Fundamental Study

Equational programming in $\lambda$-calculus via SL-systems.
Part 1$^\star$

Enrico Tronci$^*,^1$

Dipartimento di Matematica Pura ed Applicata, Università dell'Aquila, Via Vetoio, Coppito,
67100 L'Aquila, Italy

Received January 1992; revised July 1995
Communicated by C. Böhm

Abstract

A system of equations in the $\lambda$-calculus is a set of formulas of $\Lambda$ (the equations) together with a finite set of variables of $\Lambda$ (the unknowns). A system $\mathcal{S}$ is said to be $\beta$-solvable ($\beta\eta$-solvable) if there exists a simultaneous substitution with closed $\lambda$-terms for the unknowns that makes the equations of $\mathcal{S}$ theorems in the theory $\beta$ ($\beta\eta$). A system $\mathcal{S}$ can be viewed as a set of specifications (the equations) for a finite set of programs (the unknowns) whereas a solution for $\mathcal{S}$ yields executable codes for such programs.

A class $\mathcal{G}$ of systems for which the solvability problem is effectively decidable defines an equational programming language and a system solving algorithm for $\mathcal{G}$ defines a compiler for such language.

This leads us to consider separation-like systems (SL-systems), i.e. systems with equations having form $xM = z$, where $x$ is an unknown and $z$ is a free variable which is not an unknown.

We show that the $\beta$ ($\beta\eta$)-solvability problem for SL-systems is undecidable.

On the other hand we are able to define a class of SL-systems (regular SL-systems) for which the $\beta$-solvability problem is decidable in Polynomial Time. Such class yields an equational programming language in which self-application is handled, constraints on executable code to be generated by the compiler can be specified by the user and (properties of) data structures can be described in an abstract way.

Keywords: Systems of equations in the $\lambda$-calculus; $\lambda$-calculus; Equational programming; Functional programming; Automated synthesis of programs
0. Introduction

Functions can be specified with equations.

0.0. Example. A primitive recursive function \( f : \mathbb{N} \to \mathbb{N} \) can be specified with the equations \( a \in \mathbb{N} \) and \( H : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \):

0.0.0. \( f(0) = a \),

0.0.1. \( f(n + 1) = H(f(n), n) \).

However equations 0.0.0, 1 (i.e. 0.0.0 and 0.0.1) do not specify a program. We can specify a program reading equations 0.0.0, 1 as rewrite rules. We have:

0.0.2. \( f(0) \mapsto a \),

0.0.3. \( f(n + 1) \mapsto H(f(n), n) \).

An interpreter for a programming language based on rewrite rules is described in [11].

If a compiler is wanted, a natural approach is to write specifications 0.0.2, 3 in a language in which also programs can be represented. This leads us to consider \( \lambda \)-calculus since it is a formal language whose expressions (\( \lambda \)-terms) can be interpreted as programs.

Equations 0.0.0, 1 can be easily written as a system of equations in \( \lambda \)-calculus.

0.1. Example. Find a \( \lambda \)-term \( F \) such that \( a, H, n, 0, s \) are, respectively, representations for (as in 0.0) \( a, H, n, 0, s \) (successor function):

0.1.0. \( F0 = a \),

0.1.1. \( Fs = H(Fn)s \).
Of course any solution $F$ for $0.1.0, 1$ is a representation for the function $f$ defined in Example 0.0. However, as $0.0.0, 1$, equations $0.1.0, 1$ do not specify a program.

The rewrite rules 0.0.2, 3 can also be written as systems of equations in $\lambda$-calculus.

0.2. Example. Find a $\lambda$-term $L$ such that $(a, H, n, 0, s$, as in 0.1):

0.2.0. $L0 = y_0,$

0.2.1. $L(s n) = y_1(Ln)n,$

where $y_0$ and $y_1$ are free (and fresh) variables (i.e. parameters).

A solution for 0.2.0, 1 is a $\lambda$-term $L$ (containing $y_0$ and $y_1$) satisfying 0.2.0, 1. Any solution $L$ for 0.2.0, 1 yields a representation $F \equiv L[y_0 := a, y_1 := H]$ (see Section 2 for $:=)$ for the program (and hence for the function $f$) specified by 0.0.2, 3.

A compiler for a programming language based on equations like 0.2.0, 1 is in [3].

To avoid hidden parts the equations can be arranged so that we always look for closed $\lambda$-terms. Moreover $n$ can also be replaced by a free variable. Thus, equations 0.2.0, 1 become:

Find a closed $\lambda$-term $D$ such that (where $y_0, y_1$ and $z$ are free variables):

0.2.2. $Dy_0y_10 = y_0,$

0.2.3. $Dy_0y_1(sz) = y_1(Dy_0y_1z)z.$

The program specified by 0.2.2, 3 can be represented with the $\lambda$-term $G = DaH$ (= $F$).

In fact we have $G0 = DaH0 = a$ and $G(sz) = DaH(sz) = H(DaHz)z = H(Gz)z.$ Thus, we can regard 0.2.2, 3 as equations equivalent to 0.0.2, 3.

Of course from $D$ we can obtain $L$ defining $L = Dy_0y_1$ and from $L$ we can obtain $D$ defining $D = \lambda y_0y_1. L.$

0.3. Remark. Equations 0.2.2, 3 depend only on the LHS of the rewrite rules 0.0.2, 3, i.e. equations 0.2.2, 3 depend only on the schemata of specifications allowed in our programming language. Hence, a solution $D$ for 0.2.2, 3 represents the control structure of the program (i.e. pattern matching, etc.) that depends only on the schemata of equations used in the specifications. Thus, given a program (e.g. 0.0.2, 3) in a functional language we can divide the compilation process into four steps.

Step 0: Type checking (which essentially depends only on the RHS parts of the specifications) (e.g. $a, H$ in 0.2 for 0.0.2, 3).

Step 1: Generation of the equations describing the control structure of the program (which essentially depends only on the LHS parts of the specifications) (e.g. 0.2.2, 3 for 0.0.2, 3).

Step 2: Construction of executable code for the control structure (e.g. $D$ satisfying 0.2.2, 3).
Step 3: Generation of a code for the given program (e.g. $G$ as in 0.2). Steps 1 and 3 are straightforward. In this paper we focus on Step 2.

Of course an analogous approach can be taken for partial recursive functions.

**0.4. Example.** A partial recursive function $f: \mathbb{N} \rightarrow \mathbb{N}$ can be specified with the rewrite rules $(a \in \mathbb{N}$ and $H: (\mathbb{N} \rightarrow \mathbb{N}) \times \mathbb{N} \rightarrow \mathbb{N})$:

0.4.0. $f(0) \rightarrow a$.

0.4.1. $f(n + 1) \rightarrow H(f, n)$.

Reasoning as in 0.2 the control structure for 0.4.0,1 can be found looking for a closed term $Q$ satisfying the equations:

0.4.2. $Q_{y_0y_1}\downarrow = y_0$.

0.4.3. $Q_{y_0y_1}(\beta z) = y_1(Q_{y_0y_1})z$.

Thus, reasoning as in 0.2, any solution $Q$ for 0.4.2,3 yields a representation $G' \equiv QaH$ for the program (and hence for the function) specified by 0.4.1,1. Remark 0.3 still holds.

Of course equations 0.4.2,3 are more general than 0.2.2,3. From a solution $Q$ for 0.4.2,3 we can obtain a solution $D$ for 0.2.2,3 defining $D = \lambda y_0y_1. Q_{y_0}( \lambda f z. y_1 (f z))z$.

Since $G' \equiv (Q_{y_0y_1})[y_0 := a, y_1 := H]$ by abuse of language we will also call $Q_{y_0y_1}$ a representation for the program 0.4.0,1.

According to the previous discussion a system $\mathcal{S}$ of equations in the $\lambda$-calculus can be viewed as a set of specifications (the equations) (e.g. 0.2.2,3) for a finite set of programs (the unknowns) e.g. $D$ in 0.2.2,3 whereas a solution for $\mathcal{S}$ (e.g. an actual value for $D$) yields executable codes for these programs.

A class of systems $\mathfrak{S}$ for which the solvability problem is effectively decidable defines an equational programming language and an algorithm to solve the systems in $\mathfrak{S}$ yields a compiler (which may or may not carry out code optimization) for such language. Both specifications and results of the compilation process can be represented inside the $\lambda$-calculus. This would not be possible in a term rewriting system without some abstraction mechanism. Moreover, specifying sets of $\lambda$-terms with equations does not leave out any interesting set, to be precise: any recursively enumerable and $\beta$-closed set of closed $\lambda$-terms is the set of solutions to a combinator equation [14].

These features make $\lambda$-calculus appealing as a *calculus* for automated synthesis of programs when specifications are expressed with equations. Unfortunately, the existence of a solution for a system (as well as the existence of a program satisfying given specifications) is, in general, undecidable. Nevertheless many interesting equational languages have been defined in the literature (e.g. interpreter [11], compiler [3, 7, 12]).
Though almost all the equational languages allow the specification of wide classes of recursive functions, there are drastic differences with respect to the kind of equations the user is allowed to write. The more schemata of equations are allowed in a language, the easier is to write specifications. Moreover, limiting the class of admissible equations might reduce the class of definable program properties.

0.5. Example. Consider the numerical system \((\mathcal{Q}, s, p, \text{Zero})\) (as in [2]), where (the \(\lambda\)-terms \(I, U_\uparrow\) are defined in 2.0):

\[0 = \lambda ab. b, \quad s \equiv \lambda ab. bab, \quad p \equiv \lambda b. bU_\downarrow U_\uparrow, \quad \text{Zero} \equiv \lambda b. b(U_\downarrow U_\uparrow)U_\uparrow.\]

Find a \(\lambda\)-term \(F\) s.t.:

0. \(F\) represents the program in 0.4.0, 1;

1. \(F\) has form \(\lambda t. tF_1F_2\).

We look for \(D \in \mathcal{A}^0\) s.t. \((\vec{y} = y_0, y_1)\):

0.5.0. \(D\vec{y}0 = y_0(D\vec{y})\vec{0}\) \((\text{constraint } 0)\).

0.5.1. \(D\vec{y}(sz) = y_1(D\vec{y})z,\)

0.5.2. \(D\vec{y}(\lambda ab. z) = z\) \((\text{constraint } 1)\).

A possible \(\beta\)-solution is

\[D \equiv GG = \lambda y. t (\lambda a_1a_2a_3. y_1(t(U_\downarrow I)GG\vec{y})a_1)(y_0(tIIGG\vec{y})t),\]

where

\[G \equiv \lambda u. t(\lambda a_1a_2a_3. y_1(t(U_\downarrow I)uu\vec{y})a_1)(y_0(tIIGG\vec{y})t).\]

Hence, \(F \equiv D\vec{y}\) (note that \(F\) has normal form). Equations 0.5.1, 1 are sufficient to specify any partial recursive function (e.g. see [15, 0.0]), however they cannot express any constraint on the executable code of a program. This can be done using equation 0.5.2. The system 0.5.0, 1 can be transformed into an X-separability problem (3.1) [7, 8], but the system 0.5.0–0.5.2 (i.e. 0.5.0, 0.5.1, 0.5.2) cannot because of the presence of equation 0.5.2.

An unpleasant feature of the known compilers for equational programming is that the user (or someone else for him) has to specify the actual representation of the data structures (as we did in 0.1–0.5). Since data structures are control structures we can leave this task to our compiler.

0.6. Example. Find a \(\lambda\)-term \(F\) and a numerical system \((\mathcal{Q}, s, p, \text{Zero})\) s.t.:

0. \(F\) represents the program in 0.4.0, 1;

1. \(F\) has form \(\lambda t. tF_1F_2\);

2. a numeral applied to its constructors realizes an arbitrarily given partial recursive function (e.g. specified with equations similar to 0.4.0, 1).
We look for $D, D_0, D_s, D_p, D_{\text{Zero}} \in A^0$ s.t. ($\bar{y} \equiv y_0, y_1, y_2, y_3, y_4, y_5$):

0.6.0. $D_p \bar{y}(D_0 \bar{y}) = (D_0 \bar{y})$ (equations 0.6.0–3 specify the data structure),

0.6.1. $D_p \bar{y}(D_s \bar{y}z) = z$ ($p = D_p \bar{y}$ is a left inverse for $s = D_s \bar{y}$),

0.6.2. $D_{\text{Zero}} \bar{y}(D_0 \bar{y}) = y_2$ ($\text{Zero} = D_{\text{Zero}} \bar{y}$ recognizes the constructors (i.e. $\bot \equiv D_0 \bar{y}, s$)),

0.6.3. $D_{\text{Zero}} \bar{y}(D_s \bar{y}z) = y_3$,

0.6.4. $D \bar{y}(D_0 \bar{y}) = y_0(D \bar{y})(D_0 \bar{y})(D_s \bar{y})(D_s \bar{y})(D_{\text{Zero}} \bar{y})$ (constraint 0),

0.6.5. $D \bar{y}(D_s \bar{y}z) = y_1(D \bar{y})(D_0 \bar{y})(D_s \bar{y})(D_p \bar{y})(D_{\text{Zero}} \bar{y})z$.

0.6.6. $D \bar{y}(\lambda ab. z) = z$ (constraint 1)

0.6.7. $D_0 \bar{y}(D_s \bar{y})(D_0 \bar{y}) = y_4(D \bar{y})(D_0 \bar{y})(D_s \bar{y})(D_p \bar{y})(D_{\text{Zero}} \bar{y})$ (constraint 2),

0.6.8. $D_s \bar{y}z(D_s \bar{y})(D_0 \bar{y}) = y_5(D \bar{y})(D_0 \bar{y})(D_s \bar{y})(D_p \bar{y})(D_{\text{Zero}} \bar{y})z$.

The system 0.6.0–0.6.8 has a $\beta$-solution. Defining $F = D \bar{y}, \bot = D_0 \bar{y}, s = D_s \bar{y}, p = D_p \bar{y}, \text{Zero} = D_{\text{Zero}} \bar{y}$, we solve our problem. With this specification of the data structure we always get a one shot predecessor $p$ (i.e. the length of the computation of $p \eta$ is constant since $s$ is left-invertible in $A$). Consider the equation

0.6.9. $D \bar{y}U^2_\frac{1}{2} = U^2_\frac{1}{2}$.

The system 0.6.0–0.6.9 is still $\beta$-solvable, however the system 0.5.0–0.5.2, 0.6.9 (with $\bar{y}$ as in 0.6) is no longer $\beta$-solvable because there is a conflict between 0.5.0 and 0.6.9 since we chose $\bot = U^2_\frac{1}{2}$ in 0.5. To avoid these unnatural conflicts the actual representation of the data structures should be chosen by the compiler (i.e. the constructors of the data structures should be regarded as unknowns in the system).

The system 0.6.0–0.6.8 cannot be transformed into an $X$-separability problem (3.1) because of the presence of equations 0.6.1, 6. Moreover, the system 0.6.0–0.6.8 cannot be solved with the methods in [3] because the representation of the data structure is an unknown (and it is determined together with the representation for the program).

Because after all the constructors of a data structure are programs nothing prevents us from putting additional constraints on them.

0.7. Example. Find a representation $(\bot, s, p, \text{Zero})$ for the natural numbers s.t.: 0. the application of two natural numbers realizes an arbitrarily given partial recursive function; 1. the $\lambda$-term representing the successor has form $\lambda ab. bL_1L_2$. 
We look for \( D_0, D_s, D_p, D_{\text{Zero}} \in A^0 \) s.t. (\( \bar{y} \equiv y_0, y_1, y_2, y_3, y_4, y_5 \)):

0.7.0. \( D_p \bar{y}(D_0 \bar{y}) = D_0 \bar{y} \) (specification of the data structure (as in 0.6.0–3)),

0.7.1. \( D_p \bar{y}(D_s \bar{y})z = z \).

0.7.2. \( D_{\text{Zero}} \bar{y}(D_0 \bar{y}) = y_4 \),

0.7.3. \( D_{\text{Zero}} \bar{y}(D_s \bar{y})z = y_5 \),

0.7.4. \( D_0 \bar{y}(D_0 \bar{y}) = y_0(D_0 \bar{y})(D_s \bar{y})(D_p \bar{y})(D_{\text{Zero}} \bar{y}) \) (constraint 0),

0.7.5. \( D_0 \bar{y}(D_s \bar{y})z = y_1(D_0 \bar{y})(D_s \bar{y})(D_p \bar{y})(D_{\text{Zero}} \bar{y})z \).

0.7.6. \( D_s \bar{y}z(D_0 \bar{y}) = y_2(D_0 \bar{y})(D_s \bar{y})(D_p \bar{y})(D_{\text{Zero}} \bar{y})z \),

0.7.7. \( D_s \bar{y}u(D_s \bar{y})z = y_3(D_0 \bar{y})(D_s \bar{y})(D_p \bar{y})(D_{\text{Zero}} \bar{y})uz \),

0.7.8. \( D_s \bar{y} \Omega(\lambda abz) = z \) (constraint 1).

Choosing \( 0 \equiv D_0 \bar{y}, s \equiv D_s \bar{y}, p \equiv D_p \bar{y}, \text{Zero} = D_{\text{Zero}} \bar{y} \) yields the wanted representation for the natural numbers.

A possible \( \beta \)-solution for this system is (\( \bar{v} \equiv v_1, v_2, v_3, v_4 \)) (the terms \( P_\alpha \) are defined in 2.0):

\[
D_0 \equiv G_0^* G_0^* G_1^* D_p D_{\text{Zero}}, \quad D_s \equiv G_0^* G_0^* G_1^* D_p D_{\text{Zero}}, \quad D_p \equiv \lambda \bar{y} t. t P_2^u \cup_3 t,
\]

\[
D_{\text{Zero}} \equiv \lambda \bar{y} t. t P_2(\lambda a_1 \ldots a_4. y_3)(\lambda a_1 a_2 a_3. y_4)t,
\]

where

\[
G_0^* \equiv \lambda \bar{v} \bar{y} t.((G_0 \bar{y} t)[y_0 := y_0(t(U_1^2 I)v_1 \bar{v} \bar{y}))(t(U_1^2 I)v_2 \bar{v} \bar{y}))(v_3 \bar{y}))(v_4 \bar{y})),
\]

\[
y_1 \equiv y_1(t(U_1^2 I)v_1 \bar{v} \bar{y}))(t(U_1^2 I)v_2 \bar{v} \bar{y}))(v_3 \bar{y}))(v_4 \bar{y})),
\]

\[
G_1^* \equiv \lambda \bar{v} \bar{y} t. t_2.[y_2 := y_2(t_2(U_1^2 I)v_1 \bar{v} \bar{y}))(t_2(U_1^2 I)v_2 \bar{v} \bar{y}))(v_3 \bar{y}))(v_4 \bar{y})),
\]

\[
y_3 \equiv y_3(t_2(U_1^2 I)v_1 \bar{v} \bar{y}))(t_2(U_1^2 I)v_2 \bar{v} \bar{y}))(v_3 \bar{y}))(v_4 \bar{y})),
\]

\[
G_0 \equiv \lambda \bar{y} t. t(P_2(\lambda a_1 \ldots a_6. y_1 a_2)(\lambda a_1 \ldots a_4. y_0)t),
\]

\[
G_1 \equiv \lambda \bar{y} t. t_2(t_2 P_2(\lambda a_1 \ldots a_7. y_3 a_7 a_8)(\lambda a_1 \ldots a_4. y_2)t_2) t_1.
\]

System 0.7.0–0.7.8 cannot be transformed into an \( X \)-separability problem (3.1) because of the presence of equations 0.7.1,8 and, as in 0.6, cannot be solved with the methods in [3].

Of course analogous considerations apply to any data structure definable with a (heterogeneous) term algebra (6.4).

Not every program specification with rewrite rules can be \textit{faithfully} transformed into a system of equations like those in 0.5–0.7.
0.8. Example. Let Boole = \{false, true\}. Consider the rewrite rules (\(H_0, H_1 \in \text{Boole}\)):

0.8.0. \(g(g(\text{false})) \rightarrow H_0\),

0.8.1. \(g(\text{false}) \rightarrow H_1\),

0.8.2. \(g(H_1) \rightarrow H_0\),

0.8.3. \(g(\text{false}) \rightarrow H_1\).

The specifications in 0.8.0,1 and 0.8.2,3 are consistent (i.e. confluent) iff \([H_1 = \text{false} \Rightarrow H_0 = H_1]\). To the rules 0.8.0,1 we associate the system (\(\text{false} = U_1^\text{false} \text{ and } \text{true} = U_2^\text{true}\))

0.8.4. \(G_{y_0 y_1}(G_{y_0 y_1} \text{false}) = y_0\),

0.8.5. \(G_{y_0 y_1} \text{false} = y_1\),

with as unknown the closed term \(G\).

However, the system 0.8.4,5 does not have any solution. In fact let \(G\) be a closed term satisfying 0.8.4,5. Then from 0.8.4,5 we have \(G_{y_0 y_1} y_1 = y_0\) and \(G_{y_0 y_1} \text{false} = y_1\). Thus \(G_{y_0 y_1} \text{false} = \text{false}\). This is absurd, thus 0.8.4,5 does not have any solution even though when \(H_0 = H_1 = \text{false}\), the specifications 0.8.0,1 are consistent. The reason for this fact is that there are values for \(H_0\) and \(H_1\) that make the specifications 0.8.0,1 inconsistent (i.e. not confluent), whereas the \(\beta\)-solvability of 0.8.4,5 would guarantee the consistency of the specifications 0.8.0,1 for any choice of \(H_0\) and \(H_1\).

The rules 0.8.2,3 are translated into the system (with as unknown the closed term \(G\))

0.8.6. \(G_{y_0 y_1} H_1 = y_0\),

0.8.7. \(G_{y_0 y_1} \text{false} = y_1\).

The system 0.8.6,7 has a \(\beta\)-solution iff \(H_1 \neq \text{false}\). Note that the condition \(H_1 \neq \text{false}\) guarantees the consistency of the specifications 0.8.2,3 for any choice of \(H_0\).

When \(H_0 = H_1 = \text{false}\), the specifications 0.8.0,1 and 0.8.2,3 are consistent, but the systems 0.8.4,5 and 0.8.6,7 have no \(\beta\)-solutions.

More in general, the \(\beta\)-solvability of the system of equations \(\mathcal{S}\) that we associate with a specification \(R\) given by rewrite rules guarantees the consistency of \(R\) and of any specification \(R'\) differing from \(R\) only for the values of the “\(H\)’s” in the RHS terms. Thus, specifications like 0.8.4,5 or 0.8.6,7 with \(H_1 = \text{false}\) are rejected by our compiler.

The problem of finding a common \(\beta\) (\(\beta\eta\))-left-inverse for a finite set \(\mathfrak{F} = \{M_1, \ldots, M_n\}\) of closed \(\lambda\)-terms (combinators) can also be formulated as the \(\beta\)-solvability problem for systems like those in 0.5-0.7 (namely find a combinator \(L\) s.t. \(\forall i \{1, \ldots, n\} L(M_iz) = z\)). It is known [4,10] that the problem of finding a \(\beta\)-left-inverse for a combinator is decidable. However, here we show (4.1) that the problem of finding a common \(\beta\) (\(\beta\eta\))-left-inverse for a finite set of combinators is undecidable.
Systems like those in 0.5–0.7 cannot be solved with the methods in [3, 5–8] or [12] (3.2). Unfortunately the $\beta$ ($\beta\eta$)-solvability problem for this kind of systems is, in general, undecidable (4.1 and 4.3). The simultaneous presence of self-application (e.g. equation 0.7.4) and bounding on the size of (subterms of) the solutions (e.g. equation 0.7.8) are the main difficulties to face for this kind of systems. Consider equation 0.5.2. It implies that $D$ has form $\lambda t. t L_1 \ldots L_m$. Moreover, $m$ must be equal to the number of abstractions in $\lambda a b. z$. Thus, $m = 2$ and $D = \lambda t. t L_1 L_2$. The smaller $m$ the more cleverness we need to find a solution. Since, in general, too much cleverness is needed, the $\beta$ ($\beta\eta$)-solvability problem for systems similar to those in 0.5–0.7 is, in general, undecidable.

In this paper we define a class of systems (5.3, 6.2) (strictly larger than the classes introduced in [7, 8]) containing systems like those in 0.5–0.7 and for which the $\beta$-solvability problem is decidable in Polynomial Time (5.2, 6.2). This class defines an equational programming language in which constraints on the executable codes to be generated by the compiler can be specified (e.g. as 0.5.2) and (properties of) data structures can be described in an abstract way (e.g. as in 0.6, 7).

1. Summary

Section 2 reviews a few definitions about the $\lambda$-calculus and introduces some conventions.

Section 3 defines SL-systems (or, if we like, SL-specifications). SL-systems are a particular class of systems of equations (i.e. of program specifications). All the systems that we will study can (and will) be transformed into SL-systems.

Section 4 gives (alas!) some undecidability results. In particular we show that the following problems are undecidable: (4.1) $\beta$-solvability for SL-systems (i.e. find executable codes, if any, satisfying a given set of SL-specifications); (4.3) $\beta$-solvability for systems like those in 0.5–0.7.

Section 5 gives our main result: There is an interesting class of SL-systems (regular SL-systems) for which the following tasks can be carried out in Polynomial Time: to test if an SL-system is regular; to test if a regular SL-system has a $\beta$-solution; to construct a $\beta$-solution (if any) to a regular SL-system.

Section 6 is the final step toward the construction of our compiler. Using the algorithms in Section 5 we give a Polynomial Time algorithm to construct $\beta$-solutions (if any) for a class of systems of equations which is large enough to be used as an equational programming language (e.g. it includes specifications like those in 0.5, 0.6, 0.7). It is in this sense that regular SL-systems are interesting.

2. The $\lambda$-calculus

We review a few definitions about the $\lambda$-calculus. We assume the reader familiar with [1] of which, unless otherwise stated, we use notations and conventions.
Var is the set of variables of A, the symbol \( \equiv \) denotes syntactic equality; \( \bar{M} \equiv M_1, \ldots, M_n; |\bar{M}| = n; \{\bar{M}\} = \{M_1, \ldots, M_n\}.

2.0. Example. The following are \( \lambda \)-terms: 

\[ U_i^n \equiv \lambda x_1 \ldots x_n. x_i \quad (1 \leq i \leq n), \]

\[ I \equiv U_1^1, \]

\[ K \equiv U_2^2, \]

\[ \omega \equiv \lambda x. xx, \quad \Omega \equiv \omega \omega \] \( (\Omega \) represents a nonterminating program), 

\[ P_q \equiv \lambda x_1 \ldots x_{q+1}. x_{q+1}x_1 \ldots x_q. \langle M_1, \ldots, M_n \rangle \equiv P_p M_1 \ldots M_n. \]

A term \( M \) is said to be \( \lambda \)-free if \( M \equiv \eta \bar{M} \). \( \bar{M} \subseteq A \) is said to be \( \lambda \)-free if its elements are \( \lambda \)-free. \( A[\_][A^0[\_]] \) is the set of contexts on \( A \) (with no free variables) \( [1, 2.1.18] \).

If \( Z \subseteq \text{Var} \) then \( M[Z := N] \) denotes the result of substituting \( N \) for all the (free) occurrences of each \( z \in Z \) in \( M \). If \( z \in \text{Var} \) we write \( M[z := N] \) for \( M[(z} := N] \). If \( z \in \text{Var} \) we write \( M[z := N] \) for \( M[(z} := N] \) and \( (M\equiv N) \) \( \equiv [Z := Q] \) for \( M[Z := Q] \equiv N[Z := Q] \). Form \( (A) \) \( = \{M = N \in A|M, N \in A\} \). The elements of Form \( (A) \) are called formulas of \( A \). A theory \( T \) is a set of formulas of \( A \).

If \( M = N \in T \) we write \( M =_T N \) and we say that \( M = N \) is a theorem in \( T \). We write \( M =_{(n)} N \) for \( M = N \in \lambda (\lambda \eta) [1, 2.1.4, 28] \) (it will be clear from the mathematical context if \( M = N \) is a formula or a theorem of \( \lambda \)).

A \( \lambda \)-term \( M \) is said to be \( \beta \)-solvable iff there exists a \( \lambda \)-term \( \lambda a. y \bar{Q} \) s.t. \( M = \lambda a. y \bar{Q} \). SOL is the set of solvable \( \lambda \)-terms \( [1, 2.2.10-12] \). We say that \( M \) has \( \eta \)-normal form \( [1, 3.1.8] \).

Boole = \{true, false\} is the set of boolean values. If \( f \) is a function from \( Q \) to Boole and \( q \in Q \) we say that \( q \) satisfies \( f \) iff \( f(q) = true \).

2.1. Definition. (i) A system (of equations) is a pair \((\Gamma, X)\) where \( \Gamma \subseteq \text{Form}(A) \) and \( X \) is a finite subset of \( \text{Var} \). Unless otherwise stated we will assume that \( \Gamma \) is finite.

(ii) Let \( \mathcal{S} = (\Gamma, X) \) be a system. A formula \( M = N \in \Gamma \) is said to be an equation of \( \mathcal{S} \). By abuse of language we write also \( M = N \in \mathcal{S} \). A variable \( x \in X \) is said to be an unknown of \( \mathcal{S} \). Unless otherwise stated equations are considered up to \( \beta \)-conversion, e.g. \( \mathcal{S}_1 = (\{z. xz)z = x\}, \{x\}) = (\{xz = x\}, \{x\}) \) is a system with one equation \( (xz = x) \) and one unknown \( (x) \).

(iii) Let \( T \) be a theory and \( \mathcal{S} = (\Gamma, \{x_1, \ldots, x_n\}) \) be a system. \( \mathcal{S} \) is said to be \( T \)-solvable iff there exists \( D_1, \ldots, D_n \in A^0 \) s.t. \( \forall M = N \in \mathcal{S} \) \( D[M] =_T D[N] \), where \( D[\_] \equiv (\lambda x_1 \ldots x_n[\_]) D_1 \ldots D_n. \)

(iv) If \( \mathcal{S} = (\Gamma, \{x\}) \) and \( D[\_] \equiv (\lambda x[\_]) D \) is a \( T \)-solution for \( \mathcal{S} \) by abuse of language we say also that \( D \) is a \( T \)-solution for \( \mathcal{S} \).

(v) The \( T \)-solvability problem for a system \( \mathcal{S} \) is the problem of deciding if \( \mathcal{S} \) is \( T \)-solvable.

2.2. Notation. When, as we did in Sections 0 and 1, we regard a system \( \mathcal{S} = (\Gamma, X) \) as a set of program specifications we have: \( \Gamma \) is the set of program specifications in \( \mathcal{S} \), \( X \) is the set of unknown programs (i.e. the programs for which we need to find executable codes) and a \( \beta \)-solution for \( \mathcal{S} \) constitutes the executable codes satisfying the specifications in \( \mathcal{S} \). Thus, to test if there are programs (executable codes) satisfying

...
the specifications in $\mathcal{S}$ means to test if $\mathcal{S}$ has a $\beta$-solution ($\beta$-solvability problem for $\mathcal{S}$).

Of course it is a matter of taste which words to use. In the following we will usually speak about systems of equations and $\beta$-solutions, however we will speak about specifications, programs and executable codes whenever we feel this can help to convey the intuition behind formal definitions.

2.3. Notation. In the following we only consider $\lambda$-terms having a finite Böhm-tree. Moreover, as usual, we assume of having an oracle that given a $\lambda$-term $M$ computes $BT(M)$ (or, equivalently, we assume that $\lambda$-terms are given in $\beta\Omega$-normal form). Moreover, when dealing with complexity issues we assume that any label in $BT(M)$ can be represented in a cell of memory.

2.4. Definition. We measure complexity using the following definitions. Let $\mathcal{S}$ be a finite subset of $\Lambda$ and $\mathcal{S} = (\Gamma, X)$ be a system.

- $Node(\mathcal{S}) = \max\{\text{number of nodes in } BT(M) \mid M \in \mathcal{S}\}$.
- $Size(\mathcal{S}) = Card(\mathcal{S}) \cdot Nodes(\mathcal{S})$.
- $Size(\mathcal{S}) = Size(\{M \mid \exists M = N \in \mathcal{S}\}) + Size(\{N \mid \exists M = N \in \mathcal{S}\}) + Card(X)$.

The input size for an algorithm taking a system $\mathcal{S}$ as input is $Size(\mathcal{S})$, e.g. a Polynomial Time algorithm to test the $\beta$-solvability for a system $\mathcal{S}$ is an algorithm that runs in time polynomial in $Size(\mathcal{S})$.

3. SL-systems

Many interesting systems (e.g. those in 0.5–0.7) can be transformed into systems with equations having form $xM = z$. We call the latter SL-systems (or, if we like, SL-specifications). Solving SL-systems will be the core of our system solving algorithm.

3.0. Definition. (i) A system $\mathcal{S} = (\Gamma, X)$ is said to be an SL-system (separation like) if its equations have form $xM = z$, where $x \in X$ and $z \notin X$.

(ii) An SL-system $\mathcal{S} = (\Gamma, \{x\})$ is said to be an HSL-system (head separation like) if its equations have form $xM = z$, where $x \notin FV(\bar{M})$. Thus, an HSL-system is an SL-system without self-application.

3.1. Example. Separability [1, 10.4.4; 9] and $X$-separability [7, 8] are $\beta$-solvability problems for particular HSL-systems and SL-systems.

Let $\mathcal{S} = \{M_1, \ldots, M_m\} \subset \Lambda$ and $\bar{y} \equiv y_1 \ldots y_m$ be fresh variables.

(a) $\mathcal{S}$ is said to be separable iff the HSL-system $\mathcal{S} = (\{x\bar{y}M_1 = y_1, \ldots, x\bar{y}M_m = y_m\}, \{x\})$ is $\beta$-solvable.
(b) Let $\mathcal{F}$ be $\lambda$-free and $X = \{x_1, \ldots, x_n\}$ ($(X \cap \text{FV}(\mathcal{F}))$ may be nonempty). $\mathcal{F}$ is said to be $X$-separable iff the SL-system $\mathcal{S} = \{(\lambda x_1 \ldots x_n. M_i) \ (x_1 \bar{y}) \ldots (x_n \bar{y}) = y_1 | i = 1, \ldots, m\}, X)$ is $\beta$-solvable. Separability is a particular case of $X$-separability.

3.2. Example. An HSL-system is more general than a separability problem and an SL-system is more general than an $X$-separability problem.

Let $\mathcal{S} = \{(x(\lambda a. aU^2_1(aS\Omega)) = z, x(\lambda a. aU^2_2(a\Omega z\Omega)) = z, x(\lambda a. z) = z\}, \{x\}$.

A possible $\beta$-solution for $\mathcal{S}$ is $D = t \lambda t. tH_1 \ldots H_q t_i \ldots t_{ir}$, but there is no $\beta$-solution for $\mathcal{S}$ having form

$$3.2.0. \quad \lambda t_1 \ldots t_e \ldots t_p. t_e H_1 \ldots H_q t_i \ldots t_{ir},$$

where $H_1, \ldots, H_q \in \Lambda^0$ and $r \leq p$, i.e. a $\beta$-solution for $\mathcal{S}$ must have a local memory, e.g. the rightmost occurrence of $t$ in $D$.

More in general, we say that an SL-system $\mathcal{Q}$ requires local memory iff $\mathcal{Q}$ does not have any $\beta$-solution of the form in 3.2.0.

The SL-system $\mathcal{S}$ requires local memory. This can be proved by contradiction. Let $L$ be a $\beta$-solution for $\mathcal{S}$. Assume that $L$ has form as in 3.2.0.

Case 0: $L = \lambda t. tH$. Then

$$L(\lambda a. aU^2_1(aS\Omega)) = HU^2_2(H\Omega z\Omega),$$

$$L(\lambda a. aU^2_2(a\Omega z\Omega)) = HU^2_2(H\Omega z\Omega).$$

Because $H \in \Lambda^0 \cap \text{SOL}$ we have two subcases:

$H = \lambda t_1 t_2. t_1 Q$. Then $(H\Omega z\Omega) \notin \text{SOL}$. Absurd.

$H = \lambda t_1 t_2. t_2 Q$. Then $(H\Omega z\Omega) \notin \text{SOL}$. Absurd.

Case 1: $L = \lambda t. tt$. Then $L(\lambda a. aU^2_1(aS\Omega)) \neq z$. Absurd. Thus, the thesis follows.

On the other hand if a $(X)$-separability problem has a $\beta$-solution then it has a $\beta$-solution having form 3.2.0 [1, 10.4.12; 6, 9].

As a matter of fact HSL-systems are so powerful that their $\beta$ ($\beta\eta$)-solvability problem is, in general, undecidable (4.1). However, the $\beta$-solvability for an interesting class of SL-systems can be characterized (5.2).

4. Undecidability results

To avoid hopeless search for necessary and sufficient conditions of solvability for systems of equations we give some undecidability results. In particular in 4.1 we show that the $\beta$ ($\beta\eta$)-solvability problem for HSL-systems (3.0) is undecidable. Moreover, we show that the $\beta$-solvability problem for systems like those in 0.5–0.7 is, in general, undecidable (4.3).

The following idea comes from [13].
4.0. Definition. Let $k \in \mathbb{N}$. We define $P_k \equiv \lambda x. G_k x [ k ] \ast \mathbb{I} \mathbb{I}$, where $G_k \equiv \lambda t. F_k^t$, $F_k^t$ is obtained from $F_k$ replacing each redex $(\lambda a. P)Q$ in $F_k$ by $(\lambda a. P)Q$ (hence $F_k^t$ is in nf), $F_k \lambda$-defines $\{k\}$ (the $k$th partial recursive function) as in [1, 8.4]. Note that $P_k$ has nf. We have $P_k \mathbb{I} \equiv \beta$ if $\{k\}(k) \downarrow$ then $\mathbb{I}$ else unsolvable.

The $\beta (\beta \eta)$-solvability problem for HSL-systems (and thus, a fortiori, for SL-systems) is undecidable (4.1).

4.1. Proposition. Let $\mathcal{S} = \{x(\lambda a. a z) = z, x(\lambda a. a(a u)) = u, x(\lambda a. a(a(P_k a y))) = y\}, \{x\}$. Then $\mathcal{S}$ is $\beta (\beta \eta)$-solvable iff $\{k\}(k) \downarrow$.

Proof. ($\Rightarrow$) $D \equiv \lambda t. \mathbb{I}$ is a $\beta$-solution for $\mathcal{S}$.

($\Rightarrow$) Let $D[\ ]$ be a $\beta \eta$-solution for $\mathcal{S}$. Let $\sigma_1$ be the standard reduction $D[x(\lambda a. a z)] \rightarrow_{\beta} F \equiv \lambda t_1 \ldots t_r. (\lambda a. a z)H_{1,1} \ldots H_{1,h} \rightarrow_{\eta} Q \equiv \lambda t_1 \ldots t_q. z Q_{1,1} \ldots Q_{1,q} \rightarrow_{\eta} z$, where $F$ is the first term in the reduction $\sigma_1$ in which $(\lambda a. a z)$ is on the head and the leftmost occurrence of $z$ in $F$ will come on the head in $\sigma_1$ and $Q$ is the first term in $\sigma_1$ where $z$ is on the head. Let $* \equiv [\{z, u, y\} : = \Omega]$.

We have $\forall i \in \{1, \ldots, q\} [Q_{1,i} \rightarrow_{\eta} t_i$ and $Q_{1,i} \rightarrow_{\eta} t_i]$. and

$$\lambda t_1 \ldots t_r. (\lambda a. a z)H_{1,1} \ldots H_{1,h} \rightarrow_{\eta} \lambda t_1 \ldots t_r. H_{1,1}^* z H_{1,2} \ldots H_{1,h}^*$$

Consider the standard reduction $\sigma_2 : D[x(\lambda a. a(a u))] \rightarrow_{\eta} u$. We have

$$D[x(\lambda a. a(a u))] \rightarrow_{\beta} \lambda t_1 \ldots t_r. (\lambda a. a(a u))H_{2,1} \ldots H_{2,h}$$

$$\rightarrow_{\eta} \lambda t_1 \ldots t_r. H_{2,1}(H_{2,1} u)H_{2,2} \ldots H_{2,h} \rightarrow_{\eta} \lambda t_1 \ldots t_q. H_{2,1} u Q_{2,1} \ldots Q_{2,q} \rightarrow_{\eta} u,$$

where $\forall i \in \{1, \ldots, h\} \exists L_i \in \Lambda^0 [H_{1,i} = L_i(\lambda a. a z)$ and $H_{2,i} = L_i(\lambda a. a(a u))]$ and $\forall i \in \{1, \ldots, q\} \exists V_i \in \Lambda^0 [Q_{1,i} = V_i(\lambda a. a z)$ and $Q_{2,i} = V_i(\lambda a. a(a u))]$.

Hence, $\forall i \in \{1, \ldots, q\} [Q_{1,i} \subseteq Q_{2,i}].$ This implies $Q_{2,i} = t_i$. Hence, $H_{2,1} u = n u.$ Suppose $H_{2,1}^* u \notin \text{SOL}$, then $H_{2,1}^* z \notin \text{SOL}$. This is absurd. Hence, $H_{2,1}^* u \in \text{SOL}$. This implies $H_{2,1}^* = n \ast I$ and because $H_{2,1}^* \subseteq H_{2,1}$ we have also $H_{2,1} = n \ast I$.

Consider the standard reduction $\sigma_3 : D[x(\lambda a. a(a(P_k a y)))] \rightarrow_{\eta} y$. We have $D[x(\lambda a. a(a(P_k a y)))] \rightarrow_{\beta} \lambda t_1 \ldots t_r. (\lambda a. a(a(P_k a y)))H_{3,1} \ldots H_{3,h} \rightarrow_{\eta} \lambda t_1 \ldots t_r. H_{3,1}(H_{3,1}(P_k H_{3,1} y))H_{3,2} \ldots H_{3,h} \rightarrow_{\eta} \lambda t_1 \ldots t_q. (H_{3,1}(P_k H_{3,1} y))Q_{3,1} \ldots Q_{3,q}$.

As before we have $\forall i \in \{1, \ldots, q\} [Q_{3,i} \subseteq Q_{3,i}$ and $Q_{3,i} = t_i].$ Moreover, $I = n \ast H_{2,1}^* \subseteq H_{3,1}$, hence $H_{3,1} = n \ast I$. This implies $P_k y \in \text{SOL}$ and $P_k I = n \ast I \in \text{SOL}$. Hence $\{k\}(k) \downarrow$.

Thus, though both one side (left, right) invertibility problems are decidable in $\beta [4, 10]$, the problem of finding a common left inverse for a finite set of combinators is, in general, undecidable in $\beta$ as well as in $\beta \eta$. 

The existence of a common $\beta$ ($\beta\eta$)-right inverse for a finite set of combinators is undecidable as well [13] (4.2). We report the proof with a system $\mathcal{S}$ (inspired from [12, 4.6]) slightly different from that in [13].

### 4.2. Proposition (Statman [13]).

Let

$$\mathcal{S} = \{ xyU_2^2(xyU_2^2(P_k(xyU_2^2))(xyU_2^2)) = y, \ xyU_2^2(P_k(xyU_2^2))(xyU_2^2) = y \}, \{ x \}.$$

Then $\mathcal{S}$ is $\beta$ ($\beta\eta$)-solvable iff $\{ k \}(k)^\downarrow$.

**Proof.** ($\Rightarrow$) $D = \lambda t_1 t_2 t_3 \ I t_1$ is a $\beta$-solution for $\mathcal{S}$.

($\Rightarrow$) Let $D$ be a $\beta\eta$-solution for $\mathcal{S}$. Then $DyU_2^2 = \eta y$. If $DyU_2^2 = \eta y$ then $DyU_2^2 (P_k(DyU_2^2)) = \eta y (DyU_2^2) = \eta y$, which is impossible. Hence, $DQyU_2^2 = \eta y$. Then $DQyU_2^2 = \eta I$. We have $I(I(P_k I)(DyU_2^2)) = \eta P_k I(DyU_2^2) = \eta y$. Hence, $P_k I \in \text{SOL}$. This implies $\{ k \}(k)^\downarrow$. □

The $\beta$-solvability problem for systems like those in 0.5–0.7 is, in general, undecidable (4.3.1 and 4.1).

### 4.3. Proposition. The following systems are $\beta$-solvable iff $\{ k \}(k)^\downarrow$:

0. $\mathcal{S} = \{ x y(\lambda a b. P_k(xy(U_2^2 I)f)a)(xy(\lambda u v. z)f) = yz, \{ x \} \}$.

1. $\mathcal{S} = \{ x y(\lambda a b. P_k(xy(U_2^2 I)f)a)(xy(\lambda u v. z)f) = y(x y)z, \{ x \} \}$.

**Proof.** 0. ($\Rightarrow$) $D = \lambda t_1 t_2 t_3. t_2(t_1 t_3)\Omega$ is a $\beta$-solution for $\mathcal{S}$.

($\Rightarrow$) Let $D = \lambda t_1 t_2 t_3. t_2(D_1 t_1 t_3) \ldots (D_4 t_2 t_3)$ a $\beta$-solution for $\mathcal{S}$. We have $(Dy(\lambda u v. z)f) = z(D_3 y(\lambda u v. z)f) \ldots (D_q y(\lambda u v. z)f)$, this implies $q = 2$. Hence

$$Dy(\lambda a b. P_k(Dy(U_2^2 I)f)a)(Dy(\lambda u v. z)f) = P_k(U_2^2 I(D_1 \ldots))(D_2 \ldots)(D_1 \ldots) = P_k I(D_1 \ldots) = yz.$$

This implies $P_k I \in \text{SOL}$ and $\{ k \}(k)^\downarrow$.

1. ($\Rightarrow$) Let $G = \lambda u t_1 t_2 t_3. t_2(t_1(t_2 \Omega \mu t_1 t_3)\Omega$ and $D = GG$. $D$ is a $\beta$-solution for $\mathcal{S}$.

($\Rightarrow$) Let $D$ be a $\beta$-solution for $\mathcal{S}$, then $D' = \lambda y. D(\lambda a. y)$ is a $\beta$-solution for $\mathcal{S}$ (in 0). □

### 5. Regular SL-systems

Though $\beta$-solvability for SL-systems is undecidable (4.1) we show (5.2) that there is an interesting class of SL-systems (regular SL-systems) for which the $\beta$-solvability problem is decidable in Polynomial Time.

The following definition comes from [8, 6.7].
5.0. Definition. An SL-system \( S = (F, X) \) is said to be quasi-regular iff there exists a function \( e : X \rightarrow \mathbb{N} \) s.t. (see [1, 10.1.7, 13, 10.2.18] for Seq and \( M_{x} \)):

0. for all \( x \in X \) \( 1 \leq e(x) \leq \min\{m | xM_{1} ... M_{m} = z \text{ is in } F} \};

1. for all equations \( xM_{1} ... M_{m} = z \) in \( F[\alpha_{e}(x)] \), is solvable;

2. for all equations \( M = z \in F \) for all \( \alpha \in \text{Seq} \) if \( [\alpha \neq < >) \) and \( M_{x} = \alpha_{a_{1}} ... a_{k}. xM_{1} ... M_{m} \text{ and } x \in X \) then \( m < e(x) \).

In [8, 6.7] it is shown that for a particular class of quasi-regular SL-systems (namely those that yield an \( X \)-separability problem (3.1)) the \( \beta \)-solvability problem is decidable. On the other hand Proposition 4.1 shows that the \( \beta \)-solvability problem for quasi-regular SL-systems is undecidable. Here we isolate the very reason of undecidability for quasi-regular SL-systems, namely: a shortage of abstractions on the LHS occurrences of the RHS variables. More precisely, we prove that given a quasi-regular SL-system \( S \) it can be transformed in Polynomial Time into a quasi-regular SL-system \( S' \) having a decidable \( \beta \)-solvability problem (however, in general, \( S' \) will not yield an \( X \)-separability problem). This transformation (called \textit{relaxation}) consists of adding abstractions to the LHS occurrences of the RHS variables of \( S \).

5.1. Definition. We define relaxation for SL-systems. Let \( M = z, M' = z \) be formulas.

(i) \( \text{relax}(M = z, M' = z) \) is true iff \( M' \) is obtained from \( M \) replacing one occurrence of \( z \) in \( M \) with \( \lambda a_{1} ... a_{k}. z \) (where \( k > 0 \)). In this situation we also say that \( M' = z \) is a relaxation of \( M = z \).

(ii) \( \text{relax}^{*} \) is the reflexive and transitive closure of \( \text{relax} \).

(iii) Let \( (F, X), (F', X) \) be SL-systems. Then \( \text{relax}_{\text{SL}}((F, X), (F', X)) \) is true iff there exists a bijection \( \varphi \) from \( F \) to \( F' \) s.t. for each equation \( M = z \) in \( F \) we have \( \text{relax}^{*}(M = z, \varphi(M = z)) \). In this situation we also say that \( (F, X) \) is a relaxation of \( (F, X) \).

The following theorem (5.2) is our main result. It is the core of our system solving algorithm.

Theorems 5.2.0, 5.2.2 give a Polynomial Time necessary condition \( \text{OK}_\text{NEC} \) of \( \beta \)-solvability for SL-systems.

Theorems 5.2.1, 5.2.3 give a Polynomial Time sufficient condition \( \text{OK}_\text{SUFF} \) of \( \beta \)-solvability for SL-systems and a Polynomial Time algorithm to construct a \( \beta \)-solution (if any) to an SL-system satisfying \( \text{OK}_\text{SUFF} \).

Since, by 4.1, \( \beta \)-solvability is undecidable \( \text{OK}_\text{NEC} \) and \( \text{OK}_\text{SUFF} \) cannot be equal. However for each SL-system \( S \) s.t. \( \text{regular}_{\text{SL}}(S) = \text{OK}_\text{NEC}(S) \) then \( \text{OK}_\text{SUFF}(S) \) else true] is true the \( \beta \)-solvability problem is decidable. Again by 4.1 the function \( \text{regular}_{\text{SL}} \) cannot be identically true. In some sense the goodness of our result is measured by how often regular\_SL is true.

Theorem 5.2.4 says that regular\_SL takes value true often enough to isolate the very reason of undecidability for quasi-regular SL-systems, namely a shortage of abstractions on the LHS occurrences of the RHS variables. Moreover, we are able to compute in Polynomial Time an upper bound for such shortage, i.e. given any...
quasi-regular SL-system $\mathcal{S}$ we can always find in Polynomial Time a relaxation $\mathcal{S}'$ of $\mathcal{S}$ s.t. $\text{regular}_{-}\text{SL}(\mathcal{S}') = \text{true}$.

5.2. **Theorem.** There are functions $\text{OK}_{-}\text{NEC}$, $\text{OK}_{-}\text{SUFF}$ from SL-systems to Boole s.t.:

0. For all SL-systems $\mathcal{S}$ $\text{OK}_{-}\text{NEC}(\mathcal{S})$ is computable in time polynomial in $\text{Size}(\mathcal{S})$.
1. For all SL-systems $\mathcal{S}$ $\text{OK}_{-}\text{SUFF}(\mathcal{S})$ is computable in time polynomial in $\text{Size}(\mathcal{S})$.
2. For all SL-systems $\mathcal{S}$ if $\mathcal{S}$ is $\beta$-solvable then $\text{OK}_{-}\text{NEC}(\mathcal{S}) = \text{true}$.
3. For all SL-systems $\mathcal{S}$ if $\text{OK}_{-}\text{SUFF}(\mathcal{S}) = \text{true}$ then $\mathcal{S}$ is $\beta$-solvable and we can construct a $\beta$-solution for $\mathcal{S}$ in time polynomial in $\text{Size}(\mathcal{S})$.
4. Let $\text{regular}_{-}\text{SL}$ be the Polynomial Time function from SL-systems to Boole defined as follows: $\text{regular}_{-}\text{SL}(\mathcal{S}) = \text{if } \text{OK}_{-}\text{NEC}(\mathcal{S}) \text{ then } \text{OK}_{-}\text{SUFF}(\mathcal{S}) \text{ else } \text{true}$. Then:

For any quasi-regular SL-system $\mathcal{S}$ there exists an SL-system $\mathcal{S}'$ s.t. $\mathcal{S}'$ is a relaxation of $\mathcal{S}$ and $\text{regular}_{-}\text{SL}(\mathcal{S}') = \text{true}$ and $\mathcal{S}'$ can be computed from $\mathcal{S}$ in Polynomial Time.

**Proof.** See Appendix A. □

5.3. **Definition.** (a) From now on $\text{regular}_{-}\text{SL}$, $\text{OK}_{-}\text{SUFF}$ and $\text{OK}_{-}\text{NEC}$ are the functions defined in (the proof of) 5.2. However, the reader not interested in the technical details can read the following without looking at such definitions. In this case the examples in 5.5 should be read as corollaries of Theorem 5.2.

(b) An SL-system $\mathcal{S}$ is said to be regular iff $\text{regular}_{-}\text{SL}(\mathcal{S}) = \text{true}$.

5.4. **Remark.** If $\text{regular}_{-}\text{SL}(\mathcal{S}) = \text{true}$ then the $\beta$-solvability problem for $\mathcal{S}$ is decidable. In fact, by 5.2, we have: if $\text{regular}_{-}\text{SL}(\mathcal{S})$ then $[\mathcal{S}$ is $\beta$-solvable iff $\text{OK}_{-}\text{NEC}(\mathcal{S}) = \text{true}]$.

5.5. **Example.** (i) Let $\mathcal{S} = \{xy\Omega(\lambda a.x\Omega) = y, x\Omega(\lambda ab.z) = z\}, \{x\}$. $\mathcal{S}$ is regular and $\text{OK}_{-}\text{NEC}(\mathcal{S}) = \text{true}$. By 5.2 $\mathcal{S}$ is $\beta$-solvable. A $\beta$-solution is

$$D = \lambda y t_1 t_2 \cdot t_2 t_1 (U_1^3 y).$$

(ii) The system $\mathcal{S} = \{xx(x(\lambda a.z)) = z, xx(x(xz)) = z, xv(\lambda ab.z) = z\}, \{x\}$ is regular and $\beta$-solvable.

(iii) The systems in Examples 0.5–0.7, 3.2 are regular.

(iv) $\mathcal{Q} = \{x(\lambda ab.z) = z, x(\lambda ab.ay)\Omega = y\}, \{x\}$ is regular.

(v) $\mathcal{S} = \{x(\lambda a.a z) = z, x(\lambda a.a(au)) = u\}, \{x\}$ is not regular (look at $\mathcal{S}$ in 4.1).

(vi) A separability problem (3.1) [1, 10.4.4; 9] is a regular SL-system.

(vii) All the systems studied in [7, 8] are regular SL-systems. However, there are regular SL-systems that are not in any of the classes studied in [6–9] or [12], e.g. those in 0.5–0.7, $\mathcal{Q}$ above, $\mathcal{S}$ in 6.4. On the other hand, not all the systems studied in [6] or [12] are regular SL-systems. As a matter of fact we can show that the $\beta$-solvability problem for the systems studied in [12] is NP-complete [17, 2.6].
Outside the class of regular SL-systems the $\beta$-solvability problem, though decidable, can be harder.

5.6. Proposition. There is a class of SL-systems for which the $\beta$-solvability problem is NP-complete.

Proof. See Appendix B. $\square$

6. Applications

The interest of regular SL-systems rests on the fact that they can be used to build a compiler for an equational programming language in which constraints on the executable codes to be generated by the compiler can be specified by the user (e.g. as in 0.5–0.7, 6.4) (properties of) data structures can be described in an abstract way (e.g. as in 0.5–0.7, 6.4), $\lambda$-terms representing programs have normal form (6.2.2) and inverse functions of constructors (of a data structure) run in constant time (e.g. as in 0.6, 6.4). In this section we build (6.2) such compiler.

The system $\mathcal{S} = (\{xa = a, xb = b\}, \{x\})$ is $\beta$-solvable (with $D[\ ] = (\lambda x. [\ ])I$), however, from a solution for $\mathcal{S}$ we cannot get a solution for $\mathcal{S}' = (\{xa = Ha, xb = Lb\}, \{x\})$, where $H$ and $L$ are arbitrarily given combinators. This is because $\{x, y\}$ is not $\beta$-separable (3.1). In other words, $a$ and $b$ are parameters which value cannot be assigned from inside $\lambda$-calculus. Assignable variables (6.0) are assignable from inside $\lambda$-calculus.

6.0. Definition. Let $\mathcal{S} = (\Gamma, \{x_1 \ldots x_n\})$ be a system and $\vec{y} \in (\text{Var} - \{x_1 \ldots x_n\})$. The sequence $\vec{y}$ is said to be assignable in $\mathcal{S}$ ($\vec{y}$ is asg in $\mathcal{S}$) iff $\forall M = N \in \mathcal{S} \exists Q \in A$ s.t. $M = (\lambda x_1 \ldots x_n. Q)(x_1 \vec{y}) \ldots (x_n \vec{y})$, where $\vec{y} \notin \text{FV}(Q)$.

6.1. Example. Let $\vec{y}$ be as in 3.1 and $\mathcal{S}$ be a separability or an $X$-separability problem (see 3.1). Then $\vec{y}$ is asg in $\mathcal{S}$.

The hypotheses of 6.2 define our equational programming language (note that $\mathcal{S}^*$ in 6.2 is an SL-system) and the algorithm in the proof of 6.2 defines a compiler for our language.

6.2. Theorem. Let $\mathcal{S} = (\Gamma_1 \cup \Gamma_2, \{x_1 \ldots x_n\})$ be a system and $\vec{y} \in (\text{Var} - \{x_1 \ldots x_n\})$ s.t.:

H0. $\vec{y}$ is assignable in $\mathcal{S}$ (see 6.0).

H1. Each equation in $\Gamma_1$ has form $xM = y(x_1 \vec{y}) \ldots (x_n \vec{y})\vec{z}$, where $x \in \{x_1 \ldots x_n\}$, $y \in \{\vec{y}\}$, $\vec{z} \notin \{x_1 \ldots x_n, \vec{y}\}$ and the variables in $\{y \mid xM = y(x_1 \vec{y}) \ldots (x_n \vec{y})\vec{z} \in \Gamma_1\}$ are pairwise distinct.
H2. Each equation in $\Gamma_2$ has form $x \bar{M} = z$, where $x \in \{x_1 \ldots x_n\}$ and $z \notin \{x_1 \ldots x_n, \bar{y}\}$.

H3. $\mathcal{S}^* = (\{x \bar{M} = y | x \bar{M} = y(x_1 \bar{y}) \ldots (x_n \bar{y}) \bar{z} \in \Gamma_1\} \cup \{u_{yz} \bar{M} = z | x \bar{M} = y(x_1 \bar{y}) \ldots (x_n \bar{y}) \bar{z} \in \Gamma_1\}$ and $z \in \{\bar{z}\}$ and $u_{yz} \in \{\bar{u}\} \cup \Gamma_2$, $\{x_1 \ldots x_n, \bar{u}\}$) is regular, where $\bar{u}$ is a sequence of fresh variables s.t. $\{\bar{u}\} = \{u_{yz} | x \bar{M} = y(x_1 \bar{y}) \ldots (x_n \bar{y}) \bar{z} \in \Gamma_1$ and $z \in \{\bar{z}\}$.

Then:

0. $\mathcal{S}$ is $\beta$-solvable iff OK$_{\text{NEC}}(\mathcal{S}^*) = \text{true}$. Thus $\beta$-solvability is decidable in Polynomial Time.

1. If $\mathcal{S}$ is $\beta$-solvable then a $\beta$-solution for $\mathcal{S}$ can be constructed in Polynomial Time.

2. If $\mathcal{S}$ is $\beta$-solvable and Card($\Gamma_1 \cup \Gamma_2$) > 1 then $\mathcal{S}$ has a $\beta$-solution having normal form.

**Proof.** See Appendix C. □

6.3. Example. (a) Let $\mathcal{S} = (\{xy\Omega(\lambda a. xyv) = y(xy)v, xy\Omega(\lambda ab. z) = z\}, \{x\})$. Since OK$_{\text{NEC}}(\mathcal{S}^*) = \text{true}$ by 6.2 $\mathcal{S}$ is $\beta$-solvable. A $\beta$-solution is

$$Q \equiv \lambda yab.((Hyab)[y := \lambda xuab. yx])$$

where

$$H \equiv \lambda yab.((Lyab)[y := \lambda xuab. yxyab(bQU^2)])$$

$$L \equiv FF = \lambda yt_1t_2. t_2t_1(U^2_1(y(t_2\Omega(U^2_1I)FF)y)t_1t_2),$$

$$F \equiv \lambda uyt_1t_2. ((Gyt_1t_2)[y := \lambda c. y(t_2\Omega(U^2_1I)uuy)y]),$$

$$G \equiv \lambda yt_1t_2. t_2t_1(U^2_1(yyt_1t_2)).$$

(b) Let $\mathcal{S}'' = (\{x_{y\Omega} = y_0(x_{y\Omega})Q. x_{y\Omega}(sz) = y_1(x_{y\Omega})z, x_{y\Omega}(\lambda ab. z) = z\}, \{x\})$ (0 and $s$ as in 0.5). By 6.2 $\mathcal{S}''$ is $\beta$-solvable. A $\beta$-solution is $D$ as in 0.5.

6.4. Example. It is well known that many interesting data structures can be defined using (heterogeneous) term algebras (e.g. [3]).

Let $\mathcal{A}_1 = \langle A_{j,1}, \ldots, A_{j,n(j)} \rangle$, $\{g_{j,i} : A_{j,k(j,1)} \times \ldots \times A_{j,k(j,a(j,0))} \rightarrow A_{j,b(j,0)} | i = 1, \ldots, m(j)\} >$ be term algebras ($j = 1, 2$).

A partial recursive function $f$ from $\mathcal{A}_1$ to $\mathcal{A}_2$ can be represented in the $\lambda$-calculus solving the system $\mathcal{S} = (\Gamma_1 \cup \Gamma_2 \cup \Gamma_3, \{f, \bar{g}, \bar{p}, \bar{d}\})$, where:

$$\Gamma_1 = \{p_{j,1,h} \bar{y}((g_{j,i} \bar{y}z_1 \ldots z_{k(j,a(j,0))}) = z_k | i = 1, \ldots, m(j) \}$$

and $j = 1, 2$ (left-invertibility of the constructors);

$$\Gamma_2 = \{d_{j,1} \bar{y}((g_{j,i} \bar{y}z_1 \ldots z_{k(j,a(j,0))}) = v_{0,j,1} | j = 1, 2 \}$$

and $i = 1, \ldots, m(j)$ (recognizability of the constructors);
The equations in $r_1 \cup r_2$ describe the data structures and the equations in $r_3$ describe the function $f$ (a set of functions if we consider types). By 6.2 $\mathcal{S}$ is $\beta$-solvable. A solution $H[ \ ] = (\lambda f \bar{g} \bar{p} \bar{d}. \ [ \ ] ) F \bar{G} \bar{P} \bar{D}$ for $\mathcal{S}$ yields the wanted representations for program and data structures ($F \bar{Y}$ for the program for the function $f$, etc). Any partial recursive function can be specified (and hence represented) replacing $y_{1,1}, \ldots, y_{1,m(1)}$ with suitable combinators (see 0.2–0.4). Of course it is possible to add equations to $\mathcal{S}$. In this way it is possible to choose a representation with some particular property (e.g. as we did in 0.5–0.7) or to synthesize more than one program at the same time, etc. Consider $\mathcal{S}' = (r_1 \cup r_2 \cup r_3 \cup \{ f \bar{y}(\lambda a b. z) = z \}, \{ f, \bar{g}, \bar{p}, \bar{d} \})$. By 6.2 the system $\mathcal{S}'$ is $\beta$-solvable. This means that any partial recursive function from $2^\mathbb{N}$ to $2^\mathbb{N}$ can be represented with a $\lambda$-term $F$ having from $\lambda t. tG1G2$, with $G1, G2 \lambda$-terms. The system $\mathcal{S} (\mathcal{S}')$ cannot be transformed into an $X$-separability problem (3.1) because of the presence of the equations in $r_1 \cup r_2 \cup \{ f \bar{y}(\lambda a b. z) = z \}$ and, as in 0.6, cannot be solved with the methods in [3].

7. Conclusions

Though the $\beta$-solvability problem for SL-systems (3.0) is undecidable (4.1) there is an interesting class of SL-systems (regular SL-systems) definable in Polynomial Time and for which the $\beta$-solvability problem is decidable in Polynomial Time (5.2).

Regular SL-systems yield (6.2) an equational programming language in which:

- a moderate amount of self-application is allowed (e.g. as in 0.7);
- constraints on executable code to be generated by the compiler can be specified by the user (e.g. as in 0.5–0.7, 6.4);
- (properties of) data structures can be described in an abstract way (e.g. as in 6.4);
- $\lambda$-terms representing programs have normal form (6.2.1);
- left-inverse functions of constructors (of a data structure) run in constant time (e.g. as in 0.6, 6.4).

To widen the language introduced seems to be next step.

Acknowledgements

I am grateful to Rick Statman for the many in-depth discussions we had on the topics of this paper and for his comments and suggestions. Without his questions the complexity analysis of the algorithms would not be here.
I am also grateful to Mariangiola Dezani, Henk Barendregt, Corrado Böhm, Peter B. Andrews, Michael Albert and Robert Harper for their questions, comments and suggestions about preliminary versions of this paper.

Appendix A. Proof of 5.2

This proof is quite long so we divide it into many parts. In A.0 we give some useful basic definitions. In A.1 we define OK-NEC and prove 5.2.0. In A.2. we define OK-SUFF and prove 5.2.1. In A.3 we prove 5.2.2. In A.4 we prove 5.2.3. In A.5 we prove 5.2.4.

A.0. More on the $\lambda$-calculus

A theory $T$ is called a $\lambda$-theory if $T$ is consistent and $T = \lambda + T [1, 2.1.30, 4.1.1]$, e.g. $\lambda$ and $\lambda\eta$ (i.e. $\beta$ and $\beta\eta$) are $\lambda$-theories. We write $T$ is sms for $T$ is semisensible $[1, 84, 4.1.7]$ (e.g. $\beta$ and $\beta\eta$ are sms theories). Conventions: $A \subseteq B$ stands for $A \subset B$ and $A$ is finite, $\max 0 = 0$, $P(A) (P(A))$ is the set of (finite) subsets of the set $A$, $\forall i \in \mathbb{N}$ $\Omega_i = \Omega$, if $D[ ]$ is a context and $Q \subseteq A[ ]$ we write $\exists D[ ] (\equiv (\lambda x_1 \ldots x_n[ ]) D_1 \ldots D_n \in Q \text{ s.t. } A)$ for $\exists D[ ] (\equiv (\lambda x_1 \ldots x_n[ ]) D_1 \ldots D_n \in Q \text{ s.t. } A)$.

Let $f, g : \mathbb{N} \rightarrow \mathbb{N}$. We write $f(n) = O(g(n))$ iff $\exists k, m \in \mathbb{N} \forall n \geq m \ [f(n) \leq kg(n)]$.

A.0.0. Definition. Let $M, N \in A$, $\bar{\rho} \subseteq_f A$ and $\alpha \in \text{Seq}$ (see [1, p. xiii]).
- $\bar{\rho}_\alpha = \{M \mid M \in \bar{\rho}\}$ (see [1, 10.1.7–13, 10.2.18]).
- We write $\alpha \vdash_\bar{\rho}(BT(M))$ iff $\forall \beta < \alpha[\beta \in BT(M) \Rightarrow M_\beta \in \text{SOL}]$.
- We write $\alpha \vdash_\bar{\rho}(BT(\bar{\rho}))$ iff $\forall Q \in \bar{\rho}[\alpha \vdash_\bar{\rho}(Q)]$.
- We write $M \mid \alpha\downarrow$ iff $[\alpha \vdash_\bar{\rho}(BT(M))$ and $M_\alpha \in \text{SOL}]$, $M \mid \alpha\uparrow$ otherwise.
- We write $\bar{\rho} \mid \alpha\downarrow$ iff $\forall Q \in \bar{\rho}[\alpha \downarrow]$.
- Let $Q \in A$. We define: $M[\langle \rangle := Q] \equiv QM$,

$$M[\langle \rangle := Q]$$

\[\equiv \text{if } M = \lambda x_1 \ldots x_n. y M_0 \ldots M_{m-1} \text{ then if } j < m \text{ then } \lambda x_1 \ldots x_n. y M_0 \ldots (M_j[\alpha := Q]) \ldots M_{m-1} \text{ else } \lambda x_1 \ldots x_n t_m \ldots t_j.y M_0 \ldots M_{m-1} t_m \ldots (t_j[\alpha := Q]) \text{ else } QM.\]

- We define the functions $\text{deg}$ (degree), $\text{ord}$ (order), $\text{head}$ (head) as follows:
  - if $M = \lambda x_1 \ldots x_n. y M_0 \ldots M_{m-1}$ then $\text{deg}(M) = m$, $\text{ord}(M) = n$, $\text{head}(M) = y$;
  - if $M \notin \text{SOL}$ then $\text{deg}(M) = \text{ord}(M) = 0$ and $\text{head}(M) \uparrow$.
- We write $M \subseteq N$ for $\text{BT}(M) \subseteq \text{BT}(N)$ (see [1, 10.2.3]).
- Let $C[ ], D[ ] \in A[ ]$. We write $C[ ] \subseteq D[ ]$ for $C[z] \subseteq D[z]$ ($z$ fresh).
\( M \sim_a M \) iff \([M | x \uparrow] \) or \([M | x \downarrow] \) and \( \deg(M_x) = \deg(N_x) - \ord(M_x) = \ord(N_x) \) and \( \text{head}(M_x) = \text{head}(N_x) \) (see \([1, 10.2.19-21]\)).

- The node \( x \) is said to be useful for \( \tilde{F} \) (usf for \( \tilde{F} \)) iff \([\tilde{F} | x \downarrow] \) and \([\text{Card} (\tilde{F}) = 1 \) or \( \exists M, N \in \tilde{F} \) \( M \sim_a N ] \) (see \([1, 10.4.6]\).

- We say that \( \tilde{F} \) agrees up to \( x \) (ag for \( \tilde{F} \)) iff \([\forall M, N \in \tilde{F} \forall \beta < x M \sim_a N \).

- The node \( x \) is said to be \( \tilde{F} \)-adherent (adh for \( \tilde{F} \)) iff \([\exists M \in \tilde{F} x \in \text{BT}(M) \) (see \([7, 8]\)).

- The binary relation \( \text{ind}(\tilde{F}) \) is defined as follows:
  \[ \forall P, Q \in \tilde{F} [\text{ind}(\tilde{F}, P, Q) \iff \exists M, N \in \tilde{F} \forall x \in \text{Seq} [\exists x \text{usf and agt for } \tilde{F} \text{ and Card}(\tilde{F}) > 1] \Rightarrow [P \gamma Q \text{ and ind}([M \sim_a M], P, Q)])] \]

- Let \( \tilde{x} = x_1, \ldots, x_n \in \text{Var} \) and \( \tilde{y} \in \text{Var} \). \( (\tilde{x}, \tilde{y}) \) is the sequence \( (x_1, \tilde{y}), \ldots, (x_n, \tilde{y}) \).

**A.0.1. Remark** (Böhm and Tronci \([7]\)). Let \( \tilde{F} \subset A \) and \( \alpha \in \text{Seq} \). If \( \alpha \) is usf and agt for \( \tilde{F} \) then \( \alpha \) is usf, agt and adh for \( \tilde{F} \).

**A.0.2. Definition.** Let \( T \) be a theory and \( \mathcal{S} = [\Gamma, \{x_1, \ldots, x_n\}] \) be a system.

(i) Let \( D[\ ] = (\lambda x_1 \ldots x_n [\ ]D_1 \ldots D_n \in A[\ ] \) s.t. \( \forall i \in \{1, \ldots, n\} FV(D_i) \subseteq \{\tilde{a}\} \).

We say that \( D[\ ] \) is a family of \( T \)-solutions for \( \mathcal{S} \) if the variables in \( \tilde{a} \) do not occur in \( \mathcal{S} \) and \( \forall M = N \in \mathcal{S} D[M] = \Gamma D[N] \). By abuse of language we also say that \( D[\ ] \) is a \( T \)-solution for \( \mathcal{S} \). Of course for all substitutions \( \sigma = [b := H_b | b \in \{\tilde{a}\} \) and \( H_b \in \Lambda^0] \)

\( D^\sigma[\ ] = (\lambda x_1 \ldots x_n [\ ])D^\sigma_1 \ldots D^\sigma_n \) is a \( T \)-solution for \( \mathcal{S} \).

(ii) Let \( \mathcal{S} = (\Gamma, X) \) be a system. We define \( \text{left}(\mathcal{S}) = \text{left}(\Gamma) = \{M | \exists M = N \in \mathcal{S}\} \).

(iii) Let \( \mathcal{S} = (\Gamma, X) \) be a system. We define: \( \text{right}(\mathcal{S}) = \text{right}(\Gamma) = \{N | \exists M = N \in \mathcal{S}\} \).

(iv) Let \( \mathcal{R} \) be an equivalence relation on \( A \). Then we extend \( \mathcal{R} \) to \( \mathcal{R}^\text{left} \) on Form(\( A \)) as follows: \( (M = N) \mathcal{R}^\text{left}(P = Q) \) iff \( MP \mathcal{R} P \). Let \( \mathcal{S} = (\Gamma, X) \) be a system. We write \( \Gamma/\mathcal{R} \) for \( \Gamma/\mathcal{R}^\text{left} \) and \( \mathcal{S}/\mathcal{R} = \{(A, X) | A \in \Gamma/\mathcal{R}\} \).

(v) We define \( \text{Card}(\mathcal{S}) = \text{Card}(\Gamma) \).

**A.1. Definition of OK Nec and Proof of 5.2.0**

We define (A.1.0) the predicate OK-Nec. OK-Nec gives a necessary condition of \( \beta \)-solvability for SL-systems. The objects used in A.1.0 (i.e. canonical, LR-distinct, PFR) are defined, respectively, in A.1.1, A.1.4–A.1.5, A.1.7.

**A.1.0. Definition.** Let \( \mathcal{S} = (\Gamma, X) \) be an SL-system. \( \text{OK-Nec}(S) = \text{true} \) iff \( \exists \mathcal{S}^+ \in \text{canonical}(\mathcal{S}) [\mathcal{S}^+ \text{ is PFR and LR-distinct}] \).

An SL-system is said to be canonical (A.1.1) iff each RHS variable occurs on the LHS and the RHS variables are pairwise distinct and there is no garbage in the LHS terms.
A.1.1. Definition. Let $\mathcal{S} = (\Gamma, X)$ be an SL-system (3.0).

(i) $\mathcal{S}$ is said to be canonical iff the following conditions are satisfied ($M$ has form $x\tilde{M}$):

(a) $\forall M = z \in \mathcal{S} \exists x \in BT(M) \ [\text{head}(M_z) \equiv z]$. (The variable $z$ occurs in $M$.)
(b) The variables in right($\mathcal{S}$) are pairwise distinct.
(c) $\forall M = z \in \mathcal{S} \left[\text{FV}(M) \subseteq (X \cup \{z\})\right]$. (The free variables in $M$ are in $X \cup \{z\}$.)

(ii) A canonical version of $\mathcal{S}$ is a system $\mathcal{S} = (\Pi, X)$ s.t. $\mathcal{S}$ is canonical and $\Pi$ is obtained from $\Gamma$ replacing each $M = z$ in $\Gamma$ with $M' = z'$ where $M' \equiv M[\tilde{v} := \Omega \mid \tilde{v} \in (\text{FV}(M) - (X \cup \{z\}))]$ and $\ast = [z := u]$ with $u$ fresh variable.

(iii) We write $\mathcal{S} \in \text{canonical}(\mathcal{S})$ iff $\mathcal{S}$ is a canonical version of $\mathcal{S}$.

A.1.2. Remark. Let $\mathcal{S} = (\Gamma, X)$ be an SL-system and $\mathcal{T}$ be a sms theory (e.g. $\beta$ or $\beta\eta$). Then $\mathcal{S}$ is $\mathcal{T}$-solvable iff there exists $\mathcal{S} \in \text{canonical}(\mathcal{S})$ s.t. $\mathcal{S}$ is $\mathcal{T}$-solvable. Moreover, up to redenomination of the names of the free variables, there is at most one canonical version of $\mathcal{S}$ and it can be computed in time polynomial in $\text{Size}(\mathcal{S})$ (see 2.4). Hence, it is not restrictive to consider only canonical SL-systems.

A.1.3. Example. Let $\mathcal{P}$ be as in 5.5. Then $\mathcal{P}$ is $\beta$-solvable iff $\mathcal{P}^+ = (\{xx(x(\lambda a. u)) = u, xx(x(xy)) = y, x\Omega(\lambda b. z) = z\}, \{x\})$ is $\beta$-solvable. $\mathcal{P}^+$ is a canonical version of $\mathcal{P}$.

Essentially the $\beta$-solvability of an SL-system depends on two factors: how severe are the constraints on the executable codes to be generated by the compiler; how much IHS terms and initial parts (proper or improper) of IHS terms differ one from another. Unfortunately such notions are not so neatly distinct. Roughly speaking formal definitions for such notions are, respectively, in A.1.4-A.1.5, A.1.7.

A.1.4. Notation. Let $X \subset_f \text{Var}$. We define $Q(X) = \{e : X \times \{0, 1, 2, 3\} \rightarrow \mathbb{N} \mid \forall x \in X \ [1 \leq e(x, 0) \leq e(x, 1) \text{ and } e(x, 2) \geq 1] \text{ and } \forall x, x' \in X \ [e(x, 2) - e(x', 2) = e(x, 1) - e(x, 0) + e(x', 0) - e(x', 1) \Rightarrow x \equiv x']\}$.

Roughly speaking a node $x$ is $(Z, X, e, f)$-safe (A.1.5.0) iff its order is not too small and the path leading to $x$ does not involve too much self-application. A finite set of $\lambda$-terms $\mathcal{F}$ is $(Z, X, e, f)$-distinct (A.1.5.1) [Z-distinct (A.1.5.2)] iff we can distinguish its terms using only $(Z, X, e, f)$-safe nodes [without any restriction on the self-application].

A.1.5. Definition. Let $X, Z \subset_f \text{Var}, \mathcal{F} \subset_f \Lambda, e \in Q(X)$ and $f(X, e) : P_{\omega}(\Lambda) \times \text{Seq} \rightarrow \mathbb{N}$.

0. A node $x \in \text{Seq}$ is said to be $(Z, X, e, f)$-safe in $\mathcal{F}$ iff it satisfies the following conditions:

a. $\forall M, N \in \mathcal{F} \left[\text{head}(M_x) \equiv x \in X \text{ and } N_x = \lambda \tilde{a}. a_i \tilde{Q} \text{ and } a_i \in \{\tilde{a}\}\right] \Rightarrow [e(x, 2) \not\equiv e(x, 1) - e(x, 0) + \text{deg}(N_x) - \text{ord}(N_x) + i]$

b. $\forall \beta \leq \alpha \forall M \in \mathcal{F} \left[\text{head}(M_\beta) \in X \Rightarrow \text{deg}(M_\beta) < e(\text{head}(M_\beta), 0)\right]$

c. $\forall M \in \mathcal{F} \left[\text{head}(M_x) \in Z \Rightarrow [\text{ord}(M_x) = f(X, e, \mathcal{F}, x) \text{ and } \text{deg}(M_x) = 0]\right]$. 

1. We say that \( \mathcal{H} \) is \((Z, X, e, f)\)-distinct iff the following conditions are satisfied:
   a. If \( \text{Card}(\mathcal{H}) = 1 \) then \[ Z \neq \emptyset \Rightarrow \exists x \text{ usf and agt for } x \text{ is } (Z, X, e, f)\)-safe in \( \mathcal{H} \) and \( \forall M \in \mathcal{H} \left[ \text{head}(M_\varnothing) \in Z \right] \);
   b. If \( \text{Card}(\mathcal{H}) > 1 \) then \( \exists \mathcal{H} \) usf and agt for \( \mathcal{H} \) s.t. \( x \) is \((Z, X, e, f)\)-safe in \( \mathcal{H} \) and \( \forall S \in \mathcal{H} \) \( x \) is \((Z, X, e, f)\)-distinct.

2. We say that \( \mathcal{H} \) is \( Z \)-distinct iff \( \mathcal{H} \) is \((Z, X, e, f)\)-distinct.

3. The SL-system \( \mathcal{S} = (\Gamma, X) \) is said to be LR-distinct iff \( \text{left}(\mathcal{S}) \) is \( \text{right}(\mathcal{S}) \)-distinct (note that \( \text{right}(\mathcal{S}) \subset \text{Var} \)).

A.1.6. Remark. Note that the \( \emptyset \)-distinction (A.1.5.2) is the distinction introduced in [9] (also in [1, 10.4.7]).

If an SL-system is \( \beta \)-solvable then any initial part (proper or improper) of an LHS term can be distinguished from any LHS term (A.1.7.1). Definition A.1.7.1 stems from [8, 4.1.4].

A.1.7. Definition. Let \( \mathcal{S} = (\Gamma, X) \) be an SL-system. We define:

0. prefix(\( \mathcal{S} \)) = \( [xM_1 \ldots M_m \mid xM_1 \ldots M_m = z \in \mathcal{S}] \cup \{ xM_1 \ldots M_m \mid xM_1 \ldots M_{m+k} = z \in \mathcal{S} \) and \( k > 0 \} \).

1. \( \mathcal{S} \) is said to be PFR (\( \mathcal{S} \) satisfies the prefix rule) iff prefix(\( \mathcal{S} \)) is distinct (A.1.6).

The definition of OK-NEC is now complete (see A.1.0, A.1.1, A.1.5.3, A.1.7). We can now check that OK-NEC(\( \mathcal{S} \)) can be computed in Polynomial Time (A.1.8).

A.1.8. Proof of 5.2.0. The thesis follows from A.1.2 and the following facts. Let \( \mathcal{H} \subset \Gamma \) and \( Z \subset \text{Var} \) s.t. \( \text{Card}(Z) \leq \text{Size}(\mathcal{H}) \). Then we can test if \( \mathcal{H} \) is \( Z \)-distinct in time polynomial in \( \text{Size}(\mathcal{H}) \); we can test if \( \mathcal{S} \) is PFR (A.1.7.1) in time polynomial in \( \text{Size}(\mathcal{S}) \).

A.2. Definition of OK-SUFF and Proof of 5.2.1

We define (A.2.0) the predicate OK-SUFF. OK-SUFF gives a sufficient condition of \( \beta \)-solvability for SL-systems. The predicate \( e \)-good (used in A.2.0) is defined in A.2.1–A.2.3 (canonical(\( \mathcal{S} \)) and \( Q(X) \) where defined, respectively, in A.1.1, A.1.4).

A.2.0. Definition. Let \( \mathcal{S} = (\Gamma, X) \) be an SL-system.

\[ \text{OK-SUFF}(\mathcal{S}) = \text{true} \iff \exists \mathcal{S}^+ \in \text{canonical}(\mathcal{S}) \exists e \in Q(X) [\mathcal{S}^+ \text{ is } e \text{-good}] . \]

Intuitively \( \text{OK-SUFF}(\mathcal{S}) = \text{true} \) iff we can distinguish \( \mathcal{S} \) LHS terms using only nodes \( x \) s.t. for each equation \( M = z \) in \( \mathcal{S} \) the following conditions are satisfied:

\text{Condition 0:} \, \text{If the head of } M_\varnothing \text{ is an unknown then the degree of } M_\varnothing \text{ is not too large.}
\text{Condition 1:} \, \text{If the head of } M_\varnothing \text{ is } z \text{ then the order of } M_\varnothing \text{ is large enough.}
Condition 0 ensures that $\beta$-solvability can be tested in Polynomial Time (if unrestrained self-application is allowed then we get an NP-complete $\beta$-solvability problem [17, 2.6]).

Condition 1 ensures that a system solving algorithm exists (remember that, in general, for SL-systems $\beta$-solvability is undecidable). In particular it allows us to use the Böhm-out technique in our system solving algorithm.

All above conditions are formalized by $e$-good. In A.2.1–A.2.3 we define $e$-good.

Let $M = z \in S$ and head($M_{a}$) = $z$. Using the function rad (A.2.1.0) we can check if ord($M_{a}$) is large enough. Taking into account what our system solving algorithm can do $(Z, X, e)$-distinction (A.2.1.4) yields $\beta$-solvability. Note that $Z$-distinction (A.1.5.2) does not yield $\beta$-solvability.

**A.2.1. Definition.** Let $X, Z \subseteq \text{Var}$, $e \in Q(X)$, $\mathfrak{G} \subseteq A$, $M \in \mathfrak{F}$, and $\alpha \in \text{Seq}$.

0. We have

$$
\text{rad}(X, e, \mathfrak{G}, M, \alpha)
$$

= case

$M | \alpha \uparrow$ then $0$;

head($M_{a}$) \in X then $e(\text{head}(M_{a}), 1) - \deg(M_{a}) + \text{ord}(M_{a})$;

$\exists \beta < \alpha$ head($M_{a}$) = $\text{head}(M_{a})$

then $\max\{\deg(N_{\beta})|N \leq \alpha$ and head($N_{\beta}$) = head($M_{a}$) and $N \in \mathfrak{G}\}$

$+ 1 - \deg(M_{a}) + \text{ord}(M_{a})$;

$\neg \exists \beta < \alpha$ head($M_{a}$) = head($M_{a}$) then $\text{ord}(M_{a})$;

end.

1. $\text{rad}(X, e, \mathfrak{G}, M, \alpha) = \max\{\text{rad}(X, e, \mathfrak{G}, M, \alpha) | M \in \mathfrak{G}\}$.

2. $\text{rad}(X, e) = \lambda \mathfrak{G} \alpha. \text{rad}(X, e, \mathfrak{G}, \alpha)$.

3. A node $\alpha$ is said to be $(Z, X, e)$-safe in $\mathfrak{G}$ iff it is $(Z, X, e, \text{rad}(X, e))$-safe in $\mathfrak{G}$.

4. $\mathfrak{G}$ is said to be $(Z, X, e)$-distinct iff $\mathfrak{G}$ is $(Z, X, e, \text{rad}(X, e))$-distinct (A.1.5.1).

To test consistency of specifications we use the set PFRLR($S$).

**A.2.2. Definition.** Let $S$ be an SL-system. $\text{PFRLR}(S) = \{\langle x, M_{1}, \ldots, M_{m}, \Omega \rangle | xM_{1} \ldots M_{m} = z \in S \} \cup \{\langle \langle x, M_{1}, \ldots, M_{m}, \alpha \rangle | \exists k > 0 | xM_{1} \ldots M_{m+k} = z \in S \} \text{ and } \mathfrak{G} = \{x := \lambda \mathfrak{G} \Omega_{1} \ldots \Omega_{k} | x \in \mathfrak{G} \} xM_{1} \ldots M_{m+k} \mathfrak{G}\}$.

Finally $e$-good can be defined.

**A.2.3. Definition.** Let $S = (\Gamma, X)$ be a canonical SL-system and $e \in Q(X)$ (see A.1.4). $S$ is said to be $e$-good iff the following conditions are satisfied:

0. $\forall x \in (X \cap \text{head}({\text{left}(S)})) \{e(x, 1) = \min\{m | xM_{1} \ldots M_{m} = z \in S\}\}$.

1. $\forall xM_{1} \ldots M_{m} = z \in S [M_{e(x, 0)} \in \text{SOL}]$. 


2. \( \forall x M_1 \ldots M_m = z \in \mathcal{S} \) [\( \text{head}(M_{e(x,0)} \equiv z \Rightarrow e(x,2) = e(x,1) - m + \text{ord}(M_{e(x,0)}) \)].

3. \( \forall x M_1 \ldots M_m = z \in \mathcal{S} \) [\( \text{head}(M_{e(x,0)} \neq \text{FV}(M_{e(x,0)}) \Rightarrow e(x,2) \geq \text{ord}(M_{e(x,0)}) \)].

4. \( \forall x M_1 \ldots M_m = z \in \mathcal{S} \) [\( \text{head}(M_{e(x,0)}) \equiv x' \in X \Rightarrow e(x,2) \geq e(x',1) - \text{deg}(M_{e(x,0)}) + \text{ord}(M_{e(x,0)}) \)].

5. \( \text{PFRLR}(\mathcal{S}) \) is \((\text{right}(\mathcal{S}), X, e)\)-distinct.

6. Let \( \mathcal{F}_x = \{ M \in \text{PFRLR}(\mathcal{S}) | \text{head}(M_{e(0)}) \equiv x \} \). Then \( \forall x \in (X \cap \text{head(left}(\mathcal{S}))) \) [\( \langle e(x,0) \rangle \) is \((\text{right}(\mathcal{S}), X, e)\)-safe in \( \mathcal{F}_x \)].

The definition of \( \text{OK}_-\text{SUFF} \) is now complete (see A.2.0, A.1.1, A.1.4, A.2.3). We can now check that \( \text{OK}_-\text{SUFF}(\mathcal{S}) \) can be computed in Polynomial Time (A.2.4).

A.2.4. \textbf{Proof of 5.2.1.} Let \( e \in Q(X), \mathcal{F} \subset_f A, X, Z \subset_f \text{Var} \) s.t. \( \text{Card}(X) \leq \text{Size}(\mathcal{F}) \) and \( \text{Card}(Z) \leq \text{Size}(\mathcal{F}) \) and \( \alpha \in \text{Seq} \). Then the following tasks can be carried out in time polynomial in \( \text{Size}(\mathcal{F}) \): to compute \( \text{rad}(X, e, \mathcal{F}, M, \alpha) \) (in A.2.1.0); to test if \( \alpha \) is \((Z, X, e)\)-safe in \( \mathcal{F} \) (A.2.1.3); to test if \( \mathcal{F} \) is \((Z, X, e)\)-distinct (A.2.1.4). Then conditions A.2.3.0–A.2.3.6 can be tested in time polynomial in \( \text{Size}(\mathcal{F}) \). Thus, \( e \)-good can be computed in time polynomial in \( \text{Size}(\mathcal{F}) \). The only functions \( e \) that we need to consider in A.2.0 are defined below.

Let \( X = \{ x_1, \ldots, x_n \} \) and \( k = \max \{ \text{deg}(M_\alpha) | M \in \text{left}(\mathcal{S}) \text{ and } \alpha \in \text{BT}(M) \} \) + \( \max \{ \text{ord}(M_\alpha) | M \in \text{left}(\mathcal{S}) \text{ and } \alpha \in \text{BT}(M) \} \) + \( n + 1 \). For all \( i \in \{ 1, \ldots, n \} \) define

\[
e(x_i, 1) = \begin{cases} \text{if } x_i \notin \text{head(left}(\mathcal{S})) \text{ then } \min \{ m | x_i M_1 \ldots M_m = z \in \mathcal{S} \} \text{ else } \\ \max \{ \text{deg}(M_\alpha) | M \in \text{left}(\mathcal{S}) \text{ and } \alpha \in \text{BT}(M) \} + 1; \end{cases}
\]

\[
e(x_i, 0) = \begin{cases} \text{if } x_i \in \text{head(left}(\mathcal{S})) \text{ then } \text{an arbitrary } h \in \mathbb{N} \text{ s.t. } 1 \leq h \leq e(x_i, 1) \text{ else } \\ e(x_i, 1); \end{cases}
\]

\[
e(x_i, 2) = \begin{cases} \text{if } \exists x_j M_1 \ldots M_m = z \in \mathcal{S} \text{ [head}(M_{e(x_j,0)} \equiv z) \text{ then } \max \{ e(x_i, 1) - m + \text{ord}(M_{e(x_j,0)}) | \exists x_j M_1 \ldots M_m = z \in \mathcal{S} \text{ [head}(M_{e(x_j,0)} \equiv z) \} \\ \text{else } k + e(x_i, 1) - e(x_i, 0) + i; \end{cases}
\]

\[
e(x_i, 3) = 0.
\]

For each \( x \in X \) there are at most \( \text{Node(left}(\mathcal{S})) \) possible choices for \( e(x,0) \). Thus there are at most \( \text{Card}(X) \text{ Node(left}(\mathcal{S})) = \Theta(\text{Size}(\mathcal{S})^2) \) possible choices for \( e \). From this fact and the definition of \( \text{OK}_-\text{SUFF}(A.2.0) \) the thesis follows. \( \square \)

A.3. \textbf{Proof of 5.2.2}

We show (A.3.0) that \( \text{OK}_-\text{NEC} \) gives a necessary condition of \( \beta \)-solvability for \( SL \)-systems. This proves 5.2.2. The proof is in A.3.3–A.3.5.

A.3.0. \textbf{Proof of 5.2.2.} Follows from A.1.2, A.3.4, A.3.5, A.1.1. \( \square \)

A.3.1. \textbf{Example.} (i) Let \( Q = (\{ x(\lambda a. z) = z, x(\lambda a. u) = u \}, \{ x \}) \). Then \( Q \) is not \( LR \)-distinct (A.1.5.3). Thus, by A.1.0, \( \text{OK}_-\text{NEC}(Q) = false \) and, by 5.2.2, \( Q \) is not \( \beta \)-solvable.
(ii) Let $\mathcal{S} = \{ (x(λa.z) = z, xΩy = y), \{x\} \}$. Then $\mathcal{S}$ is not PFR (A.1.7.1). Thus, by A.1.0, $\text{OK}_\text{NEC}(\mathcal{S}) = \text{false}$ and, by 5.2.2, $\mathcal{S}$ is not $\beta$-solvable.

A.3.2. **Counterexample.** Of course, by 4.1, $\text{OK}_\text{NEC}$ is not a sufficient condition of $\beta$-solvability. Let $\mathcal{S} = \{ (x(λa.az) = z, x(λa.a(au)) = u, x(λa.a(α(αΩy))) = y), \{x\} \}$. We have $\text{OK}_\text{NEC}(\mathcal{S}) = \text{true}$, but $\mathcal{S}$ is not $\beta$-solvable.

We start the proof of 5.2.2. If $\mathcal{S}$ is a $\beta$-solvable SL-system and $M = z \in \mathcal{S}$ and head($M_# \equiv z$) then ord($M_# \equiv z$) cannot be too small. Thus the $\beta$-solvability of an SL-system depends also on the order of subterms of the LHS terms (A.3.3).

A.3.3. **Lemma.** Let $\mathcal{S} = (Γ, X)$ be a canonical SL-system. If $\mathcal{S}$ is $\beta$-solvable then $\exists α \text{ usf and agt for } left(\mathcal{S})$ s.t.: $∀M \in left(\mathcal{S}) \exists α \text{ usf and agt for } left(\mathcal{S})$ s.t.: $[\text{head}(M_# \equiv z) \in right(\mathcal{S}) \Rightarrow [\text{ord}(M_# \equiv z) = \max \{\text{ord}(L_# \equiv z) | L \in left(\mathcal{S})\}] and deg(M_# \equiv z) = 0]\}$.

**Proof.** If Card(left(\mathcal{S})) = 1 trivial. Let Card(left(\mathcal{S})) > 1. As in the proof of [1, 14.4.13] we can prove that there exists $α$ usf and agt for left(\mathcal{S}).

Let $D[\ ]$ be a $\beta$-solution for $\mathcal{S}$. Hence $∀M = z \in \mathcal{S}, D[M] = z$. Consider the standard reduction (* is a suitable substitution) $σ: D[M] \rightarrow M_# \equiv z$ where $α \in \text{Seq}$ is the first node usf and agt for left(\mathcal{S}) that comes on the head during the standard reduction $σ$ and $M = z \in \mathcal{S}$ (note that $α$ does not depend on the choice of $M = z \in \mathcal{S}$).

Because $α$ is the first useful node that comes on the head we have $∀M, N \in left(\mathcal{S}) |{\overline{H}_M} | ≥ \max \{\text{ord}(Q_# \equiv z) | Q \in left(\mathcal{S})\}$. Suppose that $\text{head}(M_# \equiv z) \in right(\mathcal{S})$.

Then, taking into account that $M \in left(\mathcal{S})$, we have $\max \{\text{ord}(L_# \equiv z) | L \in left(\mathcal{S})\} ≥ \text{ord}(M_# \equiv z) = |{\overline{H}_M} | ≥ \max \{\text{ord}(L_# \equiv z) | L \in left(\mathcal{S})\}$ and $\text{deg}(M_# \equiv z) = 0$. Thus, the thesis follows. □

Lemma A.3.3 can be strengthen as follows.

A.3.4. **Lemma.** Let $\mathcal{S} = (Γ, X)$ be a canonical SL-system. If $\mathcal{S}$ is $\beta$-solvable then $\mathcal{S}$ is $LR$-distinct.

**Proof.** By induction on Card(\mathcal{S}).

Case 0: Card(\mathcal{S}) = 1. Trivial.

Case 1: Card(\mathcal{S}) > 1. By Lemma A.3.3 $\exists α \text{ usf and agt for } left(\mathcal{S})$ s.t. $∀M \in left(\mathcal{S}) \exists α \text{ usf and agt for } left(\mathcal{S})$ s.t.: $[\text{head}(M_# \equiv z) \in right(\mathcal{S}) \Rightarrow [\text{ord}(M_# \equiv z) = \max \{\text{ord}(L_# \equiv z) | L \in left(\mathcal{S})\}] and deg(M_# \equiv z) = 0]\}$.

Let $\mathcal{S} \in \mathcal{S} /\sim_\sigma$ (A.0.2). $\mathcal{S}$ is $\beta$-solvable and, by induction hypothesis, left(\mathcal{S}) is right(\mathcal{S})-distinct. Hence, left(\mathcal{S}) is right(\mathcal{S})-distinct. This implies that left(\mathcal{S}) is right(\mathcal{S})-distinct. □

Consistency of specifications follows from $\beta$-solvability (A.3.5).
A.3.5. Lemma. Let $\mathcal{S} = (\Gamma, X)$ be a canonical SL-system and $\mathcal{T}$ be a sms theory. If $\mathcal{S}$ is $\mathcal{T}$-solvable then $\mathcal{S}$ is PFR.

**Proof.** Let $D \llbracket \cdot \rrbracket$ be a $\mathcal{T}$-solution for $\mathcal{S}$. Note that $\forall M \in \text{prefix}(\mathcal{S}) D[M] \in \text{SOL}$. If $\mathcal{S}$ is not PFR then $\text{prefix}(\mathcal{S})$ is not distinct (A.1.7.1). Hence there are $M, N \in \text{prefix}(\mathcal{S})$ s.t. $\text{ind}(\text{prefix}(\mathcal{S}), M, N)$ (A.0.0). Then, by [8, 3.4.0], $\text{ind}(D[\text{prefix}(\mathcal{S})], D[M], D[N])$. Hence $\text{head}(D[M] = \text{head}(D[N])$. This is absurd since $\mathcal{S}$ is canonical. □

A.4. Proof of 5.2.3

We show (A.4.0) that OK-SUFF gives a Polynomial Time sufficient condition of $\beta$-solvability for SL-systems. This proves 5.2.3. The proof is in Sections A.4.2, A.4.3.

A.4.0. Proof of 5.2.3. Follows from A.1.2, A.4.3.2 and A.4.3.3. □

A.4.1. Example. (i) Let $\mathcal{S} = \{xy\Omega(\lambda a. x\Omega) = y, \ x\Omega(\lambda ab. z) = z\}, \{x\}$. Since OK-SUFF($\mathcal{S}$) then, by 5.2.3, $\mathcal{S}$ is $\beta$-solvable. A $\beta$-solution is $D = \lambda y t_1, t_2, t_3, (U^3_1 y)$.

(ii) The system $\mathcal{S}$ in 5.5 is $\beta$-solvable.

The proof of 5.2.3 proceeds as follows. First (in A.4.2) we give an algorithm to find $\beta$-solutions to HSL-systems (3.0). Then (in A.4.3) we use such algorithm to build an algorithm to find $\beta$-solutions to SL-systems.

A.4.2. Solving HSL-systems

In A.4.2.0–A.4.2.4 we solve HSL-systems with equations having form $xM = z$. In A.4.2.5 we solve HSL-systems.

At a safe node (A.2.1.3) nonsubstitutible variables are harmless for Böhm-out (A.4.2.0).

A.4.2.0. Lemma. Let $X = \{x_1, \ldots, x_n\}, \ Z \subseteq \text{Var}, \ b \in (\text{Var} - (X \cup Z))$, $\mathcal{F} \subseteq \Lambda(X \cup Z)$ with $\mathcal{F}$ $\lambda$-free, $\alpha \in \text{Seq}$ s.t. $\mathcal{F} \mid \alpha \downarrow$, $\alpha$ is agt and adh for $\mathcal{F}$ and $\alpha$ is $(Z, \emptyset, 0)$-safe in $\mathcal{F}$. Then $\exists G[\cdot] \equiv (\lambda x_1 \ldots x_n. [\cdot])G \in \Lambda[\cdot]$ s.t.:

0. $\forall M \in \mathcal{F}[G[M] = \text{if head}(M) \in Z \text{ then head}(M) \text{ else } b\tilde{H}_{M, n}]$;

1. $\forall M, N \in \mathcal{F}[M \sim_n N \Rightarrow |\tilde{H}_{M, n}| = |\tilde{H}_{N, n}|].$

**Proof.** By induction on length($\alpha$).

Case 0: $\alpha = \langle \cdot \rangle$. Since $\langle \cdot \rangle$ is $(Z, \emptyset, 0)$-safe in $\mathcal{F}$ we can choose $G[\cdot] \equiv (\lambda x_1 \ldots x_n. [\cdot])G_1 \ldots G_n$, where $\forall i \in \{1, \ldots, n\} G_i \equiv \text{if } x_i \in \text{head}(\mathcal{F}) \text{ then } b \text{ else arbitrary}.$

Case 1: $\alpha = \langle j \rangle \ast \beta$. Then $\exists M \in \mathcal{F} \text{head}(M) \in Z$, because $\alpha$ is agt and adh for $\mathcal{F}$. Moreover, there are $x \in X$ and $m \in \mathbb{N}$ s.t. $\forall M \in \mathcal{F}[\text{head}(M) \equiv x$ and $\text{deg}(M) = m]$. W.l.o.g. we can assume $\text{head}(\mathcal{F}) = \{x_1\}$. Let $d = \max\{\text{deg}(M) | \theta \leq \alpha \text{ and } \text{head}(M) \equiv x_1 \text{ and } M \in \mathcal{F}\}$, $C[\cdot] \equiv (\lambda x_1. [\cdot])P_d$, $C[\mathcal{F}_{\langle j \rangle}] = \{C[M_{\langle j \rangle}] | M \in \mathcal{F}\}$. We have $C[\mathcal{F}_{\langle j \rangle}] \downarrow$, $\beta$ is agt and adh for $C[\mathcal{F}_{\langle j \rangle}]$. From this it follows (see [16, Part 1,
7.6.4 and 8.0] that \( \beta \) is \((Z, \emptyset, 0)\)-safe in \( C[\Sigma] \). Let \( r = \max \{ \text{ord}(C[\Sigma]) \mid M \in \mathcal{F} \} \),
\( \tilde{u} = u_1 \ldots u_r \) (\( \tilde{u} \) fresh) and \( \mathcal{G} = \{ C[\Sigma] \tilde{u} \mid M \in \mathcal{F} \} \). We have \( \mathcal{G} \subset \mathcal{L} \)
\( \{ x_2, \ldots, x_n, \tilde{u} \} \cup Z \), \( \mathcal{G} \) is \( \lambda \)-free, \( \mathcal{G}[\beta] \downarrow \), \( \beta \) is agt and adh for \( \mathcal{G} \) and \( \beta \) is \((Z, \emptyset, 0)\)-safe in \( \mathcal{G} \). Hence by induction hypothesis
\( \exists D[ ] = (\lambda x_2 \ldots x_n u[ ] D_2 \ldots D_n E_1 \ldots E_r, \tilde{F} \in A[ ] \)
* s.t. \( \forall Q \in \mathcal{G} \{ D[Q] = \text{if head}(Q_\beta) \in Z \text{ then head}(Q_\beta) \text{ else } b\tilde{H}_{Q, \beta} \} \) and
\( \forall P, Q \in \mathcal{G} \{ P \sim \beta Q \Rightarrow | \tilde{H}_{P, \beta}| = | \tilde{H}_{Q, \beta}| \} \).

Note that \( \forall M \in \mathcal{G} \{ \text{head}(C[\Sigma] \tilde{u}) \} \in Z \text{ if head}(M_\beta) \in Z \) and \( \forall M, N \in \mathcal{G} \{ M \sim_\beta N \Rightarrow M[\Sigma] \sim_\beta N[\Sigma] \Rightarrow C[\Sigma] \tilde{u} \sim_\beta C[\Sigma] \tilde{u} \Rightarrow | \tilde{H}_{CM, \Sigma} | | \tilde{u}| = | \tilde{H}_{CN, \Sigma} | | \tilde{u}| \} \). Let \( q \in \mathbb{N} \) s.t. \( q \geq d - m + 1 \).

Define \( G[ ] = (\lambda x_1 \ldots x_n[ ] P_q D_2 \ldots D_n A_1 \ldots A_q, \text{ where } \forall i \in \{ 1, \ldots, q \} A_i \equiv \text{if } i = d - m + 1 \text{ then } (\tilde{t}_0 \ldots t_{q+m-2}, t_1 E_1 \ldots E_r, \tilde{F}) \text{ else arbitrary.} \)

When \( \mathcal{G} \subset Z \) we can get rid of the context \( G[ ] \) in A.4.2.0.

A.4.2.1. Corollary. Let \( Z \subset \text{Var}, b \in (\text{Var} - Z), \mathcal{G} \subset Z, x \in \text{Seq} \) s.t. \( \mathcal{G} \mid \alpha \downarrow \), \( \alpha \) is agt and adh for \( \mathcal{G} \) and \( \alpha \) is \((Z, \emptyset, 0)\)-safe in \( \mathcal{G} \). Then \( \exists F \in A(\{ b \}) \) s.t.:

0. \( \forall M \in \mathcal{G} \{ FM = \text{if head}(M_\alpha) \in Z \text{ then head}(M_\alpha) \text{ else } b\tilde{H}_{M, \alpha} \}; \)
1. \( \forall M, N \in \mathcal{G} \{ M \sim_\alpha N \Rightarrow | \tilde{H}_{M, \alpha}| = | \tilde{H}_{N, \alpha}| \} \).

Proof. Let \( q = \max \{ \text{ord}(M) \mid M \in \mathcal{G} \} \) and \( \mathcal{G} = \mathcal{G} a_1 \ldots a_q = \{ M a_1 \ldots a_q \mid M \in \mathcal{G} \} \)
\( (a_1, \ldots, a_q \) fresh). Then, by A.4.2.0, there exists \( G[ ] = (\lambda a_1 \ldots a_q[ ] \) satisfying A.4.2.0.0.1. Define \( F \equiv \lambda t.tG \).

Terms nonequivalent at a given node can be separated (see 3.1) [9].

A.4.2.2. Lemma (Coppo et al. [9]). Let \( Z \subset \text{Var}, \mathcal{G} \subset Z, \alpha \in \text{Seq} \) s.t. \( \mathcal{G} \mid \alpha \downarrow \), \( \alpha \) is agt and adh for \( \mathcal{G} \) and \( \alpha \) is \((Z, \emptyset, 0)\)-safe in \( \mathcal{G} \). Then \( \exists G \in A \forall M, N \in \mathcal{G} \{ GM = b_M \text{ and } [b_M = b_N \Rightarrow M \sim_\alpha N] \text{ and } [b_M \notin Z] \} \).

Proof. Follows from [1, 10.3.13 and 10.4.11].

At a safe node nonsubstitutable variables are harmless to separation (A.4.2.3).

A.4.2.3. Lemma. Let \( Z \subset \text{Var}, \mathcal{G} \subset Z, \alpha \in \text{Seq} \) s.t. \( \mathcal{G} \mid \alpha \downarrow \), \( \alpha \) is agt and adh for \( \mathcal{G} \) and \( \alpha \) is \((Z, \emptyset, 0)\)-safe in \( \mathcal{G} \). Then \( \exists G \in A \forall M, N \in \mathcal{G} \{ GM = \text{if head}(M_\alpha) \in Z \text{ then head}(M_\alpha) \text{ else } b\tilde{H}_{M, \alpha} \} \text{ and } [b_{M, \alpha} \notin Z] \text{; } \)
1. \( [b_M = b_N \Rightarrow M \sim_\alpha N] \).

Proof. Let \( u \in (\text{Var} - Z) \). Using A.4.2.1 define \( E \in A(\{ u \}) \) s.t. \( \forall M, N \in \mathcal{G} \{ EM = \text{if head}(M_\alpha) \in Z \text{ then head}(M_\alpha) \text{ else } u\tilde{H}_{M, \alpha} \} \text{ and } [M \sim_\alpha N \Rightarrow | \tilde{H}_{M, \alpha}| = | \tilde{H}_{N, \alpha}| ] \). Let \( \mathcal{B} = \{ M \mid M \in \mathcal{G} \text{ and head}(M_\alpha) \notin Z \} \). Using A.4.2.2 define \( F \in A \) s.t. \( \forall M, N \in \mathcal{B} \)
\[ F[M = b_{M,a} \text{ and } [b_{M,a} = b_{N,a} \Rightarrow M \sim_z N] \text{ and } [b_{M,a} \notin Z]]. \]

Let \( * = [b_{M,a} := \lambda u_1 \ldots u_\omega M.a | M \in \mathcal{B}] \).

Define \( G = \lambda t.(\lambda u. E)(F^* t^t) t. \]

In A.4.2.4 is a sufficient condition for the existence of a common \( \beta \)-left-inverse for a finite set of combinators.

**A.4.2.4. Lemma.** Let \( \mathcal{S} = (\Gamma, \{x\}) \) be a canonical HSL-system with equations having from \( xM = z \). If \( \text{left}(\mathcal{S}) \) is \( (\text{right}(\mathcal{S}), \emptyset, 0) \)-distinct then \( \mathcal{S} \) is \( \beta \)-solvable.

**Proof.** By induction on \( \text{Card}(\mathcal{S}) \). Let \( Z = \text{right}(\mathcal{S}) \).

Case 0: \( \text{Card}(\mathcal{S}) = 1 \). Then \( \Gamma = \{xM = z\} \). Let \( a \in \text{BT}(M) \) s.t. \( \text{head}(M_a) \in Z \) and \( \text{deg}(M_a) = 0 \). Then by A.4.2.3 (with \( Y = \{M\} \)) \( \exists G \in \Lambda^\omega G. M = z \).

Case 1: \( \text{Card}(\mathcal{S}) > 1 \). Then there exists \( z = \langle 0 \rangle \beta \) usf and agt for \( \text{left}(\mathcal{S}) \) s.t. \( z \) is \( (Z, \emptyset, 0) \)-safe for \( \text{left}(\mathcal{S}) \). Then, by A.4.2.3, \( \exists G \in \Lambda \forall xM \in \text{left}(\mathcal{S}) \) \( [G. M = z \text{ if } \text{head}(M) \in Z \text{ then } \text{head}(M) \text{ else } b_{M,a} M] \). Let \( \Theta = \{xM = z | xM = z \in \mathcal{S} \text{ and } \text{head}(x(M)) \notin Z \} \) and \( \Theta / \sim_a = \{\Sigma_1, \ldots, \Sigma_k\} \) (see A.0.2). By A.1.5.1.1 \( \forall \Sigma \in \Theta / \sim_a \text{ left}(\Sigma) \) is \( (Z, \emptyset, 0) \)-distinct, hence by induction hypothesis, \( \forall j \in \{1, \ldots, k\} \exists F_j \in \Lambda^0 \forall xM = z \in \Sigma_j [F_j M = z] \). Then \( D \equiv G [b_{M,a} := F_{\min(j, xM = z \in \Sigma_j)} | xM = z \in \Theta] \) is a \( \beta \)-solution for \( \mathcal{S} \).

From A.4.2.4 we obtain a sufficient condition of \( \beta \)-solvability for HSL-systems (A.4.2.5).

**A.4.2.5. Lemma.** Let \( \mathcal{S} = (\Gamma, \{x\}) \) be a canonical HSL-system (see (A.2.2, A.2.1.4). If \( \text{PFRLR}(\mathcal{S}) \) is \( (\text{right}(\mathcal{S}), \emptyset, 0) \)-distinct then \( \mathcal{S} \) is \( \beta \)-solvable.

**Proof.** By induction on \( \text{Card}(\{\text{deg}(M) | M \in \text{left}(\mathcal{S})\}) = d(\mathcal{S}) \).

Case 0: \( d(\mathcal{S}) = 1 \). Let \( m = \min(\text{deg}(M) | M \in \text{left}(\mathcal{S})) \), \( \Gamma' = \{x(M_1, \ldots, M_m) = z | xM_1 \ldots M_m = z \in \Gamma\} \) and \( \mathcal{S}' = (\Gamma', \{x\}) \). By A.4.2.4 \( \mathcal{S}' \) is \( \beta \)-solvable. Let \( F \) be a \( \beta \)-solution for \( \mathcal{S}' \). Then \( D \equiv \lambda t_1 \ldots t_m. F(t_1, \ldots, t_m) \) is a \( \beta \)-solution for \( \mathcal{S} \).

Case 1: \( d(\mathcal{S}) > 1 \). Let \( m = \min(\text{deg}(M) | M \in \text{left}(\mathcal{S})) \),

\[ \mathcal{S}_1,m = (\Gamma_{1,m}\{x\}) = (\{xM_1 \ldots M_m = z | xM_1 \ldots M_m = z \in \mathcal{S}, X\}), \]

\[ \mathcal{S}_2,m = (\Gamma_{2,m}\{x\}) = (\{xM_1 \ldots M_m = z | \exists k > 0 [xM_1 \ldots M_{m+k} = z \in \mathcal{S} \text{ and } \}

\[ * \equiv [x := \lambda t. t\Omega_1 \ldots \Omega_k | x \in \text{BT}(xM_1 \ldots M_{m+k}) \text{ and } \]

\[ \text{head}(xM_1 \ldots M_{m+k}) \equiv \}] \) and

\[ (xM_1 \ldots M_{m+k})^* = xM'_1 \ldots M'_{m+k}) \}). \{x\} \).

\[ \mathcal{S}_3,m = (\Gamma_{3,m}\{x\}) = (\Gamma_{1,m} \cup \Gamma_{2,m}, \{x\}). \]
Since PFRLR(\(\mathcal{S}\)) is (right(\(\mathcal{S}\)), \(\emptyset, 0\))-distinct then left(\(\mathcal{S}_{3,m}\)) is (right(\(\mathcal{S}_{3,m}\)), \(\emptyset, 0\))-distinct. Hence by induction hypothesis \(d(\mathcal{S}_{3,m}) = 1\) \(\mathcal{S}_{3,m}\) is \(\beta\)-solvable.

Let \(F\) be a \(\beta\)-solution for \(\mathcal{S}_{3,m}\). Let \(\mathcal{S}_{4,m} = (\nu, m, \{x\}) = (\{x_{M_1 \ldots M_n = z | xM_1 \ldots M_n = z \in \mathcal{S} and n > m\}, \{x\})\). Note that PFRLR(\(\mathcal{S}_{4,m}\)) is (right(\(\mathcal{S}_{4,m}\)), \(\emptyset, 0\))-distinct and \(d(\mathcal{S}_{4,m}) < d(\mathcal{S})\). Hence, by induction hypothesis, \(\mathcal{S}_{4,m}\) is \(\beta\)-solvable.

Let \(H\) be a \(\beta\)-solution for \(\mathcal{S}_{4,m}\). Then \(D \equiv \lambda t_1 \ldots t_m. F(Ht_1 \ldots t_m)t_1 \ldots t_m\) is a \(\beta\)-solution for \(\mathcal{S}\).

A.4.2.6. Remark. The algorithms in A.4.2.0–A.4.2.5 are all Polynomial Time.

A.4.2.7. Example. (i) Let \(\mathcal{S} = (\{x(ab. z) = z, x(ab. ay)\Omega = y\}, \{x\})\). By A.4.2.5 \(\mathcal{S}\) is \(\beta\)-solvable. A \(\beta\)-solution for \(\mathcal{S}\) is \(D \equiv \lambda t. t(ab.t\Omega)\Omega\).

(ii) Let \(\mathcal{S}' = (\{x(ab. z) = z, x(abc.y)\Omega = y\}, \{x\})\). By A.4.2.5 \(\mathcal{S}'\) is \(\beta\)-solvable. A \(\beta\)-solution for \(\mathcal{S}'\) is \(D \equiv \lambda t. t\Omega\Omega\).

(iii) Let \(\mathcal{S}'' = (\{x(ab. az) = z, x(ab.a(u)) = u, x(ab.a(aIy)) = y\}, \{x\})\). PFRLR(\(\mathcal{S}''\)) is not (right(\(\mathcal{S}''\)), \(\emptyset, 0\))-distinct. However \(\mathcal{S}''\) is \(\beta\)-solvable. A \(\beta\)-solution for \(\mathcal{S}''\) is \(G \equiv \lambda t. tI\).

A.4.3. Solving SL-systems

Using the algorithm in A.4.2.5 we give (A.4.3.0–A.4.3.3) a Polynomial Time algorithm to construct a \(\beta\)-solution to an SL-system satisfying OK \_SUFF. This concludes the proof of 5.2.3.

A.4.3.0. Notation. Let \(\{\bar{x}\} = \{x_1, \ldots, x_n\} \subset \text{Var}\) and \(e \in Q(\{\bar{x}\})\) (see A.1.4).

0. \(D_{\bar{x}, e} \equiv \lambda t_{e(x_0)} \ldots t_{e(x_1)} t_{e(x_0)}(t_{e(x_0)}(t_{e(x_0)}(t_{e(x_0)}(\Omega_1 \ldots t \Omega_{e(x_1)})) \ldots t \Omega_{e(x_0)})) \ldots \Omega_{e(x_1)} \Omega_2 \ldots \Omega_{e(x_2)}\).

1. \(D_{\bar{x}, e}[\ ] \equiv (\lambda x_1 \ldots x_n[\ ])) D_{x_1, e} \ldots D_{x_n, e}\).

We break self-application by finding (A.4.3.1) a common \(\beta\)-solution to an infinite class of systems (this is the core of the algorithm in A.4.3.2).

A.4.3.1. Lemma. Let \(\mathcal{S} = (\Gamma, \{\bar{x}, u\})\) be a canonical SL-system with equations having form \(u\bar{M} = z\) where \(u \notin FV(\bar{M})\).

If OK \_SUFF(\(\mathcal{S}\)) \(\Rightarrow \exists e \in Q(\{\bar{x}\}) \exists F \in A^0 \forall G[\ ] \exists D_{\bar{x}, e}[\ ] \forall uM_1 \ldots M_m = z \in \mathcal{S} FG[M_1] \ldots G[M_m] = z\).

Proof. Let \(e' \in Q(\{\bar{x}, u\})\) s.t. \(\mathcal{S}\) is \(e'\)-good and \(e \in Q(\{\bar{x}\})\) be the restriction of \(e'\) to \(\{\bar{x}\}\).
Since \(\mathcal{S}\) is \(e'\)-good PFRLR(\(\mathcal{S}\)) is (right(\(\mathcal{S}\), \(\{\bar{x}, u\}, e')\))-distinct. Moreover, since \(\forall u\bar{M} = z \in \mathcal{S} \{u \notin FV(\bar{M})\}\), we have that PFRLR(\(\mathcal{S}\)) is (right(\(\mathcal{S}\), \(\{\bar{x}\}, e\)))-distinct (see A.2.3). From this follows (see [16, Part 1, 9.2.2 and 7.8]) that \(D_{\bar{x}, e}[\text{PFRLR}(\mathcal{S})]\) is
(right($) or $\emptyset$, 0)-distinct. Let $S_1 = \{uD_{X_e}[M_1] \ldots D_{X_e}[M_m] = z | uM_1 \ldots M_m = z \in S\}, \{u\}$). From the above discussion follows that $S_1$ is e-good. Hence, by A.4.2.5, $S_1$ is $\beta$-solvable. Let $F$ be a $\beta$-solution for $S_1$. Let $G[ ] \supseteq D_{X_e}[M_1] \ldots D_{X_e}[M_m] = z$.

**A.4.3.2. Lemma.** Let $S = (\Gamma, X)$ be a canonical SL-system. If OK _SUFF($S$) then $S$ is $\beta$-solvable.

**Proof.** Let $e \in Q(X)$ (A.1.4) st.:
0. $S$ is e-good (A.2.3);
1. $d = \max \{\max \{|\deg(M_{e(x,0)})| xM_1 \ldots M_m \in \left(S\right)\}, \max \{|e(x,2)| x \in X\}\}$

By A.2.0 such an $e$ exists. Let $x \in X$ e(x, 3) = $d + 1 + \max \{\ord(M_{e(x,0)}) - \deg(M_{e(x,0)})| xM_1 \ldots M_m \in \left(S\right)\}$. Then $\forall x \in X$ OK _SUFF($S(x)$). Hence, by A.4.3.1, $\forall x \in X \exists F_x \subseteq A \forall C[ ] \supseteq D_{X_e}[ ] \forall uM_1 \ldots M_m = z \in S(x) F_x C[M_1] \ldots C[M_m] = z$.

Let $x \in X$ define:

$A(x) = \{(i, d + 1 + \ord(M_{e(x,0)}) - \deg(M_{e(x,0)}))| xM_1 \ldots M_m \in \left(S\right)\}$

$B(x) = \{(e(x',0) + \ord(M_{e(x,0)}) - \deg(M_{e(x,0)})| xM_1 \ldots M_m \in \left(S\right)\}$

Note that $\forall i \in N \ [(i, i) \notin A(x)$ and $(i, i) \notin B(x)]$.

0. Remark. $\forall x \in X[A(x) \cap B(x) = \emptyset]$. In fact (with obvious notation):

$\forall (i, d + 1 + b - q) \in A(x)$

$\forall (e(x',0) + b' - q'', d + 1 + b' - q' + e(x',1) - e(x',2)) \in B(x)$

we have

$[i = e(x',0) + b' - q', d + 1 + b - q = d + 1 + b' - q' + e(x',1) - e(x',2)]$

$\Rightarrow [e(x',0) + b' - q' \leq b$ and $e(x',2) - e(x',1) + b - q = b' - q' ]$

$\Rightarrow [e(x',2) \leq e(x',1) - e(x',0) + q]$.

Absurd by A.2.3.6 and A.1.5.0.0.
We have \( b'' - q'' - b' + q' = e(x',0) - e(x'',0) \) and \( e(x'',2) - e(x',2) = e(x'',1) - e(x',0) \) \( e(x',0) - e(x',1) \) \( \Rightarrow [x' = x'' \text{ and } b' - q' = b'' - q''] \) (by A.1.4).

\( \diamond_1 \)

\( \forall x \in X \) define:

1(\( x \)) \( \equiv [a_{x,i,i} := P_d | \exists k \in \mathbb{N}(i,k) \in A(x)] \),

2(\( x \)) \( \equiv [a_{x,i,k} := \lambda t_1 \ldots t_{2(d-k)} + e(x,1) + e(x,2) + e(x,3)] F_{x} t_{d+1-k} + e(x,3) \ldots t_{d-k} + e(x,1) + e(x,3) | (i,k) \in A(x)] \),

3(\( x \)) \( \equiv a_{x,i,i} := P_d | \exists k \in \mathbb{N}(i,k) \in B(x)] \),

4(\( x \)) \( \equiv [a_{x,i,k} := \lambda t_1 \ldots t_{2(d-k)} + e(x,1) + e(x,2) + e(x,3)] F_{x} t_{d+1-k} + e(x,3) \ldots t_{d-k} + e(x,1) + e(x,3) | (i,k) \in B(x)] \),

\( D(x,e) \) \( \equiv \lambda t_1 \ldots t_{e(x,0)} \ldots t_{e(x,1)} \cdot t_{e(x,0)} \)

\( (t_{e(x,0)} a_{x,1,1} \ldots a_{x,1,e(x,3)} t_{1} \ldots t_{e(x,0)} \ldots t_{e(x,1)}) \)

\( \ldots \)

\( (t_{e(x,0)} a_{x,e(x,2),1} \ldots a_{x,e(x,2),e(x,3)} t_{1} \ldots t_{e(x,0)} \ldots t_{e(x,1)}). \)

\( \forall x \in X \) define

\( G_x \equiv D(x,e)^{1(x)^2(x)} 3(x)^4(x) \) \( \quad \text{and} \quad G[ ] \equiv (x_1 \ldots x_n[ ]) G_{x_1} \ldots G_{x_n}. \)

We verify that \( G[ ] \) is a \( \beta \)-solution for \( \mathcal{F} \).

Check: Let \( xM_1 \ldots M_m = z \in \mathcal{F} \). We have (with obvious meaning of the symbols):

\( G[xM_1 \ldots M_m] = G[M_{e(x,0)}] (G[M_{e(x,0)}] \bar{A}_{x,1} G[M_1] \ldots G[M_{e(x,1)}]) \)

\( \ldots \)

\( (G[M_{e(x,0)}] \bar{A}_{x,e(x,2)} G[M_1] \ldots G[M_{e(x,1)}]) \)

\( G[M_{e(x,1)} + 1] \ldots G[M_m]. \)

Case 0: \( M_{e(x,0)} = \lambda a_1 \ldots a_r \). By A.2.3.2 \( e(x,2) + m - e(x,1) = r \). We have \( G[M_{e(x,0)}] = (\lambda a_1 \ldots a_r \cdot L_1 \ldots L_{m-e(x,1)} + e(x,2)) = z. \)

Case 1: \( M_{e(x,0)} = \lambda a_1 \ldots a_r Q_1 \ldots Q_r \). By A.2.3.3 \( e(x,2) \geq r \). We have

\( G[xM_1 \ldots M_m] = G[M_{e(x,0)}] A_{x,h_1} \ldots A_{x,h,e(x,3)} G[M_1] \ldots G[M_{e(x,1)}] \)

\( B_1 \ldots B_{e(x,2)} - r e G[M_{e(x,1)} + 1] \ldots G[M_m] \)
\begin{align*}
&= A_{x,h} G[Q^*] \cdots G[Q^*_q] A_{x,h,r_1} \cdots A_{x,h,e(x,3)} G[M_1] \cdots G[M_{e(x,1)}] \\
& B_1 \cdots B_{e(x,2)} - h + q G[M_{e(x,1)+1}] \cdots G[M_m] \\
&= A_{x,h,d+1 + r - q} G[Q^*_1] \cdots G[Q^*_q] A_{x,h,r_1} \cdots A_{x,h,d+r-q} \\
& A_{x,h,d+2 + r - q} \cdots A_{x,h,e(x,3)} G[M_1] \cdots G[M_{e(x,1)}] \\
& B_1 \cdots B_{e(x,2)} - r + q G[M_{e(x,1)+1}] \cdots G[M_m] \quad \text{(from } 1(x)) \\
&= z \quad \text{(from } 2(x)),
\end{align*}

Case 2: $M_{e(x,0)} = \lambda a_1 \ldots a_r \ yQ_1 \ldots Q_q$, $y \in X$. By A.2.3.4 $e(x,2) \geq e(y,1)$ $- q + r > r$.

We have

\begin{align*}
G[M_1 \ldots M_m] &\quad = G_y G[Q^*] \cdots G[Q^*_q] (G[M_{e(x,0)}] A_{x,r+1} G[M_1] \cdots G[M_{e(x,1)}]) \\
&\vdots \\
& (G[M_{e(x,0)}] A_{x,e(x,2)} G[M_1] \cdots G[M_{e(x,1)}]) \\
& G[M_{e(x,1)+1}] \cdots G[M_m] \\
&= G[M_{e(x,0)}] A_{x,e(y,0)+r-q} G[M_1] \cdots G[M_{e(x,1)}] C_1 \cdots C_{e(x,2)+e(y,2)-e(y,1)-r+q} \\
& G[M_{e(x,1)+1}] \cdots G[M_m] \\
&= G_y G[Q^*] \cdots G[Q^*_q] A_{x,e(y,0)+r-q} G[M_1] \cdots A_{x,e(y,0)+r-q,e(x,3)} \\
& G[M_1] \cdots G[M_{e(x,1)}] \\
& C_1 \cdots C_{e(x,2)+e(y,2)-e(y,1)-r+q} G[M_{e(x,1)+1}] \cdots G[M_m] \\
&= A_{x,e(y,0)+r-q,e(x,0)+r-q} L_1 \cdots L_{e(y,2)} A_{x,e(y,0)+r-q,e(y,1)+r-q} \\
& A_{x,e(y,0)+r-q,e(x,3)} G[M_1] \cdots G[M_{e(x,1)}] \\
& C_1 \cdots C_{e(x,2)+e(y,2)-e(y,1)-r+q} G[M_{e(x,1)+1}] \cdots G[M_m] \\
&= A_{x,e(y,0)+r-q,d+1+e(y,1)-e(y,2)+r-q} L_1 \cdots L_{e(y,2)} H_1 \cdots H_{e(x,3)-e(y,1)-r+q} \\
& G[M_1] \cdots G[M_{e(x,1)}] C_1 \cdots C_{e(x,2)+e(y,2)-e(y,1)-r+q} \\
& G[M_{e(x,1)+1}] \cdots G[M_m] \quad \text{(from } 3(x)) \\
&= z \quad \text{(from } 4(x)). \quad \square
A.4.3.3. Remark. Let $\mathcal{S}$ be an SL-system. If $\text{OK}_{\text{SUFF}}(\mathcal{S})$ then a $\beta$-solution for $\mathcal{S}$ can be constructed in Polynomial Time. This follows from A.1.1, A.4.2.6 and the algorithms in A.4.3.1, A.4.3.2.

A.5. Proof of 5.2.4

Let $\mathcal{S} = (\Gamma, X)$ be a quasi-regular SL system s.t. $\text{OK}_{\text{NEC}}(\mathcal{S}) = \text{true}$. By 5.2.2, A.1.0 there exists $\mathcal{S}^+ \in \text{canonical}(\mathcal{S})$ [$\mathcal{S}^+$ is PFR and LR-distinct]. Let $\text{Good}_{\text{Candidates}}$ be the set of $e \in Q(X)$ (A.1.4) s.t. $e$ is chosen as in the proof of A.2.4 but using always the "else" branch for the if-then-else defining $e(x_i, 2)$ and $\mathcal{S}^+$ satisfies conditions A.2.3.0, A.2.3.1, A.2.3.3, A.2.3.4. Note that, as in A.2.4, $\text{Card}(\text{Good}_{\text{Candidates}}) = O(\text{Size}(\mathcal{S}^+))$.

Given $e$ in $\text{Good}_{\text{Candidates}}$ we define the $e$-relaxation $\mathcal{S}^e$ of $\mathcal{S}^+$ as follows. Let $\mathcal{S}1$ be the relaxation of $\mathcal{S}^+$ s.t. $\mathcal{S}1$ satisfies A.2.3.2. $\mathcal{S}1$ can be computed in Polynomial Time from $\mathcal{S}^+$ simply using A.2.3.2. We call the relaxation that takes from $\mathcal{S}^+$ to $\mathcal{S}1$ relaxation 1. A $z$-occurrence in an SL-system $Q$ is a LHS occurrence of a RHS variable of $Q$. Let $k$ be a large integer (e.g. $k$ as in the proof of A.2.4). Let $\mathcal{S}^e$ be the relaxation of $\mathcal{S}1$ s.t. all the $z$-occurrences of $\mathcal{S}^+$ (and hence of $\mathcal{S}^e$) not relaxed in relaxation 1 have order $k$.

Since $\mathcal{S}^+$ is canonical, PFR and LR-distinct from A.2.1.0, A.2.1.4 it follows that there is $e$ in $\text{Good}_{\text{Candidates}}$ s.t. $\mathcal{S}^e$ is $e$-good. Note that $e$ and $\mathcal{S}^e$ can be found in Polynomial Time. We have $\text{OK}_{\text{SUFF}}(\mathcal{S}^e) = \text{true}$. Let $\mathcal{S}1$ be the relaxation of $\mathcal{S}$ s.t. $\mathcal{S}^e \in \text{canonical}(\mathcal{S}1)$. Then $\text{OK}_{\text{SUFF}}(\mathcal{S}1) = \text{true}$. Thus, $\text{regular}_{\text{SL}}(\mathcal{S}1) = \text{true}$ and the thesis follows. For the attentive reader it will not be difficult to find a more parsimonious choice of $k$.

Appendix B. Proof of 5.6

We codify the satisfiability problem for propositional formulas with SL-systems. Let $\text{PropForm}$ (PropVar) be the set of Propositional Formulas (Variables). Let $L: \text{PropForm} \mapsto A$ be defined as follows: $L(x) = x$, $L(\neg A) = L(A)^\dagger U_2^\dagger$, $L(A \lor B) = L(A)U_2^\dagger L(B)$. We are representing true with $U_2^\dagger$ and false with $U_1^\dagger$. Let $A \in \text{PropForm}$ s.t. $FV(A) = \{x_1, \ldots, x_n\}$. We define $\text{Transl}(A) = \{(L(A)z\Omega = z) \cup \{x_i^z = z \mid i = 1, \ldots, n\}, \{x_1, \ldots, x_n\}\}$. $\text{Transl}(A)$ is an SL-system and is $\beta$-solvable iff $A$ is satisfiable. In fact:

($\Rightarrow$) Let $D[\ ] = (\lambda x_1 \ldots x_n.[\ ]D_1 \ldots D_n$ be a $\beta$-solution for $\text{Transl}(A)$. Then $\forall i \in \{1, \ldots, n\} [D_i = U_1^\dagger$ or $D_i = U_2^\dagger$] and $D[L(A)] = U_2^\dagger$. Choosing $* \equiv [x_i := \text{if } D_i = U_1^\dagger \text{ then true else false}]$ we have $A^* = \text{true}$.

($\Leftarrow$) Let $* \equiv A^* = \text{true}$. $\forall i \in \{1, \ldots, n\}$ define $D_i \equiv \text{if } x_i^* = \text{true then } U_1^\dagger \text{ else } U_2^\dagger$. Then $D[\ ] \equiv (\lambda x_1 \ldots x_n.[\ ]D_1 \ldots D_n$ is a $\beta$-solution for $\text{Transl}(A)$. Define $\text{SLNP} = \{\text{Transl}(A) \mid A \in \text{PropForm}\}$. Then the $\beta$-solvability problem for the systems in $\text{SLNP}$ is NP-complete. Note that the systems in $\text{SLNP}$ are not regular.
Appendix C. Proof of 6.2

The proof is divided into several parts (C.0–C.11).

C.0. Definition. (i) Let $\mathcal{S} = (\Gamma, X)$ be a system with equations having form $xM = \lambda a. yQ$, where $x \in X$ and $y \in (\text{FV}(\lambda a. yQ) - X)$. We define

$$\mathcal{S}_\Omega = \{ M[(\text{FV}(M) - (X \cup \{\text{head}(N)\})) := \Omega] = \text{head}(N) | M = N \in \mathcal{S} \}, X).$$

(ii) Let $\mathcal{S} = (\Gamma, X)$ be a system with equations having form $xM = \lambda a. yQ$, where $x \in X$ and $y \in (\text{FV}(\lambda a. yQ) - X)$. $\mathcal{S}$ is said to be $e$-good iff $\mathcal{S}_\Omega$ is $e$-good (see A.2.3).

Assignable sequences of variables (see 6.0) are assignable from inside the $\lambda$-calculus (e.g. as in C.2).

C.1. Proposition. Let $\mathcal{T}$ be a $\lambda$-theory and $\mathcal{S} = (\Gamma_1 \cup \Gamma_2, X)$ be a system s.t.:

H0. Each equation in $\mathcal{S}$ has form $M = \lambda a. y\tilde{Q}$ with $y \in (\text{FV}(\lambda a. y\tilde{Q}) - X)$.

H1. head(right($\Gamma_1$)) = $\{ y_1, ..., y_k \}$ and $\tilde{y} \equiv y_1, ..., y_k$ is assignable in $\mathcal{S}$.

If $\mathcal{S}$ is $\mathcal{T}$-solvable then $\mathcal{G}^* \equiv \{ y := H_yy \tilde{y} \} \in \{ \tilde{y} \}$ and $H_y \in \Lambda^0$. $\mathcal{S}' = \{ M = \lambda a. z\tilde{Q}' | M = \lambda a. y\tilde{Q} \in (\Gamma_1 \cup \Gamma_2), X \}$ is $\mathcal{T}$-solvable (where $\tilde{Q}'$ is obtained from $\tilde{Q}$ replacing some occurrence of $y \in \{ \tilde{y} \}$ by $H_yy\tilde{y}$).

Proof. Let $X = \{ x_1, ..., x_n \}$ and $D[ \ ] = (\lambda x_1 ... x_n [ \ ]) D_{x_1}...D_{x_n}$ be a $\mathcal{T}$-solution for $\mathcal{S}$ with $\forall x \in X \ D_x = \lambda t_1 ... t_{p(x)}. h(t_{x_1})D_{x_1}...D_{x,q(x)} \forall x \in X$ define $G_x = \text{if } p(x) \geq k \text{ then } \lambda t_1 ... t_{p(x)}. h(t_{x_1})D_{x_1}...D_{x,q(x)} \text{ else } \lambda t_1 ... t_{p(x)}. h(t_{x_1})D_{x_1}...D_{x,q(x)} G(x)^{\star} \ }$.

Then $G[ \ ] = (\lambda x_1 ... x_n [ \ ]) D_{x_1}...D_{x_n}$ is a $\mathcal{T}$-solution for $\mathcal{S}$ (because $\tilde{y}$ is assignable in $\mathcal{S}$).

C.2. Example. Let $\mathcal{S} = (\{ y(x\lambda y. b) = y(xy)y, xy(\lambda y. z) = z \}, \{ x \})$ and $G$ be a $\beta$-solution for $\mathcal{S}$. Then $G^* = \lambda y. G(H_yy)$ is a $\beta$-solution for $\mathcal{S}^* = (\{ y(x\lambda y. b) = H_yy(xy), xy(\lambda y. z) = z \}, \{ x \})$.

C.3. Corollary. Let $\mathcal{S} = (\Gamma_1 \cup \Gamma_2, X)$ be a system s.t.:

H0. Each equation in $\Gamma_1$ has form $xM = y\tilde{Q}$ with $x \in X$, $y \notin X$.

H1. $\forall x \in X \ \tilde{M} = y\tilde{Q}, \tilde{N} = y\tilde{P} \in \Gamma_1 \ y \equiv y' \Rightarrow \text{deg}(y\tilde{Q}) = \text{deg}(y'\tilde{P})$.

H2. head(right($\Gamma_1$)) = $\{ y_1, ..., y_k \}$ and $\tilde{y} \equiv y_1, ..., y_k$ is assignable in $\mathcal{S}$.

H3. Each equation in $\Gamma_2$ has form $xM = z$, where $x \in \{ \tilde{x} \}$ and $z \notin \{ \tilde{x}, \tilde{y} \}$.

If $\mathcal{S}$ is $\beta$-solvable then $\text{OK} \_ \text{NEC}(\mathcal{S}_\Omega) = \text{true}$.
Proof. If $\mathcal{S}$ is $\beta$-solvable then by C.1 $\mathcal{S}_\Omega$ is $\beta$-solvable. The thesis follows from 5.2.2. □

In C.4–C.5 we take care of recursive definitions.

C.4. Definition. Let $\mathbb{T}$ be a smn theory and $\mathcal{S} = (\Gamma_1 \cup \Gamma_2, X)$ be a system s.t.:

- Each equation in $\mathcal{S}$ has form $x\bar{M} = \lambda a. y\bar{Q}$ with $y \in (\text{FV}(\lambda a. y\bar{Q}) - X)$ and $x \in X$;
- $\text{head}(\text{right}(\Gamma_1)) = \{y_0, \ldots, y_{k-1}\}$ and $\bar{y} \equiv y_0, \ldots, y_{k-1}$ is assignable in $\mathcal{S}$;
- $D[\ ]$ is a $\mathbb{T}$-solution for $\mathcal{S}$.

0. The variable $u \in X$ is said to be critical for $(x\bar{M} = \lambda a. y_i\bar{Q} \in \Gamma_1, D[\ ])$ at $x \in \text{BT}(x\bar{M})$ iff $D[(x\bar{M})[\alpha^*\langle i \rangle := \lambda t. \Omega]] \in \text{SOL}$ (i.e. the leftmost occurrence of $y_i$ in $D[\alpha a. y_i\bar{Q}]$ comes from the occurrence of $y_i$ at $\alpha^*\langle i \rangle$ in $x\bar{M}$).

1. The variable $u \in X$ is said to be critical for $(x\bar{M} = \lambda a. y_i\bar{Q} \in \Gamma_1, D[\ ])$ iff $\exists \alpha \in \text{BT}(x\bar{M})$ s.t. $u$ is critical for $(x\bar{M} = \lambda a. y_i\bar{Q}, D[\ ])$ at $\alpha$.

2. The variable $u \in X$ is said to be critical for $(\mathcal{S}, D[\ ], y_i)$ iff there exists $x\bar{M} = \lambda a. y_i\bar{Q} \in \Gamma_1$ s.t. $u$ is critical for $(x\bar{M} = \lambda a. y_i\bar{Q}, D[\ ])$.

3. $D[\ ]$ is said to be a singular solution for $\mathcal{S}$ iff $\exists y_i \in \{\bar{y}\} \exists x \in X$ critical for $(\mathcal{S}, D[\ ], y_i)$ s.t. $D_x = \lambda t_1 \ldots t_{p(x)}, t_{h(x)}D_{x_1} \ldots D_{x_{q(x)}}$ and $h(x) \leq k$.

Many specifications containing recursion admit solutions having normal form (e.g. as in C.6). Proposition C.1 and Lemma C.5 give a sharpening and a generalization of [8, 6.3].

C.5. Lemma. Let $\mathbb{T}$ be a smn theory and $\mathcal{S} = (\Gamma_1 \cup \Gamma_2, \{\bar{x}\})$ be a system s.t.:

H0. Each equation in $\mathcal{S}$ has form $x\bar{M} = \lambda a. y\bar{Q}$ with $y \in (\text{FV}(\lambda a. y\bar{Q}) - X)$ and $x \in X$.

H1. $\text{head}(\text{right}(\Gamma_1)) = \{y_1, \ldots, y_k\}$ and $\bar{y} \equiv y_1, \ldots, y_k$ is assignable in $\mathcal{S}$.

0. If $\mathcal{S}$ is $\mathbb{T}$-solvable then $\mathcal{S}' = (\{M = \lambda a. y(\bar{y}; z)\bar{Q}' \mid M = \lambda a. y\bar{Q} \in \Gamma_1\} \cup \{M = \lambda a. z\bar{Q}' \mid M = \lambda a. z\bar{Q} \in \Gamma_2\}, \{\bar{x}\})$ is $\mathbb{T}$-solvable ($\bar{Q}'$ analogous to C.1).

1. If $\mathcal{S}$ has a nonsingular solution with $\text{nf}$ then $\mathcal{S}''$ has a nonsingular solution with $\text{nf}$.

Proof. 0.1. Let $\{\bar{x}\} = \{x_1, \ldots, x_n\}$ and $D[\ ] = (\lambda x_1 \ldots x_n[\ ] \ D_{x_1} \ldots D_{x_n}$ be a $\mathbb{T}$-solution for $\mathcal{S}$ with $\forall x \in \{\bar{x}\} \ D_x = \lambda t_1 \ldots t_{p(x)}, t_{h(x)}D_{x_1} \ldots D_{x_{q(x)}}$.

$\forall u \in X \forall y_i \in \{\bar{y}\}$ define:

$$L(u, y_i) = \text{if } u \text{ is critical for } (\mathcal{S}, D[\ ], y_i) \text{ and } h(u) \leq k \text{ then } 1$$

$$\text{else } t_{h(u)}H_{y_i}, \text{ where } H_{y_i} \text{ is computed as follows:}$$

If $u$ is not critical for $(\mathcal{S}, D[\ ], y_i)$ then $H_{y_i}$ is arbitrary.

Suppose that $u$ is critical for $(\mathcal{S}, D[\ ], y_i)$. Let $x\bar{M} = \lambda a. y_i\bar{Q} \in \Gamma_1$ and $\alpha \in \text{BT}(x\bar{M})$ s.t. $u$ is critical for $(x\bar{M} = \lambda a. y_i\bar{Q}, D[\ ])$ at $\alpha$. Consider the head reduction $\sigma : D[x\bar{M}] \rightarrow_\beta D[((x\bar{M})_a)^*] \bar{A} \equiv E \equiv D[\lambda \bar{b}. u \ldots y_i \ldots ((x\bar{M})_{\langle h(u) - 1 \rangle})^* \ldots] \bar{A} \rightarrow_\beta$
\[(\lambda \delta. D[(((xM)_{x \in (h(u) - 1)})^{**}]) \overrightarrow{\beta} N = \lambda \delta. D[\lambda \alpha. y_1(Q)] (\text{see } [1, 16.2.1]), \text{ where the leftmost occurrence of } y_1 \text{ in } E \text{ comes on the head in } \sigma.\]

This implies \[D[(((xM)_{x \in (h(u) - 1)})^{**}]) \in \text{SOL and head}(D[(((xM)_{x \in (h(u) - 1)})^{**}]) \notin FV(D[(((xM)_{x \in (h(u) - 1)})^{**}])\text{ because the occurrence of } y_1 \text{ that comes on the head in } \sigma \text{ is not in } \hat{A} \text{ nor in } D[(((xM)_{x \in (h(u) - 1)})^{**}]).\]

Hence there exists \(\hat{H} \in A^0\) (easily computable) s.t. \(\forall xM = \lambda \alpha. y_1(Q) \in \Gamma \ \forall x \in BT(xM)\ [\text{if } u \text{ is critical for } (xM - \lambda \alpha. y_1(Q), D[ \ ] ] \text{ at } \tau \text{ then } D[(((xM)_{x \in (h(u) - 1)})^{**}])\hat{H} = I].\)

We choose \(\hat{H}_1, \hat{H}_2\).

\(\forall x \in X \text{ define } (\text{reasoning as in the proof of C.1 w.l.o.g. we can assume } p(x) \geq 0):\)

\[G_x \equiv \lambda u_1 \ldots u_n t_1 \ldots t_{p(x)} \ (\text{if } u \text{ is critical for } xM = \lambda \alpha. y_1(Q) \in \Gamma \ \forall x \in BT(xM)\ [\text{if } u \text{ is critical for } (xM - \lambda \alpha. y_1(Q), D[ \ ] ] \text{ at } \tau \text{ then } D[(((xM)_{x \in (h(u) - 1)})^{**}])\hat{H} = I].\)

\[G_x = \lambda u_1 \ldots u_n t_1 \ldots t_{p(x)} \ (\text{if } u \text{ is critical for } xM = \lambda \alpha. y_1(Q) \in \Gamma \ \forall x \in BT(xM)\ [\text{if } u \text{ is critical for } (xM - \lambda \alpha. y_1(Q), D[ \ ] ] \text{ at } \tau \text{ then } D[(((xM)_{x \in (h(u) - 1)})^{**}])\hat{H} = I].\)

Then \(G[ \ ] \equiv (\lambda x_1 \ldots x_n. [\ ]\ G_{x_1} \ldots G_{x_n} \text{ is a } T\text{-solution for } \mathcal{S}'. \text{ Moreover if } D[ \ ] \text{ has nf and is nonsingular for } \mathcal{T} \text{ then } G[ \ ] \text{ has nf and is nonsingular for } \mathcal{S}' \). \(\Box\)

C.6. Example. Let \(\mathcal{S} = (\{xy_Q(\lambda a. xy_Q) = y(x)y_Q(\lambda a. xy_Q), xy_Q(\lambda a. xy_Q) = z\}, \{x\})\) and \(\mathcal{S}' = (\{xy_Q(\lambda a. xy_Q) = y(x)y_Q(\lambda a. xy_Q), xy_Q(\lambda a. xy_Q) = z\}, \{x\})\). \(G\) as in 6.3 is a \(T\)-solution for \(\mathcal{S}\). Then a possible \(T\)-solution for \(\mathcal{S}'\) is \(L\) as in 6.3.

Using C.5 we can have function symbols on the RHS (i.e. fixed points). Using C.7, C.9 we can also take an argument of a function symbol from the LHS to the RHS (e.g. as in C.10).

C.7. Lemma. Let \(\mathcal{S} = (\Gamma_1 \cup \Gamma_2, \{\vec{x}\}) \) be a system and \(\vec{y} \in (\text{Var} - \{\vec{x}\}) \) s.t.:

H0. Each equation in \(\Gamma_1\) has form \(xM = \vec{y}M\), where \(x \in \{\vec{x}\}\) and \(y \in \{\vec{y}\}\).

H1. head(right(\(\Gamma_1\))) \subseteq \{\vec{y}\} \text{ and } \vec{y} = y_1, \ldots, y_k \text{ is assignable in } \mathcal{S}.

H2. \(\forall xM = \vec{y}M\), \(xN = \vec{y}N \in \Gamma_1[ y = y' \Rightarrow \deg(yM) = \deg(y'N)].\)

H3. Each equation in \(\Gamma_2\) has form \(\vec{x} = \vec{z}\), where \(x \in \{\vec{x}\}\) and \(z \notin \{\vec{x}, \vec{y}\}\).

H4. \(\mathcal{N}_{\mathcal{S}}\) is regular.

Then \(\mathcal{S}\) is \(T\)-solvable iff \(\text{OK}_\text{NET}(\mathcal{S}_{\mathcal{S}}) = \text{true}\).

Proof. \((\Leftarrow)\) Follows from C.3.

\((\Rightarrow)\) Let \(\vec{y} \equiv [\{\vec{y}\} := \vec{y}], (\Gamma_1)^1 = \{x\vec{y}M_1 \ldots M_m = y\vec{y}M_1 \ldots M_m \mid x\vec{y}M_1 \ldots M_m \in \Gamma_1\}, (\Gamma_2)^1 = \{(x\vec{y})^\circ = z \mid x\vec{y} = z \in \Gamma_2\}, \mathcal{P} = ((\Gamma_1)^1 \cup (\Gamma_2)^1, \{\vec{x}\}).\)

Since \(\mathcal{S}\) is regular we have \(\mathcal{P}\) is regular and, by C.0, \(\mathcal{P}_0\) is regular. Since (A.1.0) there exists \(\mathcal{P}_0^+\) canonical (\(\mathcal{P}_0\)) s.t. \(\mathcal{P}_0^+\) is PFR and LR-distinct there exists also \(\mathcal{P}_0^\circ\) canonical (\(\mathcal{P}_0\)) s.t. \(\mathcal{P}_0^\circ\) is PFR and LR-distinct. Hence, by A.1.0, 5.3, \(\text{OK}_\text{SUFF}(\mathcal{P}_0)\).

Moreover, by A.1.1, there exists \(\mathcal{P}_0^+ = (\Gamma^+, \{\vec{x}\})\) canonical (\(\mathcal{P}_0\)) s.t. \(\text{OK}_\text{SUFF}(\mathcal{P}_0^+)\). Thus, by A.4.3.2, \(\mathcal{P}_0^+\) is \(T\)-solvable.

Let \(\Gamma_1^+ = \{x\vec{y} = y \mid x\vec{y} = y \in \mathcal{P}_0^+\} \) and \(y \in \{\vec{y}\}\) and \(\Gamma_2^+ = \Gamma^+ - \Gamma_1^+\). Let \(d(\mathcal{P}, x) = \text{Card}(\{\deg(xN) \mid xN = yN \in \mathcal{P}\})\) and \(d^*(\mathcal{P}) = \max\{d(\mathcal{P}, x) \mid x \in \{\vec{x}\}\}\). By induction on \(d^*(\mathcal{P})\) we construct a solution for \(\mathcal{P}\) and hence for \(\mathcal{S}\).
Case 0: \(d^*(\mathcal{P}) = 1\). Let \(\{x\} = \{x_1, \ldots, x_n\}\) and \(D[.] = (\lambda x_1 \ldots x_n[.])D_{x_1} \ldots D_{x_n}\) be a \(\beta\)-solution for \(\mathcal{P}_Q\) (by A.4.3.2) with \(\forall x \in \{x\}\) \(D_x = \lambda t_1 \ldots t_{p(x)}. t_{h(x)}D_{x_1} \ldots D_{x,q(x)}\). For all \(x \in \{x\}\) define \(G_x = \lambda t_1 \ldots t_{p(x)}. ((D_{x_1}t_1 \ldots D_{x_k}t_k)1 = t_1 \ldots t_{p(x)}1 i = 1, \ldots, k).\) Then \(G[.] = (\lambda x_1 \ldots x_n[.]) G_{x_1} \ldots G_{x_n}\) is a \(\beta\)-solution for \(\mathcal{P}\).

Case 1: \(d^*(\mathcal{P}) > 1\). \(\forall x \in \{x\}\) define

\[
m(x) = \min \{\deg(x\bar{N}) | x\bar{N} = y\bar{N} \in \mathcal{P}\},
\]

\[
v \equiv v_1, \ldots, v_n \text{ (fresh)}, \quad u \equiv u_1, \ldots, u_n \text{ (fresh)}, \quad \ast \equiv [x := x\bar{v}] x \in \{x\},
\]

\[
\Gamma_1^+(x) = \{x\bar{v} M_1^* \ldots M_{m(x)}^* = y\bar{v} M_1^* \ldots M_{m(x)}^* | x M_1 \ldots x M_n = y \in \Gamma_1^+\},
\]

\[
\Gamma_1^-(x) = \{x\bar{u} M_1^* \ldots M_{m(x)}^* = v\bar{u} M_1^* \ldots M_{m(x)}^* | x\bar{M}_1 \ldots x\bar{M}_n = v \in \Gamma_1^+\}
\]

\[
\exists k > 0 \{x M_1 \ldots x M_n+k = y \in \Gamma_1^+\},
\]

\[
\Gamma_2^+(x) = \{u\bar{v} M_1^* \ldots M_n^* = y\bar{v} M_1^* \ldots M_n^* | x M_1 \ldots x M_n = y \in \Gamma_2^+\}.
\]

Let \(Q = ([\cup \{\Gamma_1^+(x) \cup \Gamma_1^-(x) \cup \Gamma_2^+(x) | x \in \{x\}\}] \cup (\Gamma_2^+)^*, \{x, u\})\). \(Q\) is regular and \(d^*(Q) < d^*(\mathcal{P})\), hence by induction hypothesis there exists a \(\beta\)-solution \(D[.] = (\lambda x_1 \ldots x_n[.])D_{x_1} \ldots D_{x_n}G_1 \ldots G_n\) for \(Q\). Then \(G[.] = (\lambda x_1 \ldots x_n[.]) (D_{x_1}G_1 \ldots D_{x_n}G_n) = (D_{x_1}G_1 \ldots D_{x_n}G_n)\) is a \(\beta\)-solution for \(\mathcal{P}'' = ([x\bar{M} = yG_1 \ldots y\bar{M} | x\bar{M} = y\bar{M} \in (\Gamma_1^+)^1] \cup (\Gamma_2^+)^1, \{x\})\).

The thesis follows erasing \(G_1, \ldots, G_n\) using C.1. \(\Box\)

C.8. Example. Let \(\mathcal{S} = (\{xy\bar{Q}(\lambda a. xy\bar{Q}) = y\bar{y}\bar{Q}(\lambda a. xy\bar{Q}), xy\bar{Q}(\lambda ab. z) = z, \{x\}\})\). By C.7 \(\mathcal{S}\) is \(\beta\)-solvable. A \(\beta\)-solution is \(G = \lambda y_1t_1t_2.((Dyt_1t_2) [y := yyt_1t_2])\), where \(D\) is as in 5.5.

C.9. Lemma. Let \(\mathcal{S} = (\Gamma_1 \cup \Gamma_2, \{x\})\) be a system and \(\bar{y} \in (\text{Var} - \{\bar{x}\})\) s.t:

H0. \(\bar{y} \equiv y_1, \ldots, y_k\) is assignable in \(\mathcal{S}\).

H1. Each equation in \(\Gamma_1\) has form \(x\bar{M} = y(x\bar{y})\bar{M}z\), where \(x \in \{x\}\), \(y \in \{\bar{y}\}\), \(z \notin \{x, \bar{y}\}\) and the variables in head(right(\(\Gamma_1\))) are pairwise distinct.

H2. Each equation in \(\Gamma_2\) has form \(x\bar{M} = z\), where \(x \in \{x\}\) and \(z \notin \{x, \bar{y}\}\).

H3. \(\mathcal{S}^* = (\{x\bar{M} = y(x\bar{y})\bar{M}z \in \Gamma_1\} \cup \{u_\bar{y} \bar{M} = z | x\bar{M} = y(x\bar{y})\bar{M}z \in \Gamma_1\} \cup z \in \{z\} \cup u_\bar{y} \in \{\bar{u}\} \cup \Gamma_2, \{\bar{x}, \bar{y}\})\) is regular, where \(\bar{u}\) is a sequence of fresh variable s.t \(\{\bar{u}\} = \{u_\bar{y} \bar{M} = z(x\bar{y})\bar{M}z \in \Gamma_1\} \cup z \in \{z\}\).

0. \(\mathcal{S}\) is \(\beta\)-solvable iff \(\text{OK}_{\text{NEC}}(\mathcal{S}^*) = \text{true}\).

1. If \(\mathcal{S}\) is \(\beta\)-solvable and \(\text{Card}(\Gamma_1 \cup \Gamma_2) > 1\) then \(\mathcal{S}\) has a \(\beta\)-solution having normal form.

Proof. 0. (\(\Rightarrow\)) If \(\mathcal{S}\) is \(\beta\)-solvable then, by C.1 \(\mathcal{S}_1 = ([x\bar{M} = y | x\bar{M} = y(x\bar{y})\bar{M}z \in \Gamma_1])\), \(\{x\}\) is \(\beta\)-solvable, hence by 5.2.2 there exists \(\mathcal{S}_2^* \in \text{canonical}(\mathcal{S}_1)\) s.t. \(\mathcal{S}_2^*\) is PFR and LR-distinct. By C.1 \(\forall x\bar{M} = y(x\bar{y})\bar{M}z \in \Gamma_1 \forall z \in \{z\}\) the equation \(\mathcal{S}_{yz} = ([x\bar{M} = z],\).
The thesis follows from definition A.1.0.

(C) By Definition 5.2.4 $\mathcal{S}^*$ is $\beta$-solvable. Hence by C.7, C.5 the system (6) analogous to $\mathcal{Q}$ in C.1) $\mathcal{Q}' = \{\mathcal{Q}\}$ is $\beta$-solvable. Let $a = a_1, \ldots, a_k$. By C.1 there is a $\beta$-solution $G'$ for $\mathcal{Q}' = \{\mathcal{Q}\}$. Then $G'$ is a $\beta$-solution for $\mathcal{S}$.

1. Since $\mathcal{S}^*$ is regular by A.2.3.2 and A.1.4 the solution constructed in 0 is nonsingular for $\mathcal{S}$. The thesis follows from C.5.

C.10. Example. (a) Let $\mathcal{S} = \{\mathcal{Q}\} = \{\mathcal{Q}\}$. By C.9 $\mathcal{S}$ is $\beta$-solvable. A $\beta$-solution is $H$ as in 6.3.

(b) Let $\mathcal{S} = \{\mathcal{Q}\}$. By C.9 $\mathcal{S}$ is $\beta$-solvable. A $\beta$-solution is $H$ as in 6.3.


References


