Table of Contents

A Flow Calculus of *mwp*-Bounds for Complexity Analysis

Neil D. Jones[1] and Lars Kristiansen[2,3]

[1] Department of Computer Science, University of Copenhagen
neil@diku.dk WWW home page: http://www.diku.dk/~neil
[2] Oslo University College, Faculty of Engineering
[3] Department of Mathematics, University of Oslo
larsk@math.uio.no WWW home page: http://www.iu.hio.no/~larskri

1 Introduction

Program complexity analysis seems naturally to decompose into two parts: a termination analysis and a data size analysis. Termination analyses aim to discover program-dependent well-founded orders on changes in data values around program loops. Most analyses focus mainly on values that are *copied* or *decreased*; values that may increase are ignored or treated very conservatively.

This paper considers the other part: data size analysis. Values that increase around loops are allowed and accounted for. The analysis aims, given a program, to find out whether its variables have acceptable growth rates.

1.1 The *mwp*-Calculus

Our approach is logical. We define a semantic "growth bound" relation $\models \texttt{C}:M$ where C is a program and M is an *mwp*-matrix. (For now, just consider the *mwp*-matrix as a collection of data describing some bounds on the values computed by C.) It follows straightforwardly from our definitions that there exists M such that $\models \texttt{C}:M$ holds if, and only if, every value computed by C is bounded by a polynomial in the inputs. However, the relation $\models \texttt{C}:M$ is of course undecidable, that is, no algorithm can decide if the relation holds.

After defining the semantic relation $\models \texttt{C} : M$, we proceed and introduce a corresponding provability relation $\vdash \texttt{C} : M$. We provide a syntactical proof calculus and define $\vdash \texttt{C}:M$ to hold if, and only if, $\vdash \texttt{C}:M$ is derivable in this calculus. The paper's main achievement is the soundness theorem stating that $\vdash \texttt{C}:M$ implies $\models \texttt{C}:M$.

By means of exhaustive proof search, an algorithm can decide if there exists M such that the relation $\vdash \texttt{C} : M$ holds, and thus, our results yield a computational method for certifying that the values computed by a program will be bounded by polynomials in the program's inputs.

1.2 Open Questions

This paper leaves some prominent questions open. We consider imperative programs on the natural numbers (non-negative integers) built from simple assignments and `skip` using sequential composition, conditionals and while commands, and the familiar iteration construct `loop X {C}`. Thus, our programming language is very rudimentary, and it is natural to ask if our methods extend to richer languages, for example, with pointers, classes, arrays or inductive data types. And even if we restrict the discussion to our simple programming language, how powerful are really our methods? We found such questions too extensive to discuss to any length in this first journal article on our approach. Still, we provide plenty of small examples which indicate that our methods might be powerful, and the key to deal with richer languages is likely to be buried somewhere in a successful analysis of our rudimentary, but essential, fragment.

Some readers may ask why we do not provide (implicit) characterisations of complexity classes. Well, this could be done by Turing machine simulations as seen in [2, 13, 15] and several others, but we do not think such characterisations will say anything substantial about the power of our methods.

In the present paper, we will explain and motivate the *mwp*-calculus, and if we manage to give readable and understandable proofs of the soundness properties, we will be satisfied. These proofs are long, technical and occasionally highly nontrivial. In this paper's sequel, we will analyse the power of the calculus and perhaps discuss possible extensions.

1.3 Related Work

The *mwp*-calculus is based on a careful and detailed analysis of the relationship between the resource requirements of a computation and the way data might flow during the computation. This analysis extends and refines work in the Implicit Complexity research community, e.g. Bellantoni & Cook ([2] normal and safe variables), Simmons ([24], active and dormant variables), Leivant ([20], ramification), and in particular, Kristiansen & Niggl ([15, 16] measures). The insight that there is a relationship between the absence and presence of successor-like functions and the computational complexity of programs is a part of the foundation of our calculus. Related work includes Jones [9, 10], Kristiansen & Voda [17, 18], Kristiansen [13, 14] and Hofmann's use of linear types to identify non-increasing values [8].

The overall goal of this research is to achieve a better understanding of the relationship between syntactical constructions in natural programming languages and the computational resources required to execute the programs. Some research has been conducted along these lines previously. This includes are a thesis by Caseiro [5]; papers by Lee, Jones & Ben-Amram, and Jones & Bohr [19, 11] which analyse the relationship between program syntax and program termination; and a thesis by Frederiksen [7] that contains syntactical flow analyses sufficient to recognise that a functional program to runs in polynomial time.

There are also relations to work of Kristiansen & Niggl [15, 16], and various work by Niggl (see [22] for an overview).

The recent research of Marion et al. on resource control and quasi-interpretations also seems related to the research presented in this paper, see Bonfante, Marion & Moyen [4] and Marion & Péchoux [21]. The latter's "sup-interpretations" have some similarities to our *mwp*-algebra.

We use 2-dimensional matrices to trace data flow between variables as commands are executed. The general approach is not new; Bergeretti & Carr (1985) also use matrices for data-flow analysis of essentially the same language in [3], but do not analyse value bounds. Further, size-change termination analysis [19, 11] and related methods from logic programming, transition invariants, etc., may naturally be expressed in terms of data flow matrices.

A recent paper by Niggl & Wunderlich [23] employs matrices for complexity analysis of imperative programs, but in a different way than in our calculus. The main technical distinction is that our matrices represent *mwp*-bounds whereas Niggl & Wunderlich's matrices represents plain polynomials. An effect is that for the programming language considered in the current paper, their method shows fewer programs to be polynomially bounded than our method. However, Niggl & Wunderlich deal with a richer programming language.

2 Commands and Expressions

We will consider deterministic imperative programs that manipulate natural numbers held in a fixed number of program variables $\mathtt{X}_1, \ldots, \mathtt{X}_n$. Programs may be iterative but not recursive. For simplicity we omit constants and operators other than `+` and `*`, but their treatment is straightforward.

2.1 Syntax

Expressions and *Commands* have forms given by the grammar

$$\begin{array}{rcl} \mathtt{X} \in \text{Variable} & ::= & \mathtt{X}_1 \mid \mathtt{X}_2 \mid \mathtt{X}_3 \mid \ldots \\ \mathtt{b} \in \text{Boolean exp} & ::= & \mathtt{e} = \mathtt{e}\ ,\ \mathtt{e} < \mathtt{e}\ ,\ \text{etc.} \\ \mathtt{e} \in \text{Expression} & ::= & \mathtt{X} \mid \mathtt{e\ +\ e} \mid \mathtt{e\ *\ e} \\ \mathtt{C} \in \text{Command} & ::= & \mathtt{skip} \mid \mathtt{X}\mathtt{:=}\ \mathtt{e} \mid \mathtt{C;C} \mid \mathtt{loop}\ \mathtt{X}\ \{\mathtt{C}\} \\ & \mid & \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{C}\ \mathtt{else}\ \mathtt{C} \mid \mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\mathtt{C}\} \end{array}$$

The variable `X` is not allowed to occur in the body `C` of the iteration `loop X {C}`.

Convention: We use `teletype` font for syntactic objects (variables, expressions, commands), and mathematical fonts for semantic objects, e.g., the value v currently assigned to a variable.

2.2 Semantics

A command is executed as expected from its syntax, so we omit a detailed formalisation. At any point in time each variable $\texttt{X}_i$ holds a natural number x_i (possibly 0), and the expressions are evaluated in a standard way without any side effects; the operators `*` and `+` are respectively multiplication and addition. The loop command `loop X {C}` executes the command `C` in its body m times in a row, where m is the value stored in `X` when the loop starts. The command `if b then` $\texttt{C}_1$ `else` $\texttt{C}_2$ executes the command $\texttt{C}_1$ (respectively $\texttt{C}_2$) if `b` evaluates to true (respectively false). The command $\texttt{C}_1\texttt{;C}_2$ executes first the command $\texttt{C}_1$ and then the command $\texttt{C}_2$. Commands of the form `X:= e` are ordinary assignment statements, and the command `skip` does nothing. Finally, the command `while b do {C}` executes the command `skip` if `b` evaluates to false, and the command `C; while b do {C}` if `b` evaluates to true. Hence, the semantics of our language is very standard.

Definition 1. *Let* `C` *be a command whose variables are a subset of* $\{\texttt{X}_1, \ldots, \texttt{X}_n\}$*. The command execution relation*

$$[\![\texttt{C}]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n)$$

holds iff

- *the variables* $\texttt{X}_1, \ldots, \texttt{X}_n$ *respectively hold the numbers* $x_1, \ldots, x_n$ *when the execution of* `C` *starts*
- *the execution terminates*
- *the variables* $\texttt{X}_1, \ldots, \texttt{X}_n$ *respectively hold the numbers* $x'_1, \ldots, x'_n$ *when the execution terminates.*

(So, the relation does not hold if the execution does not terminate.) Similarly, the expression evaluation relation

$$[\![\texttt{e}]\!](x_1, \ldots, x_n \rightsquigarrow v)$$

holds iff the evaluation of `e`*, when variables* $\texttt{X}_1, \ldots, \texttt{X}_n$ *respectively hold the numbers* $x_1, \ldots, x_n$*, yields numeric value* v*.* □

3 Statements and Truth

3.1 The Phenomenon being Studied: Variable Value Growth

Given a command `C`, our goal is to discover polynomially bounded data-flow relations between the *initial values* $x_1, \ldots, x_n$ of respectively $\texttt{X}_1, \ldots, \texttt{X}_n$ and the *final value* x'_i of $\texttt{X}_i$ (for $i = 1, \ldots, n$) that hold whenever $[\![\texttt{C}]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n)$.

Example 1. Consider simple commands $\texttt{C} \equiv \texttt{X}_1\texttt{:=X}_2\texttt{+X}_3$ and $\texttt{C}' \equiv \texttt{X}_1\texttt{:=X}_1\texttt{+X}_1$. The sequential compositions `C; C`, and `C; C'`, and the iteration `loop X {C}` all have polynomially bounded data-flow, e.g., $[\![\texttt{C;C}']\!](x_1, x_2, x_3 \rightsquigarrow x'_1, x'_2, x'_3)$ implies

$x_1' \leq 2x_2 + 2x_3$ and $x_2' \leq x_2$ and $x_3' \leq x_3$. On the other hand, the data flow of the command

$$\mathtt{C}'' \equiv \mathtt{X}_1\mathtt{:=1;\,loop\,X}_2\,\{\,\mathtt{X}_1\mathtt{:=X}_1\mathtt{+X}_1\,\}$$

is not polynomially bounded, since $[\![\mathtt{C}'']\!](x_1, x_2 \rightsquigarrow x_1', x_2')$ implies $x_1' = 2^{x_2}$. □

We will develop a calculus by thoroughly analysing how data might flow in computations. The observation that certain patterns of data-flow guarantee polynomial bounds on the computed values, whereas other patterns do not, leads us to the class of *mwp*-bounds and their particular form.

3.2 *mwp*-Bounds on Value Growth

Our calculus records three different types of flow: m-flow, w-flow and p-flow. In *mwp*, m stands for "maximum", w stands for "weak polynomial", and p stands for "polynomial".

An *mwp*-bound (denoted $W, V, U, \ldots$) is a number-theoretic expression of form

$$\max(\vec{x}, poly_1(\vec{y})) + poly_2(\vec{z})$$

where $\vec{x}$, $\vec{y}$, and $\vec{z}$ are disjoint lists of variables, and $poly_1$ and $poly_2$ are honest polynomials. An *honest* polynomial is a polynomial build up from constants in $\mathbb{N}$ and variables by applying the operators $+$ (addition) and $\times$ (multiplication). Note that any honest polynomial p is monotone in all its variables, i.e. we have $p(\vec{x}, y, \vec{z}) \leq p(\vec{x}, y+1, \vec{z})$ for all $\vec{x}, y, \vec{z}$.

The *m-variables* of an *mwp*-bound $W \equiv \max(\vec{x}, poly_1(\vec{y})) + poly_2(\vec{z})$ are those in $\vec{x}$; the *w-variables* of W are those in $\vec{y}$; and the *p-variables* of W are those in $\vec{z}$. The notation $W(\vec{x}; \vec{y}; \vec{z})$ displays all the variables in an *mwp*-bound W where $\vec{x}$, $\vec{y}$ and $\vec{z}$ are respectively the m-, w-and p-variables of W. Any of the three variable lists $\vec{x}$,$\vec{y}$,$\vec{z}$ might be empty, and neither the polynomial $poly_1$ nor the polynomial $poly_2$ needs to be present.

We will use *mwp*-bounds to describe bounds on variables' value growth, e.g., if $[\![\mathtt{X}_1\mathtt{:=X}_1\mathtt{+X}_2]\!](x_1, x_2 \rightsquigarrow x_1', x_2')$, then we have $x_1' \leq W(x_1;;x_2)$ where $W \equiv x_1 + x_2$. Slightly more sophisticated examples are given below.

Example 2. We have

$$[\![\mathtt{loop\,X}_3\,\{\mathtt{X}_1\mathtt{:=X}_1\mathtt{+X}_2\}]\!](x_1, x_2, x_3 \rightsquigarrow x_1', x_2', x_3') \Rightarrow$$
$$x_1' \leq W_1 \wedge x_2' \leq W_2 \wedge x_3' \leq W_3 \quad (*)$$

where $W_1(x_1;;x_2,x_3) \equiv x_1 + x_2 \cdot x_3$ and $W_2(x_2;;) \equiv x_2$ and $W_3(x_3;;) \equiv x_3$. □

Example 3. Crucially different *mwp*-bounds might be numerically equal to the same polynomial. E.g., the three *mwp*-bounds[4]

$$\begin{aligned} U(;x_1,x_2;) &\equiv \max(0, x_1 + x_2) + 0 \\ V(x_1;;x_2) &\equiv \max(x_1, 0) + x_2 \\ W(x_2;;x_1) &\equiv \max(x_2, 0) + x_1 \end{aligned}$$

[4] Occasionally we will write e.g. $U(;x_1,x_2;) \equiv \max(0, x_1 + x_2) + 0$ (and not $U(;x_1,x_2;) \equiv x_1 + x_2$) to emphasise that the list of m-variables and the list of p-variables are empty.

are all numerically equal to $x_1 + x_2$. Thus, if $[\![\mathtt{X}_1\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_2]\!](x_1, x_2 \rightsquigarrow x_1', x_2')$, we have $x_1' \leq U$, but also $x_1' \leq V$ and $x_1' \leq W$. Which *mwp*-bound to choose? In particular situations, such choices will matter for the precision of our analysis. □

Example 4. Our analysis will be able to distinguish two types of polynomial bounds: those that depend on the *iteration counts*, and those that do not. Consider the iteration

$$\mathtt{C} \equiv \mathtt{loop}\,\mathtt{X}_1\,\{\,\mathtt{X}_3\mathtt{:=}\mathtt{X}_2\mathtt{*}\mathtt{X}_2\mathtt{+}\mathtt{X}_5\mathtt{;}\ \mathtt{X}_4\mathtt{:=}\mathtt{X}_4\mathtt{+}\mathtt{X}_5\,\}$$

Every value computed by C is polynomially bounded in the input, and we have

$$[\![\mathtt{C}]\!](x_1, x_2, x_3, x_4, x_5 \rightsquigarrow x_1', x_2', x_3', x_4', x_5') \Rightarrow \\ x_3' \leq \max(x_3, 2x_2 + x_5) \ \wedge \ x_4' \leq \max(x_4, 0) + x_1 \cdot x_5$$

However, the bound on the value computed into $\mathtt{X}_3$ is *iteration-independent* as it does not depend on the value of the iteration variable $\mathtt{X}_1$; in contrast, the bound on the value computed into $\mathtt{X}_4$ is *iteration-dependent* as the bound does depend on the value of $\mathtt{X}_1$.

Note that we have written the iteration-independent bound of the form $W(x_3; x_2, x_5;) \equiv \max(x_3, 2x_2 + x_5)$, that is, as an *mwp*-bound W containing m-variables and w-variables, but no p-variables. In contrast, we have written the iteration-dependent bound as an *mwp*-bound containing p-variables. It will be seen that there is a correspondence between iteration-independence and iteration-dependence, on the one hand, and the absence and presence of p-variables on the other hand. □

3.3 *mwp*-Bounds Represented by Vectors and Matrices

When it comes to establish the *existence* of polynomial bounds it will be profitable to keep track only of certain vital information on the data flow, rather than information of more number-theoretic nature, such as coefficients and degrees of polynomials. This will lead us to a proof calculus that uses matrices in a book-keeping process, to record information on data flow.

We will now carry out an abstraction process along the lines of the tradition of program analysis by abstract interpretation, e.g. as by Cousot & Cousot [6] and explained expositorily by Jones & Nielson [12].

We abstract away the exact polynomials, but preserve the form of the *mwp*-bounds. The notation $W^{(x)}$ relates an *mwp*-bound W and a variable x:

$$W^{(x)} = \begin{cases} m & \text{if } x \text{ is an } m\text{-variable of } W \\ w & \text{if } x \text{ is an } w\text{-variable of } W \\ p & \text{if } x \text{ is an } p\text{-variable of } W \\ 0 & \text{otherwise, i.e., if } x \text{ does not occur in } W \end{cases}$$

We represent an *mwp*-bound W over the variables $x_1, \ldots, x_n$ by a column vector

$$V = \begin{pmatrix} W^{(x_1)} \\ W^{(x_2)} \\ \vdots \\ W^{(x_n)} \end{pmatrix}$$

e.g., if $n = 5$, an *mwp*-bound of the form $\max(x_5, \mathit{poly}_1(x_2, x_4)) + \mathit{poly}_2(x_1)$ is represented by the vector $\begin{pmatrix} p \\ w \\ 0 \\ w \\ m \end{pmatrix}$. An $n \times n$ matrix consists of n column vectors $(V_1, \ldots, V_n)$, and thus an $n \times n$ matrix over $\{0, m, w, p\}$ will represent a collection of n *mwp*-bounds.

Example 5. The statement (*) in Example 2 can now be expressed much more concisely by

$$\mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_1\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_2\} : \begin{pmatrix} m & 0 & 0 \\ p & m & 0 \\ p & 0 & m \end{pmatrix} \qquad (**)$$

but the exact polynomials involved are lost. The statement (**) can be read as asserting the existence of certain *mwp*-bounds: There exist *mwp*-bounds W_1, W_2, W_3 such that whenever

$$[\![\mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_1\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_2\}]\!](x_1, x_2, x_3 \rightsquigarrow x_1', x_2', x_3')$$

we have $x_i' \leq W_i$ for $i = 1, 2, 3$. Furthermore, (**) asserts that W_i has the form given by the i'th column vector, that is, $W_1(x_1; ; x_2, x_3) \equiv \max(x_1, 0) + \mathit{poly}(x_2, x_3)$, and $W_2(x_2; ;) \equiv \max(x_2, 0) + 0$, and $W_3(x_3; ;) \equiv \max(x_3, 0) + 0$.

In general, a statement of the form $\mathtt{C} : M$ can be read as asserting: For $i = 1, \ldots, n$ there exists an *mwp*-bound W_i such that, whenever $[\![\mathtt{C}]\!](\vec{x} \rightsquigarrow \vec{x}')$, we have $x_i' \leq W_i$ where W_i has the form given by the i'th column vector of M. □

3.4 Statements about Expressions and Commands

We will now formally define a *statement* and the *truth* of a statement.

Definition 2.

1. *A* statement *has the form* $\mathtt{e}{:}V$ *where* $\mathtt{e}$ *is an expression with program variables* $\mathtt{X}_1, \ldots, \mathtt{X}_n$ *and* V *is an* $n \times 1$ *column vector; or* $\mathtt{C}{:}M$ *where* $\mathtt{C}$ *is a command with program variables* $\mathtt{X}_1, \ldots, \mathtt{X}_n$ *and* M *is an* $n \times n$ *matrix.*
2. *The statement* $\mathtt{e}{:}V$ *is* true, *written* $\models \mathtt{e}{:}V$, *iff* $V = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix}$ *and there exists an mwp-bound* W *such that*

   (a) $[\![\mathtt{e}]\!](x_1, \ldots, x_n \rightsquigarrow v)$ *implies* $v \leq W$

(b) $W^{(x_i)} = \alpha_i$ *for* $i \in \{1, \ldots, n\}$.

3. *The statement* $\mathtt{C}\!:M$ *is* true*, written* $\models \mathtt{C}\!:M$*, iff there exist mwp-bounds* $W_1, \ldots, W_n$ *such that*
   (a) $[\![\mathtt{C}]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n)$ *implies* $x'_1 \leq W_1 \wedge \ldots \wedge x'_n \leq W_n$
   (b) ${W_j}^{(x_i)} = M_{ij}$ *for* $i, j \in \{1, \ldots, n\}$.

□

We call a command C *feasible* iff $\models \mathtt{C}\!:M$ for some M. Niggl & Kristiansen [15] use essentially the same concept of "feasible", but no matrices or *mwp*-bounds appear in [15].

The truth of $\mathtt{C}\!:M$ implies that any values computed by the command C will be polynomially bounded in the inputs. If a command C computes a function growing exponentially, the statement $\mathtt{C} : M$ will be false for all matrices M. For example, the value computed into $\mathtt{X}_1$ by the command $\mathtt{loop}\,\mathtt{X}_2\,\{\mathtt{X}_1\mathtt{:=X}_1\mathtt{+X}_1\}$ grows exponentially, and thus the command has no *mwp*-bounds. Hence, the statement $\mathtt{loop}\,\mathtt{X}_2\,\{\mathtt{X}_1\mathtt{:=X}_1\mathtt{+X}_1\}\!:M$ is false for any M.

4 *mwp*-Algebra and Notation

We now introduce an *mwp*-algebra and some basic vector and matrix operations. Let $\mathcal{M}$ denote the set of $(n \times n)$ matrices over the set $\{0, m, w, p\}$. We develop an algebraic structure $(\mathcal{M}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ that is a finite closed semiring. For the definition of a closed semiring and more on related algebra, see [1].

This section is quite standard and not peculiar to our problem, except for the concrete choices of the set $\mathcal{V}$ of scalars, and operations $+, \times$ on them.

4.1 The Matrix Algebra

Scalars A *scalar* is an element of $\mathcal{V} = \{0, m, w, p\}$. The elements in $\mathcal{V}$ are ordered as follows: $0 < m < w < p$. We use small Greek letters $\alpha, \beta, \gamma \ldots$ to denote the elements in $\mathcal{V}$.

The *least upper bound* of $\alpha, \beta \in \mathcal{V}$ is denoted by $\alpha + \beta$, i.e., $\alpha + \beta = \alpha$ if $\alpha \geq \beta$; otherwise $\alpha + \beta = \beta$. Let $\alpha_1, \ldots, \alpha_n$ be a sequence of values from $\mathcal{V}$, then $\sum_{i=1\ldots n} \alpha_i \stackrel{\text{def}}{=} \alpha_1 + \cdots + \alpha_n$.

The *product* of $\alpha, \beta \in \mathcal{V}$ is denoted by $\alpha \times \beta$ and defined by $\alpha \times \beta = \alpha + \beta$ if $\alpha, \beta \in \{m, w, p\}$; otherwise $\alpha \times \beta = 0$.

The algebraic structure $(\mathcal{V}, +, \times, 0, m)$ is easily verified to be a semiring.

Vectors We use $V, U, T \ldots$ to denote vectors over $\mathcal{V}$, and $V_{\downarrow i}$ denotes the i'th element in the vector V. The *least upper bound* $T \oplus U$ of the vectors T and U is defined by $V = T \oplus U$ iff $V_{\downarrow i} = T_{\downarrow i} + U_{\downarrow i}$ for $i \in \{1, \ldots, n\}$. A vector should be thought of as a column vector in an $n \times n$ matrix.

The *scalar product* αV, where α is a scalar and V is a vector, is defined by

$$\alpha \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} \alpha \times \alpha_1 \\ \vdots \\ \alpha \times \alpha_n \end{pmatrix}$$

Matrices We use $M, A, B, C, \ldots$ to denote $(n \times n)$ matrices over $\mathcal{V}$, and M_{ij} denotes the element in the i'th row and j'th column in the matrix M.

The *least upper bound* $A \oplus B$ of the matrices A and B is defined component wise, i.e., $C = A \oplus B$ iff $C_{ij} = A_{ij} + B_{ij}$ for $i, j \in \{1, \ldots, n\}$.

The matrix M is an *upper bound* the matrix A, in notation $M \geq A$, if there exists a matrix B such that $M = A \oplus B$. Thus we have a partial ordering of the universe of matrices. The ordering symbols $\geq, \leq, >, <$ have their standard meaning with respect to this ordering, and we will use standard terminology, that is, we may say that A lies above B when $A \geq B$, that A is a matrix strictly below B when $A < B$, etcetera.

The *product* $A \otimes B$ of the matrices A and B is defined by $M = A \otimes B$ iff $M_{ij} = \sum_{k=1\ldots n} A_{ik} \times B_{kj}$ (standard matrix multiplication). The *zero matrix* is denoted by $\mathbf{0}$. We define $\mathbf{0}$ by $M = \mathbf{0}$ iff $M_{ij} = 0$ for all indices i, j. We have $\mathbf{0} \oplus M = M \oplus \mathbf{0} = M$ for any matrix M. The *identity matrix* is denoted by $\mathbf{1}$. We define $\mathbf{1}$ by $M = \mathbf{1}$ iff $M_{ij} = m$ for $i = j$, and $M_{ij} = 0$ for $i \neq j$. We have $\mathbf{1} \otimes M = M \otimes \mathbf{1} = M$ for any matrix M. Further, let $M^0 = \mathbf{1}$ and $M^{n+1} = M \otimes M^n$.

The Closure Operator A unary operation on matrices, denoted $*$ and called the *closure* operator, is defined by the infinite sum

$$M^* \;=\; \mathbf{1} \,\oplus\, M \,\oplus\, M^2 \,\oplus\, M^3 \,\oplus\, \ldots$$

The closure operator is well defined in any closed semiring, and we have $M^* = 1 \oplus (M \otimes M^*)$. We will refer to the matrix M^* as the *closure of* the matrix M.

4.2 Some Notations to Manipulate Vectors and Matrices

Let M be a matrix and let V be a vector. Then $M \stackrel{k}{\leftarrow} V$ denotes the matrix obtained by replacing the k'th column vector in M by the vector V, that is, $M' = M \stackrel{k}{\leftarrow} V$ iff $M'_{ij} = V_{\downarrow i}$ if $j = k$, and $M'_{ij} = M_{ij}$ if $j \neq k$.

Example 6. If we work with (4×4)-matrices and $V = \begin{pmatrix} m \\ p \\ 0 \\ p \end{pmatrix}$, then

$$\mathbf{1} \stackrel{2}{\leftarrow} V \;=\; \begin{pmatrix} m & 0 & 0 & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & m & 0 \\ 0 & 0 & 0 & m \end{pmatrix} \stackrel{2}{\leftarrow} \begin{pmatrix} m \\ p \\ 0 \\ p \end{pmatrix} \;=\; \begin{pmatrix} m & m & 0 & 0 \\ 0 & p & 0 & 0 \\ 0 & 0 & m & 0 \\ 0 & p & 0 & m \end{pmatrix}$$

□

Our vectors and matrices will be rather sparse (most elements are 0) so we devise some more compact ways to write them in examples and proofs. The idea is to write a non-0 vector entry $V_{\downarrow i} = \alpha$ vertically as ${}^{\alpha}_{i}$, and identify the vector V with the set of all of its non-0 entries, e.g., the set $\{{}^{m}_{1}, {}^{m}_{3}, {}^{p}_{4}\}$ is identified with the vector $\begin{pmatrix} m \\ 0 \\ m \\ p \end{pmatrix}$. Along similar lines, we will say that the triplet ${}^{\alpha}_{i}\!\!\rightarrow j$ is an entry of the matrix M when $M_{ij} = \alpha \neq 0$, and we will identify a matrix M with its set of non-0 entries M_{ij}, that is, identifying M with the set $\{ {}^{\alpha}_{i}\!\!\rightarrow j \mid M_{ij} = \alpha \neq 0 \}$.

Example 7. Assume we are working with (4×4)-matrices. Then,

$$\{{}^{m}_{1}\!\!\rightarrow 1, {}^{p}_{4}\!\!\rightarrow 2, {}^{m}_{4}\!\!\rightarrow 4\} \qquad \text{and} \qquad \begin{pmatrix} m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & p & 0 & m \end{pmatrix}$$

are two alternative ways of denoting the same matrix. □

5 A Calculus for Deriving Statements

The following proof calculus allows formal derivations of true statements about programs.

5.1 Assigning Vectors to Expressions

The *variables of the expression* e, written *var*(e), is a set of natural numbers. Let $i \in var(\mathtt{e})$ iff the variable $\mathtt{X}_i$ occurs in e. We derive $\vdash \mathtt{e}\!:\!V$, for expression e and vector V, by the rules

$$(E1)\ \vdash \mathtt{X}_i\!:\!\{{}^{m}_{i}\} \qquad\qquad (E2) \vdash \mathtt{e}\!:\!\{{}^{w}_{i} \mid i \in var(\mathtt{e})\}$$

$$(E3)\ \frac{\vdash \mathtt{e}_1\!:\!V_1 \qquad \vdash \mathtt{e}_2\!:\!V_2}{\vdash \mathtt{e}_1\mathtt{+}\mathtt{e}_2\!:\!pV_1 \oplus V_2} \quad (E4)\ \frac{\vdash \mathtt{e}_1\!:\!V_1 \qquad \vdash \mathtt{e}_2\!:\!V_2}{\vdash \mathtt{e}_1\mathtt{+}\mathtt{e}_2\!:\!V_1 \oplus pV_2}$$

When $\vdash \mathtt{e}\!:\!V$, we will say that the calculus *assigns* the vector V to the expression e. Further, if $\vdash \mathtt{e}\!:\!V$ and $V_{\downarrow i} = \alpha \in \{m, w, p\}$, we will say that the variable $\mathtt{X}_i$ in the expression e is labeled α.

The rule (E1) says that if an expression is just a single variable, then this variable can be labeled m. The rule (E2) says that we always can label all the variables in an expression by w's. The calculus might assign several different vectors to the same expression, in particular, the rules (E3) and (E4) give two options for how to label the variables in an expression containing the operator +.

Example 8. We have $\vdash \mathtt{X}_1\mathtt{+}\mathtt{X}_2 : \{{}^{w}_{1}{}^{w}_{2}\}$, i.e., $\vdash \mathtt{X}_1\mathtt{+}\mathtt{X}_2 : \begin{pmatrix} w \\ w \end{pmatrix}$, by (E2). Further, we derive

$$\frac{\vdash \mathtt{X}_1\!:\!\{{}^{m}_{1}\} \qquad \vdash \mathtt{X}_2\!:\!\{{}^{m}_{2}\}}{\vdash \mathtt{X}_1\mathtt{+}\mathtt{X}_2\!:\!p\{{}^{m}_{1}\} \oplus \{{}^{m}_{2}\}}$$

by (E1) and (E3), and thus, since $p\{{}^{m}_{1}\} \oplus \{{}^{m}_{2}\} = \{{}^{pm}_{1\,2}\}$, we have $\vdash$ $\mathtt{X}_1$+$\mathtt{X}_2$: $\{{}^{pm}_{1\,2}\}$. By a symmetric derivation applying (E4), we also have $\vdash$ $\mathtt{X}_1$+$\mathtt{X}_2$: $\{{}^{mp}_{1\,2}\}$. Thus the calculus assigns (at least) three different vectors to the expression $\mathtt{X}_1$+$\mathtt{X}_2$. The reader should compare the three vectors to the three *mwp*-bounds given in Example 3. □

The net effect of the rules is that at most one variable in an expression can be labeled m, and in the case when one variable is labeled m, then all the other variables have to be labeled p.

5.2 Assigning Matrices to Commands

We derive $\vdash \mathtt{C}\!:\!M$, for command `C` and matrix M, by the inference rules

(S)

$$\vdash \mathtt{skip}\!:\!\mathbf{1}$$

(A)

$$\frac{\vdash \mathtt{e}\!:\!V}{\vdash \mathtt{X}_j\,\mathtt{:=e}\!:\!\mathbf{1} \stackrel{j}{\leftarrow} V}$$

(C)

$$\frac{\vdash \mathtt{C}_1\!:\!A \qquad \vdash \mathtt{C}_2\!:\!B}{\vdash \mathtt{C}_1\mathtt{;C}_2\!:\!A \otimes B}$$

(I)

$$\frac{\vdash \mathtt{C}_1\!:\!A \qquad\qquad \vdash \mathtt{C}_2\!:\!B}{\vdash \mathtt{if\ b\ then\ C}_1\ \mathtt{else\ C}_2\!:\!A \oplus B}$$

(L)

$$\frac{\vdash \mathtt{C}\!:\!M}{\vdash \mathtt{loop\ X}_\ell\ \{\mathtt{C}\}\!:\!M^* \oplus \{\,{}_{\ell}\!\stackrel{p}{\to} j \mid \exists\, i\,[M^*_{ij} = p]\,\}}\ (\text{if } \forall i[M^*_{ii} = m])$$

The side condition says that the closure M^* shall have nothing but m's on its diagonal. The rule is not applicable if this condition is not fulfilled.

(W)

$$\frac{\vdash \mathtt{C}\!:\!M}{\vdash \mathtt{while\ b\ do}\ \{\mathtt{C}\}\!:\!M^*}\ (\text{if } \forall i[M^*_{ii} = m] \wedge \forall ij[M^*_{ij} \neq p])$$

The side condition says that the closure M^* shall have nothing but m's on its diagonal, and further, there should be no p's at all in M^*. The rule is not applicable if this condition is not fulfilled.

Definition 3. *The relation $\vdash$ C : M holds iff there exists a derivation in the calculus where $\vdash$ C : M is the bottom line. When $\vdash$ C : M, we will say that the calculus* assigns *the matrix M to the command C. Further, we will say that a command is* derivable *if the calculus assigns at least one matrix to the command.* □

The following theorem will be proved in detail in Section 7.

Theorem 1 (Soundness). $\vdash$ *C*:M *implies* $\models$ *C*:M.

6 Explanations and Examples

We now explain the calculus and its inference rules informally and by examples.

The calculus can be seen as a book-keeping system for recording data-flow. If the calculus assigns a matrix M to a command $\mathtt{C}$, the entry M_{ij} will characterise the data-flow from the source variable $\mathtt{X}_i$ to the target variable $\mathtt{X}_j$ during the execution of $\mathtt{C}$. When $M_{ij} = 0$, a polynomial bound for the value computed into $\mathtt{X}_j$ does not depend on the initial value of $\mathtt{X}_i$; when $M_{ij} = \alpha \in \{m, w, p\}$, a polynomial bound for the value computed into $\mathtt{X}_j$ depends on the initial value of $\mathtt{X}_i$, and we will say that *data α-flows from $\mathtt{X}_i$ to $\mathtt{X}_j$*.

6.1 m-Flow

m-Flow is harmless in the sense that data flowing in a program where only m-flow occurs, will be trivially polynomially bounded. None of the input values will ever be increased, and hence, any value computed by such a program will be bounded by $\max(\vec{x})$ where $\vec{x}$ are the input values. Thus, our calculus does not impose any restrictions on the m-flow in the derivable programs, still, the calculus will keep track of the m-flow in programs. We will now study some examples illustrating the book-keeping facilities of the calculus. The examples show derivations of commands where data m-flows between the variables $\mathtt{X}_1, \mathtt{X}_2, \mathtt{X}_3, \mathtt{X}_4$.

Primitive Commands By the axiom (S) we have $\vdash \mathtt{skip} : \mathbf{1}$ where $\mathbf{1}$ is the identity matrix. The derivation reflects the fact that in a command doing nothing, there is an m-flow from each variable to itself.

The derivation

$$\frac{\vdash \mathtt{X}_2 : \begin{pmatrix} 0 \\ m \\ 0 \\ 0 \end{pmatrix}}{\vdash \mathtt{X}_1 := \mathtt{X}_2 : \begin{pmatrix} 0 & 0 & 0 & 0 \\ m & m & 0 & 0 \\ 0 & 0 & m & 0 \\ 0 & 0 & 0 & m \end{pmatrix}}$$

starts by an application of the axiom (E1) and proceeds by one application of the assignment rule (A). The derivation registers the m-flow from $\mathtt{X}_2$ to $\mathtt{X}_1$, and the m-flow from $\mathtt{X}_i$ to $\mathtt{X}_i$ for $i = 2, 3, 4$.

Sequential Composition and Matrix Multiplication In the command

$$\mathtt{X}_1 := \mathtt{X}_2 \,;\, \mathtt{X}_2 := \mathtt{X}_3 \,;\, \mathtt{X}_3 := \mathtt{X}_1$$

there is obviously m-flow from $\mathtt{X}_2$ to $\mathtt{X}_1$, and m-flow from $\mathtt{X}_3$ to $\mathtt{X}_2$. There is also m-flow from $\mathtt{X}_2$ to $\mathtt{X}_3$ since when the assignment $\mathtt{X}_3 := \mathtt{X}_1$ takes places, the initial content of $\mathtt{X}_1$ will be replaced by the initial content of $\mathtt{X}_2$. The next derivation shows how our calculus keeps track of the data flow in the program by matrix multiplication.

$$
\dfrac{
\dfrac{
\dfrac{\vdash \mathtt{X}_2 : \begin{pmatrix} 0 \\ m \\ 0 \\ 0 \end{pmatrix}}{\vdash \mathtt{X}_1 \mathtt{:=} \mathtt{X}_2 : \begin{pmatrix} 0 & 0 & 0 & 0 \\ m & m & 0 & 0 \\ 0 & 0 & m & 0 \\ 0 & 0 & 0 & m \end{pmatrix}}
\qquad
\dfrac{\vdash \mathtt{X}_3 : \begin{pmatrix} 0 \\ 0 \\ m \\ 0 \end{pmatrix}}{\vdash \mathtt{X}_2 \mathtt{:=} \mathtt{X}_3 : \begin{pmatrix} m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & m & m & 0 \\ 0 & 0 & 0 & m \end{pmatrix}}
}{\vdash \mathtt{X}_1 \mathtt{:=} \mathtt{X}_2 \mathtt{;}\ \mathtt{X}_2 \mathtt{:=} \mathtt{X}_3 : \begin{pmatrix} 0 & 0 & 0 & 0 \\ m & 0 & 0 & 0 \\ 0 & m & m & 0 \\ 0 & 0 & 0 & m \end{pmatrix}}
\qquad
\dfrac{\vdash \mathtt{X}_1 : \begin{pmatrix} m \\ 0 \\ 0 \\ 0 \end{pmatrix}}{\vdash \mathtt{X}_3 \mathtt{:=} \mathtt{X}_1 : \begin{pmatrix} m & 0 & m & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & m \end{pmatrix}}
}{\vdash \mathtt{X}_1 \mathtt{:=} \mathtt{X}_2 \mathtt{;}\ \mathtt{X}_2 \mathtt{:=} \mathtt{X}_3 \mathtt{;}\ \mathtt{X}_3 \mathtt{:=} \mathtt{X}_1 : \begin{pmatrix} 0 & 0 & 0 & 0 \\ m & 0 & m & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & 0 & m \end{pmatrix}}
$$

Conditionals and Matrix Addition The inference rule for the if-then-else construction works rather straightforwardly, and no examples should be required to convince the reader that if the matrix A keeps tracks of the m-flow in the command $\mathtt{C}_1$, and the matrix B keeps track of the m-flow in the command $\mathtt{C}_2$, then the matrix $A \oplus B$ will keep track of (give an upper bound for) the m-flow in the command $\mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{C}_1\ \mathtt{else}\ \mathtt{C}_2$.

Loops and the Closure Operator Let $\mathtt{C}^0 \equiv \mathtt{skip}$ and $\mathtt{C}^{t+1} \equiv \mathtt{C}^t \mathtt{;C}$. In general the calculus keeps track of the flow in the command $\mathtt{C}_1\mathtt{;}\ \mathtt{C}_2$ by multiplying the matrix assigned to $\mathtt{C}_1$ by the matrix assigned to $\mathtt{C}_2$. Hence, if the matrix M records the m-flow in the command $\mathtt{C}$, then the matrix $\mathbf{1}$ keeps track of the m-flow in $\mathtt{C}^0$; the matrix M^1 keeps track of the m-flow in $\mathtt{C}^1$; the matrix M^2 keeps track of the m-flow in $\mathtt{C}^2$; and so on. The closure M^* where

$$M^* \;=\; \mathbf{1} \oplus M \oplus M^2 \oplus M^3 \oplus \ldots$$

will keep track of the m-flow in the commands $\mathtt{loop}\,\mathtt{X}\,\{\mathtt{C}\}$ and $\mathtt{while}\,\mathtt{b}\,\mathtt{do}\,\{\mathtt{C}\}$.

Let us return to our example. We have assigned the matrix $M = \begin{pmatrix} 0 & 0 & 0 & 0 \\ m & 0 & m & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & 0 & m \end{pmatrix}$ to the command $\mathtt{C} \equiv \mathtt{X}_1 \mathtt{:=} \mathtt{X}_2 \mathtt{;}\ \mathtt{X}_2 \mathtt{:=} \mathtt{X}_3 \mathtt{;}\ \mathtt{X}_3 \mathtt{:=} \mathtt{X}_1$, and now we want to assign a matrix to the command $\mathtt{loop}\ \mathtt{X}_4\ \{\mathtt{C}\}$ by applying the inference rule

$$\frac{\vdash \mathtt{C} : M}{\vdash \mathtt{loop}\ \mathtt{X}_\ell\ \{\mathtt{C}\} : M^* \oplus \{\, \overset{p}{\underset{\ell}{\to}} j \mid \exists i\, [M^*_{ij} = p]\,\}}\ (\text{if } \forall i [M^*_{ii} = m]) \qquad \text{(L)}$$

We have

$$M^* \;=\; \begin{pmatrix} m & 0 & 0 & 0 \\ m & m & m & 0 \\ m & m & m & 0 \\ 0 & 0 & 0 & m \end{pmatrix}$$

There are only m's on the diagonal of M^*, and thus the side condition of the inference rule is satisfied. Further, we have

$$M^* \oplus \{\, \overset{p}{\underset{\ell}{\to}} j \mid \exists i\, [M^*_{ij} = p]\,\} \;=\; M^* \oplus \mathbf{0} \;=\; M^*$$

and thus

$$\frac{\vdash \mathtt{C} : M}{\vdash \mathtt{loop}\ \mathtt{X}_4\ \{\mathtt{C}\} : M^*}$$

is a valid instantiation of inference rule (L). Furthermore,

$$\frac{\vdash \texttt{C}: M}{\vdash \texttt{while b do } \{\texttt{C}\}: M^*}$$

is a valid instantiation of inference rule for the while-loop. It is left to the reader to check that

$$M^* = \begin{pmatrix} m & 0 & 0 & 0 \\ m & m & m & 0 \\ m & m & m & 0 \\ 0 & 0 & 0 & m \end{pmatrix}$$

actually keeps track of (gives an upper bound for) the m-flow in the two loop commands. Note that while-loop in the conclusion might very well not terminate even if the inference rule is applicable.

6.2 *w*-Flow and *p*-Flow

In contrast to m-flow, both w-flow and p-flow might be harmful in the sense that certain patterns of such flow might not be polynomially bounded. Thus, our calculus has to impose restrictions on the w- and p-flow of the derivable commands, and it turns out that it is sufficient to preclude *reflexive* w- and p-flow inside loops.

In contrast to irreflexive p-flow, irreflexive w-flow is *iteration-independent*, and by taking advantage of this nuance between p-flow and w-flow, we achieve a more complete calculus, that is, a calculus where the inference rules are not weaker than necessary. In the following we will study some examples and elaborate on w-flow, p-flow, (ir)reflexivity and iteration-(in)dependence.

Tracing the Flow The technical machinery for tracing w-flow and p-flow is of course an extension of the machinery tracing the m-flow. The ordering of the scalars $0 < m < w < p$, and the the scalar product based on this ordering, play a significant role. The product is carefully adjusted to enable the calculus to trace the data-flow in commands: In a command where data α-flows from $\texttt{X}_i$ to $\texttt{X}_j$, and then β-flows from $\texttt{X}_j$ to $\texttt{X}_\ell$, there will also be a $\alpha \times \beta$-flow from $\texttt{X}_i$ to $\texttt{X}_\ell$. Let us study an example.

In the command $\texttt{X}_2\texttt{:=}\texttt{X}_1$ data m-flows from $\texttt{X}_1$ to $\texttt{X}_2$, and in the command $\texttt{X}_3\texttt{:=}\texttt{X}_2\texttt{*}\texttt{X}_2$ data w-flows from $\texttt{X}_2$ to $\texttt{X}_3$, and thus, in $\texttt{X}_2\texttt{:=}\texttt{X}_1\texttt{;}\,\texttt{X}_3\texttt{:=}\texttt{X}_2\texttt{*}\texttt{X}_2$ data will w-flow from $\texttt{X}_1$ to $\texttt{X}_3$. The fact that $m \times w = w$ enables the calculus to trace this by matrix multiplication:

$$\cfrac{\cfrac{\vdash \texttt{X}_1 : \begin{pmatrix} m \\ 0 \\ 0 \end{pmatrix}}{\vdash \texttt{X}_2\texttt{:=}\texttt{X}_1 : \begin{pmatrix} m & m & 0 \\ 0 & 0 & 0 \\ 0 & 0 & m \end{pmatrix}} \qquad \cfrac{\vdash \texttt{X}_2\texttt{*}\texttt{X}_2 : \begin{pmatrix} 0 \\ w \\ 0 \end{pmatrix}}{\vdash \texttt{X}_3\texttt{:=}\texttt{X}_2\texttt{*}\texttt{X}_2 : \begin{pmatrix} m & 0 & 0 \\ 0 & m & w \\ 0 & 0 & 0 \end{pmatrix}}}{\vdash \texttt{X}_2\texttt{:=}\texttt{X}_1\texttt{;}\,\texttt{X}_3\texttt{:=}\texttt{X}_2\texttt{*}\texttt{X}_2 : \begin{pmatrix} m & m & w \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}$$

Reflexivity We will say that the w-flow (p-flow) in a command is reflexive when there is a w-flow (p-flow) from a variable to itself. When there is w-flow (p-flow) from a source variable $\mathtt{X}_j$ to a target variable $\mathtt{X}_i$, the data flowing might be increased, e.g., doubled. Hence, if data w-flows (p-flows) from $\mathtt{X}_i$ to $\mathtt{X}_i$ in a command $\mathtt{C}$, the content of $\mathtt{X}_i$ might be doubled by executing $\mathtt{C}$, and the data flow in the program $\mathtt{loop}\ \mathtt{X}_\ell\ \{\mathtt{C}\}$ (or the program $\mathtt{while}\ \mathtt{b}\ \{\mathtt{C}\}$) might not be polynomially bounded.

The side conditions on the loop rules prevent derivations of commands where reflexive w- and p-flow takes place inside loops. To apply any of the loop rules, it is required that the closure of the matrix in the premise has nothing but m's on the diagonal. If there are nothing but m's on the diagonal, there cannot be any w's or p's there, and thus, there will be no reflexive w-flow or p-flow when the command in the premise is executed inside a loop.

In the command $\mathtt{X}_1\mathtt{:=X}_1\mathtt{+X}_1$ there is harmful reflexive flow as data flowing from $\mathtt{X}_1$ to $\mathtt{X}_1$ might be increased during the flow. (If $x_1 \neq 0$, the data *will* be increased.) Let us study what happens when we search for a derivation of the command $\mathtt{loop}\,\mathtt{X}_2\,\{\mathtt{X}_1\mathtt{:=X}_1\mathtt{+X}_1\}$. The calculus assigns two different vectors to the expression $\mathtt{X}_1\mathtt{+X}_1$. We have $\vdash \mathtt{X}_1\mathtt{+X}_1 : \binom{w}{0}$ by (E2), and we have

$$\frac{\vdash \mathtt{X}_1 : \binom{m}{0} \qquad \vdash \mathtt{X}_1 : \binom{m}{0}}{\vdash \mathtt{X}_1\mathtt{+X}_1 : \binom{m}{0} \oplus p\binom{m}{0} = \binom{p}{0}}$$

by (E1) and (E4). These are the only two vectors the calculus assigns to the expression. (Though, the assignment $\vdash \mathtt{X}_1\mathtt{+X}_1 : \binom{p}{0}$ has several derivations.) Let us search for a derivation from both assignments. We proceed by applying the assignment rule (A), and then we try to apply the loop rule (L).

$$\frac{\dfrac{\vdash \mathtt{X}_1\mathtt{+X}_1 : \binom{w}{0}}{\vdash \mathtt{X}_1\mathtt{:=X}_1\mathtt{+X}_1 : \begin{pmatrix} w & 0 \\ 0 & m \end{pmatrix}}}{\vdash \mathtt{loop}\,\mathtt{X}_2\,\{\mathtt{X}_1\mathtt{:=X}_1\mathtt{+X}_1\} : ?} \qquad \frac{\dfrac{\vdash \mathtt{X}_1\mathtt{+X}_1 : \binom{p}{0}}{\vdash \mathtt{X}_1\mathtt{:=X}_1\mathtt{+X}_1 : \begin{pmatrix} p & 0 \\ 0 & m \end{pmatrix}}}{\vdash \mathtt{loop}\,\mathtt{X}_2\,\{\mathtt{X}_1\mathtt{:=X}_1\mathtt{+X}_1\} : ?}$$

In both cases we find that the side condition $\forall i[M^*_{ii} = m]$ for applying the rule (L) is violated, and we conclude that the command is not derivable.

Iteration-Independence We will explicate the difference between w-flow and p-flow by studying some simple examples. In the two commands $\mathtt{X}_2\mathtt{:=X}_1\mathtt{+X}_1$ and $\mathtt{X}_2\mathtt{:=X}_2\mathtt{+X}_1$ data flows from $\mathtt{X}_1$ to $\mathtt{X}_2$, and in either case the content of $\mathtt{X}_2$ might be increased.

In the first command data w-flows as the content of $\mathtt{X}_1$ *will not be* added to the old content of $\mathtt{X}_2$, whereas in the second command data p-flows as the data in $\mathtt{X}_1$ *will be* added to the old content of $\mathtt{X}_2$. What happens when we execute each of the commands k times in a row? When we study the commands

$$\underbrace{\mathtt{X}_2\mathtt{:=X}_1\mathtt{+X}_1\mathtt{;}\ \ldots\mathtt{;}\ \mathtt{X}_2\mathtt{:=X}_1\mathtt{+X}_1}_{k} \quad \text{and} \quad \underbrace{\mathtt{X}_2\mathtt{:=X}_2\mathtt{+X}_1\mathtt{;}\ \ldots\mathtt{;}\ \mathtt{X}_2\mathtt{:=X}_2\mathtt{+X}_1}_{k}$$

it is easy to see that if $k > 0$, the value flowing into $\mathtt{X}_2$ *does not* depend on k in the first case whereas it *does* in the second. The difference does matter when we the commands are executed inside loops.

In the command $\mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_2\mathtt{:=}\mathtt{X}_2\mathtt{+}\mathtt{X}_1\}$, a bound on the value flowing into $\mathtt{X}_2$ depends on the iteration count, and thus, data will p-flow from the iteration variable $\mathtt{X}_3$ to $\mathtt{X}_2$. In the command $\mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_2\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_1\}$, a bound on the value flowing into $\mathtt{X}_2$ does not depend on the iteration count, and there will be no flow at all from the iteration variable $\mathtt{X}_3$ to $\mathtt{X}_2$. (If $\mathtt{X}_3$ stores 0, the assignment $\mathtt{X}_2\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_1$ will not be executed; otherwise it will be executed. Thus, even though the value computed into $\mathtt{X}_2$ does depend on $\mathtt{X}_3$, there *exists a polynomial bound* on the value which does not.)

In general, if data p-flows from *some* source variable to a target variable inside a loop, data will will also p-flow from the loop's iteration variable into the target variable; if data does not p-flow from any variable to a target variable, there will be no flow at all from the loop's iteration variable to the target variable. The rule

$$(L)\ \frac{\vdash \mathtt{C}\!:\!M}{\vdash \mathtt{loop}\ \mathtt{X}_\ell\ \{\mathtt{C}\}\!:\!M^* \oplus \{\, \ell \overset{p}{\to} j \mid \exists\, i\, [M^*_{ij} = p]\,\}}\ (\text{if } \forall i[M^*_{ii} = m])$$

records the flow from the loop's iteration variable $\mathtt{X}_\ell$ to the variables in the loop's body by adding the matrix $\{\, \ell \overset{p}{\to} j \mid \exists\, i\, [M^*_{ij} = p]\,\}$ to the matrix M^*.

Some Derivations We will now give two derivations of $\mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_2\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_1\}$. This is the command where irreflexive w-flow, but no p-flow, takes place in the loop's body. We have

$$\dfrac{\dfrac{\dfrac{\vdash \mathtt{X}_1\!:\!\begin{pmatrix} m\\0\\0\end{pmatrix} \qquad \vdash \mathtt{X}_1\!:\!\begin{pmatrix} m\\0\\0\end{pmatrix}}{\vdash \mathtt{X}_1\mathtt{+}\mathtt{X}_1 : \begin{pmatrix} p\\0\\0\end{pmatrix}}}{\vdash \mathtt{X}_2\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_1 : \begin{pmatrix} m & p & 0\\ 0&0&0\\0&0&m\end{pmatrix}}}{\vdash \mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_2\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_1\}\!:\!\begin{pmatrix} m & p & 0\\ 0&m&0\\0&p&m\end{pmatrix}}$$

by the inference rules (E1), (E4), (A) and (L). The application of the loop rule is correct since

$$\begin{pmatrix} m & p & 0\\ 0&0&0\\0&0&m\end{pmatrix}^* \oplus \{\, \ell \overset{p}{\to} j \mid \exists\, i\, [\begin{pmatrix} m & p & 0\\ 0&0&0\\0&0&m\end{pmatrix}^*_{ij} = p]\,\} = \begin{pmatrix} m & p & 0\\ 0&m&0\\0&0&m\end{pmatrix} \oplus \begin{pmatrix} 0 & 0 & 0\\ 0&0&0\\0&p&0\end{pmatrix} = \begin{pmatrix} m & p & 0\\ 0&m&0\\0&p&m\end{pmatrix}.$$

We also have

$$\dfrac{\dfrac{\vdash \mathtt{X}_1\mathtt{+}\mathtt{X}_1 : \begin{pmatrix} w\\0\\0\end{pmatrix}}{\vdash \mathtt{X}_2\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_1 : \begin{pmatrix} m & w & 0\\ 0&0&0\\0&0&m\end{pmatrix}}}{\vdash \mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_2\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_1\} : \begin{pmatrix} m & w & 0\\ 0&m&0\\0&0&m\end{pmatrix}}$$

by (E2), (A) and (L). Now, the application of the loop rule is correct since

$$\begin{pmatrix} m & w & 0 \\ 0 & 0 & 0 \\ 0 & 0 & m \end{pmatrix}^* \oplus \{ \overset{p}{\ell} \!\!\to j \mid \exists i \, [\begin{pmatrix} m & w & 0 \\ 0 & 0 & 0 \\ 0 & 0 & m \end{pmatrix}^*_{ij} = p] \} = \begin{pmatrix} m & w & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{pmatrix} \oplus \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} m & w & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{pmatrix}.$$

Now, assume

$$[\![\texttt{loop}\, \texttt{X}_3 \, \{ \texttt{X}_2 \texttt{:=} \texttt{X}_1 \texttt{+} \texttt{X}_1 \}]\!](x_1, x_2, x_3 \rightsquigarrow x_1', x_2', x_3') \; .$$

By The Soundness Theorem the two derivations yields *mwp*-bounds for the output value x_2' in terms of the input values x_1, x_2, x_3. The second derivation yields a bound $x_2' \leq \max(x_2, \mathit{poly}(x_1))$, whereas the first one yields a bound $x_2' \leq x_2 + \mathit{poly}(x_1, x_3)$. Thus, the second derivation is the preferred one in the sense that the derivation actually records that we can find a polynomial bound on the value computed into the variable $\texttt{X}_2$ not depending on the input value of $\texttt{X}_3$.

More Derivations Let us search for derivations of $\texttt{loop}\, \texttt{X}_3 \, \{ \texttt{X}_2 \texttt{:=} \texttt{X}_2 \texttt{+} \texttt{X}_1 \}$. The body of this loop contains only *irreflexive* p-flow. If we start the derivation by applying (E2) and proceed by applying (A), we get

$$\dfrac{\dfrac{\vdash \texttt{X}_2\texttt{+}\texttt{X}_1 : \begin{pmatrix} w \\ w \\ 0 \end{pmatrix}}{\vdash \texttt{X}_2 \texttt{:=} \texttt{X}_2\texttt{+}\texttt{X}_1 : \begin{pmatrix} m & w & 0 \\ 0 & w & 0 \\ 0 & 0 & m \end{pmatrix}}}{\vdash \texttt{loop}\, \texttt{X}_3 \, \{ \texttt{X}_1 \texttt{:=} \texttt{X}_1 \texttt{+} \texttt{X}_2 \} : ?}$$

and then we are stuck. The side condition for applying the loop rule (L) is not fulfilled as the closure $\begin{pmatrix} m & w & 0 \\ 0 & w & 0 \\ 0 & 0 & m \end{pmatrix}^* = \begin{pmatrix} m & w & 0 \\ 0 & w & 0 \\ 0 & 0 & m \end{pmatrix}$ has a w on its diagonal. If we start the derivation by applying (E1), and then proceed by (E4) and (A), we get

$$\dfrac{\dfrac{\dfrac{\vdash \texttt{X}_2 : \begin{pmatrix} 0 \\ m \\ 0 \end{pmatrix} \qquad \vdash \texttt{X}_1 : \begin{pmatrix} m \\ 0 \\ 0 \end{pmatrix}}{\vdash \texttt{X}_2\texttt{+}\texttt{X}_1 : \begin{pmatrix} p \\ m \\ 0 \end{pmatrix}}}{\vdash \texttt{X}_2 \texttt{:=} \texttt{X}_1\texttt{+}\texttt{X}_2 : \begin{pmatrix} m & p & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{pmatrix}}}{\vdash \texttt{loop}\, \texttt{X}_3 \, \{ \texttt{X}_2 \texttt{:=} \texttt{X}_1 \texttt{+} \texttt{X}_1 \} : \begin{pmatrix} m & p & 0 \\ 0 & m & 0 \\ 0 & p & m \end{pmatrix}}$$

The loop rule (L) becomes applicable since the closure $\begin{pmatrix} m & p & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{pmatrix}^* = \begin{pmatrix} m & p & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{pmatrix}$ has only m's on the diagonal. Note that the rule force us to "add a p" in matrix of the conclusion. The p signifies that data p-flows from the iteration variable $\texttt{X}_3$ to the the variable $\texttt{X}_2$.

The Inference Rule for the While-Loop There exists no upper bound on the number of times the body of a while-loop might be executed. However, if there are nothing but m-flow and irreflexive w-flow, that is, iteration-independent flow, in the loop's body, the data flow will be polynomially bounded even if the loop goes on forever. This explains the side condition of the rule

$$\frac{\vdash \texttt{C}:M}{\vdash \texttt{while b do \{C\}}:M^*} \ (\text{if } \forall i[M^*_{ii} = m] \wedge \forall ij[M^*_{ij} \neq p])$$

The side condition precludes reflexive w-flow and any p-flow in the loop's body.

7 The Proofs of Soundness Properties

Our main result, i.e. Theorem 1, follows straightforwardly from the theorems proved in this section. We will prove that $\vdash \texttt{e} : V$ implies $\models \texttt{e} : V$, for any expression e and vector V. Further, for each inference rule assigning a matrix to a command, we will prove a theorem stating that if the premises of the rule are true, then the conclusion will also be true.

7.1 Properties of Honest Polynomials and *mwp*-Bounds

We use $e_1[x/e_2]$ to denote the result of simultaneously substituting the expression e_2 for each occurrence of the variable x in the expression e_1. Further, $var(e)$ denotes the set of variables in the numerical expression e, and $\Sigma\vec{x}$ denotes the sum of the variables in the list $\vec{x}$. Whenever we write a polynomial of the form $p(\vec{x})$ or $p(x_1, \ldots, x_m)$, then all of its variables are displayed, i.e., all the polynomial's variables lie in the set $\{x_1, \ldots, x_m\}$.

Lemma 1 (Monotonicity). *For any mwp-bound W, any expression* e*, any vectors V, V', any command* C*, and any matrices M, M', we have*

1. $x \leq y$ *implies* $W \leq W[x/y]$
2. $\models \texttt{e} : V$ *and* $V \leq V'$ *implies* $\models \texttt{e} : V'$.
3. $\models \texttt{C} : M$ *and* $M \leq M'$ *implies* $\models \texttt{C} : M'$.

Proof. Straight forward consequences of the form of *mwp*-bounds. Any polynomial occurring in an *mwp*-bound is honest. ◻

Lemma 1 states very basic properties of *mwp*-bounds, vectors and matrices. We will tacitly use the lemma in several pivotal proofs.

Lemma 2 (Splitting). *For any honest polynomial $p(\vec{x}, \vec{y})$ there exist honest polynomials $p_1(\vec{x})$ and $p_2(\vec{y})$ such that $p(\vec{x}, \vec{y}) \leq p_1(\vec{x}) + p_2(\vec{y})$.*

Proof. Recall that an honest polynomial is build up from constants in $\mathbb{N}$ and variables by applying the operators $+$ and $\times$. We prove the lemma by induction on the structure of an honest polynomial.

The lemma is obvious when p is a constant or a single variable. Assume $p(\vec{x}, \vec{y}) = q(\vec{x}, \vec{y}) \times r(\vec{x}, \vec{y})$ By the induction hypothesis we have honest polynomials q_1, q_2, r_1, r_2 such that $q(\vec{x}, \vec{y}) \leq q_1(\vec{x}) + q_2(\vec{y})$ and $r(\vec{x}, \vec{y}) \leq r_1(\vec{x}) + r_2(\vec{y})$. Observe that for any u, v we have

$$u \times v \ \leq\ \max(u, v)^2 \ \leq\ \max(u^2, v^2) \ \leq\ u^2 + v^2 \qquad (*)$$

Thus, we have

$$\begin{aligned} p(\vec{x},\vec{y}) &= q(\vec{x},\vec{y}) \times r(\vec{x},\vec{y}) \leq (q_1+q_2)\times(r_1+r_2) \\ &= q_1r_1 + q_1r_2 + q_2r_1 + q_2r_2 \\ &\leq q_1^2 + r_1^2 + q_1^2 + r_2^2 + q_2^2 + r_1^2 + q_2^2 + r_2^2 \qquad (*) \\ &= 2q_1^2 + 2r_1^2 + 2q_2^2 + 2r_2^2 \end{aligned}$$

and the lemma holds when $p_1 = 2q_1^2 + 2r_1^2$ and $p_2 = 2q_2^2 + 2r_2^2$.

The case when $p(\vec{x},\vec{y})$ is of the form $q(\vec{x},\vec{y}) + r(\vec{x},\vec{y})$ is easy, and we omit the details. □

Lemma 3 (Factorisation). *For any honest polynomial $p(x,\vec{y})$ there exist a fixed $k \in \mathbb{N}$ and an honest polynomial $q(\vec{y})$ such that $p(x,\vec{y}) \leq (x+2)^k q(\vec{y})$.*

Proof. This lemma is a straight forward consequence of the previous lemma. By Lemma 2 there exist honest polynomials $q'(x)$ and $q''(\vec{y})$ such that $p(x,\vec{y}) \leq q'(x) + q''(\vec{y})$. Pick k such that $q'(x) \leq (x+2)^k$ for all x. Then we have

$$p(x,\vec{y}) \leq q'(x) + q''(\vec{y}) \leq (x+2)^k + q''(\vec{y}) \leq (x+2)^k(q''(\vec{y})+1)\ .$$

Thus, the lemma holds when $q(\vec{y}) = q''(\vec{y}) + 1$. □

Lemma 4 (Regrouping). *For any honest polynomials p and q there exist honest polynomials p' and q' such that*

$$\max(\vec{x},\vec{y},\vec{z},q(\vec{y},\vec{z})) + p(\vec{z}) \leq \max(\vec{x},q'(\vec{y})) + p'(\vec{z})$$

Proof. Let $r(\vec{y},\vec{z}) = \Sigma\vec{y} + \Sigma\vec{z} + q(\vec{y},\vec{z})$. By Lemma 2 there exist honest polynomials r_1 and r_2 such that $r(\vec{y},\vec{z}) \leq r_1(\vec{y}) + r_2(\vec{z})$. The lemma holds when $q'(\vec{y}) = r_1(\vec{y})$ and $p'(\vec{z}) = r_2(\vec{z}) + p(\vec{z})$. □

7.2 Soundness of the Expression Rules and the Assignment Rule

Theorem 2. *If $\vdash \mathtt{e}:V$, then $\models \mathtt{e}:V$.*

Proof. We prove the theorem by induction over the structure of the derivation of $\vdash \mathtt{e}:V$. There are four cases.

Assume $\vdash \mathtt{e}:V$ is derived by the axiom $(E1)$ $\vdash \mathtt{X}_i : \{{}^m_i\}$. Let $W(x_i;;) = \max(x_i, 0) + 0$. Then we have $[\![\mathtt{X}_i]\!](\vec{x} \rightsquigarrow x_i)$ and $x_i \leq W$. Thus, $\models \mathtt{X}_i : \{{}^m_i\}$ holds by the definition.

Assume that we have $\vdash \mathtt{e} : \{{}^w_{i_1}, \ldots {}^w_{i_k}\}$ by an application of the axiom (E2). Now $\mathtt{e}$ is an expression and so a honest polynomial.[5] Let

$$W(;x_{i_1},\ldots,x_{i_k};) \equiv \max(0,\mathtt{e}) + 0\ .$$

Then, $[\![\mathtt{e}]\!](\vec{x} \rightsquigarrow v)$ implies $v \leq W$ as required.

[5] For convenience we are slightly informal.

Assume that (E3) is the last applied rule in the derivation $\vdash \mathtt{e} : V$, i.e., we have a derivation of the form

$$\frac{\vdash \mathtt{e}_1 : V_1 \qquad\qquad \vdash \mathtt{e}_2 : V_2}{\vdash \mathtt{e}_1\mathtt{+}\mathtt{e}_2 : pV_1 \oplus V_2}$$

By the induction hypothesis, we have $\models \mathtt{e}_1 : V_1$ and $\models \mathtt{e}_2 : V_2$, and by Lemma 1, we have $\models \mathtt{e}_1 : pV_1$. Hence, we have *mwp*-bounds W_1 and W_2 such that

$$[\![\mathtt{e}_1]\!](\vec{x} \rightsquigarrow v_1) \;\Rightarrow\; v_1 \leq W_1 \quad \text{and} \quad [\![\mathtt{e}_2]\!](\vec{x} \rightsquigarrow v_2) \;\Rightarrow\; v_2 \leq W_2$$

where W_1 and W_2 are of the forms

$$\begin{array}{lcl} W_1(;;\vec{c}) & \equiv & \max(0,0) + p(\vec{c}) \\ W_2(\vec{u};\vec{v};\vec{w}) & \equiv & \max(\vec{u}, q'(\vec{v})) + p'(\vec{w}) \; . \end{array}$$

(W_1 has only p-variables since pV_1 has only 0- and p-entries). Let

$$W'(\vec{u};\vec{v};\vec{w},\vec{c}) = \max(\vec{u}, q'(\vec{v})) + (p'(\vec{w}) + p(\vec{c})) \; .$$

Obviously, $[\![\mathtt{e}_1\mathtt{+}\mathtt{e}_2]\!](\vec{x} \rightsquigarrow v_1 + v_2) \;\Rightarrow\; v_1 + v_2 \leq W'$. Note that we have ${W'}^{(x_i)} = p$ for every $i \in \mathit{var}(\mathtt{e}_1)$, and ${W'}^{(x_i)} = {W_2}^{(x_i)}$ for every $i \in \mathit{var}(\mathtt{e}_2) \setminus \mathit{var}(\mathtt{e}_1)$. Thus, by Lemma 4 there is an *mwp*-bound W such that

- $[\![\mathtt{e}_1\mathtt{+}\mathtt{e}_2]\!](\vec{x} \rightsquigarrow v_1 + v_2) \;\Rightarrow\; v_1 + v_2 \leq W$
- $W^{(x_i)} = (pV_1 \oplus V_2)_{\downarrow i}$ for $i = 1, \ldots, n$.

This proves that $\models \mathtt{e}_1\mathtt{+}\mathtt{e}_2 : pV_1 \oplus V_2$. The case when $\vdash \mathtt{e} : V$ is derived by (E4) is symmetric to the case for (E3). □

Theorem 3. *If* $\models \mathtt{e} : V$*, then* $\models \mathtt{X}_k\mathtt{:=e} : \mathbf{1} \stackrel{k}{\leftarrow} V$.

Proof. Assume $\models \mathtt{e} : V$. By the definition of $\models$ we have an *mwp*-bound U such that (1) $[\![\mathtt{e}]\!](\vec{x} \rightsquigarrow v)$ implies $U \geq v$ and (2) $U^{(x_i)} = V_{\downarrow i}$ for $i \in \{1, \ldots, n\}$. Let $W_k = U$, and let $W_i = \max(x_i, 0) + 0$ when $i \neq k$. Then we have *mwp*-bounds $W_1, \ldots, W_n$ such that

1. $[\![\mathtt{C}]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n)$ implies $x'_1 \leq W_1 \wedge \ldots \wedge x'_n \leq W_n$
2. ${W_j}^{(x_i)} = (\mathbf{1} \stackrel{k}{\leftarrow} V)_{ij}$ for $i, j \in \{1, \ldots, n\}$.

By the definition of $\models$, we have $\models \mathtt{X}_k\mathtt{:=e} : \mathbf{1} \stackrel{k}{\leftarrow} V$. (The notation $\mathbf{1} \stackrel{k}{\leftarrow} V$ is explained in Example 6 in Section 4.2.) □

7.3 Soundness of the Composition Rule

Lemma 5 (Substitution). *For any mwp-bounds U and V there exists an mwp-bound W such that*

1. $U[x/V] \leq W$

2. $W^{(y)} = (V^{(y)} \times U^{(x)}) + U^{(y)}$ *for any* $y \in \mathrm{var}(W)$*.*

Proof. Let

$$\begin{aligned} U(\vec{a};\vec{b};\vec{c}) &= \max(\vec{a}, q(\vec{b})) + p(\vec{c}) \\ V(\vec{u};\vec{v};\vec{w}) &= \max(\vec{u}, q'(\vec{v})) + p'(\vec{w}) \ . \end{aligned}$$

We consider the four cases: (i) $U^{(x)} = 0$, (ii) $U^{(x)} = m$, (iii) $U^{(x)} = w$, (iv) $U^{(x)} = p$.

Case *(i)* $U^{(x)} = 0$. This is the case when x does not occur in U. Let $W = U$. Then (1) holds trivially, and (2) holds since

$$W^{(y)} = U^{(y)} = (V^{(y)} \times 0) + U^{(y)} = (V^{(y)} \times U^{(x)}) + U^{(y)} \ .$$

Case *(ii)* $U^{(x)} = m$. We can w.l.o.g. assume that $\vec{a} = x, \vec{a_0}$. We have

$$\begin{aligned} U[x/V] = \max(\max(\vec{u}, q'(\vec{v})) + p'(\vec{w}), \vec{a_0}, q(\vec{b})) + p(\vec{c}) \\ \leq \max(\vec{u}, \vec{a_0}, q'(\vec{v}) + q(\vec{b})) + p'(\vec{w}) + p(\vec{c}) \ . \end{aligned}$$

Let

$$W(\vec{u}, \vec{a_0}; \vec{v}, \vec{b}; \vec{w}, \vec{c}) = \max(\vec{u}, \vec{a_0}, q''(\vec{v}, \vec{b})) + p''(\vec{w}, \vec{c})$$

where $q''(\vec{v}, \vec{b}) = q'(\vec{v}) + q(\vec{b})$ and $p''(\vec{w}, \vec{c}) = p'(\vec{w}) + p(\vec{c})$. Then (1) obviously holds, and by Lemma 4 we can w.l.o.g. assume that W is a formally correct *mwp*-bound, that is, we can assume that the list $\vec{u}, \vec{a_0}$ of m-variables, the list $\vec{v}, \vec{b}$ of w-variables, and the list $\vec{w}, \vec{c}$ of p-variables, are disjoint. (If the three lists are not disjoint, we can apply the lemma and find a true *mwp*-bound W' such that $W \leq W'$.) To verify that (2) holds, we observe that

$$W^{(y)} = (V^{(y)} \times U^{(x)}) + U^{(y)} = (V^{(y)} \times m) + U^{(y)} = V^{(y)} + U^{(y)}$$

By inspecting the *mwp*-bounds, it is easily checked that we indeed have $W^{(y)} = V^{(y)} + U^{(y)}$.

Case *(iii)* $U^{(x)} = w$. We can w.l.o.g. assume that $\vec{b} = x, \vec{b_0}$. Let $r(\vec{u}, \vec{v}, \vec{w}) = \Sigma\vec{u} + q'(\vec{v}) + p'(\vec{w})$. Then we have $V \leq r(\vec{u}, \vec{v}, \vec{w})$. By Lemma 2 there exist honest polynomials r_1 and r_2 such that $q(r(\vec{u}, \vec{v}, \vec{w}), \vec{b_0}) \leq r_1(\vec{u}, \vec{v}, \vec{b_0}) + r_2(\vec{w})$. We have

$$\begin{aligned} U[x/V] = \max(\vec{a}, q(V, \vec{b_0})) + p(\vec{c}) \ &\leq \ \max(\vec{a}, q(r(\vec{u}, \vec{v}, \vec{w}), \vec{b_0})) + p(\vec{c}) \\ &\leq \ \max(\vec{a}, r_1(\vec{u}, \vec{v}, \vec{b_0})) + r_2(\vec{w}) + p(\vec{c}) \end{aligned}$$

Thus, (1) holds when we let

$$W(\vec{a}; \vec{u}, \vec{v}, \vec{b_0}; \vec{w}, \vec{c}) = \max(\vec{a}, r_1(\vec{u}, \vec{v}, \vec{b_0})) + p_1(\vec{w}, \vec{c})$$

where $p_1(\vec{w}, \vec{c}) = r_2(\vec{w}) + p(\vec{c})$, and by Lemma 4 we can assume that W is a true *mwp*-bound where the lists of m-variables, w-variables and p-variables are disjoint. To check that (2) holds, we will consider the subcases $y \notin var(V)$ and

$y \in var(V)$ separately. In the case when $y \notin var(V)$, the variable y will occur among the variables $\vec{a}$, $\vec{b_0}$ and $\vec{c}$. Further, we have

$$W^{(y)} = (V^{(y)} \times U^{(x)}) + U^{(y)} = (0 \times U^{(x)}) + U^{(y)} = U^{(y)}$$

and it is easily checked that we have $W^{(y)} = U^{(y)}$ for any y in the three lists $\vec{a}$, $\vec{b_0}$ and $\vec{c}$. In the case when $y \in var(V)$, the variable y will occur among the variables $\vec{u}$, $\vec{v}$ and $\vec{w}$. Further, we have

$$W^{(y)} = (V^{(y)} \times U^{(x)}) + U^{(y)} = (V^{(y)} \times w) + U^{(y)} \ .$$

To check that everything is all right, recall that $m \times w = w \times w = w$ and $p \times w = p$, and observe that $W^{(y)} = w$ iff y occurs in the list $\vec{u}, \vec{v}$; $W^{(y)} = p$ iff y occurs in the list $\vec{w}$.

Case *(iv)* $U^{(x)} = p$. We can w.l.o.g. assume that $\vec{c} = x, \vec{c_0}$. Let $r = \Sigma\vec{u} + q'(\vec{v}) + p'(\vec{w})$. We have

$$U[x/V] = \max(\vec{a}, q(\vec{b})) + p(V, \vec{c_0})) \leq \max(\vec{a}, q(\vec{b})) + p(r, \vec{c_0})$$

Thus, (1) holds when we let

$$W(\vec{a}; \vec{b}; \vec{u}, \vec{v}, \vec{w}, \vec{c_0}) = \max(\vec{a}, q(\vec{b})) + p''(\vec{u}, \vec{v}, \vec{w}, \vec{c_0})$$

where $p''(\vec{u}, \vec{v}, \vec{w}, \vec{c_0}) = p(r, \vec{c_0})$. By Lemma 4 we can assume that W is a true *mwp*-bound. We leave to the reader to check that (2) also holds. □

Theorem 4. *If* $\models \mathtt{C}_1 : A$ *and* $\models \mathtt{C}_2 : B$*, then* $\models \mathtt{C}_1\mathtt{;}\mathtt{C}_2 : A \otimes B$*.*

Proof. We assume $\models \mathtt{C}_1 : A$ and $\models \mathtt{C}_2 : B$. By the definitions there exist *mwp*-bounds $V_1, \ldots, V_n$ such that

– $[\![\mathtt{C}_1]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n) \ \Rightarrow \ x'_1 \leq V_1 \ \wedge \ \ldots \ \wedge \ x'_n \leq V_n$
– ${V_j}^{(x_i)} = A_{ij}$ (*)

and *mwp*-bounds $U_1, \ldots, U_n$ such that

– $[\![\mathtt{C}_2]\!](x'_1, \ldots, x'_n \rightsquigarrow x''_1, \ldots, x''_n) \ \Rightarrow \ x''_1 \leq U_1 \ \wedge \ \ldots \ \wedge \ x''_n \leq U_n$
– ${U_j}^{(x'_i)} = B_{ij}$ (**)

We will construct *mwp*-bounds $W_1, \ldots, W_n$ such that

– $[\![\mathtt{C}_1\mathtt{;}\mathtt{C}_2]\!](x_1, \ldots, x_n \rightsquigarrow x''_1, \ldots, x''_n) \ \Rightarrow \ x''_1 \leq W_1 \ \wedge \ \ldots \ \wedge \ x''_n \leq W_n$ (†)
– ${W_j}^{(x_i)} = (A \otimes B)_{ij}$ (‡)

(Then we have $\models \mathtt{C}_1\mathtt{;}\mathtt{C}_2 : A \otimes B$ by the definition.)

Fix $j \in \{1, \ldots, n\}$. We construct W_j. By using Lemma 5 once, we have an *mwp*-bound T_1 such that $U_j[x'_1/V_1] \leq T_1$ and for any $x_i \in var(T_1)$

$${T_1}^{(x_i)} = ({V_1}^{(x_i)} \times {U_j}^{(x'_1)}) + {U_j}^{(x_i)} \ .$$

Further, since x_i does not occur in U_j, we have

$$T_1{}^{(x_i)} = (V_1{}^{(x_i)} \times U_j{}^{(x'_1)}) + U_j{}^{(x_i)} = (V_1{}^{(x_i)} \times U_j{}^{(x'_1)}) + 0 = V_1{}^{(x_i)} \times U_j{}^{(x'_1)} \ .$$

By using the lemma once more, we have an *mwp*-bound T_2 such that $T_1[x'_2/V_2] \leq T_2$ and

$$T_2{}^{(x_i)} = (V_2{}^{(x_i)} \times T_1{}^{(x'_2)}) + T_1{}^{(x_i)} = (V_2{}^{(x_i)} \times U_j{}^{(x'_2)}) + T_1{}^{(x_i)} = (V_2{}^{(x_i)} \times U_j{}^{(x'_2)}) + V_1{}^{(x_i)} \times U_j{}^{(x'_1)} = \sum_{k=1,2} V_k{}^{(x_i)} \times U_j{}^{(x'_k)}$$

By using the lemma a third time, we have T_3 such that $T_2[x'_3/V_3] \leq T_3$ and

$$T_3{}^{(x_i)} = \sum_{k=1,2,3} V_k{}^{(x_i)} \times U_j{}^{(x'_k)}$$

and so on. We use the lemma n times, and we let $W_j = T_n$. Then we have

$$x''_j \leq U_j[x'_1/V_1] \dots [x'_n/V_n] \leq W_j \ .$$

This proves that (†) holds. Further we have

$$\begin{aligned} W_j{}^{(x_i)} &= \sum_{k=1,\dots,n} V_k{}^{(x_i)} \times U_j{}^{(x'_k)} \\ &= \sum_{k=1,\dots,n} A_{ik} \times B_{kj} && \text{by (*) and (**)} \\ &= (A \otimes B)_{ij} \ . && \text{by the def. of } \otimes \end{aligned}$$

Hence, $W_j{}^{(x_i)} = \alpha$ iff $(A \otimes B)_{ij} = \alpha$. This proves that (‡) holds. □

7.4 Soundness of the If-Then-Else Rule

Theorem 5. *If* $\models \mathtt{C}_1 : A$ *and* $\models \mathtt{C}_2 : B$*, then* $\models$ `if b then` $\mathtt{C}_1$ `else` $\mathtt{C}_2 : A \oplus B$*.*

Proof. We assume $\models \mathtt{C}_1 : A$ and $\models \mathtt{C}_2 : B$. By the definitions there exist *mwp*-bounds $V_1, \dots, V_n$ such that

$$[\![\mathtt{C}_1]\!](x_1, \dots, x_n \rightsquigarrow y_1, \dots, y_n) \ \Rightarrow \ y_1 \leq V_1 \wedge \dots \wedge y_n \leq V_n \quad \text{and} \quad V_j{}^{(x_i)} = A_{ij}$$

and *mwp*-bounds $U_1, \dots, U_n$ such that

$$[\![\mathtt{C}_2]\!](x_1, \dots, x_n \rightsquigarrow z_1, \dots, z_n) \ \Rightarrow \ z_1 \leq U_1 \wedge \dots \wedge z_n \leq U_n \quad \text{and} \quad U_j{}^{(x_i)} = B_{ij} \ .$$

We will construct *mwp*-bounds $W_1, \ldots, W_n$ such that

$$[\![\texttt{if b then C}_1\texttt{ else C}_2]\!](x_1,\ldots,x_n \rightsquigarrow x'_1,\ldots,x'_n) \;\Rightarrow\; x'_1 \leq W_1 \;\wedge\; \ldots \;\wedge\; x'_n \leq W_n \quad (*)$$

and ${W_j}^{(x_i)} = (A \oplus B)_{ij}$. (Then we have $\models \texttt{if b then C}_1\texttt{ else C}_2 : A \oplus B$ by the definition.)

For $i = 1, \ldots, n$, let

$$U_j \equiv \max(\vec{a}, q_j(\vec{b})) + p_j(\vec{c})$$

where $\vec{a} = \{x_i \mid A_{ij} = m\}$, $\vec{b} = \{x_i \mid A_{ij} = w\}$ and $\vec{c} = \{x_i \mid A_{ij} = p\}$, and let

$$V_j \equiv \max(\vec{u}, q'_j(\vec{v})) + p'_j(\vec{w})$$

where $\vec{u} = \{x_i \mid B_{ij} = m\}$, $\vec{v} = \{x_i \mid B_{ij} = w\}$ and $\vec{w} = \{x_i \mid B_{ij} = p\}$. By Lemma 4 there exists an *mwp*-bound W_j such that

$$\max(U_j, V_j) \;\leq\; \max(\vec{a}, \vec{u}, q_j(\vec{b}) + q'_j(\vec{v})) + p_j(\vec{c}) + p'_j(\vec{w}) \;\leq\; W_j$$

and ${W_j}^{(x_i)} = (A \oplus B)_{ij}$. Obviously, (*) holds. □

7.5 Some Examples

We will prepare the reader for the soundness proof for the loop rules by giving two examples.

Example 9. Let $\texttt{C} \equiv \texttt{if b then X}_3\texttt{:=X}_1\texttt{+X}_1\texttt{ else X}_3\texttt{:=X}_3\texttt{+X}_2$. We have

$$\vdash \texttt{C} : \begin{pmatrix} m & 0 & w & 0 \\ 0 & m & p & 0 \\ 0 & 0 & m & 0 \\ 0 & 0 & 0 & m \end{pmatrix}$$

Thus, by soundness of the calculus, we expect to find an *mwp*-bounds W_1, W_2, W_3 and W_4 such that

$$[\![\texttt{C}]\!](x_1,x_2,x_3,x_4 \rightsquigarrow x'_1,x'_2,x'_3,x'_4) \;\Rightarrow\; x'_1 \leq W_1(x_1;;) \wedge x'_2 \leq W_2(x_2;;) \;\wedge\; x'_3 \leq W_3(x_3;x_1;x_2) \;\wedge\; x'_4 \leq W_4(x_4;;)\,.$$

It is easy to find an exact expression for W_3. We have $x'_3 \leq \max(2x_1, x_3 + x_2) \leq \max(x_3, 2x_1) + x_2$. It is even easier to find concrete expressions for W_1, W_2 and W_4; we have $x'_1 \leq x_1$ and $x'_2 \leq x_2$ and $x'_4 \leq x_4$. Thus

$$[\![\texttt{C}]\!](x_1,x_2,x_3,x_4 \rightsquigarrow x'_1,x'_2,x'_3,x'_4) \;\Rightarrow\; x'_1 \leq x_1 \;\wedge\; x'_2 \leq x_2 \;\wedge\; x'_3 \leq \max(x_3, 2x_1) + x_2 \;\wedge\; x'_4 \leq x_4 \quad (1)$$

By applying the inference rule for the loop statement, we have

$$\frac{\vdash \mathtt{C}\colon \begin{pmatrix} m & 0 & w & 0 \\ 0 & m & p & 0 \\ 0 & 0 & m & 0 \\ 0 & 0 & 0 & m \end{pmatrix}}{\vdash \mathtt{loop}\,\mathtt{X}_4\,\{\mathtt{C}\}\ :\ \begin{pmatrix} m & 0 & w & 0 \\ 0 & m & p & 0 \\ 0 & 0 & m & 0 \\ 0 & 0 & p & m \end{pmatrix}}$$

Now, we expect to find an *mwp*-bound U_3 such that

$$[\![\mathtt{loop}\,\mathtt{X}_4\,\{\mathtt{C}\}]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x'_1, x'_2, x'_3, x'_4) \ \Rightarrow\ x'_3 \leq U_3(x_3; x_1; x_2, x_4)$$

We will work out a concrete expression for U_3. Let $\mathtt{C}^0 \equiv \mathtt{skip}$ and $\mathtt{C}^{t+1} \equiv \mathtt{C}^t\mathtt{;C}$. We will prove by induction on t that

$$[\![\mathtt{C}^t]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x'_1, x'_2, x'_3, x'_4) \ \Rightarrow\ x'_3 \leq \max(x_3, 2x_1) + tx_2 \tag{2}$$

It is obvious that (2) holds when $t = 0$. Now, assume that

$$[\![\mathtt{C}^t]\!](x_1, x_2, x_3, x_4 \rightsquigarrow y_1, y_2, y_3, y_4) \text{ and } [\![\mathtt{C}]\!](y_1, y_2, y_3, y_4 \rightsquigarrow x'_1, x'_2, x'_3, x'_4)\ .$$

By the induction hypothesis and (1) we have

$$\begin{aligned} x'_3 \leq \max(y_3, 2y_1) + y_2 \ &\leq\ \max(\max(x_3, 2x_1) + tx_2, 2x_1) + x_2 \\ &\leq\ \max(\max(x_3, 2x_1), 2x_1) + tx_2 + x_2 \ =\ \max(x_3, 2x_1) + (t+1)x_2 \end{aligned}$$

This proves (2). Now, it is easy to see that $\max(x_3, 2x_1) + x_4x_2$ can serve as an expression for U_3. That is,

$$[\![\mathtt{loop}\,\mathtt{X}_4\,\{\mathtt{C}\}]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x'_1, x'_2, x'_3, x'_4) \ \Rightarrow\ x'_3 \leq \max(x_3, 2x_1) + x_4x_2\ .$$

□

Example 10. Let

$$\mathtt{C} \equiv \mathtt{X}_3\mathtt{:=}\mathtt{X}_3\mathtt{+}\mathtt{X}_4\mathtt{;}\ \mathtt{X}_2\mathtt{:=}\mathtt{X}_2\mathtt{+}\mathtt{X}_3\mathtt{;}\ \mathtt{X}_1\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_2\ .$$

We obviously have

$$\begin{aligned} &[\![\mathtt{C}]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x'_1, x'_2, x'_3, x'_4) \ \Rightarrow \\ &\quad x'_1 \leq x_1 + x_2 + x_3 + x_4\,,\ x'_2 \leq x_2 + x_3 + x_4\,,\ x'_3 \leq x_3 + x_4\,,\ x'_4 \leq x_4\ . \end{aligned} \tag{1}$$

We will study the effect of executing $\mathtt{C}$ several times in a row. Let $\mathtt{C}^0 \equiv \mathtt{skip}$ and $\mathtt{C}^{t+1} \equiv \mathtt{C}^t\mathtt{;C}$. The calculus makes the assignment $\vdash \mathtt{C}\colon M$ where

$$M \ =\ \begin{pmatrix} m & 0 & 0 & 0 \\ p & m & 0 & 0 \\ p & p & m & 0 \\ p & p & p & m \end{pmatrix} \ =\ M^*$$

The closure M^* yields a matrix for the command $\mathtt{C}^t$ where t is an arbitrary but fixed number. In this particular case we have $M = M^*$. This matrix indicates that there exist *mwp*-bounds W_1, W_2, W_3, W_4 such that

$$[\![\mathtt{C}^t]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x_1', x_2', x_3', x_4') \;\Rightarrow\; x_1' \leq W_1(x_1;;x_2,x_3,x_4)\,, \\ x_2' \leq W_2(x_2;;x_3,x_4)\,,\; x_3' \leq W_3(x_3;;x_4)\,,\; x_4' \leq W_4(x_4;;)\;.$$

We will work out a concrete expression for each and one of the four *mwp*-bounds. The reader should note that before we can determine an expression for W_1 we need to determine expressions for W_2, W_3 and W_4; before we can determine an expression for W_2 we need to determine expressions for W_3 and W_4; before we can determine an expression for W_3 we need to determine expressions for W_4. That we have to determine the expressions in this particular order corresponds the p-flow given by M^*. If data p-flows, or w-flows, from $\mathtt{X}_i$ to $\mathtt{X}_j$, that is, if $M^*_{ij} \in \{w, p\}$, then we have to determine the expression for W_i before we can determine the expression for W_j. No data p-flows, or w-flows, into $\mathtt{X}_4$, and thus, we can start with W_4.

It is very easy to find a concrete expression for W_4, we have

$$[\![\mathtt{C}^t]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x_1', x_2', x_3', x_4') \;\Rightarrow\; x_4' \leq x_4\;. \tag{2}$$

Further, it should not hard to see that $x_3 + tx_4$ can serve as a concrete expression for W_3. We will prove by induction on t that we indeed have

$$[\![\mathtt{C}^t]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x_1', x_2', x_3', x_4') \;\Rightarrow\; x_3' \leq x_3 + tx_4\;. \tag{3}$$

It is obvious that (3) holds when $t = 0$. Now, assume that

$$[\![\mathtt{C}^t]\!](x_1, x_2, x_3, x_4 \rightsquigarrow y_1, y_2, y_3, y_4) \text{ and } [\![\mathtt{C}]\!](y_1, y_2, y_3, y_4 \rightsquigarrow x_1', x_2', x_3', x_4')\;.$$

By the induction hypothesis and (1) we have

$$x_3' \leq y_3 + y_4 \;\leq\; (x_3 + tx_4) + x_4 = x_3 + (t+1)x_4\;.$$

This proves (3).

Some effort is required to puzzle out a candidate for W_2, especially an expression giving a tight bound, requires some thought. We will prove

$$[\![\mathtt{C}^t]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x_1', x_2', x_3', x_4') \;\Rightarrow\; x_2' \leq x_2 + \sum_{i<t}(x_3 + ix_4 + x_4). \tag{4}$$

by induction on t.[6] The assertion is true when $t = 0$ since

$$[\![\mathtt{C}^0]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x_1', x_2', x_3', x_4') \;\Rightarrow\; x_2' \leq x_2$$

and $x_2 + \sum_{i<0}(x_3 + ix_4 + x_4) = x_2 + 0 = x_2$. To complete the induction proof, assume that

$$[\![\mathtt{C}^t]\!](x_1, x_2, x_3, x_4 \rightsquigarrow y_1, y_2, y_3, y_4) \;\text{ and }\; [\![\mathtt{C}]\!](y_1, y_2, y_3, y_4 \rightsquigarrow x_1', x_2', x_3', x_4')\,. \tag{*}$$

[6] We define $\sum_{i<t} e$ by $\sum_{i<0} e = 0$ and $\sum_{i<t+1} e = (\sum_{i<t} e) + e[i/t]$.

We have

$$
\begin{array}{rcll}
x_2' & \leq & y_2 + y_3 + y_4 & \text{by (1) and (*)} \\
& \leq & y_2 + y_3 + x_4 & \text{by (2) and (*)} \\
& \leq & y_2 + (x_3 + tx_4) + x_4 & \text{by (3) and (*)} \\
& \leq & x_2 + \sum_{i<t}(x_3 + ix_4 + x_4) + (x_3 + tx_4) + x_4 & \text{by ind. hyp. and (*)} \\
& = & x_2 + \sum_{i<t+1}(x_3 + ix_4 + x_4) \ . &
\end{array}
$$

This proves (4).

The numerical expression given in (4), that is $x_2 + \sum_{i<t}(x_3 + ix_4 + x_4)$, has the form of an *mwp*-bound since t is a fixed number, furthermore, the expression gives the tightest possible bound on x_2' since we indeed have that $x_2' = x_2 + \sum_{i<t}(x_3 + ix_4 + x_4)$ in (4). However, the expression does not look too nice, and if we proceed along the same line to determine an expression for W_1, we will end up with the outright nasty expression

$$x_1 \ + \ \sum_{j<t}\Big(\sum_{i<j}(x_3 + ix_4 + x_4) + x_3 + jx_4 + x_4\ \Big) \ .$$

To work out a more transparent expression for W_1, observe that

$$x_2 + \sum_{i<t}(x_3 + ix_4 + x_4) \quad \leq \quad x_2 + t^2(x_3 + 2x_4)$$

and prove by induction on t that

$$
\begin{array}{l}
[\![\mathtt{C}^t]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x_1', x_2', x_3', x_4') \ \Rightarrow \\
\qquad\qquad x_1' \leq x_1 + \sum_{i<t}(x_2 + i^2(x_3 + 2x_4) + x_3 + ix_4 + x_4)
\end{array} \tag{5}
$$

Further, observe that

$$
\begin{array}{l}
x_1 + \sum_{i<t}(x_2 + i^2(x_3 + 2x_4) + x_3 + ix_4 + x_4) \\
\qquad\qquad\qquad \leq \ x_1 + t^3(x_2 + 2x_3 + 4x_4)
\end{array} \tag{6}
$$

Thus, from (5) and (6) it follows that

$$x_1 + t^3(x_2 + 2x_3 + 4x_4) \tag{7}$$

can serve as a concrete expression for W_1. Note that (7) is of the form

$$\max(\vec{u}, q(\vec{y})) + t^k p(\vec{z})$$

where p and q are polynomials not depending on t or k. It is no coincidence that the bound given in Lemma 8 at page 33 is of a similar form. □

7.6 Partial orderings of $\{1, \dots, n\}$ and determination of *mwp*-bounds

Definition 4. *A binary relation $R \subseteq (\mathcal{A} \times \mathcal{A})$ is a* partial ordering[7] *of the set $\mathcal{A}$ when R is irreflexive, antisymmetric and transitive, that is, we have*

- $\neg aRa$ *(irreflexivity)*
- $a \neq b \wedge aRb \Rightarrow \neg bRa$ *(antisymmetry)*
- $aRb \wedge bRc \Rightarrow aRc$ *(transitivity)*

*for any $a, b, c \in \mathcal{A}$. If aRb holds, we will say that a is an R-*predecessor *of b, and if c has no R-predecessors, we will say that c is R-*minimal. *A partial ordering R is* well-founded *if there does not exist an infinite R-decreasing sequence, that is, and infinite sequence $a_0, a_1, a_2, \dots$ such that $a_{i+1} R a_i$ for any $i \in \mathbb{N}$.* □

Well-founded partial orderings admit of induction proofs. Let P be a property which an element of a set $\mathcal{A}$ may possess, and let us write $P(a)$ when the element a possesses the property. If R is a well-founded partial ordering of $\mathcal{A}$, then we can prove that every element of $\mathcal{A}$ possesses the property P by proving

- $P(a)$ holds when a is R-minimal (induction basis)
- if $P(b)$ holds whenever b is an R-predecessor of a, then $P(a)$ will also hold (induction step).

Lemma 6. *Let M be an $(n \times n)$ mwp-matrix such that $M^*_{ii} = m$ for all i, and let the relation $a \prec b$ hold iff $M^*_{ab} \in \{w, p\}$. Then, $\prec$ is well-founded partial ordering of the set $\{1, \dots, n\}$.*

Proof. For any $a \in \{1, \dots n\}$, we have $\neg a \prec a$ since $M^*_{ii} = m \notin \{w, p\}$, and hence, $\prec$ is irreflexive.

For any matrices A and B and any indices i, j, k, we have

$$(A \otimes B)_{ij} \quad \geq \quad A_{ik} \times B_{kj} \; . \tag{1}$$

Further, we have $M^* \otimes M^* = M^*$ (2). Assertion (1) and (2) follow from the definition of matrix multiplication and the definition of the closure operator. We will use the two assertions to prove that the relation $\prec$ is antisymmetric. Suppose that $\prec$ is not antisymmetric. Then, there exist a and b such that

$$a \neq b \;\wedge\; a \prec b \;\wedge\; b \prec a$$

and then, by the definition of $\prec$, we have $M^*_{ab} \in \{w, p\}$ and $M^*_{ba} \in \{w, p\}$. Now,

$$M^*_{ab} \times M^*_{ba} \;\stackrel{(1)}{\leq}\; (M^* \otimes M^*)_{aa} \;\stackrel{(2)}{=}\; M^*_{aa} \;=\; m$$

and hence, we have $\alpha, \beta \in \{w, p\}$ such that $\alpha \times \beta \leq m$, but according to our definition of scalar multiplication we have $\alpha \times \beta > m$ for any $\alpha, \beta \in \{w, p\}$. Thus, we have a contradiction, and we conclude that $\prec$ is antisymmetric.

[7] Also called *strict partial ordering* in the literature.

It follows from the definition of the closure operator that $\prec$ is a transitive relation. We omit the details of the argument and conclude that $\prec$ is a partial ordering of the set $\{1, \ldots, n\}$. The ordering is well-founded since the set $\{1, \ldots, n\}$ is finite. $\square$

Example 11. Let $\mathtt{C} \equiv \mathtt{X}_1\mathtt{:=X}_2\mathtt{+X}_3\mathtt{;}\ \mathtt{X}_2\mathtt{:=X}_2\mathtt{+X}_4\mathtt{;}\ \mathtt{X}_3\mathtt{:=X}_4\mathtt{+X}_4$, and let $\mathtt{C}^0 \equiv \mathtt{skip}$ and $\mathtt{C}^{t+1} \equiv \mathtt{C}^t\mathtt{;C}$. In this example, we will work out bounds W_1, W_2, W_3, W_4 such that the implication

$$[\![\mathtt{C}^t]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x'_1, x'_2, x'_3, x'_4) \;\Rightarrow\; x'_1 \leq W_1\ ,\ x'_2 \leq W_2\ ,\ x'_3 \leq W_3\ ,\ x'_4 \leq W_4\ .$$

holds. We encourage the readers to try to do this on their own before they read on.

The main point of this example is to make the reader realise that the bounds have to be worked out in a certain order. If we know the bounds W_2 and W_3, then we can determine W_1 by letting $W_1 = W_2 + W_3$; if we know W_4, then we can determine W_2 and W_3 by letting $W_2 = x_2 + tW_4$ and $W_3 = W_4 + W_4$. It does not matter which one of W_2 and W_3 we determine first, but we have to determine both W_2 and W_3 (and W_4) before we can determine W_1. Finally, the bound W_4 can be determined without knowing any of the other bounds since we have $x'_4 \leq x_4 = W_4$. (Thus, we get $W_2 = x_2 + tx_4$; $W_3 = 2x_4$; $W_1 = x_2 + (t+2)x_4$.) This particular order of computing the bounds induces a partial ordering R of the set $\{1, 2, 3, 4\}$ where 4 is the sole minimal element, where 2 and 3 are incomparable elements, etc., we have

$$R = \{(4,1), (4,2), (4,3), (3,1), (2,1)\}\ .$$

Now, let us derive the program $\mathtt{C}$ in our calculus:

$$\dfrac{\dfrac{\vdash \mathtt{X}_2\mathtt{+X}_3 : \begin{pmatrix} 0 \\ w \\ w \\ 0 \end{pmatrix}}{\vdash \mathtt{X}_1\mathtt{:=X}_2\mathtt{+X}_3 : \begin{pmatrix} 0 & 0 & 0 & 0 \\ w & m & 0 & 0 \\ w & 0 & m & 0 \\ 0 & 0 & 0 & m \end{pmatrix}} \quad \dfrac{\dfrac{\vdots}{\vdash \mathtt{X}_2\mathtt{+X}_4 : \begin{pmatrix} 0 \\ m \\ 0 \\ p \end{pmatrix}}}{\vdash \mathtt{X}_2\mathtt{:=X}_2\mathtt{+X}_4 : \begin{pmatrix} m & 0 & 0 & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & m & 0 \\ 0 & p & 0 & m \end{pmatrix}}}{\vdash \mathtt{X}_1\mathtt{:=X}_2\mathtt{+X}_3\mathtt{;}\ \mathtt{X}_2\mathtt{:=X}_2\mathtt{+X}_4 : \begin{pmatrix} 0 & 0 & 0 & 0 \\ w & m & 0 & 0 \\ w & 0 & m & 0 \\ 0 & p & 0 & m \end{pmatrix}}$$

$$\dfrac{\vdash \mathtt{X}_4\mathtt{+X}_4 : \begin{pmatrix} 0 \\ 0 \\ 0 \\ w \end{pmatrix}}{\vdash \mathtt{X}_3\mathtt{:=X}_4\mathtt{+X}_4 : \begin{pmatrix} m & 0 & 0 & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & w & m \end{pmatrix}}$$

$$\dfrac{\cdots \qquad \cdots}{\vdash \mathtt{X}_1\mathtt{:=X}_2\mathtt{+X}_3\mathtt{;}\ \mathtt{X}_2\mathtt{:=X}_2\mathtt{+X}_4\mathtt{;}\ \mathtt{X}_3\mathtt{:=X}_4\mathtt{+X}_4 : \begin{pmatrix} 0 & 0 & 0 & 0 \\ w & m & 0 & 0 \\ w & 0 & m & 0 \\ 0 & p & p & m \end{pmatrix}}$$

The closure of the matrix M at the bottom line of the derivation is

$$M^* \;=\; \begin{pmatrix} 0 & 0 & 0 & 0 \\ w & m & 0 & 0 \\ w & 0 & 0 & 0 \\ 0 & p & w & m \end{pmatrix}^* \;=\; \begin{pmatrix} m & 0 & 0 & 0 \\ w & m & 0 & 0 \\ w & 0 & m & 0 \\ p & p & w & m \end{pmatrix}.$$

Let the relation $\prec$ be defined by $a \prec b$ iff $M^*_{ab} \in \{w, p\}$. There is no coincidence that the relation $\prec$ is the same partial ordering of the set $\{1, 2, 3, 4\}$ as the relation R given above, i.e. $R = \prec$. In general, if

1. $\vdash \texttt{C}: M$
2. $M^*_{ii} = m$ for any $i \in \{1, \dots, n\}$
3. $a \prec b$ iff $M^*_{ab} \in \{w, p\}$

then, by following the order given by $\prec$, we can work out bounds $W_1, \dots, W_n$ such that

$$[\![\texttt{C}^t]\!](x_1, \dots, x_n \rightsquigarrow x'_1, \dots, x'_n) \;\Rightarrow x'_1 \leq W_1 \,,\; \dots \,,\; x'_n \leq W_n \;.$$

□

Example 12. Let $\texttt{C} \equiv \texttt{X}_1\texttt{:=}\texttt{X}_2\texttt{;}\ \texttt{X}_2\texttt{:=}\texttt{X}_3\texttt{;}\ \texttt{X}_3\texttt{:=}\texttt{X}_4\texttt{;}\ \texttt{X}_4\texttt{:=}\texttt{X}_1\texttt{;}\ \texttt{X}_5\texttt{:=}\texttt{X}_5\texttt{+}\texttt{X}_1$, and let $\texttt{C}^0 \equiv \texttt{skip}$ and $\texttt{C}^{t+1} \equiv \texttt{C}^t\texttt{;C}$. Once again, we will work out bounds W_1, W_2, W_3, W_4 such that the implication

$$[\![\texttt{C}^t]\!](x_1, x_2, x_3, x_4, x_5 \rightsquigarrow x'_1, x'_2, x'_3, x'_4, x'_5) \;\Rightarrow \\ x'_1 \leq W_1 \,,\; x'_2 \leq W_2 \,,\; x'_3 \leq W_3 \,,\; x'_4 \leq W_4 \;,x'_5 \leq W_5 \;.$$

holds, and once again, we encourage the readers to try on their own before reading on.

The program $\texttt{C}^t$ is cycling values between the variables $\texttt{X}_1$, $\texttt{X}_2$, $\texttt{X}_3$, $\texttt{X}_4$, and for each time $\texttt{C}$ is executed, one of the values will be added to the current content of $\texttt{X}_5$.

The calculus makes the assignment $\vdash \texttt{C}: M$ where

$$M \;=\; \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ m & 0 & 0 & m & p \\ 0 & m & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 \\ 0 & 0 & 0 & 0 & m \end{pmatrix}$$

We have

$$M^* \;=\; \begin{pmatrix} m & 0 & 0 & 0 & 0 \\ m & m & m & m & p \\ m & m & m & m & p \\ m & m & m & m & p \\ 0 & 0 & 0 & 0 & m \end{pmatrix}$$

and when we define the relation $\prec$ by $a \prec b$ iff $M^*_{ab} \in \{w, p\}$, we have

$$\prec \,=\; \{(2,5), (3,5), (4,5)\} \;.$$

As seen in Example 11, we can determine the bounds $W_1, \dots, W_5$ by following the ordering given by $\prec$. The element 5 has three $\prec$-predecessors, namely 2, 3 and 4. Thus, if we determine the bounds W_2, W_3, W_4, we can determine W_5. By inspecting the program, we can check that this is indeed the case since $x'_5 \leq x_5 + t\max(W_2, W_3, W_4)$. Further, there are four $\prec$-minimal elements, namely 1, 2, 3 and 4. This suggests that we can determine the bounds W_1, W_2, W_3, W_4 without knowing any of the other bounds. This might seem a bit surprising, but as e.g. 3 is $\prec$-minimal, we have

$$M^*_{i3} \neq 0 \;\;\Rightarrow M^*_{i3} = m$$

for any $i \in \{1, \dots, n\}$. (If we had $M^*_{i3} \neq 0$ and $M^*_{i3} \in \{w, p\}$, then 3 would not be $\prec$-minimal.) And thus, the value x'_3, that is, the value held by the variable

$\mathtt{X}_3$ when the program $\mathtt{C}^t$ terminates, has found its way into $\mathtt{X}_3$ by m-flowing between variables, and no value will be increased during an m-flow. Hence, we can determine W_3 by

$$x'_3 \;\leq\; \max\{x_i \mid M^*i3 = m\} \;\leq\; \max(x_2, x_3, x_4) \;=\; W_3 \ .$$

By the same token, we let $W_1 = \max(x_1, x_2, x_3, x_4)$ and $W_2 = W_4 = \max(x_2, x_3, x_4)$.

Now, we can determine W_5 since we have

$$x'_5 \leq x_5 + t\max(W_2, W_3, W_4) = x_5 + t\max(x_2, x_3, x_4) \\ \leq x_5 + t(x_2 + x_3 + x_4) = W_5(x_5; ; x_2, x_3, x_4) \ .$$

□

Lemma 7. *Let $\mathtt{C}^0 \equiv$ **skip** and $\mathtt{C}^{t+1} \equiv \mathtt{C}^t\mathtt{;C}$. Assume that $\models \mathtt{C}\colon M$ and that $M^*_{ii} = m$ for all i. Further, define the relation $\prec$ by $a \prec b$ iff $M^*_{ab} \in \{w, p\}$. By Lemma 6, $\prec$ is a well-founded partial ordering. Let j be $\prec$-minimal. Then, we have*

$$[\![\mathtt{C}^t]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n) \;\Rightarrow\; x'_j \leq \max\{x_i \mid M^*_{ij} = m\}$$

for any $t \in \mathbb{N}$.

We skip the details of the somewhat cumbersome, but indeed not difficult, proof of Lemma 7. Anyone who understands what the lemma says, is likely to be convinced that the lemma holds by studying Example 12.

7.7 Soundness of the Loop Rules

The next lemma is the key to the soundness proofs for the loop rules. The proof of the lemma is hard. The two examples of Section 7.5 motivate the form of the bound in the statement marked $(*_j)$. The examples of the previous section is meant to explicate the structure of the proof and the induction process.

Lemma 8. *Let $\mathtt{C}^0 \equiv$ **skip** and $\mathtt{C}^{t+1} \equiv \mathtt{C}^t\mathtt{;C}$. Assume that $\models \mathtt{C}\colon M$ and that $M^*_{ii} = m$ for all i. Then, for any $j \in \{1, \ldots, n\}$ there exist a fixed number k and honest polynomials p, q such that for any t we have*

$$[\![\mathtt{C}^t]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n) \;\Rightarrow\; x'_j \leq \max(\vec{u}, q(\vec{y})) + (t+2)^k p(\vec{z}) \qquad (*_j)$$

where $\vec{u} = \{x_i \mid M^*_{ij} = m\}$ and $\vec{y} = \{x_i \mid M^*_{ij} = w\}$ and $\vec{z} = \{x_i \mid M^*_{ij} = p\}$. Moreover, neither the polynomial p nor the polynomial q depends on k or t; and if the list $\vec{z}$ is empty, then $p(\vec{z}) = 0$.

Proof. First, we assume that $\models \mathtt{C} : M$ and that $M^*_{ii} = m$ for all i. Then, we define the relation $\prec$ by $a \prec b$ iff $M^*_{ab} \in \{w, p\}$. By Lemma 6, $\prec$ is a well-founded partial ordering of the set $\{1, \ldots, n\}$. We will prove that $(*_j)$ holds for any $j \in \{1, \ldots, n\}$ by induction over this ordering.

First we prove $(*_j)$ when j is $\prec$-minimal. We fix a $\prec$-minimal j. It follows straightforwardly from the definition of $\prec$ that we have $M^*_{ij} \notin \{w, p\}$ for any i, and thus, $(*_j)$ is equivalent to

$$[\![\mathtt{C}^t]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n) \;\Rightarrow\; x'_j \leq \max\{x_i \mid M^*_{ij} = m\}$$

and thus, $(*_j)$ holds by Lemma 7.

We will now turn to the induction step in the proof of $(*_j)$. By the assumption $\models \mathtt{C}\!:\!M$, we have an *mwp*-bound U_j such that

$$[\![\mathtt{C}]\!](y_1, \ldots, y_n \rightsquigarrow x'_1, \ldots, x'_n) \;\Rightarrow\; x'_j \leq U_j \;. \tag{1}$$

To avoid cluttered notation, we assume that U_j has the form

$$U_j(y_j; y_a; y_b) \equiv \max(y_j, q_0(y_a)) + p_0(y_b) \tag{2}$$

that is, we assume that a is the only number such that $M_{aj} = w$ holds; that b is the only number such that $M_{bj} = p$ holds; and that j is the only number such that $M_{jj} = m$ holds. (The proof can easily be generalised to work for any *mwp*-bound U_j.)

Now, since $M^* \geq M$, we have $M^*_{aj} \in \{w, p\}$ and $M^*_{bj} \in \{w, p\}$, and hence, we have $a \prec j$ and $b \prec j$ by the definition of $\prec$. Thus, $(*_a)$ and $(*_b)$ hold by the induction hypothesis, that is, we have

$$[\![\mathtt{C}^t]\!](x_1, \ldots, x_n \rightsquigarrow y_1, \ldots, y_n) \;\Rightarrow\; y_a \;\leq\; \max(\vec{u}_a, q_a(\vec{v}_a)) + (t+2)^{k_a} p_a(\vec{z}_a) \tag{$*_a$}$$

and

$$[\![\mathtt{C}^t]\!](x_1, \ldots, x_n \rightsquigarrow y_1, \ldots, y_n) \;\Rightarrow\; y_b \;\leq\; \max(\vec{u_b}, q_b(\vec{v_b})) + (t+2)^{k_b} p_b(\vec{z_b}) \tag{$*_b$}$$

where the polynomials q_a, q_b, p_a, p_b, the variable lists $\vec{u}_a, \vec{u}_b, \vec{v}_a, \vec{v}_b, \vec{z}_a, \vec{z}_b$ and the constants k_a, k_b have the properties stated by our lemma.

> **(Claim)** (i) For any x_i in the list $\vec{u}_a\vec{v_a}$ we have $M^*_{ij} = w$. (ii) For any x_i in the list $\vec{z_a}\vec{u}_b\vec{v}_b\vec{z}_b$ we have $M^*_{ij} = p$.

We prove the the claim. Assume that x_i occurs in the list $\vec{u}_a$. Then we have $M_{ia} = m$, and since $M^*_{aj} = w$, we get $M^*_{ij} = w$. This is a consequence of the matrix algebra and the fact that $m \times w = w$. Further, assume that x_i occurs in the list $\vec{v}_a$. Then we have $M_{ia} = w$, and since $M^*_{aj} = w$, we also also $M^*_{ij} = w$. This is a consequence of the matrix algebra and the fact that $w \times w = w$. This proves Clause (i) of the claim. The proof of (ii) is similar. This completes the proof of the claim.

Let q_1, q_2 be polynomials such that

$$q_0(\Sigma\vec{u}_a + q_a(\vec{v}_a) + (t+2)^{k_a} p_a(\vec{z}_a)) \;\leq\; q_1(\vec{u}_a, \vec{v}_a) + q_2(t, \vec{z}_a) \;.$$

The polynomials q_1 and q_2 exist by Lemma 2. To improve the readability, we let Q denote $q_1(\vec{u}_a, \vec{v}_a)$. Thus, we have

$$q_0((\Sigma \vec{u}_a) + q_a(\vec{v}_a) + (t+2)^{k_a} p_a(\vec{z}_a)) \ \leq \ Q + q_2(t, \vec{z}_a) \tag{3}$$

where Q is independent of t, and by (Claim), x_i occurs in Q iff $M^*_{ij} = w$. Let k be a number and p' be a polynomial independent of t and k such that

$$q_2(t, \vec{z}_a) + p_0((\Sigma \vec{u}_b) + q_b(\vec{v}_b) + (t+2)^{k_b} p_b(\vec{z}_b)) \ \leq \ (t+2)^k p'(\vec{z}_a, \vec{u}_b, \vec{v}_b, \vec{z}_b)$$

The polynomial p' and the number k exist by Lemma 3. To improve the readability, we let P denote $p'(\vec{z}_a, \vec{u}_b, \vec{v}_b, \vec{z}_b)$. Hence,

$$q_2(t, \vec{z}_a) + p_0((\Sigma \vec{u}_b) + q_b(\vec{v}_b) + (t+2)^{k_b} p_b(\vec{z}_b)) \ \leq \ (t+2)^k P \tag{4}$$

where P is independent of t, and by (Claim), x_i occurs in P iff $M^*_{ij} = p$.

We will prove

$$[\![\mathtt{C}^t]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n) \ \Rightarrow \ x'_j \leq \max(x_j, Q) + \sum_{\imath < t} (\imath + 2)^k P \tag{5}$$

by induction on t. The theorem follows easily from (5).

It is trivial that (5) holds when $t = 0$. Further, assume by induction hypothesis that (5) holds. We prove that

$$[\![\mathtt{C}^t ; \mathtt{C}]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n) \ \Rightarrow \ x'_j \leq \max(x_j, Q) + \sum_{\imath < t+1} (\imath + 2)^k P \ .$$

Let

$$[\![\mathtt{C}^t]\!](x_1, \ldots, x_n \rightsquigarrow y_1, \ldots, y_n) \ \text{ and } \ [\![\mathtt{C}]\!](y_1, \ldots, y_n \rightsquigarrow x'_1, \ldots, x'_n) \,. \tag{6}$$

We have

$$
\begin{aligned}
x'_j &\leq \max(y_j, q_0(y_a)) + p_0(y_b) && (1), (2) \text{ and } (6)\\
&\leq \max(y_j, q_0(\Sigma(\vec{u}_a) + q_a(\vec{v}_a) + (t+2)^{k_a} p_a(\vec{z_a}))) + p_0(y_b) && (*_a)\\
&\leq \max(y_j, Q + q_2(t, \vec{z_a})) + p_0(y_b) && (3)\\
&\leq \max(y_j, Q) + q_2(t, \vec{z_a}) + p_0(y_b) &&\\
&\leq \max(y_j, Q) + q_2(t, \vec{z_a}) &&\\
&\qquad\qquad + \; p_0((\Sigma\vec{u}_b) + q_b(\vec{v}_b) + (t+2)^{k_b} p_b(\vec{z}_b)) && (*_b)\\
&\leq \max(y_j, Q) + (t+2)^k P && (4)\\
&\leq \max(\max(x_j, Q) + \sum_{\imath<t} (\imath+2)^k P, Q) + (t+2)^k P && \text{ind. hyp. on } t\\
&= \max(x_j, Q) + (\sum_{\imath<t} (\imath+2)^k P) + (t+2)^k P &&\\
&= \max(x_j, Q) + \sum_{\imath<t+1} (\imath+2)^k P &&
\end{aligned}
$$

This proves (5), and the lemma follows since

$$\max(x_j, Q) + \sum_{\imath<t} (\imath+2)^k P \;\leq\; \max(x_j, Q) + (t+2)^{k+1} P \;.$$

By inspecting the proof, the reader can check that $(*_j)$ indeed holds with $p(\vec{z}) = 0$ whenever the list $\vec{z}$ is empty. □

Theorem 6. *If* $\models \texttt{C}\!:\!M$ *and* $M^*_{ii} = m$ *for all* i*, then*

$$\models \texttt{loop}\ \texttt{X}_\ell\ \{\texttt{C}\}\!:\!M^* \oplus \{ \overset{p}{\underset{\ell}{\to}} j \mid \exists i\, [M^*_{ij} = p] \}$$

Proof. Assume $\models \texttt{C}\!:\!M$ and $M^*_{ii} = m$, for $i = 1, \ldots, n$. We will prove that

$$\models \texttt{loop}\ \texttt{X}_\ell\ \{\texttt{C}\}\!:\!M^* \oplus \{ \overset{p}{\underset{\ell}{\to}} j \mid \exists i\, [M^*_{ij} = p] \} \;.$$

Let $M^{*+} = M^* \oplus \{ \overset{p}{\underset{\ell}{\to}} j \mid \exists i\, [M^*_{ij} = p] \}$. According to the definition of $\models$ we have to prove that for any $j \in \{1, \ldots, n\}$ there exists an *mwp*-bound W_j such that

- $[\![\texttt{loop}\ \texttt{X}_\ell\ \{\texttt{C}\}]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n) \;\Rightarrow\; x'_j \leq W_j$ (I)
- ${W_j}^{(x_i)} = M^{*+}_{ij}$, for $i = 1, \ldots, n$. (II)

By Lemma 8 there exist honest polynomials p_j, q_j and $k \in \mathbb{N}$ such that

$$[\![\texttt{loop X}_\ell\ \{\texttt{C}\}]\!](x_1, \dots, x_n \rightsquigarrow x'_1, \dots, x'_n) \ \Rightarrow\ x'_j \le \max(\vec{u}, q_j(\vec{y})) + (x_\ell + 2)^k p_j(\vec{z})$$

where $\vec{u} = \{x_i \mid M^*_{ij} = m\}$ and $\vec{y} = \{x_i \mid M^*_{ij} = w\}$ and $\vec{z} = \{x_i \mid M^*_{ij} = p\}$. Moreover, if the list $\vec{z}$ is empty, then $p_j(\vec{z}) = 0$. We split the proof two cases: (i) $\vec{z}$ is nonempty and (ii) $\vec{z}$ is empty.

Case (i). Let $W_j \equiv \max(\vec{u}, q_j(\vec{y})) + (x_\ell + 2)^k p_j(\vec{z})$. Then (I) holds. Further, when $i \neq \ell$, we have ${W_j}^{(x_i)} = M^*_{ij} = M^{*+}_{ij}$; and when $i = \ell$, we have ${W_j}^{(x_i)} = p = M^{*+}_{ij}$. Thus, (II) also holds.

Case (ii). In this case we have $p_j(\vec{z}) = 0$. Thus, let $W_j \equiv \max(\vec{u}, q_j(\vec{y}))$ and (I) holds. Further we have ${W_j}^{(x_i)} = M^*_{ij} = M^{*+}_{ij}$ for for $i = 1, \dots, n$. Thus, (II) holds. □

Theorem 7. *If* $\models \texttt{C}\!:\!M$*, and* $M^*_{ii} = m$ *for all* i*, and* $M^*_{ij} \neq p$ *for all* i, j*, then* $\models \texttt{while}\ b\ \{\texttt{C}\}\!:\!M^*$

Proof. Assume that $\models \texttt{C}\!:\!M$ where $M^*_{ii} = m$ and $M^*_{ij} \neq p$ for all i, j. Further, let $\texttt{C}^{\,0} \equiv \texttt{skip}$ and $\texttt{C}^{\,t+1} \equiv \texttt{C}^{\,t}\texttt{;C}$. It follows from Lemma 8 that for any $j \in \{1, \dots, n\}$ there exists an honest polynomial q_j such that for any $t \in \mathbb{N}$

$$[\![\texttt{C}^{\,t}]\!](x_1, \dots, x_n \rightsquigarrow x'_1, \dots, x'_n) \ \Rightarrow\ x'_j \le \max(\vec{u}, q_j(\vec{y}))$$

where $\vec{u} = \{x_i \mid M^*_{ij} = m\}$ and $\vec{y} = \{x_i \mid M^*_{ij} = w\}$. Furthermore, t does not occur in the expression $\max(\vec{u}, q(\vec{y}))$. The theorem follows. □

8 The Indeterminacy of the Calculus

The calculus is not deterministic in the sense that many different matrices might be assigned to a single command. This drastically increases the number of possible proof trees, and thus, the computational complexity of a proof search. In this section we argue that this indeterminacy indeed is necessary. We will also briefly discuss the complexity of the derivability problem.

8.1 Examples Showing the Need for Indeterminacy

To improve the readability, we introduce some additional compact matrix notation for the upcoming examples. We continue to identify a matrix M with its set of non-0 entries M_{ij}, writing $M \equiv \{\ {}^{\alpha}_{i}\!\!\rightarrow j \mid M_{ij} = \alpha \neq 0\ \}$, and we will say that the triplet ${}^{\alpha}_{i}\!\!\rightarrow j$ is an entry of the matrix M when $M_{ij} = \alpha \neq 0$. Further, we will use

$${}^{\alpha_1}_{i_1}{}^{\alpha_2}_{i_2} \dots {}^{\alpha_k}_{i_k}\!\!\rightarrow j$$

to abbreviate the entries

$${}^{\alpha_1}_{i_1}\!\!\rightarrow j\ ,\ {}^{\alpha_2}_{i_2}\!\!\rightarrow j\ ,\ \dots\ ,\ {}^{\alpha_k}_{i_k}\!\!\rightarrow j$$

and thus, ${}^{\alpha_1}_{i_1} \ldots {}^{\alpha_k}_{i_k}\!\!\to j$ denotes the j'th column vector of a matrix. Following this line, when $\Gamma \equiv {}^{\alpha_1}_{i_1} \ldots {}^{\alpha_k}_{i_k}$, we will abbreviate $\Gamma \to j_1, \ldots, \Gamma \to j_\ell$ to $\Gamma \to j_1|\ldots|j_\ell$. Hence, the matrix

$$\begin{pmatrix} m & 0 & 0 & 0 \\ m & p & p & 0 \\ m & m & m & 0 \\ 0 & 0 & 0 & m \end{pmatrix}$$

can be written $\{{}^{mmm}_{1\,2\,3}\!\!\to 1, {}^{pm}_{2\,3}\!\!\to 2|3, {}^{m}_{4}\!\!\to 4\}$. Finally, some obvious matrix entries might not be displayed, e.g., if we deal with commands over the variables $\mathtt{X}_1, \mathtt{X}_2, \mathtt{X}_3, \mathtt{X}_4$, we might write $\vdash \mathtt{X}_1 := \mathtt{X}_3 : \{{}^{m}_{3}\!\!\to 1, {}^{m}_{3}\!\!\to 3\}$ in place of

$$\vdash \mathtt{X}_1 := \mathtt{X}_3 : \{{}^{m}_{3}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{m}_{3}\!\!\to 3, {}^{m}_{4}\!\!\to 4\}\ .$$

Let us turn to the examples. The calculus assigns three different minimal matrices to the command $\mathtt{X}_3 := \mathtt{X}_1 + \mathtt{X}_2$. We have

- $\vdash \mathtt{X}_3 := \mathtt{X}_1 + \mathtt{X}_2 : \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{mp}_{1\,2}\!\!\to 3\}$
- $\vdash \mathtt{X}_3 := \mathtt{X}_1 + \mathtt{X}_2 : \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{pm}_{1\,2}\!\!\to 3\}$
- $\vdash \mathtt{X}_3 := \mathtt{X}_1 + \mathtt{X}_2 : \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{ww}_{1\,2}\!\!\to 3\}$.

The three matrices are minimal in the sense that we have $\nvdash \mathtt{X}_3 := \mathtt{X}_1 + \mathtt{X}_2 : M$ whenever $M \not\geq \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{mp}_{1\,2}\!\!\to 3\}$ and $M \not\geq \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{pm}_{1\,2}\!\!\to 3\}$ and $M \not\geq \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{ww}_{1\,2}\!\!\to 3\}$.

The three options are available since the command $\mathtt{X}_3 := \mathtt{X}_1 + \mathtt{X}_2$ might be invoked in different contexts. The command might be a subcommand of a command where

1. data is flowing from $\mathtt{X}_3$ to $\mathtt{X}_1$, but not from $\mathtt{X}_3$ to $\mathtt{X}_2$
2. data is flowing from $\mathtt{X}_3$ to $\mathtt{X}_2$, but not from $\mathtt{X}_3$ to $\mathtt{X}_1$
3. data is flowing from $\mathtt{X}_3$ to both $\mathtt{X}_2$ and $\mathtt{X}_1$
4. data is not flowing from $\mathtt{X}_3$ to $\mathtt{X}_2$ or $\mathtt{X}_1$.

We will discuss the four situations one by one.

1. Data flow from X_3 to X_1, but not from X_3 to X_2. This happens e.g., in the command

$$\mathtt{loop}\,\mathtt{X}_4\,\{\,\mathtt{X}_3 := \mathtt{X}_1 + \mathtt{X}_2\,;\,\mathtt{X}_1 := \mathtt{X}_3\} \tag{1}$$

The command is feasible and should hence be derivable. We will try to derive the command by assigning the matrix $\{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{pm}_{1\,2}\!\!\to 3\}$ to $\mathtt{X}_3 := \mathtt{X}_1 + \mathtt{X}_2$. The calculus assigns a unique minimal matrix to $\mathtt{X}_1 := \mathtt{X}_3$, namely $\{{}^{m}_{3}\!\!\to 1, {}^{m}_{3}\!\!\to 3\}$, and by applying the derivation rule for composition we have

$$\frac{\vdash \mathtt{X}_3 := \mathtt{X}_1 + \mathtt{X}_2 : \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{pm}_{1\,2}\!\!\to 3\} \qquad \vdash \mathtt{X}_1 := \mathtt{X}_3 : \{{}^{m}_{3}\!\!\to 1, {}^{m}_{3}\!\!\to 3\}}{\vdash \mathtt{X}_3 := \mathtt{X}_1 + \mathtt{X}_2\,;\,\mathtt{X}_1 := \mathtt{X}_3 : \{{}^{pm}_{1\,2}\!\!\to 1|3, {}^{m}_{2}\!\!\to 2\}}\ C$$

Now, when we try to complete the derivation by applying the inference rule

$$\frac{\vdash \mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2\,\mathtt{;}\ \mathtt{X}_1\,\mathtt{:=}\,\mathtt{X}_3 : \{{}^{pm}_{1\,2}\!\to 1|3,\ {}^{m}_{2}\!\to 2\}}{\vdash \mathtt{loop}\,\mathtt{X}_4\,\{\,\mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2\,\mathtt{;}\ \mathtt{X}_1\,\mathtt{:=}\,\mathtt{X}_3\} : ?}\ L$$

we are stuck. The rule is not admissible since ${}^{p}_{1}\!\to 1 \in \{{}^{pm}_{1\,2}\!\to 1|3,\ {}^{m}_{2}\!\to 2\}^*$.

If we try to derive the command by assigning the matrix $\{{}^{m}_{1}\!\to 1,\ {}^{m}_{2}\!\to 2,\ {}^{mp}_{1\,2}\!\to 3\}$ to $\mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2$, we will succeed. We have

$$\frac{\vdash \mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2 : \{{}^{m}_{1}\!\to 1,\ {}^{m}_{2}\!\to 2,\ {}^{mp}_{1\,2}\!\to 3\} \qquad \vdash \mathtt{X}_1\,\mathtt{:=}\,\mathtt{X}_3 : \{{}^{m}_{3}\!\to 1,\ {}^{m}_{3}\!\to 3\}}{\vdash \mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2\,\mathtt{;}\ \mathtt{X}_1\,\mathtt{:=}\,\mathtt{X}_3 : \{{}^{mp}_{1\,2}\!\to 1|3,\ {}^{m}_{2}\!\to 2\}}\ C$$

Further, we have

$$\{{}^{mp}_{1\,2}\!\to 1|3,\ {}^{m}_{2}\!\to 2\}^* \ = \ \{{}^{mp}_{1\,2}\!\to 1,\ {}^{m}_{2}\!\to 2,\ {}^{mpm}_{1\,2\,3}\!\to 3\}\ .$$

and we see that there are no entries of the form ${}^{w}_{i}\!\to i$ or the form ${}^{p}_{i}\!\to i$. Thus, the inference rule (L) is admissible, and we can complete the derivation by

$$\frac{\vdash \mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2\,\mathtt{;}\ \mathtt{X}_1\,\mathtt{:=}\,\mathtt{X}_3 : \{{}^{mp}_{1\,2}\!\to 1|3,\ {}^{m}_{2}\!\to 2\}}{\vdash \mathtt{loop}\,\mathtt{X}_4\,\{\,\mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2\,\mathtt{;}\ \mathtt{X}_1\,\mathtt{:=}\,\mathtt{X}_3\} : \{{}^{mpp}_{1\,2\,4}\!\to 1,\ {}^{m}_{2}\!\to 2,\ {}^{mpmp}_{1\,2\,3\,4}\!\to 3,\ {}^{m}_{4}\!\to 4\}}\ L$$

If we try to derive the command (1) by assigning the matrix $\{{}^{m}_{1}\!\to 1,\ {}^{m}_{2}\!\to 2,\ {}^{ww}_{1\,2}\!\to 3\}$ to $\mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2$, we will fail. The conditions for applying the rule (L) will not be fulfilled since an edge of the form ${}^{w}_{i}\!\to i$ will appear in (the closure of) the matrix assigned to $\mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2\,\mathtt{;}\ \mathtt{X}_1\,\mathtt{:=}\,\mathtt{X}_3$.

2. Data flow from X_3 to X_2, but not from X_3 to X_1. This happens e.g., in the command

$$\mathtt{loop}\,\mathtt{X}_4\,\{\,\mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2\,\mathtt{;}\ \mathtt{X}_2\,\mathtt{:=}\,\mathtt{X}_3\}\ . \tag{2}$$

The command is symmetric to the command (1). To derive the command we have to make the assignment

$$\vdash \mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2 : \{{}^{m}_{1}\!\to 1,\ {}^{m}_{2}\!\to 2,\ {}^{pm}_{1\,2}\!\to 3\}\ .$$

If we assign $\{{}^{m}_{1}\!\to 1,\ {}^{m}_{2}\!\to 2,\ {}^{mp}_{1\,2}\!\to 3\}$ or $\{{}^{m}_{1}\!\to 1,\ {}^{m}_{2}\!\to 2,\ {}^{ww}_{1\,2}\!\to 3\}$ to the subcommand $\mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2$, we will fail.

3. Data flow from X_3 to both X_1 and X_2. This happens e.g., in the command

$$\mathtt{loop}\,\mathtt{X}_4\,\{\,\mathtt{X}_3\,\mathtt{:=}\,\mathtt{X}_1\mathtt{+}\mathtt{X}_2\,\mathtt{;}\ \mathtt{X}_1\,\mathtt{:=}\,\mathtt{X}_3\,\mathtt{;}\ \mathtt{X}_2\,\mathtt{:=}\,\mathtt{X}_3\}\ . \tag{3}$$

In contrast to the commands (1) and (2), the command (3) is infeasible. Assume that the variables $\mathtt{X}_1, \mathtt{X}_2, \mathtt{X}_3, \mathtt{X}_4$ respectively hold the numbers x_1, x_2, x_3, x_4 when the execution starts. After the loop of the command is executed n times, where $n > 1$, the variable $\mathtt{X}_3$ will hold the number $2^{n-1}(x_1 + x_2)$. Thus, the number hold by $\texttt{X}_3$ when the commands terminates, is not bounded by a polynomial in x_1, x_2, x_3, x_4, and hence, the command is infeasible and should not be derivable.

Now, what will happen when we search for a derivation of this infeasible command? Let us try to find a derivation based on the assignment

$$\vdash \texttt{X}_3\,\texttt{:=}\,\texttt{X}_1\texttt{+}\texttt{X}_2 : \{{}^{m}_{1}\!\to 1, {}^{m}_{2}\!\to 2, {}^{mp}_{1\,2}\!\to 3,\} \qquad \text{(Asm 1)}$$

The calculus assigns a unique minimal matrix to the subcommands $\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3$ and $\texttt{X}_2\,\texttt{:=}\,\texttt{X}_3$, and hence we have

$$\dfrac{\dfrac{(\text{Asm 1}) \qquad \vdash \texttt{X}_1\,\texttt{:=}\,\texttt{X}_3 : \{{}^{m}_{3}\!\to 1, {}^{m}_{3}\!\to 3\}}{\vdash \texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3 : \{{}^{mp}_{1\,2}\!\to 1|3, {}^{m}_{2}\!\to 2\}}\,C \qquad \vdash \texttt{X}_2\,\texttt{:=}\,\texttt{X}_3 : \{{}^{m}_{3}\!\to 2, {}^{m}_{3}\!\to 3\}}{\dfrac{\vdash \texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3 : \{{}^{mp}_{1\,2}\!\to 1|2|3\}}{\vdash \texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3\,\texttt{;}\,\texttt{X}_2\,\texttt{:=}\,\texttt{X}_3\}:?}\,L}\,C$$

We have $\{{}^{mp}_{1\,2}\!\to 1|2|3\}^* = \{{}^{pp}_{12}\!\to 1|2, {}^{ppm}_{1\,2\,3}\!\to 3\}$ and cannot apply the rule (L) since there are several entries of the form ${}^{p}_{i}\!\to i$. If we search for a derivation based on the assignment

$$\vdash \texttt{X}_3\,\texttt{:=}\,\texttt{X}_1\texttt{+}\texttt{X}_2 : \{{}^{m}_{1}\!\to 1, {}^{m}_{2}\!\to 2, {}^{pm}_{1\,2}\!\to 3\} \qquad \text{(Asm 2)}$$

we get

$$\dfrac{\dfrac{(\text{Asm 2}) \qquad \vdash \texttt{X}_1\,\texttt{:=}\,\texttt{X}_3 : \{{}^{m}_{3}\!\to 1, {}^{m}_{3}\!\to 3\}}{\vdash \texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3 : \{{}^{pm}_{1\,2}\!\to 1|3, {}^{m}_{2}\!\to 2\}}\,C \qquad \vdash \texttt{X}_2\,\texttt{:=}\,\texttt{X}_3 : \{{}^{m}_{3}\!\to 2, {}^{m}_{3}\!\to 3\}}{\dfrac{\vdash \texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3 : \{{}^{pm}_{1\,2}\!\to 1|2|3\}}{\vdash \texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3\,\texttt{;}\,\texttt{X}_2\,\texttt{:=}\,\texttt{X}_3\}:?}\,L}\,C$$

and $\{{}^{pm}_{1\,2}\!\to 1|2|3\}^* = \{{}^{pp}_{12}\!\to 1|2, {}^{ppm}_{1\,2\,3}\!\to 3\}$. Again we find entries of the form ${}^{p}_{i}\!\to i$ in the matrix. When we resort to the last option, namely the assignment

$$\vdash \texttt{X}_3\,\texttt{:=}\,\texttt{X}_1\texttt{+}\texttt{X}_2 : \{{}^{m}_{1}\!\to 1, {}^{m}_{2}\!\to 2, {}^{ww}_{1\,2}\!\to 3\} \qquad \text{(Asm 3)}$$

we find entries of the form ${}^{w}_{i}\!\to i$ in the crucial matrix:

$$\dfrac{\dfrac{(\text{Asm 3}) \qquad \vdash \texttt{X}_1\,\texttt{:=}\,\texttt{X}_3 : \{{}^{m}_{3}\!\to 1, {}^{m}_{3}\!\to 3\}}{\vdash \texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3 : \{{}^{ww}_{1\,2}\!\to 1|3, {}^{m}_{2}\!\to 2\}}\,C \qquad \vdash \texttt{X}_2\,\texttt{:=}\,\texttt{X}_3 : \{{}^{m}_{3}\!\to 2, {}^{m}_{3}\!\to 3\}}{\dfrac{\vdash \texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3 : \{{}^{ww}_{1\,2}\!\to 1|2|3\}}{\vdash \texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3\,\texttt{;}\,\texttt{X}_2\,\texttt{:=}\,\texttt{X}_3\}:?}\,L}\,C$$

and $\{{}^{ww}_{1\,2}\!\to 1|2|3\}^* = \{{}^{ww}_{1\,2}\!\to 1|2, {}^{wwm}_{1\,2\,3}\!\to 3\}$. Thus, we see that the calculus assigns three different minimal matrices to the loop's body, that is, we have

- $\vdash \texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3\,\texttt{;}\,\texttt{X}_2\,\texttt{:=}\,\texttt{X}_3 : \{{}^{mp}_{1\,2}\!\to 1|2|3\}$
- $\vdash \texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3\,\texttt{;}\,\texttt{X}_2\,\texttt{:=}\,\texttt{X}_3 : \{{}^{pm}_{1\,2}\!\to 1|2|3\}$
- $\vdash \texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3\,\texttt{;}\,\texttt{X}_2\,\texttt{:=}\,\texttt{X}_3 : \{{}^{ww}_{1\,2}\!\to 1|2|3\}$.

In either of three cases the condition for applying the inference rule (L) prevents the derivation of the command $\texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\texttt{;}\,\texttt{X}_1\,\texttt{:=}\,\texttt{X}_3\,\texttt{;}\,\texttt{X}_2\,\texttt{:=}\,\texttt{X}_3\}$.

4. Data does not flow from X_3 to X_1 or X_2. This happens e.g., in the command

$$\texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\} \tag{4}$$

and in the command

$$\texttt{loop}\,\texttt{X}_5\,\{\,\texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\}\,\texttt{;}\,\texttt{X}_4\texttt{:=}\texttt{X}_3\,\}\;. \tag{5}$$

Note that (4) is a subcommand of (5). The command (4) turns out to be derivable not matter which of the three optional matrices we assign to $\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2$. However, (5) is derivable if and only if the matrix $\{\overset{m}{\underset{1}{\to}} 1, \overset{m}{\underset{2}{\to}} 2, \overset{ww}{\underset{1\,2}{\to}} 3\}$ is assigned to the subcommand $\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2$.

The inference

$$\frac{\vdash \texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2 : \{\overset{m}{\underset{1}{\to}} 1, \overset{m}{\underset{2}{\to}} 2, \overset{mp}{\underset{1\,2}{\to}} 3\}}{\vdash \texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\} : \{\overset{m}{\underset{1}{\to}} 1, \overset{m}{\underset{2}{\to}} 2, \overset{mpmp}{\underset{1\,2\,3\,4}{\to}} 3, \overset{m}{\underset{4}{\to}} 4\}}\; L \tag{6}$$

is admissible. We have

$$\{\overset{m}{\underset{1}{\to}} 1, \overset{m}{\underset{2}{\to}} 2, \overset{mp}{\underset{1\,2}{\to}} 3\}^* \;=\; \{\overset{m}{\underset{1}{\to}} 1, \overset{m}{\underset{2}{\to}} 2, \overset{mpm}{\underset{1\,2\,3}{\to}} 3\}\;. \tag{*}$$

Thus, for every entry $\overset{\alpha}{\underset{i}{\to}} i$ in the matrix (*), we have $\alpha = m$, and we can apply the inference rule (L) as displayed in (6). We obtain the matrix in the bottom line of (6) by adding the entry $\overset{p}{\underset{4}{\to}} 3$ to the matrix (*). The inference rule (L) force us to add this entry, and adding the entry suggests that a bound on the output value of $\texttt{X}_3$ depends on the input value of $\texttt{X}_4$. But this is not the case. By inspecting the command (4), we see that the output value of $\texttt{X}_3$ is bounded by $\max(x_3, x_1 + x_2)$ where x_1, x_2, x_3 are the input values of respectively $\texttt{X}_1, \texttt{X}_2$ and $\texttt{X}_3$. If we assign the matrix $\{\overset{m}{\underset{1}{\to}} 1, \overset{m}{\underset{2}{\to}} 2, \overset{mp}{\underset{1\,2}{\to}} 3\}$ to the subcommand $\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2$, we end up in a similar situation. We will derive

$$\vdash \texttt{loop}\,\texttt{X}_4\,\{\,\texttt{C}\,\} : \{\overset{m}{\underset{1}{\to}} 1, \overset{m}{\underset{2}{\to}} 2, \overset{pmmp}{\underset{1\,2\,3\,4}{\to}} 3, \overset{m}{\underset{4}{\to}} 4\}$$

and the matrix assigned to the command (4) suggests a superfluous dependency between $\texttt{X}_4$ and $\texttt{X}_3$.

Let us try to derive the command (5) by assigning $\{\overset{m}{\underset{1}{\to}} 1, \overset{m}{\underset{2}{\to}} 2, \overset{mp}{\underset{1\,2}{\to}} 3\}$ to $\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2$. The calculus assigns a unique minimal matrix to $\texttt{X}_4\texttt{:=}\texttt{X}_3$, i.e.,

$$\vdash \texttt{X}_4\texttt{:=}\texttt{X}_3 : \{\overset{m}{\underset{3}{\to}} 3, \overset{m}{\underset{3}{\to}} 4\}\;. \tag{7}$$

By (6), (7) and the inference rule (C), we have

$$\vdash \texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\}\,\texttt{;}\,\texttt{X}_4\texttt{:=}\texttt{X}_3 : \{\overset{m}{\underset{1}{\to}} 1, \overset{m}{\underset{2}{\to}} 2, \overset{mpmp}{\underset{1\,2\,3\,4}{\to}} 3|4\}\;. \tag{8}$$

The matrix in (8) contains the entry $\overset{p}{\underset{4}{\to}} 4$ (and hence the closure of the matrix contains at least one entry of the form $\overset{p}{\underset{i}{\to}} i$). Thus, the condition for applying the inference rule

$$\frac{\vdash \texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\}\,\texttt{;}\,\texttt{X}_4\texttt{:=}X_3 : (\overset{m}{\underset{1}{\to}} 1, \overset{m}{\underset{2}{\to}} 2, \overset{mpmp}{\underset{1\,2\,3\,4}{\to}} 3|4)}{\vdash \texttt{loop}\,\texttt{X}_5\,\{\,\texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\}\,\texttt{;}\,\texttt{X}_4\texttt{:=}\texttt{X}_3\,\} : ?}$$

is not fulfilled, and our attempt to derive the command (5) fails. The entry making the inference rule inadmissible, that is the entry ${}^{p}_{4}\!\!\to 4$, can be traced back to the superfluous dependence we were forced to introduce in the inference (6).

Let us give the one and only derivation of the command (5). We have to assign $\{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{ww}_{12}\!\!\to 3\}$ to $\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2$ and apply the inference rule

$$\frac{\vdash \texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2 : \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{ww}_{12}\!\!\to 3\}}{\vdash \texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\} : \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{wwm}_{123}\!\!\to 3, {}^{m}_{4}\!\!\to 4\}}\; L \qquad (9)$$

The inference (9) is admissible since

$$\{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{ww}_{12}\!\!\to 3\}^{*} \;=\; \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{wwm}_{123}\!\!\to 3\}$$

and thus $\alpha = m$ for every entry of the form ${}^{w}_{i}\!\!\to i$. Note that we are not required to add any p-entries when we apply the inference rule. By (9), (7) and the inference rule (C), we have

$$\vdash \texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\}\,\texttt{;}\,\texttt{X}_4\texttt{:=}\texttt{X}_3 : \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{wwm}_{123}\!\!\to 3|4\} \qquad (10)$$

and

$$\{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{wwm}_{123}\!\!\to 3|4\}^{*} = \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{wwm}_{123}\!\!\to 3, {}^{wwmm}_{1234}\!\!\to 4\}\;.$$

Thus, the closure of the matrix in (10) has nothing but m's on its diagonal, and we can complete the derivation of the command (5) by

$$\frac{\vdash \texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\}\,\texttt{;}\,\texttt{X}_4\texttt{:=}\texttt{X}_3 : \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{wwm}_{123}\!\!\to 3|4\}}{\vdash \texttt{loop}\,\texttt{X}_5\,\{\,\texttt{loop}\,\texttt{X}_4\,\{\,\texttt{X}_3\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\,\}\,\texttt{;}\,\texttt{X}_4\texttt{:=}\texttt{X}_3\,\} : \{{}^{m}_{1}\!\!\to 1, {}^{m}_{2}\!\!\to 2, {}^{wwm}_{123}\!\!\to 3, {}^{wwmm}_{1234}\!\!\to 4\}}$$

8.2 Complexity of Derivability

Theorem 8 (Complexity). *The derivability problem is in* NP.

Proof. This is by a "guess and verify" algorithm. Given a command `C`, we must decide whether $\vdash \texttt{C}\!:\!M$ for some matrix M. If there is such a matrix, it can be expressed in $O(n^2)$ bits, where n is the size of `C`.

It is clearly PTIME-decidable whether the premises and conclusion of any inference rule are satisfied. Further, all the calculus rules are compositional in their syntactic program arguments, since the matrix or vector for any command or expression is defined in terms of those of its subcommands or subexpressions.

Thus there exists a polynomial bound on the depth and total size of any proof tree that concludes $\vdash \texttt{C}\!:\!M$. A nondeterministic algorithm is thus to guess the matrices and vectors in such a tree, and check whether the inference rules of the calculus are satisfied at every node. This can be done bottom-up, beginning with a bottom node of form $\vdash \texttt{C}\!:\!\_$. □

Conjecture: The problem is NP-complete.

9 Conclusion

We have devised, proven sound, and investigated the complexity of a novel and automatable program analysis to detect polynomially bounded variables in a given input program.

References

1. A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley Publishing Co., Reading, Mass.-London-Amsterdam, 1975.
2. S.J. Bellantoni and S. Cook. *A New Recursion-Theoretic Characterization of the Polytime Functions.* Computational Complexity, 2 (1992), 97-110.
3. J.-F. Bergeretti and B.A. Carr. *Information-flow and data-flow analysis of while-programs.* ACM Trans. Program. Lang. Syst. 7 (1985), 37-61.
4. G. Bonfante, J.-Y. Marion and J.-Y. Moyen. *Quasi-interpretations – a way to control resources.* Submitted.
5. V.H. Caseiro. *Equations for Defining Poly-time Functions.* Ph.D. thesis, Dept. of Informatics, Faculty of Mathematics and Natural Sciences University of Oslo, February 1997
6. P. Cousot and R. Cousot. *Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints.* In “Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages”, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, NY, USA.
7. C.C. Frederiksen. *Automatic runtime analysis for first order functional programs.* Master Thesis, Dept. of Computer Science, University of Copenhagen, 2002.
8. M. Hofmann. *Linear Types and Non Size-Increasing Polynomial Time Computation.* Information and Computation 183 (2003), 57-85.
9. N.D. Jones. *The expressive power of higher-order types or, life without CONS.* J. Functional Programming 11 (2001), 55-94.
10. N.D. Jones. *LOGSPACE and PTIME characterized by programming languages.* Theoretical Computer Science 228 (1999), 151-174.
11. N.D. Jones and N. Bohr. *Termination analysis of the untyped lambda-calculus.* Rewriting Techniques and Applications, Springer LNCS Volume 3091 (2004), 1-23.
12. N.D. Jones and F. Nielson. *Abstract interpretation: a semantics-based tool for program analysis.* In “Handbook of Logic in Computer Science”, Oxford University Press, 1994, 527-629.
13. L. Kristiansen. *Neat function algebraic characterizations of LOGSPACE and LINSPACE.* Computational Complexity 14 (2005), 72-88.
14. L. Kristiansen. *Complexity-theoretic hierarchies induced by fragments of Gödel’s T.* Theory of Computing Systems (accepted).
15. L. Kristiansen and K.-H. Niggl. *On the computational complexity of imperative programming languages.* Theoretical Computer Science 318 (2004), 139-161.
16. L. Kristiansen and K.-H. Niggl. *The Garland Measure and Computational Complexity of Stack Programs.* Electronic Notes in Theoretical Computer Science, Volume 90, Elsevier 2003.
17. L. Kristiansen and P.J. Voda. *Complexity classes and fragments of C.* Information Processing Letters 88 (2003), 213-218.

18. L. Kristiansen and P.J. Voda. *Programming languages capturing complexity classes.* Nordic Journal of Computing 12 (2005), 89-115.
19. C.S. Lee, N. Jones, and A.M. Ben-Amram. *The size-change principle for program termination.* ACM Principles of Programming Languages ACM Press 2001, 81-92.
20. D. Leivant. *Intrinsic theories and computational complexity.* In “Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC ’94”, Leivant (ed.), 177-194, Springer, 1995.
21. J.Y. Marion and R. Péchoux *Resource analysis by sup-interpretations.* in M. Hagiya and P. Wadler editors, Functional and LOgic Programming 8th international Symposium (FLOPS 2006), LNCS 3945, April 2006.
22. K.-H. Niggl. *Control Structures in Programs and Computational Complexity.* Ann. Pure Appl. Logic 133 (2005), 247–273.
23. K.-H. Niggl and H. Wunderlich. *Certifying polynomial time and linear/polynomial space for imperative programs.* SIAM J. Comput. 35 (2006), 1122–1147.
24. H. Simmons. *The realm of primitive recursion.* Arch. Math. Logic 27 (1988), 177-188.