ELSEVIER

Theoretical Computer Science 160 (1996) 145–184

Theoretical Computer Science

Fundamental Study

Equational programming in λ-calculus via SL-systems. Part 1☆

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

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

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

We show that the β ($\beta\eta$)-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 β-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 λ-calculus; λ-calculus; Equational programming; Functional programming; Automated synthesis of programs

☆This paper is a revised version of [15; 16, Part 1].

*Email: tronci@smaq20.univaq.it.

[1] This research has been partially supported by grant 203.01.50 from CNR, Italy.

SSDI 0304-3975(95)00105-0

Contents

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. $f(0) \to a$,

0.0.3. $f(n+1) \to H(f(n), n)$.

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 λ-calculus since it is a formal language whose expressions (λ-terms) can be interpreted as programs.

Equations 0.0.0, 1 can be easily written as a system of equations in λ-calculus.

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

0.1.0. $F\underline{0} = \underline{a}$,

0.1.1. $F(\underline{s}\,\underline{n}) = \underline{H}(F\,\underline{n})\underline{n}$.

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 λ-calculus.

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

0.2.0. $\quad L\underline{0} = y_0,$

0.2.1. $\quad L(\underline{s}\underline{n}) = y_1(L\underline{n})\underline{n},$

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

A solution for 0.2.0, 1 is a λ-term L (containing y_0 and y_1) satisfying 0.2.0, 1. Any solution L for 0.2.0, 1 yields a representation $F \equiv L[y_0 := \underline{a}, y_1 := \underline{H}]$ (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 λ-terms. Moreover $\underline{n}$ can also be replaced by a free variable. Thus, equations 0.2.0, 1 become:

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

0.2.2. $\quad Dy_0y_1\underline{0} = y_0,$

0.2.3. $\quad Dy_0y_1(\underline{s}z) = y_1(Dy_0y_1z)z.$

The program specified by 0.0.2, 3 can be represented with the λ-term $G = D\underline{a}\underline{H}$ $(= F)$. In fact we have $G\underline{0} = D\underline{a}\underline{H}\underline{0} = \underline{a}$ and $G(\underline{s}z) = D\underline{a}\underline{H}(\underline{s}z) = \underline{H}(D\underline{a}\underline{H}z)z = \underline{H}(Gz)z$. 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. $\underline{a}$, $\underline{H}$ 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 $f:\mathbb{N}\to\mathbb{N}$ can be specified with the rewrite rules ($a\in\mathbb{N}$ and $H:(\mathbb{N}\to\mathbb{N})\times\mathbb{N}\to\mathbb{N}$):

0.4.0. $f(0)\to a$,

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

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. $Qy_0y_1\underline{0}=y_0$,

0.4.3. $Qy_0y_1(\underline{s}z)=y_1(Qy_0y_1)z$.

Thus, reasoning as in 0.2, any solution Q for 0.4.2, 3 yields a representation $G'\equiv Q\underline{a}\underline{H}$ 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 then 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 $D\equiv\lambda y_0y_1.\,Qy_0(\lambda fz.\,y_1\,(fz)z)$.

Since $G'\equiv(Qy_0y_1)[y_0:=\underline{a},\,y_1:=\underline{H}]$ by abuse of language we will also call Qy_0y_1 a representation for the program 0.4.0, 1.

According to the previous discussion a system $\mathcal{S}$ of equations in the λ-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 λ-calculus. This would not be possible in a term rewriting system without some abstraction mechanism. Moreover, specifying sets of λ-terms with equations does not leave out any interesting set, to be precise: any recursively enumerable and β-closed set of closed λ-terms is the set of solutions to a combinator equation [14]. These features make λ-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 $(\underline{0}, \underline{s}, \underline{p}, \underline{\text{Zero}})$ (as in [2]), where (the λ-terms $\mathbf{I}$, $\mathbf{U}_i^n$ are defined in 2.0):

$$\underline{0} = \lambda ab.b, \quad \underline{s} \equiv \lambda ab.bab, \quad \underline{p} \equiv \lambda b.b\mathbf{U}_1^3\mathbf{U}_2^2, \quad \underline{\text{Zero}} \equiv \lambda b.b(\mathbf{U}_1^4\mathbf{U}_2^2)\mathbf{U}_1^2.$$

Find a λ-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 $D \in \Lambda^0$ s.t. ($\vec{y} \equiv y_0, y_1$):

0.5.0. $\quad D\vec{y}\underline{0} = y_0(D\vec{y})\underline{0}$ (constraint 0).

0.5.1. $\quad D\vec{y}(\underline{s}z) = y_1(D\vec{y})z$,

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

A possible β-solution is

$$D \equiv GG = \lambda\vec{y}t.t\,(\lambda a_1a_2a_3.\ y_1(t(\mathbf{U}_1^3\mathbf{I})GG\vec{y})a_1)(y_0(t\mathbf{I}\mathbf{I}GG\vec{y})t),$$

where

$$G \equiv \lambda u\vec{y}t.t(\lambda a_1a_2a_3.y_1(t(\mathbf{U}_1^3\mathbf{I})uu\vec{y})a_1)(y_0(t\mathbf{I}\mathbf{I}GG\vec{y})t).$$

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 λ-term F and a numerical system $(\underline{0}, \underline{s}, \underline{p}, \underline{\text{Zero}})$ 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 $D, D_{\underline{0}}, D_{\underline{s}}, D_{\underline{p}}, D_{\underline{\text{Zero}}} \in \Lambda^0$ s.t. ($\vec{y} \equiv y_0, y_1, y_2, y_3, y_4, y_5$):

0.6.0. $D_{\underline{p}}\vec{y}(D_{\underline{0}}\vec{y}) = (D_{\underline{0}}\vec{y})$ (equations 0.6.0–3 specify the data structure),

0.6.1. $D_{\underline{p}}\vec{y}(D_{\underline{s}}\vec{y}z) = z$ ($\underline{p} \equiv D_{\underline{p}}\vec{y}$ is a left inverse for $\underline{s} \equiv D_{\underline{s}}\vec{y}$),

0.6.2. $D_{\underline{\text{Zero}}}\vec{y}(D_{\underline{0}}\vec{y}) = y_2$ ($\underline{\text{Zero}} \equiv D_{\underline{\text{Zero}}}\vec{y}$ recognizes the constructors (i.e. $\underline{0} \equiv D_{\underline{0}}\vec{y}, \underline{s}$)),

0.6.3. $D_{\underline{\text{Zero}}}\vec{y}(D_{\underline{s}}\vec{y}z) = y_3$,

0.6.4. $D\vec{y}(D_{\underline{0}}\vec{y}) = y_0(D\vec{y})(D_{\underline{0}}\vec{y})(D_{\underline{s}}\vec{y})(D_{\underline{p}}\vec{y})(D_{\underline{\text{Zero}}}\vec{y})$ (constraint 0),

0.6.5. $D\vec{y}(D_{\underline{s}}\vec{y}z) = y_1(D\vec{y})(D_{\underline{0}}\vec{y})(D_{\underline{s}}\vec{y})(D_{\underline{p}}\vec{y})(D_{\underline{\text{Zero}}}\vec{y})z$.

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

0.6.7. $D_{\underline{0}}\vec{y}(D_{\underline{s}}\vec{y})(D_{\underline{0}}\vec{y}) = y_4(D\vec{y})(D_{\underline{0}}\vec{y})(D_{\underline{s}}\vec{y})(D_{\underline{p}}\vec{y})(D_{\underline{\text{Zero}}}\vec{y})$ (constraint 2),

0.6.8. $D_{\underline{s}}\vec{y}z(D_{\underline{s}}\vec{y})(D_{\underline{0}}\vec{y}) = y_5(D\vec{y})(D_{\underline{0}}\vec{y})(D_{\underline{s}}\vec{y})(D_{\underline{p}}\vec{y})(D_{\underline{\text{Zero}}}\vec{y})z$.

The system 0.6.0–0.6.8 has a β-solution. Defining $F \equiv D\vec{y}$, $\underline{0} \equiv D_{\underline{0}}\vec{y}$, $\underline{s} \equiv D_{\underline{s}}\vec{y}$, $\underline{p} = D_{\underline{p}}\vec{y}$, $\underline{\text{Zero}} = D_{\underline{\text{Zero}}}\vec{y}$, we solve our problem. With this specification of the data structure we always get a *one shot predecessor* $\underline{p}$ (i.e. the length of the computation of $\underline{p}\ \underline{n}$ is constant since $\underline{s}$ is left-invertible in Λ). Consider the equation

0.6.9. $D\vec{y}\mathbf{U}_2^2 = \mathbf{U}_1^2$.

The system 0.6.0–0.6.9 is still β-solvable, however the system 0.5.0–0.5.2, 0.6.9 (with $\vec{y}$ as in 0.6) is no longer β-solvable because there is a conflict between 0.5.0 and 0.6.9 since we chose $\underline{0} = \mathbf{U}_2^2$ 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 ($\underline{0}, \underline{s}, \underline{p}, \underline{\text{Zero}}$) for the natural numbers s.t.:

0. the application of two natural numbers realizes an arbitrarily given partial recursive function;

1. the λ-term representing the successor has form $\lambda ab.bL_1L_2$.

We look for $D_{\underline{0}}, D_{\underline{s}}, D_{\underline{p}}, D_{\underline{\text{Zero}}} \in \Lambda^0$ s.t. ($\vec{y} \equiv y_0, y_1, y_2, y_3, y_4, y_5$):

0.7.0. $D_{\underline{p}}\vec{y}(D_{\underline{0}}\vec{y}) = D_{\underline{0}}\vec{y}$ (specification of the data structure (as in 0.6.0–3)),

0.7.1. $D_{\underline{p}}\vec{y}(D_{\underline{s}}\vec{y}z) = z$,

0.7.2. $D_{\underline{\text{Zero}}}\vec{y}(D_{\underline{0}}\vec{y}) = y_4$,

0.7.3. $D_{\underline{\text{Zero}}}\vec{y}(D_{\underline{s}}\vec{y}z) = y_5$,

0.7.4. $D_{\underline{0}}\vec{y}(D_{\underline{0}}\vec{y}) = y_0(D_{\underline{0}}\vec{y})(D_{\underline{s}}\vec{y})(D_{\underline{p}}\vec{y})(D_{\underline{\text{Zero}}}\vec{y})$ (constraint 0),

0.7.5. $D_{\underline{0}}\vec{y}(D_{\underline{s}}\vec{y}z) = y_1(D_{\underline{0}}\vec{y})(D_{\underline{s}}\vec{y})(D_{\underline{p}}\vec{y})(D_{\underline{\text{Zero}}}\vec{y})z$,

0.7.6. $D_{\underline{s}}\vec{y}z(D_{\underline{0}}\vec{y}) = y_2(D_{\underline{0}}\vec{y})(D_{\underline{s}}\vec{y})(D_{\underline{p}}\vec{y})(D_{\underline{\text{Zero}}}\vec{y})z$,

0.7.7. $D_{\underline{s}}\vec{y}u(D_{\underline{s}}\vec{y}z) = y_3(D_{\underline{0}}\vec{y})(D_{\underline{s}}\vec{y})(D_{\underline{p}}\vec{y})(D_{\underline{\text{Zero}}}\vec{y})uz$,

0.7.8. $D_{\underline{s}}\vec{y}\Omega(\lambda ab.z) = z$ (constraint 1).

Choosing $\underline{0} \equiv D_{\underline{0}}\vec{y}$, $\underline{s} \equiv D_{\underline{s}}\vec{y}$, $\underline{p} \equiv D_{\underline{p}}\vec{y}$, $\underline{\text{Zero}} = D_{\underline{\text{Zero}}}\vec{y}$ yields the wanted representation for the natural numbers.

A possible β-solution for this system is ($\vec{v} \equiv v_1, v_2, v_3, v_4$) (the terms $\mathbf{P}_q$ are defined in 2.0):

$$D_{\underline{0}} \equiv G_0^* G_0^* G_1^* D_{\underline{p}} D_{\underline{\text{Zero}}}, \quad D_{\underline{s}} = G_1^* G_0^* G_1^* D_{\underline{p}} D_{\underline{\text{Zero}}}, \quad D_{\underline{p}} \equiv \lambda\vec{y}t.t\mathbf{P}_2\mathbf{U}_2^4\mathbf{U}_3^3 t,$$

$$D_{\underline{\text{Zero}}} \equiv \lambda\vec{y}t.t\mathbf{P}_2(\lambda a_1 \dots a_4.y_5)(\lambda a_1 a_2 a_3.y_4)t,$$

where

$$G_0^* \equiv \lambda\vec{v}\vec{y}t.((G_0\vec{y}t)[y_0 := y_0(t(\mathbf{U}_1^2\mathbf{I})v_1\vec{v}\vec{y})(t(\mathbf{U}_1^2\mathbf{I})v_2\vec{v}\vec{y})(v_3\vec{y})(v_4\vec{y}),$$
$$y_1 := y_1(t(\mathbf{U}_1^3\mathbf{I})v_1\vec{v}\vec{y})(t(\mathbf{U}_1^3\mathbf{I})v_2\vec{v}\vec{y})(v_3\vec{y})(v_4\vec{y})]),$$

$$G_1^* \equiv \lambda\vec{v}\vec{y}t_1t_2.((G_1\vec{y}t_1t_2)[y_2 := y_2(t_2(\mathbf{U}_1^2\mathbf{I})v_1\vec{v}\vec{y})(t_2(\mathbf{U}_1^2\mathbf{I})v_2\vec{v}\vec{y})(v_3\vec{y})(v_4\vec{y}),$$
$$y_3 := y_3(t_2(\mathbf{U}_1^3\mathbf{I})v_1\vec{v}\vec{y})(t_2(\mathbf{U}_1^3\mathbf{I})v_2\vec{v}\vec{y})(v_3\vec{y})(v_4\vec{y})]),$$

$$G_0 \equiv \lambda\vec{y}t.t(t\mathbf{P}_2(\lambda a_1 \dots a_6.y_1a_2)(\lambda a_1 \dots a_4.y_0)t),$$

$$G_1 \equiv \lambda\vec{y}t_1t_2.t_2(t_2\mathbf{P}_2(\lambda a_1 \dots a_7.y_3a_7a_6)(\lambda a_1 \dots a_4.y_2)t_2)t_1.$$

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 *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 ($H_0, H_1 \in$ Boole):

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

0.8.1. $g(\text{false}) \to H_1$;

0.8.2. $g(H_1) \to H_0$,

0.8.3. $g(\text{false}) \to H_1$.

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 ($\underline{\text{false}} = \mathbf{U}_1^2$ and $\underline{\text{true}} = \mathbf{U}_2^2$)

0.8.4. $Gy_0y_1(Gy_0y_1\underline{\text{false}}) = y_0$,

0.8.5. $Gy_0y_1\underline{\text{false}} = y_1$,

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 $Gy_0y_1y_1 = y_0$ and $Gy_0y_1\underline{\text{false}} = y_1$. Thus $Gy_0\underline{\text{false}}\,\underline{\text{false}} = \underline{\text{false}}$. 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 β-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. $Gy_0y_1\underline{H}_1 = y_0$,

0.8.7. $Gy_0y_1\,\underline{\text{false}} = y_1$.

The system 0.8.6, 7 has a β-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 β-solutions.

More in general, the β-solvability of the system of equations $\mathscr{S}$ 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 β ($\beta\eta$)-left-inverse for a finite set $\mathfrak{F} = \{M_1, \ldots, M_m\}$ of closed λ-terms (combinators) can also be formulated as the β-solvability problem for systems like those in 0.5–0.7 (namely find a combinator L s.t. $\forall i\,\{1, \ldots, n\}$ $L(M_i z) = z$). It is known [4, 10] that the problem of finding a β-left-inverse for a combinator is decidable. However, here we show (4.1) that the problem of finding a common β ($\beta\eta$)-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 β ($\beta\eta$)-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 $\lambda\vec{y}t.tL_1 \ldots L_m$. Moreover, m must be equal to the number of abstractions in $\lambda ab.z$. Thus, $m = 2$ and $D = \lambda\vec{y}t.tL_1L_2$. The smaller m the more cleverness we need to find a solution. Since, in general, too much cleverness is needed, the β ($\beta\eta$)-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 β-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 λ-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) β-solvability for SL-systems (i.e. find executable codes, if any, satisfying a given set of SL-specifications); (4.3) β-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 β-solution; to construct a β-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 β-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 λ-calculus

We review a few definitions about the λ-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 Λ, the symbol $\equiv$ denotes syntactic equality; $\vec{M} \equiv M_1, \ldots, M_n$; $|\vec{M}| = n$; $\{\vec{M}\} = \{M_1, \ldots, M_n\}$.

2.0. Example. The following are λ-terms: $\mathbf{U}_i^n \equiv \lambda x_1 \ldots x_n . x_i$ $(1 \leqslant i \leqslant n)$, $\mathbf{I} \equiv \mathbf{U}_1^1$, $\mathbf{K} \equiv \mathbf{U}_1^2$, $\omega \equiv \lambda x . xx$, $\Omega \equiv \omega\omega$ (Ω represents a nonterminating program), $\mathbf{P}_q \equiv \lambda x_1 \ldots x_q x_{q+1} . x_{q+1} x_1 \ldots x_q$, $\langle M_1, \ldots, M_n \rangle \equiv \mathbf{P}_n M_1 \ldots M_n$.

A term M is said to be λ-free if $M \equiv y\vec{M}$. $\mathfrak{F} \subseteq \Lambda$ is said to be λ-free if its elements are λ-free. $\Lambda[\ ]$ ($\Lambda^0[\ ]$) is the set of contexts on Λ (with no free variables) [1, 2.1.18].

If $Z \subseteq$ Var then $M[Z := N]$ denotes the result of substituting N for all the (free) occurrences of each $z \in Z$ in M. If $z \in$ Var we write $M[z := N]$ for $M[\{z\} := N]$ and $(M = N)\,[Z := Q]$ for $M[Z := Q] = N[Z := Q]$. Form $(\Lambda) = \{M = N \in \Lambda \mid M, N \in \Lambda\}$. The elements of Form (Λ) are called formulas of Λ. A theory $\mathbb{T}$ is a set of formulas of Λ. If $M = N \in \mathbb{T}$ we write $M =_{\mathbb{T}} N$ and we say that $M = N$ is a theorem in $\mathbb{T}$. We write $M =_{(\eta)} N$ for $M = N \in \boldsymbol{\lambda}$ $(\boldsymbol{\lambda\eta})$ [1 , 2.1.4, 28] (it will be clear from the mathematical context if $M = N$ is a formula or a theorem of $\boldsymbol{\lambda}$).

A λ-term M is said to be solvable iff there exists a λ-term $\lambda\vec{a}.\, y\,\vec{Q}$ s.t. $M = \lambda\vec{a}.\, y\vec{Q}$. SOL is the set of solvable λ-terms [1, 2.2.10–12]. We say that M has nf if M has β-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 (Γ, X) where $\Gamma \subseteq$ Form(Λ) and X is a finite subset of Var. Unless otherwise stated we will assume that Γ is finite.

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

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

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

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

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

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

Of course it is a matter of taste which words to use. In the following we will usually speak about systems of equations and β-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 λ-terms having a finite Böhm-tree. Moreover, as usual, we assume of having an oracle that given a λ-term M computes $\mathrm{BT}(M)$ (or, equivalently, we assume that λ-terms are given in $\beta\Omega$-normal form). Moreover, when dealing with complexity issues we assume that any label in $\mathrm{BT}(M)$ can be represented in a cell of memory.

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

- $\mathrm{Node}(\mathfrak{F}) = \max\{\text{number of nodes in } \mathrm{BT}(M) \mid M \in \mathfrak{F}\}$.
- $\mathrm{Size}(\mathfrak{F}) = \mathrm{Card}(\mathfrak{F})\,\mathrm{Nodes}(\mathfrak{F})$.
- $\mathrm{Size}(\mathscr{S}) = \mathrm{Size}(\{M \mid \exists M = N \in \mathscr{S}\}) + \mathrm{Size}(\{N \mid \exists M = N \in \mathscr{S}\}) + \mathrm{Card}(X)$.

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

3. SL-systems

Many interesting systems (e.g. those in 0.5–0.7) can be transformed into systems with equations having form $x\vec{M} = z$. 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 $\mathscr{S} = (\Gamma, X)$ is said to be an SL-system (separation like) if its equations have form $x\vec{M} = z$, where $x \in X$ and $z \notin X$.

(ii) An SL-system $\mathscr{S} = (\Gamma, \{x\})$ is said to be an HSL-system (head separation like) if its equations have form $x\vec{M} = z$, where $x \notin \mathrm{FV}(\vec{M})$. 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 β-solvability problems for particular HSL-systems and SL-systems.

Let $\mathfrak{F} = \{M_1, \ldots, M_m\} \subset \Lambda$ and $\vec{y} \equiv y_1 \ldots y_m$ be fresh variables.

(a) $\mathfrak{F}$ is said to be separable iff the HSL-system $\mathscr{S} = (\{x\vec{y}M_1 = y_1, \ldots, x\vec{y}M_m = y_m\}, \{x\})$ is β-solvable.

(b) Let $\mathfrak{F}$ be λ-free and $X = \{x_1, \ldots, x_n\}$ ($(X \cap \mathrm{FV}(\mathfrak{F}))$ may be nonempty). $\mathfrak{F}$ is said to be X-separable iff the SL-system $\mathscr{S} = (\{(\lambda x_1 \ldots x_n. M_i)\ (x_1\vec{y}) \ldots (x_n\vec{y}) = y_1 \mid i = 1, \ldots, m\}, X)$ is β-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 $\mathscr{S} = (\{x(\lambda a.\, a\mathbf{U}_1^2(az\Omega\Omega)) = z, x(\lambda a.\, a\mathbf{U}_2^2(a\Omega z\Omega)) = z,\ x(\lambda a.\, z) = z\},\ \{x\})$.

A possible β-solution for $\mathscr{S}$ is $D \equiv \lambda t.\, t(\lambda ab.\, t\mathbf{P}_3\Omega\langle\lambda uv.\, u\mathbf{U}_1^3, \lambda uv.\, u\mathbf{U}_2^3\rangle)$, but there is no β-solution for $\mathscr{S}$ having form

3.2.0. $\quad \lambda t_1 \ldots t_e \ldots t_p.\, t_e H_1 \ldots H_q t_{i_1} \ldots t_{i_r},$

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

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

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

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

$$L(\lambda a.\, a\mathbf{U}_1^2(az\Omega\Omega)) = H\mathbf{U}_1^2(Hz\Omega\Omega),$$

$$L(\lambda a.\, a\mathbf{U}_2^2(a\Omega z\Omega)) = H\mathbf{U}_2^2(H\Omega z\Omega).$$

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

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

$H = \lambda t_1 t_2.\, t_2\vec{Q}$. Then $(Hz\Omega\Omega) \notin \mathrm{SOL}$. Absurd.

Case 1: $L = \lambda t.\, tt$. Then $L\ (\lambda a.\, a\mathbf{U}_1^2\ (az\Omega\Omega)) \neq z$. Absurd. Thus, the thesis follows.

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

As a matter of fact HSL-systems are so powerful that their β ($\beta\eta$)-solvability problem is, in general, undecidable (4.1). However, the β-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 β ($\beta\eta$)-solvability problem for HSL-systems (3.0) is undecidable. Moreover, we show that the β-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 $P_k \equiv \lambda x.\ G_k x \lceil k \rceil \mathbf{K\,I\,I}$, where $G_k \equiv \lambda t.\ F_k^{\#}$, $F_k^{\#}$ is obtained from F_k replacing each redex $(\lambda a.\ P)Q$ in F_k by $t(\lambda a.\ P)Q$ (hence $F_k^{\#}$ is in nf), F_k λ-defines $\{k\}$ (the kth partial recursive function) as in [1, 8.4]. Note that P_k has nf. We have $P_k\ \mathbf{I} =_\beta$ **if** $\{k\}(k)\downarrow$ **then I else** unsolvable.

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

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

Proof. ($\Leftarrow$) $D \equiv \lambda t.\ t\mathbf{I}$ is a β-solution for $\mathcal{S}$.

($\Rightarrow$) Let $D[\ ]$ be a $\beta\eta$-solution for $\mathcal{S}$. Let σ_1 be the standard reduction $D[x(\lambda a.\ az)] \twoheadrightarrow_\beta F \equiv \lambda t_1 \dots t_r.\ (\lambda a.\ az)H_{1,1} \dots H_{1,h} \twoheadrightarrow_\beta Q \equiv \lambda t_1 \dots t_q.\ zQ_{1,1} \dots Q_{1,q} \twoheadrightarrow_{\beta\eta} z$, where F is the first term in the reduction σ_1 in which $(\lambda a.\ az)$ is on the head and the leftmost occurrence of z in F will come on the head in σ_1 and Q is the first term in σ_1 where z is on the head. Let $* \equiv [\{z, u, y\} := \Omega]$.

We have $\forall i \in \{1, \dots, q\}$ $[Q_{1,i} \twoheadrightarrow_{\beta\eta} t_i$ and $Q_{1,i}^* \twoheadrightarrow_{\beta\eta} t_i]$ and

$$\lambda t_1 \dots t_r.(\lambda a.\ az)H_{1,1}^* \dots H_{1,h}^* \twoheadrightarrow_\beta \lambda t_1 \dots t_r.\ H_{1,1}^* z H_{1,2}^* \dots H_{1,h}^*$$

$$\twoheadrightarrow_\beta \lambda t_1 \dots t_q.\ zG_{1,1} \dots G_{1,q} \twoheadrightarrow_{\beta\eta} z.$$

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

$$D[x(\lambda a.\ a(au))] \twoheadrightarrow_\beta \lambda t_1 \dots t_r.\ (\lambda a.\ a(au))H_{2,1} \dots H_{2,h}$$

$$\twoheadrightarrow_\beta \lambda t_1 \dots t_r.\ H_{2,1}(H_{2,1}u)H_{2,2} \dots H_{2,h} \twoheadrightarrow_\beta \lambda t_1 \dots t_q.\ H_{2,1}uQ_{2,1} \dots Q_{2,q} \twoheadrightarrow_{\beta\eta} u,$$

where $\forall i \in \{1, \dots, h\}\ \exists L_i \in \Lambda^0\ [H_{1,i} = L_i(\lambda a.\ az)$ and $H_{2,i} = L_i(\lambda a.\ a(au))]$ and $\forall i \in \{1, \dots, q\}\ \exists V_i \in \Lambda^0 [Q_{1,i} = V_i(\lambda a.\ az)$ and $Q_{2,i} = V_i(\lambda a.\ a(au))]$.
Hence, $\forall i \in \{1, \dots, q\}\ [Q_{1,i}^* \sqsubseteq Q_{2,i}]$. This implies $Q_{2,i} =_\eta t_i$. Hence, $H_{2,1}u =_\eta u$. Suppose $H_{2,1}^* u \notin \mathrm{SOL}$, then $H_{1,1}^* z \notin \mathrm{SOL}$. This is absurd. Hence, $H_{2,1}^* u \in \mathrm{SOL}$. This implies $H_{2,1}^* =_\eta \mathbf{I}$ and because $H_{2,1}^* \sqsubseteq H_{2,1}$ we have also $H_{2,1} =_\eta \mathbf{I}$.
Consider the standard reduction $\sigma_3 : D[x(\lambda a.\ a(a(P_k ay)))] \twoheadrightarrow_{\beta\eta} y$. We have $D[x(\lambda a.\ a(aP_k ay)))] \twoheadrightarrow_\beta \lambda t_1 \dots t_r.(\lambda a.\ a(aP_k ay)))H_{3,1} \dots H_{3,h} \twoheadrightarrow_\beta \lambda t_1 \dots t_r.\ H_{3,1}(H_{3,1}(P_k H_{3,1} y))H_{3,2} \dots H_{3,h} \twoheadrightarrow_\beta \lambda t_1 \dots t_q.(H_{3,1}(P_k H_{3,1} y))Q_{3,1} \dots Q_{3,q}$.

As before we have $\forall i \in \{1, \dots, q\}$ $[Q_{1,i}^* \sqsubseteq Q_{3,i}$ and $Q_{3,i} =_\eta t_i]$. Moreover, $\mathbf{I} =_\eta H_{2,1}^* \sqsubseteq H_{3,1}$, hence $H_{3,1} =_\eta \mathbf{I}$. This implies $P_k \mathbf{I} y \in \mathrm{SOL}$ and $P_k \mathbf{I} \in \mathrm{SOL}$. Hence $\{k\}(k)\downarrow$. $\square$

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

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

4.2. Proposition (Statman [13]). *Let*

$$\mathscr{S} = (\{xy\mathbf{U}_1^2(xy\mathbf{U}_1^2(P_k(xy\mathbf{U}_1^2))(xy\mathbf{U}_2^2)) = y,$$
$$xy\mathbf{U}_1^2(P_k(xy\mathbf{U}_1^2))(xy\mathbf{U}_2^2) = y\}, \{x\}).$$

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

Proof. ($\Leftarrow$) $D \equiv \lambda t_1 t_2 . t_2 \mathbf{I} t_1$ is a β-solution for $\mathscr{S}$.

($\Rightarrow$) Let D be a $\beta\eta$-solution for $\mathscr{S}$. Then $Dy\mathbf{U}_1^2 y =_\eta y$. If $Dy\mathbf{U}_1^2\Omega =_\eta y$ then $Dy\mathbf{U}_1^2$ $(P_k(Dy\mathbf{U}_1^2))\,(Dy\mathbf{U}_2^2) =_\eta y(Dy\mathbf{U}_2^2) =_\eta y$, which is impossible. Hence, $D\Omega\mathbf{U}_1^2 y =_\eta y$. Then $D\Omega\mathbf{U}_1^2 =_\eta \mathbf{I}$. We have $\mathbf{I}(\mathbf{I}(P_k\mathbf{I})(Dy\mathbf{U}_2^2)) =_\eta P_k\mathbf{I}(Dy\mathbf{U}_2^2) =_\eta y$. Hence, $P_k\mathbf{I} \in \mathrm{SOL}$. This implies $\{k\}(k)\downarrow$. □

The β-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 β-solvable iff $\{k\}(k)\downarrow$:*

0. $\mathscr{S} = (\{xy(\lambda ab.P_k(xy(\mathbf{U}_1^3\mathbf{I})f)a)(xy(\lambda uv.z)f) = yz\}, \{x\})$.
1. $\mathscr{P} = (\{xy(\lambda ab.P_k(xy(\mathbf{U}_1^3\mathbf{I})f)a)(xy(\lambda uv.z)f) = y(xy)z\}, \{x\})$.

Proof. 0. ($\Leftarrow$) $D \equiv \lambda t_1 t_2 t_3 . t_2(t_1 t_3)\Omega$ is a β-solution for $\mathscr{S}$.

($\Rightarrow$) Let $D \equiv \lambda t_1 t_2 t_3 . t_2(D_1 t_1 t_2 t_3)\dots(D_q t_1 t_2 t_3)$ a β-solution for $\mathscr{S}$. We have $(Dy(\lambda uv.z)f) = z(D_3 y(\lambda uv.z)f)\dots(D_q y(\lambda uv.z)f)$, this implies $q = 2$. Hence

$$Dy(\lambda ab.P_k(Dy(\mathbf{U}_1^3\mathbf{I})f)a)(Dy(\lambda uv.z)f)$$
$$= P_k(\mathbf{U}_1^3\mathbf{I}(D_1\dots)(D_2\dots))\,(D_1\dots) = P_k\mathbf{I}(D_1\dots) = yz.$$

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

1. ($\Leftarrow$) Let $G \equiv \lambda u t_1 t_2 t_3 . t_2\,(t_1(t_2\mathbf{I}\Omega u u t_1)t_3)\Omega$ and $D \equiv GG$. D is a β-solution for $\mathscr{P}$.

($\Rightarrow$) Let D be a β-solution for $\mathscr{P}$, then $D' \equiv \lambda y.D(\lambda a.y)$ is a β-solution for $\mathscr{S}$ (in 0). □

5. Regular SL-systems

Though β-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 β-solvability problem is decidable in Polynomial Time.

The following definition comes from [8, 6.7].

5.0. Definition. An SL-system $\mathcal{S} = (\Gamma, X)$ is said to be quasi-regular iff there exists a function $e: X \to \mathbb{N}$ s.t. (see [1, 10.1.7–13, 10.2.18] for Seq and M_α):

0. for all $x \in X\,[1 \leqslant e(x) \leqslant \min\{m \mid xM_1 \ldots M_m = z \text{ is in } \Gamma\}]$;
1. for all equations $xM_1 \ldots M_m = z$ in $\Gamma[M_{e(x)}$ is solvable];
2. for all equations $M = z \in \Gamma$ for all $\alpha \in \mathrm{Seq}$

if $[\alpha \neq \langle\ \rangle$ and $M_\alpha = \lambda a_1 \ldots a_k.\, xM_1 \ldots M_m$ 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 β-solvability problem is decidable. On the other hand Proposition 4.1 shows that the β-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 $\mathcal{S}$ it can be transformed in Polynomial Time into a quasi-regular SL-system $\mathcal{S}'$ having a decidable β-solvability problem (however, in general, $\mathcal{S}'$ will not yield an X-separability problem). This transformation (called *relaxation*) consists of adding abstractions to the LHS occurrences of the RHS variables of $\mathcal{S}$.

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

(i) relax$(M = z, M' = z)$ is true iff M' is obtained from M replacing one occurrence of z in M with $\lambda a_1 \ldots a_k.\, z$ (where $k > 0$). In this situation we also say that $M' = z$ is a *relaxation* of $M = z$.

(ii) relax* is the reflexive and transitive closure of relax.

(iii) Let (Γ, X), (Γ', X) be SL-systems. Then relax_SL$((\Gamma, X), (\Gamma', X))$ is true iff there exists a bijection φ form Γ to Γ' s.t. for each equation $M = z$ in Γ we have relax*$(M = z, \varphi(M = z))$. In this situation we also say that (Γ, X) is a *relaxation* of (Γ, 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 (OK_NEC) of β-solvability for SL-systems.

Theorems 5.2.1, 5.2.3 give a Polynomial Time sufficient condition (OK_SUFF) of β-solvability for SL-systems and a Polynomial Time algorithm to construct a β-solution (if any) to an SL-system satisfying OK_SUFF.

Since, by 4.1, β-solvability is undecidable OK_NEC and OK_SUFF cannot be equal. However for each SL-system $\mathcal{S}$ s.t. regular_SL$(\mathcal{S})$ = [**if** OK_NEC$(\mathcal{S})$ **then** OK_SUFF$(\mathcal{S})$ **else** true] is true the β-solvability problem is decidable. Again by 4.1 the function regular_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 $\mathscr{S}$ we can always find in Polynomial Time a relaxation $\mathscr{S}'$ of $\mathscr{S}$ s.t. regular_SL($\mathscr{S}'$) = true.

5.2. Theorem. *There are functions* OK_NEC, OK_SUFF *from SL-systems to Boole s.t.:*

0. *For all SL-systems* $\mathscr{S}$ OK_NEC($\mathscr{S}$) *is computable in time polynomial in* Size($\mathscr{S}$).
1. *For all SL-systems* $\mathscr{S}$ OK_SUFF($\mathscr{S}$) *is computable in time polynomial in* Size($\mathscr{S}$).
2. *For all SL-systems* $\mathscr{S}$ *if* $\mathscr{S}$ *is* β*-solvable then* OK_NEC($\mathscr{S}$) = true.
3. *For all SL-systems* $\mathscr{S}$ *if* OK_SUFF($\mathscr{S}$) = true *then* $\mathscr{S}$ *is* β*-solvable and we can construct a* β*-solution for* $\mathscr{S}$ *in time polynomial in* Size($\mathscr{S}$).
4. *Let* regular_SL *be the Polynomial Time function from SL-systems to Boole defined as follows:* regular_SL($\mathscr{S}$) = **if** OK_NEC($\mathscr{S}$) **then** OK_SUFF($\mathscr{S}$) **else** *true. Then:*

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

Proof. See Appendix A. □

5.3. Definition. (a) From now on regular_SL, OK_SUFF and OK_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 $\mathscr{S}$ is said to be regular iff regular_SL($\mathscr{S}$) = true.

5.4. Remark. If regular_SL($\mathscr{S}$) = true then the β-solvability problem for $\mathscr{S}$ is decidable. In fact, by 5.2, we have: if regular_SL($\mathscr{S}$) then [$\mathscr{S}$ is β-solvable iff OK_NEC($\mathscr{S}$) = true].

5.5. Example. (i) Let $\mathscr{S} = (\{xy\Omega(\lambda a.\, x\Omega\Omega) = y,\ x\Omega\Omega(\lambda ab.\, z) = z\},\ \{x\})$. $\mathscr{S}$ is regular and OK_NEC($\mathscr{S}$) = true. By 5.2 $\mathscr{S}$ is β-solvable. A β-solution is

$$D \equiv \lambda y t_1 t_2.\, t_2 t_1 (\mathrm{U}_1^3 y).$$

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

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

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

(v) $\mathscr{G} = (\{x(\lambda a.\, az) = z,\ x(\lambda a.a(au)) = u),\ \{x\})$ is not regular (look at $\mathscr{S}$ 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, $\mathbb{Q}$ above, $\mathscr{S}$ 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 β-solvability problem for the systems studied in [12] is NP-complete [17, 2.6].

Outside the class of regular SL-systems the β-solvability problem, though decidable, can be harder.

5.6. Proposition. *There is a class of SL-systems for which the β-solvability problem is* NP-*complete.*

Proof. See Appendix B. □

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

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

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

The hypotheses of 6.2 define our equational programming language (note that $\mathscr{S}^{\#}$ 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 $\mathscr{S} = (\Gamma_1 \cup \Gamma_2, \{x_1 \dots x_n\})$ be a system and $\vec{y} \in (\mathrm{Var} - \{x_1 \dots x_n\})$ s.t.:*

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

H1. *Each equation in Γ_1 has form $x\vec{M} = y(x_1\vec{y}) \dots (x_n\vec{y})\vec{z}$, where $x \in \{x_1 \dots x_n\}$, $y \in \{\vec{y}\}$, $\vec{z} \notin \{x_1 \dots x_n, \vec{y}\}$ and the variables in $\{y \mid x\vec{M} = y(x_1\vec{y}) \dots (x_n\vec{y})\vec{z} \in \Gamma_1\}$ are pairwise distinct.*

H2. *Each equation in* Γ_2 *has form* $x\vec{M} = z$, *where* $x \in \{x_1 \dots x_n\}$ *and* $z \notin \{x_1 \dots x_n, \vec{y}\}$.

H3. $\mathscr{S}^{\#} = (\{x\vec{M} = y \mid x\vec{M} = y(x_1\vec{y}) \dots (x_n\vec{y})\vec{z} \in \Gamma_1\} \cup \{u_{yz}\vec{M} = z \mid x\vec{M} = y(x_1\vec{y}) \dots (x_n\vec{y})\vec{z} \in \Gamma_1$ *and* $z \in \{\vec{z}]$ *and* $u_{yz} \in \{\vec{u}\}\} \cup \Gamma_2, \{x_1 \dots x_n, \vec{u}\})$ *is regular, where* $\vec{u}$ *is a sequence of fresh variables s.t.* $\{\vec{u}\} = \{u_{yz} \mid x\vec{M} = y(x_1\vec{y}) \dots (x_n\vec{y})\vec{z} \in \Gamma_1$ *and* $z \in \{\vec{z}\}\}$.

Then:

0. $\mathscr{S}$ *is* β*-solvable iff* OK_NEC($\mathscr{S}^{\#}$) = true. *Thus* β*-solvability is decidable in Polynomial Time.*

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

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

Proof. See Appendix C. □

6.3. Example. (a) Let $\mathscr{S} = (\{xy\Omega(\lambda a.xyv) = y(xy)v,\ xy\Omega(\lambda ab.z) = z\},\ \{x\})$. Since OK_NEC($\mathscr{S}^{\#}$) = true by 6.2 $\mathscr{S}$ is β-solvable. A β-solution is

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

where

$$H \equiv \lambda yab.((Lyab)[y := \lambda xuab.yxyab(b\Omega\mathbf{U}_1^2)]),$$

$$L \equiv FF = \lambda yt_1t_2.t_2t_1(\mathbf{U}_1^3(y(t_2\Omega(\mathbf{U}_1^3\mathbf{I})FFy)yt_1t_2),$$

$$F \equiv \lambda uyt_1t_2.((Gyt_1t_2)[y := \lambda c.y(t_2\Omega(\mathbf{U}_1^3\mathbf{I})uuy)y]),$$

$$G \equiv \lambda yt_1t_2.t_2t_1(\mathbf{U}_1^3(yyt_1t_2)).$$

(b) Let $\mathscr{S}' = (\{x\vec{y}\underline{0} = y_0(x\vec{y})\underline{0}, x\vec{y}(\underline{s}z) = y_1(x\vec{y})z, x\vec{y}(\lambda ab.z) = z\}, \{x\})$ ($\underline{0}$ and $\underline{s}$ as in 0.5). By 6.2 $\mathscr{S}'$ is β-solvable. A β-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 $\mathfrak{A}_j = \langle\{A_{j,1}, \dots, A_{j,n(j)}\}, \{g_{j,i} : A_{j,k(j,1)} \times \dots \times A_{j,k(j,a(j,i))} \to A_{j,b(j,i)} \mid i = 1, \dots, m(j)\}\rangle$ be term algebras ($j = 1, 2$).

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

$$\Gamma_1 = \{p_{j,1,h}\vec{y}(g_{j,i}\vec{y}z_1 \dots z_{k(j,a(j,i))}) = z_h \mid i = 1, \dots, m(j) \text{ and } h = 1, \dots, k(j, a(j, i))$$

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

$$\Gamma_2 = \{d_j\vec{y}(g_{j,i}\vec{y}z_1 \dots z_{k(j,a(j,i))}) = y_{0,j,i} \mid j = 1, 2 \text{ and } i = 1, \dots, m(j)\}$$

(recognizability of the constructors);

$\Gamma_3 = \{ f\vec{y}(g_{1,i}\vec{y}z_1 \dots z_{k(1,a(1,i))}) = y_{1,i}(f, \vec{g}, \vec{p}, \vec{d} : \vec{y})z_1 \dots z_{k(1,a(1,i))} \mid i = 1, \dots, m(1) \}$

(specification of the function f);

$\vec{g} \equiv g_{1,1}, \dots, g_{2,m(2)}; \qquad \vec{p} \equiv p_{1,1,1}, \dots, p_{2,m(2),k(2,a(2,m(2)))}; \qquad \vec{d} \equiv d_1, d_2;$

$\vec{y} \equiv y_{0,1,1}, \dots, y_{0,2,m(2)} y_{1,1}, \dots, y_{1,m(1)}.$

The equations in $\Gamma_1 \cup \Gamma_2$ describe the data structures and the equations in Γ_3 describe the function f (a set of functions if we consider types). By 6.2 $\mathscr{S}$ is β-solvable. A solution $H[\ ] \equiv (\lambda f \vec{g} \vec{p} \vec{d}.\ [\ ])\ F\vec{G}\vec{P}\vec{D}$ for $\mathscr{S}$ yields the wanted representations for program and data structures ($F\vec{y}$ for the program for the function f, etc). Any partial recursive function can be specified (and hence represented) replacing $y_{1,1}, \dots, y_{1,m(1)}$ with suitable combinators (see 0.2–0.4). Of course it is possible to add equations to $\mathscr{S}$. 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 $\mathscr{S}' = (\Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \{ f\vec{y}(\lambda ab.z) = z \}, \{ f, \vec{g}, \vec{p}, \vec{d} \})$. By 6.2 the system $\mathscr{S}'$ is β-solvable. This means that any partial recursive function from $\mathfrak{A}_1$ to $\mathfrak{A}_2$ can be represented with a λ-term F having from $\lambda\vec{y}t.tG1G2$, with $G1, G2$ λ-terms. The system $\mathscr{S}$ ($\mathscr{S}'$) cannot be transformed into an X-separability problem (3.1) because of the presence of the equations in $\Gamma_1 (\Gamma_1 \cup \{ f\vec{y}(\lambda ab.z) = z \})$ and, as in 0.6, cannot be solved with the methods in [3].

7. Conclusions

Though the β-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 β-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);
- λ-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 λ-calculus

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

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

A.0.0. Definition. Let $M, N \in \Lambda$, $\mathfrak{F} \subset_f \Lambda$ and $\alpha \in \mathrm{Seq}$ (see [1, p. xiii]).

- $\mathfrak{F}_\alpha = \{M_\alpha \mid M \in \mathfrak{F}\}$ (see [1, 10.1.7–13, 10.2.18]).
- We write $\alpha \in_\eta \mathrm{BT}(M)$ iff $\forall \beta < \alpha [\beta \in \mathrm{BT}(M) \Rightarrow M_\beta \in \mathrm{SOL}]$.
- We write $\alpha \in_\eta \mathrm{BT}(\mathfrak{F})$ iff $\forall Q \in \mathfrak{F} [\alpha \in_\eta \mathrm{BT}(Q)]$.
- We write $M \mid \alpha\downarrow$ iff $[\alpha \in_\eta \mathrm{BT}(M)$ and $M_\alpha \in \mathrm{SOL}]$, $M \mid \alpha\uparrow$ otherwise.
- We write $\mathfrak{F} \mid \alpha\downarrow$ iff $\forall Q \in \mathfrak{F}\, Q \mid \alpha\downarrow$.
- Let $Q \in \Lambda$. We define: $M[\langle\ \rangle := Q] \equiv QM$,

$$
\begin{aligned}
&M[\langle j\rangle^*\alpha := Q] \\
&\quad \equiv \textbf{if } M = \lambda x_1 \dots x_n.\, yM_0 \dots M_{m-1} \\
&\qquad \textbf{then if } j < m \textbf{ then } \lambda x_1 \dots x_n.\, yM_0 \dots (M_j[\alpha := Q]) \dots M_{m-1} \\
&\qquad\quad \textbf{else } \lambda x_1 \dots x_n t_m \dots t_j.\, yM_0 \dots M_{m-1} t_m \dots (t_j[\alpha := Q]) \\
&\qquad \textbf{else } QM.
\end{aligned}
$$

- We define the functions deg (degree), ord (order), head (head) as follows: if $M = \lambda x_1 \dots x_n.\, yM_1 \dots M_m$ then $\deg(M) = m$, $\mathrm{ord}(M) = n$, $\mathrm{head}(M) \equiv y$; if $M \notin \mathrm{SOL}$ then $\deg(M) = \mathrm{ord}(M) = 0$ and $\mathrm{head}(M)\uparrow$.
- We write $M \sqsubseteq N$ for $\mathrm{BT}(M) \subseteq \mathrm{BT}(N)$ (see [1, 10.2.3]).
- Let $C[\ ], D[\ ] \in \Lambda[\ ]$. We write $C[\ ] \sqsubseteq D[\ ]$ for $C[z] \sqsubseteq D[z]$ (z fresh).

- $M \sim_\alpha M$ iff $[[M \mid \alpha\uparrow$ and $N \mid \alpha\uparrow]$ or $[M \mid \alpha\downarrow$ and $N \mid \alpha\downarrow$ and $\deg(M_\alpha) - \mathrm{ord}(M_\alpha) = \deg(N_\alpha) - \mathrm{ord}(N_\alpha)$ and $\mathrm{head}(M_\alpha) \equiv \mathrm{head}(N_\alpha)]]$ (see [1, 10.2.19–21]).
- The node α is said to be useful for $\mathfrak{F}$ (usf for $\mathfrak{F}$) iff $[\mathfrak{F} \mid \alpha\downarrow$ and $[\mathrm{Card}(\mathfrak{F}) = 1$ or $\exists M, N \in \mathfrak{F} \neg [M \sim_\alpha N]]]$ (see [1, 10.4.6]).
- We say that $\mathfrak{F}$ agrees up to α (α is agt for $\mathfrak{F}$) iff $[\forall M, N \in \mathfrak{F} \forall \beta < \alpha\, M \sim_\beta N]$.
- The node α is said to be $\mathfrak{F}$-adherent (α is adh for $\mathfrak{F}$) iff $[\exists M \in \mathfrak{F}\, \alpha \in \mathrm{BT}(M)]$ (see [7, 8]).
- The binary relation $\mathrm{ind}(\mathfrak{F})$ is defined as follows:
  $\forall P, Q \in \Lambda\ [\mathrm{ind}(\mathfrak{F}, P, Q)$ iff
  $[P, Q \in \mathfrak{F}$ and $[\forall \alpha \in \mathrm{Seq}\ [[\alpha$ is usf and agt for $\mathfrak{F}$ and $\mathrm{Card}(\mathfrak{F}) > 1]$
  $\Rightarrow [P \sim_\alpha Q$ and $\mathrm{ind}(\{M \in \mathfrak{F} \mid P \sim_\alpha M\}], P, Q)]]]]]]$.
- Let $\vec{x} \equiv x_1, \ldots, x_n \in \mathrm{Var}$ and $\vec{y} \in \mathrm{Var}$. $(\vec{x} : \vec{y})$ is the sequence $(x_1 \vec{y}), \ldots, (x_n \vec{y})$.

A.0.1. Remark (Böhm and Tronci [7]). Let $\mathfrak{F} \subset_f \Lambda$ and $\alpha \in \mathrm{Seq}$. If α is usf and agt for $\mathfrak{F}$ then α is usf, agt and adh for $\mathfrak{F}$.

A.0.2. Definition. Let $\mathbb{T}$ be a theory and $\mathscr{S} = [\Gamma, \{x_1, \ldots, x_n\})$ be a system.

(i) Let $D[\ ] \equiv (\lambda x_1 \ldots x_n.[\ ])D_1 \ldots D_n \in \Lambda[\ ]$ s.t. $\forall i \in \{1, \ldots, n\}\, FV(D_i) \subseteq \{\vec{a}\}$. We say that $D[\ ]$ is a family of $\mathbb{T}$-solutions for $\mathscr{S}$ if the variables in $\vec{a}$ do not occur in $\mathscr{S}$ and $\forall M = N \in \mathscr{S}\, D[M] =_{\mathbb{T}} D[N]$. By abuse of language we also say that $D[\ ]$ is a $\mathbb{T}$-solution for $\mathscr{S}$. Of course for all substitutions $^\sigma \equiv [b := H_b \mid b \in \{\vec{a}\}$ and $H_b \in \Lambda^0]$ $D^\sigma[\ ] \equiv (\lambda x_1 \ldots x_n.[\ ])D_1^\sigma \ldots D_n^\sigma$ is a $\mathbb{T}$-solution for $\mathscr{S}$.

(ii) Let $\mathscr{S} = (\Gamma, X)$ be a system. We define $\mathrm{left}(\mathscr{S}) = \mathrm{left}(\Gamma) = \{M \mid \exists M = N \in \mathscr{S}\}$.

(iii) Let $\mathscr{S} = (\Gamma, X)$ be a system. We define: $\mathrm{right}(\mathscr{S}) = \mathrm{right}(\Gamma) = \{N \mid \exists M = N \in \mathscr{S}\}$.

(iv) Let $\mathscr{R}$ be an equivalence relation on Λ. Then we extend $\mathscr{R}$ to $\mathscr{R}^{\mathrm{left}}$ on $\mathrm{Form}(\Lambda)$ as follows: $(M = N)\mathscr{R}^{\mathrm{left}}(P = Q)$ iff $M \mathscr{R} P$. Let $\mathscr{S} = (\Gamma, X)$ be a system. We write $\Gamma/\mathscr{R}$ for $\Gamma/\mathscr{R}^{\mathrm{left}}$ and $\mathscr{S}/\mathscr{R} = \{(\Delta, X) \mid \Delta \in \Gamma/\mathscr{R}\}$.

(v) We define $\mathrm{Card}(\mathscr{S}) = \mathrm{Card}(\Gamma)$.

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 β-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 $\mathscr{S} = (\Gamma, X)$ be an SL-system. $\mathrm{OK\_NEC}(S) = \mathrm{true}$ iff $\exists \mathscr{S}^+ \in \mathrm{canonical}(\mathscr{S})\, [\mathscr{S}^+$ 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 $\mathscr{S} = (\Gamma, X)$ be an SL-system (3.0).

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

(a) $\forall M = z \in \mathscr{S}\, \exists \alpha \in \mathrm{BT}(M)\ [\mathrm{head}(M_\alpha) \equiv z]$. (The variable z occurs in M.)

(b) The variables in $\mathrm{right}(\mathscr{S})$ are pairwise distinct.

(c) $\forall M = z \in \mathscr{S}\, [\mathrm{FV}(M) \subseteq (X \cup \{z\})]$. (The free variables in M are in $X \cup \{z\}$.)

(ii) A canonical version of $\mathscr{S}$ is a system $\mathscr{G} = (\Pi, X)$ s.t. $\mathscr{G}$ is canonical and Π is obtained from Γ replacing each $M = z$ in Γ with $M'^* = z^*$ where $M' \equiv M[v := \Omega \mid v \in (\mathrm{FV}(M) - (X \cup \{z\}))]$ and $* \equiv [z := u]$ with u fresh variable.

(iii) We write $\mathscr{G} \in \mathrm{canonical}(\mathscr{S})$ iff $\mathscr{G}$ is a canonical version of $\mathscr{S}$.

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

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

Essentially the β-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* LHS terms and initial parts (proper or improper) of LHS 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 $X \subset_{\mathrm{f}} \mathrm{Var}$. We define $Q(X) = \{e : X \times \{0,1,2,3\} \to \mathbb{N} \mid \forall x \in X\ [1 \leqslant e(x,0) \leqslant e(x,1)$ and $e(x,2) \geqslant 1]$ 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 α is (Z, X, e, f)-safe (A.1.5.0) iff its order is not *too small* and the path leading to α does not involve *too much* self-application. A finite set of λ-terms $\mathfrak{F}$ 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 $X, Z \subset_{\mathrm{f}} \mathrm{Var}$, $\mathfrak{F} \subset_{\mathrm{f}} \Lambda$, $e \in Q(X)$ and $f(X, e) : P_\omega(\Lambda) \times \mathrm{Seq} \to \mathbb{N}$.

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

a. $\forall M, N \in \mathfrak{F}\ [[\mathrm{head}(M_\alpha) \equiv x \in X$ and $N_\alpha = \lambda\vec{a}.a_i\vec{Q}$ and $a_i \in \{\vec{a}\}]$
$\Rightarrow [e(x,2) \neq e(x,1) - e(x,0) + \deg(\mathrm{N}_\alpha) - \mathrm{ord}(N_\alpha) + i]]$;

b. $\forall \beta \leqslant \alpha\, \forall M \in \mathfrak{F}\, [\mathrm{head}(M_\beta) \in X \Rightarrow \deg(M_\beta) < e(\mathrm{head}(M_\beta), 0)]$;

c. $\forall M \in \mathfrak{F}\ [\mathrm{head}(M_\alpha) \in Z \Rightarrow [\mathrm{ord}(M_\alpha) = f(X, e, \mathfrak{F}, \alpha)$ and $\deg(M_\alpha) = 0]]$.

1. We say that $\mathfrak{F}$ is (Z, X, e, f)-distinct iff the following conditions are satisfied:
   a. If $\mathrm{Card}(\mathfrak{F}) = 1$ then $[Z \neq \emptyset \Rightarrow \exists\alpha$ usf and agt for $\mathfrak{F}$ $[\alpha$ is (Z, X, e, f)-safe in $\mathfrak{F}$ and $\forall M \in \mathfrak{F}[\mathrm{head}(M_\alpha) \in Z]]]$;
   b. If $\mathrm{Card}(\mathfrak{F}) > 1$ then $\exists\alpha$ usf and agt for $\mathfrak{F}$ s.t. $[\alpha$ is (Z, X, e, f)-safe in $\mathfrak{F}$ and $\forall\mathscr{B} \in \mathfrak{F}/\sim_\alpha \mathscr{B}$ is (Z, X, e, f)-distinct$]$.
2. We say that $\mathfrak{F}$ is Z-distinct iff $\mathfrak{F}$ is $(Z, \emptyset, 0, \lambda, \mathfrak{F}\alpha.\max\{\mathrm{ord}(L_\alpha) \mid L \in \mathfrak{F}\})$-distinct.
3. The SL-system $\mathscr{S} = (\Gamma, X)$ is said to be LR-distinct iff $\mathrm{left}(\mathscr{S})$ is $\mathrm{right}(\mathscr{S})$-distinct (note that $\mathrm{right}(\mathscr{S}) \subset_{\mathrm{f}} \mathrm{Var}$).

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 β-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 $\mathscr{S} = (\Gamma, X)$ be an SL-system. We define:

0. $\mathrm{prefix}(\mathscr{S}) = [xM_1 \ldots M_m\Omega \mid xM_1 \ldots M_m = z \in \mathscr{S}\} \cup \{xM_1 \ldots M_m z \mid xM_1 \ldots M_{m+k} = z \in \mathscr{S}$ and $k > 0\}$.
1. $\mathscr{S}$ is said to be PFR ($\mathscr{S}$ satisfies the prefix rule) iff $\mathrm{prefix}(\mathscr{S})$ 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($\mathscr{S}$) 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 $\mathfrak{F} \subset_{\mathrm{f}} \Lambda$ and $Z \subset_{\mathrm{f}} \mathrm{Var}$ s.t. $\mathrm{Card}(Z) \leqslant \mathrm{Size}(\mathfrak{F})$. Then we can test if $\mathfrak{F}$ is Z-distinct in time polynomial in $\mathrm{Size}(\mathfrak{F})$; we can test if $\mathscr{S}$ is PFR (A.1.7.1) in time polynomial in $\mathrm{Size}(\mathscr{S})$. □

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 β-solvability for SL-systems. The predicate e-good (used in A.2.0) is defined in A.2.1–A.2.3 (canonical($\mathscr{S}$) and $Q(X)$ where defined, respectively, in A.1.1, A.1.4)

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

$$\mathrm{OK\_SUFF}(\mathscr{S}) = \text{true iff } \exists\mathscr{S}^+ \in \mathrm{canonical}(\mathscr{S})\, \exists e \in Q(X)\, [\mathscr{S}^+ \text{ is } e\text{-good}].$$

Intuitively OK_SUFF($\mathscr{S}$) = true iff we can *distinguish* $\mathscr{S}$ LHS terms using only nodes α s.t. for each equation $M = z$ in $\mathscr{S}$ the following conditions are satisfied:

Condition 0: If the head of M_α is an unknown then the degree of M_α is not *too large*.

Condition 1: If the head of M_α is z then the order of M_α is *large enough*.

Condition 0 ensures that β-solvability can be tested in Polynomial Time (if unrestrained self-application is allowed then we get an NP-complete β-solvability problem [17, 2.6]).

Condition 1 ensures that a system solving algorithm exists (remember that, in general, for SL-systems β-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 $M = z \in \mathscr{S}$ and $\mathrm{head}(M_\alpha) \equiv z$. Using the function rad (A.2.1.0) we can check if $\mathrm{ord}(M_\alpha)$ is *large enough*. Taking into account what our system solving algorithm can do (Z, X, e)-distinction (A.2.1.4) yields β-solvability. Note that Z-distinction (A.1.5.2) does not yield β-solvability.

A.2.1. Definition. Let $X, Z \subset_{\mathrm{f}} \mathrm{Var}$, $e \in Q(X)$, $\mathfrak{F} \subset_{\mathrm{f}} \Lambda$, $M \in \mathfrak{F}$ and $\alpha \in \mathrm{Seq}$.

0. We have

$$
\begin{aligned}
&\mathrm{rad}(X, e, \mathfrak{F}, M, \alpha)\\
&\quad = \textbf{case}\\
&\qquad M \mid \alpha\uparrow \ \textbf{then}\ 0;\\
&\qquad \mathrm{head}(M_\alpha) \in X \ \textbf{then}\ e(\mathrm{head}(M_\alpha), 1) - \deg(M_\alpha) + \mathrm{ord}(M_\alpha);\\
&\qquad \exists\beta < \alpha\, \mathrm{head}(M_\beta) \equiv \mathrm{head}(M_\alpha)\\
&\qquad\quad \textbf{then}\ \max\{\deg(N_\theta) \mid \theta \leqslant \alpha \text{ and } \mathrm{head}(N_\theta) \equiv \mathrm{head}(M_\alpha) \text{ and } N \in \mathfrak{F}\}\\
&\qquad\qquad + 1 - \deg(M_\alpha) + \mathrm{ord}(M_\alpha);\\
&\qquad \neg\exists\beta < \alpha\, \mathrm{head}(M_\beta) \equiv \mathrm{head}(M_\alpha) \ \textbf{then}\ \mathrm{ord}(M_\alpha);\\
&\quad \textbf{end}.
\end{aligned}
$$

1. $\mathrm{rad}(X, e, \mathfrak{F}, \alpha) = \max\{\mathrm{rad}(X, e, \mathfrak{F}, M, \alpha) \mid M \in \mathfrak{F}\}$.
2. $\mathrm{rad}(X, e) = \lambda\mathfrak{F}\ \alpha.\, \mathrm{rad}(X, e, \mathfrak{F}, \alpha)$.
3. A node α is said to be (Z, X, e)-safe in $\mathfrak{F}$ iff it is $(Z, X, e, \mathrm{rad}(X, e))$-safe in $\mathfrak{F}$.
4. $\mathfrak{F}$ is said to be (Z, X, e)-distinct iff $\mathfrak{F}$ is $(Z, X, e.\ \mathrm{rad}(X, e))$-distinct (A.1.5.1).

To test *consistency* of specifications we use the set $\mathrm{PFRLR}(\mathscr{S})$.

A.2.2. Definition. Let $\mathscr{S}$ be an SL-system. $\mathrm{PFRLR}(\mathscr{S}) = \{\langle x, M_1, \ldots, M_m, \Omega\rangle \mid xM_1 \ldots M_m = z \in \mathscr{S}\} \cup \{\langle x, M'_1, \ldots M'_m, z\rangle \mid \exists k > 0 \mid [xM_1 \ldots M_{m+k} = z \in \mathscr{S}$ and $* \equiv [\alpha := \lambda t.\, t\Omega_1 \ldots \Omega_k \mid \alpha \in \mathrm{BT}(xM_1 \ldots M_{m+k})$ and $\mathrm{head}((xM_1 \ldots M_{m+k})_\alpha) \equiv z]$ and $(xM_1 \ldots M_{m+k})^* = xM'_1 \ldots M'_{m+k}]\}$.

Finally e-good can be defined.

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

0. $\forall x \in (X \cap \mathrm{head}(\mathrm{left}(\mathscr{S})))\, [e(x, 1) = \min\{m \mid xM_1 \ldots M_m = z \in \mathscr{S}\}]$.
1. $\forall xM_1 \ldots M_m = z \in \mathscr{S}\, [M_{e(x,0)} \in \mathrm{SOL}]$.

2. $\forall x M_1 \dots M_m = z \in \mathscr{S}\,[\mathrm{head}(M_{e(x,0)} \equiv z \Rightarrow e(x,2) = e(x,1) - m + \mathrm{ord}(M_{e(x,0)})]$.
3. $\forall x M_1 \dots M_m = z \in \mathscr{S}\,[\mathrm{head}(M_{e(x,0)}) \notin \mathrm{FV}(M_{e(x,0)}) \Rightarrow e(x,2) \geqslant \mathrm{ord}(M_{e(x,0)})]$.
4. $\forall x M_1 \dots M_m = z \in \mathscr{S}\,[\mathrm{head}(M_{e(x,0)}) \equiv x' \in X \Rightarrow e(x,2) \geqslant e(x',1) - \deg(M_{e(x,0)}) + \mathrm{ord}(M_{e(x,0)})]$.
5. $\mathrm{PFRLR}(\mathscr{S})$ is $(\mathrm{right}(\mathscr{S}), X, e)$-distinct.
6. Let $\mathfrak{F}_x = \{M \in \mathrm{PFRLR}(\mathscr{S}) \mid \mathrm{head}(M_{\langle 0\rangle}) \equiv x\}$. Then $\forall x \in (X \cap \mathrm{head}(\mathrm{left}(\mathscr{S})))$ $[\langle e(x,0)\rangle$ is $(\mathrm{right}(\mathscr{S}), X, e)$-safe in $\mathfrak{F}_x]$.

The definition of OK_SUFF is now complete (see A.2.0, A.1.1, A.1.4, A.2.3). We can now check that OK_SUFF($\mathscr{S}$) can be computed in Polynomial Time (A.2.4).

A.2.4. Proof of 5.2.1. Let $e \in Q(X)$, $\mathfrak{F} \subset_{\mathrm{f}} \Lambda$, X, $Z \subset_{\mathrm{f}} \mathrm{Var}$ s.t. $\mathrm{Card}(X) \leqslant \mathrm{Size}(\mathfrak{F})$ and $\mathrm{Card}(Z) \leqslant \mathrm{Size}(\mathfrak{F})$ and $\alpha \in \mathrm{Seq}$. Then the following tasks can be carried out in time polynomial in $\mathrm{Size}(\mathfrak{F})$: to compute $\mathrm{rad}(X, e, \mathfrak{F}, M, \alpha)$ (in A.2.1.0); to test if α is (Z, X, e)-safe in $\mathfrak{F}$ (A.2.1.3); to test if $\mathfrak{F}$ 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 $\mathrm{Size}(\mathscr{S})$. Thus, e-good can be computed in time polynomial in $\mathrm{Size}(\mathscr{S})$. The only functions e that we need to consider in A.2.0 are defined below.

Let $X = \{x_1, \dots, x_n\}$ and $k = \max\{\deg(M_\alpha) \mid M \in \mathrm{left}(\mathscr{S})$ and $\alpha \in \mathrm{BT}(M)\} + \max\{\mathrm{ord}(M_\alpha) \mid M \in \mathrm{left}(\mathscr{S})$ and $\alpha \in \mathrm{BT}(M)\} + n + 1$. For all $i \in \{1, \dots, n\}$ define

$e(x_i, 1) =$ **if** $x_i \in \mathrm{head}(\mathrm{left}(\mathscr{S}))$ **then** $\min\{m \mid x_i M_1 \dots M_m = z \in \mathscr{S}\}$ **else** $\max\{\deg(M_\alpha) \mid M \in \mathrm{left}(\mathscr{S})$ and $\alpha \in \mathrm{BT}(M)\} + 1$;

$e(x_i, 0) =$ **if** $\mathbf{x}_i \in \mathrm{head}(\mathrm{left}(\mathscr{S}))$ **then** an arbitrary $h \in \mathbb{N}$ s.t. $1 \leqslant h \leqslant e(x_i, 1)$ **else** $e(x_i, 1)$;

$e(x_i, 2) =$ **if** $\exists x_i M_1 \dots M_m = z \in \mathscr{S}\,[\mathrm{head}(M_{e(x_i,0)}) \equiv z]$ **then** $\max\{e(x_i, 1) - m + \mathrm{ord}(M_{e(x_i,0)}) \mid \exists x_i M_1 \dots M_m = z \in \mathscr{S}\,[\mathrm{head}(M_{e(x_i,0)}) \equiv z]\}$
else $k + e(x_i, 1) - e(x_i, 0) + i$;

$e(x_i, 3) = 0$.

For each $x \in X$ there are at most $\mathrm{Node}(\mathrm{left}(\mathscr{S}))$ possible choices for $e(x, 0)$. Thus there are at most $(\mathrm{Card}(X)\,\mathrm{Node}(\mathrm{left}(\mathscr{S}))) = \mathrm{O}(\mathrm{Size}(\mathscr{S})^2)$ possible choices for e. From this fact and the definition of OK_SUFF(A.2.0) the thesis follows. □

A.3. Proof of 5.2.2

We show (A.3.0) that OK_NEC gives a necessary condition of β-solvability for SL-systems. This proves 5.2.2. The proof is in A.3.3–A.3.5.

A.3.0. Proof of 5.2.2. Follows from A.1.2, A.3.4, A.3.5, A.1.1. □

A.3.1. Example. (i) Let $\mathbb{Q} = (\{x(\lambda ab.z) = z,\ x(\lambda a.u) = u\},\ \{x\})$. Then $\mathbb{Q}$ is not LR-distinct (A.1.5.3). Thus, by A.1.0, OK_NEC($\mathbb{Q}$) = false and, by 5.2.2, $\mathbb{Q}$ is not β-solvable.

(ii) Let $\mathscr{S} = (\{x(\lambda a.z) = z,\ x\Omega y = y\},\ \{x\})$. Then $\mathscr{S}$ is not PFR (A.1.7.1). Thus, by A.1.0, OK_NEC(S) = false and, by 5.2.2, $\mathscr{S}$ is not β-solvable.

A.3.2. Counterexample. Of course, by 4.1, OK_NEC is not a sufficient condition of β-solvability. Let $\mathscr{S} = (\{x(\lambda a.az) = z,\ x(\lambda a.a(au)) = u,\ x(\lambda a.a\ (a(a\Omega y))) = y\},\ \{x\})$. We have OK_NEC($\mathscr{S}$) = true, but $\mathscr{S}$ is not β-solvable.

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

A.3.3. Lemma. *Let $\mathscr{S} = (\Gamma, X)$ be a canonical SL-system. If $\mathscr{S}$ is β-solvable then $\exists\alpha$ usf and agt for* $\mathrm{left}(\mathscr{S})$ *s.t.:* $\forall M \in \mathrm{left}(\mathscr{S})[\mathrm{head}(M_\alpha) \in \mathrm{right}(\mathscr{S}) \Rightarrow [\mathrm{ord}(M_\alpha) = \max\{\mathrm{ord}(L_\alpha) \mid L \in \mathrm{left}(\mathscr{S})\}$ *and* $\deg(M_\alpha) = 0]]$.

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

Let $D[\ ]$ be a β-solution for $\mathscr{S}$. Hence $\forall M = z \in \mathscr{S}, D[M] = z$. Consider the standard reduction (* is a suitable substitution) $\sigma: D[M] \twoheadrightarrow M_\alpha^* \vec{H}_M \twoheadrightarrow z$ where $\alpha \in \mathrm{Seq}$ is the first node usf and agt for $\mathrm{left}(\mathscr{S})$ that comes on the head during the standard reduction σ and $M = z \in \mathscr{S}$ (note that α does not depend on the choice of $M = z \in \mathscr{S}$). Because α is the first useful node that comes on the head we have $\forall M, N \in \mathrm{left}(\mathscr{S})$ $|\vec{H}_M| = |\vec{H}_N|$. Hence $\forall M \in \mathrm{left}(\mathscr{S}) |\vec{H}_M| \geqslant \max\{\mathrm{ord}(Q_\alpha) \mid Q \in \mathrm{left}(\mathscr{S})\}$. Suppose that $\mathrm{head}(M_\alpha) \in \mathrm{right}(\mathscr{S})$.

Then, taking into account that $M \in \mathrm{left}(\mathscr{S})$, we have $\max\{\mathrm{ord}(L_\alpha) \mid L \in \mathrm{left}(S)\} \geqslant \mathrm{ord}(M_\alpha) = |\vec{H}_M| \geqslant \max\{\mathrm{ord}(L_\alpha) \mid L \in \mathrm{left}(\mathscr{S})\}$ and $\deg(M_\alpha) = 0$. Thus, the thesis follows. □

Lemma A.3.3 can be strengthen as follows.

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

Proof. By induction on $\mathrm{Card}(\mathscr{S})$.

Case 0: $\mathrm{Card}(\mathscr{S}) = 1$. Trivial.

Case 1: $\mathrm{Card}(\mathscr{S}) > 1$. By Lemma A.3.3 $\exists\alpha$ usf and agt for $\mathrm{left}(\mathscr{S})$ s.t. $\forall M \in \mathrm{left}(\mathscr{S})[\mathrm{head}(M_\alpha) \in \mathrm{right}(\mathscr{S}) \Rightarrow [\mathrm{ord}(M_\alpha) = \max\{\mathrm{ord}(L_\alpha) \mid L \in \mathrm{left}(\mathscr{S})\}$ and $\deg(M_\alpha) = 0]]$.

Let $\mathscr{P} \in \mathscr{S}/\sim_\alpha$ (A.0.2). $\mathscr{P}$ is β-solvable and, by induction hypothesis, $\mathrm{left}(\mathscr{P})$ is $\mathrm{right}(\mathscr{P})$-distinct. Hence, $\mathrm{left}(\mathscr{P})$ is $\mathrm{right}(\mathscr{S})$-distinct. This implies that $\mathrm{left}(\mathscr{S})$ is $\mathrm{right}(\mathscr{S})$-distinct. □

Consistency of specifications follows from β-solvability (A.3.5).

A.3.5. Lemma. *Let $\mathcal{S} = (\Gamma, X)$ be a canonical SL-system and $\mathbb{T}$ be a sms theory. If $\mathcal{S}$ is $\mathbb{T}$-solvable then $\mathcal{S}$ is PFR.*

Proof. Let $D[\ ]$ be a $\mathbb{T}$-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 prefix($\mathcal{S}$) is not distinct (A.1.7.1). Hence there are $M, N \in \text{prefix}(\mathcal{S})$ s.t. ind(prefix($\mathcal{S}$), M, N) (A.0.0). Then, by [8, 3.4.0], ind($D[\text{prefix}(\mathcal{S})]$, $D[M], D[N]$). Hence head($D[M] \equiv$ 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 β-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 $\mathcal{S} = (\{xy\Omega(\lambda a.\, x\Omega\Omega) = y,\ \ x\Omega\Omega(\lambda ab.\, z) = z\},\ \{x\})$. Since OK_SUFF($\mathcal{S}$) then, by 5.2.3, $\mathcal{S}$ is β-solvable. A β-solution is $D \equiv \lambda y t_1 t_2.\, t_2 t_1\ (\mathbf{U}_1^3 y)$.
(ii) The system $\mathcal{P}$ in 5.5 is β-solvable.

The proof of 5.2.3 proceeds as follows. First (in A.4.2) we give an algorithm to find β-solutions to HSL-systems (3.0). Then (in A.4.3) we use such algorithm to build an algorithm to find β-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* $X = \{x_1, \ldots, x_n\}$, $Z \subset_f \text{Var}$, $b \in (\text{Var} - (X \cup Z))$, $\mathfrak{F} \subset_f \Lambda(X \cup Z)$ *with* $\mathfrak{F}$ *λ-free,* $\alpha \in \text{Seq}$ s.t. $\mathfrak{F}|\alpha\downarrow$, α *is agt and adh for* $\mathfrak{F}$ *and* α *is* $(Z, \emptyset, 0)$*-safe in* $\mathfrak{F}$. *Then* $\exists G[\ ] \equiv (\lambda x_1 \ldots x_n.[\ ])\vec{G} \in \Lambda[\ ]$ *s.t.:*
0. $\forall M \in \mathfrak{F}[G[M] = \textbf{if}\ \text{head}(M_\alpha) \in Z\ \textbf{then}\ \text{head}(M_\alpha)\ \textbf{else}\ b\vec{H}_{M,\alpha}]$;
1. $\forall M, N \in \mathfrak{F}[M \sim_\alpha N \Rightarrow |\vec{H}_{M,\alpha}| = |\vec{H}_{N,\alpha}|]$.

Proof. By induction on length(α).

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

Case 1: $\alpha = \langle j\rangle{*}\beta$. Then $\neg\exists M \in \mathfrak{F}\, \text{head}(M) \in Z$, because α is agt and adh for $\mathfrak{F}$. Moreover, there are $x \in X$ and $m \in \mathbb{N}$ s.t. $\forall M\ \mathfrak{F}[\text{head}(M) \equiv x$ and $\deg(M) = m]$. W.l.o.g. we can assume $\text{head}(\mathfrak{F}) = \{x_1\}$. Let $d = \max\{\deg(M_\theta) | \theta \leqslant \alpha$ and $\text{head}(M_\theta) \equiv x_1$ and $M \in \mathfrak{F}\}$, $C[\ ] \equiv (\lambda x_1.[\ ])\, \mathbf{P}_d$, $C[\mathfrak{F}_{\langle j\rangle}] = \{C[M_{\langle j\rangle}] \mid M \in \mathfrak{F}\}$. We have $C[\mathfrak{F}_{\langle j\rangle}]|\beta\downarrow$, β is agt and adh for $C[\mathfrak{F}_{\langle j\rangle}]$. From this it follows (see [16, Part 1,

7.6.4 and 8.0]) that β is $(Z, \emptyset, 0)$-safe in $C[\mathfrak{F}_{\langle j\rangle}]$. Let $r = \max\{\mathrm{ord}(C[M_{\langle j\rangle}]) \mid M \in \mathfrak{F}\}$, $\vec{u} \equiv u_1 \dots u_r$ ($\vec{u}$ fresh) and $\mathfrak{G} = \{C[M_{\langle j\rangle}]\vec{u} \mid M \in \mathfrak{F}\}$. We have $\mathfrak{G} \subset_{\mathrm{f}} \Lambda(\{x_2, \dots, x_n, \vec{u}\} \cup Z)$, $\mathfrak{G}$ is λ-free, $\mathfrak{G}|\beta\downarrow$, β is agt and adh for $\mathfrak{G}$ and β is $(Z, \emptyset, 0)$-safe in $\mathfrak{G}$. Hence by induction hypothesis $\exists D[\ ] \equiv (\lambda x_2 \dots x_n \vec{u}.[\ ])D_2 \dots D_n E_1 \dots E_r \vec{F} \in \Lambda[\ ]$ s.t. $\forall Q \in \mathfrak{G}\,[D[Q] = \textbf{if}\ \mathrm{head}(Q_\beta) \in Z\ \textbf{then}\ \mathrm{head}(Q_\beta)\ \textbf{else}\ b\vec{H}_{Q,\beta}]$ and $\forall P, Q \in \mathfrak{G}\,[P \sim_\beta Q \Rightarrow |\vec{H}_{P,\beta}| = |\vec{H}_{Q,\beta}|]$.

Note that $\forall M \in \mathfrak{F}\,[\mathrm{head}((C[M_{\langle j\rangle}]\vec{u})_\beta) \in Z$ iff $\mathrm{head}(M_\alpha) \in Z]$ and $\forall M, N \in \mathfrak{F}[M \sim_\alpha N \Rightarrow M_{\langle j\rangle} \sim_\beta N_{\langle j\rangle} \Rightarrow C[M_{\langle j\rangle}] \sim_\beta C[N_{\langle j\rangle}] \Rightarrow C[M_{\langle j\rangle}]\vec{u} \sim_\beta C[N_{\langle j\rangle}]\vec{u} \Rightarrow |\vec{H}_{C[M_{\langle j\rangle}]\vec{u},\beta}| = \vec{H}_{C[N_{\langle j\rangle}]\vec{u},\beta}|]$. Let $q \in \mathbb{N}$ s.t. $q \geqslant d - m + 1$.

Define $G[\ ] \equiv (\lambda x_1 \dots x_n.[\ ])\mathbf{P}_d D_2 \dots D_n A_1 \dots A_q$, where $\forall i \in \{1, \dots, q\}\, A_i \equiv \textbf{if}\ i = d - m + 1\ \textbf{then}\ (\lambda t_0 \dots t_{q+m-2}.\, t_j E_1 \dots E_r \vec{F})\ \textbf{else}$ arbitrary. □

When $\mathfrak{F} \subset_{\mathrm{f}} \Lambda(Z)$ we can get rid of the context $G[\ ]$ in A.4.2.0.

A.4.2.1. Corollary. *Let $Z \subset_{\mathrm{f}} \mathrm{Var}$, $b \in (\mathrm{Var} - Z)$, $\mathfrak{F} \subset_{\mathrm{f}} \Lambda(Z)$, $\alpha \in \mathrm{Seq}$ s.t. $\mathfrak{F}|\alpha\downarrow$, α is agt and adh for $\mathfrak{F}$ and α is $(Z, \emptyset, 0)$-safe in $\mathfrak{F}$. Then $\exists F \in \Lambda(\{b\})$ s.t.:*

0. $\forall M \in \mathfrak{F}[FM = \textbf{if}\ \mathrm{head}(M_\alpha) \in Z\ \textbf{then}\ \mathrm{head}(M_\alpha)\ \textbf{else}\ b\vec{H}_{M,\alpha}]$;
1. $\forall M, N \in \mathfrak{F}\,[M \sim_\alpha N \Rightarrow |\vec{H}_{M,\alpha}| = |\vec{H}_{N,\alpha}|]$.

Proof. Let $q = \max\{\mathrm{ord}(M) \mid M \in \mathfrak{F}\}$ and $\mathfrak{G} = \mathfrak{F}a_1 \dots a_q = \{Ma_1 \dots a_q \mid M \in \mathfrak{F}\}$ ($a_1, \dots, a_q$ fresh). Then, by A.4.2.0, there exists $G[\ ] \equiv (\lambda a_1 \dots a_q.[\ ])\vec{G}$ satisfying A.4.2.0.0, 1. Define $F \equiv \lambda t.\, t\vec{G}$. □

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

A.4.2.2. Lemma (Coppo et al. [9]). *Let $Z \subset_{\mathrm{f}} \mathrm{Var}$, $\mathfrak{F} \subset_{\mathrm{f}} \Lambda(Z)$ and $\alpha \in \mathrm{Seq}$. s.t. $\mathfrak{F}|\alpha\downarrow$, α is agt and adh for $\mathfrak{F}$ and $\forall M \in \mathfrak{F}\,\mathrm{head}(M_\alpha) \notin Z$. Then $\exists G \in \Lambda\ \forall M, N \in \mathfrak{F}\,[GM = b_M$ and $[b_M \equiv b_N \Rightarrow M \sim_\alpha N]$ and $[b_M \notin Z]]$.*

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

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

A.4.2.3. Lemma. *Let $Z \subset_{\mathrm{f}} \mathrm{Var}$, $\mathfrak{F} \subset_{\mathrm{f}} \Lambda(Z)$ and $\alpha \in \mathrm{Seq}$ s.t.: $\mathfrak{F}|\alpha\downarrow$, α is agt and adh for $\mathfrak{F}$ and α is $(Z, \emptyset, 0)$-safe in $\mathfrak{F}$. Then $\exists G \in \Lambda\ \forall M, N \in \mathfrak{F}$ s.t.:*

0. $[[GM = \textbf{if}\ \mathrm{head}(M_\alpha) \in Z\ \textbf{then}\ \mathrm{head}(M_\alpha)\ \textbf{else}\ b_{M,\alpha}M]$ *and* $[b_{M,\alpha} \notin Z]]$;
1. $[b_{M,\alpha} \equiv b_{N,\alpha} \Rightarrow M \sim_\alpha N]$.

Proof. Let $u \in (\mathrm{Var} - Z)$. Using A.4.2.1 define $E \in \Lambda(\{u\})$ s.t. $\forall M, N \in \mathfrak{F}[EM = \textbf{if}\ \mathrm{head}(M_\alpha) \in Z\ \textbf{then}\ \mathrm{head}(M_\alpha)\ \textbf{else}\ u\vec{H}_{M,\alpha}]$ and $[M \sim_\alpha N \Rightarrow |\vec{H}_{M,\alpha}| = |\vec{H}_{N,\alpha}|)]$. Let $\mathfrak{B} = \{M \mid M \in \mathfrak{F}$ and $\mathrm{head}(M_\alpha) \notin Z\}$. Using A.4.2.2 define $F \in \Lambda$ s.t. $\forall M, N \in$

$\mathfrak{F}[FM = b_{M,\alpha}$ and $[b_{M,\alpha} \equiv b_{N,\alpha} \Rightarrow M \sim_\alpha N]$ and $[b_{M,\alpha} \notin Z]]$. Let $* \equiv [b_{M,\alpha} := \lambda t a_1 \ldots a_{|\bar{H}_{M,\alpha}|}. b_{M,\alpha} t \mid M \in \mathfrak{B}]$.

Define $G \equiv \lambda t.(\lambda u. E)(F^* tt)t$. □

In A.4.2.4 is a sufficient condition for the existence of a common β-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 left($\mathcal{S}$) is* (right($\mathcal{S}$), $\emptyset$, 0)*-distinct then $\mathcal{S}$ is β-solvable.*

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

Case 0: $\text{Card}(\mathcal{S}) = 1$. Then $\Gamma = \{xM = z\}$. Let $\alpha \in \text{BT}(M)$ s.t. $\text{head}(M_\alpha) \in Z$ and $\deg(M_\alpha) = 0$. Then by A.4.2.3 (with $\mathfrak{F} = \{M\}$) $\exists G \in \Lambda^\circ\, GM = z$.

Case 1: $\text{Card}(\mathcal{S}) > 1$. Then there exists $\alpha = \langle 0 \rangle^* \beta$ usf and agt for left($\mathcal{S}$) s.t. α is $(Z, \emptyset, 0)$-safe in left($\mathcal{S}$). Then, by A.4.2.3, $\exists G \in \Lambda\, \forall xM \in \text{left}(\mathcal{S})$ $[GM =$ **if** $\text{head}(M_\beta) \in Z$ **then** $\text{head}(M_\beta)$ **else** $b_{M,\beta} M]$. Let $\Theta = \{xM = z \mid xM = z \in \mathcal{S}$ and $\text{head}((xM)_\alpha) \notin Z\}$ and $\Theta/\sim_\alpha = \{\Sigma_1, \ldots \Sigma_k\}$ (see A.0.2). By A.1.5.1.1 $\forall \Sigma \in \Theta/\sim_\alpha \text{left}(\Sigma)$ 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 $D \equiv G[b_{M,\beta} := F_{\min\{j \mid xM = z \in \Sigma_j\}} \mid xM = z \in \Theta]$ is a β-solution for $\mathcal{S}$. □

From A.4.2.4 we obtain a sufficient condition of β-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* PFRLR($\mathcal{S}$) *is* (right($\mathcal{S}$), $\emptyset$, 0)*-distinct then $\mathcal{S}$ is β-solvable.*

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

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

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

$$\mathcal{S}_{1,m} = (\Gamma_{1,m}\{x\}) = (\{x\Omega M_1 \ldots M_m = z \mid xM_1 \ldots M_m = z \in \mathcal{S}\}, X),$$

$$\begin{aligned}\mathcal{S}_{2,m} = (\Gamma_{2,m}\{x\}) = (\{xzM'_1 \ldots M'_m = z \mid \exists k > 0\, [xM_1 \ldots M_{m+k} = z \in \mathcal{S} \text{ and} \\ * \equiv [\alpha := \lambda t.\, t\Omega_1 \ldots \Omega_k \mid \alpha \in \text{BT}(xM_1 \ldots M_{m+k}) \text{ and} \\ \text{head}((xM_1 \ldots M_{m+k})_\alpha) \equiv z] \text{ and} \\ (xM_1 \ldots M_{m+k})^* = xM'_1 \ldots M'_{m+k}]\}, \{x\}),\end{aligned}$$

$$\mathcal{S}_{3,m} = (\Gamma_{3,m}, \{x\}) = (\Gamma_{1,m} \cup \Gamma_{2,m}, \{x\}).$$

Since PFRLR($\mathscr{S}$) is (right($\mathscr{S}$), $\emptyset$, 0)-distinct then left($\mathscr{S}_{3,m}$) is (right($\mathscr{S}_{3,m}$), $\emptyset$, 0)-distinct. Hence by induction hypothesis ($d(\mathscr{S}_{3,m}) = 1$)$\mathscr{S}_{3,m}$ is β-solvable.

Let F be a β-solution for $\mathscr{S}_{3,m}$. Let $\mathscr{S}_{4,m} = (\Gamma_{4,m} \ \{x\}) = (\{xM_1 \dots M_n = z \mid xM_1 \dots M_n = z \in \mathscr{S}$ and $n > m\}, \{x\})$. Note that PFRLR($\mathscr{S}_{4,m}$) is (right($\mathscr{S}_{4,m}$), $\emptyset$, 0)-distinct and $d(\mathscr{S}_{4,m}) < d(\mathscr{S})$. Hence, by induction hypothesis, $\mathscr{S}_{4,m}$ is β-solvable. Let H be a β-solution for $\mathscr{S}_{4,m}$. Then $D \equiv \lambda t_1 \dots t_m. F(Ht_1 \dots t_m)t_1 \dots t_m$ is a β-solution for $\mathscr{S}$. □

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 $\mathscr{S} = (\{x(\lambda ab.z) = z, x(\lambda ab.ay)\Omega = y\}, \{x\})$. By A.4.2.5 $\mathscr{S}$ is β-solvable. A β-solution for $\mathscr{S}$ is $D \equiv \lambda t. t(\lambda a. t\mathbf{I}\Omega)\Omega$.

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

(iii) Let $\mathscr{S}'' = (\{x(\lambda a.az) = z, x(\lambda a.a(au)) = u, x(\lambda a.a(a(a\mathbf{I}y))) = y\}, \{x\})$. PFRLR($\mathscr{S}''$) is not (right($\mathscr{S}''$), $\emptyset$, 0)-distinct. However $\mathscr{S}''$ is β-solvable. A β-solution for $\mathscr{S}''$ is $G \equiv \lambda t. t\mathbf{I}$.

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 β-solution to an SL-system satisfying OK_SUFF. This concludes the proof of 5.2.3.

A.4.3.0. Notation. Let $\{\vec{x}\} = \{x_1, \dots, x_n\} \subset_f$ Var and $e \in Q(\{\vec{x}\})$ (see A.1.4).

0. $D_{x,e} \equiv \lambda t_1 \dots t_{e(x,0)} \dots t_{e(x,1)}. t_{e(x,0)}(t_{e(x,0)}\Omega_1 \dots \Omega_{e(x,3)}t_1 \dots t_{e(x,0)-1}\Omega_{e(x,0)} \dots \Omega_{e(x,1)})\Omega_2 \dots \Omega_{e(x,2)}$.
1. $D_{\vec{x},e}[\ ] \equiv (\lambda x_1 \dots x_n.[\ ])D_{x_1,e} \dots D_{x_n,e}$.

We break self-application by finding (A.4.3.1) a common β-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 $\mathscr{S} = (\Gamma, \{\vec{x}, u\})$ be a canonical SL-system with equations having form $u\vec{M} = z$ where $u \notin FV(\vec{M})$.*

If OK_SUFF($\mathscr{S}$) *then* $\exists e \in Q(\{\vec{x}\}) \exists F \in \Lambda^0 \forall G[\ ] \sqsupseteq D_{\vec{x},e}[\ ] \forall uM_1 \dots M_m = z \in \mathscr{S}\ FG[M_1] \dots G[M_m] = z$.

Proof. Let $e' \in Q(\{\vec{x}, u\})$ s.t. $\mathscr{S}$ is e'-good and $e \in Q(\{\vec{x}\})$ be the restriction of e' to $\{\vec{x}\}$. Since $\mathscr{S}$ is e'-good PFRLR($\mathscr{S}$) is (right($\mathscr{S}$), $\{\vec{x}, u\}, e'$)-distinct. Moreover, since $\forall u\vec{M} = z \in \mathscr{S}\,[u \notin \mathrm{FV}(\vec{M})]$, we have that PFRLR($\mathscr{S}$) is (right($\mathscr{S}$), $\{\vec{x}\}$, e)-distinct (see A.2.3). From this follows (see [16, Part 1, 9.2.2 and 7.8]) that $D_{\vec{x},e}$[PFRLR($\mathscr{S}$)] is

(right($\mathscr{S}$), $\emptyset$, 0)-distinct. Let $\mathscr{S}_1 = (\{uD_{\vec{x},e}[\ M_1] \dots D_{\vec{x},e}[M_m] = z \mid uM_1 \dots M_m = z \in \mathscr{S}\}, \{u\})$. From the above discussion follows that $\mathscr{S}_1$ is e-good. Hence, by A.4.2.5, $\mathscr{S}_1$ is β-solvable. Let F be a β-solution for $\mathscr{S}_1$. Let $G[\ ] \sqsupseteq D_{\vec{x},e}[\ ]$. Then

$$\forall uM_1 \dots M_m = z \in \mathscr{S}\ FG[M_1] \dots G[M_m] \sqsupseteq FD_{\vec{x},e}[M_1] \dots D_{\vec{x},e}[M_m] = z. \quad \square$$

A.4.3.2. Lemma. *Let $\mathscr{S} = (\Gamma, X)$ be a canonical SL-system. If* OK_SUFF($\mathscr{S}$) *then $\mathscr{S}$ is β-solvable.*

Proof. Let $e \in Q(X)$ (A.1.4) s.t.:

0. $\mathscr{S}$ is e-good (A.2.3);
1. $d = \max\{\max\{\deg(M_{e(x,0)}) \mid xM_1 \dots M_{e(x,1)} \in \text{left}(\mathscr{S})\}, \max\{e(x,2) \mid x \in X\}, \max\{e(x,1) \mid x \in X\}\}$;
2. $\forall x \in X\ e(x,3) = d + 1 + \max\{\text{ord}(M_{e(x,0)}) - \deg(M_{e(x,0)}) \mid xM_1 \dots M_m \in \text{left}(\mathscr{S})\}$.

By A.2.0 such an e exists. Let $x \in X = \{\vec{x}\} = \{x_1, \dots, x_n\}$.

$\forall x \in X$ let $\mathscr{S}(x) = (\{uM_1 \dots M_m = z \mid xM_1 \dots M_m = z \in \mathscr{S}\}, \{u, \vec{x}\})$ (u fresh). Then $\forall x \in X$ OK_SUFF($\mathscr{S}(x)$). Hence, by A.4.3.1, $\forall x \in X\ \exists F_x \in \Lambda^0\ \forall C[\ ] \sqsupseteq D_{\vec{x},e}[\ ]\ \forall uM_1 \dots M_m = z \in \mathscr{S}(x)\ F_x C[M_1] \dots C[M_m] = z$.

$\forall x \in X$ define:

$$A(x) = \{(i, d + 1 + \text{ord}(M_{e(x,0)}) - \deg(M_{e(x,0)})) \mid xM_1 \dots M_m \in \text{left}(\mathscr{S}) \text{ and } M_{e(x,0)} = \lambda\vec{b}.\, b_i\vec{Q} \text{ and } b_i \in \{\vec{b}\}\},$$

$$B(x) = \{(e(x',0) + \text{ord}(M_{e(x,0)}) - \deg(M_{e(x,0)}),\ d + 1 + \text{ord}(M_{e(x,0)}) - \deg(M_{e(x,0)}) + e(x',1) - e(x',2)) \mid xM_1 \dots M_m \in \text{left}(\mathscr{S}) \text{ and } M_{e(x,0)} = \lambda\vec{b}.\, x'\vec{Q} \text{ and } x' \in X\}.$$

Note that $\forall i \in \mathbb{N}\ [(i,i) \notin A(x)$ 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

$$[i = e(x',0) + b' - q' \text{ and } d + 1 + b - q = d + 1 + b' - q' + e(x',1) - e(x',2)]$$
$$\Rightarrow [e(x',0) + b' - q' \leqslant b \text{ and } e(x',2) - e(x',1) + b - q = b' - q']$$
$$\Rightarrow [e(x',2) \leqslant e(x',1) - e(x',0) + q].$$

Absurd by A.2.3.6 and A.1.5.0.0. $\diamond_0$

1. Remark. Let $(e(x',0) + b' - q',\ d + 1 + b' - q' + e(x',1) - e(x',2)), (e(x'',0) + b'' - q'', d + 1 + b'' - q'' + e(x'',1) - e(x'',2)) \in B(x)$ s.t. $[e(x',0) + b' - q' = e(x'',0) + b'' - q'''$ and $d + 1 + b' - q' + e(x',1) - e(x',2) = d + 1 + b'' - q'' + e(x'',1) - e(x'',2)]$.

We have $[b'' - q'' - b' + q' = e(x',0) - e(x'',0)$ and $e(x'',2) - e(x',2) = e(x'',1) - e(x'',0) + e(x',0) - e(x',1)] \Rightarrow [x' \equiv x''$ and $b' - q' = b'' - q'']$ (by A.1.4).

$\diamond_1$

$\forall x \in X$ define:

$$1(x) \equiv [a_{x,i,i} := \mathbf{P}_d \mid \exists k \in \mathbb{N}\,(i,k) \in A(x)],$$

$$2(x) \equiv [a_{x,i,k} := \lambda t_1 \dots t_{2(d-k)+1+e(x,1)+e(x,2)+e(x,3)} \cdot$$

$$F_x t_{d+1-k+e(x,3)} \cdots t_{d-k+e(x,1)+e(x,3)} \mid (i,k) \in A(x)],$$

$$3(x) \equiv a_{x,i,i} := \mathbf{P}_d \mid \exists k \in \mathbb{N}(i,k) \in B(x)],$$

$$4(x) \equiv [a_{x,i,k} := \lambda t_1 \dots t_{2(d-k)+1+e(x,1)+e(x,2)+e(x,3)} \cdot$$

$$F_x t_{d+1-k+e(x,3)} \cdots t_{d-k+e(x,1)+e(x,3)} \mid (i,k) \in B(x)],$$

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

$$(t_{e(x,0)} a_{x,1,1} \cdots a_{x,1,e(x,3)} t_1 \cdots t_{e(x,0)} \cdots t_{e(x,1)})$$

$$\dots$$

$$(t_{e(x,0)} a_{x,e(x,2),1} \cdots a_{x,e(x,2),e(x,3)} t_1 \cdots t_{e(x,0)} \cdots t_{e(x,1)}).$$

$\forall x \in X$ define

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

We verify that $G[\ ]$ is a β-solution for $\mathscr{S}$.

Check: Let $xM_1 \dots M_m = z \in \mathscr{S}$. We have (with obvious meaning of the symbols):

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

$$\dots$$

$$(G[M_{e(x,0)}]\vec{A}_{x,e(x,2)} G[M_1] \dots G[M_{e(x,1)}])$$

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

Case 0: $M_{e(x,0)} = \lambda a_1 \dots a_r . z$. By A.2.3.2 $e(x,2) + m - e(x,1) = r$. We have $G[xM_1 \dots M_m] = (\lambda a_1 \dots a_r . z) L_1 \dots L_{m-e(x,1)+e(x,2)} = z$.

Case 1: $M_{e(x,0)} = \lambda a_1 \dots a_r . a_h Q_1 \dots Q_q$. By A.2.3.3 $e(x,2) \geqslant r$. We have

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

$$B_1 \dots B_{e(x,2)-r+q} G[M_{e(x,1)+1}] \dots G[M_m]$$

$$= A_{x,h,h} G[Q_1^*] \dots G[Q_q^*] A_{x,h,r+1} \dots A_{x,h,e(x,3)} G[M_1] \dots G[M_{e(x,1)}]$$

$$B_1 \dots B_{e(x,2)-r+q} G[M_{e(x,1)+1}] \dots G[M_m]$$

$$= A_{x,h,d+1+r-q} G[Q_1^*] \dots G[Q_q^*] A_{x,h,r+1} \dots A_{x,h,d+r-q}$$

$$A_{x,h,d+2+r-q} \dots A_{x,h,e(x,3)} G[M_1] \dots G[M_{e(x,1)}]$$

$$B_1 \dots B_{e(x,2)-r+q} G[M_{e(x,1)+1}] \dots G[M_m] \quad \text{(from } 1(x))$$

$$= z \text{ (from } 2(x)),$$

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

We have

$$G[xM_1 \dots M_m]$$

$$= G_y G[Q_1^{**}] \dots G[Q_q^{**}] \, (G[M_{e(x,0)}] \vec{A}_{x,r+1} G[M_1] \dots G[M_{e(x,1)}])$$

$$\dots$$

$$(G[M_{e(x,0)}] \vec{A}_{x,e(x,2)} G[M_1] \dots G[M_{e(x,1)}])$$

$$G[M_{e(x,1)+1}] \dots G[M_m]$$

$$= G[M_{e(x,0)}] \vec{A}_{x,e(y,0)+r-q} G[M_1] \dots G[M_{e(x,1)}] C_1 \dots C_{e(x,2)+e(y,2)-e(y,1)-r+q}$$

$$G[M_{e(x,1)+1}] \dots G[M_m]$$

$$= G_y G[Q_1^*] \dots G[Q_q^*] A_{x,e(y,0)+r-q,r+1} \dots A_{x,e(y,0)+r-q,e(x,3)}$$

$$G[M_1] \dots G[M_{e(x,1)}]$$

$$C_1 \dots C_{e(x,2)+e(y,2)-e(y,1)-r+q} G[M_{e(x,1)+1}] \dots G[M_m]$$

$$= A_{x,e(y,0)+r-q,e(y,0)+r-q} L_1 \dots L_{e(y,2)} A_{x,e(y,0)+r-q,e(y,1)+r-q+1}$$

$$\dots A_{x,e(y,0)+r-q,e(x,3)} G[M_1] \dots G[M_{e(x,1)}]$$

$$C_1 \dots C_{e(x,2)+e(y,2)-e(y,1)-r+q} G[M_{e(x,1)+1}] \dots G[M_m]$$

$$= A_{x,e(y,0)+r-q,d+1+e(y,1)-e(y,2)+r-q} L_1 \dots L_{e(y,2)} H_1 \dots H_{e(x,3)-e(y,1)-r+q-1}$$

$$G[M_1] \dots G[M_{e(x,1)}] C_1 \dots C_{e(x,2)+e(y,2)-e(y,1)-r+q}$$

$$G[M_{e(x,1)+1}] \dots G[M_m] \quad \text{(from } 3(x))$$

$$= z \quad \text{(from } 4(x)). \qquad \square$$

A.4.3.3. Remark. Let $\mathscr{S}$ be an SL-system. If OK_SUFF($\mathscr{S}$) then a β-solution for $\mathscr{S}$ 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 $\mathscr{S} = (\Gamma, X)$ be a quasi-regular SL system s.t. OK_NEC($\mathscr{S}$) = true. By 5.2.2, A.1.0 there exists $\mathscr{S}^+ \in$ canonical($\mathscr{S}$) [$\mathscr{S}^+$ is PFR and LR-distinct]. Let Good_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 $\mathscr{S}^+$ 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, Card(Good_Candidates) = O(Size($\mathscr{S}$)^2).

Given e in Good_Candidates we define the e-relaxation $\mathscr{S}^e$ of $\mathscr{S}^+$ as follows. Let $\mathscr{S}1$ be the relaxation of $\mathscr{S}^+$ s.t. $\mathscr{S}1$ satisfies A.2.3.2. $\mathscr{S}1$ can be computed in Polynomial Time from $\mathscr{S}^+$ simply using A.2.3.2. We call the relaxation that takes from $\mathscr{S}^+$ to $\mathscr{S}1$ relaxation_1. A z-occurrence in an SL-system $\mathbb{Q}$ is a LHS occurrence of a RHS variable of $\mathbb{Q}$. Let k be a large integer (e.g. k as in the proof of A.2.4). Let $\mathscr{S}^e$ be the relaxation of $\mathscr{S}1$ s.t. all the z-occurrences of $\mathscr{S}^+$ (and hence of $\mathscr{S}^e$) not relaxed in relaxation_1 have order k.

Since $\mathscr{S}^+$ is canonical, PFR and LR-distinct from A.2.1.0, A.2.1.4 it follows that there is e in Good_Candidates s.t. $\mathscr{S}^e$ is e-good. Note that e and $\mathscr{S}^e$ can be found in Polynomial Time. We have OK_SUFF($\mathscr{S}^e$) = true. Let $\mathscr{S}$ 2 be the relaxation of $\mathscr{S}$ s.t. $\mathscr{S}^e \in$ canonical ($\mathscr{S}2$). Then OK_SUFF($\mathscr{S}2$) = true. Thus, regular_SL($\mathscr{S}2$) = 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 PropForm (PropVar) be the set of Propositional Formulas (Variables). Let L: PropForm $\to \Lambda$ be defined as follows: $L(x) = x$, $L(\neg A) = L(A)\mathbf{U}_2^2\mathbf{U}_1^2$, $L(A \vee B) = L(A)L(B)\mathbf{U}_2^2$, $L(A \wedge B) = L(A)\mathbf{U}_1^2 L(B)$ (we are representing true with $\mathbf{U}_1^2$ and false with $\mathbf{U}_2^2$). Let $A \in$ PropForm s.t. FV(A) = $\{x_1, \ldots, x_n\}$. We define Transl(A) = $(\{L(A)z\Omega = z\} \cup \{x_i zz = z \mid i = 1, \ldots, n\}, \{x_1, \ldots, x_n\})$. Transl($A$) is an SL-system and is β-solvable iff A is satisfiable. In fact:

($\Rightarrow$) Let $D[\ ] = (\lambda x_1 \ldots x_n.[\ ])D_1 \ldots D_n$ be a β-solution for Transl(A). Then $\forall i \in \{1, \ldots, n\}\,[D_i = \mathbf{U}_1^2$ or $D_i = \mathbf{U}_2^2]$ and $D[L(A)] = \mathbf{U}_1^2$. Choosing $* \equiv [x_i := \textbf{if}\ D_i = \mathbf{U}_1^2\ \textbf{then}$ true **else** false $\mid i = 1, \ldots, n]$ we have $A^* =$ true.

($\Leftarrow$) Let $*$ s.t. $A^* =$ true. $\forall i \in \{1, \ldots, n\}$ define $D_i \equiv \textbf{if}\ x_i^* =$ true **then** $\mathbf{U}_1^2$ **else** $\mathbf{U}_2^2$.

Then $D[\ ] \equiv (\lambda x_1 \ldots x_n.[\ ])D_1 \ldots D_n$ is a β-solution for Transl(A). Define SLNP = {Transl(A) | $A \in$ PropForm}. Then the β-solvability problem for the systems in SLNP is NP-complete. Note that the systems in 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 $\mathscr{S} = (\Gamma, X)$ be a system with equations having form $x\vec{M} = \lambda\vec{a}.\, y\vec{Q}$, where $x \in X$ and $y \in (\mathrm{FV}(\lambda\vec{a}.\, y\vec{Q}) - X)$. We define

$$\mathscr{S}_\Omega = (\{M[(\mathrm{FV}(M) - (X \cup \{\mathrm{head}(N)\})) := \Omega] = \mathrm{head}(N) \mid M = N \in \mathscr{S}\},\ X).$$

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

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

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

H0. *Each equation in $\mathscr{S}$ has form $M = \lambda\vec{a}.\, y\vec{Q}$ with $y \in (\mathrm{FV}(\lambda\vec{a}.\, y\vec{Q}) - X)$.*
H1. $\mathrm{head}(\mathrm{right}(\Gamma_1)) = \{y_1, \ldots, y_k\}$ *and* $\vec{y} \equiv y_1, \ldots, y_k$ *is assignable in* $\mathscr{S}$.
If $\mathscr{S}$ is $\mathbb{T}$-solvable then $\forall^* \equiv [y := H_y y\vec{y} \mid y \in \{\vec{y}\}$ *and* $H_y \in \Lambda^0]\, \mathscr{S}' = (\{M = \lambda\vec{a}.\, z^*\vec{Q}' \mid M = \lambda\vec{a}.\, z\vec{Q} \in (\Gamma_1 \cup \Gamma_2)\},\ X)$ *is $\mathbb{T}$-solvable (where $\vec{Q}'$ is obtained from $\vec{Q}$ replacing some occurrence of $y \in \{\vec{y}\}$ by $H_y y\vec{y}$).*

Proof. Let $X = \{x_1, \ldots, x_n\}$ and $D[\ ] = (\lambda x_1 \ldots x_n.[\ ])\, D_{x_1} \ldots D_{x_n}$ be a $\mathbb{T}$-solution for $\mathscr{S}$ with $\forall x \in X\ D_x = \lambda t_1 \ldots t_{p(x)}.\, t_{h(x)} D_{x,1} \ldots D_{x,q(x)}$. $\forall x \in X$ define $G_x =$ **if** $p(x) \geqslant k$ **then** D_x **else** $\lambda t_1 \ldots t_{p(x)} t_{p(x)+1} \ldots t_k.\, t_{h(x)} D_{x,1} \ldots D_{x,q(x)} t_{p(x)+1} \ldots t_k \equiv \lambda t_1 \ldots t_{p^*(x)}.\, t_{h(x)} D_{x,1} \ldots D_{x,q^*(x)}$ and $G[\ ] \equiv (\lambda x_1 \ldots x_n.[\ ])\, Gx_1 \ldots Gx_n$. Then $G[\ ]$ is also a $\mathbb{T}$-solution for $\mathscr{S}$ (because $\vec{y}$ is assignable in $\mathscr{S}$).

$\forall x \in X$ define $L_x \equiv \lambda t_1 \ldots t_{p^*(x)}.((G_x t_1 \ldots t_{p^*(x)})[t_i := H_{y_i} t_i t_1 \ldots t_k \mid i = 1, \ldots, k])$. Then $L[\ ] = (\lambda x_1 \ldots x_n.[\ ]) L_{x_1} \ldots L_{x_n}$ is a $\mathbb{T}$-solution for $\mathscr{S}'$. $\square$

C.2. Example. Let $\mathscr{S} = (\{xy(\lambda ab.\, b) = y(xy)y,\ xy(\lambda ab.\, z) = z\},\ \{x\})$ and G be a β-solution for $\mathscr{S}$. Then $G^* \equiv \lambda y.\, G(Hyy)$ is a β-solution for $\mathscr{S}^* = (\{xy(\lambda ab.\, b) = Hyy(xy)\,(Hyy),\ xy(\lambda ab.\, z) = z\},\ \{x\})$.

C.3. Corollary. *Let $\mathscr{S} = (\Gamma_1 \cup \Gamma_2, X)$ be a system s.t.:*
H0. *Each equation in Γ_1 has form $x\vec{M} = y\vec{Q}$ with $x \in X$, $y \notin X$.*
H1. $\forall x\vec{M} = y\vec{Q},\ x\vec{N} = y'\vec{P} \in \Gamma_1\ [y \equiv y' \Rightarrow \deg(y\vec{Q}) = \deg(y'\vec{P})]$.
H2. $\mathrm{head}(\mathrm{right}(\Gamma_1)) = \{y_1, \ldots, y_k\}$ and $\vec{y} \equiv y_1, \ldots, y_k$ *is assignable in* $\mathscr{S}$.
H3. *Each equation in Γ_2 has form $x\vec{M} = z$, where $x \in \{\vec{x}\}$ and $z \notin \{\vec{x}, \vec{y}\}$.*

If $\mathscr{S}$ is β-solvable then $\mathrm{OK\_NEC}(\mathscr{S}_\Omega) = \mathrm{true}$.

Proof. If $\mathscr{S}$ is β-solvable then by C.1 $\mathscr{S}_\Omega$ is β-solvable. The thesis follows from 5.2.2. $\square$

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

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

a. Each equation in $\mathscr{S}$ has form $x\vec{M} = \lambda\vec{a}.\, y\vec{Q}$ with $y \in (\mathrm{FV}(\lambda\vec{a}.\, y\vec{Q}) - X$ and $x \in X$;

b. $\mathrm{head}(\mathrm{right}(\Gamma_1)) = \{y_0, \ldots, y_{k-1}\}$ and $\vec{y} \equiv y_0, \ldots, y_{k-1}$ is assignable in $\mathscr{S}$;

c. $D[\ ]$ is a $\mathbb{T}$-solution for $\mathscr{S}$.

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

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

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

3. $D[\ ]$ is said to be a singular solution for $\mathscr{S}$ iff $\exists y_i \in \{\vec{y}\}\, \exists x \in X$ critical for $(\mathscr{S}, D[\ ], y_i)$ s.t. $D_x = \lambda t_1 \ldots t_{p(x)}.\, t_{h(x)} D_{x,1} \ldots D_{x,q(x)}$ and $h(x) \leqslant k$.

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 sms theory and $\mathscr{S} = (\Gamma_1 \cup \Gamma_2, \{\vec{x}\})$ be a system s.t.:*

H0. *Each equation in $\mathscr{S}$ has form $x\vec{M} = \lambda\vec{a}.\, y\vec{Q}$ with $y \in (\mathrm{FV}(\lambda\vec{a}.\, y\vec{Q}) - X)$* and *$x \in X$.*

H1. $\mathrm{head}(\mathrm{right}(\Gamma_1)) = \{y_1, \ldots, y_k\}$ *and $\vec{y} \equiv y_1, \ldots, y_k$ is assignable in $\mathscr{S}$.*

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

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

Proof. 0,1. Let $\{\vec{x}\} = \{x_1, \ldots, x_n\}$ and $D[\ ] = (\lambda x_1 \ldots x_n.[\ ])\, D_{x_1} \ldots D_{x_n}$ be a $\mathbb{T}$-solution for $\mathscr{S}$ with $\forall x \in \{\vec{x}\}\ D_x = \lambda t_1 \ldots t_{p(x)}.\, t_{h(x)} D_{x,1} \ldots D_{x,q(x)}$.

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

$$L(u, y_i\} \equiv \textbf{if } u \text{ is critical for } (\mathscr{S}, D[\ ], y_i) \text{ and } h(u) \leqslant k$$
$$\textbf{then I}$$
$$\textbf{else } t_{h(u)}\vec{H}_{y_i} \text{ where } \vec{H}_{y_i} \text{ is computed as follows:}$$

If u is not critical for $(\mathscr{S}, D[\ ], y_i)$ then $\vec{H}_{y_i}$ is arbitrary.

Suppose that u is critical for $(\mathscr{S}, D[\ ], y_i)$. Let $x\vec{M} = \lambda\vec{a}.\, y_i\vec{Q} \in \Gamma_1$ and $\alpha \in \mathrm{BT}(x\vec{M})$ s.t. u is critical for $(x\vec{M} = \lambda\vec{a}.\, y_i\vec{Q}, D[\ ])$ at α. Consider the head reduction $\sigma: D[x\vec{M}] \twoheadrightarrow_\beta D[((x\vec{M})_\alpha)^*]\vec{A} \equiv E \equiv D[\lambda\vec{b}.\, u \ldots y_i \ldots ((x\vec{M})_{\alpha^*\langle h(u)-1\rangle})^* \ldots]\vec{A} \twoheadrightarrow_\beta$

$(\lambda\vec{v}.\, D[((x\vec{M})_{\alpha^*\langle h(u)-1\rangle})^{**}]\vec{U}) \twoheadrightarrow_\beta N = \mathscr{H}^* D[\lambda\vec{a}.\, y_i\vec{Q}]$ (see [1, 16.2.1]), where the leftmost occurrence of y_i in E comes on the head in σ.
This implies $D[((x\vec{M})_{\alpha^*\langle h(u)-1\rangle})^{**}] \in \mathrm{SOL}$ and $\mathrm{head}(D[((x\vec{M})_{\alpha^*\langle h(u)-1\rangle})^{**}]) \notin \mathrm{FV}(D[((x\vec{M})_{\alpha^*\langle h(u)-1\rangle})^{**}])$ because the occurrence of y_i that comes on the head in σ is not in $\vec{A}$ nor in $D[((x\vec{M})_{\alpha^*\langle h(u)-1\rangle})^{*}]$. Hence there exists $\vec{H} \in \Lambda^0$ (easily computable) s.t. $\forall x\vec{M} = \lambda\vec{a}.\, y_i\vec{Q} \in \Gamma_1\, \forall\alpha \in \mathrm{BT}(x\vec{M})$ [if u is critical for $(x\vec{M} = \lambda\vec{a}.\, y_i\vec{Q}, D[\ ])$ at α then $D[((x\vec{M})_{\alpha^*\langle h(u)-1\rangle})^{**}]\vec{H} = \mathbf{I}$]. We choose $\vec{H}_{y_i} \equiv \vec{H}$.
$\forall x \in X$ define (reasoning as in the proof of C.1 w.l.o.g. we can assume $p(x) \geqslant k$):

$$G_x \equiv \lambda u_1 \ldots u_n t_1 \ldots t_{p(x)}.((D_x t_1 \ldots t_{p(x)})$$

$$[t_i := t_i(L(x, y_i)u_1u_1 \ldots u_n t_1 \ldots t_k) \ldots (L(x, y_i)u_n u_1 \ldots u_n t_1 \ldots t_k) \mid i \in \{1, \ldots, k\}]);$$

$$G_x^* \equiv G_x G_{x_1} \ldots G_{x_n}.$$

Then $G[\ ] \equiv (\lambda x_1 \ldots x_n.[\ ])G_{x_1}^* \ldots G_{x_n}^*$ is a $\mathbb{T}$-solution for $\mathscr{S}'$. Moreover if $D[\ ]$ has nf and is nonsingular for $\mathscr{S}$ then $G[\ ]$ has nf and is nonsingular for $\mathscr{S}'$. □

C.6. Example. Let $\mathscr{S} = (\{xy\Omega(\lambda a.\, xy\Omega) = yy\Omega(\lambda a.\, xy\Omega),\ xy\Omega(\lambda ab.\, z) = z\},\ \{x\})$ and $\mathscr{S}' = (\{xy\Omega(\lambda a.\, xy\Omega) = y(xy)y\Omega(\lambda a.\, xy\Omega),\ xy\Omega(\lambda ab.\, z) = z\},\ \{x\})$. G as in 6.3 is a β-solution for $\mathscr{S}$. Then a possible β-solution for $\mathscr{S}'$ 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 $\mathscr{S} = (\Gamma_1 \cup \Gamma_2, \{\vec{x}\})$ be a system and $\vec{y} \in (\mathrm{Var} - \{\vec{x}\})$ s.t.:*
H0. Each equation in Γ_1 has form $x\vec{M} = y\vec{M}$, where $x \in \{\vec{x}\}$ and $y \in \{\vec{y}\}$.
H1. $\mathrm{head}(\mathrm{right}(\Gamma_1)) \subseteq \{\dot{y}\}$ *and* $\dot{y} \equiv y_1, \ldots, y_k$ *is assignable in* $\mathscr{S}$.
H2. $\forall x\vec{M} = y\vec{M}, x\vec{N} = y'\vec{N} \in \Gamma_1[y \equiv y' \Rightarrow \deg(y\vec{M}) = \deg(y'\vec{N})]$.
H3. *Each equation in Γ_2 has form $x\vec{M} = z$, where $x \in \{\vec{x}\}$ and $z \notin \{\vec{x}, \vec{y}\}$.*
H4. *$\mathscr{S}_\Omega$ is regular.*
Then $\mathscr{S}$ is β-solvable iff $\mathrm{OK\_NEC}(\mathscr{S}_\Omega) = \mathrm{true}$.

Proof. ($\Rightarrow$) Follows from C.3.
($\Leftarrow$) Let $\diamond \equiv [\{\vec{y}\} := \Omega]$, $(\Gamma_1)^1 = \{x\vec{y}M_1^\diamond \ldots M_m^\diamond = y\vec{y}M_1^\diamond \ldots M_m^\diamond \mid x\vec{y}M_1 \ldots M_m = y\vec{y}M_1 \ldots M_m \in \Gamma_1\}$, $(\Gamma_2)^1 = \{(x\vec{M})^\diamond = z \mid x\vec{M} = z \in \Gamma_2\}$, $\mathscr{P} = ((\Gamma_1)^1 \cup (\Gamma_2)^1,\ \{\vec{x}\})$. Since $\mathscr{S}$ is regular we have $\mathscr{P}$ is regular and, by C.0, $\mathscr{P}_\Omega$ is regular. Since (A.1.0) there exists $\mathscr{S}_\Omega^+ \in \mathrm{canonical}(\mathscr{S}_\Omega)$ s.t. $\mathscr{S}_\Omega^+$ is PFR and LR-distinct there exists also $\mathscr{P}'_\Omega \in \mathrm{canonical}\ (\mathscr{P}_\Omega)$ s.t. $\mathscr{P}'_\Omega$ is PFR and LR-distinct. Hence, by A.1.0, 5.3, $\mathrm{OK\_SUFF}(\mathscr{P}_\Omega)$. Moreover, by A.1.1, there exists $\mathscr{P}_\Omega^+ = (\Gamma^+,\ \{\vec{x}\}) \in \mathrm{canonical}\ (\mathscr{P}_\Omega)$ s.t. $\mathrm{OK\_SUFF}(P_\Omega^+)$. Thus, by A.4.3.2, P_Ω^+ is β-solvable.
Let $\Gamma_1^+ = \{x\vec{M} = y \mid x\vec{M} = y \in \mathscr{P}_\Omega^+$ and $y \in \{\vec{y}\}\}$ and $\Gamma_2^+ = \Gamma^+ - \Gamma_1^+$. Let $d(\mathscr{P}, x) = \mathrm{Card}(\{\deg(x\vec{N}) \mid x\vec{N} = y\vec{N} \in \mathscr{P}\})$ and $d^*(\mathscr{P}) = \max\{d(\mathscr{P}, x) \mid x \in \{\vec{x}\}\}$. By induction on $d^*(\mathscr{P})$ we construct a solution for $\mathscr{P}$ and hence for $\mathscr{S}$.

Case 0: $d^*(\mathscr{P}) = 1$. Let $\{\vec{x}\} = \{x_1, \ldots, x_n\}$ and $D[\ ] = (\lambda x_1 \ldots x_n.[\ ])D_{x_1} \ldots D_{x_n}$ be a β-solution for $\mathscr{P}_\Omega^+$ (by A.4.3.2) with $\forall x \in \{\vec{x}\}$ $D_x = \lambda t_1 \ldots t_{p(x)}.t_{h(x)}D_{x,1} \ldots D_{x,q(x)}$. For all $x \in \{\vec{x}\}$ define $G_x \equiv \lambda t_1 \ldots t_{p(x)}.((D_x t_1 \ldots t_{p(x)})[t_i := t_i t_1 \ldots t_{p(x)} \mid i = 1, \ldots, k])$. Then $G[\ ] \equiv (\lambda x_1 \ldots x_n.[\ ])\, G_{x_1} \ldots G_{x_n}$ is a β-solution for $\mathscr{P}$.

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

$$m(x) = \min\{\deg(x\vec{N}) \mid x\vec{N} = y\vec{N} \in \mathscr{P}\},$$

$$\vec{v} \equiv v_{x_1}, \ldots, v_{x_n} \text{ (fresh)}, \quad \vec{u} \equiv u_{x_1}, \ldots, u_{x_n} \text{ (fresh)}, \quad * \equiv [x := x\vec{v} \mid x \in \{\vec{x}\}],$$

$$\Gamma_1^+(x) = \{x\vec{v}M_1^* \ldots M_{m(x)}^* = y\vec{v}M_1^* \ldots M_{m(x)}^* \mid xM_1 \ldots M_{m(x)} = y \in \Gamma_1^+\},$$

$$\Gamma_1^\wedge(x) = \{x\vec{v}M_1^* \ldots M_{m(x)}^* = v_x\vec{v}M_1^* \ldots M_{m(x)}^* \mid$$
$$\exists k > 0[xM_1 \ldots M_{m(x)+k} = y \in \Gamma_1^+]\},$$

$$\Gamma_1^\#(x) = \{u_x\vec{v}M_1^* \ldots M_n^* = y\vec{v}M_1^* \ldots M_n^* \mid xM_1 \ldots M_n = y \in \Gamma_1^+ \text{ and } n > m(x)\},$$

$$(\Gamma_2^+)^* = \{(x\vec{M})^* = z \mid x\vec{M} = z \in \Gamma_2^+\}.$$

Let $\mathbb{Q} = ((\cup\{\Gamma_1^+(x) \cup \Gamma_1^\wedge(x) \cup \Gamma_1^\#(x) \mid x \in \{\vec{x}\}\}) \cup (\Gamma_2^+)^*, \{\vec{x}, \vec{u}\})$. $\mathbb{Q}$ is regular and $d^*(\mathbb{Q}) < d^*(\mathscr{P})$, hence by induction hypothesis there exists a β-solution $D[\ ] \equiv (\lambda x_1 \ldots x_n u_{x_1} \ldots u_{x_n}.[\ ])D_{x_1} \ldots D_{x_n} G_1 \ldots G_n$ for $\mathbb{Q}$. Then $G[\ ] \equiv (\lambda x_1 \ldots x_n.[\ ])$ $(D_{x_1} G_1 \ldots G_n) \ldots (D_{x_n} G_1 \ldots G_n)$ is a β-solution for $\mathscr{P}'' = (\{x\vec{M} = yG_1 \ldots G_n\vec{M} \mid x\vec{M} = y\vec{M} \in (\Gamma_1)^1\} \cup (\Gamma_2)^1, \{\vec{x}\})$.

The thesis follows erasing $G_1, \ldots, G_n$ using C.1. □

C.8. Example. Let $\mathscr{S} = (\{xy\Omega(\lambda a.xy\Omega) = yy\Omega(\lambda a.xy\Omega), xy\Omega(\lambda ab.z) = z\}, \{x\})$. By C.7 $\mathscr{S}$ is β-solvable. A β-solution is $G \equiv \lambda y t_1 t_2.((Dyt_1t_2)[y := yyt_1t_2])$, where D is as in 5.5.

C.9. Lemma. *Let $\mathscr{S} = (\Gamma_1 \cup \Gamma_2, \{\vec{x}\})$ be a system and $\vec{y} \in (\mathrm{Var} - \{\vec{x}\})$ s.t.:*

H0. *$\vec{y} \equiv y_1, \ldots, y_k$ is assignable in $\mathscr{S}$.*

H1. *Each equation in Γ_1 has form $x\vec{M} = y(\vec{x}:\vec{y})\vec{M}\vec{z}$, where $x \in \{\vec{x}\}$, $y \in \{\vec{y}\}$, $\vec{z} \notin \{\vec{x}, \vec{y}\}$ and the variables in* head(right(Γ_1)) *are pairwise distinct.*

H2. *Each equation in Γ_2 has form $x\vec{M} = z$, where $x \in \{\vec{x}\}$ and $z \notin \{\vec{x}, \vec{y}\}$.*

H3. *$\mathscr{S}^\# = ((\{x\vec{M} = y \mid x\vec{M} = y(\vec{x}:\vec{y})\vec{M}\vec{z} \in \Gamma_1\} \cup \{u_{yz}\vec{M} = z \mid x\vec{M} = y(\vec{x}:\vec{y})\vec{M}\vec{z} \in \Gamma_1$ and $z \in \{\vec{z}\}$ and $u_{yz} \in \{\vec{u}\}\} \cup \Gamma_2), \{\vec{x}, \vec{u}\})$ is regular, where $\vec{u}$ is a sequence of fresh variable s.t $\{\vec{u}\} = \{u_{yz} \mid x\vec{M} = y(\vec{x}:\vec{y})\vec{M}\vec{z} \in \Gamma_1$ and $z \in \{\vec{z}\}\}$.*

0. *$\mathscr{S}$ is β-solvable iff* OK_NEC($\mathscr{S}^\#$) = true.

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

Proof. 0. ($\Rightarrow$) If $\mathscr{S}$ is β-solvable then, by C.1 $\mathscr{S}_1 = (\{x\vec{M} = y \mid x\vec{M} = y(\vec{x}:\vec{y})\vec{M}\vec{z} \in \Gamma_1\}, \{\vec{x}\})$ is β-solvable, hence by 5.2.2 there exists $\mathscr{S}_1^+ \in$ canonical($\mathscr{S}_1$) s.t. $\mathscr{S}_1^+$ is PFR and LR-distinct. By C.1 $\forall x\vec{M} = y(\vec{x}:\vec{y})\vec{M}\vec{z} \in \Gamma_1$ $\forall z \in \{\vec{z}\}$ the equation $\mathscr{S}_{yz} = (\{x\vec{M} = z\},$

$\{\vec{x}\}$) is β-solvable, hence by 5.2.2 there exists $\mathscr{S}^+_{yz} \in \text{canonical}(\mathscr{S}_{yz})$ s.t. $\mathscr{S}^+_{yz}$ is PFR and LR-distinct. This implies that there exists $\mathscr{S}^+ \in \text{canonical}(\mathscr{S}^\#)$ s.t. $\mathscr{S}^+$ is PFR and LR-distinct. The thesis follows from definition A.1.0.

($\Leftarrow$) By 5.2.4 $\mathscr{S}^\#$ is β-solvable. Hence by C.7, C.5 the system ($\vec{y}'$ analogous to $\vec{Q}'$ in C.1) $\mathbb{Q} = (\{x\vec{y}\vec{M} = y(\vec{x},\vec{u}:\vec{y})\vec{y}' \ \vec{M} \mid x\vec{y}\vec{M} = y(\vec{x}:\vec{y})\vec{y}\vec{M}\vec{z} \in \Gamma_1\} \cup \{u_{yz}\vec{y}\vec{M} = z \mid u_{yz}\vec{y}\vec{M} = z \in \mathscr{S}^\#\} \cup \Gamma_2, \{\vec{x},\vec{u}\})$ is β-solvable. Let $\vec{a} \equiv a_1,\ldots,a_k$. By C.1 there is a β-solution $G[\ ]$ for $\mathbb{Q}' = (\{x\vec{y}\vec{M} = H_y y\vec{y}(\vec{x},\vec{u}:\vec{y})\vec{y}' \ \vec{M} \mid x\vec{y}\vec{M} = y(\vec{x}:\vec{y})\vec{y}\vec{M}\vec{z} \in \Gamma_1\} \cup \{u_{yz}\vec{y}\vec{M} = z \mid u_{yz}\vec{y}\vec{M} = z \in \mathscr{S}^\#\} \cup \Gamma_2, \{\vec{x},\vec{u}\})$ with $\forall x\vec{y}\vec{M} = y(\vec{x}:\vec{y})\vec{y}\vec{M}z_1 \ldots z_h \in \Gamma_1 H_y \equiv \lambda y\vec{y}\vec{x}\vec{u}\vec{a}t_1 \ldots t_{|\vec{M}|}.y\vec{x}\vec{y}t_1 \ldots t_{|\vec{M}|}(u_{yz_1}t_1 \ldots t_{|\vec{M}|}) \ldots (u_{yz_h}t_1 \ldots t_{|\vec{M}|})$. Then $G[\ ]$ is a β-solution for $\mathscr{S}$.

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

C.10. Example. (a) Let $\mathscr{S} = (\{xy\Omega(\lambda a.xy\Omega) = y(xy)y\Omega(\lambda a.xy\Omega),\ xy\Omega(\lambda ab.z) = z\}, \{x\})$. By C.9 $\mathscr{S}$ is β-solvable. A β-solution is L as in 6.3.

(b) Let $\mathscr{S} = (\{xy\Omega(\lambda a.xyv) = y(xy)y\Omega(\lambda a.xyv)v,\ xy\Omega(\lambda ab.z) = z\}, \{x\})$. By C.9 $\mathscr{S}$ is β-solvable. A β-solution is: H as in 6.3.

C.11. Proof of 6.2. From C.9 and C.1 observing that for regular SL-systems all the algorithms in Appendix C run in Polynomial Time. □

References

[1] H.P. Barendregt, *The Lambda Calculus* (North-Holland, Amsterdam, 1984).

[2] A. Berarducci, Programmazione funzionale e rappresentabilità in alcuni sistemi di Logica Combinatoria, Tesi di Laurea in Matematica, 1983 (advisor: C. Böhm).

[3] C. Böhm and A. Berarducci, Automatic synthesis of typed Λ-programs on term algebras, *Theoret. Comput. Sci.* **39** (1985) 135–154.

[4] C. Böhm and M. Dezani-Ciancaglini, Combinatorial problems, combinator equations and normal forms, Lecture Notes in Computer Science, Vol. 14 (Springer, Berlin, 1974) 185–199.

[5] C. Böhm and A. Piperno, Characterizing X-separability and one-side invertibility in $\lambda\beta\Omega$-calculus, in: *Proc. LICS 88*, Edinburgh, 4–8 July 1988 (IEEE Computer Soc. Press, Silver Spring, MD, 1988).

[6] C. Böhm, A. Piperno and E. Tronci, Solving equations in λ-calculus, in: *Proc. Logic Colloq. 88*, Padova (North-Holland, Amsterdam, 1989).

[7] C. Böhm and E. Tronci, *X-Separability and Left-Invertibility in Lambda-Calculus, LICS 87* (IEEE Computer Soc. Press, Silver Spring, MD, 1987) 320–328.

[8] C. Böhm and E. Tronci, About systems of equations, X-separability and left-invertibility in the λ-calculus, *Inform. and Comput.* **90** (1991) 1–32.

[9] M. Coppo, M. Dezani-Ciancaglini and S. Ronchi della Rocca (semi-) separability of finite sets of terms in Scott's D_∞ models of the λ-calculus, Lecture Notes in Computer Science, Vol. 62 (Springer, Berlin, 1978) 142–164.

[10] I. Margaria and M. Zacchi, Right and left invertibility in λ-β-calculus, *RAIRO Inform. Theor. Appl.* **17** (1983) 71–88.

[11] J. O'Donnell, *Equational Logic as a Programming Language*, MIT Press Series in the Foundation of Computation (MIT Press, Cambridge, MA, 1985).

[12] A. Piperno and E. Tronci, Regular systems of equations in the λ-calculus, *Internat. J. Found. Comput. Sci.* **1** (3) (1990) 325–339.
[13] R. Statman, Private communication.
[14] R. Statman, On sets of solutions to combinator equations, *Theoret. Comput. Sci.* **66** (1989) 99–104.
[15] E. Tronci, Equational programming in λ-calculus, in: *Proc. LICS 91*, Amsterdam, 15–18 July, 1991 (IEEE Computer Soc. Press, Silver Spring, MD, 1991) 191–202.
[16] E. Tronci, Equational programming in λ-calculus via SL-systems, Ph.D. Thesis, Carnegie Mellon University, 1991 (advisor: R. Statman).
[17] E. Tronci, Equational programming in λ-calculus via SL-systems. Part 2, *Theoret. Comput. Sci.* **160** (this Vol.) (1996) 185–216.