a function variable in E_1'. Also we know that:

$$\mathcal{C}(E_1', f_1', E_2, f_2) \quad < \quad \mathcal{C}(E_1, f_1, E_2, f_2)$$

From the induction hypothesis it follows that there is a Presburger formula $P_{ij}'(\overline{x}_1, \overline{x}_2)$ such that:

$$P_{ij}'(\overline{x}_1, \overline{x}_2) \quad \text{iff} \quad (\mathcal{SPEC} \models f_1'(\overline{x}_1) = f_2(\overline{x}_2))$$

Now $P_{ij}(\overline{x}_1, \overline{x}_2)$ can be defined to be:

$$P_{ij}(\overline{x}_1, \overline{x}_2) \quad = \quad P_{ij}'(\overline{x}_1 + \overline{\beta}, \overline{x}_2)$$

which is a Presburger formula.
5. If $t_{2j}(\overline{x})$ is a function variable expression of the form $f_2'(\overline{x} + \overline{\beta})$, where $f_2' \neq f_2$. This case is similar to case (4) above.
6. If $t_{1i}(\overline{x})$ is a function variable expression of the form $f_1(\overline{x} + \overline{\beta})$. By proposition 41 we can check whether $p_{1i}(\overline{x})$ is finite with respect to $\overline{\beta}$ or not. If $p_{1i}(\overline{x})$ is not finite with respect to $\overline{\beta}$ then it follows by lemma 44 that the proof is trivial. If $p_{1i}(\overline{x})$ is finite with respect to $\overline{\alpha}$ then let $B(\overline{x})$ be the boundary of the lattice with origin $\overline{x}$, basis $\overline{\beta}$ and constraint $p_{1i}(\overline{y})$. By lemma 48 it follows that:

$$(\mathcal{SPEC} \models f_1(\overline{x}_1) = f_2(\overline{x}_2)) \quad \text{iff} \quad \left(\forall \mu \in B(\overline{x}).\, \mathcal{SPEC} \models f(\overline{x}_1 + \mu \cdot \overline{\beta}) = f(\overline{x}_2)\right) \quad (6.139)$$

let $bf(\overline{x}, \mu)$ be the boundary formula corresponding to $B(\overline{x})$, then:

$$bf(\overline{x}, \mu) \quad = \quad (\mu \in B(\overline{x})) \quad (6.140)$$

Let E_1^* be the equation system we get from E_1 by replacing the equation:

$$f_1(\overline{x}) \quad = \quad \mathcal{F}_1(\overline{x})$$

in E_1 by the equation:

$$f_1^*(\overline{x}) \;=\; \mathcal{F}_1^*(\overline{x})$$

where $\mathcal{F}_1^*(\overline{x})$ is the result of deleting the case $p_{1i}(\overline{x}) \Rightarrow t_{1i}(\overline{x})$ in $\mathcal{F}_1(\overline{x})$. By proposition 32 we know that for each $\mu \in B(\overline{x})$, $p_{1i}(\overline{x} + \mu \cdot \overline{\beta})$ is false. By the acyclicity of $\mathcal{F}_1(\overline{x})$ it follows that:

$$\forall \mu \in B(\overline{x}).\ (f_1(\overline{x} + \mu \cdot \overline{\beta}) = f_1^*(\overline{x} + \mu \cdot \overline{\beta})) \tag{6.141}$$

We know that $\mathcal{C}(E_1^*, f_1^*, E_2, f_2) < \mathcal{C}(E_1, f_1, E_2, f_2)$. By the induction hypothesis it follows that there is a Presburger formula $P_{ij}^*(\overline{x}_1, \overline{x}_2)$ such that:

$$P_{ij}^*(\overline{x}_1, \overline{x}_2) \;\text{ iff }\; (\mathcal{SPEC} \models f_1^*(\overline{x}_1) = f_2(\overline{x}_2)) \tag{6.142}$$

From (6.139), (6.140), (6.141), and (6.142) it follows that $P_{ij}(\overline{x}_1, \overline{x}_2)$ can be defined as:

$$P_{ij}(\overline{x}_1, \overline{x}_2) \;=\; \forall \mu.\ bf(\overline{x}, \mu) \longrightarrow P_{ij}^*(\overline{x}_1 + \mu \cdot \overline{\beta}, \overline{x}_2)$$

which is a Presburger formula.

7. If $t_{2j}(\overline{x})$ is a function variable expression of the form $f_2(\overline{x} + \overline{\beta})$. This case is similar to case (6) above.

8. If $t_{1i}(\overline{x})$ is a linear systolic term in which f_1 does not occur. Let $t_{1i}(\overline{x})$ and $t_{2j}(\overline{x})$ be of the forms $g_1(t'_{11}(\overline{x}), \ldots, t'_{1n_1}(\overline{x}))$ and $g_2(t'_{21}(\overline{x}), \ldots, t'_{2n_2}(\overline{x}))$ respectively. Then if $g_1 \neq g_2$ then from the definition of semantics it follows that:

$$P_{ij}(\overline{x}_1, \overline{x}_2) \;=\; \text{false}$$

which is a Presburger formula. If $g_1 = g_2$ then let $n = n_1 = n_2$. From the definition of semantics it follows that:

$$(\mathcal{SPEC} \models f_1(\overline{x}_1) = f_2(\overline{x}_2)) \;\text{ iff }\; \bigwedge_{l=1}^{n} (\mathcal{SPEC} \models t'_{1l}(\overline{x}_1) = t'_{2l}(\overline{x}_2))$$

We will show that for each $1 \leq l \leq m$ there is a Presburger formula $P_{ij}^{(l)}(\overline{x}_1, \overline{x}_2)$ such that:

$$P_{ij}^{(l)}(\overline{x}_1, \overline{x}_2) \;\text{ iff }\; (\mathcal{SPEC} \models t'_{1l}(\overline{x}_1) = t'_{2l}(\overline{x}_2))$$

There are a number of cases. In each case we will write $P_{ij}^{(l)}(\overline{x}_1, \overline{x}_2)$ as a Presburger formula.

- If $t_{1l}(\overline{x})$ and $t_{2l}(\overline{x})$ are stream expressions. This case is similar to case (1) above.
- If $t'_{1l}(\overline{x})$ is a function variable expression, whose function variable is not f_1. This case is similar to case (4) above.
- If $t'_{2l}(\overline{x})$ is a function variable expression, whose function variable is not f_2. This case is similar to case (5) above.

It follows that $P_{ij}(\overline{x}_1, \overline{x}_2)$ can be defined as:

$$P_{ij}(\overline{x}_1, \overline{x}_2) \quad = \quad \bigwedge_{l=1}^{n} P_{ij}^{(l)}(\overline{x}_1, \overline{x}_2)$$

which is a Presburger formula.

9. If $t_{2j}(\overline{x})$ is a linear systolic term in which f_2 does not occur. This case is similar to case (8) above.

10. If both $t_{1i}(\overline{x})$ and $t_{2j}(\overline{x})$ are linear systolic terms. Let the operation symbols in $t_{1i}(\overline{x})$ and $t_{2j}(\overline{x})$ be g_1 and g_2 respectively. If $g_1 \neq g_2$ then from the definition of semantics it follows that:

$$P_{ij}(\overline{x}_1, \overline{x}_2) \quad = \quad \text{false}$$

which is a Presburger formula. If $g_1 = g_2$ then let $\overline{\delta}_1 =< \overline{\delta}_{11}, \ldots, \overline{\delta}_{1m} >$ and $\overline{\delta}_2 =< \overline{\delta}_{21}, \ldots, \overline{\delta}_{2m} >$ be the common recursive vectors of f_1 and f_2 in $t_{1i}(\overline{x})$ and $t_{2j}(\overline{x})$. Let $t'_{11}(\overline{x}), \ldots, t'_{1n}(\overline{x})$ and $t'_{21}(\overline{x}), \ldots, t'_{2n}(\overline{x})$ be the common non-recursive terms of f_1 and f_2 in $t_{1i}(\overline{x})$ and $t_{2j}(\overline{x})$. Let $L^c(\overline{x}_1, \overline{x}_2)$, $B^c(\overline{x}_1, \overline{x}_2)$ be the common lattice, and the boundary of the common lattice with origins $\overline{x}_1, \overline{x}_2$, bases $\overline{\delta}_1, \overline{\delta}_2$, and constraints $p_{1i}(\overline{x}), p_{2j}(\overline{x})$.

From lemma 47 it follows that:

$$\begin{gathered} \mathcal{SPEC} \models f_1(\overline{x}_1) = f_2(\overline{x}_2) \\ \text{iff} \\ \forall \overline{\mu} \in B^c(\overline{x}_1, \overline{x}_2).\ \mathcal{SPEC} \models f_1(\overline{x}_1 + \overline{\mu} \cdot \overline{\delta}_1) = f_2(\overline{x}_2 + \overline{\mu} \cdot \overline{\delta}_2) \\ \wedge \\ \bigwedge_{l=1}^{n} \left(\forall \overline{\mu} \in L^c(\overline{x}_1, \overline{x}_2).\ \mathcal{SPEC} \models t'_{1l}(\overline{x}_1 + \overline{\mu} \cdot \overline{\delta}_1) = t'_{2l}(\overline{x}_2 + \overline{\mu} \cdot \overline{\delta}_2) \right) \end{gathered} \tag{6.143}$$

Let $lf^c(\overline{x}_1, \overline{x}_2, \overline{\mu})$ and $bf^c(\overline{x}_1, \overline{x}_2, \overline{\mu})$ be the lattice and the boundary formulas of $L^c(\overline{x}_1, \overline{x}_2)$. This means that:

$$lf^c(\overline{x}_1, \overline{x}_2, \overline{\mu}) \quad = \quad (\overline{\mu} \in L^c(\overline{x}_1, \overline{x}_2)) \tag{6.144}$$

and:

$$bf^c(\overline{x}_1, \overline{x}_2, \overline{\mu}) \quad = \quad (\overline{\mu} \in B^c(\overline{x}_1, \overline{x}_2)) \tag{6.145}$$

Let E_1^* and E_2^* be the equation systems we get from E_1 and E_2 by replacing the equation:

$$f_1(\overline{x}) \quad = \quad \mathcal{F}_1(\overline{x})$$

in E_1 by the equation:

$$f_1^*(\overline{x}) \quad = \quad \mathcal{F}_1^*(\overline{x})$$

and the equation:

$$f_2(\overline{x}) \quad = \quad \mathcal{F}_2(\overline{x})$$

in E_2 by the equation:

$$f_2^*(\overline{x}) \;=\; \mathcal{F}_2^*(\overline{x})$$

where $\mathcal{F}_1^*(\overline{x})$ and $\mathcal{F}_2^*(\overline{x})$ are the results of deleting the cases $p_{1i}(\overline{x}) \Rightarrow t_{1i}(\overline{x})$ and $p_{2j}(\overline{x}) \Rightarrow t_{2j}(\overline{x})$ in $\mathcal{F}_1(\overline{x})$ and $\mathcal{F}_2(\overline{x})$ respectively. By proposition 32 we know that for each $\overline{\mu} \in B^c(\overline{x}_1, \overline{x}_2)$, $p_{1i}(\overline{x}_1 + \overline{\mu} \cdot \overline{\delta}_1) \wedge p_{2j}(\overline{x}_2 + \overline{\mu} \cdot \overline{\delta}_2)$ is false. From the acyclicity of $\mathcal{F}_1(\overline{x})$ and $\mathcal{F}_2(\overline{x})$ it follows that:

$$\forall \overline{\mu} \in B^c(\overline{x}_1, \overline{x}_2).\ (\neg p_{1i}(\overline{x}_1 + \overline{\mu} \cdot \overline{\delta}_1) \;\longrightarrow\; f_1(\overline{x}_1 + \overline{\mu} \cdot \overline{\delta}_1) = f_1^*(\overline{x}_1 + \overline{\mu} \cdot \overline{\delta}_1)) \quad (6.146)$$

and:

$$\forall \overline{\mu} \in B^c(\overline{x}_1, \overline{x}_2).\ (\neg p_{2j}(\overline{x}_2 + \overline{\mu} \cdot \overline{\delta}_2) \;\longrightarrow\; f_2(\overline{x}_2 + \overline{\mu} \cdot \overline{\delta}_2) = f_2^*(\overline{x}_2 + \overline{\mu} \cdot \overline{\delta}_2)) \quad (6.147)$$

We know that:

$$\mathcal{C}(E_1^*, f_1^*, E_2, f_2) \;<\; \mathcal{C}(E_1, f_1, E_2, f_2)$$

and:

$$\mathcal{C}(E_1, f_1, E_2^*, f_2^*) \;<\; \mathcal{C}(E_1, f_1, E_2, f_2)$$

By the induction hypothesis it follows that there are Presburger formulas $P_{ij1}^*(\overline{x}_1, \overline{x}_2)$ and $P_{ij2}^*(\overline{x}_1, \overline{x}_2)$ such that:

$$P_{ij1}^*(\overline{x}_1, \overline{x}_2) \quad \text{iff} \quad (f_1^*(\overline{x}_1) = f_2(\overline{x}_2)) \quad (6.148)$$

and:

$$P_{ij2}^*(\overline{x}_1, \overline{x}_2) \quad \text{iff} \quad (f_1(\overline{x}_1) = f_2^*(\overline{x}_2)) \quad (6.149)$$

In a similar manner to case (8) above we can show that for each $1 \leq l \leq m$ there is a Presburger formula $P_{ij}^{(l)}(\overline{x}_1, \overline{x}_2)$ such that:

$$P_{ij}^{(l)}(\overline{x}_1, \overline{x}_2) \quad \text{iff} \quad (t_{1l}(\overline{x}_1) = t_{2l}(\overline{x}_2)) \quad (6.150)$$

From (6.143), (6.144), (6.145), (6.146), (6.147), (6.148), (6.149), and (6.150), it follows that $P_{ij}(\overline{x})$ can be defined as:

$$P_{ij}(\overline{x}) = \left(\begin{array}{c} \left(\forall \overline{\mu}.\ bf^c(\overline{x}_1, \overline{x}_2, \overline{\mu}) \;\longrightarrow\; \neg p_{1i}(\overline{x}_1 + \overline{\mu} \cdot \overline{\delta}_1) \;\longrightarrow\; P_{ij1}^*(\overline{x}_1 + \overline{\mu} \cdot \overline{\delta}_1, \overline{x}_2 + \overline{\mu} \cdot \overline{\delta}_2)\right) \\ \wedge \\ \left(\forall \overline{\mu}.\ bf^c(\overline{x}_1, \overline{x}_2, \overline{\mu}) \;\longrightarrow\; \neg p_{2j}(\overline{x}_2 + \overline{\mu} \cdot \overline{\delta}_2) \;\longrightarrow\; P_{ij2}^*(\overline{x}_1 + \overline{\mu} \cdot \overline{\delta}_1, \overline{x}_2 + \overline{\mu} \cdot \overline{\delta}_2)\right) \\ \wedge \\ \bigwedge_{l=1}^{n} \left(\forall \overline{\mu}.\ lf^c(\overline{x}_1, \overline{x}_2, \overline{\mu}) \;\longrightarrow\; P_{ij}^{(l)}(\overline{x}_1 + \overline{\mu} \cdot \overline{\delta}_1, \overline{x}_2 + \overline{\mu} \cdot \overline{\delta}_2)\right) \end{array} \right)$$

which is a Presburger formula.

Now the formula $P(\overline{x}_1, \overline{x}_2)$ in the claim of the lemma can be defined as:

$$P(\overline{x}_1, \overline{x}_2) \;=\; \bigwedge_{i=1}^{k_1} \bigwedge_{j=1}^{k_2} (p_{1i}(\overline{x}_1) \;\wedge\; p_{2j}(\overline{x}_2) \;\wedge\; P_{ij}(\overline{x}_1, \overline{x}_2))$$

which is a Presburger formula. □

6.6.6 Equality of Systems of Recurrence Equations

Theorem 50 *Let f_1 and f_2 be function variables in acyclic systems of recurrence equations with a signature $\mathcal{SIG}$. Then there is a Presburger formula $P(\overline{x}_1, \overline{x}_2)$ such that:*

$$P(\overline{x}_1, \overline{x}_2) \quad \textit{iff} \quad (\mathcal{SPEC} \models f_1(\overline{x}_1) = f_2(\overline{x}_2))$$

where $\mathcal{SPEC}$ is the algebraic specification of the class of all $\mathcal{SIG}$-algebras.

Proof: We know that:

$$\mathcal{SPEC} \models (f_1(\overline{x}_1) = f_2(\overline{x}_2))$$

iff

$$\mathcal{SPEC} \models \left(\begin{array}{c} (f_1(\overline{x}_1) = \bot) \wedge (f_2(\overline{x}_2) = \bot) \\ \vee \\ (f_1(\overline{x}_1) \neq \bot) \wedge (f_2(\overline{x}_2) \neq \bot) \wedge (f_1(\overline{x}_1) = f_2(\overline{x}_2)) \end{array} \right) \tag{6.151}$$

From lemma 46 it follows that there are Presburger formulas $P_1(\overline{x})$ and $P_2(\overline{x})$ such that:

$$P_1(\overline{x}) \quad \text{iff} \quad (f_1(\overline{x}) = \bot) \tag{6.152}$$

and:

$$P_2(\overline{x}) \quad \text{iff} \quad (f_2(\overline{x}) = \bot) \tag{6.153}$$

From lemma 49 it follows that there is a Presburger formula $P_3(\overline{x}_1, \overline{x}_2)$ such that:

$$\begin{array}{l} (f_1(\overline{x}_1) \neq \bot) \wedge (f_2(\overline{x}_2) \neq \bot) \longrightarrow \\ \quad (P_3(\overline{x}_1, \overline{x}_2) \text{ iff } (\mathcal{SPEC} \models f_1(\overline{x}_1) = f_2(\overline{x}_2))) \end{array} \tag{6.154}$$

From (6.151), (6.152), (6.153), and (6.154) it follows that:

$$\mathcal{SPEC} \models (f_1(\overline{x}_1) = f_2(\overline{x}_2))$$

iff

$$(P_1(\overline{x}_1) \wedge P_2(\overline{x}_2)) \vee (\neg P_1(\overline{x}_1) \wedge \neg P_2(\overline{x}_2) \wedge P_3(\overline{x}_1, \overline{x}_2))$$

It follows that $P(\overline{x}_1, \overline{x}_2)$ in the claim of the lemma can be defined as:

$$P(\overline{x}_1, \overline{x}_2) \;=\; (P_1(\overline{x}_1) \wedge P_2(\overline{x}_2)) \vee (\neg P_1(\overline{x}_1) \wedge \neg P_2(\overline{x}_2) \wedge P_3(\overline{x}_1, \overline{x}_2))$$

which is a Presburger formula. □

Corollary 51 *Let f_1 and f_2 be function variables in acyclic systems of recurrence equations over a signature $\mathcal{SIG}$. Let $p(\overline{x}_1, \overline{x}_2)$ be a conjunction of linear predicates. Then it is decidable to check the validity of:*

$$\mathcal{SPEC} \;\models\; p(\overline{x}_1, \overline{x}_2) \longrightarrow (f_1(\overline{x}_1) = f_2(\overline{x}_2))$$

where $\mathcal{SPEC}$ is the algebraic specification of all $\mathcal{SIG}$-algebras.

Proof: From theorem 50 it follows that there is a Presburger formual $P'(\overline{x}_1, \overline{x}_2)$ such that:

$$P'(\overline{x}_1, \overline{x}_2) \quad \text{iff} \quad (\mathcal{SPEC} \models f_1(\overline{x}_1) = f_2(\overline{x}_2))$$

We define:

$$P(\overline{x}_1, \overline{x}_2) \quad = \quad (p(\overline{x}_1, \overline{x}_2) \longrightarrow P'(\overline{x}_1, \overline{x}_2))$$

It is obvious that $P(\overline{x}_1, \overline{x}_2)$ is a Presburger formula and that:

$$P(\overline{x}_1, \overline{x}_2) \quad \text{iff} \quad \mathcal{SPEC} \models p(\overline{x}_1, \overline{x}_2) \longrightarrow (f_1(\overline{x}_1) = f_2(\overline{x}_2))$$

The decidability of the problem follows from the decidability of Presburger formulas (see section 6.6.1). □

6.6.7 Second Step of Verification of The String Matching Circuit: Checking Equalities

In section 7.5.2 we carried out the first step of the verification of the string matching circuit. We described the signals of the circuit as an acyclic system of recurrence equations. In this section we give a sketch of how the second step can be performed. We use the ideas of theorem 50 and corollary 51 to check whether the values of *loc* at different cells and time instants agree with the values demanded by the circuit specification.

Let $\mathcal{SIG} =< \{D\}, \{g_1, g_2, g_3\} >$, and let $\mathcal{SPEC}$ be the algebraic specification of the class of all $\mathcal{SIG}$-algebras. We recall from section 3.2.2 that the specification formula of the circuit was of the form:

$$\begin{array}{l} spec(x, \ell_1, \ell_2, t) = \\ \quad (0 < \ell_1) \wedge (0 < \ell_2) \wedge (x = \ell_2 - \ell_1) \wedge (t = \ell_1 + \ell_2 + 1) \longrightarrow \\ \qquad (loc(x, \ell_1, \ell_2, t) = d(\ell_1, \ell_2, \ell_1, \ell_2)) \end{array}$$

Let:

$$p(x, \ell_1, \ell_2, t) \quad = \quad (0 < \ell_1) \wedge (0 < \ell_2) \wedge (x = \ell_2 - \ell_1) \wedge (t = \ell_1 + \ell_2 + 1)$$

According to corollary 51 there is a Presburger formula $P(x, \ell_1, \ell_2, t)$ such that:

$$P(x, \ell_1, \ell_2, t) \quad \text{iff} \quad \mathcal{SPEC} \models spec(x, \ell_1, \ell_2, t)$$

Furthermore $P(x, \ell_1, \ell_2, t)$ is of the form:

$$p(x, \ell_1, \ell_2, t) \quad \longrightarrow \quad P'(x, \ell_1, \ell_2, t)$$

where $P'(x, \ell_1, \ell_2, t)$ is a Presburger formula such that:

$$P'(x, \ell_1, \ell_2, t) \quad \text{iff} \quad \mathcal{SPEC} \models loc(x, \ell_1, \ell_2, t) = d(\ell_1, \ell_2, \ell_1, \ell_2)$$

According to theorem 50, $P'(x, \ell_1, \ell_2, t)$ is of the form:

$$P'(x, \ell_1, \ell_2, t) \quad = \quad \left(\begin{array}{c} (P_1(\ell_1, \ell_2, \ell_1, \ell_2) \wedge P_2(x, \ell_1, \ell_2, t)) \\ \vee \\ (\neg P_1(\ell_1, \ell_2, \ell_1, \ell_2) \wedge \neg P_2(x, \ell_1, \ell_2, t) \wedge P_3(x, \ell_1, \ell_2, t)) \end{array} \right) \quad (6.155)$$

where:

$$
\begin{array}{rcl}
P_1(y_1, y_2, \ell_1, \ell_2) & \text{iff} & (d(y_1, y_2, \ell_1, \ell_2) = \perp) \\
P_2(x, \ell_1, \ell_2, t) & \text{iff} & (loc(x, \ell_1, \ell_2, t) = \perp) \\
P_3(x, \ell_1, \ell_2, t) & \text{iff} & \left(\begin{array}{c} \neg P_1(\ell_1, \ell_2, \ell_1, \ell_2) \ \wedge \ \neg P_2(x, \ell_1, \ell_2, t) \\ \longrightarrow \\ \mathcal{SPEC} \models (loc(x, \ell_1, \ell_2, t) = d(x, \ell_1, \ell_2, t)) \end{array} \right)
\end{array}
$$

In the following we will sketch how $P_1(y_1, y_2, \ell_1, \ell_2)$, $P_2(x, \ell_1, \ell_2, t)$ and $P_3(x, \ell_1, \ell_2, t)$ can be constructed. From (3.47) we know that:

$$
\begin{array}{l}
d(y_1, y_2, \ell_1, \ell_2) = \\
\textbf{case} \\
\qquad \left\{ \begin{array}{c} y_1 = 0 \\ y_2 = 0 \end{array} \right\} \implies g_1 \\
\qquad \left\{ \begin{array}{c} 0 < y_1 \leq \ell_1 \\ y_2 = 0 \end{array} \right\} \implies g_2(d(y_1 - 1, y_2, \ell_1, \ell_2)) \\
\qquad \left\{ \begin{array}{c} y_1 = 0 \\ 0 < y_2 \leq \ell_2 \end{array} \right\} \implies g_2(d(y_1, y_2 - 1, \ell_1, \ell_2)) \\
\qquad \left\{ \begin{array}{c} 0 < y_1 \leq \ell_1 \\ 0 < y_2 \leq \ell_2 \end{array} \right\} \implies g_3 \ (a(y_1), b(y_2), d(y_1, y_2 - 1, \ell_1, \ell_2), \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad d(y_1 - 1, y_2, \ell_1, \ell_2), d(y_1 - 1, y_2 - 1, \ell_1, \ell_2)) \\
\textbf{endcase}
\end{array}
\tag{6.156}
$$

From section 6.5.1 we know that:

$$
\begin{array}{l}
loc(x,\ell_1,\ell_2,t) = \\
\textbf{case} \\
\left\{ \begin{array}{c} 0 \le t < 2 \\ -2\ell_1 - 1 \le x < 2\ell_2 + 1 \\ x = 0 \end{array} \right\} \implies g_1 \\
\\
\left\{ \begin{array}{c} 2 \le t \\ -2\ell_1 - 1 \le x \le 2\ell_2 \\ x < 0 \\ t = -x + 1 \end{array} \right\} \implies g_2(loc(x+1,\ell_1,\ell_2,t-1)) \\
\\
\left\{ \begin{array}{c} 2 \le t \\ -2\ell_1 \le x \le 2\ell_2 + 1 \\ 0 < x \\ t = x + 1 \end{array} \right\} \implies g_2(loc(x-1,\ell_1,\ell_2,t-1)) \\
\\
\left\{ \begin{array}{c} 2 \le t \\ -2\ell_1 \le x \le 2\ell_2 \\ -x + 1 < t \\ x + 1 < t \\ t + x + 1 \ mod\ 2 = 0 \end{array} \right\} \implies g_3(\ out_1(x-1,\ell_1,\ell_2,t-1), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad out_2(x+1,\ell_1,\ell_2,t-1), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad loc(x-1,\ell_1,\ell_2,t-1), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad loc(x+1,\ell_1,\ell_2,t-1), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad loc(x,\ell_1,\ell_2,t-2)) \\
\textbf{endcase}
\end{array}
\tag{6.157}
$$

Recalling equation (6.156) above, let:

$$
\begin{array}{rcl}
pd_1(y_1,y_2,\ell_1,\ell_2) & = & (y_1 = 0) \wedge (y_2 = 0) \\
pd_2(y_1,y_2,\ell_1,\ell_2) & = & (0 < y_1 \le \ell_1) \wedge (y_2 = 0) \\
pd_3(y_1,y_2,\ell_1,\ell_2) & = & (y_1 = 0) \wedge (0 < y_2 \le \ell_2) \\
pd_4(y_1,y_2,\ell_1,\ell_2) & = & (0 < y_1 \le \ell_1) \wedge (0 < y_2 \le \ell_2) \\
td_1(y_1,y_2,\ell_1,\ell_2) & = & g_1 \\
td_2(y_1,y_2,\ell_1,\ell_2) & = & g_2(d(y_1 - 1,y_2,\ell_1,\ell_2)) \\
td_3(y_1,y_2,\ell_1,\ell_2) & = & g_2(d(y_1,y_2 - 1,\ell_1,\ell_2)) \\
td_4(y_1,y_2,\ell_1,\ell_2) & = & g_3\ (a(y_1),b(y_2),d(y_1,y_2-1,\ell_1,\ell_2), \\
 & & \quad d(y_1-1,y_2,\ell_1,\ell_2),d(y_1-1,y_2-1,\ell_1,\ell_2))
\end{array}
$$

Recalling equation (6.157) above, let:

$$
\begin{array}{rcl}
pl_1(x,\ell_1,\ell_2,t) & = & (0 \le t < 2) \wedge (-2\ell_1 - 1 \le x < 2\ell_2 + 1) \wedge (x = 0) \\
pl_2(x,\ell_1,\ell_2,t) & = & (2 \le t) \wedge (-2\ell_1 - 1 \le x \le 2\ell_2) \wedge (x < 0) \wedge (t = -x + 1) \\
pl_3(x,\ell_1,\ell_2,t) & = & (2 \le t) \wedge (-2\ell_1 \le x \le 2\ell_2 + 1) \wedge (0 < x) \wedge (t = x + 1) \\
pl_4(x,\ell_1,\ell_2,t) & = & (2 \le t) \wedge (-2\ell_1 \le x \le 2\ell_2) \wedge (-x + 1 < t) \wedge \\
 & & (x + 1 < t) \wedge (t + x + 1\ mod\ 2 = 0) \\
tl_1(x,\ell_1,\ell_2,t) & = & g_1
\end{array}
$$

$$
\begin{aligned}
tl_2(x,\ell_1,\ell_2,t) &= g_2(loc(x+1,\ell_1,\ell_2,t-1)) \\
tl_3(x,\ell_1,\ell_2,t) &= g_2(loc(x-1,\ell_1,\ell_2,t-1)) \\
tl_4(x,\ell_1,\ell_2,t) &= g_3(\ out_1(x-1,\ell_1,\ell_2,t-1), out_2(x+1,\ell_1,\ell_2,t-1), \\
&\qquad loc(x-1,\ell_1,\ell_2,t-1), loc(x+1,\ell_1,\ell_2,t-1), loc(x,\ell_1,\ell_2,t-2))
\end{aligned}
$$

Construction of $P_1(y_1,y_2,\ell_1,\ell_2)$:
According to lemma 46, $P_1(y_1,y_2,\ell_1,\ell_2)$ is of the form:

$$
P_1(y_1,y_2,\ell_1,\ell_2) = \left(\begin{array}{c} \Lambda_{i=1}^{4} \neg pd_i(y_1,y_2,\ell_1,\ell_2) \\ \vee \\ \vee_{i=1}^{4} (pd_i(y_1,y_2,\ell_1,\ell_2) \wedge Pd_i(y_1,y_2,\ell_1,\ell_2)) \end{array} \right)
$$

where:

$$
Pd_i(y_1,y_2,\ell_1,\ell_2) \text{ iff } (td_i(y_1,y_2,\ell_1,\ell_2) = \perp)
$$

It can be checked (by linear integer programming) that:

$$
p(x,\ell_1,\ell_2,t) \wedge \left(\bigwedge_{i=1}^{4} \neg pd_i(\ell_1,\ell_2,\ell_1,\ell_2) \right)
$$

is unsatisfiable, and that:

$$
p(x,\ell_1,\ell_2,t) \wedge pd_i(\ell_1,\ell_2,\ell_1,\ell_2)
$$

is unsatisfiable for $i = 1,2,3$. This means that $P_1(\ell_1,\ell_2,\ell_1,\ell_2)$ in (6.155) can be replaced by:

$$
pd_4(\ell_1,\ell_2,\ell_1,\ell_2) \wedge Pd_4(\ell_1,\ell_2,\ell_1,\ell_2)
$$

Now we sketch how $Pd_4(y_1,y_2,\ell_1,\ell_2)$ can be constructed.
Construction of $Pd_4(y_1,y_2,\ell_1,\ell_2)$
We use the method of lemma 46 to construct $Pd_4(y_1,y_2,\ell_1,\ell_2)$. The recursive vector $\overline{\delta}$ of d in $td_4(y_1,y_2,\ell_1,\ell_2)$ is given by:

$$
\overline{\delta} = << 0,-1,0,0 >, < -1,0,0,0 >, < -1,-1,0,0 >>
$$

The non-recursive terms of d in $td_4(y_1,y_2,\ell_1,\ell_2)$ are $a(y_1)$ and $b(y_2)$.

It can be shown by the method of proposition 41 that $pd_4(y_1,y_2,\ell_1,\ell_2)$ is finite with respect to $\overline{\delta}$. From the method of lemma 46 we know that:

$$
Pd_4(y_1,y_2,\ell_1,\ell_2)
$$

iff

$$
\left(\begin{array}{c} \exists < \mu_1,\mu_2,\mu_3 > \in B(y_1,y_2,\ell_1,\ell_2).\ d(y_1-\mu_2-\mu_3, y_2-\mu_1-\mu_3, \ell_1,\ell_2) = \perp \\ \vee \\ \exists < \mu_1,\mu_2,\mu_3 > \in L(y_1,y_2,\ell_1,\ell_2).\ a(y_1-\mu_2-\mu_3) = \perp \\ \vee \\ \exists < \mu_1,\mu_2,\mu_3 > \in L(y_1,y_2,\ell_1,\ell_2).\ b(y_2-\mu_1-\mu_3) = \perp \end{array} \right) \quad (6.158)
$$

where $L(y_1, y_2, \ell_1, \ell_2)$ and $B(y_1, y_2, \ell_1, \ell_2)$ are the lattice and the boundary of the lattice with origin $< y_1, y_2, \ell_1, \ell_2 >$, basis $\overline{\delta}$, and constraint $pd_4(y_1, y_2, \ell_1, \ell_2)$. We know that:

$$\forall < \mu_1, \mu_2, \mu_3 > \in B(y_1, y_2, \ell_1, \ell_2). \left(\begin{array}{c} d(y_1 - \mu_2 - \mu_3, y_2 - \mu_1 - \mu_3, \ell_1, \ell_2) \\ = \\ d^*(y_1 - \mu_2 - \mu_3, y_2 - \mu_1 - \mu_3, \ell_1, \ell_2) \end{array} \right) \quad (6.159)$$

where:

$$\begin{array}{l} d^*(y_1, y_2, \ell_1, \ell_2) = \\ \textbf{case} \\ \qquad \left\{ \begin{array}{c} y_1 = 0 \\ y_2 = 0 \end{array} \right\} \implies g_1 \\ \\ \qquad \left\{ \begin{array}{c} 0 < y_1 \leq \ell_1 \\ y_2 = 0 \end{array} \right\} \implies g_2(d^*(y_1 - 1, y_2, \ell_1, \ell_2)) \\ \\ \qquad \left\{ \begin{array}{c} y_1 = 0 \\ 0 < y_2 \leq \ell_2 \end{array} \right\} \implies g_2(d^*(y_1, y_2 - 1, \ell_1, \ell_2)) \\ \textbf{endcase} \end{array}$$

As the equation system is acyclic we know that $pd_4(y_1, y_2, \ell_1, \ell_2)$ is stable with respect to $\overline{\delta}$. By corollary 40 it follows that:

$$(< \mu_1, \mu_2, \mu_3 > \in L(y_1, y_2, \ell_1, \ell_2)) \quad \text{iff} \quad lf(y_1, y_2, \ell_1, \ell_2, \mu_1, \mu_2, \mu_3) \quad (6.160)$$

where $lf(y_1, y_2, \ell_1, \ell_2, \mu_1, \mu_2, \mu_3)$ is the lattice formula corresponding to $L(y_1, y_2, \ell_1, \ell_2)$, and is defined by:

$$\begin{array}{l} lf(y_1, y_2, \ell_1, \ell_2, \mu_1, \mu_2, \mu_3) = \\ \qquad pd_4(y_1, y_2, \ell_1, \ell_2) \ \wedge \ pd_4(y_1 - \mu_2 - \mu_3, y_2 - \mu_1 - \mu_3, \ell_1, \ell_2) \end{array}$$

Similarly:

$$(< \mu_1, \mu_2, \mu_3 > \in B(y_1, y_2, \ell_1, \ell_2)) \quad \text{iff} \quad bf(y_1, y_2, \ell_1, \ell_2, \mu_1, \mu_2, \mu_3) \quad (6.161)$$

where $bf(y_1, y_2, \ell_1, \ell_2, \mu_1, \mu_2, \mu_3)$ is the boundary formula corresponding to $B(y_1, y_2, \ell_1, \ell_2)$, and is defined by:

$$\begin{array}{l} bf(y_1, y_2, \ell_1, \ell_2, \mu_1, \mu_2, \mu_3) = \\ \left(\begin{array}{c} \left(\begin{array}{c} \neg pd_4(y_1, y_2, \ell_1, \ell_2) \\ \wedge \\ (\mu_1 = 0) \ \wedge \ (\mu_2 = 0) \ \wedge \ (\mu_3 = 0) \end{array} \right) \\ \vee \\ \left(\begin{array}{c} pd_4(y_1, y_2, \ell_1, \ell_2) \\ \wedge \\ \neg pd_4(y_1 - \mu_2 - \mu_3, y_2 - \mu_1 - \mu_3, \ell_1, \ell_2) \\ \wedge \\ \left(\begin{array}{c} pd_4(y_1 - \mu_2 - \mu_3, y_2 - \mu_1 - \mu_3 + 1, \ell_1, \ell_2) \\ \vee \\ pd_4(y_1 - \mu_2 - \mu_3 + 1, y_2 - \mu_1 - \mu_3, \ell_1, \ell_2) \\ \vee \\ pd_4(y_1 - \mu_2 - \mu_3 + 1, y_2 - \mu_1 - \mu_3 + 1, \ell_1, \ell_2) \end{array} \right) \end{array} \right) \end{array} \right) \end{array}$$

We know also from the method of lemma 46 that:

$$(a(y_1 - \mu_2 - \mu_3) = \perp) \;=\; \text{false} \tag{6.162}$$

and:

$$(b(y_2 - \mu_1 - \mu_3) = \perp) \;=\; \text{false} \tag{6.163}$$

From (6.158), (6.159), (6.160), (6.161), (6.162), and (6.163) we get:

$$Pd_4(y_1, y_2, \ell_1, \ell_2) = \tag{6.164}$$
$$\exists \mu_1, \mu_2, \mu_3. \left(\begin{array}{c} bf(y_1, y_2, \ell_1, \ell_2, \mu_1, \mu_2, \mu_3) \\ \wedge \\ d^*(y_1 - \mu_2 - \mu_3, y_2 - \mu_1 - \mu_3, \ell_1, \ell_2) = \perp \end{array} \right)$$

By applying lemma 46 recursively on $d^*(y_1, y_2, \ell_1, \ell_2, \mu_1, \mu_2, \mu_3)$ we get a Presburger formula $Pd_4^*(y_1, y_2, \ell_1, \ell_2, \mu_1, \mu_2, \mu_3)$ such that:

$$Pd_4^*(y_1, y_2, \ell_1, \ell_2, \mu_1, \mu_2, \mu_3) \quad \text{iff} \quad (d^*(y_1 - \mu_2 - \mu_3, y_2 - \mu_1 - \mu_3, \ell_1, \ell_2) = \perp) \tag{6.165}$$

From (6.164) and (6.165) we get:

$$Pd_4(y_1, y_2, \ell_1, \ell_2) \;=\; \exists \mu_1, \mu_2, \mu_3. \left(\begin{array}{c} bf(y_1, y_2, \ell_1, \ell_2, \mu_1, \mu_2, \mu_3) \\ \wedge \\ Pd_4^*(y_1, y_2, \ell_1, \ell_2, \mu_1, \mu_2, \mu_3) \end{array} \right)$$

which is a Presburger formula.

Construction of $P_2(x, \ell_1, \ell_2, t)$
The formula $P_2(x, \ell_1, \ell_2, t)$ can be constructed in a similar manner to $P_1(\ell_1, \ell_2, \ell_1, \ell_2)$.

Construction of $P_3(x, \ell_1, \ell_2, t)$
According to lemma 49, $P_3(x, \ell_1, \ell_2, t)$ is of the form:

$$P_3(x, \ell_1, \ell_2, t) \;=\; \bigwedge_{i=1}^{4} \bigwedge_{j=1}^{4} (pd_i(\ell_1, \ell_2, \ell_1, \ell_2) \wedge pl_j(x, \ell_1, \ell_2, t) \wedge P_{ij}(x, \ell_1, \ell_2, t))$$

where:

$$P_{ij}(x, \ell_1, \ell_2, t) \quad \text{iff} \quad \mathcal{SPEC} \models (td_i(\ell_1, \ell_2, \ell_1, \ell_2) = tl_j(x, \ell_1, \ell_2, t))$$

It can be checked (by integer linear programming) that:

$$p(x, \ell_1, \ell_2, t) \wedge pd_i(\ell_1, \ell_2, \ell_1, \ell_2) \wedge pl_j(x, \ell_1, \ell_2, t)$$

is unsatisfiable if $1 \leq i \leq 3$ or $1 \leq j \leq 3$. This means that $P_3(x, \ell_1, \ell_2, t)$ in (6.155) can be replaced by:

$$pd_4(\ell_1, \ell_2, \ell_1, \ell_2) \wedge pl_4(x, \ell_1, \ell_2, t) \wedge P_{4,4}(x, \ell_1, \ell_2, t)$$

Now we sketch how $P_{4,4}$ can be constructed.

Construction of $P_{4,4}$

We use the method of lemma 49 to construct $P_{4,4}(x, \ell_1, \ell_2, t)$.

The common recursive vectors of d and loc in $pd_4(y_1, y_2, \ell_1, \ell_2)$ and $pl_4(x, \ell_1, \ell_2, t)$ are given by:

$$\overline{\delta}_1 = << 0, -1, 0, 0 >, < -1, 0, 0, 0 >, < -1, -1, 0, 0 >>$$

and:

$$\overline{\delta}_2 = << -1, 0, 0, -1 >, < 1, 0, 0, -1 >, < 0, 0, 0, -2 >>$$

The common non-recursive terms of d and loc in $pd_4(y_1, y_2, \ell_1, \ell_2)$ and $pl_4(x, \ell_1, \ell_2, t)$ are $a(y_1)$, $b(y_2)$, and $out_1(x, \ell_1, \ell_2, t)$, $out_2(x, \ell_1, \ell_2, t)$.

From the method of lemma 49 we know that:

$$P_{4,4}(x, \ell_1, \ell_2, t)$$

iff

$$\left(\begin{array}{l} \left(\begin{array}{l} \forall < \mu_1, \mu_2, \mu_3 > \in B^c(x, \ell_1, \ell_2, t). \\ \left(\begin{array}{c} d(\ell_1 - \mu_2 - \mu_3, \ell_2 - \mu_1 - \mu_3, \ell_1, \ell_2) \\ = \\ loc(x - \mu_1 + \mu_2, \ell_1, \ell_2, t - \mu_1 - \mu_2 - 2\mu_3) \end{array} \right) \end{array} \right) \\ \\ \wedge \\ \\ \left(\begin{array}{l} \forall < \mu_1, \mu_2, \mu_3 > \in L^c(x, \ell_1, \ell_2, t). \\ \left(\begin{array}{c} a(\ell_1 - \mu_2 - \mu_3) \\ = \\ out_1(x - \mu_1 + \mu_2 - 1, \ell_1, \ell_2, t - \mu_1 - \mu_2 - 2\mu_3 - 1) \end{array} \right) \end{array} \right) \\ \\ \wedge \\ \\ \left(\begin{array}{l} \forall < \mu_1, \mu_2, \mu_3 > \in L^c(x, \ell_1, \ell_2, t). \\ \left(\begin{array}{c} b(\ell_2 - \mu_1 - \mu_3) \\ = \\ out_2(x - \mu_1 + \mu_2 + 1, \ell_1, \ell_2, t - \mu_1 - \mu_2 - 2\mu_3 - 1) \end{array} \right) \end{array} \right) \end{array} \right) \tag{6.166}$$

where $L^c(x, \ell_1, \ell_2, t)$ and $B^c(x, \ell_1, \ell_2, t)$ are the common lattice and the boundary of the common lattice with origins $< \ell_1, \ell_2, \ell_1, \ell_2 >$ and $< x, \ell_1, \ell_2, t >$, bases $\overline{\delta}_1$ and $\overline{\delta}_2$ and constraints $pd_4(\ell_1, \ell_2, \ell_1, \ell_2)$ and $pl_4(x, \ell_1, \ell_2, t)$. We know that:

$$\forall < \mu_1, \mu_2, \mu_3 > \in B^c(x, \ell_1, \ell_2, t). \left(\begin{array}{c} \neg pd_4(\ell_1 - \mu_2 - \mu_3, \ell_2 - \mu_1 - \mu_3, \ell_1, \ell_2) \\ \longrightarrow \\ \left(\begin{array}{c} d(\ell_1 - \mu_2 - \mu_3, \ell_2 - \mu_1 - \mu_3, \ell_1, \ell_2) \\ = \\ d^*(\ell_1 - \mu_2 - \mu_3, \ell_2 - \mu_1 - \mu_3, \ell_1, \ell_2) \end{array} \right) \end{array} \right) \tag{6.167}$$

and:

$$\forall < \mu_1, \mu_2, \mu_3 > \in B^c(x, \ell_1, \ell_2, t). \left(\begin{array}{c} \neg pl_4(x - \mu_1 + \mu_2, \ell_1, \ell_2, t - \mu_1 - \mu_2 - 2\mu_3) \\ \longrightarrow \\ \left(\begin{array}{c} loc(x - \mu_1 + \mu_2, \ell_1, \ell_2, t - \mu_1 - \mu_2 - 2\mu_3) \\ = \\ loc^*(x - \mu_1 + \mu_2, \ell_1, \ell_2, t - \mu_1 - \mu_2 - 2\mu_3) \end{array} \right) \end{array} \right) \quad (6.168)$$

where:

$$\begin{array}{l} d^*(y_1, y_2, \ell_1, \ell_2) = \\ \textbf{case} \\ \qquad \left\{ \begin{array}{c} y_1 = 0 \\ y_2 = 0 \end{array} \right\} \implies g_1 \\ \qquad \left\{ \begin{array}{c} 0 < y_1 \leq \ell_1 \\ y_2 = 0 \end{array} \right\} \implies g_2(d^*(y_1 - 1, y_2, \ell_1, \ell_2)) \\ \qquad \left\{ \begin{array}{c} y_1 = 0 \\ 0 < y_2 \leq \ell_2 \end{array} \right\} \implies g_2(d^*(y_1, y_2 - 1, \ell_1, \ell_2)) \\ \textbf{endcase} \end{array}$$

and:

$$\begin{array}{l} loc^*(x, \ell_1, \ell_2, t) \; = \\ \textbf{case} \\ \qquad \left\{ \begin{array}{c} 0 \leq t < 2 \\ -2\ell_1 - 1 \leq x < 2\ell_2 + 1 \\ x = 0 \end{array} \right\} \implies g_1 \\ \qquad \left\{ \begin{array}{c} 2 \leq t \\ -2\ell_1 - 1 \leq x \leq 2\ell_2 \\ x < 0 \\ t = -x + 1 \end{array} \right\} \implies g_2(loc^*(x + 1, \ell_1, \ell_2, t - 1)) \\ \qquad \left\{ \begin{array}{c} 2 \leq t \\ -2\ell_1 \leq x \leq 2\ell_2 + 1 \\ 0 < x \\ t = x + 1 \end{array} \right\} \implies g_2(loc^*(x - 1, \ell_1, \ell_2, t - 1)) \\ \textbf{endcase} \end{array}$$

As the equation systems are acyclic we know that $pd_4(y_1, y_2, \ell_1, \ell_2)$ is stable under $\overline{\delta}_1$, and $pl_4(x, \ell_1, \ell_2, t)$ is stable under $\overline{\delta}_2$. By proposition 39 it follows that:

$$(< \mu_1, \mu_2, \mu_3 > \in L^c(x, \ell_1, \ell_2, t)) \quad \text{iff} \quad lf^c(x, \ell_1, \ell_2, t, \mu_1, \mu_2, \mu_3) \qquad (6.169)$$

where $lf^c(x, \ell_1, \ell_2, t, \mu_1, \mu_2, \mu_3)$ is the lattice formula corresponding to $L^c(x, \ell_1, \ell_2, t)$, and

is defined by:

$$lf^c(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) = \left(\begin{array}{c} pd_4(\ell_1,\ell_2,\ell_1,\ell_2) \\ \wedge \\ pl_4(x,\ell_1,\ell_2,t) \\ \wedge \\ pd_4(\ell_1-\mu_2-\mu_3,\ell_2-\mu_1-\mu_3,\ell_1,\ell_2) \\ \wedge \\ pl_4(x-\mu_1+\mu_2,\ell_1,\ell_2,t-\mu_1-\mu_2-2\mu_3) \end{array}\right)$$

Similarly:

$$(<\mu_1,\mu_2,\mu_3>\in B^c(x,\ell_1,\ell_2,t)) \text{ iff } bf^c(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) \tag{6.170}$$

where $bf^c(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3)$ is the boundary formula corresponding to $B^c(x,\ell_1,\ell_2,t)$, and is defined by:

$$bf^c(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) =$$
$$\left(\begin{array}{c} \left(\begin{array}{c} \neg pd_4(\ell_1,\ell_2,\ell_1,\ell_2) \vee \neg pl_4(x,\ell_1,\ell_2,t) \\ \wedge \\ (\mu_1=0) \wedge (\mu_2=0) \wedge (\mu_3=0) \end{array}\right) \\ \vee \\ \left(\begin{array}{c} pd_4(\ell_1,\ell_2,\ell_1,\ell_2) \wedge pl_4(x,\ell_1,\ell_2,t) \\ \wedge \\ \left(\begin{array}{c} \neg pd_4(\ell_1-\mu_2-\mu_3,\ell_2-\mu_1-\mu_3,\ell_1,\ell_2) \\ \vee \\ \neg pl_4(x-\mu_1+\mu_2,\ell_1,\ell_2,t-\mu_1-\mu_2-2\mu_3) \end{array}\right) \\ \wedge \\ \left(\begin{array}{c} \left(\begin{array}{c} pd_4(y_1-\mu_2-\mu_3,y_2-\mu_1-\mu_3+1,\ell_1,\ell_2) \\ \wedge \\ pl_4(x-\mu_1+\mu_2+1,\ell_1,\ell_2,t-\mu_1-\mu_2-2\mu_3+1) \end{array}\right) \\ \vee \\ \left(\begin{array}{c} pd_4(y_1-\mu_2-\mu_3+1,y_2-\mu_1-\mu_3,\ell_1,\ell_2) \\ \wedge \\ pl_4(x-\mu_1+\mu_2-1,\ell_1,\ell_2,t-\mu_1-\mu_2-2\mu_3+1) \end{array}\right) \\ \vee \\ \left(\begin{array}{c} pd_4(y_1-\mu_2-\mu_3+1,y_2-\mu_1-\mu_3+1,\ell_1,\ell_2) \\ \wedge \\ pl_4(x-\mu_1+\mu_2,\ell_1,\ell_2,t-\mu_1-\mu_2-2\mu_3+2) \end{array}\right) \end{array}\right) \end{array}\right) \end{array}\right)$$

By applying lemma 49 we get Presburger formulas $P_{4,4,1}(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3)$, $P_{4,4,2}(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3)$, $P_{4,4,3}(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3)$, and $P_{4,4,4}(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3)$ such that:

$$P_{4,4,1}(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) \text{ iff } \left(\begin{array}{c} \neg pd_4(\ell_1-\mu_2-\mu_3,\ell_2-\mu_1-\mu_3,\ell_1,\ell_2) \\ \longrightarrow \\ \left(\begin{array}{c} d^*(\ell_1-\mu_2-\mu_3,\ell_2-\mu_1-\mu_3,\ell_1,\ell_2) \\ = \\ loc(x-\mu_1+\mu_2,\ell_1,\ell_2,t-\mu_1-\mu_2-2\mu_3) \end{array}\right) \end{array}\right) \tag{6.171}$$

and:

$$P_{4,4,2}(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) \text{ iff } \left(\begin{array}{c} \neg pl_4(x-\mu_1+\mu_2,\ell_1,\ell_2,t-\mu_1-\mu_2-2\mu_3) \\ \longrightarrow \\ \left(\begin{array}{c} d(\ell_1-\mu_2-\mu_3,\ell_2-\mu_1-\mu_3,\ell_1,\ell_2) \\ = \\ loc^*(x-\mu_1+\mu_2,\ell_1,\ell_2,t-\mu_1-\mu_2-2\mu_3) \end{array} \right) \end{array} \right) \quad (6.172)$$

and:

$$P_{4,4,3}(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) \text{ iff } \left(\begin{array}{c} a(\ell_1-\mu_2-\mu_3) \\ = \\ out_1(x-\mu_1+\mu_2-1,\ell_1,\ell_2,t-\mu_1-\mu_2-2\mu_3-1) \end{array} \right) \quad (6.173)$$

and:

$$P_{4,4,4}(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) \text{ iff } \left(\begin{array}{c} b(\ell_2-\mu_1-\mu_3) \\ = \\ out_2(x-\mu_1+\mu_2+1,\ell_1,\ell_2,t-\mu_1-\mu_2-2\mu_3-1) \end{array} \right) \quad (6.174)$$

From (6.166), (6.167), (6.168), (6.169), (6.170), (6.171), (6.172), (6.173), (6.174) we get:

$$P_{4,4}(x,\ell_1,\ell_2,t)$$
$$=$$
$$\left(\begin{array}{ll} \forall < \mu_1,\mu_2,\mu_3 >. bf^c(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) & \longrightarrow P_{4,4,1}(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) \\ \forall < \mu_1,\mu_2,\mu_3 >. bf^c(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) & \longrightarrow P_{4,4,2}(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) \\ \forall < \mu_1,\mu_2,\mu_3 >. lf^c(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) & \longrightarrow P_{4,4,3}(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) \\ \forall < \mu_1,\mu_2,\mu_3 >. lf^c(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) & \longrightarrow P_{4,4,4}(x,\ell_1,\ell_2,t,\mu_1,\mu_2,\mu_3) \end{array} \right)$$

which is a Presburger formula.

Chapter 7

Class of Boolean Algebras

In this chapter we consider verification of systolic circuits which operate on boolean algebras. We will construct a decision method for automatic verification of a class of such circuits. The cell computations in the circuits of this class are defined by the boolean operators $\vee$, $\wedge$, $\neg$, ff, and tt. The methods of this chapter can be applied to circuits which are modeled on the "bit level", where the data on which the cells compute are "1"s and "0"s. We will also consider a class of circuits where the cells operate on two disjoint sets of function symbols. The elements of the first set are interpreted as the operators of a boolean algebra, while the elements of the second set are interpreted in some arbitrary algebra A. This is used to model *"data-dependent"* circuits. In a *data-dependent* circuit the computations inside a cell do not depend only on the time instant and the cell position, but also on the data input to the cell. This data dependency is described by a set of relations over the algebra A. An example of a data-dependent circuit is the substring detecting circuit of section 3.3, where at each clock cycle the inputs of a cell are compared for equality. The results sent out through the outputs of the cell in the next clock cycle are dependent on the result of comparison.

Examples of circuits on which the methods of this chapter can be applied are the substring detecting circuit in section 3.3, the boolean matrix multiplication circuit in [Ull84], the graph transitive closure circuit in [Ull84], the palindrome recognizer circuits in [Hen86] and [LS81], and circuits for realization of bit addition, multiplication, convolution, etc.

We will illustrate the ideas of the chapter by sketching how the verification method can be applied to the substring detecting circuit introduced in section 3.3.

In section 7.1 we give some preliminaries on boolean algebras. In section 7.2 we define a class of systolic circuits which we call *tail-recursive boolean systolic circuits*. In section 7.3 we study what the general definition of semantics amounts to when restricting ourselves to the class of boolean algebras. In section 7.4 we give an overview of a decision method for automatic verification of a subclass of tail-recursive boolean circuits. In section 7.5 we show that the values of signals in the implementation and specification of a tail-recursive boolean systolic circuit can be described as a class of guarded expressions over boolean algebras. In section 7.6 we show, for a class of guarded boolean expressions, that deciding equality can be reduced to Presburger's arithmetic which is decidable [End72], and hence the verification problem is decidable.

7.1 Preliminaries

We will use the same notation as in the previous chapters. Linear QI-polynomials, linear inequalities, linear equalities, linear modulo predicates, and linear predicates have the same meanings as defined in section 5.1. We will work with an arbitrary boolean algebra $B =< B, \vee, \wedge, \neg, tt, ff >$, and with a signature $\mathcal{SIG}$, where $\mathcal{SIG} =< \mathcal{S} \cup \{B\}, \mathcal{G} >$, and $B \notin \mathcal{S}$. We call each element of $\mathcal{G}$ an *uninterpreted operation symbol.* We assume that we have a set $\mathcal{A}$ of stream variables, where each stream variable has a sort $S \in \mathcal{S}$.

The class of *uninterpreted terms* over $\mathcal{SIG}$ and their *sorts* are defined in the following way:

- Each linear stream expression $a(\overline{q}(\overline{x}))$ is an uninterpreted term over $\mathcal{SIG}$ with the same sort as the sort of a.
- If $t_1(\overline{x}), \ldots, t_n(\overline{x})$ are uninterpreted terms over $\mathcal{SIG}$ of sorts $S_1, \ldots, S_n$ respectively, and $g \in \mathcal{G}$ is an uninterpreted operation symbol with a domain sort $S_1 \cdot \cdots \cdot S_n$ and a range sort $S \in \mathcal{S}$, then $g(t_1(\overline{x}), \ldots, t_n(\overline{x}))$ is an uninterpreted term over $\mathcal{SIG}$ of sort S.

An *atomic boolean expression* over $\mathcal{SIG}$ is of the form $g(t_1(\overline{x}), \ldots, t_n(\overline{x}))$, where $t_1(\overline{x})$, $\ldots, t_n(\overline{x})$ are uninterpreted terms over $\mathcal{SIG}$ of sorts $S_1, \ldots, S_n$ respectively, and $g \in \mathcal{G}$ has a domain sort $S_1 \cdot \cdots \cdot S_n$ and a range sort B.

The class of *boolean expressions* over $\mathcal{SIG}$ is defined by:

- tt and ff are boolean expressions over $\mathcal{SIG}$.
- Each atomic boolean expression over $\mathcal{SIG}$ is a boolean expression over $\mathcal{SIG}$.
- If $b_1(\overline{x})$ and $b_2(\overline{x})$ are boolean expressions over $\mathcal{SIG}$, then $t_1(\overline{x}) \vee t_2(\overline{x})$, $t_1(\overline{x}) \wedge t_2(\overline{x})$, and $\neg t_1(\overline{x})$ are boolean expressions over $\mathcal{SIG}$.
- If $b(\overline{x}, i)$ is a boolean expression over $\mathcal{SIG}$, and $\ell(\overline{x})$ is a linear QI-polynomial then $\bigvee_{i=0}^{\ell(\overline{x})} b(\overline{x}, i)$ and $\bigwedge_{i=0}^{\ell(\overline{x})} b(\overline{x}, i)$ are boolean expressions over $\mathcal{SIG}$.

We will denote boolean expressions by b.

A *linear guarded boolean expression* over $\mathcal{SIG}$ is of the form:

$$\begin{array}{l} \textbf{case} \\ \quad \begin{array}{lcl} p_1(\overline{x}) & \Longrightarrow & b_1(\overline{x}) \\ & \vdots & \\ p_n(\overline{x}) & \Longrightarrow & b_n(\overline{x}) \end{array} \\ \textbf{endcase} \end{array}$$

where $b_i(\overline{x})$ is a boolean expression over $\mathcal{SIG}$, and $p_i(\overline{x})$ is a conjunction of linear predicates. In addition, we assume that $p_i(\overline{\alpha}) \wedge p_j(\overline{\alpha})$ is false for each $\overline{\alpha}$ if $j \neq i$, and that $b_i(\overline{x})$ is well-defined under $p_i(\overline{x})$ (well-definedness is defined in a similar way to that in section 5.1). Each $p_i(\overline{x})$ is called a *guard* of the expression, and each $b_i(\overline{x})$ is called a *result* of the expression, while every $p_i(\overline{x}) \Rightarrow b_i(\overline{x})$ is called a *case* of the expression. It can easily be shown that each linear guarded boolean expression can be defined by a system of recurrence equations over B and $\mathcal{SIG}$. We use $e, e_1, e_2, e_3, \ldots$ to range over linear guarded boolean expressions.

7.2 Tail-recursive Boolean Circuits

A *tail-recursive boolean systolic circuit* is a linear systolic circuit, where certain restrictions are imposed on the cell computations and the specification formula of the circuit.

With each tail-recursive boolean circuit is associated a signature $\mathcal{SIG} =< \mathcal{S} \cup \{B\}, \mathcal{G} >$, where $B \notin \mathcal{S}$, called the *signature* of the circuit. The circuit operates on a boolean algebra B and and a $\mathcal{SIG}$-algebra A

The set of signals in a tail-recursive boolean circuit can be divided into two disjoint sets. One set of signals carry values from the sorts of $\mathcal{S}$ and are called the *uninterpreted signals* of the circuit. The other set of signals carry values from the boolean algebra B and are called the *boolean signals*. The uninterpreted signals of the substring detecting circuit of section 3.3 are in_1, out_1, and loc_1, while the boolean signals are in_2, out_2, in_3, out_3, and loc_2.

Cell Computations in Tail-recursive Boolean Systolic Circuits Each uninterpreted signal s has a sort $S \in \mathcal{S}$. The set of *uninterpreted cell computation terms* and their *sorts* are defined by:

- If s is an uninterpreted input or a local variable of sort S, then $s(\overline{x}, \overline{\ell}, t - \tau)$, where τ is a positive integer, is an uninterpreted cell computation term with sort S.
- If $ct_1(\overline{x}, \overline{\ell}, t), \ldots, ct_n(\overline{x}, \overline{\ell}, t)$ are uninterpreted cell computation terms with sorts S_1, $\ldots, S_n$ respectively, and if $g \in \mathcal{G}$ is an uninterpreted operation symbol with a domain sort $S_1 \cdot \cdots \cdot S_n$ and a range sort $S \in \mathcal{S}$, then $g(ct_1(\overline{x}, \overline{\ell}, t), \ldots, ct_n(\overline{x}, \overline{\ell}, t))$ is an uninterpreted cell computation term with a sort S.

An *atomic boolean cell computation term* is of the form $g(ct_1(\overline{x}), \ldots, ct_n(\overline{x}))$, where $ct_1(\overline{x})$, $\ldots, ct_n(\overline{x})$ are uninterpreted cell computation terms of sorts $S_1, \ldots, S_n$ respectively, and $g \in \mathcal{G}$ is an uninterpreted operation symbol with a domain sort $S_1 \cdot \cdots \cdot S_n$ and a range sort B.

The class of *boolean cell computation terms* is defined by the following:

- Each atomic boolean cell computation term is a boolean cell computation term.
- If $b_1(\overline{x}, \overline{\ell}, t)$ and $b_2(\overline{x}, \overline{\ell}, t)$ are boolean cell computation terms, then $b_1(\overline{x}, \overline{\ell}, t) \vee b_2(\overline{x}, \overline{\ell}, t)$, $b_1(\overline{x}, \overline{\ell}, t) \wedge b_2(\overline{x}, \overline{\ell}, t)$, and $\neg b_1(\overline{x}, \overline{\ell}, t)$ are boolean cell computation terms.

The *cell computation* associated with a signal s of a tail-recursive boolean circuit is of one of the following two forms:

- If s is an uninterpreted signal then the cell computation associated with s will be of the form:

$$s(\overline{x}, \overline{\ell}, t) \;=\; s'(\overline{x}, \overline{\ell}, t - \tau)$$

  where τ is a positive integer, s' is the input corresponding to s if s is an output, while s' is the same as s if s is a local variable

- If s is a boolean signal then the cell computation associated with s is of the form:

$$
\begin{array}{l}
s(\overline{x},\overline{\ell},t) = \\
\textbf{case} \\
\quad \begin{array}{rcl}
p_1(\overline{x},\overline{\ell},t) & \Longrightarrow & b_1(\overline{x},\overline{\ell},t) \\
\vdots & & \\
p_m(\overline{x},\overline{\ell},t) & \Longrightarrow & b_m(\overline{x},\overline{\ell},t) \\
p_{m+1}(\overline{x},\overline{\ell},t) & \Longrightarrow & s'(\overline{x},\overline{\ell},t-\tau) \,\square\, b_{m+1}(\overline{x},\overline{\ell},t) \\
\vdots & & \\
p_n(\overline{x},\overline{\ell},t) & \Longrightarrow & s'(\overline{x},\overline{\ell},t-\tau) \,\square\, b_n(\overline{x},\overline{\ell},t)
\end{array} \\
\textbf{endcase}
\end{array}
\tag{7.1}
$$

where $\square$ is $\vee$ or $\wedge$, and $b_i(\overline{x},\overline{\ell},t)$ is a boolean cell computation term. The *dependency relation* $\prec_D$ is defined among the boolean signals of a tail-recursive boolean circuit in a similar way to that in tail-recursive ring circuits (see section 5.2.2). In the class of tail-recursive boolean circuits, the dependency relation is acyclic, i.e. for each signal s, we have $s \not\prec_D^+ s$, where $\prec_D^+$ is the transitive closure of $\prec_D$. The dependency relation of the substring detecting circuit in section 3.3 is given by the set $\{< out_3, in_2 >, < out_3, out_2 >, < in_3, in_2 >, < in_3, out_2 >, < loc_2, in_2 >, < loc_2, out_2 >, < loc_2, in_3 >, < loc_2, out_3 >\}$. Observe that the relation is acyclic.

Specification Formulas of Tail-recursive Boolean Systolic Circuits The specification formula of a tail-recursive boolean circuit is of the form:

$$
\begin{array}{l}
spec(\overline{x},\overline{\ell},t) = \\
(p_1(\overline{x},\overline{\ell},t) \longrightarrow (s_1(\overline{x},\overline{\ell},t) = b_1(\overline{x},\overline{\ell},t))) \wedge \cdots \wedge (p_m(\overline{x},\overline{\ell},t) \longrightarrow (s_m(\overline{x},\overline{\ell},t) = b_m(\overline{x},\overline{\ell},t)))
\end{array}
$$

where s_i is a boolean signal in the circuit, and $b_i(\overline{x},\overline{\ell},t)$ is a boolean expression which is well-defined under $p_i(\overline{x},\overline{\ell},t)$.

7.3 Semantics

Let $\mathcal{SIG} =< \mathcal{S} \cup \{B\}, \mathcal{G} >$, where $B \notin \mathcal{S}$ be a signature. Let B be any boolean algebra, and let A be any $\mathcal{SIG}$-algebra such that[1] $B^A = B$. A stream interpretation $\mathcal{I}$ over A maps each stream variable into a function from tuples of integers into the domains of A. Thus if $\mathcal{I}$ is a stream interpretation then, for each $a \in \mathcal{A}$ of arity n and sort S, $\mathcal{I}(a)$ is a function of type $I^n \rightarrow \mathcal{S}^A$.

Let $\perp$ be an element such that $\perp \notin \mathcal{S} \cup \{B\}$. By $[\![t(\overline{x})]\!]_{A,\mathcal{I}}$ we mean the *interpretation* of the uninterpreted term $t(\overline{x})$ under A and $\mathcal{I}$. Let a be a stream variable, and

[1]By $B^A = B$ we mean that A interprets the sort B in the signature $\mathcal{SIG}$ as the domain of the boolean algebra B. Throughout this chapter we assume that for each boolean algebra B, signature $\mathcal{SIG} =< \mathcal{S} \cup \{B\}, \mathcal{G} >$, and $\mathcal{SIG}$-algebra A, $B^A = B$.

$g(t_1(\overline{x}), \ldots, t_n(\overline{x}))$ an uninterpreted term, then:

$$[\![a(\overline{q}(\overline{x}))]\!]_{A,\mathcal{I}} = \begin{cases} a^{\mathcal{I}}(\overline{q}(\overline{x})) & \text{if } \overline{q}(\overline{x}) \text{ is a tuple of integers} \\ \bot & \text{otherwise} \end{cases}$$

$$[\![g(t_1(\overline{x}), \ldots, t_n(\overline{x}))]\!]_{A,\mathcal{I}} = \begin{cases} g^A([\![t_1(\overline{x})]\!]_{A,\mathcal{I}}, \ldots, [\![t_n(\overline{x})]\!]_{A,\mathcal{I}}) & \text{if } [\![t_i(\overline{x})]\!]_{A,\mathcal{I}} \neq \bot \text{ for } 1 \leq i \leq n \\ \bot & \text{otherwise} \end{cases}$$

By $[\![b(\overline{x})]\!]_{B,A,\mathcal{I}}$ we mean the *interpretation* of the boolean expression $b(\overline{x})$ under B, A and $\mathcal{I}$. Let $g(t_1(\overline{x}), \ldots, t_n(\overline{x}))$ be an atomic boolean expression, and $b(\overline{x}), b_1(\overline{x}), b_2(\overline{x})$ any boolean expressions, then:

$$[\![ff]\!]_{B,A,\mathcal{I}} = ff$$

$$[\![tt]\!]_{B,A,\mathcal{I}} = tt$$

$$[\![g(t_1(\overline{x}),\ldots,t_n(\overline{x}))]\!]_{B,A,\mathcal{I}} = \begin{cases} g^A([\![t_1(\overline{x})]\!]_{A,\mathcal{I}},\ldots,[\![t_n(\overline{x})]\!]_{A,\mathcal{I}}) & \text{if } [\![t_i(\overline{x})]\!]_{A,\mathcal{I}} \neq \perp \\ & \text{for } 1 \leq i \leq n \\ \perp & \text{otherwise} \end{cases}$$

$$[\![b_1(\overline{x}) \vee b_2(\overline{x})]\!]_{B,A,\mathcal{I}} = \begin{cases} [\![b_1(\overline{x})]\!]_{B,A,\mathcal{I}} \vee [\![b_2(\overline{x})]\!]_{B,A,\mathcal{I}} & \text{if } [\![b_i(\overline{x})]\!]_{B,A,\mathcal{I}} \neq \perp \\ & \text{for } i = 1,2 \\ \perp & \text{otherwise} \end{cases}$$

$$[\![\neg b(\overline{x})]\!]_{B,A,\mathcal{I}} = \begin{cases} \neg\, [\![b(\overline{x})]\!]_{B,A,\mathcal{I}} & \text{if } [\![b(\overline{x})]\!]_{B,A,\mathcal{I}} \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

$$[\![b_1(\overline{x}) \wedge b_2(\overline{x})]\!]_{B,A,\mathcal{I}} = \begin{cases} [\![b_1(\overline{x})]\!]_{B,A,\mathcal{I}} \wedge [\![b_2(\overline{x})]\!]_{B,A,\mathcal{I}} & \text{if } [\![b_i(\overline{x})]\!]_{B,A,\mathcal{I}} \neq \perp \\ & \text{for } i = 1,2 \\ \perp & \text{otherwise} \end{cases}$$

$$\left[\!\!\left[\bigvee_{i=0}^{\ell(\overline{x})} b(\overline{x},i)\right]\!\!\right]_{B,A,\mathcal{I}} = \begin{cases} \bigvee_{i=0}^{\ell(\overline{x})} [\![b(\overline{x},i)]\!]_{B,A,\mathcal{I}} & \text{if } \ell(\overline{x}) \text{ is a nonnegative integer} \\ \perp & \text{otherwise} \end{cases}$$

$$\left[\!\!\left[\bigwedge_{i=0}^{\ell(\overline{x})} b(\overline{x},i)\right]\!\!\right]_{B,A,\mathcal{I}} = \begin{cases} \bigwedge_{i=0}^{\ell(\overline{x})} [\![b(\overline{x},i)]\!]_{B,A,\mathcal{I}} & \text{if } \ell(\overline{x}) \text{ is a nonnegative integer} \\ \perp & \text{otherwise} \end{cases}$$

$$\left[\!\!\left[\begin{array}{l} \textbf{case} \\ \quad p_1(\overline{x}) \Rightarrow b_1(\overline{x}); \\ \quad\quad \vdots \\ \quad p_m(\overline{x}) \Rightarrow b_m(\overline{x}) \\ \textbf{endcase} \end{array}\right]\!\!\right]_{B,A,\mathcal{I}} = \begin{cases} [\![b_i(\overline{x})]\!]_{B,A,\mathcal{I}} & \text{if } p_i(\overline{x}) \text{ is true} \\ \perp & \text{if } p_i(\overline{x}) \text{ is false for } 1 \leq i \leq m \end{cases}$$

The well-definedness and equality properties for boolean expressions are defined in a similar way to that for ring expressions (section 5.3).

7.4 Overview of the Verification Decision Method

We will give a sketch of an automatic decision method for a subclass of tail-recursive boolean circuits. In section 7.2 we mentioned that the specification of a circuit with a signature $\mathcal{SIG}$ was of the form:

$$spec(\overline{x}) = \tag{7.2}$$

$$(p_1(\overline{x}) \longrightarrow (s_1(\overline{x}) = b_1(\overline{x}))) \wedge \cdots \wedge (p_m(\overline{x}) \longrightarrow (s_m(\overline{x}) = b_m(\overline{x})))$$

where $p_1(\overline{x}), \ldots, p_m(\overline{x})$ are conjunctions of linear predicates, $s_1, \ldots, s_m$ are boolean signals in the circuit, and $b_1(\overline{x}), \ldots, b_m(\overline{x})$ are boolean expressions over $\mathcal{SIG}$ such that $b_i(\overline{x})$ is well-defined under $p_i(\overline{x})$.

In section 7.3 we defined formally the notion of equality for linear guarded boolean expressions with a signature $\mathcal{SIG}$ over any boolean algebra B and $\mathcal{SIG}$-algebra A.

Let $\mathcal{B}$ be a class of boolean algebras and $\mathcal{K}$ a class of $\mathcal{SIG}$-algebras. By the *circuit verification* over $\mathcal{B}$ and $\mathcal{K}$, we mean that the specification formula in (7.2) is valid when interpreted in every boolean algebra $B \in \mathcal{B}$ and $\mathcal{SIG}$-algebra $A \in \mathcal{K}$; in symbols:

$$\mathcal{B}, \mathcal{K} \models spec(\overline{x})$$

Notice that $spec(\overline{x})$ contains stream variables (which occur in $b_1, \ldots, b_m$) and integer variables ($\overline{x}$, $\overline{\ell}$, and t). Thus to interpret $spec(\overline{x})$, the stream variables are interpreted in the algebras of $\mathcal{K}$, while the integer variables are interpreted in the standard model of integers.

The verification process is carried out in the following two steps, each of which is carried out automatically:

1. The first verification step is similar to the first step of verification of tail-recursive ring circuits (described in section 5.5). In section 7.5 we show that, for each tail-recursive boolean systolic circuit, the value of each boolean signal in the circuit can be described by a class of functions which we will introduce in section 7.5.1, and which we call *tail-recursive boolean functions*. Furthermore we will show that for each tail-recursive boolean function over a signature $\mathcal{SIG}$, there is a linear guarded boolean expression over $\mathcal{SIG}$, which is equal to it over each boolean algebra B and $\mathcal{SIG}$-algebra A. This means that, considering the specification formula in (7.2), there are linear guarded boolean expressions $e_1(\overline{x}), \ldots, e_m(\overline{x})$ such that:

$$s_1(\overline{x}) = e_1(\overline{x}) \quad , \quad \ldots \quad , \quad s_m(\overline{x}) = e_m(\overline{x})$$

over each boolean algebra B and $\mathcal{SIG}$-algebra A, so that the specification formula can be rewritten as:

$$spec(\overline{x}) = \\ (p_1(\overline{x}) \longrightarrow (e_1(\overline{x}) = b_1(\overline{x}))) \wedge \cdots \wedge (p_m(\overline{x}) \longrightarrow (e_m(\overline{x}) = b_m(\overline{x}))) \tag{7.3}$$

which is equivalent over $\mathcal{B}$ and $\mathcal{K}$.

2. In section 7.6 we consider decidability of validity of formulas of the general form:

$$p(\overline{x}) \longrightarrow (e(\overline{x}) = b(\overline{x})) \tag{7.4}$$

where $p(\overline{x})$ is a conjunction of linear predicates, $e(\overline{x})$ is a linear guarded boolean expression over a signature $\mathcal{SIG}$, and $b(\overline{x})$ is a boolean expression over $\mathcal{SIG}$ which is well-defined under $p(\overline{x})$. From the formula in (7.3) we conclude that if the validity, over $\mathcal{B}$ and $\mathcal{K}$, of the formula in (7.4) is decidable, then the validity of the specification formula is decidable.

We know that $e(\overline{x})$ in (7.4) is of the form:

$$\textbf{case} \quad p'_1(\overline{x}) \Rightarrow b'_1(\overline{x}) \; ; \; \ldots \; ; \; p'_n(\overline{x}) \Rightarrow b'_n(\overline{x}) \quad \textbf{endcase}$$

Thus, for $\mathcal{B}$ and $\mathcal{K}$, the validity of the formula in (7.4) is equivalent to the validity of the formula:

$$\begin{array}{c} p(\overline{x}) \longrightarrow (p'_1(\overline{x}) \vee \cdots \vee p'_n(\overline{x})) \\ \wedge \\ (p(\overline{x}) \wedge p'_1(\overline{x}) \longrightarrow (b(\overline{x}) = b'_1(\overline{x}))) \wedge \cdots \wedge (p(\overline{x}) \wedge p'_n(\overline{x}) \longrightarrow (b(\overline{x}) = b'_n(\overline{x}))) \end{array} \tag{7.5}$$

We will consider the special case where $\mathcal{B}$ is the two-valued boolean algebra B_2 (introduced in appendix B), and $\mathcal{K}$ is the algebraic specification $\mathcal{SPEC}$ of the class of all $\mathcal{SIG}$-algebras, where $\mathcal{SIG}$ is the signature of the boolean expressions in (7.5). We will define a subclass of boolean expressions (by imposing certain restrictions on the existence of the $\neg$ operator and the number of the higher order operators $\bigvee$ and $\bigwedge$ which occur in a boolean expression) and show that deciding equality of any pair of boolean expressions in the subclass over B_2 and $\mathcal{SPEC}$ is equivalent to the validity of a Presburger formula. Regarding equation (7.5), this implies that there are Presburger formulas $P_1(\overline{x}), \ldots, P_n(\overline{x})$ such that:

$$P_i(\overline{x}) \quad \text{iff} \quad (b(\overline{x}) = b'_i(\overline{x}))$$

over B_2 and $\mathcal{SPEC}$. It follows that the formula in (7.5) can be rewritten into:

$$\begin{array}{c} p(\overline{x}) \longrightarrow (p'_1(\overline{x}) \vee \cdots \vee p'_n(\overline{x})) \\ \wedge \\ (p(\overline{x}) \wedge p'_1(\overline{x}) \longrightarrow P_1(\overline{x})) \wedge \cdots \wedge (p(\overline{x}) \wedge p'_n(\overline{x}) \longrightarrow P_n(\overline{x})) \end{array} \tag{7.6}$$

which is equivalent over B_2 and $\mathcal{SPEC}$. We notice that the formula in (7.6) is a Presburger formula. The decidability of the problem follows from the decidability of Presburger formulas (see section 6.6.1).

7.5 Boolean Circuits as Guarded Boolean Expressions

In this section we carry out the first step of the automatic verification of tail-recursive boolean systolic circuits. We show that, for a tail-recursive boolean circuit, the values of outputs, inputs, and local variables of the different cells at different time instants can be described as linear guarded boolean expressions. The algorithm is similar to that given in section 5.5, and is performed in two steps. In the first step we define the class of *tail-recursive boolean functions*, and show that for each tail-recursive boolean function over a signature $\mathcal{SIG}$, there is a linear guarded boolean expression over $\mathcal{SIG}$, which is equal to it over each boolean algebra B and $\mathcal{SIG}$-algebra A. In the second step we show that the values of cell outputs, inputs, and local variables can be described as tail-recursive boolean functions, and hence as linear guarded boolean expressions.

7.5.1 Tail-Recursive Boolean Functions

A *tail-recursive boolean function* f over a signature $\mathcal{SIG}$ and a tuple $\overline{\delta}$ of integers, where at least one element δ_i of $\overline{\delta}$ is not equal to 0, is of the following form:

$$
\begin{array}{lll}
f(\overline{x}) & = & \textbf{case} \\
 & & \quad p_1(\overline{x}) \Rightarrow b_1(\overline{x}) \; ; \; \ldots \; ; \; p_m(\overline{x}) \Rightarrow b_m(\overline{x}) \; ; \\
 & & \quad p_{m+1}(\overline{x}) \Rightarrow f(\overline{x}+\overline{\delta}) \; \square \; b_{m+1}(\overline{x}) \; ; \; \ldots \; ; \; p_n(\overline{x}) \Rightarrow f(\overline{x}+\overline{\delta}) \; \square \; b_n(\overline{x}) \\
 & & \textbf{endcase}
\end{array}
\tag{7.7}
$$

where $b_i(\overline{x})$ is a boolean expression over $\mathcal{SIG}$, $p_i(\overline{x})$ is a conjunction of linear predicates, and each $\square$ is a $\vee$ or a $\wedge$. In addition, $b_i(\overline{x})$ is well-defined under $p_i(\overline{x})$, and $p_i(\overline{\alpha}) \wedge p_j(\overline{\alpha})$ is false for each $\overline{\alpha}$ if $j \neq i$. The semantics of tail-recursive boolean functions can be defined in a similar manner to that of tail-recursive ring functions (see section 5.5.1).

Theorem 52 *For each tail-recursive boolean function f over a signature $\mathcal{SIG}$, there is a linear guarded boolean expression e over $\mathcal{SIG}$, such that $f(\overline{x}) = e(\overline{x})$ over each boolean algebra B and $\mathcal{SIG}$-algebra A.*

Proof: The proof is similar to that of theorem 15. □

An *uninterpreted linear guarded expression* over a signature $\mathcal{SIG}$ is of the form:

$$
\begin{array}{lcl}
\textbf{case} & & \\
\quad p_1(\overline{x}) & \Longrightarrow & t_1(\overline{x}) \\
 & \vdots & \\
\quad p_n(\overline{x}) & \Longrightarrow & t_n(\overline{x}) \\
\textbf{endcase} & &
\end{array}
$$

where each $t_i(\overline{x})$ is an uninterpreted term over $\mathcal{SIG}$.

Theorem 53 *Consider a tail-recursive boolean systolic circuit with a signature $\mathcal{SIG}$. Let s be any boolean signal in the circuit. Then there is a linear guarded boolean expression e over $\mathcal{SIG}$ such that:*

$$s(\overline{x}, \overline{\ell}, t) \;=\; e(\overline{x}, \overline{\ell}, t)$$

over each boolean algebra B and $\mathcal{SIG}$-algebra A.

Proof: It can easily be shown that for each uninterpreted signal s in the circuit, there is an uninterpreted linear guarded expression e such that:

$$s(\overline{x}, \overline{\ell}, t) \;=\; e(\overline{x}, \overline{\ell}, t)$$

over each $\mathcal{SIG}$-algebra A.

Given a boolean signal s in the circuit, we can use theorem 52 and a similar method to that of theorem 16 to show that there is a linear guarded boolean expression e over $\mathcal{SIG}$ which fulfills the claim of the lemma. □

7.5.2 First Step of Verification of The Substring Detecting Circuit: Describing it by Linear Guarded Boolean Expressions

Theorem 53 can be applied to the equations describing the implementation of the substring detecting circuit in section 3.3.1 to express the values of the signals of the circuit as linear guarded boolean expressions. This can be done in a similar manner to which theorem 16 is applied in section 5.5.3 to the equations describing the implementation of the convolution circuit in section 3.1.1 to describe the signals of the circuit as linear guarded ring expressions.

The value of loc_2 can be expressed as:

$loc_2(x, \ell_1, \ell_2, t) =$

case

$$\left\{\begin{array}{c} 1 \le x \le \ell_1 \\ 0 < t \\ (t+x)\ mod\ 2 = 0 \\ x \le t \\ t \le 2\ell_2 + x - 2 \\ t + x = 2\ell_1 \end{array}\right\} \Longrightarrow \bigvee_{j=0}^{x-1} \bigwedge_{i=0}^{\frac{t-x}{2}} \left(\begin{array}{c} a\left(\frac{t-2i+2j-x+2}{2}\right) \\ = \\ b(\ell_2 - i) \end{array}\right)$$

$$\left\{\begin{array}{c} 1 \le x \le \ell_1 \\ 0 < t \\ (t+x)\ mod\ 2 = 0 \\ x \le t \\ t \le 2\ell_2 + x - 2 \\ x = 1 \\ t + x > 2\ell_1 \end{array}\right\} \Longrightarrow \bigvee_{j=0}^{\frac{t-2\ell_1+1}{2}} \bigwedge_{i=0}^{\ell_1-1} \left(\begin{array}{c} a(\ell_1 - i) \\ = \\ b\left(\frac{2\ell_1+2\ell_2-t-2i+2j-1}{2}\right) \end{array}\right) \tag{7.8}$$

$$\left\{\begin{array}{c} 1 \le x \le \ell_1 \\ 0 < t \\ (t+x+1)\ mod\ 2 = 0 \\ x \le t \\ t \le 2\ell_2 + x - 1 \\ x = 1 \\ t + x > 2\ell_1 \end{array}\right\} \Longrightarrow \bigvee_{j=0}^{\frac{t-2\ell_1}{2}} \bigwedge_{i=0}^{\ell_1-1} \left(\begin{array}{c} a(\ell_1 - i) \\ = \\ b\left(\frac{2\ell_1+2\ell_2-t-2i+2j}{2}\right) \end{array}\right)$$

endcase

7.6 Equality Checking

In this section we will carry out the second step of the verification of tail-recursive boolean circuits. We will study validity of formulas of the general form:

$$p(\overline{x}) \quad \longrightarrow \quad (e(\overline{x}) = b(\overline{x})) \tag{7.9}$$

where $p(\overline{x})$ is a conjunction of linear predicates, $e(\overline{x})$ is a linear guarded boolean expression, and $b(\overline{x})$ is a boolean expression which is well-defined under $p(\overline{x})$.

We need the notion of normal forms for boolean expressions.

7.6.1 Normal Forms for Boolean Expressions

The class of *first order boolean expressions* is the class of expressions including the class of atomic boolean expressions and closed under $\vee$, $\wedge$, and $\neg$. This means that a first order boolean expression is a boolean expression which does not contain the higher order operators $\bigvee$ and $\bigwedge$.

A first order boolean expression is said to be in *disjunctive normal form* if it is of the form:

$$(b_{11}(\overline{x}) \wedge \cdots \wedge b_{1m_1}(\overline{x})) \vee \cdots \vee (b_{n1}(\overline{x}) \wedge \cdots \wedge b_{nm_n}(\overline{x}))$$

where each $b_{ij}(\overline{x})$ is either an atomic boolean expression, or the negation of an atomic boolean expression.

A first order boolean expression is said to be in *conjunctive normal form* if it is of the form:

$$(b_{11}(\overline{x}) \vee \cdots \vee b_{1m_1}(\overline{x})) \wedge \cdots \wedge (b_{n1}(\overline{x}) \vee \cdots \vee b_{nm_n}(\overline{x}))$$

where each $b_{ij}(\overline{x})$ is either an atomic boolean expression, or the negation of an atomic boolean expression.

A first order boolean expression is said to be in *normal form* if it is in disjunctive or conjunctive normal form.

A *normal boolean expression* is of the form:

$$\prod_{i_1=0}^{\ell_1(\overline{x})} \prod_{i_2=0}^{\ell_2(\overline{x},i_1)} \cdots \prod_{i_m=0}^{\ell_m(\overline{x},i_1,\ldots,i_{m-1})} b(\overline{x}, i_1, \ldots, i_m)$$

where each $\prod$ is a $\bigvee$ or a $\bigwedge$, and $b(\overline{x}, i_1, \ldots, i_m)$ is a normal first order boolean expression.

Lemma 54 *For each boolean expression $b(\overline{x})$, over a signature $\mathcal{SIG}$, there is a normal boolean expression $N(\overline{x})$ over $\mathcal{SIG}$, such that $b(\overline{x}) = N(\overline{x})$ over each boolean algebra B and $\mathcal{SIG}$-algebra A.*

Proof: The proof can be carried out easily by structural induction on boolean expressions, and rewriting by distributive laws as:

$$\begin{aligned}
b_1(\overline{x}) \wedge (b_2(\overline{x}) \vee b_3(\overline{x})) &= (b_1(\overline{x}) \wedge b_2(\overline{x})) \vee (b_1(\overline{x}) \wedge b_3(\overline{x})) \\
b_1(\overline{x}) \vee (b_2(\overline{x}) \wedge b_3(\overline{x})) &= (b_1(\overline{x}) \vee b_2(\overline{x})) \wedge (b_1(\overline{x}) \vee b_3(\overline{x})) \\
\neg(b_1(\overline{x}) \wedge b_2(\overline{x})) &= \neg b_1(\overline{x}) \vee \neg b_2(\overline{x}) \\
\neg(b_1(\overline{x}) \vee b_2(\overline{x})) &= \neg b_1(\overline{x}) \wedge \neg b_2(\overline{x}) \\
\left(\bigwedge_{i=0}^{\ell(\overline{x})} b_1(\overline{x}, i)\right) \wedge b_2(\overline{x}) &= \bigwedge_{i=0}^{\ell(\overline{x})} b_1(\overline{x}, i) \wedge b_2(\overline{x}) \\
\left(\bigvee_{i=0}^{\ell(\overline{x})} b_1(\overline{x}, i)\right) \vee b_2(\overline{x}) &= \bigvee_{i=0}^{\ell(\overline{x})} b_1(\overline{x}, i) \vee b_2(\overline{x}) \\
\left(\bigwedge_{i=0}^{\ell(\overline{x})} b_1(\overline{x}, i)\right) \vee b_2(\overline{x}) &= \bigwedge_{i=0}^{\ell(\overline{x})} b_1(\overline{x}, i) \vee b_2(\overline{x}) \\
\left(\bigvee_{i=0}^{\ell(\overline{x})} b_1(\overline{x}, i)\right) \wedge b_2(\overline{x}) &= \bigvee_{i=0}^{\ell(\overline{x})} b_1(\overline{x}, i) \wedge b_2(\overline{x})
\end{aligned}$$

$$\neg \bigvee_{i=0}^{\ell(\overline{x})} b(\overline{x}, i) = \bigwedge_{i=0}^{\ell(\overline{x})} \neg b(\overline{x}, i)$$
$$\neg \bigwedge_{i=0}^{\ell(\overline{x})} b(\overline{x}, i) = \bigvee_{i=0}^{\ell(\overline{x})} \neg b(\overline{x}, i)$$

□

From now on we consider only normal expressions.

7.6.2 Boolean Conditional Equalities

A *boolean conditional equality* over a signature $\mathcal{SIG}$ is a formula of the form:

$$p(\overline{x}) \longrightarrow (e(\overline{x}) = b(\overline{x}))$$

where $p(\overline{x})$ is a conjunction of linear predicates, $e(\overline{x})$ is a linear guarded boolean expression over $\mathcal{SIG}$, and $b(\overline{x})$ is a boolean expression over $\mathcal{SIG}$, which is well-defined under $p(\overline{x})$.

In the rest of this chapter we consider decidability of validity of boolean conditional equalities. To do this we need to consider equality of boolean expressions.

7.6.3 Equality of Boolean Expressions

In this section we consider a subclass of boolean expressions, and show that the equality of any two boolean expressions, with signatures $\mathcal{SIG}$, in the subclass, over the two valued boolean algebra B_2 (introduced in appendix B) and the algebraic specification $\mathcal{SPEC}$ of the class of all $\mathcal{SIG}$-algebras, is equivalent to the validity of a Presburger formula. To define the subclass and perform the proofs we need some definitions.

A normal boolean expression is said to be *disjunctive* if it is first order and in disjunctive form, or if it is of the form $\bigvee_{i=0}^{\ell(\overline{x})} b(\overline{x}, i)$. A normal boolean expression is said to be *conjunctive* if it is first order and in conjunctive form, or if it is of the form $\bigwedge_{i=0}^{\ell(\overline{x})} b(\overline{x}, i)$.

The *alternation degree* $\mathcal{D}(b(\overline{x}))$ of a normal boolean expression $b(\overline{x})$ is defined as follows:

- If $b(\overline{x})$ is disjunctive and first order, where each element of the disjunction is either an atomic expression or the negation of an atomic expression, then $\mathcal{D}(b(\overline{x})) = 1$. If $b(\overline{x})$ is conjunctive and first order, where each element of the conjunction is either an atomic expression or the negation of an atomic expression, then $\mathcal{D}(b(\overline{x})) = 1$.

- If $b(\overline{x})$ is disjunctive and first order, where at least one element of the disjunction is a conjunction then $\mathcal{D}(b(\overline{x})) = 2$. If $b(\overline{x})$ is conjunctive and first order, where at least one element of the conjunction is a disjunction then $\mathcal{D}(b(\overline{x})) = 2$.

- If $b(\overline{x})$ is of the form $\bigvee_{i=0}^{\ell(\overline{x})} b'(\overline{x}, i)$, where $b'(\overline{x}, i)$ is disjunctive then $\mathcal{D}(b(\overline{x})) = \mathcal{D}(b'(\overline{x}))$. If $b(\overline{x})$ is of the form $\bigvee_{i=0}^{\ell(\overline{x})} b'(\overline{x}, i)$, where $b'(\overline{x}, i)$ is not disjunctive then $\mathcal{D}(b(\overline{x})) = \mathcal{D}(b'(\overline{x})) + 1$. If $b(\overline{x})$ is of the form $\bigwedge_{i=0}^{\ell(\overline{x})} b'(\overline{x}, i)$, where $b'(\overline{x}, i)$ is conjunctive then $\mathcal{D}(b(\overline{x})) = \mathcal{D}(b'(\overline{x}))$. If $b(\overline{x})$ is of the form $\bigwedge_{i=0}^{\ell(\overline{x})} b'(\overline{x}, i)$, where $b'(\overline{x}, i)$ is not conjunctive then $\mathcal{D}(b(\overline{x})) = \mathcal{D}(b'(\overline{x})) + 1$.

Let $b_1(\overline{x})$ and $b_2(\overline{x})$ be normal boolean expressions. We say that $b_1(\overline{x})$ and $b_2(\overline{x})$ are *similar*[2] if $\mathcal{D}(b_1(\overline{x})) = \mathcal{D}(b_2(\overline{x}))$, and both are disjunctive or both are conjunctive.

Two uninterpreted terms $t_1(\overline{x})$ and $t_2(\overline{x})$ are said to be *matching* if:

- $b_1(\overline{x})$ and $b_2(\overline{x})$ are of the forms $a_1(\overline{q}_1(\overline{x}))$ and $a_2(\overline{q}_2(\overline{x}))$, and $a_1 = a_2$.
- $b_1(\overline{x})$ and $b_2(\overline{x})$ are of the forms $g_1(t_{11}(\overline{x}),\ldots,t_{1n_1}(\overline{x}))$ and $g_2(t_{21}(\overline{x}),\ldots,t_{2n_2}(\overline{x}))$ respectively, and $g_1 = g_2$ and $t_{1i}(\overline{x})$ and $t_{2i}(\overline{x})$ are matching for $1 \leq i \leq n_1 = n_2$.

Two atomic boolean expressions $g_1(t_{11}(\overline{x}),\ldots,t_{1n_1}(\overline{x}))$ and $g_2(t_{21}(\overline{x}),\ldots,t_{2n_2}(\overline{x}))$ are said to be *matching* if $g_1 = g_2$ and $t_{1i}(\overline{x})$ and $t_{2i}(\overline{x})$ are matching for $1 \leq i \leq n_1 = n_2$.

A boolean expression is said to be *plain* if it does not contain the $\neg$ operator.

The *index* of an uninterpreted term $t(\overline{x})$ is a tuple of linear QI-polynomials defined as follows:

- If $t(\overline{x})$ is of the form $a(\overline{q}(\overline{x}))$ then the index of $t(\overline{x})$ is $\overline{q}(\overline{x})$.
- If $t(\overline{x})$ is of the form $g(t_1(\overline{x}),\ldots,t_n(\overline{x}))$ then the index of $t(\overline{x})$ is $< \overline{q}_1(\overline{x}),\ldots,\overline{q}_n(\overline{x}) >$, where $\overline{q}_i(\overline{x})$ is the index of $t_i(\overline{x})$.

The *index* of an atomic boolean expression $g(t_1(\overline{x}),\ldots,t_n(\overline{x}))$ is $< \overline{q}_1(\overline{x}),\ldots,\overline{q}_n(\overline{x}) >$, where $\overline{q}_i(\overline{x})$ is the index of $t_i(\overline{x})$.

Lemma 55 *Let $b_1(\overline{x})$ and $b_2(\overline{x})$ be similar plain boolean expressions over a signature $\mathcal{SIG}$, each with alternation degrees ≤ 2. Let $\mathcal{SPEC}$ be the algebraic specification of the class of all $\mathcal{SIG}$-algebras. Then there is a Presburger formula $P(\overline{x})$ such that for each $\overline{\alpha}$, if $b_1(\overline{\alpha})$ and $b_2(\overline{\alpha})$ are well-defined, then $P(\overline{\alpha})$ is true iff $b_1(\overline{\alpha}) = b_2(\overline{\alpha})$ over B_2 and $\mathcal{SPEC}$.*

Proof: We will carry out the proof for some of the cases where all the atomic boolean expressions in $b_1(\overline{x})$ and $b_2(\overline{x})$ are matching. The proof can easily be generalized to any similar pair of plain boolean expressions with alternation degrees less than or equal to 2.

We will assume that $\mathcal{D}(b_1(\overline{x})) = \mathcal{D}(b_2(\overline{x})) = 2$. The proofs for $\mathcal{D}(b_1(\overline{x})) < 2$ or $\mathcal{D}(b_2(\overline{x})) < 2$ will be special cases.

We will consider a number of cases:

1. If $b_1(\overline{x})$ and $b_2(\overline{x})$ are of the forms:

$$(b_{11}(\overline{x}) \wedge \cdots \wedge b_{1m_1}(\overline{x})) \vee \cdots \vee (b_{n1}(\overline{x}) \wedge \cdots \wedge b_{nm_n}(\overline{x}))$$

and:

$$(d_{11}(\overline{x}) \wedge \cdots \wedge d_{1l_1}(\overline{x})) \vee \cdots \vee (d_{k1}(\overline{x}) \wedge \cdots \wedge d_{kl_k}(\overline{x}))$$

respectively. Let the indices of $b_{ij}(\overline{x})$ and $d_{ij}(\overline{x})$ be $\overline{qb}_{ij}(\overline{x})$ and $\overline{qd}_{ij}(\overline{x})$ respectively. Then $P(\overline{x})$ can be defined as:

$$P(\overline{x}) = \left(\begin{array}{c} \forall(1 \leq i_1 \leq n).\ \exists(1 \leq j_1 \leq k).\ \forall(1 \leq j_2 \leq l_{j_1}).\ \exists(1 \leq i_2 \leq m_{i_1}).\ \left(\overline{qb}_{i_1 i_2}(\overline{x}) = \overline{qd}_{j_1 j_2}(\overline{x})\right) \\ \wedge \\ \forall(1 \leq j_1 \leq k).\ \exists(1 \leq i_1 \leq n).\ \forall(1 \leq i_2 \leq m_{i_1}).\ \exists(1 \leq j_2 \leq l_{j_1}).\ \left(\overline{qb}_{i_1 i_2}(\overline{x}) = \overline{qd}_{j_1 j_2}(\overline{x})\right) \end{array} \right)$$

[2]The definitions of the notions of *similarity* and *matching* as defined in this chapter are not related to their definitions in chapter 5.

Now we will show that for each $\overline{\alpha}$, $P(\overline{\alpha})$ is true iff $b_1(\overline{\alpha}) = b_2(\overline{\alpha})$ over B_2 and $\mathcal{SPEC}$.
($\longrightarrow$): Suppose that $P(\overline{\alpha})$ is true. We will show that $b_1(\overline{\alpha}) = tt$ over B_2 and $\mathcal{SPEC}$ iff $b_2(\overline{\alpha}) = tt$ over B_2 and $\mathcal{SPEC}$.

First we will show that $[\![b_1(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt$ iff $[\![b_2(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt$ for any $\mathcal{SIG}$-algebra A and stream interpretation $\mathcal{I}$ over A.

Suppose that:

$$[\![b_1(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt$$

then:

$$\exists(1 \leq i_1 \leq n).\ \forall(1 \leq i_2 \leq m_{i_1}).\ \left([\![b_{i_1 i_2}(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt\right)$$

From the definition of $P(\overline{x})$ it follows that:

$$\exists(1 \leq j_1 \leq k).\ \forall(1 \leq j_2 \leq l_{j_1}).\ \left([\![d_{j_1 j_2}(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt\right)$$

This implies that:

$$[\![b_2(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt$$

Now suppose that $b_1(\overline{\alpha}) = tt$ over B_2 and $\mathcal{SPEC}$. This means that $[\![b_1(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt$ for each $\mathcal{SIG}$-algebra A and stream interpretation $\mathcal{I}$ over A. From the discussion above it follows that $[\![b_2(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt$ for each $\mathcal{SIG}$-algebra A and stream interpretation $\mathcal{I}$ over A. This implies that $b_2(\overline{\alpha}) = tt$ over B_2 and $\mathcal{SPEC}$. It follows that if $b_1(\overline{\alpha}) = tt$ over B_2 and $\mathcal{SPEC}$ then $b_2(\overline{\alpha}) = tt$ over B_2 and $\mathcal{SPEC}$.

In a similar manner we can show that if $b_2(\overline{\alpha}) = tt$ over B_2 and $\mathcal{SPEC}$ then $b_1(\overline{\alpha}) = tt$ over B_2 and $\mathcal{SPEC}$.
($\longleftarrow$): Suppose that $P(\overline{\alpha})$ is false. We will find a $\mathcal{SIG}$-algebra A and a stream interpretation $\mathcal{I}$ over A such that $[\![b_1(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} \neq [\![b_2(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}}$.

That $P(\overline{\alpha})$ is false means that either:

$$\exists(1 \leq i_1 \leq n).\ \forall(1 \leq j_1 \leq k).\ \exists(1 \leq j_2 \leq l_{j_1}).\ \forall(1 \leq i_2 \leq m_{i_1}).\ \left(\overline{qb}_{i_1 i_2}(\overline{\alpha}) \neq \overline{qd}_{j_1 j_2}(\overline{\alpha})\right) \quad (7.10)$$

or:

$$\exists(1 \leq j_1 \leq k).\ \forall(1 \leq i_1 \leq n).\ \exists(1 \leq i_2 \leq m_{i_1}).\ \forall(1 \leq j_2 \leq l_{j_1}).\ \left(\overline{qb}_{i_1 i_2}(\overline{\alpha}) \neq \overline{qd}_{j_1 j_2}(\overline{\alpha})\right) \quad (7.11)$$

In case the formula in (7.10) is true we will find a $\mathcal{SIG}$-algebra A and a stream interpretation $\mathcal{I}$ over A such that $[\![b_1(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt$ and $[\![b_2(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = ff$. Similarly if the formula in (7.11) is true we will find a $\mathcal{SIG}$-algebra A and a stream interpretation $\mathcal{I}$ over A such that $[\![b_2(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt$ and $[\![b_1(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = ff$.

Suppose that the formula in (7.10) is true. We define a $\mathcal{SIG}$-algebra A and a stream interpretation $\mathcal{I}$ over A in the following way:

$$[\![b_{i_1 i_2}(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt \quad (7.12)$$

for each $1 \leq i_2 \leq m_{i_1}$, and:

$$[\![d_{j_1 j_2}(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = ff \quad (7.13)$$

for each $1 \leq j_1 \leq k$. It is clear that such an algebra and stream interpretation exist. From (7.12) we get:

$$\begin{array}{l} \left[\!\left[b_{i_1 1}(\overline{\alpha}) \wedge \cdots \wedge b_{i_1 m_{i_1}}(\overline{\alpha}) \right]\!\right]_{B_2,A,\mathcal{I}} = \\ [\![b_{i_1 1}(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} \wedge \cdots \wedge \left[\!\left[b_{i_1 m_{i_1}}(\overline{\alpha}) \right]\!\right]_{B_2,A,\mathcal{I}} = tt \end{array}$$

which means that:

$$\begin{array}{l} [\![b_1(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = \\ [\![(b_{11}(\overline{\alpha}) \wedge \cdots \wedge b_{1m_1}(\overline{\alpha})) \vee \cdots \vee (b_{n1}(\overline{\alpha}) \wedge \cdots \wedge b_{nm_n}(\overline{\alpha}))]\!]_{B_2,A,\mathcal{I}} = \\ [\![b_{11}(\overline{\alpha}) \wedge \cdots \wedge b_{1m_1}(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} \vee \cdots \vee [\![b_{n1}(\overline{\alpha}) \wedge \cdots \wedge b_{nm_n}(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt \end{array}$$

From (7.13) we get:

$$\begin{array}{l} \left[\!\left[d_{j_1 1}(\overline{\alpha}) \wedge \cdots \wedge d_{j_1 l_{j_1}}(\overline{\alpha}) \right]\!\right]_{B_2,A,\mathcal{I}} = \\ [\![d_{j_1 1}(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} \wedge \cdots \wedge \left[\!\left[d_{j_1 l_{j_1}}(\overline{\alpha}) \right]\!\right]_{B_2,A,\mathcal{I}} = ff \end{array}$$

for each $1 \leq j_1 \leq k$, which means that:

$$\begin{array}{l} [\![b_2(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = \\ [\![(d_{11}(\overline{\alpha}) \wedge \cdots \wedge d_{1l_1}(\overline{\alpha})) \vee \cdots \vee (d_{k1}(\overline{\alpha}) \wedge \cdots \wedge d_{kl_k}(\overline{\alpha}))]\!]_{B_2,A,\mathcal{I}} = \\ [\![d_{11}(\overline{\alpha}) \wedge \cdots \wedge d_{1l_1}(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} \vee \cdots \vee [\![d_{k1}(\overline{\alpha}) \wedge \cdots \wedge d_{kl_k}(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = ff \end{array}$$

In a similar manner we can show that if the formula in (7.11) is true then we can find a $\mathcal{SIG}$-algebra A and a stream interpretation $\mathcal{I}$ over A such that $[\![b_2(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = tt$ and $[\![b_1(\overline{\alpha})]\!]_{B_2,A,\mathcal{I}} = ff$.

2. If $b_1(\overline{x})$ and $b_2(\overline{x})$ are of the forms:

$$(b_{11}(\overline{x}) \vee \cdots \vee b_{1m_1}(\overline{x})) \wedge \cdots \wedge (b_{n1}(\overline{x}) \vee \cdots \vee b_{nm_n}(\overline{x}))$$

and:

$$(d_{11}(\overline{x}) \vee \cdots \vee d_{1l_1}(\overline{x})) \wedge \cdots \wedge (d_{k1}(\overline{x}) \vee \cdots \vee d_{kl_k}(\overline{x}))$$

respectively. The proof can be carried out in a similar manner to case (1) above.

3. If $b_1(\overline{x})$ and $b_2(\overline{x})$ are of the forms[3]:

$$\bigvee_{\overline{i}=\overline{0}}^{\overline{\ell}_1(\overline{x})} (b_{11}(\overline{x},\overline{i}) \wedge \cdots \wedge b_{1m_1}(\overline{x},\overline{i})) \vee \cdots \vee (b_{n1}(\overline{x},\overline{i}) \wedge \cdots \wedge b_{nm_n}(\overline{x},\overline{i}))$$

and:

$$\bigvee_{\overline{j}=\overline{0}}^{\overline{\ell}_2(\overline{x})} (d_{11}(\overline{x},\overline{j}) \wedge \cdots \wedge d_{1l_1}(\overline{x},\overline{j})) \vee \cdots \vee (d_{k1}(\overline{x},\overline{j}) \wedge \cdots \wedge d_{kl_k}(\overline{x},\overline{j}))$$

[3]We write $\bigvee_{\overline{i}=\overline{0}}^{\overline{\ell}(\overline{x})} b(\overline{x},\overline{i})$ instead of $\bigvee_{i_1=0}^{\ell_1(\overline{x})} \cdots \bigvee_{i_n=0}^{\ell_n(\overline{x},i_1,\ldots,i_{n-1})} b(\overline{x},i_1,\ldots,i_n)$, whenever n is known or irrelevant in the context. The same applies to $\bigwedge_{\overline{i}=\overline{0}}^{\overline{\ell}(\overline{x})} b(\overline{x},\overline{i})$.

Let the indices of $b_{ij}(\overline{x},\overline{i})$ and $d_{ij}(\overline{x},\overline{j})$ be $\overline{qb}_{ij}(\overline{x},\overline{i})$ and $\overline{qd}_{ij}(\overline{x},\overline{j})$ respectively. Then $P(\overline{x})$ can be defined as:

$$P(\overline{x}) = \left(\begin{array}{c} \left(\begin{array}{l} \forall(\overline{0} \leq \overline{i} \leq \overline{\ell}_1(\overline{x})).\ \forall(1 \leq i_1 \leq n).\ \exists(\overline{0} \leq \overline{j} \leq \overline{\ell}_2(\overline{x})).\ \exists(1 \leq j_1 \leq k). \\ \forall(1 \leq j_2 \leq l_{j_1}).\ \exists(1 \leq i_2 \leq m_{i_1}).\ \left(\overline{qb}_{i_1 i_2}(\overline{x},\overline{i}) = \overline{qd}_{j_1 j_2}(\overline{x},\overline{j})\right) \end{array} \right) \\ \wedge \\ \left(\begin{array}{l} \forall(\overline{0} \leq \overline{j} \leq \overline{\ell}_2(\overline{x})).\ \forall(1 \leq j_1 \leq k).\ \exists(\overline{0} \leq \overline{i} \leq \overline{\ell}_1(\overline{x})).\ \exists(1 \leq i_1 \leq n). \\ \forall(1 \leq i_2 \leq m_{i_1}).\ \exists(1 \leq j_2 \leq l_{j_1}).\ \left(\overline{qb}_{i_1 i_2}(\overline{x},\overline{i}) = \overline{qd}_{j_1 j_2}(\overline{x},\overline{j})\right) \end{array} \right) \end{array} \right)$$

The rest of the proof can be carried out in a similar manner to case (1) above.

4. If $b_1(\overline{x})$ and $b_2(\overline{x})$ are of the forms:

$$\bigwedge_{\overline{i}=\overline{0}}^{\overline{\ell}_1(\overline{x})} (b_{11}(\overline{x},\overline{i}) \vee \cdots \vee b_{1m_1}(\overline{x},\overline{i})) \wedge \cdots \wedge (b_{n1}(\overline{x},\overline{i}) \vee \cdots \vee b_{nm_n}(\overline{x},\overline{i}))$$

and:

$$\bigwedge_{\overline{j}=\overline{0}}^{\overline{\ell}_2(\overline{x})} (d_{11}(\overline{x},\overline{j}) \vee \cdots \vee d_{1l_1}(\overline{x},\overline{j})) \wedge \cdots \wedge (d_{k1}(\overline{x},\overline{j}) \vee \cdots \vee d_{kl_k}(\overline{x},\overline{j}))$$

The proof can be carried out in a similar manner to case (3) above.

5. If $b_1(\overline{x})$ and $b_2(\overline{x})$ are of the forms:

$$\bigvee_{\overline{i}_1=\overline{0}}^{\overline{\ell}_1(\overline{x})} \bigwedge_{\overline{i}_2=\overline{0}}^{\overline{\ell}_2(\overline{x},\overline{i}_1)} (b_1(\overline{x},\overline{i}_1,\overline{i}_2) \wedge \cdots \wedge b_n(\overline{x},\overline{i}_1,\overline{i}_2))$$

and:

$$\bigvee_{\overline{j}_1=\overline{0}}^{\overline{\ell}_3(\overline{x})} \bigwedge_{\overline{j}_2=\overline{0}}^{\overline{\ell}_4(\overline{x},\overline{j}_1)} (d_1(\overline{x},\overline{j}_1,\overline{j}_2) \wedge \cdots \wedge d_m(\overline{x},\overline{j}_1,\overline{j}_2))$$

Let the indices of $b_i(\overline{x},\overline{i}_1,\overline{i}_2)$ and $d_i(\overline{x},\overline{j}_1,\overline{j}_2)$ be $\overline{qb}_i(\overline{x},\overline{i}_1,\overline{i}_2)$ and $\overline{qd}_i(\overline{x},\overline{j}_1,\overline{j}_2)$ respectively. Then $P(\overline{x})$ can be defined as:

$$P(\overline{x}) = \left(\begin{array}{c} \left(\begin{array}{l} \forall(\overline{0} \leq \overline{i}_1 \leq \overline{\ell}_1(\overline{x})).\ \exists(\overline{0} \leq \overline{j}_1 \leq \overline{\ell}_3(\overline{x})).\ \forall(\overline{0} \leq \overline{j}_2 \leq \overline{\ell}_4(\overline{x},\overline{j}_1)).\ \forall(1 \leq j \leq m). \\ \exists(\overline{0} \leq \overline{i}_2 \leq \overline{\ell}_2(\overline{x},\overline{i}_1)).\ \exists(1 \leq i \leq n).\ \left(\overline{qb}_i(\overline{x},\overline{i}_1,\overline{i}_2) = \overline{qd}_j(\overline{x},\overline{j}_1,\overline{j}_2)\right) \end{array} \right) \\ \wedge \\ \left(\begin{array}{l} \forall(\overline{0} \leq \overline{j}_1 \leq \overline{\ell}_3(\overline{x})).\ \exists(\overline{0} \leq \overline{i}_1 \leq \overline{\ell}_1(\overline{x})).\ \forall(\overline{0} \leq \overline{i}_2 \leq \overline{\ell}_2(\overline{x},\overline{i}_1)).\ \forall(1 \leq i \leq n). \\ \exists(\overline{0} \leq \overline{j}_2 \leq \overline{\ell}_4(\overline{x},\overline{j}_1)).\ \exists(1 \leq j \leq m).\ \left(\overline{qb}_i(\overline{x},\overline{i}_1,\overline{i}_2) = \overline{qd}_j(\overline{x},\overline{j}_1,\overline{j}_2)\right) \end{array} \right) \end{array} \right)$$

The rest of the proof can be carried out in a similar manner to case (1) above.

6. If $b_1(\overline{x})$ and $b_2(\overline{x})$ are of the forms:

$$\bigwedge_{\overline{i}_1=\overline{0}}^{\overline{\ell}_1(\overline{x})} \bigvee_{\overline{i}_2=\overline{0}}^{\overline{\ell}_2(\overline{x},\overline{i}_1)} (b_1(\overline{x},\overline{i}_1,\overline{i}_2) \vee \cdots \vee b_n(\overline{x},\overline{i}_1,\overline{i}_2))$$

and:

$$\bigwedge_{\overline{j}_1=\overline{0}}^{\overline{\ell}_3(\overline{x})} \bigvee_{\overline{j}_2=\overline{0}}^{\overline{\ell}_4(\overline{x},\overline{j}_1)} (d_1(\overline{x},\overline{j}_1,\overline{j}_2) \vee \cdots \vee d_m(\overline{x},\overline{j}_1,\overline{j}_2))$$

The proof can be carried out in a similar manner to case (5) above. □

7.6.4 Deciding the Validity of Boolean Conditional Equalities

In this section we use the results of section 7.6.3 to define a subclass of boolean conditional equalities and show that the problem of validity of the elements of the subclass is decidable.

Consider a boolean conditional equality $\phi(\overline{x})$ of the form:

$$p(\overline{x}) \quad \longrightarrow \quad (e(\overline{x}) = b(\overline{x}))$$

where $e(\overline{x})$ is of the form:

$$\textbf{case} \quad p_1(\overline{x}) \Rightarrow b_1(\overline{x}) \; ; \; \ldots \; ; \; p_m(\overline{x}) \Rightarrow b_m(\overline{x}) \quad \textbf{endcase}$$

The *alternation degree* $\mathcal{D}(\phi(\overline{x}))$ of $\phi(\overline{x})$ is $max(\mathcal{D}(b(\overline{x})), \mathcal{D}(b_1(\overline{x})), \ldots, \mathcal{D}(b_m(\overline{x})))$. We say that $\phi(\overline{x})$ is *balanced* if $b_1(\overline{x}), \ldots, b_m(\overline{x})$ all are similar to $b(\overline{x})$. We say that $\phi(\overline{x})$ is *plain* if $b(\overline{x}), b_1(\overline{x}), \ldots, b_m(\overline{x})$ all are plain.

Lemma 56 *The validity of plain balanced boolean conditional equalities, with signatures $\mathcal{SIG}$ and alternation degrees ≤ 2, over B_2 and the class $\mathcal{SPEC}$ of all $\mathcal{SIG}$-algebras, can be reduced to the validity of Presburger formulas.*

Proof: Consider a conditional equality $\phi(\overline{x})$, with a signature $\mathcal{SIG}$ and an alternation degree ≤ 2, of the form:

$$p(\overline{x}) \quad \longrightarrow \quad (e(\overline{x}) = b(\overline{x}))$$

Let $e(\overline{x})$ be of the form:

$$\textbf{case} \quad p_1(\overline{x}) \Rightarrow b_1(\overline{x}) \; ; \; \ldots \; ; \; p_m(\overline{x}) \Rightarrow b_m(\overline{x}) \quad \textbf{endcase}$$

It is clear that the validity of $\phi(\overline{x})$ over B_2 and $\mathcal{SPEC}$ is equivalent to the validity of:

$$\begin{array}{c} (p(\overline{x}) \longrightarrow (p_1(\overline{x}) \vee \cdots \vee p_m(\overline{x}))) \\ \wedge \\ (p(\overline{x}) \wedge p_1(\overline{x}) \longrightarrow (b(\overline{x}) = b_1(\overline{x}))) \wedge \cdots \wedge (p(\overline{x}) \wedge p_m(\overline{x}) \longrightarrow (b(\overline{x}) = b_m(\overline{x}))) \end{array}$$

over B_2 and $\mathcal{SPEC}$.

From lemma 55 it follows that there are Presburger formulas $P_1(\overline{x}), \ldots, P_m(\overline{x})$ such that for each $\overline{\alpha}$ if $b(\overline{\alpha})$ and $b_i(\overline{\alpha})$ are well-defined then $P_i(\overline{\alpha})$ is true iff $b(\overline{\alpha}) = b_i(\overline{\alpha})$ over B_2 and $\mathcal{SPEC}$. We know that $b(\overline{x})$ is well-defined under $p(\overline{x})$, and that $b_i(\overline{x})$ is well-defined under $p_i(\overline{x})$ for $1 \leq i \leq m$. It follows that the validity of $\phi(\overline{x})$ over B_2 and $\mathcal{SPEC}$ is equivalent to:

$$\begin{array}{c} (p(\overline{x}) \longrightarrow (p_1(\overline{x}) \vee \cdots \vee p_m(\overline{x}))) \\ \wedge \\ (p(\overline{x}) \wedge p_1(\overline{x}) \longrightarrow P_1(\overline{x})) \wedge \cdots \wedge (p(\overline{x}) \wedge p_m(\overline{x}) \longrightarrow P_m(\overline{x})) \end{array}$$

which is a Presburger formula. □

Theorem 57 *The validity of plain balanced boolean conditional equalities, with signatures $\mathcal{SIG}$ and alternation degrees ≤ 2, over B_2 and the class $\mathcal{SPEC}$ of all $\mathcal{SIG}$-algebras is decidable.*

Proof: The result follows immediately from lemma 56 and decidabilty of validity of Presburger formulas (see section 6.6.1). □

7.6.5 Second Step of Verification of The Substring Detecting Circuit: Checking Boolean Conditional Equalities

In section 7.5.2 we carried out the first of the two verification steps on the substring detecting circuit. We described $loc_2(x, \ell_1, \ell_2, t)$ as a linear guarded boolean expression (see equation (7.8)).

Here we perform the second step. Let $\mathcal{SIG} =< \{D, B\}, \{=\} >$. We use the algorithm in theorem 57 to check whether the value of $loc_2(x, \ell_1, \ell_2, t)$ described by (7.8) fulfills the specification of the circuit as stated in (3.49), over the two-valued boolean algebra B_2 and the algebraic specification $\mathcal{SPEC}$ of the class of all $\mathcal{SIG}$-algebras.

Recalling the specification formula as stated in (3.49)

$$
\begin{array}{l}
spec(x, \ell_1, \ell_2, t) \; = \\
(1 \leq \ell_2) \wedge (\ell_2 \leq \ell_1) \wedge (t = \ell_1 + \ell_2 - 1) \wedge (x = \ell_1 - \ell_2 + 1) \longrightarrow \\
\quad \Big(loc_2(x, \ell_1, \ell_2, t) = \bigvee_{j=0}^{\ell_1 - \ell_2} \bigwedge_{i=0}^{\ell_2 - 1} (a(i + j + 1) = b(i + 1))\Big) \\
\qquad\qquad \wedge \\
\quad (1 \leq \ell_1) \wedge (\ell_1 < \ell_2) \wedge (x = 1) \wedge (t = 2\ell_2) \longrightarrow \\
\quad \Big(loc_2(x, \ell_1, \ell_2, t) = \bigvee_{j=0}^{\ell_2 - \ell_1} \bigwedge_{i=0}^{\ell_1 - 1} (a(i + 1) = b(i + j + 1))\Big)
\end{array}
$$

Here we will consider only the first element of the conjunction in the specification formula above. The second element can be treated similarly. Let:

$$
\begin{array}{lcl}
p(x, \ell_1, \ell_2, t) & = & (1 \leq \ell_2) \wedge (\ell_2 \leq \ell_1) \wedge (t = \ell_1 + \ell_2 - 1) \wedge (x = \ell_1 - \ell_2 + 1) \\
b(x, \ell_1, \ell_2, t) & = & \bigvee_{j=0}^{\ell_1 - \ell_2} \bigwedge_{i=0}^{\ell_2 - 1} (a(i + j + 1) = b(i + 1))
\end{array}
$$

From equation (7.8) we know that:

$loc_2(x, \ell_1, \ell_2, t) =$
case

$$\left\{\begin{array}{c} 1 \le x \le \ell_1 \\ 0 < t \\ (t+x)\ mod\ 2 = 0 \\ x \le t \\ t \le 2\ell_2 + x - 2 \\ t + x = 2\ell_1 \end{array}\right\} \implies \bigvee_{j=0}^{x-1} \bigwedge_{i=0}^{\frac{t-x}{2}} \left(a\left(\frac{t-2i+2j-x+2}{2}\right) = b(\ell_2 - i)\right)$$

$$\left\{\begin{array}{c} 1 \le x \le \ell_1 \\ 0 < t \\ (t+x)\ mod\ 2 = 0 \\ x \le t \\ t \le 2\ell_2 + x - 2 \\ x = 1 \\ t + x > 2\ell_1 \end{array}\right\} \implies \bigvee_{j=0}^{\frac{t-2\ell_1+1}{2}} \bigwedge_{i=0}^{\ell_1-1} \left(a(\ell_1 - i) = b\left(\frac{2\ell_1+2\ell_2-t-2i+2j-1}{2}\right)\right)$$

$$\left\{\begin{array}{c} 1 \le x \le \ell_1 \\ 0 < t \\ (t+x+1)\ mod\ 2 = 0 \\ x \le t \\ t \le 2\ell_2 + x - 1 \\ x = 1 \\ t + x > 2\ell_1 \end{array}\right\} \implies \bigvee_{j=0}^{\frac{t-2\ell_1}{2}} \bigwedge_{i=0}^{\ell_1-1} \left(a(\ell_1 - i) = b\left(\frac{2\ell_1+2\ell_2-t-2i+2j}{2}\right)\right)$$

endcase

Let:

$$\begin{aligned} p_1(x, \ell_1, \ell_2, t) &= (1 \le x \le \ell_1) \wedge (0 < t) \wedge ((t+x)\ mod\ 2 = 0) \wedge (x \le t) \wedge \\ &\quad (t \le 2\ell_2 + x - 2) \wedge (t + x = 2\ell_1) \end{aligned}$$

$$\begin{aligned} p_2(x, \ell_1, \ell_2, t) &= (1 \le x \le \ell_1) \wedge (0 < t) \wedge ((t+x)\ mod\ 2 = 0) \wedge (x \le t) \wedge \\ &\quad (t \le 2\ell_2 + x - 2) \wedge (x = 1) \wedge (t + x > 2\ell_1) \end{aligned}$$

$$\begin{aligned} p_3(x, \ell_1, \ell_2, t) &= (1 \le x \le \ell_1) \wedge (0 < t) \wedge ((t+x+1)\ mod\ 2 = 0) \wedge (x \le t) \wedge \\ &\quad (t \le 2\ell_2 + x - 1) \wedge (x = 1) \wedge (t + x > 2\ell_1) \end{aligned}$$

and:

$$b_1(x, \ell_1, \ell_2, t) = \bigvee_{j=0}^{x-1} \bigwedge_{i=0}^{\frac{t-x}{2}} \left(a\left(\frac{t-2i+2j-x+2}{2}\right) = b(\ell_2 - i)\right)$$

$$b_2(x, \ell_1, \ell_2, t) = \bigvee_{j=0}^{\frac{t-2\ell_1+1}{2}} \bigwedge_{i=0}^{\ell_1-1} \left(a(\ell_1 - i) = b\left(\frac{2\ell_1+2\ell_2-t-2i+2j-1}{2}\right)\right)$$

$$b_3(x, \ell_1, \ell_2, t) = \bigvee_{j=0}^{\frac{t-2\ell_1}{2}} \bigwedge_{i=0}^{\ell_1-1} \left(a(\ell_1 - i) = b\left(\frac{2\ell_1+2\ell_2-t-2i+2j}{2}\right)\right)$$

Let:

$$\phi(x,\ell_1,\ell_2,t) \;=\; (p(x,\ell_1,\ell_2,t) \longrightarrow (loc_2(x,\ell_1,\ell_2,t) = b(x,\ell_1,\ell_2,t)))$$

We notice that $\phi(x,\ell_1,\ell_2,t)$ is a boolean conditional equality which is balanced, plain, and with an alternation degree of 2.

According to the method of lemma 56, there is a Presburger formula $P(x,\ell_1,\ell_2,t)$ such that for each α_1, α_2, α_3, and α_4, $P(\alpha_1,\alpha_2,\alpha_3,\alpha_4)$ is true iff $\phi(\alpha_1,\alpha_2,\alpha_3,\alpha_4)$ is true, and:

$$P(x,\ell_1,\ell_2,t)$$
$$=$$
$$\left(\begin{array}{l} p(x,\ell_1,\ell_2,t) \longrightarrow (p_1(x,\ell_1,\ell_2,t) \vee p_2(x,\ell_1,\ell_2,t) \vee p_3(x,\ell_1,\ell_2,t)) \\ \qquad\qquad \wedge \\ p(x,\ell_1,\ell_2,t) \wedge p_1(x,\ell_1,\ell_2,t) \longrightarrow (b(x,\ell_1,\ell_2,t) = b_1(x,\ell_1,\ell_2,t)) \\ \qquad\qquad \wedge \\ p(x,\ell_1,\ell_2,t) \wedge p_2(x,\ell_1,\ell_2,t) \longrightarrow (b(x,\ell_1,\ell_2,t) = b_2(x,\ell_1,\ell_2,t)) \\ \qquad\qquad \wedge \\ p(x,\ell_1,\ell_2,t) \wedge p_3(x,\ell_1,\ell_2,t) \longrightarrow (b(x,\ell_1,\ell_2,t) = b_3(x,\ell_1,\ell_2,t)) \end{array}\right)$$

It can be shown (by integer linear programming) that:

$$p(x,\ell_1,\ell_2,t) \longrightarrow (p_1(x,\ell_1,\ell_2,t) \vee p_2(x,\ell_1,\ell_2,t) \vee p_3(x,\ell_1,\ell_2,t))$$

is valid and that:

$$p(x,\ell_1,\ell_2,t) \wedge p_2(x,\ell_1,\ell_2,t)$$

and:

$$p(x,\ell_1,\ell_2,t) \wedge p_3(x,\ell_1,\ell_2,t)$$

both are unsatisfiable. This implies that:

$$\begin{array}{l} P(x,\ell_1,\ell_2,t) \;= \\ \quad p(x,\ell_1,\ell_2,t) \wedge p_1(x,\ell_1,\ell_2,t) \longrightarrow (b(x,\ell_1,\ell_2,t) = b_1(x,\ell_1,\ell_2,t)) \end{array}$$

According to the method of lemma 55 there is a Presburger formula $P_1(x,\ell_1,\ell_2,t)$ such that for each α_1, α_2, α_3, and α_4, $P_1(\alpha_1,\alpha_2,\alpha_3,\alpha_4)$ is true iff $b(\alpha_1,\alpha_2,\alpha_3,\alpha_4) = b_1(\alpha_1,\alpha_2,\alpha_3,\alpha_4)$ over B_2 and $\mathcal{SPEC}$. The indices of:

$$a(i+j+1) = b(i+1)$$

and :

$$a\left(\frac{t-2i+2j-x+2}{2}\right) = b(\ell_2 - i)$$

are $< i + j + 1, i + 1 >$ and $< \frac{t-2i+2j-x+2}{2}, \ell_2 - i >$ respectively thus:

$$P_1(\overline{x})$$

$$=$$

$$\left(\begin{array}{l} \left(\begin{array}{l} \forall(0 \leq i_1 \leq \ell_1 - \ell_2).\ \exists(0 \leq j_1 \leq x - 1).\ \forall(0 \leq j_2 \leq \frac{t-x}{2}).\ \exists(0 \leq i_2 \leq \ell_2 - 1). \\ \left(i_1 + i_2 + 1 = \frac{t-2j_1+2j_2-x+2}{2}\right) \wedge (i_1 + 1 = \ell_2 - j_1) \end{array} \right) \\ \\ \wedge \\ \\ \left(\begin{array}{l} \forall(0 \leq j_1 \leq x - 1).\ \exists(0 \leq i_1 \leq \ell_1 - \ell_2).\ \forall(0 \leq i_2 \leq \ell_2 - 1).\ \exists(0 \leq j_2 \leq \frac{t-x}{2}). \\ \left(i_1 + i_2 + 1 = \frac{t-2j_1+2j_2-x+2}{2}\right) \wedge (i_1 + 1 = \ell_2 - j_1) \end{array} \right) \end{array} \right)$$

It follows that:

$$\begin{array}{l} P(x, \ell_1, \ell_2, t) = \\ \quad p(x, \ell_1, \ell_2, t) \ \wedge \ p_1(x, \ell_1, \ell_2, t) \ \longrightarrow \ P_1(x, \ell_1, \ell_2, t) \end{array}$$

which is a Presburger formula.

Chapter 8

Conclusion and Future Work

In this section we summarize the main contributions of the thesis, and give some directions for future work.

8.1 Results

We have presented methods for automatic verification of several classes of systolic circuits. This was done in several steps. We provided a model for the formal description of implementations and specifications of systolic circuits as systems of recurrence equations over a signature. The implementation description of a circuit reflects the mathematical operations performed inside the cells and the interconnection pattern among the cells of the circuit. The specification is a statement of the desired relation between the output data and the input data of the circuit. We used basic ideas from the theories of abstract data types and fixed points to provide a semantics for our model, and then defined the formal meaning of circuit verification. The notion of *algebras* was used to give mathematical interpretations of the operations performed inside the circuit cells. Fixed point theory was used to relate the equation system corresponding to a circuit implementation to the equation system representing the specification.

Methods for automatic verification of three classes of circuits have been described in the thesis. The cell operations of the circuits of the three classes are interpreted as operators of a commutative ring, uninterpreted function symbols, and the operators of a boolean algebra respectively. For each class, we have shown how the respective automatic verification method may be applied to a non-trivial circuit in the class.

8.2 Future Work

The main directions for future work include:

Complexity and Feasibility No formal complexity analysis has been provided for our decision methods. However integer linear programming is NP-complete, and Presburger's arithmetic is doubly exponential, which suggests that the decision methods of chapters 5, 6, and 7 have at least the above complexities. Nevertheless, our experience with manual application of the decision methods on case studies indicates that they are

feasible in practice. This is because the complexity of the equation system describing the implementation of a circuit reflects the number of dimensions of the circuit and the number of states of the cell computations. These parameters are small in all the circuits known to us from the literature. Also, the size of the circuit does not affect the complexity of the verification method. This is due to the fact that we perform verification for families of circuits. The circuit parameters (which decide the size of the circuit) are considered as parameters of the verification problem. When we say that "a circuit is verified" then we mean that the circuit implementation is correct with respect to the specification for all values of the circuit parameters and consequently for all sizes of the circuit. It seems to be interesting to investigate the possibility of providing heuristics for efficient implementation of the algorithms.

Undecidability We have considered three classes of circuits for which we have shown that the verification problem is decidable. There are many ways in which these classes can be modified and the verification problem of the respective class made undecidable. In chapter 5 we introduced linear systolic circuits, where we restricted ourselves to linear equalities and inequalities. In fact if we work with arbitrary equalities and inequalities, it can be shown that Hilbert's tenth problem (which is undecidable [Dav73]) can be reduced to the verification problem of tail-recursive ring circuits (see section 5.2.2). In chapter 7 Hilbert's tenth problem can be reduced to the verification of arbitrary tail-recursive boolean circuits (and hence the restrictions of section 7.6.3). It would be interesting to give a more precise characterization of modifications which lead to undecidable problems for the classes of circuits discussed above.

Circuit Equivalence As mentioned earlier in the thesis, the notions of synthesis and verification are closely related. Often, synthesis is carried out by starting with a correct (but possibly inefficient) circuit, and then applying a sequence of transformations to derive the final implementation. Each transformation preserves the correctness of the implementation in the sense that the circuit to which it is applied is "equivalent" to the circuit it generates. In order that the final circuit would be guaranteed to be correct each transformation should be verified to be correct, i.e. the equivalence of its input circuit with its output circuit should be proven. Two circuits may be considered equivalent if they fulfill the same class of specifications. The notion of circuit equivalence is derivable form the notion of circuit verification. Consequently our verification algorithms seem to be applicable with some minor modifications to provide decision methods for proving correctness of circuit transformations.

Other Classes of Architectures Modifications of our methods to deal with other classes of architectures, which are closely related to systolic circuits as defined in this thesis, seem to be possible. Examples of such architectures are toroidal networks [Mar81, Seq81].

Appendix A

Rings

The algebraic specification $\mathcal{R}$ of (commutative) rings is defined by the following:

$$
\begin{array}{rrcl}
\textbf{sorts:} & S & & \\
\textbf{operation symbols:} & + & : S \times S \longrightarrow & S \\
 & \cdot & : S \times S \longrightarrow & S \\
 & - & : S \longrightarrow & S \\
 & 0 & : \longrightarrow & S \\
\textbf{equations:} & r_1 + (r_2 + r_3) & = & (r_1 + r_2) + r_3 \\
 & r_1 + r_2 & = & r_2 + r_1 \\
 & r + 0 & = & r \\
 & r + (-r) & = & 0 \\
 & (r_1 + r_2) \cdot r_3 & = & r_1 \cdot r_3 + r_2 \cdot r_3 \\
 & r_1 \cdot (r_2 + r_3) & = & r_1 \cdot r_2 + r_1 \cdot r_3 \\
 & r_1 \cdot r_2 & = & r_2 \cdot r_1
\end{array}
$$

A ring R is an $\mathcal{R}$-algebra $< \{S^R\}, \{+^R, \cdot^R, -^R, 0^R\} >$. For notation convenience we will denote the ring R by $< R, +, \cdot, -, 0 >$ whenever R is known or irrelevant in the context. Note that we use the same notation to denote the algebra R and the domain of the algebra R. Examples of rings are the ring of integers under addition and multiplication, and the ring of natural numbers modulo m (for some fixed natural number m) under addition and multiplication modulo m, etc.

Appendix B

Boolean Algebras

The algebraic specification $\mathcal{B}$ of boolean algebras is defined by the following:

sorts: S

operation symbols:

$$\begin{array}{lcrcl} \vee & : & S \times S & \longrightarrow & S \\ \wedge & : & S \times S & \longrightarrow & S \\ \neg & : & S & \longrightarrow & S \\ ff & : & & \longrightarrow & S \\ tt & : & & \longrightarrow & S \end{array}$$

equations:

$$\begin{array}{rcl} b \wedge b & = & b \\ b_1 \wedge b_2 & = & b_2 \wedge b_1 \\ (b_1 \wedge b_2) \wedge b_3 & = & b_1 \wedge (b_2 \wedge b_3) \\ b_1 \wedge (b_1 \vee b_2) & = & b_1 \\ b \wedge tt & = & b \\ b_1 \wedge (b_2 \vee b_3) & = & (b_1 \wedge b_2) \vee (b_1 \wedge b_3) \\ b \wedge \neg b & = & ff \\ b \vee b & = & b \\ b_1 \vee b_2 & = & b_2 \vee b_1 \\ (b_1 \vee b_2) \vee b_3 & = & b_1 \vee (b_2 \vee b_3) \\ b_1 \vee (b_1 \wedge b_2) & = & b_1 \\ b \vee ff & = & b \\ b_1 \vee (b_2 \wedge b_3) & = & (b_1 \vee b_2) \wedge (b_1 \vee b_3) \\ b \vee \neg b & = & tt \end{array}$$

A boolean algebra B is a $\mathcal{B}$-algebra $< \{S^B\}, \{\vee^B, \wedge^B, \neg^B, ff^B, tt^B\} >$. For notation convenience we will denote the boolean algebra B by $< B, \vee, \wedge, \neg, ff, tt >$ whenever B is known or irrelevant in the context. Note that we use the same notation to denote the algebra B and the domain of the algebra B.

An example of a boolean algebra which occurs often when modeling hardware circuits is the two valued boolean algebra B_2, where:

$$B_2 \;=\; < \{tt, ff\}, \vee, \wedge, \neg, ff, tt >$$

where:

$$ff \vee ff \;=\; ff$$

$$
\begin{aligned}
tt \vee ff &= tt \\
ff \vee tt &= tt \\
tt \vee tt &= tt \\
ff \wedge ff &= ff \\
tt \wedge ff &= ff \\
ff \wedge tt &= ff \\
tt \wedge tt &= tt \\
\neg ff &= tt \\
\neg tt &= ff
\end{aligned}
$$

Bibliography

[AAG*86] M. Annartone, E. Arnold, T. Gross, H. T. Kung, M. Lam, O. Menzilcioglu, K. Sarocky, and J. A. Webb. Warp architecture and implementation. In *Proceedings of the 13th Annual Symposium on Computer Architecture*, pages 346–356, June 1986.

[ABC*87] M. Annartone, F. Bitz, E. Clune, H. T. Kung, P. Maulik, H. Ribas, P. Tseng, and J. Webb. Architecture of Warp. In *Proc. Compcon Spring 87*, February 1987.

[AS88] T. Aboulnasr and W. Steenaart. Real-time systolic processor for 2-D spatial filtering. *IEEE Transactions on Circuits and Systems*, 35(4):451–455, 1988.

[BM79] R. S. Boyer and J. S. Moore. *A Computational Logic.* Academic Press, INC., 1979.

[BT76] I. Borosh and L. B. Treybig. Bounds on positive integral solutions of linear diophantine equations. *Proc. of American Mathematical Society*, (55):299–304, 1976.

[CGB86] E. M. Clarke, O. Grumberg, and M. C. Browne. Reasoning about networks with many identical finite state processes. In *ACM PoDC*, 1986.

[Che83] M. Chen. *Space-Time Algorithms: Semantics and Methodology.* PhD thesis, California Institute of Technology, Pasadena, California, 1983.

[Dav73] M. Davis. Hilbert's tenth problem is undecidable. *Amer. Math. Monthly*, (80):233–269, 1973.

[DLT89] J. Derrick, G. Lajos, and J. V. Tucker. *Specification and Verification of Synchronous Concurrent Algorithms using the Nuprl Proof Development System.* Technical Report, Centre for Theoretical Computer Science, The University of Leeds, 1989.

[EM85] H. Erhig and B. Mahr. *Fundamentals of Algebraic Specifications.* Volume 65 of *EATCS Monograph Series*, Springer Verlag, 1985.

[End72] H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, Inc., 1972.

[ET89] S. M. Eker and J. V. Tucker. Specification, derivation and verification of concurrent line drawing algorithms and architectures: a case study of pixel planes architecture. In P. M. Dew, R. A. Earnshow, and T. R. Heywood, editors, *Parallel Processing for Vision and Display*, Addison Wesley, 1989.

[FW87] J. A. Fortes and B. W. Wah. Systolic arrays from concept to implementation. *IEEE Computer*, 20(7):12–17, 1987.

[Gor86] M. Gordon. Why higher order logic is a good formalism for specifying and verifying hardware. In G. J. Milne and P. A. Subrahmanyam, editors, *Formal Aspects of VLSI Design, Proceedings of the 1985 Edinburgh Conference on VLSI*, North-Holland, Amsterdam, 1986.

[Gor87] M. Gordon. HOL: a proof generating system for higher order logic. In G. M. Birtwistle and P. A. Subrahmanyam, editors, *VLSI Specification, Verification and Synthesis, Proceedings of the Workshop on Hardware Verification, Calgary, 12-16 January, 1987*, North-Holland, Amsterdam, 1987.

[Gri88] E. P. Gribomont. Proving systolic arrays. In *CAAP'88, 13th Colloquium on Trees in Algebra and Programming*, pages 185–199, Springer-Verlag, LNCS 299, 1988.

[GTW78] J. A. Goguen, J. W. Thatcher, and E. G. Wagner. An initial algebra approach to the specification, correctness, and implementation of abstract data types. In R. T. Yeh, editor, *Current Trends in Programming Methodology*, chapter 5, pages 81–149, Prentice hall, Inc., 1978.

[Hen86] M. Hennessy. Proving systolic systems correct. *ACM TOPLAS*, 8(3):344–387, 1986.

[HL87] C. H. Huang and C. Lengauer. The derivation of systolic implementations of programs. *Acta Informatica*, 24(6):595–632, 1987.

[HL89] C. H. Huang and C. Lengauer. An incremental mechanical development of systolic solutions to the algebraic path problem. *Acta Informatica*, 27(2):97–124, 1989.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[HTT88] K. M. Hobley, B. C. Thompson, and J. V. Tucker. *Specification and Verification of Synchronous Concurrent Algorithms: A Case Study of A Convolution Algorithm.* Technical Report, Centre for Theoretical Computer Science, The University of Leeds, March 1988.

[Kun82] H. T. Kung. Why systolic architectures? *IEEE Computer*, 15(1):37–46, 1982.

[Kwa87] H. K. Kwan. Systolic realization of linear phase FIR digital filters. *IEEE Transactions on Circuits and Systems*, 34(12):1604–1605, 1987.

[Lis88] B. Lisper. Synthesis and equivalence of concurrent systems. *Theoretical Computer Science*, 58(1-3):183–199, 1988.

[Lis89] B. Lisper. *Synthesizing Synchronous Systems by Static Scheduling in Space-Time.* Springer-Verlag, LNCS 362, 1989.

[LJ88] W. Luk and G. Jones. From specification to parametrized architectures. In *Proc. of the IFIP International Working Conference on The Fusion of Hardware Design and Verification*, pages 263–284, 1988.

[LL85] R. J. Lipton and D. Lopresti. A systolic array for rapid string comparison. In *Chapel Hill Conference on VLSI*, pages 363–376, 1985.

[LS81] C. E. Leiserson and J. B. Saxe. Optimizing synchronous systems. In *Proceedings of the 22nd Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society, October 1981.

[Luk89] W. Luk. Specifying and developing regular heterogeneous design. In *Proc. of the IMEC-IFIP International Workshop on Applied Formal Methods for Correct VLSI Design*, pages 482–500, 1989.

[Man74] Z. Manna. *Mathematical Theory of Computation.* McGraw-Hill Inc., 1974.

[Mar81] A. J. Martin. The torus: an exercise in constructing a processing surface. In *Caltech Conference on VLSI*, 1981.

[MC80] C. Mead and L. Conway. *Introduction to VLSI Systems*, chapter 8, pages 263–332. Addison-Wesley, 1980.

[MG85] J. Meseguer and J. A. Goguen. Initiality, induction, and computability. In *Algebraic Methods in Semantics*, chapter 14, Cambridge University Press, 1985.

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

[Mol82] D. I. Moldovan. On the analysis and synthesis of VLSI algorithms. *IEEE Transactions on Computers*, 31(11):1121–1126, November 1982.

[Mol87] D. I. Moldovan. Advis: a software package for the design of systolic arrays. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 6(1):33–40, January 1987.

[MR84] R. G. Melhem and W. C. Rheinbold. A mathematical model for the verification of systolic networks. *SIAM J. Comput.*, 13(3):541–565, 1984.

[MT87] A. R. Martin and J. V. Tucker. The concurrent assignment representation of synchronous systems. In *PARLE*, Springer-Verlag, LNCS 259, 1987.

[Ole87] D. P. O'leary. Systolic arrays for matrix transpose and other reorderings. *IEEE Transactions on Computers*, 36(1):117–122, January 1987.

[PG88] J. D. Provence and S. C. Gupta. Systolic arrays for Viterbi processing in communication systems with a time-dipersive channel. *IEEE Transactions on Communications*, 36(10):1148–1156, 1988.

[PS88] S. Purushothaman and P. A. Subrahmanyam. Reasoning about systolic algorithms. *Journal of Parallel and Distributed Computing*, (5):669–699, 1988.

[PS89] S. Purushothaman and P. A. Subrahmanyam. Mechanical certification of systolic algorithms. *Journal of Automated Reasoning*, 5(1):67–95, 1989.

[Qui84] P. Quinton. Automatic synthesis of systolic arrays from recurrent uniform equations. In *Proc. 11th International Symposium on Computer Architecture*, pages 208–214, June 1984.

[Qui86] P. Quinton. An introduction to systolic architectures. In P. Treleaven and M. Vanneschi, editors, *Future Parallel Computers*, pages 387–400, Springer-Verlag, LNCS 272, 1986.

[Rab77] M. O. Rabin. Decidable theories. In *Handbook of Mathematical Logic*, chapter C.3, Elsevier Science Publishing Company, Inc., 1977.

[Raj89] S. V. Rajopadhye. Synthesizing systolic arrays with control signals from recurrence equations. *Distributed Computing*, 3(2):88–105, 1989.

[Rem87] M. Rem. Trace theory and systolic computations. In *PARLE*, pages 14–33, Springer-Verlag, LNCS 258, 1987.

[RF87] S. V. Rajopadhye and R. M. Fujimoto. Systolic array synthesis by static analysis of program dependencies. In *PARLE*, pages 295–310, Springer-Verlag, LNCS 258, 1987.

[Sel80] P. H. Sellers. The theory and computation of evolutionary distances: pattern recognition. *J. Algorithms*, (1):359–373, 1980.

[Seq81] C. H. Sequin. Doubly twisted torus networks for VLSI processor arrays. In *Proc. of the 8th Annual Symposium on Computer Architecture, Minnesota*, pages 471–480, 1981.

[She85] M. Sheeran. Designing regular array architectures using higher order functions. In *Proc. of the International Conference on Functional Programming Languages and Computer Architecture*, pages 220–237, Springer-Verlag, LNCS 201, 1985.

[She86] M. Sheeran. Design and verification of regular synchronous circuits. *Proc. of IEE*, 133(5):295–304, September 1986.

[TT88] B. C. Thompson and J. V. Tucker. *Synchronous Concurrent Algorithms*. Technical Report, Centre for Theoretical Computer Science, The University of Leeds, 1988.

[Ukk85] E. Ukkonen. Finding approximate patterns in strings. *J. Algorithms*, (1985):132–137, 1985.

[Ull84] J. D. Ullman. *Computational Aspects of VLSI*, chapter 5, pages 175–208. Computer Science Press, Inc., 1984.

Index

ISSN 0283-0574
Graphic Systems, Malmö, 1990