OPERATIONS ON FIXPOINT EQUATION SYSTEMS

THOMAS NEELE \(^a\) AND JACO VAN DE POL \(^b\)

\(^a\) Eindhoven University of Technology, The Netherlands
e-mail address: t.s.neele@tue.nl

\(^b\) Aarhus University, Denmark
e-mail address: jaco@cs.au.dk

ABSTRACT. We study operations on fixpoint equation systems (FES) over arbitrary complete lattices. We investigate under which conditions these operations, such as substituting variables by their definition, and swapping the ordering of equations, preserve the solution of a FES. We provide rigorous, computer-checked proofs. Along the way, we list a number of known and new identities and inequalities on extremal fixpoints in complete lattices.

1. INTRODUCTION

This paper deals with operations on systems of fixpoint equations over an arbitrary complete lattice. We investigate when these operations preserve the solution of the equations. An example of a system of equations is the set \( \mathcal{E} := \{ X = f(X,Y,Z), Y = g(X,Y,Z), Z = h(X,Y,Z) \} \). For most results, it is required that the functions \( f, g, h \) are monotonic in the given lattice. Such systems may well have multiple solutions. In order to specify particular solutions, we introduce specifications, for example \( \mathcal{S} := [\mu X, \nu Y, \mu Z] \), indicating for each variable whether we are interested in the minimal (\( \mu \)) or maximal (\( \nu \)) solution. The order of the variables in these specifications is relevant: the leftmost variable indicates the fixpoint with the highest priority. A Fixpoint Equation System (FES) [Mad97] is a pair \((\mathcal{E}, \mathcal{S})\), where \( \mathcal{E} \) is a set of equations, and \( \mathcal{S} \) is a specification. Several well known instances are obtained by instantiating the complete lattice.

Well-known instances of FES. Boolean Equation Systems (BES) arise as FES over the complete lattice \( \bot < T \), and were proposed in [And94, AV95] for solving the model checking and equivalence checking problems on finite labeled transition systems (LTS). BES received extensive study in [Mad97, MS03, GK04, Mat06].

An equivalent notion to BES is two-player parity games [EJ91], see [Mad97] for a proof. Algorithms for solving parity games receive a lot of attention, since this is one of the few problems which is in NP and in co-NP, but not known to be in P. Recently, it has been shown that parity games (and thus BES) can be solved in quasi-polynomial time [CJK+17]. This result has also been lifted to the general setting of FES on finite lattices [HS21, JMT22]. Other types of games can also be seen as an instance of FES, for example energy parity

\[\textbf{Key words and phrases:} \text{fixpoint equation system, complete lattice.}\]
games [CD12] are FES on the lattice \( \mathbb{Z} \rightarrow \{\bot, \top\} \), ordered pointwise. A modern parity game solver is Oink [vD18].

Parameterised Boolean Equation Systems (PBES, also known as first-order, or predicate BES) arise as FES over the powerset lattice \((2^D, \subseteq)\), with \(D\) some data type, typically representing the state space of a possibly infinite LTS. In [Mat98, GM99], PBES are proposed to encode the model-checking problem of first-order \(\mu\)-calculus on infinite LTSs; they are studied in more detail in [GW05b, GW05a]. An encoding of (branching) bisimulation of infinite LTSs in PBES is proposed in [CPvdPW07]. Various procedures that operate on PBES have been studied, for example to simplify [Nee22, OW10] or solve PBES [NWG20, WWWV22, PWW11]. Algorithms for solving some timed fragments of PBES automatically are studied in [ZC05]. PBES are implemented in the mCRL2 [BGK+19] and CADP [GLMS13] model checking toolsets.

Fixpoint Equation Systems over arbitrary complete lattices (FES) are defined in [Mad97, TC02]. Some works refer to the same concept as Hierarchical Equation Systems (HES) [Sei96, KNIU19], Systems of Fixpoint Equations [BKP20] or Nested Fixpoint Equations [JMT22]. In [ZC05] it is recognized that BES and PBES (and also Modal Equation Systems) are instances of FES. FESs are mainly useful to provide generic definitions for all these kinds of equation systems. We claim that the generic semantics of a FES is more elegant than the semantics of PBES, as given in e.g. [GW05a]. In particular, equations in FES are defined in a semantic manner as functions on valuations, rather than on syntactic expressions (possibly with binders). Another advantage of FES is that one can derive a number of basic theorems for equation systems over all lattices in one stride, like in Chapter 3 of Mader’s thesis [Mad97].

Abstract dependency graphs [EGLS19] are similar to FES, but variables range over a Noetherian partial order with a least element, instead of a complete lattice. When assuming every right-hand side is effectively computable, minimal fixpoints can be computed in an iterative fashion. Dependency graphs do not contain fixpoint alternations.

Contributions. Our main goal is to study basic operations on FES, related with substituting variables in the equations by their definition or final solution, or swapping the order of equations in the specification. Substitution operations form the basis of solving BES by so-called Gauss-elimination [Mad97]. Also for PBES, Gauss elimination plays a crucial role in their solution.

Reordering the variables in the specification is useful, because it may give rise to independent subspecifications that can be solved separately. Also, swapping the order of variables may bring down the number of alternations between \(\mu\) and \(\nu\), thus lowering the complexity of certain solution algorithms. Our results consist of equalities and inequalities between FES and are three-fold:

1. Results on substitution for BES and PBES are generalized to FES.
2. Results on swapping variables are generalized and sorted out, by weakening existing conditions, and by providing alternative conditions.
3. We provide rigorous proofs of all our results. Moreover, all proofs in this paper have been proof-checked mechanically by the Coq theorem prover [Ber08, S+22] (version 8.15) as well as the PVS theorem prover [OS08] (version 7.1). Our proofs are available online [NvdP23].
Overview. Section 2 summarizes our main results in a tabular form, and compares them in detail with the existing literature. We will provide rigorous definitions and proofs of all results in later sections. The formal definition and semantics of Fixpoint Equation systems is provided in Section 4. The proofs (Section 5, 6 and 7) are quite elementary. They are mainly based on induction (to deal with the recursive definition of FES semantics, Section 4.1) and on identities and inequalities on fixpoints in complete lattices. We reproved all needed facts on fixpoints in Section 3, in order to present a self-contained account. Finally, we highlight several aspects of our Coq and PVS formalisations in Section 8.

2. Summary – Examples – Related Work

We work in an arbitrary complete lattice, so that every monotonic function has a least and greatest fixpoint. In order to state the main results, we use the notation $[\mathcal{E}, \mathcal{S}]$, denoting the solution of the FES $(\mathcal{E}, \mathcal{S})$, where $\mathcal{E}$ is a set of equations and $\mathcal{S}$ a specification. The solution is a valuation, assigning values in the complete lattice to variables in $\mathcal{S}$. We use $\eta$ for arbitrary valuations. With $\text{dom}(\mathcal{S})$ we denote the set of variables in $\mathcal{S}$. Specifications $\mathcal{S}_1$ and $\mathcal{S}_2$ are disjoint if the intersection of their domains is empty.

To express some side conditions on dependencies between variables, we need some extra notation (formalized in Section 4.3). $\text{indep}(\mathcal{E}, X, Y)$ denotes the fact that $Y$ does not occur in the right-hand side of the equation for $X$ in $\mathcal{E}$. Otherwise, we write $X \rightarrow Y$ (direct dependency). This is lifted to specifications: with $\text{indep}(\mathcal{E}, \mathcal{S}_1, \mathcal{S}_2)$ we indicate that variables in $\mathcal{S}_1$ are independent of variables in $\mathcal{S}_2$. With $X \rightarrow Y$ (respectively $X \rightarrow^+ Y$), we denote that there is a path (resp. non-empty path) in the dependency graph of $(\mathcal{E}, \mathcal{S})$ from $X$ to $Y$. Finally, $\sigma$ and $\rho$ range over the fixpoint signs ($\mu/\nu$).

Table 1 summarizes our main results. We will discuss their relevance and compare them to previous work in Section 2.1-2.3. Table 2 contains some other useful facts on FES, discussed in Section 2.4.

2.1. Substituting Definitions and Solutions. Theorem 5.3 in this form is new. The operation $\text{unfold}$ substitutes the definition of $Y$ for all occurrences of $Y$ in the definition of $X$. This generalizes [Mad97, Lemma 6.3] (for BES only) and [GW05a, Lemma 18] (for PBES only) to FES over arbitrary complete lattices. Another generalization is that we allow that $X = Y$. That is, besides unfolding the $Y$’s in the definition of some $X$ preceding $Y$, one can even unfold $Y$ in its own definition. The proof for this case is more involved (cf. Lemma 5.2). For BES this is useless, but for PBES this is useful, and already used in [OW10, PWW11] to unfold PBESs to BESs.

Theorem 7.10 is a new result, generalizing the case where $\text{use}(Y) = \emptyset$ (i.e. $Y$ is in solved form, [Mad97, GW05a]). In general, one cannot unfold $Y$ in the definition of $X$, when $Y$ precedes $X$. However, if there is no dependency path from $Y$ to $X$, then a forward substitution is allowed. The proof is based on clever reordering of equations. The following example shows that this condition is necessary:

Example 2.1. Consider the following two Boolean Equation Systems:

<table>
<thead>
<tr>
<th>$B_1$</th>
<th>$B_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu Y = X$</td>
<td>$\nu Y = X$</td>
</tr>
<tr>
<td>$\mu X = Y$</td>
<td>$\mu X = X$</td>
</tr>
</tbody>
</table>
Unfolding $Y$ in the definition of $X$ in $B_1$ yields $B_2$. However, $B_1$ has the solution $(\top, \top)$, while the solution of $B_2$ is $(\bot, \bot)$. The reader can check this with the method described in Example 2.2.

Theorem 5.4 allows to substitute a partial solution in a FES. It occurs already in [Mad97, Lemma 3.19]. However, our proof is more direct. Mader suggests that a direct inductive argument is not possible, and proves the theorem by contradiction, constructing an infinite
The substitution theorems form the basis for solving BES and PBES by Gauss-elimination. They are called the global steps. Besides global steps, one needs local steps, to eliminate \( X \) from the right hand side of its own definition. For BES, a local step is trivial, because (only) in the Boolean lattice we have \( \mu X.f(X) = f(\bot) \) and \( \nu X.f(X) = f(\top) \). Local solution for PBES is much harder, and studied in [GW05a, OW10]. We stress that our results show that the global steps hold in any FES. However, effective local solution is specific to the underlying complete lattice.

Example 2.2. The following example shows the solution of a BES by Gauss elimination. Basically, one first substitutes definitions backwards using Theorem 5.3 (along the way, we use the identity \( Y \lor (Y \land X) \equiv Y \land X \)):

\[
\begin{align*}
\mu X &= Y \lor Z \\
\nu Y &= Z \\
\mu Z &= Y \land X
\end{align*}
\]

Next, one obtains \( X = \bot \) by a local elimination step in the first equation, using that \( Y \land \bot \equiv \bot \). This solution can then be substituted forward by Theorem 7.10, to obtain the full solution \( (\bot, \bot, \bot) \). In general, steps 1 and 2 must be mixed.

The next example shows a PBES where unfolding \( X \) in its own definition makes sense.

Example 2.3. Applying Theorem 5.3 to unfold \( X \) in its own definition, we get:

\[
\begin{align*}
\nu Y &= X(\top) \\
\mu X(b) &= (b \land Y) \lor X(\neg b)
\end{align*}
\]

Applying Theorem 5.3 again, to unfold \( X \) in \( Y \) yields \( \nu Y = Y \lor X(\top) \), hence by local resolution \( Y = \top \), hence \( X(b) = \top \).

2.2. Reordering Variables. Theorem 6.1 indicates that two adjacent variables with the same sign may be interchanged. This theorem occurs already in [Mad97, Lemma 3.21]. For PBES it is repeated in [GW05a, Lemma 21]. However, [Mad97, GW05a] don’t give a full proof, but refer to Bekić Lemma. In our proof, we show exactly how Theorem 5.3 reduces to our version of Bekić Rule (Lemma 3.1.9). In other works [Sei96, KNIU19], adjacent variables with the same sign are grouped in unordered blocks. No claim is made about the correctness of such a definition.

Theorem 6.9 shows the inequality that arises when interchanging adjacent variables with different sign. It occurs already in [Mad97, Lemma 3.23], but our proof is different. Our proof depends on a probably new inequality in fixpoint calculus, which we coin Bekić Inequality (Lemma 3.2.4).

Theorem 7.9 is a new result. It states that in the special case that \( X \) and \( Y \) are not on the same dependency loop, they can be interchanged without modifying the solution. This generalizes [GW05a, Lemma 19], which requires that the right-hand side of \( Y \) in \( E \) is a constant, i.e., \( \text{indep}(E, Y, Z) \) for all \( Z \).

Finally, Theorem 6.7 in this form is new. It allows to swap whole blocks of equations. Mader [Mad97, Lemma 3.22] claims a similar result, under the condition that both
indep(ℰ, S₁, S₂) and indep(ℰ, S₂, S₁). However, [GW05a] show a counter example to this. The repair in [GW05a, Lemma 22] requires that S₃ is empty. We show a stronger result: if S₃ is empty, only one of the requirements indep(S₂, S₁) or indep(S₂, S₁) is needed.

Notably, our result even applies to nonempty S₃, provided we have indep(S₁, S₂; S₃) (or its reverse), i.e. S₁ is also independent on the variables in S₃. Note that we allow arbitrary (dependent) alternations within S₁ and S₂, and even S₂ might depend on S₁. We lifted two other unnecessary restrictions: Surprisingly, this result doesn’t require monotonicity of ℰ. Also, the results in [Mad97, GW05a] are for individual equations only, while we can swap whole blocks at the same time.

We now show an application of swapping blocks to reduce the number of μ/ν-alternations.

Example 2.4. Consider the following four Boolean Equation Systems:

\[
\begin{array}{c|c|c|c}
B₃ & B₄ & B₅ & B₆ \\
\hline
\mu X = Y & \mu X = Y & \nu Z = W & \mu X = Y \\
\mu Y = X & \nu Z = W & \mu X = Y & \mu Y = X \\
\nu Z = W & \mu Y = X & \mu W = Z & \nu Z = W \\
\mu W = Z & \mu W = Z & \\
\end{array}
\]

For these BES, the dependency graph between the variables consists of two loops, X ↔ Y and Z ↔ W. In particular, we have indep(\{X, Y\}, \{Z, W\}). We want to transform B₃ to B₅, because it has less alternations. Theorem 6.1 cannot be applied, because the sign of Z is different from all the others.

Using Theorem 6.7 on individual equations, one can show that [B₃] = [B₄], because indep(Y, \{Z, W\}). However, one cannot derive [B₄] = [B₅] using Theorem 6.7, because neither indep(X, \{Z, Y, W\}), nor indep(\{Z, Y, W\}, X) holds. However, one can prove B₃ = B₅ directly with Theorem 6.7, by swapping block [X, Y] with Z, because indeed we have indep(\{X, Y\}, \{Z, W\}). Alternatively, one can observe that X \not→ Z, and apply Theorem 7.9 to deduce that B₄ = B₅ directly.

All theorems fail to prove the equivalence of B₃−₅ with B₆. However, Theorem 6.9 guarantees that [B₆] ≤ [B₃]. As a matter of fact, the solution of B₃, B₄ and B₅ is (X = ⊥, Y = ⊥, Z = ⊤, W = ⊤), while the solution of B₆ is (X = ⊥, Y = ⊥, Z = ⊥, W = ⊥). The reader may verify this by Gauss Elimination, cf. Example 2.2. This shows that the reordering theorems cannot easily be strengthened.

2.3. Swapping Signs. The inequality of Theorem 6.10 is well known and appears for instance in [Mad97, Lemma 3.24]. Theorem 7.7 is new. It shows that the sign of variable X is only relevant when X is the most relevant variable on a dependency loop.

2.4. Other Results. Along the way, we proved (and proof checked) several lemmas on FES that may be interesting on their own. For quick reference, we summarize these in Table 2. Here η denotes an arbitrary valuation, assigning elements from the complete lattice to all variables. All these lemmas occur in some form in the literature. Lemma 4.6 states that the semantics indeed returns a solution, and follows from [Mad97, Lemma 3.5]. Lemma 6.3 corresponds to [Mad97, Lemma 3.10] (which is not proved there) and [GW05a, Lemma 7]. Actually, [Mad97] has Lemma 6.6, which is equivalent according to our Theorem 6.7. Lemma 4.7 and 6.8 are from [Mad97, Lemma 3.14] as well, and Lemma 6.5 follows directly from Lemma 6.3. We included it here to stress that right congruence doesn’t hold in general.
Finally, we needed some basic results on fixpoints in complete lattices, cf. Lemma 3.1 and 3.2. The existence and definition of least and greatest fixpoints is due to Knaster (on sets) and Tarski (on complete lattices) [Tar55], see [LNS82] for a historical account. We (re)proved a number of identities (Lemma 3.1) and inequalities (Lemma 3.2) on fixpoint expressions. Most of these results are known. Lemma 3.1.1-6 can for instance be found in [Bac02]. Rule 9 (Bekič Equality) can be found in e.g. [dB80, Bek84], but stated in a different form, involving simultaneous fixpoints. We have not found in the literature the inequality in Lemma 3.2.4, which resembles Bekič equality on terms with mixed minimal and maximal fixpoints.

3. Fixpoint Laws in Complete Lattices

A partial order on a universe $U$ is a binary relation $\leq \subseteq U \times U$, which is reflexive ($\forall x. x \leq x$), anti-symmetric ($\forall x, y, x \leq y \land y \leq x \Rightarrow x = y$) and transitive ($\forall x, y, z. x \leq y \land y \leq z \Rightarrow x \leq z$), where in all cases $x, y, z \in U$.

Given partial orders $(U, \leq)$ and $(V, \leq)$, we define partial orders $(U \times V, \leq)$ and $(U \rightarrow V, \leq)$ pointwise: $(u_1, v_1) \leq (u_2, v_2)$ iff $u_1 \leq u_2$ and $v_1 \leq v_2$, and $f \leq g$ iff $\forall x \in U. f(x) \leq g(x)$. Function $f : U \rightarrow V$ is called monotonic, iff $\forall x, y. x \leq y \Rightarrow f(x) \leq f(y)$.

Given a set $X \subseteq U$, we define its set of lower bounds in $U$ as $lb(X) := \{ y \in U \mid \forall x \in X. x \leq y \}$. If $y \in lb(X)$ and $z \leq y$ for all $z \in lb(X)$, then $y$ is called the greatest lower bound of $X$. A complete lattice is a triple $(U, \leq, glb)$, where $\leq$ is a partial order, and $glb(X)$ returns the greatest lower bound of $X$ in $U$, for all (finite or infinite) $X \subseteq U$.

Given a complete lattice $(U, \leq, glb)$, define the partial order $(U, \geq)$, by $x \geq y$ iff $y \leq x$.

We define the set of upper bounds of $X \subseteq U$ by $ub(X) := \{ y \in U \mid \forall x \in X. y \geq x \}$. Define $lub(X) := glb(ub(X))$. Clearly, for all $y \in ub(X)$, $lub(X) \leq y$. But also $lub(X) \in ub(X)$, for if $x \in X$, then $x \in lb(ub(X))$, hence $x \leq glb(ub(X))$. So $lub(X)$ yields the least upper bound of $X$, and $(U, \geq, lub)$ is a complete lattice as well.

Given a complete lattice $(U, \leq, glb)$, we define the least fixpoint ($\mu$) and greatest fixpoint ($\nu$) of any function $f : U \rightarrow U$ (not only for monotonic) as follows:

\[
\mu(f) := \text{glb}\{x \mid f(x) \leq x\}
\]
\[
\nu(f) := \text{lub}\{x \mid x \leq f(x)\}
\]

For $\sigma \in \{\mu, \nu\}$, we abbreviate $\sigma(\lambda x.f(x))$ by $\sigma x.f(x)$. Note that by definition, $\nu$ in $(U, \leq, glb)$ equals $\mu$ in $(U, \geq, lub)$, so theorems on $(\mu, \leq)$ hold for $(\nu, \geq)$ as well “by duality”. Also note that $F : U \rightarrow U$ is monotonic in $(U, \leq)$ if and only if it is monotonic in $(U, \geq)$. A direct consequence of the definition of $\mu$ is the following principle (and its dual):

\[
f(x) \leq x \Rightarrow \mu(f) \leq x \quad \mu\text{-fixpoint induction}
\]
\[
x \leq f(x) \Rightarrow x \leq \nu(f) \quad \nu\text{-fixpoint induction}
\]

We now have the following identities on fixpoint expressions:

**Lemma 3.1.** Let $(U, \leq, glb)$ be a complete lattice. Let $\sigma \in \{\mu, \nu\}$, $A \in U$, and let $F, G \in U \rightarrow U$ and $H, K \in U \times U \rightarrow U$ be monotonic functions. Then:

1. $F(\sigma(F)) = \sigma(F)$ (computation rule)
2. $\sigma x. A = A$ (constant rule)
3. $\sigma x. F(G(x)) = F(\sigma x. G(F(x)))$ (rolling rule)
4. $\sigma x. F(F(x)) = \sigma x. F(x)$ (square rule)
5. $\sigma$ is monotonic (fixpoint monotonicity)
(6) \( \sigma x. H(x, x) = \sigma x. \sigma y. H(x, y) \) (diagonal rule)
(7) \( \sigma x. H(x, x) = \sigma x. H(x, H(x, x)) \) (unfolding rule)
(8) \( \sigma x. H(x, x) = \sigma x. H(x, \sigma x. H(x, x)) \) (solve rule)
(9) \( \sigma x. H(x, \sigma y. K(y, x)) = \sigma x. H(x, \sigma y. K(y, \sigma z. H(z, y))) \) (Bekič rule)

Proof. We first prove the theorem for \( \sigma = \mu \). By the observations above, the theorem then follows for \( \sigma = \nu \) as well (“by duality”).

(1) (a) Let \( y \) with \( F(y) \leq y \) be given. Then by fixpoint induction,
\[ \mu(F) \leq y. \]
By monotonicity, \( F(\mu(F)) \leq F(y) \leq y. \) Since \( y \) is arbitrary, \( F(\mu(F)) \) is a lower bound of \( \{ x \mid F(x) \leq x \} \). Hence \( F(\mu(F)) \leq \text{glb}(\{ x \mid F(x) \leq x \}) = \mu(F) \)
(b) \( F(\mu(F)) \leq \mu(F) \) by (a), so by monotonicity,
\[ F(F(\mu(F))) \leq F(\mu(F)). \]
By fixpoint induction, \( \mu(F) \leq F(\mu(F)). \)
Then by anti-symmetry \( F(\mu(F)) = \mu(F) \).
(2) Follows directly from (1) by taking \( F := \lambda x. A \) (which is monotonic)
(3) Obviously, \( \forall x. F(G(x)) \) and \( \lambda x. G(F(x)) \) are monotonic.
(a) By (1), \( F(G(F(\mu x. G(F(x)))))) = F(\mu x. G(F(x))) \). Hence by fixpoint induction,
\[ \mu x. F(G(x)) \leq F(\mu x. G(F(x))). \]
(b)
\[
G(F(G(\mu x. F(G(x)))))) \overset{(1)}{=} G(\mu x. F(G(x)))
\Rightarrow \text{by fixpoint induction}
\mu x. G(F(x)) \leq G(\mu x. F(G(x)))
\Rightarrow \text{by monotonicity}
F(\mu x. G(F(x))) \leq F(G(\mu x. F(G(x)))) \overset{(1)}{=} \mu x. F(G(x))
\]
By anti-symmetry, we obtain \( \mu x. F(G(x)) = F(\mu x. G(F(x))). \)
(4) (a) Using (1) twice, \( F(F(\mu x. F(x))) = F(\mu x. F(x)) = \mu x. F(x). \) So by fixpoint induction,
\( \mu x. F(F(x)) \leq \mu x. F(x). \)
(b) By (3), we get \( F(\mu x. F(F(x))) = \mu x. F(F(x)). \) Hence by fixpoint induction,
\( \mu x. F(x) \leq \mu x. F(x). \)
Then by anti-symmetry, \( \mu x. F(F(x)) = \mu x. F(x). \)
(5) Assume \( f \leq g. \) Let \( y \) with \( g(y) \leq y \) be given. Then \( f(y) \leq g(y) \leq y, \) so \( \mu(f) \leq y \) by fixpoint induction. Since \( y \) is arbitrary, \( \mu(f) \) is a lower bound for \( \{ x \mid g(x) \leq x \} \). By definition \( \mu(f) \) is its greatest lower bound, so \( \mu(f) \leq \mu(g) \).
(6) (a)
\[
H(\mu x. H(x, x), \mu x. H(x, x)) \overset{(1)}{=} \mu x. H(x, x)
\Rightarrow \text{by fixpoint induction, applied with } F := \lambda x. H(x, x)
\mu y. H(\mu x. H(x, x), y) \leq \mu x. H(x, x)
\Rightarrow \text{by fixpoint induction}
\mu x. \mu y. H(x, y) \leq \mu x. H(x, x)
(b) Let us abbreviate \( A := \mu x. \mu y. H(x, y) \). Using (5) one can show that \( \lambda x. \mu y. H(x, y) \) is monotonic. Then:

\[
A \overset{(1)}{=} \mu y. H(A, y) \overset{(1)}{=} H(A, \mu y. H(A, y))
\]

\( \Rightarrow \) (by congruence and both equations above)

\[
H(A, A) = H(A, \mu y. H(A, y)) = A
\]

\( \Rightarrow \) (by fixpoint induction)

\[
\mu x. H(x, x) \leq A = \mu x. \mu y. H(x, y)
\]

By anti-symmetry, we indeed get:

\[
\mu x. H(x, x) = \mu x. \mu y. H(x, y).
\]

(7) Using (4) on \( \lambda y. H(x, y) \) yields \( \mu x. \mu y. H(x, y) = \mu x. \mu y. H(x, H(x, y)) \). Applying (6) to both sides yields \( \mu x. H(x, x) = \mu x. H(x, H(x, x)) \).

(8) We use (6) twice on the function \( \lambda (y, x). H(x, y) \):

\[
\mu x. H(x, x) \overset{(6)}{=} \mu y. \mu x. H(x, y) \overset{(1)}{=} \mu x. H(x, \mu y. \mu x. H(x, y)) \overset{(6)}{=} \mu x. H(x, \mu x. H(x, x))
\]

(9) Define \( F(y) := \mu x. H(x, y) \) and \( G(y) := \mu x. K(x, y) \). Then:

\[
\mu y. F(G(y)) \overset{(3)}{=} F(\mu y. G(F(y)))
\]

\( \Rightarrow \) (by definition of \( F, G \))

\[
\mu y. \mu x. H(x, G(y)) = F(\mu y. \mu x. K(x, F(y)))
\]

\( \Rightarrow \) (by 6, applied to left- and right-hand side)

\[
\mu x. H(x, G(x)) = F(\mu y. K(y, F(y)))
\]

\( \Rightarrow \) (by definition of \( F, G \))

\[
\mu x. H(x, \mu y. K(y, x)) = \mu x. H(x, \mu y. K(y, \mu z. H(z, y)))
\]

A careful analysis shows that all these identities can be derived in an equational style from the identities 1, 3, 4 and 6. A natural question is whether all true equalities (with \( \mu \) as second order operation, and variables ranging over monotonic functions) can be derived from these four identities in an equational manner (thus excluding the fixpoint induction rule). We don’t know the answer, but we expect that at least the equations \( \mu x. F(x) = \mu x. F^p(x) \) are needed for all primes \( p \). Results from universal algebra don’t apply directly, due to the second order nature of the fixpoint operator.

By mixing least and greatest fixpoints, we also obtain a number of inequalities. In particular, 4 is new, as far as we know. Note the similarity of (4) with Bekiçi Rule, Lemma 3.1. We will call (4) Bekiçi Inequality.

**Lemma 3.2.** Let \((U, \leq, \mathrm{glb})\) be a complete lattice. Let \( \sigma \in \{\mu, \nu\} \), \( A \in U \), and let \( F, G : U \to U \) and \( H, K : U \times U \to U \) be monotonic functions. Then:

1. \( \mu(F) \leq \nu(F) \)
2. (a) \( \mu x. x \leq A \)
   (b) \( A \leq \nu x. x \)
3. \( \mu x. \nu y. H(x, y) \leq \nu y. \mu x. H(x, y) \)
(4) (a) \( \mu x. H(x, \nu y. K(y, x)) \leq \mu x. H(x, \nu y. K(y, \mu x. H(x, y))) \)

(b) \( \nu x. H(x, \mu y. K(y, x)) \geq \nu x. H(x, \mu y. K(y, \nu x. H(x, y))) \)

**Proof.**

(1) \( F(\nu(F))^{(3.1.1)} = \nu(F) \), so by fixpoint induction, \( \mu(F) \leq \nu(F) \).

(2) (a) \( A \leq A \), hence by fixpoint induction, \( \mu x. x \leq A \). Then (b) follows by duality.

(3) Define \( F(x) := \nu y. H(x, y) \) and \( G(y) := \mu x. H(x, y) \). Note that both \( F \) and \( G \) are monotonic, using Lemma 3.1.5. Then:

\[
\mu(F) \overset{(3.1.1)}{=} F(\mu(F)) \overset{(3.1.1)}{=} H(\mu(F), F(\mu(F))) \overset{(3.1.1)}{=} H(\mu(F), \mu(F))
\]

⇒ (by fixpoint induction)

\[
G(\mu(F)) = \mu x. H(x, \mu(F)) \leq \mu(F)
\]

⇒ (monotonicity \( F \))

\[
F(G(\mu(F))) \leq F(\mu(F)) \overset{(3.1.1)}{=} \mu(F)
\]

⇒ (monotonicity \( H \))

\[
F(G(\mu(F))) \overset{(3.1.1)}{=} H(G(\mu(F)), F(G(\mu(F)))) \leq H(G(\mu(F)), \mu(F)) \overset{(3.1.1)}{=} G(\mu(F))
\]

⇒ (by fixpoint induction)

\[
\mu(F) \leq G(\mu(F))
\]

⇒ (by fixpoint induction for \( \nu \))

\[
\mu(F) \leq \nu(G)
\]

(4) (a) Define \( F(y) := \mu x. H(x, y) \) and \( G(x) := \nu y. K(y, x) \). Note that both \( F \) and \( G \) are monotonic, using Lemma 3.1.5. Then:

\[
\mu x. H(x, \nu y. K(y, x)) \overset{(3.1.6)}{=} \mu x. \mu z. H(z, \nu y. K(y, x))
\]

= \( \mu x. F(G(x)) \overset{(3.1.3)}{=} F(\mu x. G(F(x))) \)

\leq (using 1, and monotonicity of \( F \))

\[
F(\nu x. G(F(x))) \overset{(3.1.6)}{=} F(\nu x. \nu y. K(y, F(x))) \overset{(3.1.6)}{=} F(\nu y. K(y, F(y))) \overset{(3.1.6)}{=} \mu x. H(x, \nu y. K(y, \mu x. H(x, y)))
\]

Then (b) follows by “duality” (reversing \( \mu/\nu \) and \( \leq/\geq \)). More precisely, (b) is (a) in the reversed complete lattice \( (U, \geq, lub) \).

Note that (1) and (3) are their own dual.

4. Fixpoint Equation Systems

In this section we first formally define Fixpoint Equation Systems (FES). We show by examples how they generalize Boolean and Predicate Equation Systems. Subsection 4.2 introduces the semantics of a FES by defining its solutions. Finally, Subsection 4.3 defines the variable dependency graph in a FES.
4.1. Definition of Fixpoint Equation Systems. Fix a complete lattice \((U, \leq, \text{glb})\), and a set of variables \(\mathcal{X}\). Throughout the paper, we assume that equality on variables is decidable. We define the set of valuations \(\mathcal{Val} := \mathcal{X} \rightarrow U\). For \(X \in \mathcal{X}\), \(\eta \in \mathcal{Val}\), \(P \subseteq U\), we denote by \(\eta[X := P]\) the valuation that returns \(P\) on \(X\) and \(\eta(Y)\) on \(Y \neq X\). As any function, valuations can be ordered pointwise, i.e. \(\eta_1 \leq \eta_2\) iff \(\forall X \in \mathcal{X}. \eta_1(X) \leq \eta_2(X)\). Note that valuation update is monotonic, that is, if \(P \leq Q\), then \(\eta[X := P] \leq \eta[X := Q]\). To indicate that two valuations agree on a set of variables \(V \subseteq \mathcal{X}\), we write \(\eta_1 \geq V \eta_2\), formally defined as \(\forall X \in V. \eta_1(X) = \eta_2(X)\). The complement of \(V\) in \(\mathcal{X}\) is denoted \(\overline{V}\).

A set of mutually recursive equations is a member of \(\mathcal{Eqs} := \mathcal{Val} \rightarrow \mathcal{Val}\). \(\mathcal{E}\) is monotonic iff it is a monotonic function on \(\mathcal{Val}\). Note that this semantic view on equations escapes the need to introduce (and be limited) to a particular syntax.

Example 4.1. Take \(\mathcal{X} = \{X, Y, Z\}\) and \(U = \mathbb{B}\), the Boolean lattice \(\bot < \top\). We write \((a, b, c) \in \mathbb{B}^3\) as a shorthand for the valuation \(\{X=a, Y=b, Z=c\}\). The system of equations \(\{X = Y \land Z, Y = X \lor Z, Z = \lnot X\}\) is represented in our theory as the function

\[\mathcal{B} := \lambda(X, Y, Z) \in \mathbb{B}^3.(Y \land Z, X \lor Z, \lnot X)\,\] It is not monotonic, because as valuations, \((\bot, \bot, \bot) \leq (\top, \top, \top)\), but

\[\mathcal{B}(\bot, \bot, \bot) = (\bot, \bot, \bot) \not\leq (\top, \top, \top) = \mathcal{B}(\top, \top, \top)\,\]

Note that \(\mathcal{Eqs}\) is isomorphic with \(\mathcal{X} \rightarrow \mathcal{Val} \rightarrow U\). This motivates the following slight abuse of notation: Given \(\mathcal{E} \in \mathcal{Eqs}\), we will often write \(\mathcal{E}(\eta)(X)\) for \(\mathcal{E}(\eta)(X)\). This expression denotes the definition of \(X\) in \(\mathcal{E}\), possibly depending on other variables as represented by the valuation \(\eta\). Similar to valuations, agreement on variables from \(\mathcal{S}\) to \(\mathcal{X}\) consists of finite lists of signed variables:

\[\mathcal{Spec} := (\{\mu, \nu\} \times \mathcal{X}^*)\,\]

Note that a specification selects a subset of variables to be considered, assigns a fixpoint sign to these variables, and assigns an order to these variables. We use \(\sigma X\) as a notation for \(\{\sigma, X\}\), write \(\varepsilon\) for the empty list, and use \(;\) for list concatenation. We will identify a singleton list with its element. For instance, \([\mu X, \nu Y]; \mu Z\) denotes the specification \([\mu X, \nu Y, \nu Z]\). We define \(\text{dom}(\mathcal{S}) \subseteq \mathcal{X}\) as the set of variables that occur in some pair in \(\mathcal{S}\). Decidability of \(X \in \text{dom}(\mathcal{S})\) follows from finiteness of \(\mathcal{S}\) and decidability of equality on variables. We define \(\text{disjoint}(\mathcal{S}_1, \mathcal{S}_2)\) iff \(\text{dom}(\mathcal{S}_1) \cap \text{dom}(\mathcal{S}_2) = \emptyset\).

We often require that valuations or equation systems agree on the variables in a specification. Accordingly, we overload \(\leq\) so that \(\eta_1 \leq \eta_2\) (resp. \(\mathcal{E}_1 \leq \mathcal{E}_2\)) is defined as \(\forall X \in \text{dom}(\mathcal{S}). \eta_1(X) \leq \eta_2(X)\) (resp. \(\mathcal{E}_1 \leq \mathcal{E}_2\)). This also applies when a complement is involved: \(\eta_1 \geq \eta_2\) is defined as \(\forall X \in \text{dom}(\mathcal{S}). \eta_1(X) \geq \eta_2(X)\).

Finally, a fixpoint equation system (FES) \(\mathcal{F}\) on \((U, \mathcal{X})\) is simply a pair in \(\mathcal{F}\) := \(\mathcal{Eqs} \times \mathcal{Spec}\).

Example 4.2. The Boolean Equation System [Mad97] traditionally written as

\[
\begin{align*}
\mu X &= Y \land Z \\
\nu Y &= X \lor Z \\
\nu Z &= \lnot X
\end{align*}
\]

is represented in our theory as the pair \((\mathcal{B}, [\mu X, \nu Y, \nu Z])\), where \(\mathcal{B}\) is from Example 4.1.

Example 4.3. A PBES (parameterized BES [GW05a], or predicate equation system [ZC05]) is a FES over the complete lattice \(U := (\mathcal{P}(D), \subseteq)\) for some data set \(D\). Consider the
following PBES in traditional notation.

\[
\mu X (m^N, b^B) = (b \rightarrow m > 0 \land Y(m)) \land (\neg b \rightarrow m < 5 \land Y(m))
\]

\[
\nu Y (m^N) = X(m - 1, m > 4) \lor Y(m + 1)
\]

To represent this in our theory, \(Y\) should be extended by a dummy boolean argument, so that both variables in \(X := \{X, Y\}\) are predicates over the same \(D := \mathbb{N} \times \mathbb{B}\). So we actually view this system as:

\[
\mu X = \lambda m, b. (b \rightarrow m > 0 \land Y(m, \bot)) \land (\neg b \rightarrow m < 5 \land Y(m, \bot))
\]

\[
\nu Y = \lambda m, b. X(m - 1, m > 4) \lor Y(m + 1, \bot)
\]

4.2. **Semantics of FES and Basic Results.** Next, the semantics of a FES, \([\mathcal{E}, \mathcal{S}] : \mathcal{V} \rightarrow \mathcal{V}\), is defined recursively on \(\mathcal{S}\):

\[
\begin{align*}
[\mathcal{E}, \varepsilon](\eta) & := \eta \\
[\mathcal{E}, \sigma X; \mathcal{S}](\eta) & := [\mathcal{E}, \mathcal{S}](\eta[X := \sigma(F)]), \\
& \text{where } F(P) := \mathcal{E}_X([\mathcal{E}, \mathcal{S}](\eta[X := P]))
\end{align*}
\]

We now state the first results on the semantics of FES. The lemma below states monotonicity properties for the semantics. They ensure that fixpoints are well-behaved.

**Lemma 4.4.** Let \(\mathcal{E}, \mathcal{E}_1, \mathcal{E}_2 \in \mathcal{E}qs\) and \(\mathcal{S} \in \mathcal{Spec}\), then

1. If \(\mathcal{E}\) is monotonic, then \([\mathcal{E}, \mathcal{S}]\) is monotonic.
2. If \(\mathcal{E}_1\) is monotonic and \(\mathcal{E}_1 \leq \mathcal{E}_2\), then \([\mathcal{E}_1, \mathcal{S}] \leq [\mathcal{E}_2, \mathcal{S}]\).

**Proof.** (1) We prove \(\forall \eta_1, \eta_2. \eta_1 \leq \eta_2 \Rightarrow [\mathcal{E}, \mathcal{S}](\eta_1) \leq [\mathcal{E}, \mathcal{S}](\eta_2)\) by induction on \(\mathcal{S}\). The base case: assume \(\eta_1 \leq \eta_2\), then \([\mathcal{E}, \varepsilon](\eta_1) = \eta_1 \leq \eta_2 = [\mathcal{E}, \varepsilon](\eta_2)\). Induction step: Let \(\eta_1 \leq \eta_2\) be given, then for any \(P \in U\), we have \(\eta_1[X := P] \leq \eta_2[X := P]\). So

\[
\begin{align*}
& \text{(induction hypothesis)} \\
& [\mathcal{E}, \mathcal{S}](\eta_1[X := P]) \leq [\mathcal{E}, \mathcal{S}](\eta_2[X := P]) \\
\Rightarrow & \text{ (\(\mathcal{E}\) is monotonic)} \\
& \mathcal{E}_X([\mathcal{E}, \mathcal{S}](\eta_1[X := P])) \leq \mathcal{E}_X([\mathcal{E}, \mathcal{S}](\eta_2[X := P])) \\
\Rightarrow & \text{ (fixpoint monotonicity, Lemma 3.1.5)} \\
& \sigma P. \mathcal{E}_X([\mathcal{E}, \mathcal{S}](\eta_1[X := P])) \leq \sigma P. \mathcal{E}_X([\mathcal{E}, \mathcal{S}](\eta_2[X := P])) \\
\Rightarrow & \text{ (define } F_i(P) := \mathcal{E}_X([\mathcal{E}, \mathcal{S}](\eta_i[X := P])), \text{ for } i = 1, 2) \\
& \eta_1[X := \sigma(F_1)] \leq \eta_2[X := \sigma(F_2)] \\
\Rightarrow & \text{ (induction hypothesis)} \\
& [\mathcal{E}, \mathcal{S}](\eta_1[X := \sigma(F_1)]) \leq [\mathcal{E}, \mathcal{S}](\eta_2[X := \sigma(F_2)]) \\
\Leftrightarrow & \text{ (definition)} \\
& [\mathcal{E}, \sigma X; \mathcal{S}](\eta_1) \leq [\mathcal{E}, \sigma X; \mathcal{S}](\eta_2)
\end{align*}
\]

(2) Assume \(\mathcal{E}_1\) is monotonic, and \(\mathcal{E}_1 \leq \mathcal{E}_2\) (pointwise). We prove by induction on \(\mathcal{S}\) that \(\forall \eta. [\mathcal{E}_1, \mathcal{S}](\eta) \leq [\mathcal{E}_2, \mathcal{S}](\eta)\). The base case is simple: for all \(\eta\), we have \([\mathcal{E}_1, \varepsilon](\eta) = \eta = [\mathcal{E}_2, \varepsilon](\eta)\). For the induction step, let \(\eta\) be given, then for any \(P \in U\):

\[
\begin{align*}
& \text{(induction hypothesis)} \\
& [\mathcal{E}_1, \mathcal{S}](\eta[X := P]) \leq [\mathcal{E}_2, \mathcal{S}](\eta[X := P])
\end{align*}
\]
Proof. Let \( \mathcal{E}_1, \mathcal{E}_2 \in \mathcal{E}qs, S \in \mathcal{Spec}, \eta \in \mathcal{Val} \) and \( X \in \mathcal{X} \).

1. If \( X \notin \text{dom}(S) \) then \( [\mathcal{E}, S](\eta)(X) = \eta(X) \).

2. If \( \mathcal{E}_1 \subseteq \mathcal{E}_2 \), then \( [\mathcal{E}_1, S] = [\mathcal{E}_2, S] \).

3. If \( \eta_1 \preceq \eta_2 \), then \( [\mathcal{E}, S](\eta_1) = [\mathcal{E}, S](\eta_2) \).

Proof. (1) Induction on \( S \). The base case holds by definition. For the induction step, assume \( X \notin \text{dom}(\sigma Y; S) \). Then

\[
[\mathcal{E}, \sigma Y; S](\eta)(X) = \text{(by definition; for some particular } F) \]
\[
[\mathcal{E}, S](\eta[Y := \sigma(F)])(X) = \text{(induction hypothesis, note: } X \notin \text{dom}(S)) \]
\[
\eta[Y := \sigma(F)](X) = \text{(} X \notin \text{dom}(\sigma Y; S), \text{ so } X \neq Y) \]
\[
\eta(X)
\]

(2) Induction on \( S \). The base case holds by definition. For the induction step, let \( \eta \) be given, and assume \( \mathcal{E}_1 \preceq \mathcal{E}_2 \). Note that this implies \( \mathcal{E}_1 \subseteq \mathcal{E}_2 \), so we can use the induction hypothesis. For any \( P \in U \) we have:

(induction hypothesis)
\[
[\mathcal{E}_1, S](\eta[X := P]) = [\mathcal{E}_2, S](\eta[X := P])
\]
\[
\Rightarrow (X \in \text{dom}(\sigma X; S), \text{ so } \mathcal{E}_{1,X} = \mathcal{E}_{2,X} \text{ by assumption}) \]
\[
\mathcal{E}_{1,X}([\mathcal{E}_1, S](\eta[X := P])) = \mathcal{E}_{2,X}([\mathcal{E}_2, S](\eta[X := P]))
\]
\[
\Rightarrow (\text{define } F_i(P) := \mathcal{E}_{i,X}([\mathcal{E}_i, S](\eta[X := P])) \text{ for } i = 1, 2)
\]
\[ \eta[X := \sigma(F_1)] = \eta[X := \sigma(F_2)] \]
\Rightarrow (induction hypothesis)
\[ [\mathcal{E}_1, \mathcal{S}](\eta[X := \sigma(F_1)]) = [\mathcal{E}_2, \mathcal{S}](\eta[X := \sigma(F_2)]) \]
\Leftrightarrow (definition)
\[ [\mathcal{E}_1, \sigma X; \mathcal{S}](\eta) = [\mathcal{E}_2, \sigma X; \mathcal{S}](\eta) \]

(3) Induction on \( \mathcal{S} \). The base case is trivial, because both the assumption and the conclusion reduce to \( \eta_1 = \eta_2 \). The induction step is proved as follows (for arbitrary \( P \in U \)):

(assumption)
\[ \eta_1 \overset{\mathcal{S}}{\rightarrow} \eta_2 \]
\Rightarrow (extensionality)
\[ \eta_1[X := P] \overset{\mathcal{S}}{=} \eta_2[X := P] \]
\Rightarrow (induction hypothesis)
\[ [\mathcal{E}, \mathcal{S}](\eta_1[X := P]) = [\mathcal{E}, \mathcal{S}](\eta_2[X := P]) \]
\Rightarrow (define \( F_i(P) := \mathcal{E}_X([\mathcal{E}, \mathcal{S}](\eta_i[X := P])), \) for \( i = 1, 2 \))
\[ \sigma(F_1) = \sigma(F_2) \]
\Rightarrow (extensionality)
\[ \eta_1[X := \sigma(F_1)] \overset{\mathcal{S}}{=} \eta_2[X := \sigma(F_2)] \]
\Rightarrow (induction hypothesis)
\[ [\mathcal{E}, \mathcal{S}](\eta_1[X := \sigma(F_1)]) = [\mathcal{E}, \mathcal{S}](\eta_2[X := \sigma(F_2)]) \]
\Rightarrow (definition)
\[ [\mathcal{E}, \sigma X; \mathcal{S}](\eta_1) = [\mathcal{E}, \sigma X; \mathcal{S}](\eta_2) \]

Next, we can prove that the semantics as defined above indeed solves the equations for those variables occurring in the specification:

**Lemma 4.6.** Let \( \mathcal{E} \in \mathcal{E}_{qs} \) be monotonic, \( \mathcal{S} \in \mathcal{Spec}, \eta \in \mathcal{Val} \) and \( X \in \mathcal{X} \). If \( X \in \text{dom}(\mathcal{S}) \), then \( \mathcal{E}_X([\mathcal{E}, \mathcal{S}](\eta)) = [\mathcal{E}, \mathcal{S}](\eta)(X) \).

**Proof.** Let \( \mathcal{E} \) be monotonic. By induction on \( \mathcal{S} \), we will prove that for all \( X \in \text{dom}(\mathcal{S}) \) and for all \( \eta \), it holds that \( \mathcal{E}_X([\mathcal{E}, \mathcal{S}](\eta)) = [\mathcal{E}, \mathcal{S}](\eta)(X) \). The base case trivially holds, because \( X \notin \text{dom}(\varepsilon) \). For the induction step, assume \( X \in \text{dom}(\sigma Y; \mathcal{S}) \). We distinguish cases.

If \( X \in \text{dom}(\mathcal{S}) \):

\[ [\mathcal{E}, \sigma Y; \mathcal{S}](\eta)(X) \]
\[ = (\text{define } F(P) := \mathcal{E}_Y([\mathcal{E}, \mathcal{S}](\eta[Y := P])) ) \]
\[ [\mathcal{E}, \mathcal{S}](\eta[Y := \sigma(F)])(X) \]
\[ = (\text{induction hypothesis}; \ X \in \text{dom}(\mathcal{S})) \]
\[ \mathcal{E}_X([\mathcal{E}, \mathcal{S}](\eta[Y := \sigma(F)])) \]
\[ = (\text{definition}) \]
\[ \mathcal{E}_X([\mathcal{E}, \sigma Y; \mathcal{S}](\eta)) \]
Otherwise, if \( X \not\in \text{dom}(S) \) then \( X = Y \). We compute:

\[
\begin{align*}
& \llbracket E, \sigma X; S \rrbracket (\eta)(X) \\
= & \quad (\text{define } F(P) := E_X(\llbracket E, S \rrbracket (\eta[X := P]))) \\
= & \quad (\text{Lemma 4.5.1; } X \not\in \text{dom}(S)) \\
= & \quad (\text{computation rule, Lemma 3.1.1; } F \text{ is monotonic because } E_X \text{ is monotonic by assumption, and } \llbracket E, S \rrbracket \text{ is by Lemma 4.4.1}) \\
= & \quad F(\sigma(F)) \\
= & \quad E_X(\llbracket E, \sigma X; S \rrbracket (\eta))
\end{align*}
\]

We have the following left-congruence result:

**Lemma 4.7.** For \( E_1, E_2 \in Eqs, S, S_1, S_2 \in Spec, \) if \( E_1 \preceq S_1, S_2 \), and \( \llbracket E_1, S_1 \rrbracket = \llbracket E_2, S_2 \rrbracket \), then \( \llbracket E_1, S; S_1 \rrbracket = \llbracket E_2, S; S_2 \rrbracket \).

**Proof.** Induction on \( S \). The base case is trivial. For the induction step, assume \( \llbracket E_1, S_1 \rrbracket = \llbracket E_2, S_2 \rrbracket \) and \( E_1 \preceq S; S_2 \). Then also \( E_1 \preceq E_2 \). So, for any \( P \in U \):

\[
\begin{align*}
& \quad (\text{induction hypothesis}) \\
& \quad \llbracket E_1, S; S_1 \rrbracket (\eta[X := P]) = \llbracket E_2, S; S_2 \rrbracket (\eta[X := P]) \\
\Rightarrow & \quad (\text{define } F_i(P) := E_{i,X}(\llbracket E_i, S; S_i \rrbracket (\eta[X := P])); \text{ note } E_{1,X} = E_{2,X}) \\
& \quad \eta[X := \sigma(F_1)] = \eta[X := \sigma(F_2)] \\
\Rightarrow & \quad (\text{induction hypothesis}) \\
& \quad \llbracket E_1, S; S_1 \rrbracket (\eta[X := \sigma(F_1)]) = \llbracket E_2, S; S_2 \rrbracket (\eta[X := \sigma(F_2)]) \\
\Rightarrow & \quad (\text{definition}) \\
& \quad \llbracket E_1, \sigma X; S; S_1 \rrbracket (\eta) = \llbracket E_2, \sigma X; S; S_2 \rrbracket (\eta)
\end{align*}
\]

Remarkably, right-congruence doesn’t hold in general. Corollary 6.5 will state a sufficient condition for right-congruence.

**Remark 4.8.** We have not required that all variables in \( S \) are unique. This is not needed in our formalization, because from Lemma 4.5.3 it follows that If \( X \in \text{dom}(S) \), then \( \llbracket E, \sigma X; S \rrbracket = \llbracket E, S \rrbracket \). However, note that there is a hidden assumption: Even if \( X \) occurs multiple times in \( S \), possibly with different signs, there can only be one defining equation for it, because \( E \) is a function. So when transferring the results to traditional notation, one should add the (quite natural) requirement that all equations have unique variable names.
4.3. The Dependency Graph between Variables. Since we introduced a semantic notion of equations, avoiding syntactic expressions, we also need a syntactic notion of dependence between variables. Given $V_1, V_2 \subseteq X$, we define that $V_1$ is independent of $V_2$ with respect to $E$, notation $\text{indep}(E, V_1, V_2)$, as follows:

$$\forall \eta_1, \eta_2. (\eta_1 \xrightarrow{E} \eta_2) \Rightarrow (E(\eta_1) \xrightarrow{V_1} E(\eta_2)).$$

That is: the solution of variables $X \in V_1$ is the same for all those $\eta$ that differ at most on the values assigned to $Y \in V_2$. This is slightly more liberal than the usual syntactic requirement that $Y$ doesn't occur syntactically in $E_X$. We overload the definition of indep for individual variables and specifications (and any combination of those), e.g., $\text{indep}(E, S_1, S_2) = \text{indep}(E, \text{dom}(S_1), \text{dom}(S_2))$ and $\text{indep}(E, X, Y) = \text{indep}(E, \{X\}, \{Y\})$.

This notion gives rise to the variable dependency graph of a FES $(E, S)$. The variables in $\text{dom}(S)$ form the nodes of this graph; the edges $X \xrightarrow{E;S} Y$ are defined as $\neg\text{indep}(E, X, Y)$. We define $X$ depends (indirectly) on $Y$ (written $X \xrightarrow{E;S} Y$) as the reflexive, transitive closure of $\xrightarrow{E;S}$. In other words, there exists a path in the dependency graph from $X$ to $Y$. We also use the notation $X \xrightarrow{E;S}^+ Y$ to denote the transitive closure of $\xrightarrow{E;S}$, i.e. there is a non-empty path from $X$ to $Y$ in the dependency graph. Note that since $\text{dom}(S)$ is finite, both $\xrightarrow{E;S}$ and $\xrightarrow{E;S}^+$ are decidable.

5. Substituting in FES Equations

In this section, we define two substitution operations on the equations of a FES, and study under which conditions these operations preserve solutions. The first operation allows substituting variables by their definition. We show that this substitution preserves solutions in some new cases (cf. Section 2). The second operation replaces a variable in an equation by its solution.

5.1. Unfolding Definitions. We define $\text{unfold}(E, X, Y)$, where each occurrence of $Y$ in the definition of $X$ is replaced by the definition of $Y$, as follows:

$$\text{unfold}(E, X, Y)(\eta) := E(\eta)[X := E_X(\eta)[Y := E_Y(\eta)]]$$

We will use the following observation several times. It basically states that unfolding $Y$ in $X$ doesn't affect other equations than that for $X$.

Lemma 5.1. If $X \notin \text{dom}(S)$, then

1. $\text{unfold}(E, X, Y) \xrightarrow{S} E$
2. $\llbracket\text{unfold}(E, X, Y), S\rrbracket = \llbracket E, S \rrbracket$

Proof. (1) holds by definition of unfold, as it only modifies the value of $E$ on variable $X$. Then (2) holds by Lemma 4.5.2.

It is known (cf. Example 2.1) that in general one should not unfold $Y$ in $X$, if $Y$ precedes $X$ in the specification. As a new result we show that we can substitute $X$ in its own definition:

Lemma 5.2. Let $E \in \text{Eqs}$ be monotonic. Let $S = \sigma_X S_1$, and $X \notin \text{dom}(S_1)$. Then $\llbracket\text{unfold}(E, X, X), S\rrbracket = \llbracket E, S \rrbracket$. 

Theorem 5.3. Proof. For arbitrary $\eta$, define

$$F(P) := \mathcal{E}_X\left(\left[\mathcal{E}, \mathcal{S}_1\right]\eta[X := P]\right)$$

$$G(P) := \text{unfold}(\mathcal{E}, X, X)\left(\left[\mathcal{E}, \mathcal{S}_1\right]\eta[X := P]\right)$$

$$H(P, Q) := \mathcal{E}_X\left(\left[\mathcal{E}, \mathcal{S}_1\right]\eta[X := P]\right)[X := Q]\right)$$

By Lemma 4.5.1 and $X \not\in \text{dom}(S_1)$, we obtain: $\left[\mathcal{E}, \mathcal{S}_1\right](\eta[X := P])(X) = \eta[X := P](X) = P$, so

$$\left[\mathcal{E}, \mathcal{S}_1\right](\eta[X := P])[X := Y] = \left[\mathcal{E}, \mathcal{S}_1\right](\eta[X := P]) \quad (*)$$

Next, we prove that $\sigma(P) = \sigma(G)$:

$$\sigma(P) G(P)$$

= (by definition of unfold)

$$\sigma(P) \mathcal{E}_X\left(\left[\mathcal{E}, \mathcal{S}_1\right]\eta[X := P]\right)\left[\mathcal{E}_X\left(\left[\mathcal{E}, \mathcal{S}_1\right]\eta[X := P]\right)\right]$$

= (by * above)

$$\sigma(P) \mathcal{E}_X\left(\left[\mathcal{E}, \mathcal{S}_1\right]\eta[X := P]\right)\left[\mathcal{E}_X\left(\left[\mathcal{E}, \mathcal{S}_1\right]\eta[X := P]\right)[X := P]\right]$$

= (by rule, Lemma 3.1.7)

$$\sigma(P) H(P, P)$$

= (by * above)

$$\sigma(P) F(P)$$

We can now finish the proof:

$$\left[\text{unfold}(\mathcal{E}, X, X), \sigma X; S_1\right](\eta)$$

= (by definition and Lemma 5.1.2)

$$\left[\mathcal{E}, \mathcal{S}_1\right]\eta[X := \sigma(G)]$$

= (by the computation before)

$$\left[\mathcal{E}, \mathcal{S}_1\right]\eta[X := \sigma(F)]$$

= (by definition)

$$\left[\mathcal{E}, \sigma X; S_1\right](\eta) \quad \square$$

The full theorem allows to unfold $Y$ in the equations for those $X$ that precede that of $Y$, and in the equation of $Y$ itself. So in particular, the case $X = Y$ is allowed.

**Theorem 5.3.** Let $\mathcal{E} \in \text{Eqs}$ be monotonic. Let $S = S_1; \sigma Y; S_2$ and $X \not\in \text{dom}(S_2)$. Then $\left[\text{unfold}(\mathcal{E}, X, Y), S\right] = \left[\mathcal{E}, S\right]$.

**Proof.** The proof is by induction on $S_1$. The base case is $S_1 = \varepsilon$. If $X = Y$, we have to prove $\left[\text{unfold}(\mathcal{E}, X, Y), \sigma X; S_2\right] = \left[\mathcal{E}, \sigma X; S_2\right]$, which is just Lemma 5.2. Otherwise, if $X \neq Y$, then $X \not\in \text{dom}(\sigma Y; S_2)$, so by Lemma 5.1 $\left[\text{unfold}(\mathcal{E}, X, Y), \sigma Y; S_2\right] = \left[\mathcal{E}, \sigma Y; S_2\right]$.

Next, for $S = \rho Z; S_1; \sigma Y; S_2$, define $S_3 := \rho Z; S_1; \sigma Y; S_2$, and assume the induction hypothesis, $\left[\text{unfold}(\mathcal{E}, X, Y), S_3\right] = \left[\mathcal{E}, S_3\right]$. 


If \( Z \neq X \), then \( E \{ Z \} \) unfold\((E, X, Y)\). From the induction hypothesis, it follows by congruence (Lemma 4.7) that \( \text{fold}(E, X, Y), \rho Z; S_3] = [E, \rho Z; S_3]. \)

If \( Z = X \), we compute for arbitrary \( \eta \in \text{Val} \):

\[
\begin{align*}
\text{fold}(E, X, Y), \sigma X; S_3](\eta) &= \text{(by definition of the semantics)} \\
\text{fold}(E, X, Y), S_3](\eta[X := \sigma(F)]) \text{, where} \\
F(P) &= \text{fold}(E, X, Y)(\text{fold}(E, X, Y), S_3](\eta[X := P])) \\
&= \text{(by induction hypothesis)} \\
\text{fold}(E, S_3](\eta[X := \sigma(F)]) \text{, where} \\
F(P) &= \text{fold}(E, X, Y)(\text{fold}(E, S_3](\eta[X := P])) \\
&= \text{(definition unfold)} \\
&= \text{(Lemma 4.6; } Y \in \text{dom}(S_3)) \\
&= \text{(congruence and extensionality: } \zeta[Y := \zeta(Y)] = \zeta) \\
&= \text{fold}(E, S_3](\eta[X := P])) \\
&= \text{fold}(E, \sigma X; S_3](\eta)
\end{align*}
\]

5.2. Substituting a Partial Solution. The following theorem is motivated in [Mad97] as follows. Assume we know by some means the solution \( a \) for a variable \( X \) in \( (E, S) \). Then we can replace the definition of \( X \) by simply putting \( X = a \). We simplify the proof in [Mad97], which is based on an infinite series of FESs. Instead, we just use induction on \( S \) and some properties of complete lattices.

Theorem 5.4. Let \( E \in \text{Eqs} \) be monotonic and let \( a := [E, S](\eta)(X) \). Then

\[
\[E, S](\eta) = \{E[X \mapsto a], S\}(\eta)
\]

Proof. We prove the theorem by induction on \( S \). The base case is trivial, for \( [E, \varepsilon](\eta) = \eta = [E[X \mapsto a], \varepsilon](\eta) \). For the induction step \( (\sigma Y; S) \), we will need several definitions:

\[
\begin{align*}
a(\eta') &= \{E, S\}(\eta')(X) \\
b(\eta) &= \{E, \sigma Y; S\}(\eta)(X) \\
F(P) &= \text{EY}([E, S](\eta[Y := P])) \\
G(P) &= \text{EY}([E[X \mapsto b(\eta)], S]) \\
H(P, Q) &= \text{EY}([E[X \mapsto a(\eta[Y := P])], S](\eta[Y := Q]))
\end{align*}
\]

Then from the induction hypothesis: \( \forall \eta' \). \( [E, S](\eta') = [E[X \mapsto a(\eta')], S](\eta') \), we must prove: \( \forall \eta \). \( [E, \sigma Y; S](\eta) = [E[X \mapsto b(\eta)], \sigma Y; S](\eta) \).

We distinguish three cases:
If $X = Y$ and $X \not\in \text{dom}(S)$, we compute:

$$
[\mathcal{E}[X \mapsto b(\eta)], \sigma X; S](\eta)
= [\mathcal{E}[X \mapsto b(\eta)], S](\eta[X := \sigma P b(\eta)])
= (\text{by the constant rule, Lemma 3.1.2})
[\mathcal{E}[X \mapsto b(\eta)], S](\eta[X := b(\eta)])
= (\text{by Lemma 4.5.2 and } X \not\in \text{dom}(S))
[\mathcal{E}, S](\eta[X := b(\eta)])
= (\text{by Lemma 4.5.1 and } X \not\in \text{dom}(S))
[\mathcal{E}, S](\eta[X := \sigma(F)])
= [\mathcal{E}, \sigma X; S](\eta)
$$

If $X = Y$ and $X \in \text{dom}(S)$, then note that using Lemma 4.5.3, for any $\mathcal{E}'$ and appropriate $F'$, we have:

$$
[\mathcal{E}', \sigma X; S](\eta) = [\mathcal{E}', S](\eta[X := \sigma F']) = [\mathcal{E}', S](\eta)
$$

So in particular, $b(\eta) = a(\eta)$, and we can apply the induction hypothesis: Hence

$$
[\mathcal{E}[X \mapsto b(\eta)], \sigma X; S](\eta)
= (\text{by the equality above})
[\mathcal{E}[X \mapsto b(\eta)], S](\eta)
= [\mathcal{E}[X \mapsto a(\eta)], S](\eta)
= (\text{induction hypothesis})
[\mathcal{E}, S](\eta)
= (\text{by the equality above})
[\mathcal{E}, \sigma X; S](\eta)
$$

Finally, if $X \neq Y$, then we can compute:

$$
[\mathcal{E}, \sigma Y; S](\eta)
= [\mathcal{E}, S](\eta[Y := \sigma(F)])
= (\text{induction hypothesis})
[\mathcal{E}[X \mapsto a(\eta)[Y := \sigma(F)]], S](\eta[Y := \sigma(F)])
= (\text{unfold definition of } \[ ] \text{ in } b(\eta).)
[\mathcal{E}[X \mapsto b(\eta)], S](\eta[Y := \sigma(F)])
= (\text{will be proved below})
[\mathcal{E}[X \mapsto b(\eta)], \sigma Y; S](\eta)
$$
We must still prove that $\sigma(F) = \sigma(G)$.

\[
\begin{aligned}
\sigma P. F(P) \\
= \text{(induction hypothesis)} \\
\sigma P. E_{\forall}(\eta[X \mapsto a(\eta[Y := P])], S)[\eta[Y := P]]) \\
= \sigma P. H(P, P) \\
= \text{Lemma 3.1.8 (solve rule) on } \lambda P. Q. H(Q, P) \\
\sigma P. H(\sigma P. H(P, P), P) \\
= (F(P) = H(P, P) \text{ as above}) \\
\sigma P. H(\sigma P. F(P), P) \\
= \text{(unfold definition of } [ ] \text{ in } b \text{ in } G) \\
\sigma P. G(P)
\end{aligned}
\]

\[\square\]

6. Swapping Variables in FES specifications

We now study swapping the order of variables in a specification. In general, this operation doesn’t exactly preserve solutions. We first show that adjacent variables with the same sign may be swapped without changing the semantics (Section 6.1). Subsequently, we will prove that we can swap the order between blocks of equations, under certain independence criteria (Section 6.2). The main theorem of that section is new (cf. Section 2). Finally, we show that swapping a $\mu/\nu$ sequence by the corresponding $\nu/\mu$ in general leads to a greater or equal solution (Section 6.3).

6.1. Swapping Equations with the same Sign.

**Theorem 6.1.** Assume $E \in Eqs$ is monotonic. Then

\[
[E, S_1; \sigma X; \sigma Y; S_2] = [E, S_1; \sigma Y; \sigma X; S_2]
\]

**Proof.** We first compute for arbitrary $\eta \in Val$:

\[
\begin{aligned}
[E, \sigma X; \sigma Y; S_2](\eta) \\
= [E, \sigma Y; S_2](\eta[X := \sigma(F_2)]) \\
= [E, \sigma Y; S_2](\eta[X := \sigma(F_2)], Y := \sigma(F_3)) \\
= [E, S_2](\eta[X := \sigma(F_2), Y := \sigma(F_3)]) \\
= [E, S_2](\eta[X := \sigma(F_2), Y := \sigma(F_3)], Q) \\
= A(\sigma(F_2), \sigma(F_3)), where
\end{aligned}
\]
\[ A(P, Q) = \llbracket \mathcal{E}, S_2 \rrbracket(\eta[X := P, Y := Q]) \]
\[ F_1(P)(Q) = E_Y(A(P, Q)) \]
\[ F_2(P) = E_X(A(P, \sigma(F_1(P)))) \]
\[ F_3(Q) = E_Y(A(\sigma(F_2), Q)) \]

Symmetrically, we get:
\[ \llbracket \mathcal{E}, \sigma Y; \sigma X; S_2 \rrbracket(\eta) \]
\[ = B(\sigma(G_2), \sigma(G_3)) \]
\[ = B(Q, P) = \llbracket \mathcal{E}, S_2 \rrbracket(\eta[Y := Q, X := P]) \]
\[ G_1(Q)(P) = E_X(B(Q, P)) \]
\[ G_2(Q) = E_Y(B(Q, \sigma(G_1(Q)))) \]
\[ G_3(P) = E_X(B(\sigma(G_2), P)) \]

Note that the theorem is trivial when \( X = Y \). So we may assume \( X \neq Y \). Hence \( A(P, Q) = B(Q, P) \), and we have:

\[ \sigma(F_2) \]
\[ = \sigma P. E_X(A(P, \sigma(F_1(P)))) \]
\[ = \sigma P. E_X(A(P, \sigma Q. E_Y(A(P, Q)))) \]
\[ = \text{ (Bekič rule, Lemma 3.1.9, with } H(p, q) := E_X(A(p, q)) \text{ and } K(p, q) := E_Y(A(q, p)) \text{, which are monotonic, because } \mathcal{E} \text{ is by assumption, and } \llbracket \mathcal{E}, S_2 \rrbracket \text{ by Lemma 4.4.1) } \]
\[ \sigma P. E_X(A(P, \sigma Q. E_Y(A(\sigma P. E_X(A(P, Q)), Q)))) \]
\[ = \text{ (while } A(P, Q) = B(Q, P) \text{ ) } \]
\[ \sigma P. E_X(B(\sigma Q. E_Y(B(Q, \sigma P. E_X(B(Q, P))))), P) \]
\[ = \sigma P. E_X(B(\sigma Q. E_Y(B(Q, \sigma(G_1(Q)))), P)) \]
\[ = \sigma P. E_X(B(\sigma(G_2), P)) \]
\[ = \sigma(G_3) \]

We can now finish the proof:

\[ \text{ (computation above, and full symmetry) } \]
\[ \sigma(F_2) = \sigma(G_3) \text{ and } \sigma(F_3) = \sigma(G_2) \]
\[ \Rightarrow \text{ (because } A(P, Q) = B(Q, P) \text{ ) } \]
\[ A(\sigma(F_2), \sigma(F_3)) = B(\sigma(G_2), \sigma(G_3)) \]
\[ \Rightarrow \llbracket \mathcal{E}, \sigma X; \sigma Y; S_2 \rrbracket(\eta) = \llbracket \mathcal{E}, \sigma Y; \sigma X; S_2 \rrbracket(\eta) \]
\[ \Rightarrow \text{ (Lemma 4.7) } \]
\[ \llbracket \mathcal{E}, S_1; \sigma X; \sigma Y; S_2 \rrbracket(\eta) = \llbracket \mathcal{E}, S_1; \sigma Y; \sigma X; S_2 \rrbracket(\eta) \]
6.2. Migrating Independent Blocks of Equations. Our aim here is to investigate swapping blocks of equations that are independent. We first need two technical lemmas. The first lemma enables to commute updates to valuations with computing solutions:

**Lemma 6.2.** If \( X \notin \text{dom}(S) \) and \( \text{indep}(E, S, X) \), then

\[
[E, S](\eta[X := P]) = ([E, S](\eta))[X := P]
\]

**Proof.** Induction on \( S \). The base case is trivial:

\[
[E, \varepsilon](\eta[X := P]) = \eta[X := P] = [E, \varepsilon](\eta)[X := P]
\]

Case \( S = \sigma Y; S_1 \). Assume \( X \notin \text{dom}(S) \) and \( \text{indep}(E, S, X) \), then \( X \neq Y \), and also \( X \notin \text{dom}(S_1) \) and \( \text{indep}(E, S_1, X) \), so the induction hypothesis can be applied. Then

\[
[E, \sigma Y; S_1](\eta[X := P]) = [E, S_1](\eta[X := P, Y := \sigma(F)]) \quad \text{where}
\]

\[
F(Q) := \varepsilon_Y([E, S_1](\eta[X := P, Y := Q]))
\]

\[
= (\text{induction hypothesis, and } X \neq Y)
\]

\[
\varepsilon_Y([E, S_1](\eta[Y := Q])[X := P])
\]

\[
= (Y \in \text{dom}(S) \text{ is independent of } X)
\]

\[
\varepsilon_Y([E, S_1](\eta[Y := Q])) = G(Q)
\]

\[
= [E, S_1](\eta[X := P, Y := \sigma(G)])
\]

\[
= (\text{induction hypothesis, and } X \neq Y)
\]

\[
[E, S_1](\eta[Y := \sigma(G)])[X := P]
\]

\[
= [E, \sigma Y; S_1](\eta)[X := P]
\]

The next lemma states that independent specifications can be solved independently.

**Lemma 6.3.** Let \( E \in \text{Eqs} \) and \( S_1, S_2 \in \text{Spec} \). If \( \text{indep}(E, S_1, S_2) \), then for all \( \eta \in \text{Val} \),

\[
[E, S_1; S_2](\eta) = [E, S_2](\eta) = [E, S_1](\eta)
\]

**Proof.** Induction on \( S_1 \). In case \( S_1 = \varepsilon \) we obtain indeed:

\[
[E, \varepsilon; S_2](\eta) = [E, S_2](\eta) = [E, S_1](\eta)
\]

Next, consider \( S_1 = \sigma X; S \). Assume \( \text{indep}(E, S_1, S_2) \), then it follows that \( \text{indep}(E, S, S_2) \), so we can use the induction hypothesis. Define:

\[
F(P) := \varepsilon_X([E, S; S_2](\eta[X := P]))
\]

\[
G(P) := \varepsilon_X([E, S](\eta[X := P]))
\]

In order to show that \( F = G \), it suffices (because \( E_X \) is independent of \( S_2 \)) to show that for any \( P \in U \) and \( Y \notin \text{dom}(S_2) \):

\[
[E, S; S_2](\eta[X := P])(Y) = (\text{induction hypothesis})
\]

\[
[E, S_2](E)(\eta[X := P])(Y) = (\text{Lemma 4.5.1})
\]

\[
[E, S](\eta[X := P])(Y)
\]
Next, we finish the proof with the following calculation:

\[
\begin{align*}
[\mathcal{E}, \sigma X; S; S_2](\eta) &= [\mathcal{E}, S; S_2](\eta[X := \sigma(F)]) \\
&= (\text{induction hypothesis}) \\
[\mathcal{E}, S_2](\mathcal{E}, S)(\eta[X := \sigma(F)]) \\
&= (F = G, \text{ see above}) \\
[\mathcal{E}, S_2](\mathcal{E}, S)(\eta[X := \sigma(G)]) \\
&= [\mathcal{E}, S_2](\mathcal{E}, \sigma X; S)(\eta)
\end{align*}
\]

The next theorem shows that two disjoint blocks of equations can be swapped, provided one of them doesn’t depend on the other. Note that a dependence in one direction is allowed, and that it doesn’t matter in which direction by symmetry. Theorem 6.7 will generalize this by adding left- and right-contexts under certain conditions.

**Theorem 6.4.** Let disjoint \((S_1, S_2)\) and indep \((\mathcal{E}, S_1; S_2)\). Then \([\mathcal{E}, S_1; S_2] = [\mathcal{E}, S_2; S_1]\).

**Proof.** Induction on \(S_2\). The base case is trivial. For the induction step, let \(S_2 = \sigma X; S\). If we assume disjoint \((S_1, S_2)\) and indep \((\mathcal{E}, S_1; S_2)\), then we also obtain disjoint \((S_1, S)\) and indep \((\mathcal{E}, S_1; X)\), so we may apply the induction hypothesis \([\mathcal{E}, S_1; S_2] = [\mathcal{E}, S_2; S_1]\). Furthermore, from the same assumptions, we also get \(X \not\in \text{dom}(S_1)\) and indep \((\mathcal{E}, S_1; X)\). Let \(\eta\) be arbitrary.

\[
\begin{align*}
[\mathcal{E}, S_1; \sigma X; S](\eta) &= (\text{Lemma 6.3}) \\
[\mathcal{E}, \sigma X; S](\mathcal{E}, S_1)(\eta)) \\
&= [\mathcal{E}, S](\mathcal{E}, S_1)(\eta[X := \sigma(F)]) \\
&= \mathcal{E}_X(\mathcal{E}, S)(\mathcal{E}, S_1)(\eta[X := P]) \\
&= (\text{Lemma 6.2}) \\
&\mathcal{E}_X(\mathcal{E}, S_1)(\eta[X := P])) \\
&= (\text{Lemma 6.3}) \\
&\mathcal{E}_X(\mathcal{E}, S_1; S)(\eta[X := P])) \\
&= [\mathcal{E}, S](\mathcal{E}, S_1)(\eta[X := \sigma(G)]) \\
&= (\text{Lemma 6.2}) \\
&\mathcal{E}_X(\mathcal{E}, S_1)(\eta[X := \sigma(G)]) \\
&= (\text{Lemma 6.3}) \\
&\mathcal{E}_X(\mathcal{E}, S_1; S)(\eta[X := \sigma(G)]) \\
&= [\mathcal{E}, \sigma X; S_1; S](\eta) \\
&= (\text{Lemma 4.7 and induction hypothesis}) \\
&[\mathcal{E}, \sigma X; S_1; S](\eta)
\end{align*}
\]

This migration theorem has several interesting corollaries. First, we get right-congruence for independent specifications.
Corollary 6.5. Assume that \( \text{indep}(\mathcal{E}_1, S_1, S) \), \( \text{indep}(\mathcal{E}_2, S_2, S) \), disjoint\( (S, S_1; S_2) \) and that \( \mathcal{E}_1 \subseteq \mathcal{E}_2 \). Then \([\mathcal{E}_1, S_1] = [\mathcal{E}_2, S_2]\) implies \([\mathcal{E}_1, S_1; S] = [\mathcal{E}_2, S_2; S]\).

We also get the near-reverse of Lemma 6.3:

Corollary 6.6. Let \( \text{indep}(\mathcal{E}, S_1, S_2) \) and disjoint\( (S_1, S_2) \). Then for all \( \eta \in \text{Val} \), we have \([\mathcal{E}, S_1; S_2](\eta) = [\mathcal{E}, S_1]( [\mathcal{E}, S_2](\eta)) \).

Proof. Under the given assumptions, we obtain from Theorem 6.4 (applied from right to left) and Lemma 6.3:

\[
[\mathcal{E}, S_1; S_2](\eta) = [\mathcal{E}, S_2; S_1](\eta) = [\mathcal{E}, S_1]( [\mathcal{E}, S_2](\eta)) \]

\[
[\mathcal{E}, S_0; S_1; S_2; S_3] = [\mathcal{E}, S_0; S_2; S_3; S_1] = [\mathcal{E}, S_0; S_2; S_1; S_3]
\]

6.3. Inequalities by Swapping or Changing Signs. In this section, we will prove a few inequalities. Theorem 6.9 shows the consequence of swapping variables with a different sign; Theorem 6.10 shows the effect of changing the sign of a variable. But first, it will be shown that \( \leq \) is a left congruence:

Lemma 6.8. Let \( \mathcal{E}_1, \mathcal{E}_2 \in \text{Eqs} \) and \( S, S_1, S_2 \in \text{Spec} \). If \( \mathcal{E}_1 \) is monotonic, \( \mathcal{E}_1 \subseteq \mathcal{E}_2 \), and \([\mathcal{E}_1, S_1] \leq [\mathcal{E}_2, S_2]\), then \([\mathcal{E}_1, S_1; S] \leq [\mathcal{E}_2, S; S_2]\).

Proof. Induction on \( S \). The base case is trivial.

\[
[\mathcal{E}_1, \sigma X; S; S_1](\eta) = [\mathcal{E}_1, S; S_1](\eta[X := \sigma(F)]) \text{, where } F(P) := \mathcal{E}_1, X([\mathcal{E}_1, S; S_1](\eta[X := P])) \\
\leq \text{ (by induction hypothesis and } \mathcal{E}_1 \text{ monotonic) } \\
\mathcal{E}_1, X([\mathcal{E}_2, S; S_2](\eta[X := P])) \\
= [\mathcal{E}_2, S; S_2](\eta[X := P]) \\
= [\mathcal{E}_2, \sigma X; S; S_2](\eta) \]

Note that, by duality, the above lemma may also be applied if \( \mathcal{E}_2 \) is monotonic instead of \( \mathcal{E}_1 \). Moreover, note that (only) for monotonic \( \mathcal{E}_1 \), Lemma 4.7 would follow from Lemma 6.8.

Theorem 6.9. Assume \( \mathcal{E} \in \text{Eqs} \) is monotonic and \( X \neq Y \). Then

\[
[\mathcal{E}, S_1; \mu X; \nu Y; S_2] \leq [\mathcal{E}, S_1; \nu Y; \mu X; S_2]
\]
Proof. As in Theorem 6.1, and using $X \neq Y$, we obtain:

$$[\mathcal{E}, \mu X; \nu Y; S_2](\eta) = A(\mu(F_2), \nu(F_3)),$$

where

$$A(P, Q) = \mathcal{E}[X := P, Y := Q])$$

$$F_1(P)(Q) = E_X(A(P, \nu(F_1(P))))$$

$$F_2(P) = E_Y(A(\mu(F_2), Q))$$

$$F_3(Q) = E_Y(A(\mu(F_2), Q))$$

$$[\mathcal{E}, \nu Y; \mu X; S_2](\eta) = A(\mu(G_3), \nu(G_2)),$$

where

$$G_1(Q)(P) = E_X(A(P, Q))$$

$$G_2(Q) = E_Y(A(\mu(G_1(Q)), Q))$$

$$G_3(P) = E_Y(A(\mu(G_2(Q))))$$

By Lemma 3.2.4(a), $\mu(F_2) \leq \mu(G_3)$, and by Lemma 3.2.4(b), $\nu(F_3) \leq \nu(G_2)$, whence it follows that $[\mathcal{E}, \mu X; \nu Y; S_2](\eta) \leq [\mathcal{E}, \nu Y; \mu X; S_2](\eta)$. The theorem then follows by Lemma 6.8.

We end this section with another inequality:

**Theorem 6.10.** If $\mathcal{E}$ is monotonic, then $[\mathcal{E}, S_1; \mu X; S_2] \leq [\mathcal{E}, S_1; \nu X; S_2]$.

Proof. Let $\eta$ be an arbitrary valuation, and define $F : U \to U$ by $F(P) := \mathcal{E}(\mathcal{E}[X := P])$. We then have:

(Definition semantics)

$$[\mathcal{E}, \mathcal{S}_2](\eta[X := \mu P.F(P)]) \leq [\mathcal{E}, \mathcal{S}_2](\eta[X := \nu P.F(P)])$$

(Definition semantics)

$$[\mathcal{E}, \mu X; S](\eta) \leq [\mathcal{E}, \nu X; S_2](\eta)$$

(Definition semantics)

$$[\mathcal{E}, S_1; \mu X; S_2] \leq [\mathcal{E}, S_1; \nu X; S_2]$$

In the next section, we will see sufficient conditions under which the inequality signs of these theorems can be turned into equalities. These conditions will be phrased in terms of the dependency graph.

7. **Indirect Dependencies and Loops**

In Section 6.2, we studied direct dependencies between variables. Basically, a direct dependence of $X$ on $Y$ means that $Y$ occurs in the definition of $X$. We will now study the effect of indirect dependencies, written $X \xrightarrow{\mathcal{E}, S} Y$ (cf. the definitions in Section 4.3).

Given a specification $S$ and a computable predicate $f$ on variables, we define $\text{split}_f(S) = (S_1, S_2)$, where $S_1$ is the sublist of $S$ with those $X$ for which $f(X)$ does not hold and $S_2$
is the sublist of \( S \) with those \( X \) for which \( f(X) \) holds. Notice that, within \( S_1 \) and \( S_2 \), variables keep their order from \( S \).

The following basic facts follow directly from the definition of \( \text{split} \).

**Lemma 7.1.** Let \( \text{split}_f(S) = (S_1, S_2) \), then \( \text{dom}(S) = \text{dom}(S_1) \cup \text{dom}(S_2) \) and \( \text{disjoint}(S_1, S_2) \).

We first show how the equations in a FES may be rearranged if the specification is split in such a way that certain independence conditions are fulfilled.

**Lemma 7.2.** Let \( f \) be a predicate and \( S \) a specification such that \( \text{split}_f(S) = (S_1, S_2) \) and \( \text{indep}(E, S_2, S_1) \). Then \([E, S] = [E, S_1; S_2] \).

**Proof.** We perform induction on \( S \). The base case is trivial. Let \( \text{split}_f(S) = (S_1, S_2) \) and assume as induction hypothesis that \( \text{indep}(E, S_2, S_1) \) implies \([E, S] = [E, S_1; S_2] \). For the induction step, we consider the specification \( \sigma Y; S \) and distinguish two cases.

If \( f(Y) \) does not hold, then we have \( \text{split}_f(\sigma Y; S) = (\sigma Y; S_1, S_2) \). Accordingly, we assume \( \text{indep}(E, \sigma Y; S_2, S_1) \), which implies \( \text{indep}(E, S_2, S_1) \). We can thus apply the induction hypothesis and congruence (Lemma 4.7) to obtain \([E, \sigma Y; S] = [E, \sigma Y; S_1; S_2] \).

Otherwise, if \( f(Y) \) holds, we obtain \( \text{split}_f(\sigma Y; S) = (S_1, \sigma Y; S_2) \). Now we assume that \( \text{indep}(E, S_2, \sigma Y; S_1) \), which again implies \( \text{indep}(E, S_2, S_1) \). We also have \( \text{disjoint}(S_1, \sigma Y; S_2) \) (Lemma 7.1), so we may apply Theorem 6.7 below:

\[
[E, \sigma Y; S] = (\text{by induction hypothesis and Lemma 4.7})
\]

\[
[E, \sigma Y; S_1; S_2] = (\text{by Theorem 6.7})
\]

\[
[E, S_1; \sigma Y; S_2]
\]

We simplify notation and write \( \text{split}_{X,E}(S) \) for \( \text{split}_{\text{dep}_{X,E}}(S) \), defining the predicate \( \text{dep}_{X,E}(Y) \). If \( \text{indep}(E, X, Y) \) is computable (which we assume henceforth), then \( X \xrightarrow{E,S} Y \) is also computable since \( S \) is finite. Intuitively, if \( \text{split}_{X,E}(S) = (S_1, S_2) \), then \( S_1 \) is the sublist of \( S \) with those \( Y \) on which \( X \) does not depend indirectly; and \( S_2 \) is the sublist of \( S \) with those \( Z \) on which \( X \) does depend indirectly. Notice that, if \( X \notin \text{dom}(S) \), then \( \text{split}_{X,E}(S) = (S, \varepsilon) \).

We have the following lemma about splitting a specification based on the dependencies of \( X \):

**Lemma 7.3.** If \( \text{split}_{X,E}(S) = (S_1, S_2) \), then \( \text{indep}(E, S_2, S_1) \).

**Proof.** Assume that some \( Z \in \text{dom}(S_2) \) would use some \( Y \in \text{dom}(S_1) \) in its definition in \( E \). Then \( X \xrightarrow{E,S} Z \xrightarrow{E,S} Y \), so \( X \xrightarrow{E,S} Y \), and \( Y \) would be in \( S_2 \) and not in \( S_1 \).

The key theorem of this section states that the equations in a FES can be rearranged, such that all equations that \( X \) depends on precede all other equations, or vice versa. This is useful, because those parts can be solved independently, using Lemma 6.3. By repeatedly picking a variable in a terminal strongly connected component of the remaining variable dependency graph, one can thus solve all SCCs one by one.

**Theorem 7.4.** Let \( \text{split}_{X,E}(S) = (S_1, S_2) \). Then \([E, S] = [E, S_1; S_2] = [E, S_2; S_1] \).

**Proof.** The first equality follows from Lemmas 7.2 and 7.3. From Theorem 6.4 and Lemmas 7.1 and 7.3 it follows that also \([E, S_1; S_2] = [E, S_2; S_1] \).
Based on this reordering principle, we can prove three more interesting results, which we will do in the next subsections.

7.1. Swapping Signs and Dependency Loops. The first result states that the sign of a variable $X$ is only relevant if it depends on itself. Recall that $\bar{E}^{S} \vdash^{+} X$ indicates a nonempty path in the variable dependency graph. We first need an auxiliary lemma:

**Lemma 7.5.** If $X \notin \text{dom}(S)$ and we have $\text{indep}(E, S, X)$ as well as $\text{indep}(E, X, X)$, then $[E, \mu X; S] = [E, \nu X; S]$.

**Proof.** For $\sigma \in \{\mu, \nu\}$ and arbitrary valuation $\eta$, we have:

$$[E, \sigma X; S](\eta) = (\text{by definition of semantics})$$
$$[E, S](\eta[X := \sigma(P)])$$

where $F(P) := E_X([E, S](\eta[X := P]))$

$$= (\text{Lemma 6.2, and } X \notin \text{dom}(S) \text{ and } \text{indep}(E, S, X))$$
$$E_X((\|E, S\|\eta)[X := P])$$

$$= (\text{by definition of } \text{indep}(E, X, X))$$
$$E_X(\|E, S\|\eta)$$

$$= (\text{constant rule, Lemma 3.1.2})$$
$$[E, S](\eta[X := E_X(\|E, S\|\eta)])$$

So indeed $[E, \mu X; S] = [E, \nu X; S]$. □

**Lemma 7.6.** If not $X \xrightarrow{E, \mu X; S} X$, and $X \notin \text{dom}(S)$, then $[E, \mu X; S] = [E, \nu X; S]$.

**Proof.** Let $S_1$ and $S_2$ be such that $\text{split}_{X, E}(\sigma X; S) = (S_1, \sigma X; S_2)$, for $\sigma \in \{\mu, \nu\}$. Note that if not $\text{indep}(E, \mu X; S_2, X)$, then for some $Y \in \text{dom}(\mu X; S_2)$, by definition of split, $X \xrightarrow{E, \mu X; S_2} Y \xrightarrow{E} X$, which contradicts the assumption not $X \xrightarrow{E, \mu X; S_2} X$. From $\text{dom}(S_2) \subseteq \text{dom}(S)$, we obtain $X \notin \text{dom}(S_2)$. Hence, $X \notin \text{dom}(S_2)$ and $\text{indep}(E, \mu X; S_2, X)$, so Lemma 7.5 applies. Together with Theorem 7.4 and Lemma 4.7, we then compute:

$$[E, \mu X; S] = [E, S_1; \mu X; S_2] = [E, S_1; \nu X; S_2] = [E, \nu X; S]$$

Intuitively, the sign of $X$ is only relevant if $X$ is the most relevant variable (i.e. leftmost in the specification) on some loop in the dependency graph. So in the full theorem, we can restrict to dependencies through variables right from $X$:

**Theorem 7.7.** Assume that not $X \xrightarrow{E, \mu X; S_2} X$, and $X \notin \text{dom}(S_2)$. Then $[E, S_1; \mu X; S_2] = [E, S_1; \nu X; S_2]$.

**Proof.** By Lemma 7.6, $[E, \mu X; S_2] = [E, \nu X; S_2]$. The result follows by congruence, Lemma 4.7. □
7.2. Reordering Variables and Dependency Loops. The second result allows to swap any two neighbouring variables that don’t occur on a loop in the dependency graph.

**Lemma 7.8.** Let not $X \not\xrightarrow{E, \sigma X; \rho Y; \mathcal{S}} Y$. Then $[E, \sigma X; \rho Y; \mathcal{S}] = [E, \rho Y; \sigma X; \mathcal{S}]$.

**Proof.** Note that for some $S_1$ and $S_2$, we have $\text{split}_{X,E}(\sigma X; \rho Y; \mathcal{S}) = (\rho Y; S_1, \sigma X; S_2) = \text{split}_{X,E}(\rho Y; \sigma X; \mathcal{S})$.

Hence, by applying Theorem 7.4 twice, we obtain:

$$[E, \sigma X; \rho Y; \mathcal{S}] = [E, \rho Y; \sigma X; S_1; S_2] = [E, \rho Y; \sigma X; \mathcal{S}]$$

Again, we can strengthen this, by observing that $X$ and $Y$ can be swapped, when there is no loop that has either $X$ or $Y$ as its most relevant variable in the specification:

**Theorem 7.9.** Assume that not $X \not\xrightarrow{E, \sigma X; \rho Y; S_2} Y$. Then we have $[E, \sigma X; \rho Y; S_2] = [E, \rho Y; \sigma X; S_2]$. The result then follows by congruence, Lemma 4.7.

Note that this result strengthens Theorem 6.1 (the signs may now be different), Theorem 6.7 (we can have mutual dependencies on $S$, as long as no loop is introduced) and Theorem 6.9 (we have here equality rather than inequality).

7.3. Forward Substitution and Dependency Loops. The final result strengthens Theorem 5.3 by allowing unfolding of $Y$ in the definition of $X$, even if $Y$ precedes $X$ in the specification, provided $Y$ doesn’t depend on $X$:

**Theorem 7.10.** Let $E \in \mathcal{E}gs$ be monotonic. Let $S = S_1, \sigma Y, S_2$. Assume that not $Y \not\xrightarrow{E, S} X$. Then $[E, S] = [\text{unfold}(E, X, Y), S]$.

**Proof.** Assume not $Y \not\xrightarrow{E, S} X$. Then the following two observations hold:

1. for all $Z \in \mathcal{X}$, $Y \not\xrightarrow{E, S} Z \iff Y \not\xrightarrow{\text{unfold}(E, X, Y), S} Z$
2. $\text{split}_{Y,E}(S) = \text{split}_{Y,\text{unfold}(E, X, Y)}(S)$

The first item holds, because $\text{unfold}(E, X, Y)$ only modifies the definition of $X$, but $Y$ doesn’t refer to it. The second then follows from the definition of $\text{split}$.

Let $(L_1, L_2) := \text{split}_{Y,E}(S)$. Then, as $Y \not\xrightarrow{E, S} Y$, we have $L_2 = L_3; \sigma Y; L_4$. Note that $X \not\in \text{dom}(L_4)$, for we would then have $Y \not\xrightarrow{E, S} X$, contradicting the assumptions. Then we can compute:

$$[E, S]$$

$= \text{ (Theorem 7.4)}$

$$[E, L_1; L_3; \sigma Y; L_4]$$

$= \text{ (Theorem 5.3)}$

$$[\text{unfold}(E, X, Y), L_1; L_3; \sigma Y; L_4]$$

$= \text{ (Theorem 7.4, observation (2) above)}$

$$[\text{unfold}(E, X, Y), S]$$
8. Formalisation in Coq & PVS

We have formalised all of the above theory in both Coq [Ber08, S+22] and PVS [OS08]. A replication artefact containing these proofs is available at [NvdP23]. The formalized definitions and proofs follow the definitions and proof steps in this paper quite closely. Here, we highlight the main difference between the two formalisations.

In Coq, we captured the concepts of complete lattices and monotonic functions in typeclasses. For these, we defined several typeclass instances, for example the product lattice and composition of monotonic functions. In many cases, Coq is able to perform automatic typeclass resolution, saving us from manually proving monotonicity of complex functions, for example those in Lemma 3. Furthermore, Coq supports user-defined notation, allowing us to closely follow the notation used in the paper. The proofs for showing decidability of $X \xrightarrow{E,S} Y$ are extensive, something that is not reflected in the paper.

Our PVS definitions and proofs were originally developed under PVS version 4.2, but could be ported to version 7.1 with minimal effort. Contrary to Coq, PVS is built on classical logic and thus allows the law of excluded middle (for all propositions $P$, it holds $P \lor \neg P$). We thus do not need to supply proofs for decidability of $X \xrightarrow{E,S} Y$. This also means that we do not rely on finiteness of $S$, and thus the definition of $\rightarrow$ only depends on $E$ and the domain $E$ is restricted where necessary, e.g., in Theorem 7.7. This simplifies the proof of Lemma 7.2: it can operate on $\text{split}_{X,E}$ directly.

9. Conclusion

We provided several equalities and inequalities involving a range of operations on fixpoint equation systems (FES). We refer to Table 1 and 2 (Section 2) for a summary of the theorems. Lemmas 3.1 and 3.2 provide a useful overview on equalities and inequalities for nested fixed points in complete lattices.

We provided self-contained and detailed proofs of all results and mechanised these proofs in two proof assistants, Coq and PVS.

By the generic nature of FES, these results carry over to other formalisms such as Boolean equation systems (BES), parity games (and variations thereof), and parameterised (first-order) Boolean equation systems (PBES).

Acknowledgment

Large part of the research was carried out at the Centrum voor Wiskunde en Informatica.

References


