Applying Gauss elimination from boolean equation systems to simple integer equation systems

Kevin van der Pol

Eindhoven University of Technology
January 19, 2012

Abstract

We define a direct transformation from boolean to integer equation systems and prove that it is sound. We then apply the Gauss elimination technique for solving boolean equation systems to the world of integer equation systems, with partial success. The technique consists of two parts: local resolution and substitution. We have explored two techniques for local resolution: a reduction technique removing one occurrence of a variable at a time, and Kleene chain iteration removing all occurrences of a variable at once. For the reduction technique, we proved it is sound for finite solutions. If we can determine whether some subexpression solves to negative infinity, we can generalize this technique to infinite solutions. For the iterative technique, we have proven that is is correct but not that the iteration necessarily terminates. We proved that substitution is sound, without any restrictions on the order of the equations.

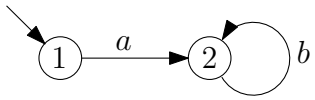
Contents

1 Introduction

In model checking, we aim to verify a property on a system. This property usually expresses that something bad can never happen in the system or that something good eventually happens. The property can be given in a number of formalisms, for example in the modal μ-calculus [6]. The system may also be expressed in several ways, for example as a labeled transition system. The model checking problem is to answer the question: does the system satisfy the property?

It is known that this problem can be expressed as a boolean equation system[7]. A boolean equation system is a list of equations on boolean variables. One of these variables then holds the answer to our original problem.

Example 1. For example, the question whether the system



satisfies the μ-calculus formula[1]

$$\nu\mathbf{x}.([\mathsf{true}]\mathbf{x} \wedge \mu\mathbf{y}.(\langle\mathsf{true}\rangle\mathbf{y} \vee \langle a\rangle\mathsf{true})),$$

can be expressed as the following boolean equation system, where the variable $\mathbf{x}$ holds the answer to our problem:

$$(\nu\mathbf{x}_1 = \mathbf{x}_2 \wedge \mathbf{y}_1)\ (\nu\mathbf{x}_2 = \mathbf{x}_2 \wedge \mathbf{y}_2)\ (\mu\mathbf{y}_1 = \mathbf{y}_2 \vee \mathsf{true})\ (\mu\mathbf{y}_2 = \mathbf{y}_2 \vee \mathsf{false})$$

It was shown by Keinänen [3] that these boolean equation systems can be transformed into a parity game. A parity game is a game with two players, played on a graph structure, which represents the original problem in some way. Which of these players wins the game from what starting position, assuming perfect strategy from both players, corresponds to the answer to our original problem.

Furthermore, these parity games can be transformed into yet another formalism, called integer equation systems [2]. This is again a list of equations, only on integer variables. The answer to our original problem now corresponds to the sign of these variables, when we solve the system. These integer equation systems resemble the boolean equation systems quite much, so in this paper we explore how they are related. We give a direct translation from boolean to integer equation systems and take a technique of solving the boolean equation systems and apply it on the integer equation systems.

We start in section 2 with some preliminaries. Then, in sections 3 and 4 we introduce boolean and integer equation systems and in section 5 we define a transformation from boolean to integer equation systems. In the remaining sections, we explore Gauss elimination and apply it to integer equation systems. This starts in section 6 by explaining what Gauss elimination is and how it works on boolean equation systems. It then continues to define a similar procedure on integer equation systems. In section 6.3.1 the reduction method is explored, which removes variables from their own defining expressions. In section 6.3.5 we try a different approach to do the same thing, using iteration. Finally, in section 6.3.6 we prove soundness of substituting a variable by its defining expression, finishing the Gauss elimination procedure on integer equation systems.

[1] The meaning of this formula is not important for this paper.

2 Preliminaries

2.1 Extended integers

We extend the set of integers $\mathbb{Z}$ with the elements $-\infty$ and ∞ to create the set of extended integers: $\overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, \infty\}$. We extend the operations addition $+ : \overline{\mathbb{Z}} \times \overline{\mathbb{Z}} \to \overline{\mathbb{Z}}$ and multiplication $\cdot : \overline{\mathbb{Z}} \times \overline{\mathbb{Z}} \to \overline{\mathbb{Z}}$ to the operands $-\infty$ and ∞:

$$
\begin{array}{llll}
x + (-\infty) = -\infty & \text{for all } x \in \overline{\mathbb{Z}} & x + \infty = \infty & \text{for all } x > -\infty \\
0 \cdot x = 0 & \text{for all } x > -\infty & x \cdot (-\infty) = -\infty & \text{for all } x > 0 \\
x \cdot \infty = \infty & \text{for all } x > 0 & x \cdot (-\infty) = \infty & \text{for all } x < 0 \\
x \cdot \infty = -\infty & \text{for all } x < 0 & &
\end{array}
$$

Many theorems in this paper only apply to structures called *complete lattices*. A complete lattice is a partially ordered set in which all subsets have both an infimum and a supremum. A partially ordered set is a set along with a partial order.

The *infimum* of a set S is the largest element, not necessarily in S, which is smaller or equal to all elements in S. Formally, the infimum is defined as follows:

$$\mathsf{inf}(S) := \mathsf{max}\{f \in \overline{\mathbb{Z}} \mid (\forall s \in S \ : \ f \leq s)\}$$

The supremum $\mathsf{sup}(S)$ is defined dually.

It is easily verified that the booleans and the implication form a complete lattice. We now show that the set $\overline{\mathbb{Z}}$ and the partial order $\leq$ form a complete lattice. It remains to show that all subsets of $\overline{\mathbb{Z}}$ have an infimum and a supremum.

To show the existence of an infimum function on our structure $(\overline{\mathbb{Z}}, \leq)$, we give a function and show that it adheres to this definition of infimum. We define the function $f : 2^{\overline{\mathbb{Z}}} \to \overline{\mathbb{Z}}$ as follows:

$$
f(S) = \begin{cases}
\infty & \text{if } S = \emptyset \\
\mathsf{min}(S) & \text{if } S \neq \emptyset \text{ and finite} \\
lb & \text{if } S \text{ is infinite and tightly bounded below by } lb \\
-\infty & \text{if } S \text{ is infinite and unbounded below}
\end{cases}
$$

Now we show that $f(S)$ is the infimum of S:

Lemma 2.
$f(S) = \mathsf{inf}(S)$, for any $S \subseteq \overline{\mathbb{Z}}$.

Proof. We consider the cases of the definition of f separately:

- $S = \emptyset$:

$$
\begin{aligned}
& f(S) \\
&= \infty \\
&= \mathsf{max}\{f \in \overline{\mathbb{Z}} \mid \mathsf{true}\} \\
&= \{\text{assumption; universal quantification over empty domain}\} \\
& \quad \mathsf{max}\{f \in \overline{\mathbb{Z}} \mid (\forall s \in S \ : \ f \leq s)\} \\
&= \mathsf{inf}(S)
\end{aligned}
$$

- $S \neq \emptyset$ and finite:

$$\begin{aligned} & f(S) \\ = \ & \mathsf{min}(S) \\ = \ & f \in S \text{ such that } (\forall s \in S \ : \ f \leq s) \\ = \ & \mathsf{max}\{f \in \overline{\mathbb{Z}} \mid (\forall s \in S \ : \ f \leq s)\} \\ = \ & \mathsf{inf}(S) \end{aligned}$$

- S is infinite and tightly bounded below by lb:

$$\begin{aligned} & = f(S) \\ & = lb \\ & = \{\text{tight bound}\} \\ & \quad \mathsf{max}\{f \in \overline{\mathbb{Z}} \mid (\forall s \in S \ : \ f \leq s)\} \\ & = \mathsf{inf}(S) \end{aligned}$$

- S is infinite and unbounded below:

$$\begin{aligned} & = f(S) \\ & = \ -\infty \\ & = \{-\infty \text{ is identity of } \mathsf{max}\} \\ & \quad \mathsf{max}\{f \in \overline{\mathbb{Z}} \mid \mathsf{false}\} \\ & = \mathsf{max}\{f \in \overline{\mathbb{Z}} \mid (\forall s \in S \ : \ f \leq s)\} \\ & = \mathsf{inf}(S) \end{aligned}$$

□

Theorem 3.
$(\overline{\mathbb{Z}}, \leq)$ *is a complete lattice.*

Proof. It is obvious that $(\overline{\mathbb{Z}}, \leq)$ is a partially ordered set. The existence of infimum for any subset follows from lemma 2; the existence of supremum is dual. It follows that $(\overline{\mathbb{Z}}, \leq)$ is a complete lattice. □

2.2 Fixed points

Both boolean equation systems and integer equation systems make use of the concept of fixed points. A fixed point of an equation $\mathbf{x} = e$, where $\mathbf{x}$ may appear in e, is a value for $\mathbf{x}$ for which the equation holds. We are usually not interested in just any fixed point, but in the least or greatest fixed point. The least fixed point is denoted μ and the greatest fixed point is denoted ν.

For some set A and some function $f : A \to A$, the least and greatest fixed points are defined as follows:

$$\begin{aligned} \mu x \in A.f &:= \mathsf{inf}\{x \in A \mid x = f(x)\} \\ \nu x \in A.f &:= \mathsf{sup}\{x \in A \mid x = f(x)\} \end{aligned}$$

We often use an expression for this function f. Of course, the variable x may then also appear in that expression. Let us look at an example.

Example 4 (Fixed points).

$$\begin{aligned}
&\nu x \in \mathbb{R}.x^3 \\
=\ &\{\text{def. greatest fixed point}\} \\
&\mathsf{sup}\{x \in \mathbb{R} \mid x = x^3\} \\
=\ &\{\text{calculus}\} \\
&\mathsf{sup}\{-1, 0, 1\} \\
=\ &1
\end{aligned}$$

Fixed points do not always exist. For example, the expression $x + 1$ has no fixed points in $\mathbb{N}$:

$$\begin{aligned}
&\nu x \in \mathbb{N}.x + 1 \\
=\ &\{\text{def. greatest fixed point}\} \\
&\mathsf{sup}\{x \in \mathbb{N} \mid x = x + 1\} \\
=\ &\mathsf{sup}\{x \in \mathbb{N} \mid \mathsf{false}\} \\
=\ &\mathsf{sup}(\emptyset), \text{ which is not defined in } \mathbb{N}
\end{aligned}$$

3 Boolean equation systems

As we have seen, model checking problems can be expressed as a problem of finding a solution to a boolean equation system. In this section, we shall make precise what boolean equation systems are and what it means for something to be a solution to a boolean equation system.

3.1 Boolean equation systems

Assume a set of propositional variables Var, which we shall denote by bold letters $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$. We construct boolean expressions with these variables, the boolean constants and the boolean operators for conjunction and disjunction. A *boolean expression* is given by the following grammar:

$$E ::= \mathbf{x} \mid \mathsf{true} \mid \mathsf{false} \mid E \wedge E \mid E \vee E$$

Note that this does not include negation.

A *boolean equation system* is a finite lists of least and greatest fixed point equations, where each right-hand side of an equation is a boolean expression. The syntax of boolean equation systems is given by the following grammar:

$$\mathcal{E} ::= \epsilon \mid (\mu \mathbf{x} = E)\ \mathcal{E} \mid (\nu \mathbf{x} = E)\ \mathcal{E},$$

where "ϵ" represents the empty string. We occasionally use the symbol σ instead of μ or ν, in cases where we do not care which of the two it is.

Before explaining about solutions of boolean equation systems, we define the following notions:

A variable is *bound* in $\mathcal{E}$ if and only if there is an equation in $\mathcal{E}$ with $\mathbf{x}$ on its left hand side. We inductively define the set of bound variables as follows:

$$\begin{aligned}
\mathsf{bnd}(\epsilon) &= \emptyset \\
\mathsf{bnd}((\mu \mathbf{x} = e)\ \mathcal{E}) &= \{\mathbf{x}\} \cup \mathsf{bnd}(\mathcal{E})
\end{aligned}$$

A variable is *occurring* in $\mathcal{E}$ if and only if it occurs in some right hand side of an equation in $\mathcal{E}$. We inductively define the set of occurring variables as follows:

$$\begin{aligned} \mathsf{occ}(\epsilon) &= \emptyset \\ \mathsf{occ}((\mu \mathbf{x} = e)\ \mathcal{E}) &= \mathsf{occ}(e) \cup \mathsf{occ}(\mathcal{E}) \\ \mathsf{occ}(\mathbf{x}) &= \{\mathbf{x}\} \\ \mathsf{occ}(\mathsf{true}) &= \mathsf{occ}(\mathsf{false}) = \emptyset \\ \mathsf{occ}(e_1 \wedge e_2) &= \mathsf{occ}(e_1 \vee e_2) = \mathsf{occ}(e_1) \cup \mathsf{occ}(e_2) \end{aligned}$$

A boolean equation system $\mathcal{E}$ is called *closed* if and only if all occurring variables are also bound, i.e. $\mathsf{occ}(\mathcal{E}) \subseteq \mathsf{bnd}(\mathcal{E})$. Otherwise, it is called *open*.

Example 5. Consider the following boolean equation system:

$$\mathcal{E} = (\mu \mathbf{x} = \mathbf{y})\ (\nu \mathbf{y} = \mathsf{false})$$

The *bound* variables are those that appear on some left hand side: $\mathsf{bnd}(\mathcal{E}) = \{\mathbf{x}, \mathbf{y}\}$.
The *occurring* variables are those that appear on some right hand side: $\mathsf{occ}(\mathcal{E}) = \{\mathbf{x}\}$.
As all variables on the right hand side also occur on the left hand side, the boolean equation system $\mathcal{E}$ is *closed*. If the second equation had been $(\nu \mathbf{y} = \mathbf{z})$, $\mathcal{E}$ would have been *open*.

We only consider boolean equation systems where variables do not occur more than once on the left hand side, i.e. boolean equation systems $\mathcal{E}\ \mathcal{F}$ where $\mathsf{bnd}(\mathcal{E}) \cap \mathsf{bnd}(\mathcal{F}) = \emptyset$.

3.2 Solutions

An *environment* $\theta : \mathsf{Var} \to \mathbb{B}$ is a function that assigns values to the variables. We often use an explicit assignment of a value to a variable. This is called an *assignment*. The assignment of value v to variable $\mathbf{x}$ in environment θ is denoted $\theta[\mathbf{x} := v]$. An assignment to an environment is again an environment, defined as follows:

$$\theta[\mathbf{x} := v](\mathbf{y}) = \begin{cases} \theta(\mathbf{y}) & \text{if } \mathbf{x} \neq \mathbf{y} \\ v & \text{if } \mathbf{x} = \mathbf{y} \end{cases}$$

An expression, in which variables may occur, only has a value with respect to the values of those variables. Put otherwise, the meaning of an expression is a function from an environment to a boolean value. This is called the *interpretation*. The interpretation of an expression e in an environment θ, denoted $[\![e]\!]\theta$, is defined as follows:

$$\begin{aligned} [\![\mathbf{x}]\!]\theta &:= \theta(\mathbf{x}) \\ [\![\mathsf{true}]\!]\theta &:= \mathsf{true} \\ [\![e_1 \wedge e_2]\!]\theta &:= [\![e_1]\!]\theta \wedge [\![e_2]\!]\theta \end{aligned}$$

The cases for false and $e_1 \vee e_2$ are immediate duals of true and $e_1 \wedge e_2$. Note that expressions are mere strings and that the interpretation function gives meaning to these strings. For example, in the definition $[\![e_1 \wedge e_2]\!]\theta := [\![e_1]\!]\theta \wedge [\![e_2]\!]\theta$, the occurrence of "$\wedge$" on the left hand side should be thought of as just a symbol, while the "$\wedge$" on the right hand side is the actual boolean conjunction operator. They do correspond, but we are defining the symbol on the left hand side while we assume the operator on the right hand side to be already defined.

An environment θ is called a *solution* to a boolean equation system $\mathcal{E}$ if and only if all of the following hold:

1. For each equation, the left hand side equals the right hand side, i.e. for all equations $(\sigma \mathbf{x} = e)$ in $\mathcal{E}$ it holds that $\theta(\mathbf{x}) = [\![e]\!]\theta$.
2. For equations with a *least* fixed point $(\mu \mathbf{x} = e)$, the value for $\mathbf{x}$ must be as *strong* as possible. Similarly, for *greatest* fixed point equations the value must be as *weak* as possible.
3. Because this second condition may be conflicting, we define the leftmost fixed point signs to have precedence over fixed point signs that follow.

We formally define the solution to a boolean equation system as follows:

$$\begin{aligned} [\![\epsilon]\!]\theta &= \theta \\ [\![(\mu \mathbf{x} = e)\ \mathcal{E}]\!]\theta &= [\![\mathcal{E}]\!]\theta[\mathbf{x} := [\![e]\!]([\![\mathcal{E}]\!]\theta[\mathbf{x} := \mathsf{false}])] \\ [\![(\nu \mathbf{x} = e)\ \mathcal{E}]\!]\theta &= [\![\mathcal{E}]\!]\theta[\mathbf{x} := [\![e]\!]([\![\mathcal{E}]\!]\theta[\mathbf{x} := \mathsf{true}])] \end{aligned}$$

This $[\![\mathcal{E}]\!]\theta$ is not to be confused with the interpretation $[\![e]\!]\theta$ of an expression e in an environment θ.

From the third condition it can be seen that the order of equations matters, as can be seen in the following example:

Example 6. Consider the following boolean equation system:

$$\mathcal{E} = (\mu \mathbf{x} = \mathbf{y})\ (\nu \mathbf{y} = \mathbf{x})$$

Let θ be an environment such that $\theta(\mathbf{x}) = \theta(\mathbf{y}) = \mathsf{false}$, and let θ' be such that $\theta'(\mathbf{x}) = \theta'(\mathbf{y}) = \mathsf{true}$. We shall determine if they are solutions to $\mathcal{E}$. Note that both these environments satisfy the first criterion for being a solution. Also, note that both seem to violate the second criterion on one of the two equations. As the third criterion states that leftmost fixed point signs take precedence, we know that in this example, the least fixed point sign is more important than the greatest fixed point sign. We conclude that θ is a solution to $\mathcal{E}$ and θ' is not. Observe that this is due to the order of the fixed point symbols: had the fixed point symbols been the other way around, θ' would have been a solution to $\mathcal{E}$ and θ would not.

We shall now calculate the solution to $\mathcal{E}$ by only unfolding definitions. Even though this is not the way one would usually calculate a solution, this should make the reader more familiar with

the definitions used.

$$
\begin{aligned}
& [\![(\mu \mathbf{x} = \mathbf{y})\ (\nu \mathbf{y} = \mathbf{x})]\!]\theta \\
= & \ \{\text{def. solution}\} \\
& [\![(\nu \mathbf{y} = \mathbf{x})]\!]\theta[\mathbf{x} := [\![\mathbf{y}]\!]([\![(\nu \mathbf{y} = \mathbf{x})]\!]\theta[\mathbf{x} := \mathsf{false}])] \\
= & \ \{\text{def. solution}\} \\
& [\![(\nu \mathbf{y} = \mathbf{x})]\!]\theta[\mathbf{x} := [\![\mathbf{y}]\!]([\![\epsilon]\!]\theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := [\![\mathbf{x}]\!]([\![\epsilon]\!]\theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := \mathsf{true}])])] \\
= & \ \{\text{def. solution}\} \\
& [\![(\nu \mathbf{y} = \mathbf{x})]\!]\theta[\mathbf{x} := [\![\mathbf{y}]\!]([\![\epsilon]\!]\theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := [\![\mathbf{x}]\!](\theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := \mathsf{true}])])] \\
= & \ \{\text{def. interpretation}\} \\
& [\![(\nu \mathbf{y} = \mathbf{x})]\!]\theta[\mathbf{x} := [\![\mathbf{y}]\!]([\![\epsilon]\!]\theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := \theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := \mathsf{true}](\mathbf{x})])] \\
= & \ \{\text{def. assignment}\} \\
& [\![(\nu \mathbf{y} = \mathbf{x})]\!]\theta[\mathbf{x} := [\![\mathbf{y}]\!]([\![\epsilon]\!]\theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := \mathsf{false}])] \\
= & \ \{\text{def. solution}\} \\
& [\![(\nu \mathbf{y} = \mathbf{x})]\!]\theta[\mathbf{x} := [\![\mathbf{y}]\!](\theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := \mathsf{false}])] \\
= & \ \{\text{def. interpretation}\} \\
& [\![(\nu \mathbf{y} = \mathbf{x})]\!]\theta[\mathbf{x} := \theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := \mathsf{false}](\mathbf{y})] \\
= & \ \{\text{def. assignment}\} \\
& [\![(\nu \mathbf{y} = \mathbf{x})]\!]\theta[\mathbf{x} := \mathsf{false}] \\
= & \ \{\text{def. solution}\} \\
& [\![\epsilon]\!]\theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := [\![\mathbf{x}]\!]([\![\epsilon]\!]\theta[\mathbf{x} := \mathsf{false}])] \\
= & \ \{\text{def. solution}\} \\
& [\![\epsilon]\!]\theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := [\![\mathbf{x}]\!](\theta[\mathbf{x} := \mathsf{false}])] \\
= & \ \{\text{def. interpretation}\} \\
& [\![\epsilon]\!]\theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := \theta[\mathbf{x} := \mathsf{false}](\mathbf{x})] \\
= & \ \{\text{def. assignment}\} \\
& [\![\epsilon]\!]\theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := \mathsf{false}] \\
= & \ \{\text{def. interpretation}\} \\
& \theta[\mathbf{x} := \mathsf{false}][\mathbf{y} := \mathsf{false}]
\end{aligned}
$$

The correct solution is indeed an environment where both variables have the value false, as we had already seen.

3.3 Standard recursive form

It is convenient to put boolean equation systems in a particular form, in which we disallow "$\wedge$" and "$\vee$" symbols to occur in the same equation. We also remove occurrences of the boolean constants true and false. This is called *standard recursive form*. Boolean equation systems in standard recursive form are given by the following grammar:

$$
\begin{aligned}
E &::= E_\wedge \mid E_\vee \\
E_\wedge &::= \mathbf{x} \mid E_\wedge \wedge E_\wedge \\
E_\vee &::= \mathbf{x} \mid E_\vee \vee E_\vee \\
\mathcal{E} &::= \epsilon \mid (\mu \mathbf{x} = E)\ \mathcal{E} \mid (\nu \mathbf{x} = E)\ \mathcal{E}
\end{aligned}
$$

Example 7. The following boolean equation systems are in standard recursive form:

- $(\mu \mathbf{x} = \mathbf{y} \wedge \mathbf{z})\ (\nu \mathbf{y} = \mathbf{x} \vee \mathbf{z})$
- $(\nu \mathbf{x} = \mathbf{x})$
- ϵ, the empty boolean equation system

These boolean equation systems are *not* in standard recursive form:

- $(\mu \mathbf{x} = (\mathbf{y} \wedge \mathbf{z}) \vee \mathbf{x})\ (\nu \mathbf{y} = \mathbf{y})$, as "$\wedge$" and "$\vee$" may not occur in the same expression.
- $(\nu \mathbf{x} = \mathsf{true})$, as the boolean constant true is excluded from standard recursive form.

It has been shown by Mader [7] and formalized by Keiren and Willemse [5], that every boolean equation system can be transformed to an equivalent boolean equation system in standard recursive form. This translation does not require the boolean equation system to be solved. The translation introduces new equations for parts of expressions that violate the definition of standard recursive form. We shall illustrate this with an example:

Example 8. Consider the following boolean equation system:

$$(\mu \mathbf{x} = (\mathbf{y} \wedge \mathbf{z}) \vee \mathbf{x})\ (\nu \mathbf{y} = \mathsf{true})$$

This is not in standard recursive form, as the first equation has "$\wedge$" and "$\vee$" symbols in the same equation and a boolean constant still appears in the second equation. We first introduce a new equation for the violating expression true:

$$(\mu \mathbf{x} = (\mathbf{y} \wedge \mathbf{z}) \vee \mathbf{x})\ (\nu \mathbf{y} = \underline{\mathbf{t}})\ \underline{(\nu \mathbf{t} = \mathbf{t})}$$

The variable $\mathbf{t}$ represents the boolean constant true. A boolean variable equals itself both for true and false. Because of the largest fixed point, the solution is the weakest, i.e. true. We then replace all occurrences of the boolean constant true with this variable $\mathbf{t}$ and achieve an equivalent boolean equation system. We still need to change the other equation before the boolean equation system is in standard recursive form. Because "$\wedge$" and "$\vee$" symbols may not occur in the same equation, we put the subexpression $\mathbf{y} \wedge \mathbf{z}$ in a separate equation:

$$(\mu \mathbf{x} = \underline{\mathbf{x}'} \vee \mathbf{x})\ \underline{(\mu \mathbf{x}' = \mathbf{y} \wedge \mathbf{z})}\ (\nu \mathbf{y} = \mathbf{t})\ (\nu \mathbf{t} = \mathbf{t})$$

This is in standard recursive form.

We shall not go into further detail about this translation. From now on, we assume every boolean equation system to be in standard recursive form.

4 Integer equation systems

We consider the *integer equation systems* [1]. Integer equation systems are very similar to boolean equation systems, with three differences:

1. Variables over a different domain: the extended integers $\overline{\mathbb{Z}}$
2. Different operators: multiplication, addition, minimum and maximum
3. Only least fixed points and no greatest fixed points

Assume a set Var of variables over the extended integers, which we shall denote by bold letters $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$. An *integer expression* is given by the following grammar:

$$E ::= \mathbf{x} \mid c \mid b \cdot E \mid E + E \mid E \wedge E \mid E \vee E$$

Where $b, c \in \overline{\mathbb{Z}}$. Furthermore, we require that $b \geq 1$.

Integer equation systems are given by the following grammar. This is similar to how boolean equation systems are defined, only using integer expressions and with no greatest fixed point symbol:

$$\mathcal{E} ::= \epsilon \mid (\mu \mathbf{x} = E)\ \mathcal{E}$$

In integer equation systems, an *environment* θ is a function from variables to values in $\overline{\mathbb{Z}}$.

Interpretation of integers expressions is defined similarly to interpretation of boolean expressions, only with different operators. In integer expressions, the symbol "$\vee$" corresponds to the maximum operator max and the symbol "$\wedge$" corresponds to the minimum operator min.

$$\begin{aligned}
[\![\mathbf{x}]\!]\theta &:= \theta(\mathbf{x}) \\
[\![c]\!]\theta &:= c \\
[\![b \cdot e]\!]\theta &:= b \cdot [\![e]\!]\theta \\
[\![e_1 + e_2]\!]\theta &:= [\![e_1]\!]\theta + [\![e_2]\!]\theta \\
[\![e_1 \wedge e_2]\!]\theta &:= [\![e_1]\!]\theta\ \mathsf{min}\ [\![e_2]\!]\theta \\
[\![e_1 \vee e_2]\!]\theta &:= [\![e_1]\!]\theta\ \mathsf{max}\ [\![e_2]\!]\theta
\end{aligned}$$

It is fairly easy to see that the interpretation function is monotonic:

Theorem 9 (Monotonicity of interpretation)**.**
The interpretation of an expression is monotonic with respect to the environment it is interpreted in. Given two environments θ and θ' such that $\theta(\mathbf{x}) \leq \theta'(\mathbf{x})$ for all variables $\mathbf{x}$, it holds that $[\![e]\!]\theta \leq [\![e]\!]\theta'$ for any expression e.

Proof. Proof by structural induction on e.

- $c \in \overline{\mathbb{Z}}$:

  $$[\![c]\!]\theta = c = [\![c]\!]\theta'$$

- $\mathbf{x}$:

  $$\begin{aligned}
  & [\![\mathbf{x}]\!]\theta \\
  = {} & \theta(\mathbf{x}) \\
  \leq {} & \{\text{assumption}\} \\
  & \theta'(\mathbf{x}) \\
  = {} & [\![\mathbf{x}]\!]\theta'
  \end{aligned}$$

- $b \cdot e$:

  Induction Hypothesis: $[\![e]\!]\theta \leq [\![e]\!]\theta'$

  $$\begin{aligned}
  & [\![b \cdot e]\!]\theta \\
  = {} & b \cdot [\![e]\!]\theta \\
  \leq {} & \{\text{Induction Hypothesis, as } b \geq 1\} \\
  & b \cdot [\![e]\!]\theta' \\
  = {} & [\![b \cdot e]\!]\theta'
  \end{aligned}$$

- $e_1 \vee e_2$:

  Induction Hypothesis: $[\![e_1]\!]\theta \leq [\![e_1]\!]\theta'$ and $[\![e_2]\!]\theta \leq [\![e_2]\!]\theta'$

  $$\begin{aligned} & [\![e_1 \vee e_2]\!]\theta \\ = \; & [\![e_1]\!]\theta \;\mathsf{max}\; [\![e_2]\!]\theta \\ \leq \; & \{\text{Induction Hypothesis}\} \\ & [\![e_1]\!]\theta' \;\mathsf{max}\; [\![e_2]\!]\theta' \\ = \; & [\![e_1 \vee e_2]\!]\theta' \end{aligned}$$

- $e_1 + e_2$ and $e_1 \wedge e_2$: analogous to $e_1 \vee e_2$.

□

4.1 Solutions

An environment θ is called a *solution* to an integer equation system $\mathcal{E}$ if and only if for all equations in $\mathcal{E}$, the left hand side indeed equals the right hand side, i.e. for all equations $(\mu\mathbf{x} = e)$ in $\mathcal{E}$ it must hold that $\theta(\mathbf{x}) = [\![e]\!]\theta$. As the least fixed point symbols suggest, we are interested in the *least* solution, i.e. a solution θ such that for all variables $\mathbf{x}$ bound in $\mathcal{E}$, $\theta(\mathbf{x}) \leq \theta'(\mathbf{x})$ for any solution θ' of $\mathcal{E}$. Finding the least solution of an integer equation system $\mathcal{E}$ is called *solving* $\mathcal{E}$.

Example 10. Consider the following integer equation system $\mathcal{E}$:

$$(\mu\mathbf{x} = 10)\ (\mu\mathbf{y} = \mathbf{x} + \mathbf{y})$$

Obviously, in any solution, $\mathbf{x}$ must equal 10. This means that for the second equation to hold, it must hold that $\mathbf{y} = 10 + \mathbf{y}$. This is only true for $\mathbf{y} = -\infty$ or $\mathbf{y} = \infty$. So, both $\theta[\mathbf{x} := 10][\mathbf{y} := -\infty]$ and $\theta[\mathbf{x} := 10][\mathbf{y} := \infty]$ are solutions for $\mathcal{E}$, for any environment θ. If we wanted to solve $\mathcal{E}$, we want the smallest of these, i.e. $\theta[\mathbf{x} := 10][\mathbf{y} := -\infty]$.

From this definition it directly follows that the least solution of an integer equation system $\mathcal{E}$ does not depend on the ordering of the equations in $\mathcal{E}$:

Theorem 11 (Least solution independent from ordering)**.**
The least solution of an integer equation system is independent from the ordering of its equations. The integer equation system $\mathcal{F} = \mathcal{E}\ (\mu\mathbf{x} = e)\ \mathcal{E}'\ (\mu\mathbf{y} = e')\ \mathcal{E}''$ has the same least solution as $\mathcal{F}' = \mathcal{E}\ (\mu\mathbf{y} = e')\ \mathcal{E}'\ (\mu\mathbf{x} = e)\ \mathcal{E}''$, where the equations for $\mathbf{x}$ and $\mathbf{y}$ have been swapped.

Proof. Let θ be the least solution of $\mathcal{F}$. So, for all equations $(\mu\mathbf{z} = e'')$ in $\mathcal{F}$ it holds that $\theta(\mathbf{z}) = [\![e'']\!]\theta$. This also holds for all equations in $\mathcal{F}'$, as these are the same equations as in $\mathcal{F}$. It is clear that any solution for $\mathcal{F}$ is also a solution for $\mathcal{F}'$ and vice-versa. Since θ is the *least* solution of $\mathcal{F}$, it also holds that $\theta(\mathbf{z}) \leq \theta'(\mathbf{z})$, for any $\mathbf{z} \in \mathsf{bnd}(\mathcal{F})$ and for any solution θ' for $\mathcal{F}$. Since any solution θ' of $\mathcal{F}$ is also a solution for $\mathcal{F}'$ and because $\mathsf{bnd}(\mathcal{F}) = \mathsf{bnd}(\mathcal{F}')$, the solution θ is also smaller or equal to any solution of $\mathcal{F}'$ in all its bound variables. We conclude that θ must also be the least solution of $\mathcal{F}'$. □

Note that this is quite different from boolean equation systems, where solutions depend on the ordering of the equations.

By Tarski's fixed point theorem [8], we can show that such a least solution for a closed integer equation system exists. This follows from the facts that $(\overline{\mathbb{Z}}, \leq)$ is a complete lattice, which we have

$$[\![\epsilon]\!]\theta = \theta$$
$$[\![(\mu \mathbf{x} = e)\ \mathcal{E}]\!]\theta = [\![\mathcal{E}]\!]\theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![e]\!]([\![\mathcal{E}]\!]\theta[\mathbf{x} := f])]$$

Table 1: The inductive definition of the least solution of an integer equation system.

already seen in theorem 3, and from the interpretation of integer expressions being a monotonic function, as already shown in theorem 9. However, we are not only interested in the *existence* of this least solution. For the remainder of this paper, we concentrate on solving an integer equation system, i.e. *finding* its least solution.

We present an inductive definition of least solution and show that it is actually equivalent to the above definition.

Let $[\![\mathcal{E}]\!]\theta$, where $\mathcal{E}$ is an integer equation system and θ is an environment, be defined as in Table 4.1. This $[\![\mathcal{E}]\!]\theta$ is not to be confused with the interpretation $[\![e]\!]\theta$ of an expression e in an environment θ.

We show that $[\![\mathcal{E}]\!]\theta$ is the least solution of closed $\mathcal{E}$ in arbitrary environment θ. We generalize this to open $\mathcal{E}$, assuming all variables that occur in $\mathcal{E}$ but are not bound by $\mathcal{E}$ have a value in θ, i.e. $(\forall \mathbf{x} \in \mathsf{occ}(\mathcal{E}) \setminus \mathsf{bnd}\ :\ \theta(\mathbf{x})$ is defined). Note that this trivially holds for closed $\mathcal{E}$, as $\mathsf{occ}(\mathcal{E}) \setminus \mathsf{bnd}$ is empty. We prove it is the least solution in two steps: we first prove that it is a solution and then we prove that all other solutions are at least as large.

We first show that $[\![\mathcal{E}]\!]\theta$ is a solution.

Lemma 12.
$[\![\mathcal{E}]\!]\theta$ is a solution for $\mathcal{E}$, assuming $(\forall \mathbf{x} \in \mathsf{occ}(\mathcal{E}) \setminus \mathsf{bnd}\ :\ \theta(\mathbf{x})$ is defined).

Proof. For $[\![\mathcal{E}]\!]\theta$ to be a solution, it must hold that for each equation $(\mu \mathbf{x} = e)$ in $\mathcal{E}$, the left hand side equals the right hand side, i.e. $(\forall (\mu \mathbf{x} = e)$ in $\mathcal{E}\ :\ [\![\mathcal{E}]\!]\theta(\mathbf{x}) = [\![e]\!]([\![\mathcal{E}]\!]\theta))$. We show this by structural induction on $\mathcal{E}$:

- ϵ: trivial (universal quantification over empty domain)

For the inductive step, we distinguish the cases of the bound variable of the first equation from the other bound variables:

> Induction Hypothesis: $(\forall (\mu \mathbf{z} = e')$ in $\mathcal{E}\ :\ [\![\mathcal{E}]\!]\theta'(\mathbf{z}) = [\![e']\!]([\![\mathcal{E}]\!]\theta'))$
> for any θ' such that $(\forall \mathbf{x} \in \mathsf{occ}(\mathcal{E}) \setminus \mathsf{bnd}\ :\ \theta'(\mathbf{x})$ is defined) holds.

- $(\mu \mathbf{x} = e)\ \mathcal{E}$, for $\mathbf{x}$:

$$
\begin{aligned}
& (\llbracket (\mu \mathbf{x} = e)\ \mathcal{E} \rrbracket \theta)(\mathbf{x}) \\
= & \{\text{def. } \llbracket \cdot \rrbracket\} \\
& (\llbracket \mathcal{E} \rrbracket \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.\llbracket e \rrbracket(\llbracket \mathcal{E} \rrbracket \theta[\mathbf{x} := f])])(\mathbf{x}) \\
= & \{\text{def. assignment}\} \\
& \mu f \in \overline{\mathbb{Z}}.\llbracket e \rrbracket(\llbracket \mathcal{E} \rrbracket \theta[\mathbf{x} := f]) \\
= & \{\text{replace } f \text{ with entire fixed point}\} \\
& \mu f \in \overline{\mathbb{Z}}.\llbracket e \rrbracket(\llbracket \mathcal{E} \rrbracket \theta[\mathbf{x} := \mu f' \in \overline{\mathbb{Z}}.\llbracket e \rrbracket(\llbracket \mathcal{E} \rrbracket \theta[\mathbf{x} := f'])]) \\
= & \{f \text{ does not occur anymore}\} \\
& \llbracket e \rrbracket(\llbracket \mathcal{E} \rrbracket \theta[\mathbf{x} := \mu f' \in \overline{\mathbb{Z}}.\llbracket e \rrbracket(\llbracket \mathcal{E} \rrbracket \theta[\mathbf{x} := f'])]) \\
= & \{\text{def. } \llbracket \cdot \rrbracket\} \\
& \llbracket e \rrbracket(\llbracket (\mu \mathbf{x} = e)\ \mathcal{E} \rrbracket \theta)
\end{aligned}
$$

- $(\mu \mathbf{x} = e)\ \mathcal{E}$, for $\mathbf{y} \neq \mathbf{x}$:

$$
\begin{aligned}
& (\llbracket (\mu \mathbf{x} = e)\ \mathcal{E} \rrbracket \theta)(\mathbf{y}) \\
= & \{\text{def. } \llbracket \cdot \rrbracket\} \\
& (\llbracket \mathcal{E} \rrbracket \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.\llbracket e \rrbracket(\llbracket \mathcal{E} \rrbracket \theta[\mathbf{x} := f])])(\mathbf{y}) \\
= & \{\text{Induction Hypothesis, as } \theta[\mathbf{x} := f] \text{ is such that the condition holds}\} \\
& \llbracket e' \rrbracket(\llbracket \mathcal{E} \rrbracket \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.\llbracket e \rrbracket(\llbracket \mathcal{E} \rrbracket \theta[\mathbf{x} := f])]) \\
= & \{\text{def. } \llbracket \cdot \rrbracket\} \\
& \llbracket e' \rrbracket(\llbracket (\mu \mathbf{x} = e)\ \mathcal{E} \rrbracket \theta)
\end{aligned}
$$

This proves that for each equation in $\mathcal{E}$, the left hand side equals the right hand side. It follows that $\llbracket \mathcal{E} \rrbracket \theta$ is a solution. □

Now the other step is to show that $\llbracket \mathcal{E} \rrbracket \theta$ is not just a solution, but that it is the *least* solution.

Theorem 13.
$\llbracket \mathcal{E} \rrbracket \theta$ is the least solution for $\mathcal{E}$, for any θ such that $(\forall \mathbf{x} \in \mathsf{occ}(\mathcal{E}) \setminus \mathsf{bnd}\ :\ \theta(\mathbf{x})$ is defined$)$. So, for any solution θ' of $\mathcal{E}$, the solution $\llbracket \mathcal{E} \rrbracket \theta$ is smaller or equal to θ' in any variable bound by $\mathcal{E}$:

$$(\forall \mathbf{y} \in \mathsf{bnd}(\mathcal{E})\ :\ \llbracket \mathcal{E} \rrbracket \theta(\mathbf{y}) \leq \theta'(\mathbf{y}))$$

Proof. We prove this by structural induction over $\mathcal{E}$:

- ϵ: trivial (empty universal quantification)

For the inductive step, we distinguish the cases whether $\mathbf{y}$ is bound in the first equation or not:

- $(\mu \mathbf{x} = e)\ \mathcal{E}$, for $\mathbf{x}$:

  Induction Hypothesis: $(\forall \mathbf{z} \in \mathsf{bnd}(\mathcal{E})\ :\ [\![\mathcal{E}]\!]\eta(\mathbf{z}) \leq \eta'(\mathbf{z}))$ for any solution η' of $\mathcal{E}$ and any η such that $(\forall \mathbf{x} \in \mathsf{occ}(\mathcal{E}) \setminus \mathsf{bnd}\ :\ \eta(\mathbf{x})$ is defined) holds.

  $[\![(\mu \mathbf{x} = e)\mathcal{E}]\!]\theta(\mathbf{x})$
  $= \{$def. least solution$\}$
  $[\![\mathcal{E}]\!]\theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![e]\!].([\![\mathcal{E}]\!]\theta[\mathbf{x} := f])](\mathbf{x})$
  $= \{$def. assignment$\}$
  $\mu f \in \overline{\mathbb{Z}}.[\![e]\!]([\![\mathcal{E}]\!]\theta[\mathbf{x} := f])$
  $\leq \{$monotonicity of interpretation, Induction Hypothesis for $\eta' = \theta'$ and $\eta = \theta[\mathbf{x} := f]$, noting that it fulfills the necessary condition$\}$
  $\mu f \in \overline{\mathbb{Z}}.[\![e]\!](\theta'[\mathbf{x} := \theta'(\mathbf{x})])$
  $= \{$no occurrences of $f\}$
  $[\![e]\!]\theta'[\mathbf{x} := \theta'(\mathbf{x})]$
  $= \{$def. assignment$\}$
  $[\![e]\!]\theta'$
  $= \{$assumption: θ' is a solution$\}$
  $\theta'(\mathbf{x})$

- $(\mu \mathbf{x} = e)\ \mathcal{E}$, for $\mathbf{y} \neq \mathbf{x}$:

  $[\![(\mu \mathbf{x} = e)\mathcal{E}]\!]\theta(\mathbf{y})$
  $= \{$def. least solution$\}$
  $[\![\mathcal{E}]\!]\theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![e]\!].([\![\mathcal{E}]\!]\theta[\mathbf{x} := f])](\mathbf{y})$
  $\leq \{$Induction Hypothesis for $\eta' = \theta'$ and $\theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![e]\!].([\![\mathcal{E}]\!]\theta[\mathbf{x} := f])]$, noting that it fulfills the necessary condition$\}$
  $\theta'(\mathbf{y})$

□

We shall give an example of how one could find the solution of an integer equation system by unfolding this definition. Again, this is not the way one would usually find a solution, but it should help the reader to understand this definition more thoroughly:

Example 14. Consider again the integer equation system of example 10:

$(\mu \mathbf{x} = 10)\ (\mu \mathbf{y} = \mathbf{x} + \mathbf{y})$

We unfold the formal definition of least solution:

$$
\begin{aligned}
& [\![(\mu \mathbf{x} = 10)\ (\mu \mathbf{y} = \mathbf{x} + \mathbf{y})]\!]\theta \\
= & \ \{\text{def. least solution}\} \\
& [\![(\mu \mathbf{y} = \mathbf{x} + \mathbf{y})]\!]\theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![10]\!]([\![(\mu \mathbf{y} = \mathbf{x} + \mathbf{y})]\!]\theta[\mathbf{x} := f])] \\
= & \ \{\text{def. interpretation}\} \\
& [\![(\mu \mathbf{y} = \mathbf{x} + \mathbf{y})]\!]\theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.10] \\
= & \ \{\text{def. least fixed point}\} \\
& [\![(\mu \mathbf{y} = \mathbf{x} + \mathbf{y})]\!]\theta[\mathbf{x} := \mathsf{min}\{f \in \overline{\mathbb{Z}} \mid f = 10\}] \\
= & \ \{\text{calculus}\} \\
& [\![(\mu \mathbf{y} = \mathbf{x} + \mathbf{y})]\!]\theta[\mathbf{x} := 10] \\
= & \ \{\text{def. least solution}\} \\
& [\![\epsilon]\!]\theta[\mathbf{x} := 10][\mathbf{y} := \mu f \in \overline{\mathbb{Z}}.[\![\mathbf{x} + \mathbf{y}]\!]([\![\epsilon]\!]\theta[\mathbf{x} := 10][\mathbf{y} := f])] \\
= & \ \{\text{def. least solution}\} \\
& [\![\epsilon]\!]\theta[\mathbf{x} := 10][\mathbf{y} := \mu f \in \overline{\mathbb{Z}}.[\![\mathbf{x} + \mathbf{y}]\!](\theta[\mathbf{x} := 10][\mathbf{y} := f])] \\
= & \ \{\text{def. interpretation}\} \\
& [\![\epsilon]\!]\theta[\mathbf{x} := 10][\mathbf{y} := \mu f \in \overline{\mathbb{Z}}.[\![\mathbf{x}]\!](\theta[\mathbf{x} := 10][\mathbf{y} := f]) + [\![\mathbf{y}]\!](\theta[\mathbf{x} := 10][\mathbf{y} := f])] \\
= & \ \{\text{def. interpretation, twice}\} \\
& [\![\epsilon]\!]\theta[\mathbf{x} := 10][\mathbf{y} := \mu f \in \overline{\mathbb{Z}}.\theta[\mathbf{x} := 10][\mathbf{y} := f](\mathbf{x}) + \theta[\mathbf{x} := 10][\mathbf{y} := f](\mathbf{y})] \\
= & \ \{\text{def. assignment, twice}\} \\
& [\![\epsilon]\!]\theta[\mathbf{x} := 10][\mathbf{y} := \mu f \in \overline{\mathbb{Z}}.10 + f] \\
= & \ \{\text{def. least fixed point}\} \\
& [\![\epsilon]\!]\theta[\mathbf{x} := 10][\mathbf{y} := \mathsf{min}\{f \in \overline{\mathbb{Z}} \mid f = 10 + f\}] \\
= & \ \{\text{addition for } \overline{\mathbb{Z}}\} \\
& [\![\epsilon]\!]\theta[\mathbf{x} := 10][\mathbf{y} := \mathsf{min}\{-\infty, \infty\}] \\
= & \ \{\text{def. minimum}\} \\
& [\![\epsilon]\!]\theta[\mathbf{x} := 10][\mathbf{y} := -\infty] \\
= & \ \{\text{def. least solution}\} \\
& \theta[\mathbf{x} := 10][\mathbf{y} := -\infty]
\end{aligned}
$$

4.2 Simple integer equation systems

Of special interest is the subclass of integer equation systems of a particular form, called *simple integer equation systems*. A *simple* integer expression is given by the following grammar, where $\mathbf{x}$ is a variable, $c \in \overline{\mathbb{Z}}$ and $a \in \mathbb{Z}$:

$$E ::= c \mid \mathbf{x} \mid E + a \mid E_1 \vee E_2 \mid E_1 \wedge E_2$$

An integer equation system is simple if and only if the right hand side of all its equations are simple.

Example 15. The following integer equation systems are simple:

- $(\mu \mathbf{x} = (\mathbf{y} \wedge (\mathbf{x} \vee \mathbf{z})) + (-4))$
- $(\mu \mathbf{x} = 3)\ (\mu \mathbf{y} = (\mathbf{x} \wedge \mathbf{y}) + 10)$

- ϵ, the empty integer equation system

These integer equation systems are *not* simple:

- $(\mu \mathbf{x} = 7 \cdot \mathbf{x})$, as multiplication is excluded from simple expressions.
- $(\mu \mathbf{x} = \mathbf{x} + \mathbf{y})$ $(\mu \mathbf{y} = 5)$, as the right operand of addition must be a finite *constant*.
- $(\mu \mathbf{x} = \mathbf{x} + \infty)$, as the right operand of addition must be a *finite* constant.

Rewriting an arbitrary integer equation system to simple form is not trivial. Consider the usual rewrite rules for the operators $\cdot, +, \mathsf{max}$ and min. We show that not every integer equation system can be rewritten to simple form with these rules:

Theorem 16.
Not every integer equation system can be rewritten to simple form by applying the usual rewrite rules for $\cdot, +, \mathsf{max}$ *and* min.

Proof. Proof by counterexample. Consider the non-simple equation system consisting of only the equation $(\mu \mathbf{x} = 2 \cdot \mathbf{x})$. Since multiplication is excluded from simple expressions, this is not in simple form. The equation can be rewritten to $(\mu \mathbf{x} = \mathbf{x} + \mathbf{x})$ but not any further. This equation is not simple, as $\mathbf{x}$ is not a constant. We conclude that not every integer equation system can be rewritten to a simple integer equation system with these rules. □

Note, however, that there always exists a simple integer equation system with the same least solution as a given closed integer equation system. Since every closed integer equation system has a least solution, we can construct a simple integer equation system with equal least solution from this least solution as follows. Let θ be the least solution of integer equation system $\mathcal{E}$. Let $\mathcal{E}'$ be the equation system $\mathcal{E}$, where for each equation $\mu \mathbf{x}_i = e_i$ in $\mathcal{E}$, the right hand side e_i is replaced with $\theta(\mathbf{x})$. Clearly, $\mathcal{E}'$ is a simple integer equation system with the same least solution θ as $\mathcal{E}$. As stated, this translation is not trivial, as it first requires us to solve $\mathcal{E}$.

Example 17. Consider the following integer equation system $\mathcal{E}$:

$$(\mu \mathbf{x} = 2 \cdot \mathbf{x})$$

The least solution to $\mathcal{E}$ is $\theta[\mathbf{x} := -\infty]$ for any environment θ. We can use this knowledge to construct the following integer equation system $\mathcal{E}'$, which is simple and has the same least solution:

$$(\mu \mathbf{x} = -\infty)$$

4.3 Disjunctive normal form

It is convenient to have all simple integer equation systems in the same form, such that any integer expression can easily be rewritten to an expression in this form. Specifically, this reduces the number of cases we distinguish in proofs. For this, we introduce the *disjunctive normal form*, which is given by the following grammar:

$$\begin{aligned} E &::= E_\vee \\ E_\vee &::= E_\wedge \mid E_\wedge \vee E_\vee \\ E_\wedge &::= E_+ \mid E_+ \wedge E_\wedge \\ E_+ &::= \mathbf{x} \mid c \mid E_+ + a \end{aligned}$$

A simple integer expression in this form is a disjunction over conjunctions over additions. Stated otherwise, an expression is in disjunctive normal form if no $\vee$-expression occurs within a $\wedge$-expression and no $\wedge$-expression occurs within a $+$-expression.

We now show that every simple expression can indeed be rewritten to this form:

Theorem 18.
Any simple integer expression can be rewritten to an equivalent simple integer expression in disjunctive normal form, i.e. an expression $((\mathbf{x} + c) \wedge e_1) \vee e_2$ such that no $\vee$-expression occurs within a $\wedge$-expression and no $\wedge$-expression occurs within a $+$-expression.

Proof. By structural induction:

- $c \in \mathbb{Z}$:

  c
  $=$ {let c be both the minimum and maximum value}
  $((\mathbf{x} + 0) \wedge c) \vee c$,
  which is in disjunctive normal form, for arbitrary $\mathbf{x}$.

- $\mathbf{x}$:

  $\mathbf{x}$
  $=$ {identity elements for addition, minimum and maximum}
  $((\mathbf{x} + 0) \wedge \infty) \vee -\infty$,
  which is in disjunctive normal form.

- $e + a$:

  Induction Hypothesis: $e = ((\mathbf{x} + c) \wedge e') \vee e''$ for some c, e' and e'' such that e is in disjunctive normal form

  $e + a$
  $=$ {Induction Hypothesis}
  $(((\mathbf{x} + c) \wedge e') \vee e'') + a$
  $=$ {distributivity}
  $((\mathbf{x} + (c + a)) \wedge (e' + a)) \vee (e'' + a)$,
  which is in disjunctive normal form.

- $e_1 \vee e_2$:

  Induction Hypothesis: $e_1 = ((\mathbf{x}_1 + c_1) \wedge e_1') \vee e_1''$ for some c_1, e_1' and e_1'' such that e_1 is in disjunctive normal form, and similarly for e_2

  $e_1 \vee e_2$
  $=$ {Induction Hypothesis}
  $(((\mathbf{x}_1 + c_1) \wedge e_1') \vee e_1'') \vee (((\mathbf{x}_2 + c_2) \wedge e_2') \vee e_2'')$
  $=$ {associativity}
  $(((\mathbf{x}_1 + c_1) \wedge e_1') \vee (e_1'' \vee (((\mathbf{x}_2 + c_2) \wedge e_2') \vee e_2''))$
  which is in disjunctive normal form.

- $e_1 \wedge e_2$:

$$
\begin{aligned}
& e_1 \wedge e_2 \\
= & \{\text{Induction Hypothesis}\} \\
& (((\mathbf{x}_1 + a_1) \wedge e_1') \vee e_1'') \wedge (((\mathbf{x}_2 + a_2) \wedge e_2') \vee e_2'') \\
= & \{\text{distributivity}\} \\
& ((\mathbf{x}_1 + a_1) \wedge e_1' \wedge (\mathbf{x}_2 + a_2) \wedge e_2') \vee ((\mathbf{x}_1 + a_1) \wedge e_1' \wedge e_2'') \vee ((\mathbf{x}_2 + a_2) \wedge e_2' \wedge e_1'') \vee (e_1'' \wedge e_2''), \\
& \text{which is in disjunctive normal form.}
\end{aligned}
$$

It follows that all simple integer expressions can be rewritten to an equivalent simple integer expression in disjunctive normal form. □

Corollary 19.
Any simple integer equation system can be rewritten to an equivalent integer equation system in disjunctive normal form.

Proof. This follows directly from the definition of simple integer equation system and theorem 18. □

For the remainder of this paper, we assume all simple integer expressions and all simple integer equation systems to be in disjunctive normal form.

5 From boolean to integer equation systems

We explore the relation between boolean and integer equation systems. It is known that boolean equation systems can be transformed into parity games (cf. [3]) and parity games can be transformed into integer equation systems (cf. [2]). In this section, we examine a direct translation from boolean to integer equation systems, without the translation to parity games in between.

5.1 Transformation via parity games

5.1.1 Parity games

A *parity game* is a graph game played by two players, called $\vee$ and $\wedge$, on a game graph in which each vertex is assigned an integer priority. A play is an infinite path in the graph in which a player does a step if a token is on a vertex for that player. Player $\vee$ wins a play if the lowest priority that occurs infinitely often in that play is even, otherwise player $\wedge$ wins the play. A vertex is *winning* for a player if that player can force every play starting at that vertex to be winning for that player.

A game graph is a directed graph $G = (V, E, p)$, where V is a set of vertices, $E \subseteq V \times V$ is a total edge relation, i.e. for each $v \in V$ there is a $w \in V$ such that $(v, w) \in E$, and $p : V \to \mathbb{N}$ is a priority function, assigning a non-negative integer priority to each vertex. We restrict ourselves to games on finite graphs.

A vertex $v \in V$ is winning for a player if that player can force every infinite play starting from v to be winning for himself. It is known that every vertex is either winning for player $\vee$ or for player $\wedge$ [3].

5.1.2 Boolean equation systems to parity games

Now we see how boolean equation systems in standard recursive form can be transformed into a parity game. This transformation is taken from Keiren [4]. We let the vertices V correspond to bound variables in $\mathcal{E}$. For simplicity, we refer to the vertices with the same name as the corresponding variable. We put an edge from one vertex $\mathbf{x}$ to a vertex $\mathbf{y}$ if and only if $\mathbf{y}$ occurs in the defining expression for $\mathbf{x}$ in $\mathcal{E}$. We put a vertex in $V_\wedge$ if the corresponding equation's right hand side is conjunctive, and we put it in $V_\vee$ otherwise. The priority of a vertex $\mathbf{x}$ is equal to the *rank* of the corresponding variable $\mathbf{x} \in \mathsf{bnd}(\mathcal{E})$, defined as follows:

$$\begin{aligned}
\mathsf{rank}(\mathbf{x}) &= \mathsf{rank}_{\nu,\mathbf{x}}(\mathcal{E}) \\
\mathsf{rank}_{\sigma,\mathbf{x}}(\epsilon) &= 0 \\
\mathsf{rank}_{\sigma,\mathbf{x}}((\sigma'\mathbf{y} = e)\ \mathcal{E}) &= \begin{cases} 0 & \text{if } \sigma = \sigma' \text{ and } \mathbf{x} = \mathbf{y} \\ \mathsf{rank}_{\sigma,\mathbf{x}}(\mathcal{E}) & \text{if } \sigma = \sigma' \text{ and } \mathbf{x} \neq \mathbf{y} \\ 1 + \mathsf{rank}_{\sigma',\mathbf{x}}((\sigma'\mathbf{y} = e)\ \mathcal{E}) & \text{if } \sigma \neq \sigma' \end{cases}
\end{aligned}$$

Note that $\mathsf{rank}(\mathbf{x})$, and thus the priority of the vertex in the translation of $\mathbf{x}$ is odd if and only if $\mathbf{x}$ is defined in a least fixed point equation.

What is important for us is the following theorem on the relation between the value of $\mathbf{x}$ in the solution of an equation system $\mathcal{E}$ and for which player the vertex $\mathbf{x}$ is winning in the parity game obtained by transformation:

Theorem 20 (Keinänen [3])**.**
Let $\mathcal{G}(\mathcal{E})$ be the transformation of $\mathcal{E}$ to a parity game. Vertex $\mathbf{x}$ is winning for player $\vee$ in $\mathcal{G}(\mathcal{E})$ if and only if $[\![\mathcal{E}]\!]\theta(\mathbf{x}) = \mathsf{true}$.

Let us look at an example of this transformation:

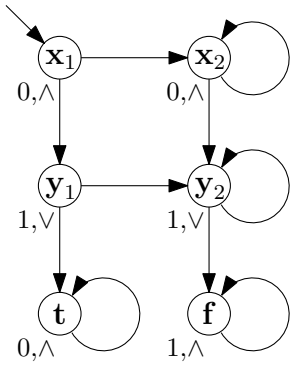
Example 21. We transform the boolean equation system of example 1 into a parity game. Let

$$\mathcal{E} = (\nu\mathbf{x}_1 = \mathbf{x}_2 \wedge \mathbf{y}_1)\ (\nu\mathbf{x}_2 = \mathbf{x}_2 \wedge \mathbf{y}_2)\ (\mu\mathbf{y}_1 = \mathbf{y}_2 \vee \mathsf{true})\ (\mu\mathbf{y}_2 = \mathbf{y}_2 \vee \mathsf{false})$$

We first have to transform this into standard recursive form:

$$\mathcal{E}' = (\nu\mathbf{t} = \mathbf{t})\ (\nu\mathbf{x}_1 = \mathbf{x}_2 \wedge \mathbf{y}_1)\ (\nu\mathbf{x}_2 = \mathbf{x}_2 \wedge \mathbf{y}_2)\ (\mu\mathbf{y}_1 = \mathbf{y}_2 \vee \mathbf{t})\ (\mu\mathbf{y}_2 = \mathbf{y}_2 \vee \mathbf{f})\ (\mu\mathbf{f} = \mathbf{f})$$

The vertices of the parity game correspond to the bound variables: $V = \{\mathbf{t}, \mathbf{x}_1, \mathbf{x}_2, \mathbf{y}_1, \mathbf{y}_2, \mathbf{f}\}$. The rank of the variables bound in the first three equations is 0, so $p(\mathbf{t}) = p(\mathbf{x}_1) = p(\mathbf{x}_2) = 0$. The rank of the latter three is 1, so $p(\mathbf{y}_1) = p(\mathbf{y}_2) = p(\mathbf{f}) = 1$. The vertices corresponding to variables with a disjunctive defining equation are put in $V_\vee = \{\mathbf{y}_1, \mathbf{y}_2\}$. The others are put in $V_\wedge = \{\mathbf{t}, \mathbf{y}_1, \mathbf{y}_2, \mathbf{f}\}$. We put an edge from one variable to another if the second occurs in the first's defining expression: $E = \{(\mathbf{t},\mathbf{t}), (\mathbf{x}_1,\mathbf{x}_2), (\mathbf{x}_1,\mathbf{y}_1), (\mathbf{x}_2,\mathbf{x}_2), (\mathbf{x}_2,\mathbf{y}_2), (\mathbf{y}_1,\mathbf{y}_2), (\mathbf{y}_1,\mathbf{t}), (\mathbf{y}_2,\mathbf{y}_2), (\mathbf{y}_2,\mathbf{f}), (\mathbf{f},\mathbf{f})\}$.



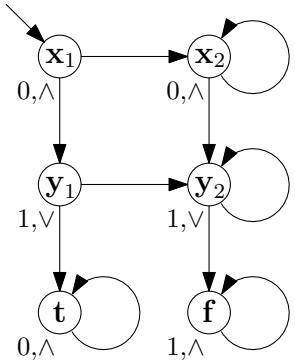
5.1.3 Parity games to integer equation systems

Gawlitza [2] proved the following translation from parity games to integer equation systems: for every vertex $\mathbf{x} \in V$, introduce a new equation $(\mu\mathbf{x} = ((e+c) \wedge m) \vee -m)$. Here, e is a subexpression constructed as follows: if the vertex $\mathbf{x}$ is in $V_\vee$, the subexpression e is a disjunction over the variables corresponding to the outgoing neighbors of vertex $\mathbf{x}$, i.e. $\bigvee\{\mathbf{y} \mid (\mathbf{x}, \mathbf{y}) \in E\}$. Similarly for when $\mathbf{x}$ is in $V_\wedge$. The constant $c = -(-|V|)^{d-p(\mathbf{x})}$, where d is the maximum priority of all vertices, is chosen such that lower priorities have greater constants. This is because vertices with lower priorities in parity games are more "important" than those with higher priorities. Furthermore, the sign of the constant corresponds to the priority being even or odd: c is positive if and only if $p(\mathbf{x})$ is even. The use of the constant $m = |V|^{d+1}$ is a way to enforce all solutions are in a finite interval, $[-m, m]$, without influencing the signs of the variables of the least solution. This sign is all-important, because of the following relation between a parity game and its transformation to an integer equation system:

Theorem 22 (Gawlitza [2])**.**
Let $\mathcal{E}$ be the transformation of parity game $\mathcal{G}$ and let θ be the least solution of $\mathcal{E}$. It holds that vertex $\mathbf{x}$ is winning for player $\vee$ if and only if $\theta(\mathbf{x})$ is positive.

We now look at an example of this transformation:

Example 23. Consider the parity game obtained in example 21:



We introduce an equation $(\mu\mathbf{v} = ((e_\mathbf{v} + c_\mathbf{v}) \wedge m) \vee -m)$ for each vertex $\mathbf{v} \in V$, where $m = |V|^{d+1} = 6^2 = 36$ and $c_\mathbf{v} = -(-|V|)^{d-p(\mathbf{v})} = -(-6)^{1-p(\mathbf{v})}$:

$$
\begin{aligned}
&(\mu\mathbf{t} = ((e_\mathbf{t} + 6) \wedge 36) \vee -36)\\
&(\mu\mathbf{x}_1 = ((e_{\mathbf{x}_1} + 6) \wedge 36) \vee -36)\\
&(\mu\mathbf{x}_2 = ((e_{\mathbf{x}_2} + 6) \wedge 36) \vee -36)\\
&(\mu\mathbf{y}_1 = ((e_{\mathbf{y}_1} + (-1)) \wedge 36) \vee -36)\\
&(\mu\mathbf{y}_2 = ((e_{\mathbf{y}_2} + (-1)) \wedge 36) \vee -36)\\
&(\mu\mathbf{f} = ((e_\mathbf{f} + (-1)) \wedge 36) \vee -36)
\end{aligned}
$$

The expressions $e_\mathbf{v}$ are as follows: for vertices owned by player $\vee$, it is the disjunction of the endpoints of all edges starting from $\mathbf{v}$. Similarly for vertices owned by $\wedge$, where the expression is

$$
\begin{aligned}
T(\mathcal{E}) &= T'_{m,|\mathcal{E}|,d,\nu}(\mathcal{E}), \\
&\quad \text{where } m = |\mathcal{E}|^{d+1} \\
&\quad \text{and } d \text{ is the maximum rank in } \mathcal{E}
\end{aligned}
$$

$$
\begin{aligned}
T'_{m,n,r,\sigma}(\epsilon) &= \epsilon \\
T'_{m,n,r,\sigma}((\sigma' X = f)\ \mathcal{E}) &= \Big(\mu X = ((f + (-(-n)^{r'})) \wedge m) \vee -m\Big)\ T'_{m,n,r',\sigma'}(\mathcal{E}), \\
&\quad \text{where } r' = \begin{cases} r & \text{, if } \sigma = \sigma' \\ r-1 & \text{, otherwise} \end{cases}
\end{aligned}
$$

Table 2: The transformation from boolean equation systems in standard recursive form to integer equation systems, by composing the transformations from via parity games.

a conjunction over the same variables. This gives us the following integer equation system:

$$
\begin{aligned}
&(\mu \mathbf{t} = ((\mathbf{t} + 6) \wedge 36) \vee -36) \\
&(\mu \mathbf{x}_1 = (((\mathbf{x}_2 \wedge \mathbf{y}_1) + 6) \wedge 36) \vee -36) \\
&(\mu \mathbf{x}_2 = (((\mathbf{x}_2 \wedge \mathbf{y}_2) + 6) \wedge 36) \vee -36) \\
&(\mu \mathbf{y}_1 = (((\mathbf{y}_2 \vee \mathbf{t}) + (-1)) \wedge 36) \vee -36) \\
&(\mu \mathbf{y}_2 = (((\mathbf{y}_2 \vee \mathbf{f}) + (-1)) \wedge 36) \vee -36) \\
&(\mu \mathbf{f} = ((\mathbf{f} + (-1)) \wedge 36) \vee -36)
\end{aligned}
$$

5.2 Direct transformation

By composing the transformation from boolean equation systems in standard recursive form to parity games, and the transformation from parity games to integer equation systems, we reach the direct transformation T of table 2.

Theorem 24.
The transformation of table 2 is sound, i.e. for all $\mathbf{x} \in \mathsf{bnd}(\mathcal{E})$, it holds that $\theta'(\mathbf{x}) > 0$ if and only if $\theta(\mathbf{x}) = \mathsf{true}$, where θ' is the least solution of the result of $T(\mathcal{E})$ and θ the solution of $\mathcal{E}$.

Proof. Theorem 20 states that $\theta(\mathbf{x}) = \mathsf{true}$ iff. $\mathbf{x}$ is winning for for player $\vee$ in the parity game, and theorem 22 states that $\mathbf{x}$ is winning for for player $\vee$ in the parity game iff. $\theta'(\mathbf{x}) < 0$. The transformation T is the composition of these transformations. The result follows by transitivity of bi-implication. □

Note that the resulting integer equation system of this transformation is in disjunctive normal form.

Now we transform the same example as before, only using the direct transformation:

Example 25. Consider the following boolean equation system $\mathcal{E}$ in standard recursive form:

$$
\mathcal{E} = (\nu \mathbf{t} = \mathbf{t})\ (\nu \mathbf{x}_1 = \mathbf{x}_2 \wedge \mathbf{y}_1)\ (\nu \mathbf{x}_2 = \mathbf{x}_2 \wedge \mathbf{y}_2)\ (\mu \mathbf{y}_1 = \mathbf{y}_2 \vee \mathbf{t})\ (\mu \mathbf{y}_2 = \mathbf{y}_2 \vee \mathbf{f})\ (\mu \mathbf{f} = \mathbf{f})
$$

The maximum rank d is 1 and thus $m = |\mathcal{E}|^{d+1} = 6^2 = 36$. We transform $\mathcal{E}$ as follows:

$$
\begin{aligned}
& T(\mathcal{E}) \\
= {} & T'_{36,6,1,\nu}(\mathcal{E}) \\
= {} & T'_{36,6,1,\nu}(\\
& \quad (\nu \mathbf{t} = \mathbf{t}) \\
& \quad (\nu \mathbf{x}_1 = \mathbf{x}_2 \wedge \mathbf{y}_1) \\
& \quad (\nu \mathbf{x}_2 = \mathbf{x}_2 \wedge \mathbf{y}_2) \\
& \quad (\mu \mathbf{y}_1 = \mathbf{y}_2 \vee \mathbf{t}) \\
& \quad (\mu \mathbf{y}_2 = \mathbf{y}_2 \vee \mathbf{f}) \\
& \quad (\mu \mathbf{f} = \mathbf{f}) \\
&) \\
= {} & (\mu \mathbf{t} = ((\mathbf{t} + 6) \wedge 36) \vee -36) \\
& T'_{36,6,1,\nu}((\nu \mathbf{x}_1 = \mathbf{x}_2 \wedge \mathbf{y}_1)\ (\nu \mathbf{x}_2 = \mathbf{x}_2 \wedge \mathbf{y}_2)\ (\mu \mathbf{y}_1 = \mathbf{y}_2 \vee \mathbf{t})\ (\mu \mathbf{y}_2 = \mathbf{y}_2 \vee \mathbf{f})\ (\mu \mathbf{f} = \mathbf{f})) \\
= {} & \vdots \\
= {} & (\mu \mathbf{t} = ((\mathbf{t} + 6) \wedge 36) \vee -36) \\
& (\mu \mathbf{x}_1 = (((\mathbf{x}_2 \wedge \mathbf{y}_1) + 6) \wedge 36) \vee -36) \\
& (\mu \mathbf{x}_2 = (((\mathbf{x}_2 \wedge \mathbf{y}_2) + 6) \wedge 36) \vee -36) \\
& T'_{36,6,0,\mu}((\mu \mathbf{y}_1 = \mathbf{y}_2 \vee \mathbf{t})\ (\mu \mathbf{y}_2 = \mathbf{y}_2 \vee \mathbf{f})\ (\mu \mathbf{f} = \mathbf{f})) \\
= {} & \vdots \\
& (\mu \mathbf{t} = ((\mathbf{t} + 6) \wedge 36) \vee -36) \\
& (\mu \mathbf{x}_1 = (((\mathbf{x}_2 \wedge \mathbf{y}_1) + 6) \wedge 36) \vee -36) \\
& (\mu \mathbf{x}_2 = (((\mathbf{x}_2 \wedge \mathbf{y}_2) + 6) \wedge 36) \vee -36) \\
& (\mu \mathbf{y}_1 = (((\mathbf{y}_2 \vee \mathbf{t}) + (-1)) \wedge 36) \vee -36) \\
& (\mu \mathbf{y}_2 = (((\mathbf{y}_2 \vee \mathbf{f}) + (-1)) \wedge 36) \vee -36) \\
& (\mu \mathbf{f} = ((\mathbf{f} + (-1)) \wedge 36) \vee -36)
\end{aligned}
$$

This is indeed equal to what was obtained in example 23.

6 Gauss elimination

We study the relation between boolean equation systems and integer equation systems. To see how ideas from one formalism can be transferred to the other, we consider the algorithm of Gauss elimination.

6.1 General procedure

Gauss elimination is a way to solve systems of equations. It consists of two different parts: local resolution and substitution. Local resolution is applied to a single equation and aims to remove the defined variable from its definition. It rewrites an equation $(\mu \mathbf{x} = e)$, where $\mathbf{x}$ can occur in e, to some $(\mu \mathbf{x} = f)$, where $\mathbf{x}$ does not occur in f. This means that the solution for $\mathbf{x}$ only depends on the other variables. This value can then be found by solving the system only for those other variables. Substitution is the other part of Gauss elimination, in which obtained results are

substituted for variables. For example, it can use an equation $(\mu \mathbf{x} = e)$ to remove occurrences of $\mathbf{x}$ from the rest of the system. These two transformations can be used in any order, i.e. it is not necessary to do local resolution steps first and then only do substitutions.

One can view the general procedure of Gauss elimination as the combination of two types of transformations on a system of equations, where the solution of the system of equations is kept invariant, i.e. the solution of the equation systems before and after a transformation are equal. The two types of transformations in Gauss elimination are sufficient to transform the original equation system to one where no variables occur on the right hand sides. Then the solution to the system is obvious. As the solution of the system was kept invariant, this is also the solution to the original system. Note that in itself, Gauss elimination is not yet a full algorithm: it is still very much left open to implementation when to apply which transformations on which equations.

The specific workings of Gauss elimination vary between different types of systems of equations. The transformations may not be applicable to just any equations, and also the specific form these transformations take are different between types of equation systems. This shall be made clear by looking at Gauss elimination on boolean equation systems.

6.2 Gauss elimination in boolean equation systems

We shall first look at Gauss elimination in boolean equation systems, as described and proven sound by Mader [6]. As stated, it consists of two parts: local resolution and substitution.

6.2.1 Local resolution

In the local resolution part of Gauss elimination, we remove the variable being defined from its defining expression, so we change an equation $(\mu \mathbf{x} = e)$, where $\mathbf{x}$ occurs in e, to an equation $(\mu \mathbf{x} = f)$, where $\mathbf{x}$ does not occur in f.

In boolean equation systems, this can simply be done by replacing all occurrences of $\mathbf{x}$ by the boolean constant true if the defining equation is a greatest fixed point equation, and by false if it is a least fixed point equation. Furthermore, we allow at least the following boolean simplifications:

$$e \wedge \mathsf{true} = e \qquad e \vee \mathsf{true} = \mathsf{true}$$
$$e \wedge \mathsf{false} = \mathsf{false} \qquad e \vee \mathsf{false} = e$$

We shall look at a simple example:

Example 26. Consider the boolean equation system of example 1:

$$\mathcal{E} = (\nu \mathbf{x}_1 = \mathbf{x}_2 \wedge \mathbf{y}_1)\ (\nu \mathbf{x}_2 = \mathbf{x}_2 \wedge \mathbf{y}_2)\ (\mu \mathbf{y}_1 = \mathbf{y}_2 \vee \mathsf{true})\ (\mu \mathbf{y}_2 = \mathbf{y}_2 \vee \mathsf{false})$$

We can locally solve the second equation as follows:

$$\begin{aligned} & (\nu \mathbf{x}_2 = \mathbf{x}_2 \wedge \mathbf{y}_2) \\ = & \{\text{local resolution, greatest fixed point}\} \\ & (\nu \mathbf{x}_2 = \mathsf{true} \wedge \mathbf{y}_2) \\ = & \{\text{simplification}\} \\ & (\nu \mathbf{x}_2 = \mathbf{y}_2) \end{aligned}$$

Similarly, we can locally solve the last two equations and obtain the following boolean equation system:

$$(\nu \mathbf{x}_1 = \mathbf{x}_2 \wedge \mathbf{y}_1)\ (\nu \mathbf{x}_2 = \mathbf{y}_2)\ (\mu \mathbf{y}_1 = \mathsf{true})\ (\mu \mathbf{y}_2 = \mathsf{false})$$

6.2.2 Substitution

With substitution, we want to replace occurrences of variables with their definitions. In boolean equation systems, one cannot simply replace any variable by its definition. We illustrate this by a short example:

Example 27. Consider the following boolean equation system of example 6:

$$\mathcal{E} = (\mu \mathbf{x} = \mathbf{y})\ (\nu \mathbf{y} = \mathbf{x})$$

The solution to this boolean equation system is an environment θ where $\theta(\mathbf{x}) = \theta(\mathbf{y}) = \mathsf{false}$. Suppose we replaced the variable $\mathbf{x}$ in the second equation by its defining expression, given in the first equation. We would obtain

$$(\mu \mathbf{x} = \mathbf{y})\ (\nu \mathbf{y} = \underline{\mathbf{y}})$$

This system has a different solution, however, namely an environment θ' where $\theta(\mathbf{x}) = \theta(\mathbf{y}) = \mathsf{true}$. In Gauss elimination, the solution should be invariant, so this substitution is not sound.

Problems arise when open expressions, i.e. expressions with variables in them, are substituted for variables in an equation occurring *after* the defining equation. This is what went wrong in the last example. This is because the order of equations is relevant and the open expression is moved from one equation to one occurring at a different position in the system. This is not a problem with closed expressions, i.e. boolean constants, as their value is fixed regardless of their position in the system. Mader [6] showed that the following substitutions are sound: an open defining expression may replace the variable being defined only in equations occurring *before* the defining equation. A closed defining expression, however, may substitute the variable being defined in any other expression. More formally, it is known that the following theorem holds:

Theorem 28 (Mader [6])**.**
Consider the following three boolean equation systems:

$$\mathcal{E} = \mathcal{E}_1\ (\sigma \mathbf{x} = e)\ \mathcal{E}_2\ (\sigma' \mathbf{y} = e')\ \mathcal{E}_3$$
$$\mathcal{E}' = \mathcal{E}_1\ (\sigma \mathbf{x} = e)\ \mathcal{E}_2\ (\sigma' \mathbf{y} = e'[\mathbf{x} := e])\ \mathcal{E}_3$$
$$\mathcal{E}'' = \mathcal{E}_1\ (\sigma \mathbf{x} = e[\mathbf{y} := e'])\ \mathcal{E}_2\ (\sigma' \mathbf{y} = e')\ \mathcal{E}_3$$

The boolean equation system $\mathcal{E}'$ is equal to $\mathcal{E}$, only with a forward *substitution, and $\mathcal{E}''$ with a* backward *substitution. Let the respective solutions of these boolean equation systems be called θ, θ' and θ''. The following two statements hold:*

$$\theta = \theta', \text{ if } e \text{ is closed}$$
$$\theta = \theta''$$

Let us finish the example of solving the boolean equation system of example 1 using Gauss elimination:

Example 29. In example 27 we obtained:

$$\mathcal{E} = (\nu \mathbf{x}_1 = \mathbf{x}_2 \wedge \mathbf{y}_1)\ (\nu \mathbf{x}_2 = \mathbf{y}_2)\ (\mu \mathbf{y}_1 = \mathsf{true})\ (\mu \mathbf{y}_2 = \mathsf{false})$$

We can solve this using substitutions. We can do a backwards substitution on an open expression, if we substitute $\mathbf{x}_2$ in the first equation with its definition. We obtain:

$$(\nu \mathbf{x}_1 = \mathbf{y}_2 \wedge \mathbf{y}_1)\ (\nu \mathbf{x}_2 = \mathbf{y}_2)\ (\mu \mathbf{y}_1 = \mathsf{true})\ (\mu \mathbf{y}_2 = \mathsf{false})$$

By doing more substitutions, we can solve this equation system to:

$$(\nu \mathbf{x}_1 = \mathsf{false})\ (\nu \mathbf{x}_2 = \mathsf{false})\ (\mu \mathbf{y}_1 = \mathsf{true})\ (\mu \mathbf{y}_2 = \mathsf{false})$$

The solution is now obvious.

6.3 Gauss elimination in integer equation systems

Now we are ready to define Gauss elimination for integer equation systems.

6.3.1 Finite and infinite reduction

We shall first look at a weaker variant of local resolution, the reduction. Local resolution aims to remove *all* occurrences of a variable from its definition. In the paper by Gawlitza *et al.* [2] there is the claim that for *finite* variable assignments θ and *positive* constant c, it holds that

$$\theta(\mathbf{x}) = [\![((\mathbf{x}+c)\wedge e_1)\vee e_2]\!]\theta \quad \text{implies} \quad \theta(\mathbf{x}) = [\![e_1 \vee e_2]\!]\theta$$

We strengthen this claim to hold in both directions and generalized for any value of c as follows:

$$\theta(\mathbf{x}) = [\![((\mathbf{x}+c)\wedge e_1)\vee e_2]\!]\theta \quad \text{iff.} \quad \theta(\mathbf{x}) = \begin{cases} [\![e_2]\!]\theta & \text{, if } c \leq 0 \\ [\![e_1 \vee e_2]\!]\theta & \text{, if } c > 0 \end{cases}$$

We prove this later as a corollary to theorem 31. This equality is not a true local resolution method, as it only removes *one* occurrence of the defined variable $\mathbf{x}$. However, since the resulting expression, either e_2 or $e_1 \vee e_2$, is still in disjunctive normal form, the reduction method can again be applied until there are no more occurrences of the defined variable $\mathbf{x}$. Since there are only finitely many occurrences of $\mathbf{x}$ in the original expression, this terminates in a finite number of steps. As mentioned earlier, this reduction only applies when the least solution θ is known to be finite. Luckily, when translating a boolean equation system to an integer equation system, it is known that the value for the bound variables in the resulting integer equation system's least solution lie in the finite interval $[-m, m]$ and thus that they are finite. This means this reduction is useful for solving those integer equation systems. Because this reduction only works for finite solutions, we call this type of reduction the *finite* reduction.

For equation systems where the least solution is infinite, this is not sound:

Example 30. Consider the equation system consisting of only the equation $(\mu\mathbf{x} = ((\mathbf{x}{+}1)\wedge 0)\vee -\infty)$. By application of the reduction theorem, this sould have the same solution as the system $(\mu\mathbf{x} = 0 \vee -\infty)$. However, the least solution of the latter is an environment θ' where $\theta'(\mathbf{x}) = 0$, whereas the original integer equation system's least solution is an environment θ where $\theta(\mathbf{x}) = -\infty$.

We remedy this by defining the *infinite* reduction as follows: observe that the value of $\mathbf{x}$ in the least solution of equation $(\mu\mathbf{x} = ((\mathbf{x}+c)\wedge e_1)\vee e_2)$ cannot be greater than the value of e_2 in that least solution. The problems with the previous definition occur only when $[\![e_2]\!]\theta = -\infty$, where θ is the least solution of the system, when $-\infty$ is wrongfully not being considered the least solution for $\mathbf{x}$. Our observation is that if $[\![e_2]\!]\theta$ equals $-\infty$, the least solution's value for $\mathbf{x}$ is always equal to $-\infty$. We incorporate this into the reduction method as follows: for the least solution θ of $\mathcal{E}$ and $(\mu\mathbf{x} = ((x+c)\wedge e_1)\vee e_2)$ an equation of $\mathcal{E}$, we show that

$$\theta(\mathbf{x}) = [\![((\mathbf{x}+c)\wedge e_1)\vee e_2]\!]\theta \quad \text{iff.} \quad \theta(\mathbf{x}) = \begin{cases} [\![e_2]\!]\theta & \text{, if } c \leq 0 \text{ or } [\![e_2]\!]\theta = -\infty \\ [\![e_1 \vee e_2]\!]\theta & \text{, otherwise} \end{cases}$$

6.3.2 Soundness of reduction for one equation

We first prove the soundness of the infinite reduction method on integer equation systems with only one equation. We want to prove that the following holds, where θ is the least solution of

$\mathcal{E}$:

$$\theta(\mathbf{x}) = [\![((\mathbf{x}+c)\wedge e_1)\vee e_2]\!]\theta \quad \text{iff.} \quad \theta(\mathbf{x}) = \begin{cases} [\![e_2]\!]\theta & \text{, if } c \leq 0 \text{ or } [\![e_2]\!]\theta = -\infty \\ [\![e_1 \vee e_2]\!]\theta & \text{, otherwise} \end{cases}$$

We restate this to the following equivalent theorem, using the inductive definition of least solution introduced in section 4.1:

Theorem 31.
For any environment θ, it holds that

$$[\![(\mu\mathbf{x} = ((\mathbf{x}+c)\wedge e_1)\vee e_2)]\!]\theta = [\![(\mu\mathbf{x} = r)]\!]\theta,$$

$$\textit{where } r = \begin{cases} e_2 & \textit{if } c \leq 0 \textit{ or } [\![e_2]\!]([\![(\mu\mathbf{x} = ((\mathbf{x}+c)\wedge e_1)\vee e_2)]\!]\theta = -\infty \\ e_1 \vee e_2 & \textit{otherwise} \end{cases}$$

Proof. We distinguish three cases: $c \leq 0$, e_2 is $-\infty$, and the rest, i.e. where $c > 0$ and e_2 is not $-\infty$.

- Case $[\![e_2]\!]([\![(\mu\mathbf{x} = ((\mathbf{x}+c)\wedge e_1)\vee e_2)]\!]\theta) = -\infty$:

We first show that $-\infty$ is a solution to the system $(\mu\mathbf{x} = ((\mathbf{x}+c)\wedge e_1)$. Because it is the smallest element of $\overline{\mathbb{Z}}$, it is surely the least solution. We show that $[\![(-\infty + c)\wedge e_1]\!]\eta = -\infty$ for any environment η:

$$\begin{aligned} & [\![(-\infty + c)\wedge e_1]\!]\eta \\ = & \{\text{def. interpretation}\} \\ & (-\infty + c) \ \mathsf{min} \ [\![e_1]\!]\eta \\ = & \{\text{def. } + \text{ for } -\infty\} \\ & -\infty \ \mathsf{min} \ [\![e_1]\!]\eta \\ = & \{-\infty \text{ is the zero element of } \mathsf{min} \ \} \\ & -\infty \end{aligned}$$

The proof for this case is now as follows:

$$\begin{aligned} & [\![(\mu\mathbf{x} = ((\mathbf{x}+c)\wedge e_1)\vee e_2]\!]\theta \\ = & \{\text{def. least solution}\} \\ & \theta[\mathbf{x} := \underline{\mu f \in \overline{\mathbb{Z}}.[\![((\mathbf{x}+c)\wedge e_1)\vee e_2]\!](\theta[\mathbf{x} := f])}] \\ = & \{\text{def. interpretation}\} \\ & \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![(\mathbf{x}+c)\wedge e_1]\!](\theta[\mathbf{x} := f]) \ \mathsf{max} \ [\![e_2]\!](\theta[\mathbf{x} := f])] \end{aligned}$$

We would like to use the assumption that e_2 is $-\infty$, but the environments do not match yet. So we substitute the entire fixed point for f.

$$
\begin{aligned}
& \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![(\mathbf{x}+c) \wedge e_1]\!](\theta[\mathbf{x} := f]) \ \mathsf{max}\ [\![e_2]\!](\theta[\mathbf{x} := \underline{\mu f' \in \overline{\mathbb{Z}}.[\![((\mathbf{x}+c) \wedge e_1) \vee e_2]\!](\theta[\mathbf{x} := f'])}])] \\
= & \{\text{def. least solution}\} \\
& \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![(\mathbf{x}+c) \wedge e_1]\!](\theta[\mathbf{x} := f]) \ \mathsf{max}\ [\![e_2]\!]([\![(\mu \mathbf{x} = ((\mathbf{x}+c) \wedge e_1) \vee e_2)]\!]\theta)] \\
= & \{\text{case } [\![e_2]\!]([\![(\mu \mathbf{x} = ((\mathbf{x}+c) \wedge e_1) \vee e_2)]\!]\theta) = -\infty\} \\
& \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![(\mathbf{x}+c) \wedge e_1]\!](\theta[\mathbf{x} := f]) \ \mathsf{max}\ -\infty] \\
= & \{-\infty \text{ is identity of } \mathsf{max}\ \} \\
& \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![(\mathbf{x}+c) \wedge e_1]\!](\theta[\mathbf{x} := f])] \\
= & \{\text{def. least solution}\} \\
& [\![(\mu \mathbf{x} = ((\mathbf{x}+c) \wedge e_1)]\!]\theta \\
= & \{-\infty \text{ is the least solution}\} \\
& \theta[\mathbf{x} := -\infty] \\
= & \{\text{case } [\![e_2]\!]([\![(\mu \mathbf{x} = ((\mathbf{x}+c) \wedge e_1) \vee e_2)]\!]\theta) = -\infty\} \\
& \theta[\mathbf{x} := [\![e_2]\!]([\![(\mu \mathbf{x} = ((\mathbf{x}+c) \wedge e_1) \vee e_2)]\!]\theta)] \\
= & \{-\infty \text{ is a solution}\} \\
& \theta[\mathbf{x} := [\![e_2]\!](\theta[\mathbf{x} := -\infty])] \\
= & \{f \text{ does not occur}\} \\
& \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![e_2]\!](\theta[\mathbf{x} := -\infty])] \\
= & \{-\infty \text{ is smallest element, monotonicity of interpretation}\} \\
& \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![e_2]\!](\theta[\mathbf{x} := f])] \\
= & \{\text{def. least solution}\} \\
& [\![(\mu \mathbf{x} = e_2)]\!]\theta
\end{aligned}
$$

- Case $c \leq 0$:

$$
\begin{aligned}
& [\![(\mu \mathbf{x} = ((\mathbf{x}+c) \wedge e_1) \vee e_2]\!]\theta \\
= & \{\text{def. least solution}\} \\
& \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![((\mathbf{x}+c) \wedge e_1) \vee e_2]\!](\theta[\mathbf{x} := f])] \\
= & \{\text{def. interpretation}\} \\
& \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.((f+c) \ \mathsf{min}\ [\![e_1]\!](\theta[\mathbf{x} := f])) \ \mathsf{max}\ [\![e_2]\!](\theta[\mathbf{x} := f])] \\
= & \{\text{def. least fixed point}\} \\
& \theta[\mathbf{x} := \mathsf{min}\{f \in \overline{\mathbb{Z}} \mid ((f+c) \ \mathsf{min}\ [\![e_1]\!](\theta[\mathbf{x} := f])) \ \mathsf{max}\ [\![e_2]\!](\theta[\mathbf{x} := f]) \leq f\}] \\
= & \{\text{property of minimum and maximum}\} \\
& \theta[\mathbf{x} := \mathsf{min}\{f \in \overline{\mathbb{Z}} \mid ((f + c \leq f) \text{ or } [\![e_1]\!](\theta[\mathbf{x} := f]) \leq f) \text{ and } [\![e_2]\!](\theta[\mathbf{x} := f]) \leq f\}]
\end{aligned}
$$

Because for any nonpositive c it always holds that $f + c \leq f$, we can simplify this expression dramatically:

$$
\begin{aligned}
& \theta[\mathbf{x} := \mathsf{min}\{f \in \overline{\mathbb{Z}} \mid [\![e_2]\!](\theta[\mathbf{x} := f]) \leq f\}] \\
= & \{\text{def. least fixed point}\} \\
& \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![e_2]\!](\theta[\mathbf{x} := f])] \\
= & \{\text{def. least solution}\} \\
& [\![(\mu \mathbf{x} = e_2)]\!]\theta
\end{aligned}
$$

- Case $c > 0$ and $[\![e_2]\!]([\![(\mu\mathbf{x} = ((\mathbf{x}+c) \wedge e_1) \vee e_2)\mathcal{E}]\!]\theta) > -\infty$:

$$
\begin{aligned}
& [\![(\mu\mathbf{x} = ((\mathbf{x}+c) \wedge e_1) \vee e_2]\!]\theta \\
= \; & \ldots \{\text{similar to the case } c \leq 0\} \\
& \theta[\mathbf{x} := \mathsf{min}\{f \in \overline{\mathbb{Z}} \mid ((f + c \leq f) \text{ or } [\![e_1]\!](\theta[\mathbf{x} := f]) \leq f) \text{ and } [\![e_2]\!](\theta[\mathbf{x} := f]) \leq f\}]
\end{aligned}
$$

The inequality $f + c \leq f$ holds if and only if f is either $-\infty$ or ∞:

$$
\begin{aligned}
& \theta[\mathbf{x} := \mathsf{min}\{f \in \overline{\mathbb{Z}} \mid (f \in \{-\infty, \infty\} \text{ or } [\![e_1]\!](\theta[\mathbf{x} := f]) \leq f) \text{ and } [\![e_2]\!](\theta[\mathbf{x} := f]) \leq f\}] \\
= \; & \{\infty \text{ is identity element of } \mathsf{min}\} \\
& \theta[\mathbf{x} := \mathsf{min}\{f \in \overline{\mathbb{Z}} \mid (f = -\infty \text{ or } [\![e_1]\!](\theta[\mathbf{x} := f]) \leq f) \text{ and } [\![e_2]\!](\theta[\mathbf{x} := f]) \leq f\}
\end{aligned}
$$

The value $-\infty$ is in this set if and only if $[\![e_2]\!](\theta[\mathbf{x} := f]) = -\infty$. By monotonicity of interpretation and because $[\![\mathcal{E}]\!]$ is the *least* solution, it must then be the case that $[\![e_2]\!]([\![(\mu\mathbf{x} = ((\mathbf{x}+c) \wedge e_1) \vee e_2)]\!]\theta) = -\infty$. This is inconsistent with the case we are considering and hence, $-\infty$ cannot be in this set. Now we can conclude:

$$
\begin{aligned}
& \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![e_1]\!](\theta[\mathbf{x} := f]) \; \mathsf{max} \; [\![e_2]\!](\theta[\mathbf{x} := f])] \\
= \; & \{\text{def. least fixed point}\} \\
& \theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![e_1 \vee e_2]\!](\theta[\mathbf{x} := f])] \\
= \; & \{\text{def. least solution}\} \\
& [\![(\mu\mathbf{x} = e_1 \vee e_2)]\!]\theta
\end{aligned}
$$

□

From this we can easily prove the soundness of the finite reduction:

Corollary 32.
For integer equation systems with finite *least solution θ, it holds that*

$$\theta(\mathbf{x}) = [\![((\mathbf{x}+c) \wedge e_1) \vee e_2]\!]\theta \quad \textit{iff.} \quad \theta(\mathbf{x}) = \begin{cases} [\![e_2]\!]\theta & \text{, if } c \leq 0 \\ [\![e_1 \vee e_2]\!]\theta & \text{, if } c > 0 \end{cases}$$

Proof. Consider the theorem 31, which states:

$$\theta(\mathbf{x}) = [\![((\mathbf{x}+c) \wedge e_1) \vee e_2]\!]\theta \quad \text{iff.} \quad \theta(\mathbf{x}) = \begin{cases} [\![e_2]\!]\theta & \text{, if } c \leq 0 \text{ or } [\![e_2]\!]\theta = -\infty \\ [\![e_1 \vee e_2]\!]\theta & \text{, otherwise} \end{cases}$$

Since θ is finite, the case where $[\![e_2]\!]\theta = -\infty$ is ruled out, as that would imply that $\theta(\mathbf{x}) = -\infty$ and thus that θ is not finite. What remains of theorem 31 is equal to what was to be shown. □

6.3.3 Expansion lemma

Now that we know the reduction methods are sound for an integer equation system of only one equation, we want to use this result within a larger integer equation system. So, we would like to use equalities on single equations in integer equation systems with more equations. More specifically, we want to ascertain that if $[\![\mathcal{E}]\!]\theta' = [\![\mathcal{E}']\!]\theta'$ holds for any environment θ', then it also holds that $[\![\mathcal{F}_1 \; \mathcal{E} \; \mathcal{F}_2]\!]\theta = [\![\mathcal{F}_1 \; \mathcal{E}' \; \mathcal{F}_2]\!]\theta$ for any θ.

Lemma 33 (Expansion lemma)**.**
If $[\![\mathcal{E}]\!]\theta' = [\![\mathcal{E}']\!]\theta'$ holds for any environment θ', then it also holds that $[\![\mathcal{F}_1 \; \mathcal{E} \; \mathcal{F}_2]\!]\theta = [\![\mathcal{F}_1 \; \mathcal{E}' \; \mathcal{F}_2]\!]\theta$ for any θ.

Proof. We already know that the ordering of equations does not influence the least solution. Therefore, it is sufficient to show that $[\![\mathcal{F}\ \mathcal{E}]\!]\theta = [\![\mathcal{F}\ \mathcal{E}']\!]\theta$, where $\mathcal{F} = \mathcal{F}_1\ \mathcal{F}_2$.

By induction on the structure of $\mathcal{F}$.

- ϵ:

$$\begin{aligned}
&[\![\epsilon\ \mathcal{E}]\!]\theta \\
=\ &\{\epsilon \text{ is the empty string}\} \\
&[\![\mathcal{E}]\!]\theta \\
=\ &\{\text{assumption}\} \\
&[\![\mathcal{E}']\!]\theta \\
=\ &\{\epsilon \text{ is the empty string}\} \\
&[\![\epsilon\ \mathcal{E}]\!]\theta
\end{aligned}$$

- $(\mu\mathbf{x} = e)\ \mathcal{F}'$:

Induction Hypothesis: $[\![\mathcal{F}'\ \mathcal{E}]\!]\theta' = [\![\mathcal{F}'\ \mathcal{E}']\!]\theta'$ for any θ'

$$\begin{aligned}
&[\![(\mu\mathbf{x} = e)\ \mathcal{F}'\ \mathcal{E}]\!]\theta \\
=\ &\{\text{def. least solution}\} \\
&[\![\mathcal{F}'\ \mathcal{E}]\!]\theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![e]\!]([\![\mathcal{F}'\ \mathcal{E}]\!]\theta[\mathbf{x} := f])] \\
=\ &\{\text{Induction Hypothesis}\} \\
&[\![\mathcal{F}'\ \mathcal{E}']\!]\theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![e]\!]([\![\mathcal{F}'\ \mathcal{E}]\!]\theta[\mathbf{x} := f])] \\
=\ &\{\text{Induction Hypothesis}\} \\
&[\![\mathcal{F}'\ \mathcal{E}']\!]\theta[\mathbf{x} := \mu f \in \overline{\mathbb{Z}}.[\![e]\!]([\![\mathcal{F}'\ \mathcal{E}']\!]\theta[\mathbf{x} := f])] \\
=\ &\{\text{def. least solution}\} \\
&[\![(\mu\mathbf{x} = e)\ \mathcal{F}'\ \mathcal{E}']\!]\theta
\end{aligned}$$

□

6.3.4 Soundness of reduction in an integer equation system

Using the expansion lemma, we can now apply the infinite reduction method on one equation in a larger system. Because the ordering of equations has no influence on the least solution, we can, without loss of generality, assume that we apply the infinite reduction method on the first equation of an integer equation system.

Theorem 34.
Infinite reduction is sound within a larger system:

$$[\![(\mu\boldsymbol{x} = ((\boldsymbol{x} + c) \wedge e_1) \vee e_2)\mathcal{E}]\!]\theta = [\![(\mu\boldsymbol{x} = r)\mathcal{E}]\!]\theta,$$

$$\text{where } r = \begin{cases} e_2 & \text{if } c \leq 0 \text{ or } [\![e_2]\!]([\![(\mu\boldsymbol{x} = ((\boldsymbol{x} + c) \wedge e_1) \vee e_2)\mathcal{E}]\!]\theta = -\infty \\ e_1 \vee e_2 & \text{otherwise} \end{cases}$$

Proof. This follows from soundness of the infinite reduction method on one equation (theorem 31) and the expansion lemma (theorem 6.3.3). □

However, this reduction is not a purely syntactic operation anymore and hence may not be very useful. First, e_2 needs to be solved before anything can be substituted. This is problematic, for example when $\mathbf{x}$ occurs in the expression e_2.

6.3.5 Local resolution by iteration

Another way to remove a variable from its defining expression, is by iteratively filling in an approximation until the result stabilizes. This is sound on complete lattices. In theorem 3 we have already proven that the extended integers together with the relation $\leq$ form a complete lattice.

The Kleene fixed point theorem states that for every monotone function over a complete lattice, the least fixed point can be found by transfinite iteration starting from the bottom element of the lattice. In our setting, this corresponds to finding the least fixed point for a variable on the left hand side of an equation, by repeatedly filling it in on the right hand side, starting from $-\infty$. We can find the least fixed point for $\mathbf{x}$ of an expression e, where $\mathbf{x}$ occurs freely, by taking as first approximation $\mathbf{x}_0 = -\infty$ and repeatedly calculating $\mathbf{x}_{i+1} = e[\mathbf{x} := \mathbf{x}_i]$. The limit of this procedure approaches the least fixed point for $\mathbf{x}$ of expression e. The procedure is summarized in the following pseudo-code:

▷ *Local resolution for* $\mu\mathbf{x} = e$

1. $i \leftarrow 0$
2. $\mathbf{x}_i \leftarrow -\infty$
3. **do** $\mathbf{x}_{i+1} \leftarrow e[\mathbf{x} := \mathbf{x}_i]$
4. $\quad i \leftarrow i + 1$
5. **until** $\mathbf{x}_i = \mathbf{x}_{i-1}$

Upon termination, the value of $\mathbf{x}_i$ is the expression sought after. It could be the case that this limit is never actually reached in a finite number of iterations, which means we still do not know the actual least fixed point value.

Example 35. Consider the following equation:

$$\mu\mathbf{x} = 0 \vee (\mathbf{x} + 1)$$

This iteration procedure yields:

$$\begin{aligned}
&\mathbf{x}_0 = -\infty \\
&\mathbf{x}_1 = 0 \\
&\mathbf{x}_2 = 1 \\
&\mathbf{x}_3 = 2 \\
&\quad\vdots
\end{aligned}$$

This does not terminate.

However, if the approximation does reach the least fixed point, it is clear that the following iterations are all equal: suppose approximation i yields the fixed point, then $\mathbf{x}_i = e[\mathbf{x} := \mathbf{x}_i]$ holds since it is a fixed point, and $\mathbf{x}_{i+1} = e[\mathbf{x} := \mathbf{x}_i]$ is equal to the previous approximation. We can detect, therefore, if the least fixed point is reached, by checking if two successive approximations are equal. Note that these approximations are not actual values in $\overline{\mathbb{Z}}$, but rather expressions with variables other than $\mathbf{x}$ still occurring in them. So, in order to check if the approximation is stable, we need to define equality on these expressions.

To define equality on expressions with variables in them, we construct a normal form for these expressions, such that if two expressions are e_1 and e_2 are syntactically equal, then they are interpreted equally in any environment θ. Syntactic equality between expressions e_1 and e_2 is

denoted $e_1 \equiv e_2$. Now we have yet to define a syntactic transformation $\mathcal{T}$ on expressions such that $\mathcal{T}(e_1) \equiv \mathcal{T}(e_2)$ implies $[\![e_1]\!]\theta = [\![e_2]\!]\theta$ for any environment θ. Note that this constraint trivially holds for the identity transformation, making this procedure sound. However, this is not very useful, as it often does not detect stabilization. We now explore a transformation which would detect more often when the approximation stabilizes. We conjecture that for this transformation, all stabilization is detected: $[\![e_1]\!]\theta = [\![e_2]\!]\theta$ for any environment θ, if *and only if*, $\mathcal{T}(e_1) \equiv \mathcal{T}(e_2)$.

We again assume the expressions are in disjunctive normal form. For the bottom level, where only constants, variables and additions with a constant are allowed, we define the following transformation $\mathcal{T}$:

$$\begin{aligned}
\mathcal{T}(c) &= c \\
\mathcal{T}(\mathbf{x}) &= \mathbf{x} \\
\mathcal{T}(c+e) &= \mathcal{T}(e+c) \\
\mathcal{T}(e+c) &= \begin{cases} \mathcal{T}_{\mathsf{var}}(e) & \text{if } \mathcal{T}_{\mathsf{var}} \not\equiv \epsilon \text{ and } \mathcal{T}_{\mathsf{const}}(e+c) = 0 \\ \mathcal{T}_{\mathsf{var}}(e) + \mathcal{T}_{\mathsf{const}}(e+c) & \text{if } \mathcal{T}_{\mathsf{var}} \not\equiv \epsilon \text{ and } \mathcal{T}_{\mathsf{const}}(e+c) \neq 0 \\ \mathcal{T}_{\mathsf{const}}(e+c) & \text{if } \mathcal{T}_{\mathsf{var}} \equiv \epsilon \end{cases}
\end{aligned}$$

$$\begin{aligned}
\mathcal{T}_{\mathsf{var}}(\mathbf{x}) &= \mathbf{x} & \mathcal{T}_{\mathsf{const}}(\mathbf{x}) &= \epsilon \\
\mathcal{T}_{\mathsf{var}}(c) &= \epsilon & \mathcal{T}_{\mathsf{const}}(c) &= c \\
\mathcal{T}_{\mathsf{var}}(c+e) &= \mathcal{T}_{\mathsf{var}}(e+c) & \mathcal{T}_{\mathsf{const}}(c+e) &= \mathcal{T}_{\mathsf{const}}(e+c) \\
\mathcal{T}_{\mathsf{var}}(e+c) &= \mathcal{T}_{\mathsf{var}}(e) & \mathcal{T}_{\mathsf{const}}(e+c) &= \mathit{Rep}(\mathit{Val}(\mathcal{T}_{\mathsf{const}}(e)) + \mathit{Val}(c))
\end{aligned}$$

Here we assume that we can do basic operations on constants, using the function Val to give the value of a string denoting a constant, and the function *Rep* to give a unique string representation of a constant. We do not go into details about this, but it is not hard to see that it is possible to do this syntactically. We similarly assume that we can syntactically determine equality and inequalities on constants, for which it is also easily verified that this is possible for extended integers.

The transformation for the cases maximum ($\vee$) and minimum ($\wedge$) are based on ordering the subexpressions and removing subexpressions which are implied by other subexpressions. First, we want to check if some subexpression e_1 is smaller or equal than some other subexpression e_2, regardless of the environment. This is a comparison on a *semantic* level, denoted by $\leq_{\mathsf{sem}}$. The inequality $e_1 \leq_{\mathsf{sem}} e_2$ holds if and only if $[\![e_1]\!]\theta \leq [\![e_2]\!]\theta$ for any θ. For example, $(\mathbf{x}-2) \leq_{\mathsf{sem}} (\mathbf{x}+3)$, as $[\![\mathbf{x}-2]\!]\theta \leq [\![\mathbf{x}+3]\!]\theta$ for any environment θ. Secondly, if expressions are incomparable on a semantic level, we want to order two subexpressions on a *syntactic* level. For this, we assume some total ordering $\leq_{\mathsf{var}}$ on the variables. The specific order is not important, but some ordering is necessary to put expressions which are different orderings of the same subexpressions in the same form, for example $\mathbf{x} \wedge \mathbf{y}$ and $\mathbf{y} \wedge \mathbf{x}$. This ordering is denoted $\leq_{\mathsf{syntax}}$.

$$
\begin{aligned}
c &\leq_{\mathsf{sem}} \mathbf{x} && \text{if } c = -\infty \\
\mathbf{x} &\leq_{\mathsf{sem}} c && \text{if } c = \infty \\
\mathbf{x} + c &\leq_{\mathsf{sem}} \mathbf{x} + c' && \text{if } Val(c) \leq Val(c') \\
\mathbf{x} + c &\leq_{\mathsf{sem}} \mathbf{x} && \text{if and only if } Val(c) \leq 0 \\
\mathbf{x} &\leq_{\mathsf{sem}} \mathbf{x} + c && \text{if and only if } Val(c) > 0 \\
\mathbf{x} + c &\leq_{\mathsf{sem}} c + \mathbf{x} \\
e_1 \wedge e_2 &\leq_{\mathsf{sem}} e_3 && \text{if } e_1 \leq_{\mathsf{sem}} e_3 \text{ or } e_2 \leq_{\mathsf{sem}} e_3 \\
e_1 &\leq_{\mathsf{sem}} e_2 \wedge e_3 && \text{if } e_1 \leq_{\mathsf{sem}} e_2 \text{ and } e_1 \leq_{\mathsf{sem}} e_3 \\
e_1 \vee e_2 &\leq_{\mathsf{sem}} e_3 && \text{if } e_1 \leq_{\mathsf{sem}} e_3 \text{ and } e_2 \leq_{\mathsf{sem}} e_3 \\
e_1 &\leq_{\mathsf{sem}} e_2 \vee e_3 && \text{if } e_1 \leq_{\mathsf{sem}} e_2 \text{ or } e_1 \leq_{\mathsf{sem}} e_3 \\
\mathbf{x} + c &\leq_{\mathsf{syntax}} \mathbf{y} + c' && \text{if } \mathbf{x} \leq_{\mathsf{var}} \mathbf{y} \\
(\mathbf{x} + c) \wedge e &\leq_{\mathsf{syntax}} (\mathbf{x} + c') \wedge e' && \text{if } Val(c) < Val(c') \text{ or } (Val(c) = Val(c)' \text{ and } e \leq_{\mathsf{syntax}} e')
\end{aligned}
$$

We use these relations to sort subexpressions of an expression. We use a variant of insertion sort where we use for example that $e \vee e'$ can be reduced to e', if we know e is less than e' in any environment. A similar property is used for the conjunctive case, where we only keep the smallest conjunct. The relation $\leq_{\mathsf{sem}}$ expresses which expression is smaller than which. If neither is smaller than the other on a semantic level, we resort to simply sorting the operands on a syntactic level. We use the symbol $\lneq\!\!\!\gneq$ to denote that two elements are incomparable.

$$
\begin{aligned}
\mathcal{T}(e_1 \vee e_2) &= \mathsf{insert}_{\vee}(\mathcal{T}(e_1), \mathcal{T}(e_2)) \\
\mathsf{insert}_{\vee}(e, e') &= \begin{cases} \mathcal{T}(e') & \text{if } e \leq_{\mathsf{sem}} \mathcal{T}(e') \\ e & \text{if } \mathcal{T}(e') \leq_{\mathsf{sem}} e \\ e \vee \mathcal{T}(e') & \text{if } \mathcal{T}(e') \lneq\!\!\!\gneq_{\mathsf{sem}} e \text{ and } (e \leq_{\mathsf{syntax}} e') \\ \mathcal{T}(e') \vee e & \text{if } \mathcal{T}(e') \lneq\!\!\!\gneq_{\mathsf{sem}} e \text{ and } (e' \leq_{\mathsf{syntax}} e) \end{cases} \\
\mathsf{insert}_{\vee}(e, e_1 \vee e_2) &= \begin{cases} \mathcal{T}(e_1 \vee e_2) & \text{if } e \leq_{\mathsf{sem}} \mathcal{T}(e_1) \\ \mathcal{T}(e \vee e_2) & \text{if } \mathcal{T}(e_1) \leq_{\mathsf{sem}} e \\ e \vee \mathcal{T}(e_1 \vee e_2) & \text{if } \mathcal{T}(e_1) \lneq\!\!\!\gneq_{\mathsf{sem}} e \text{ and } (e \leq_{\mathsf{syntax}} e_1) \\ \mathcal{T}(e_1) \vee \mathsf{insert}_{\vee}(e, \mathcal{T}(e_2) & \text{if } \mathcal{T}(e_1) \lneq\!\!\!\gneq_{\mathsf{sem}} e \text{ and } (e_1 \leq_{\mathsf{syntax}} e) \end{cases}
\end{aligned}
$$

The rules for $\mathsf{insert}_{\wedge}$ are similar, only we keep the other operand in case the two operands are semantically comparable.

This concludes the comparability of open integer expressions and thus the iteration approach to local resolution. The iteration procedure to locally solve $(\mu\mathbf{x} = e)$ is now as follows: let $\mathbf{x}_0 = -\infty$ and let $\mathbf{x}_{n+1} = e[\mathbf{x} := \mathbf{x}_n]$, until $\mathcal{T}(\mathbf{x}_i) \equiv \mathcal{T}(\mathbf{x}_{i+1})$. This $\mathbf{x}_i$ is then the local resolution.

Example 36. Let us consider a simple example. Consider the following equation:

$$(\mu\mathbf{x} = (\mathbf{y} + 3) \vee \mathbf{x})$$

We calculate the local resolution by iteration:

$$
\begin{aligned}
\mathbf{x}_0 &= -\infty \\
\mathbf{x}_1 &= ((\mathbf{y} + 3) \vee \mathbf{x})[\mathbf{x} := -\infty] \\
&= (\mathbf{y} + 3) \vee -\infty
\end{aligned}
$$

We want to know if this is equal to $-\infty$:

$$\begin{aligned}
&\mathcal{T}((\mathbf{y}+3) \vee -\infty) \\
&= \mathsf{insert}_{\vee}(\mathcal{T}(\mathbf{y}+3), \mathcal{T}(-\infty)) \\
&= \mathsf{insert}_{\vee}(\mathbf{y}+3, -\infty) \\
&= \mathbf{y}+3
\end{aligned}$$

This is not syntactically equivalent to $\mathcal{T}(-\infty) = -\infty$, so we have to do another iteration:

$$\begin{aligned}
\mathbf{x}_2 = \ & ((\mathbf{y}+3) \vee \mathbf{x})[\mathbf{x} := \mathbf{y}+3] \\
&= (\mathbf{y}+3) \vee (\mathbf{y}+3)
\end{aligned}$$

We already see that this is equivalent to $\mathbf{y}+3$. We find out by transforming again:

$$\begin{aligned}
&\mathcal{T}((\mathbf{y}+3) \vee (\mathbf{y}+3)) \\
&= \mathsf{insert}_{\vee}(\mathcal{T}(\mathbf{y}+3), \mathcal{T}(\mathbf{y}+3)) \\
&= \mathsf{insert}_{\vee}(\mathbf{y}+3, \mathbf{y}+3) \\
&= \mathbf{y}+3
\end{aligned}$$

As this is syntactically equivalent to itself, we are done. We have found that the following equations are equivalent:

$$(\mu\mathbf{x} = (\mathbf{y}+3) \vee \mathbf{x})$$

$$(\mu\mathbf{x} = \mathbf{y}+3)$$

6.3.6 Substitution

The other part of Gauss elimination is substitution. This aims to replace variables with their definitions in other equations than the defining equation. Whereas in local resolution we were only dealing with single equations, here we want to move results between equations.

First, we want to be able to move a variable's value from the environment to the integer equation system. If we know that $\theta(\mathbf{x}) = c$, where $c \in \overline{\mathbb{Z}}$ is a specific value, we want to be able to substitute c for $\mathbf{x}$ in an integer equation system and prove that this does not change the solution of that integer equation system.

Lemma 37.
In an environment θ where $\theta(\mathbf{x}) = c$ holds, substituting c for $\mathbf{x}$ in an equation system $\mathcal{E}$ does not change its solution. Formally, we prove that $[\![\mathcal{E}]\!]\theta[\mathbf{x} := c] = [\![\mathcal{E}[\mathbf{x} := c]]\!]\theta[\mathbf{x} := c]$, where $\mathbf{x} \notin \mathsf{bnd}(\mathcal{E})$.

Proof. We prove that $[\![a]\!]\theta[\mathbf{x} := c] = [\![a[\mathbf{x} := c]]\!]\theta[\mathbf{x} := c]$ for any right hand side expression a, from which the lemma immediately follows.

Proof by induction on the structure of a:

- $c' \in \overline{\mathbb{Z}}$:

$$\begin{aligned}
&[\![c']\!]\theta[\mathbf{x} := c] \\
&= c' \\
&= [\![c'[\mathbf{x} := c]]\!]\theta[\mathbf{x} := c]
\end{aligned}$$

- $\mathbf{x}$:

$$\begin{aligned}
&[\![\mathbf{x}]\!]\theta[\mathbf{x} := c] \\
&= \theta[\mathbf{x} := c](\mathbf{x}) \\
&= c \\
&= [\![c]\!]\theta[\mathbf{x} := c] \\
&= [\![\mathbf{x}[\mathbf{x} := c]]\!]\theta[\mathbf{x} := c]
\end{aligned}$$

- $a_1 \vee a_2$:

$$\begin{aligned}
&[\![a_1 \vee a_2]\!]\theta[\mathbf{x} := c] \\
&= [\![a_1]\!]\theta[\mathbf{x} := c] \vee [\![a_2]\!]\theta[\mathbf{x} := c] \\
&= \{\text{Induction Hypothesis}\} \\
&\quad [\![a_1[\mathbf{x} := c]]\!]\theta[\mathbf{x} := c] \vee [\![a_2[\mathbf{x} := c]]\!]\theta[\mathbf{x} := c] \\
&= [\![a_1[\mathbf{x} := c] \vee a_2[\mathbf{x} := c]]\!]\theta[\mathbf{x} := c]
\end{aligned}$$

- $a_1 \wedge a_2$ and $a_1 + a_2$: analogous to $a_1 \vee a_2$

□

Observe that we can prove in a similar way that expressions may be substituted for variables.

Lemma 38.
For any environment where $\theta(\mathbf{x}) = [\![e]\!]\theta$ holds, the expressions e' and $e'[\mathbf{x} := e]$ are equivalent in that environment. It holds that $[\![e']\!]\theta = [\![e'[\mathbf{x} := e]]\!]\theta$ when $\theta(\mathbf{x}) = [\![e]\!]\theta$.

Proof. The proof is very similar to the proof of lemma 37. □

Theorem 39.
Substituting a variable by its definition is sound, i.e. it does not change the solution. Formally, we prove that $[\![(\mu\boldsymbol{x} = e)\ \mathcal{E}]\!]\theta = [\![(\mu\boldsymbol{x} = e)\ \mathcal{E}[\mathbf{x} := e]]\!]\theta$, where ":=" on $\mathcal{E}$ denotes syntactic substitution on the right sides of all equations in $\mathcal{E}$.

Proof. Proof by structural induction.

- $(\mu\mathbf{x} = e)\ \epsilon$:

$$\begin{aligned}
&[\![(\mu\mathbf{x} = e)\ \epsilon]\!]\theta \\
&= \{\text{trivial, nothing to replace}\} \\
&\quad [\![(\mu\mathbf{x} = e)\ \epsilon[\mathbf{x} := e]]\!]\theta
\end{aligned}$$

- $(\mu \mathbf{x} = e)\ (\mu \mathbf{y} = e')\ \mathcal{E}$:

  Induction Hypothesis: $[\![\mathcal{E}]\!]\theta' = [\![\mathcal{E}[\mathbf{x} := e]]\!]\theta'$

  $[\![(\mu \mathbf{x} = e)\ (\mu \mathbf{y} = e')\ \mathcal{E}]\!]\theta$
  $= \{$reordering does not change least solution, theorem 11$\}$
  $[\![(\mu \mathbf{y} = e')\ (\mu \mathbf{x} = e)\ \mathcal{E}]\!]\theta$
  $= \{$def. least solution$\}$
  $[\![(\mu \mathbf{x} = e)\ \mathcal{E}]\!]\theta[\mathbf{y} := \mu f' \in \overline{\mathbb{Z}}.[\![e']\!]([\![(\mu \mathbf{x} = e)\ \mathcal{E}]\!]\theta[\mathbf{y} := f'])]$
  $= \{$Induction Hypothesis, by expansion lemma$\}$
  $[\![(\mu \mathbf{x} = e)\ \mathcal{E}[\mathbf{x} := e]]\!]\theta[\mathbf{y} := \mu f' \in \overline{\mathbb{Z}}.[\![e']\!]([\![(\mu \mathbf{x} = e)\ \mathcal{E}[\mathbf{x} := e]]\!]\theta[\mathbf{y} := f'])]$
  $= \{$lemma 38$\}$
  $[\![(\mu \mathbf{x} = e)\ \mathcal{E}[\mathbf{x} := e]]\!]\theta[\mathbf{y} := \mu f' \in \overline{\mathbb{Z}}.[\![e'[\mathbf{x} := e]]\!]([\![(\mu \mathbf{x} = e)\ \mathcal{E}[\mathbf{x} := e]]\!]\theta[\mathbf{y} := f'])]$
  $= \{$def. least solution$\}$
  $[\![(\mu \mathbf{y} = e'[\mathbf{x} := e])\ (\mu \mathbf{x} = e)\ \mathcal{E}[\mathbf{x} := e]]\!]\theta$
  $= \{$reordering does not change least solution, theorem 11$\}$
  $[\![(\mu \mathbf{x} = e)\ (\mu \mathbf{y} = e'[\mathbf{x} := e])\ \mathcal{E}[\mathbf{x} := e]]\!]\theta$

□

This concludes soundness of substitution and of Gauss elimination on integer equation systems.

7 Conclusion

We explored the relation between boolean and integer equation systems. We found a direct transformation from boolean to integer equation systems, by composing the transformations via parity games. This yielded a very straightforward transformation and we have proven that it is sound.

We then tried to apply the Gauss elimination method, used to solve boolean equation systems, to the formalism of integer equation systems. Gauss elimination consists of two parts: local resolution and substitution. For local resolution, we considered two different approaches: that of reduction and of iteration.

The reduction approach turned out to be successful for integer equation systems with finite solutions. The integer equation system obtained by translating a boolean to integer equation system yields such a system and we can use this approach to solve the system. Unfortunately, in the cases where the solution can be infinite, this approach was not very successful. It did give more insight into the problem and we now know the solution to the problem if we can determine if some subexpression solves to $-\infty$ or not. In specific instances, this may not be hard to do and then the reduction approach works.

The other approach is through iteration. This is a partial success: we have proven that this approach is sound, but the procedure does not always terminate. In instances where the procedure does terminate, this is a good approach to local resolution.

Lastly, we proved that substitution of variables by their definition is sound in integer equation systems. Furthermore, there are no restrictions as in boolean equation systems: the approach is sound regardless of the relative position of the equations.

8 Future research

To solve integer equation systems with Gauss elimination, we considered two approaches for local resolution: reduction and iteration. Further research could be done to remove the drawbacks of these approaches. We proved the reduction approach is sound, even on systems with infinite solutions, provided we can determine if some subexpression solves to $-\infty$. Further research could be done to determine if it does. This would make it possible to solve any integer equation system using Gauss elimination. The other approach, via iteration, does not necessarily terminate. It would be useful to characterize integer equation systems which cause such a nonterminating iteration, and transform them to some equivalent system which does terminate. This would complete the Gauss elimination procedure on integer equation systems.

References

[1] Thomas Gawlitza and Helmut Seidl, *Precise fixpoint computation through strategy iteration*, Proceedings of the 16th European conference on Programming (Berlin, Heidelberg), ESOP'07, Springer-Verlag, 2007, pp. 300–315.

[2] Thomas Gawlitza and Helmut Seidl, *Precise interval analysis vs. parity games*, Proceedings of the 15th international symposium on Formal Methods (Berlin, Heidelberg), FM '08, Springer-Verlag, 2008.

[3] Misa Keinänen, *Techniques for solving boolean equation systems*, Research Report A105, Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland, November 2006, Doctoral dissertation.

[4] Jeroen Keiren, *An experimental study of algorithms and optimisations for parity games, with an application to Boolean Equation Systems*, Master's thesis, Eindhoven University of Technology, July 2009.

[5] Jeroen Keiren and Tim Willemse, *Bisimulation minimisations for boolean equation systems*, Proceedings of the 5th international Haifa verification conference on Hardware and software: verification and testing (Berlin, Heidelberg), HVC'09, Springer-Verlag, 2011.

[6] Angelika Mader, *The modal μ-calculus, model checking, equation systems and Gauß elimination*, 1995.

[7] Angelika Mader, *Verification of modal properties using boolean equation systems*, Edition versal 8, Bertz Verlag, Berlin, 1997.

[8] Alfred Tarski, *A lattice-theoretical fixpoint theorem and its applications.*, Pac. J. Math. **5** (1955), 285–309 (English).