Equational Programming in λ-calculus

Enrico Tronci
Department of Mathematics, Carnegie Mellon University, Pittsburgh, PA 15213, USA
et0o@andrew.cmu.edu

Abstract
A system of equations in the λ-calculus is a pair (Γ, X) where Γ is a set of formulas of λ (the equations) and X is a finite set of variables of λ (the unknowns). A system S = (Γ, X) is said to be solvable in the theory T (T-solvable) if there exists a simultaneous substitution with closed λ-terms for the unknowns that makes the formulas of Γ theorems in the theory T. We define a class of systems for which the T-solvability problem is decidable in Polynomial Time. This class yields an equational programming language in which constraints on the code 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.

0. Introduction
A system S of equations in the λ-calculus can be viewed as a set of specifications (the equations) for a finite set of programs (the unknowns) whereas a solution for S yields executable codes for these programs.

0.0. Example. Consider the numerical system (λ, s, p, Zero) (Ber 83)), where: (the terms U_i are defined in 1.0)
Q = λa b. b, S = λa b. b a b, P = λb. b U_1 U_2,
Zero = λb. b (U_1 U_2) U_2.
Let H_0 and H_1 be given combinators.
To find F ∈ Λ^N s.t.
0.0.0. F Q = H_0 F Q,
0.0.1. F (g z) = H_1 F z,
means to find a program that satisfies the specifications expressed by 0.0.0,1 (i.e. 0.0.0 and 0.0.1). By choosing H_0 and H_1 in 0.0.0,1 any partial recursive function can be specified. First note that if F satisfies 0.0.0,1 then F satisfies also: F Q = H_0 F Q and F (g z) = H_1 F z, where g is any vector of variables. If f is a projection function or a constantly zero function or the successor function or is obtained by composition from other functions or is obtained by primitive recursion it is straightforward to choose H_0 and H_1 so that any solution F satisfying 0.0.0,1 represents the function f. We show that the minimalization operator can be specified and hence a function defined by minimalization can be specified. Let F_{min} satisfying 0.0.0,1 when we choose H_0 = λa z g. Zero (g z) z (a (g z) g) and H_1 = λa z g. Zero (g (g z)) (g z) (a (g (g z)) g). Then F_{min} represents the minimalization operator. Let f defined by minimalization from g, i.e. f(x_1, . . . , x_n) = min y [g(x_1, . . . , x_n, y) = 0]. If we choose: H_0 = λa x_1 . . . x_n, F_{min} (G x_1 . . . x_n) and H_1 = λa x_1 . . . x_n, F_{min} (G (g x_1) x_2 . . . x_n) (G is a term representing g), then any combinator F satisfying 0.0.0,1 will represent the function f.

0.1. Example. Let (λ, s, p, Zero) the numerical system in 0.0. Consider the equations
0.1.0. D y Q = y_0 (D y) Q,
0.1.1. D y (g z) = y_1 (D y) z,
where y = y_0, y_1 are variables and D is an unknown combinator. If we can find a solution D for the equations 0.1.0,1 then we can find a solution for the equations 0.0.0,1 defining F = D H_0 H_1. Hence, replacing y_0 and y_1 with suitable combinators, the equations in 0.1.0,1 can specify any partial recursive function.

A class of systems G for which the solvability problem is effectively decidable defines an equational programming language G and an algorithm to solve the systems in G yields a compiler for G. Both specifications and results of the compilation process can be represented inside the λ-calculus.

0. This work is part of a Ph.D. thesis that the author is developing under the supervision of Prof. Rick Statman.
1. This research has been partially supported by grants 203.01.50 and 203.01.50 bis from CNR, Italy.
This would not be possible in a term rewriting system without some abstraction mechanism. Moreover specifying sets of \( \lambda \)-terms with equations does not leave out any interesting set, to be precise: any recursively enumerable and \( \beta \)-closed set of closed \( \lambda \)-terms is the set of solutions to a combinator equation [Sta 89]. These features make the \( \lambda \)-calculus appealing as a calculus for automated synthesis of programs when specifications are expressed with equations. Unfortunately the existence of a solution for a system (as well as the existence of a program satisfying given specifications) is, in general, undecidable. Nevertheless many interesting equational languages have been defined in the literature (e.g. [OD 85 interpreter], [BB 85 compiler], [BT 91 compiler], [PT 90 compiler]).

Though almost all the equational languages allow the specification of wide classes of recursive functions there are drastic differences 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.

0.2. Example. Given the numerical system in 0.0 find a \( \lambda \)-term \( F \) s.t.:  
0. F represents an arbitrarily given partial recursive function;  
1. \( F \) has form \( \lambda t. t \).

We look for \( D \) s.t.:  
0.2.0-1.  
\[ D \upharpoonright \{ y_0 \} = 0 \]  
0.2.0-2  
\[ D \upharpoonright \{ y_0, y_1 \} = y_1 \]  
0.2.0-3  
\[ D \upharpoonright \{ y_0, y_1, y_2 \} = y_2 \]  
0.2.0-4  
\[ D \upharpoonright \{ y_0, y_1, y_2, y_3 \} = y_3 \]  

A possible \( \beta \)-solution is:  
0.2.0.  
\[ D \upharpoonright \{ y_0 \} = y_0 \]  
0.2.0.1  
\[ D \upharpoonright \{ y_0, y_1 \} = y_1 \]  
0.2.0.2  
\[ D \upharpoonright \{ y_0, y_1, y_2 \} = y_2 \]  
0.2.0.3  
\[ D \upharpoonright \{ y_0, y_1, y_2, y_3 \} = y_3 \]

Hence (as in 0.0.1) \( F = D \upharpoonright \{ y_0 \} \) (note that \( F \) has normal form). Equations 0.2.0,1 are sufficient to specify any partial recursive function (as in 0.0,1), however they cannot express any constraint on the code of a program. This can be done using eq. 0.2.2. The system 0.2.0,1 can be transformed in an X-separability problem (2.1.1) [BT 87, 91], but the system 0.2.0-2 (i.e. 0.2.0, 0.2.1, 0.2.2) cannot because of the presence of equation 0.2.2.

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 0.0-2). It would be much better (and in same case essential) to leave this task to the compiler.

0.3. Example. Find a \( \lambda \)-term \( F \) and a numerical system \( (0, s, p, \text{Zero}) \) s.t.:  
0. \( F \) represents an arbitrarily given partial recursive function;  
1. \( F \) has form \( \lambda t. t \);  
2. \( A \) uartual applied to its constructors realizes an arbitrarily given partial recursive function.

We look for \( D, D_p, D_e, D_{\text{Zero}} \in \Lambda^s \) s.t.:  
0.3.0.  
\[ D \upharpoonright \{ \text{Zero} \} = 0 \]  
0.3.1  
\[ D \upharpoonright \{ 0 \} = 0 \]  
0.3.2  
\[ D \upharpoonright \{ 0, 1 \} = 0 \]  
0.3.3  
\[ D \upharpoonright \{ 0, 1, 2 \} = 0 \]  
0.3.4  
\[ D \upharpoonright \{ 0, 1, 2, 3 \} = 0 \]  

A possible \( \beta \)-solution is:  
0.3.0.  
\[ D \upharpoonright \{ 0 \} = 0 \]  
0.3.0.1  
\[ D \upharpoonright \{ 0, 1 \} = 0 \]  
0.3.0.2  
\[ D \upharpoonright \{ 0, 1, 2 \} = 0 \]

Hence (as in 0.0,1) \( F = D \upharpoonright \{ 0 \} \) (note that \( F \) has normal form). Equations 0.2.0,1 are sufficient to specify any partial recursive function (as in 0.0,1), however they cannot express any constraint on the code of a program. This can be done using eq. 0.2.2. The system 0.2.0,1 can be transformed in an X-separability problem (2.1.1) [BT 87, 91], but the system 0.2.0-2 (i.e. 0.2.0, 0.2.1, 0.2.2) cannot because of the presence of equation 0.2.2. □
Because after all the constructors of a data structure are programs nothing prevents us from putting additional constraints on them.

0.4. Example. Find a representation \((0, s, p, \text{Zero})\) for the natural numbers s.t.:

0.4.0. The application of two natural numbers realizes an arbitrary given partial recursive function;

0.4.1. The \(\lambda\)-term representing the successor has form

\[\lambda a b. b \downarrow\]

We look for \(D, D, D, D, D, D, D, D, D, D\) natural numbers s.t.:

0.4.2. \((\text{constraint 0})\)

0.4.3. \((\text{constraint 1})\)

0.4.4. \((\text{constraint 2})\)

0.4.5. \((\text{constraint 3})\)

Choosing \(Q = D, y, S = D, y, P = D, y, \text{Zero} = D, y, y\) yields the wanted representation for the natural numbers. A possible \(\beta\)-solution for this system is \((\mathcal{S} = v_1, v_2, v_3, v_4)\) (the terms \(P, Q\) are defined in 1.0):

\[D_0 = G_0 * G_0 * G_1 * D_0 * D_0 * D_0 * D_0 * D_0,\]

\[D_1 = \lambda y t. t P_2 U_2 U_2 U_2 U_2,\]

\[D_2 = \lambda y t. t P_2 (\lambda a_{1}, \ldots, a_{4}, y_5) (\lambda a_{1} a_{2} a_{3}, y_4) t,\]

\[D_3 = \lambda y t. t P_2 (U_2 I) v_1 v_2 v_3 v_4,\]

\[D_4 = \lambda y t. t (U_2 I) v_1 v_2 v_3 v_4,\]

\[D_5 = \lambda y t. t P_2 (U_2 I) v_1 v_2 v_3 v_4,\]

\[y_1 = y_1 (U_2 I) v_1 v_2 v_3 v_4,\]

\[y_2 = y_2 (U_2 I) v_1 v_2 v_3 v_4,\]

\[y_3 = y_3 (U_2 I) v_1 v_2 v_3 v_4,\]

\[y_4 = y_4 (U_2 I) v_1 v_2 v_3 v_4,\]

\[y_5 = y_5 (U_2 I) v_1 v_2 v_3 v_4,\]

The system 0.4.0-8 cannot be transformed in an X-separability problem (2.1.1) because of the presence of equations 0.4.1,8 and, as in 0.3, cannot be solved with the methods in [BB 85].

Of course analogous considerations apply to any data structure definable with a (heterogeneous) term algebra (8.2). Systems like those in Ex. 0.2-4 cannot be solved with the methods in [BB 85], [BT 87, 91], [BPT 88] or [PT 90] (2.2). As a matter of fact the \(\beta\) (\(\beta\))-solvability problem for this kind of systems is, in general, undecidable (3.1 and 3.3).

The simultaneous presence of self-application (e.g. equation 0.4.4) and bounding on the degree (defined in 1.1.6) of (subterms of) the solutions (e.g. equation 0.4.8) constitutes the main difficulty to face for this kind of systems. Consider equation 0.2.2: the degree of \(D, y\) is bounded by the order (defined in 1.1.6) of (\(\lambda a b\), \(z\)). The smaller this order the more cleverness we need to find a solution. For this reason the solvability of systems similar to those in Ex.0.2-4 is, in general, undecidable. In this paper we define a class of systems (5.3, 8.0) (strictly larger then the classes introduced in [BT 87, 91]) containing systems like those in Ex. 0.2-4 and for which the \(\beta\)-solvability problem is decidable (7.3, 8.0) in Polynomial Time (9.0,1). This class defines an equational programming language in which constraints on the code generated by the compiler can be specified (e.g. as 0.2.2) and (properties of) data structures can be described in an abstract way (e.g. as in 0.3,4).

1. The \(\lambda\)-calculus

We assume the reader familiar with [Bar 84] of which, unless otherwise stated, we use notations and conventions. Var is the set of variables of \(\lambda\), the symbol \(\equiv\) denotes syntactic equality: \(\bar{M} = M_1, \ldots, M_n\); \(\bar{M} = n\);

\[\{\bar{M}\} = \{M_1, \ldots, M_n\}.\]

1.0 Example. \(U^1_1 = \lambda x_1, x_2, x_3, \ldots, x_n : t_1 (1 \leq i \leq n),\)

\[I = U^1_1, \quad K = U^1_3, \quad \omega = \lambda x, x, \omega = \omega \omega,\]

\[P_q = \lambda x_1, \ldots, x_q : t_1 x_q + 1 : t_1 x_q + 1 \ldots x_q,\]

\[<M_1, \ldots, M_n> = P_n M_1 \ldots M_n \text{ are } \lambda\text{-terms}.\]

A term \(M\) is said to be \(\lambda\)-free if \(M = y \bar{M} \) and \(\mathcal{S} \subseteq \Lambda\) is said to be \(\lambda\)-free if its elements are \(\lambda\)-free. \(\Lambda[\{\} (\lambda^0)]\) is the
set of contexts on $\Lambda$ (with no free variables) ([Bar 84, 2.1.18]). If $Z \subseteq \text{Var}$ then $M[z := N]$ is the $\lambda$-term obtained from $M$ by substituting $N$ for all the (free) occurrences of $z \in Z$ in $M$. We write $M[z := N]$ for $M(z := N)$ and $(M = N)[Z := Q]$ for $M[Z := Q] = N[Z := Q]$.

1. We write $a \in_B \text{BT}(M)$ iff $\forall \beta \prec a. [\beta \in_B \text{BT}(M) \Rightarrow M_0 \in \text{SOL}]$.

2. We define the functions $\text{deg}$ (degree), $\text{ord}$ (order), $\text{head}$ (head) as follows:

3. Let $S = (\tau, x_1, \ldots, x_n)$ be a system. $S$ is said to be $T$-solvable iff $\exists D. (\lambda x_1 \ldots x_n. [ ]) D_1 \ldots D_n \in A^0[ ]$ s.t. $\forall M \in S. D[M] = S[D[N]$. (note that $D_1, \ldots, D_n \in A^0$).

4. We define $\text{Card}(\mathcal{S}) = \text{Card}(T)$.

5. We define $\text{Card}(\mathcal{S}) = \text{Card}(T)$.
2. SL-systems
Many interesting systems (e.g. those in Ex. 0.1-4) can be transformed in systems with equations having form \( x \tilde{M} = z \) (SL-systems). Solving SL-systems will be the core of our systems solving algorithm.

2.0. Definition. 0. A system \( S = (\Gamma, X) \) is said to be an SL-system (separation like) if its equations have form \( x \tilde{M} = z \), where \( x \in X \) and \( z \notin X \).

1. An SL-system \( S = (\Gamma, \{x\}) \) is said to be an HSL-system (head separation like) if its equations have form \( x \tilde{M} = z \), where \( x \in \text{FV} (\tilde{M}) \).

A separability problem (2.1.0) ([CDR 78], [Bar 84, 10.4.4]) is a particular HSL-system and an X-separability problem (2.1.1) ([BT 87], [871, 91]) is a particular SL-system.

2.1. Example. Let \( \mathcal{S} = \{ \lambda_1, \ldots, \lambda_m \} \subset \Lambda \alpha \) and \( T \) be a \( \lambda \)-theory. \( \mathcal{S} \) is said to be \( T \)-separable iff \( S = \{(x \tilde{M}_1 = y_1, \ldots , x \tilde{M}_m = y_m), \{x\}\} \) is \( T \)-solvable. \( \mathcal{S} \) is said to be \( (\bar{x}) \)-separable iff (see 1.1.14) \( S = \{(x, \tilde{M}_1 (\bar{x} : y_i) = y_i | i = 1, \ldots , m), \{x\}\} \) is \( T \)-solvable.

However an HSL-system is more general than a separability problem and an SL-system is more general than an X-separability problem (2.2).

2.2. Example. Let \( \mathcal{S} = \{(x \lambda_a. a U_1^2 (a z \Omega \Omega) = z), x \lambda_a. a U_1^2 (a z \Omega \Omega) = z, x \lambda_a. a U_1^2 (a z \Omega \Omega) = z\} \). A possible \( \beta \)-solution for \( \mathcal{S} \) is \( D = \lambda t_1 t_2 \lambda_b. t_3 P_1 \Omega < \lambda u_1 v. u U_1^1, x \lambda a. v. u U_2^2 > \), but there is no \( \beta \)-solution for \( \mathcal{S} \) having form

2.2.0.: \( \lambda_1 \ldots \lambda_p. \lambda_1 \ldots \lambda_q. \lambda_1 \ldots \lambda_r. H_1 \ldots H_q. l_1 \ldots \lambda r \),

where \( H_1 \ldots H_q \in \Lambda \alpha \) and \( t \leq p \). (Hence a \( \beta \)-solution for \( \mathcal{S} \) must have a local memory, e.g. the rightmost occurrence of \( t \) in \( D \).) In fact suppose \( S \) has a \( \beta \)-solution \( L \) having form

2.2.0. The only possible cases yield an absurd.

Case 0. \( L = \lambda t. H. \) Then:

\( L (\lambda a. a U_1^2 (a z \Omega \Omega)) = H U_1^2 (H z \Omega \Omega), \)
\( L (\lambda a. a U_2^2 (a z \Omega \Omega)) = H U_2^2 (H z \Omega \Omega). \)

Because \( H \in \Lambda \alpha \odot \) SOL we have two subcases:

0. \( H = \lambda t_1 t_2. H. \) Then \( (H z \Omega \Omega) \notin \) SOL.
1. \( H = \lambda t_1 t_2. H. \) Then \( (H z \Omega \Omega) \notin \) SOL.

Case 1. \( L = \lambda t. t. \) Then \( L (\lambda a. a U_1^2 (a z \Omega \Omega)) \neq z \).

On the other hand if a \( (X -) \) separability problem has a \( \beta \)-solution then it has a \( \beta \)-solution having form 2.2.0 ([BPT 88], [CDR 78], [Bar 84, 10.4.12]).

As a matter of fact HSL-systems are so powerful that their \( (\beta \eta) \)-solvability problem is undecidable (3.1). However the \( \beta \)-solvability for an interesting class of SL-systems can be characterized (7.3).

3. Undecidability results
To avoid hopeless search for a characterization we give some undecidability result.

The following idea comes from [Sta 87].

3.0. Definition. [Sta 87] Let \( k \in \mathbb{N} \). We define: \( P_k = \lambda x. G_k x \rightarrow k \rightarrow K I I \), where \( G_k = \lambda t. F_k \# \), \( F_k \# \) is obtained from \( F_k \) replacing every redex \( \lambda a. P \) \( Q \) in \( F_k \) by \( t (\lambda a. P) Q \) (hence \( F_k \# \) has \( n \)). \( F_k \lambda \)-defines \( k \) (the \( k \)-th partial recursive function) as in [Bar 84, 8.4]. Note that \( P_k \) has \( n \).

We have: \( P_k I =_p \) if \( (k, k) \downarrow \) then \( I \) else unsolvable.

The \( (\beta \eta) \)-solvability problem for HSL-systems is, in general, undecidable.

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

Proof. (\( \Leftarrow \).) Then \( D = \lambda t. t I \) is a \( \beta \)-solution for \( \mathcal{S} \).

(\( \Rightarrow \).) Let \( D \) be a \( \beta \eta \)-solution for \( \mathcal{S} \). Let \( \sigma_1 \) be the standard reduction \( D [x (\lambda a. a z) = \beta] \).

\( F = \lambda t_1 \ldots t_p. (\lambda a. a z) H_1 \ldots H_q \rightarrow \beta \)
\( Q = \lambda t_1 \ldots t_q. z Q_1, 1 \ldots Q_1, q \rightarrow \beta \)
where: \( F \) is the first term in the reduction \( \sigma_1 \) in which \( (\lambda a. a z) \) is on the head and the leftmost occurrence of \( z \) in \( F \) will come on the head in \( \sigma_1 \) and \( Q \) is the first term in \( \sigma_1 \) where \( z \) is on the head. Let \( * = [x, u, y = \Omega] \). We have:

\( \forall i \in 1, \ldots q \} [Q_1, i \rightarrow \beta \eta \] and \( Q_1, i \rightarrow \beta \eta \) and
\[ \lambda t_1 \ldots t_r. (\lambda a. a \, x) \, H_1, t_1^* \ldots H_1, t_r^* \rightarrow_D \beta \]
\[ \lambda t_1 \ldots t_r. H_1, t_1^* \, z \, H_1, t_r^* \rightarrow_D \beta \]
\[ \lambda t_1 \ldots t_q. G_{1, 1} \ldots G_{1, q} \rightarrow_D \beta \]

Consider now the standard reduction
\[ \sigma_2 : D[\lambda (a. a \, (a \, u))] \rightarrow_D \beta \]
\[ \lambda t_1 \ldots t_r. (\lambda a. a \, (a \, u)) \, H_1, t_1^* \ldots H_1, t_r^* \rightarrow_D \beta \]
\[ \lambda t_1 \ldots t_r. H_1, t_1^* \, u \, H_2, t_1^* \ldots H_1, t_r^* \rightarrow_D \beta \]
\[ \lambda t_1 \ldots t_q. H_1, t_1^* \, u \, Q_{2, 1} \ldots Q_{2, q} \rightarrow_D \beta \]
\[ \text{where } \forall \, i \in \{1, \ldots, h\} \exists \, L_i \in \Lambda^\circ \]
\[ [H_{1, i} = L_i (\lambda a. a \, z) \text{ and } H_{2, i} = L_i (\lambda a. a \, (a \, u))] \text{ and } \]
\[ \forall \, i \in \{1, \ldots, q\} \exists \, V_i \in \Lambda^\circ \]
\[ [Q_{1, i} = V_i (\lambda a. a \, z) \text{ and } Q_{2, i} = V_i (\lambda a. a \, (a \, u))]. \]

Hence \( \forall \, i \in \{1, \ldots, q\} [Q_{1, i} \rightarrow_D \beta] \) and \( Q_{2, i} \rightarrow_D \beta \).

This implies \( Q_{2, i} \rightarrow_D \beta \) and because \( H_{2, i} \rightarrow_D \beta \).

Consider the standard reduction
\[ \sigma_3 : D[\lambda (a. a \, (a \, u))] \rightarrow_D \beta \]
\[ \text{We have: } [D[\lambda (a. a \, (a \, u))] \rightarrow_D \beta] \]
\[ [\lambda t_1 \ldots t_r. (\lambda a. a \, (a \, u)) \, H_1, t_1^* \ldots H_1, t_r^* \rightarrow_D \beta] \]
\[ [\lambda t_1 \ldots t_r. (H_1, t_1^* \, (P_k \, H_2, t_1^* \, y)) \, H_3, t_1 \ldots H_3, t_r^* \rightarrow_D \beta] \]
\[ [\lambda t_1 \ldots t_r. (H_2, t_1^* \, (P_k \, H_2, t_1^* \, y)) \, Q_{3, 1} \ldots Q_{3, q} \rightarrow_D \beta] \]

As before we have: \( \forall \, i \in \{1, \ldots, q\} [Q_{1, i} \rightarrow_D \beta] \) and \( Q_{3, i} \rightarrow_D \beta \).

Moreover \( I \rightarrow_D \beta \).

This implies \( P_k \, I \in \text{SOL} \) and \( P_k \, I \in \text{SOL} \).

Hence \( (k)(k) \rightarrow_D \beta \).

Thus, though both one side (left, right) invertibility problems are decidable in \( \beta \) ([BD 74], [MZ 83]), the problem of finding a common left formulation for a finite set of comonitors is, in general, undecidable in \( \beta \) as well as in \( \beta \).

The existence of a common \( \beta \)-right inverse for a finite set of comonitors is undecidable as well [Sta 87] (3.2).

We report the proof with a system \( \mathcal{S} \) (inspired from [PT 90, 4.6]) slightly different from that in [Sta 87].

### 3.2. Proposition. [Sta 87] The \( \lambda \)-system \( \mathcal{S} = \{ \chi \, y \, U_1^2 \, (P_k \, (x \, y \, U_1^2)) \, (x \, y \, U_2^2) = y, \chi \, x \, y \, U_1^2 \, (P_k \, (x \, y \, U_1^2)) \, (x \, y \, U_2^2) = y, \chi \} \) is \( \beta \)-(bn)-solvable iff \( (k)(k) \rightarrow_D \beta \).

Proof. (\( \rightarrow \)). Then \( D = \lambda t_1 \, t_2 \, t_3 \, t_4 \rightarrow_D \beta \)

Let \( D \) a \( \beta \)-solution for \( \mathcal{S} \). Then \( D \, y \, U_1^2 \, y \rightarrow_D \beta \)

If \( D \, y \, U_1^2 \, y \rightarrow_D \beta \)

Then \( D \, y \, U_1^2 \, (P_k \, (D \, y \, U_1^2)) \, (D \, y \, U_2^2) \)

This implies \( P_k \, I \in \text{SOL} \).

Hence \( P_k \, I \in \text{SOL} \). \( \therefore \)

The \( \beta \)-solvability problem for systems like those in Ex. 0.2-4 is, in general, undecidable (3.3.1 and 3.1).

### 3.3. Proposition. The following systems are \( \beta \)-solvable ifff \( (k)(k) \rightarrow_D \beta \):

0. \( \mathcal{S} = \{ (x \, y \, (\lambda a. (a \, (a \, u))) = (x \, y \, (\lambda a. (a \, (a \, u))) \}

1. \( P = \{ (x \, y \, (\lambda a. (a \, (a \, u))) = (x \, y \, (\lambda a. (a \, (a \, u))) \}

Proof. (\( \rightarrow \)). Then \( D = \lambda t_1 \, t_2 \, t_3 \, t_4 \rightarrow_D \beta \)

Let \( D \) a \( \beta \)-solution for \( \mathcal{S} \). Then \( D \, y \, U_1^2 \, y \rightarrow_D \beta \)

This implies \( P_k \, I \in \text{SOL} \).

Hence \( P_k \, I \in \text{SOL} \). \( \therefore \)

4. A necessary condition of \( \beta \)-solvability for \( \lambda \)-systems

An \( \lambda \)-system is said to be canonical ifff each RHS variable occurs on the LHS, the RHS variables are pairwise distinct and there is no garbage in the LHS terms (4.0.0.0-2).

4.0. Definition. Let \( \mathcal{S} = (\Gamma, X) \) be an \( \lambda \)-system.

We say that \( \mathcal{S} \) is canonical ifff the following conditions are satisfied (\( M \) has form \( x \, \tilde{M} \, (2.0)\)):

1. \( \forall \, M = z \, e \in \mathcal{S} \exists \, x \in BT(M) \text{ and } (M_{x \, a} = z) \).

0.1. The variables in right(\( \mathcal{S} \)) are pairwise distinct.

0.2. \( \forall \, M = z \, e \in \mathcal{S} \forall \, FV(BT(M)) \subseteq (X \, (U \, (x))) \).

1. A canonical version of \( \mathcal{S} \) is a system \( \mathcal{S}^+ = (\Gamma^+, X) \).
\( S^+ \) is canonical and \( \Gamma^+ \) is obtained from \( \Gamma \) replacing 
\( M = z \in \Gamma \) with \( M^* = z^* \) where:
\[ M' = \{ v \mapsto \Omega \mid v \in (FV(M) \setminus (\Omega \cup \{ z \})) \} \]
and 
\[ * = \{ z \mapsto u \} \] with \( u \) fresh variable.

4.1. Remark. Let \( S = (\Gamma, X) \) be an SL-system and \( T \) be a
smt theory. Then \( S \) is T-solvable iff there exists a canonical
version \( S^+ \) of \( S \) s.t. \( S^+ \) is T-solvable. Moreover, up to
renumbering of the names of the free variables, there is at
most one canonical version of \( S \). Hence it is not restrictive to
consider only canonical SL-systems.

4.2. Example. Let \( S = ((x x (x (\lambda a. z)) = z, \]
\( x x (x (x z)) = z, x v (\lambda a. z) = z), (x)) \).
Then \( S \) is \( \beta \)-solvable iff \( S^+ \) is \( \beta \)-solvable, where:
\( S^+ = ((x x (x (\lambda a. u)) = u, x x (x (x y)) = y, \]
\( x \Omega (\lambda a. b. z) = z), (x)) \) (\( S^+ \) is a canonical version of \( S \)).

Even for an HSL-system distinction ([ICDR 78], [Bar 84, 10.4.7]) (4.5)
is not a sufficient condition for \( \beta \)-solvability.

4.3. Example. Let \( S = ((x (\lambda a. b. z) = z, \]
\( x (\lambda a. u) = u), (x)) \) be a canonical HSL-system.

Then left(\( S \)) is distinct, but \( S \) is not \( \beta \)-solvable because
the order of \( (\lambda a. u) \) is too small.

If \( S \) is a \( \beta \)-solvable SL-system and \( M = z \in S \)
and head(\( M_0 \) \# \( z \)) then ord(\( M_0 \)) cannot be too small.

4.4. Lemma. Let \( S = (\Gamma, X) \) be a canonical SL-system.

If \( S \) is \( \beta \)-solvable then: \( \exists \alpha \) usf and agt for left(\( S \)) s.t.
\( \forall M \in \text{left}(S) \Rightarrow \text{head}(M_0) \in \text{right}(S) \)
\( \Rightarrow \text{ord}(M_0) = 0 \) and \( \text{ord}(M_0) = \text{max} \{ \text{ord}(L_e) \mid L_e \in \text{left}(S) \} \} \).

Proof. If Card(\( \text{left}(S) \)) = 1 trivial. Let Card(\( \text{left}(S) \)) > 1.

As in the proof of [Bar 84, 14.4.13] we can prove that there exists
\( \alpha \) usf and agt for left(\( S \)). Let \( D[\Gamma] \) a \( \beta \)-solution for
\( S \). Hence \( \forall M \in S \Rightarrow D[M] = z \). Consider the standard
reduction (* is a suitable substitution) \( \sigma : D[M] \rightarrow M_0^* \end{eqnarray}
\( M \rightarrow z \), where \( \alpha \in \text{Seq} \) is the first node usf and agt for
left(\( S \)) that comes on the head during the standard reduction \( \sigma \)
and \( M = z \in S \) (note that \( \alpha \) does not depend on the choice
of \( M = z \in S \)). Because \( \alpha \) is the first useful node that comes on
the head we have \( \forall M, N \in \text{left}(S) \Rightarrow \text{ord}(Q_0) \Rightarrow \text{ord}(Q_0) \).

Hence \( \forall M \in \text{left}(S) \Rightarrow \text{ord}(Q_0) \Rightarrow \text{ord}(Q_0) \geq \text{ord}(Q_0) \).

Suppose that head(\( M_0 \)) \( \in \text{right}(S) \). Then ord(\( M_0 \)) \( \geq \text{ord}(Q_0) \).

4.5. Definition. Let \( Z \subseteq X \) be a canonical SL-system.

If \( S \) is \( Z \)-distinct iff the following conditions are satisfied:

0. If Card(\( Z \)) = 1 then 
\( \exists \alpha \) usf and agt for \( X \) \( \forall M \in S \Rightarrow \text{ord}(M_0) = 0 \).

1. If Card(\( Z \)) > 1 then \( \exists \alpha \) usf and agt for \( S \) s.t.
\( \forall M \in S \Rightarrow \text{ord}(M_0) = 0 \) and \( \text{ord}(M_0) = \text{max} \{ \text{ord}(L_e) \mid L_e \in S \} \}

1.1. \( \forall P \in \text{Var} / \sim \alpha \) (\( P \) is \( Z \)-distinct).

Note that the \( \emptyset \)-distinction is the distinction introduced in
[ICDR 78] (also in [Bar 84, 10.4.7]).

4.6. Lemma. Let \( S = (\Gamma, X) \) be a canonical SL-system.

If \( S \) is \( \beta \)-solvable then left(\( S \)) is right(\( S \))-distinct.
(Not that right(\( S \)) \( \subseteq \text{Var} \).)


If an SL-system is \( \beta \)-solvable then any proper initial part of
an LHS term can be distinguished from an LHS term (4.7.1, 4.8).
Definition 4.7 is inspired from [BT 91, 4.1.4].

4.7. Definition. Let \( S = (\Gamma, X) \) be an SL-system.

We define:

0. \( \text{prefix}(S) = \{ \langle x, M_1, \ldots, M_m, \Omega, \rangle \mid \}
\[ x M_1 \ldots M_m = z \in S \} \cup \{ \langle x, M_1, \ldots, M_m, \Omega, \rangle \mid \}
\[ x M_1 \ldots M_m + k = z \in S \text{ and } k > 0 \} \}

1. \( S \) is said to be \( PFR \) (\( S \) satisfies the prefix rule) iff
\( \text{prefix}(S) \) is \( \emptyset \)-distinct (4.5).

4.8. Lemma. Let \( S = (\Gamma, X) \) be a canonical SL-system and \( T \)
be a smt theory. If \( S \) is T-solvable then \( S \) is PFR.

Proof. Let \( D[\Gamma] \) a T-solution for \( S \) and prefix(\( S \)) =
\( \{ x M_1 \ldots M_m \mid x M_1 \ldots M_m + k = z \in S \text{ and } k > 0 \} \).

Note that \( \forall M \in \text{left}(S) \Rightarrow \text{prefix}(\( S \)) \). Hence there
are \( M \in \text{left}(S) \) and \( N \in \text{prefix}(\( S \)) \) s.t. ind(\( \text{left}(S) \)) \( \cup \text{prefix}(\( S \)) \), \( M, N \) (1.1.13). Then, by [BT 91, 3.4.0],
ind(\( \text{left}(S) \cup \text{prefix}(\( S \)) \), \( D[M], D[N] \)). Hence
From 4.1,6,8 we get a necessary condition of $\beta$-solvability for SL-systems (4.9).

4.9. Theorem. Let $S = (\Gamma, X)$ be an SL-system. If $S$ is $\beta$-solvable then there exists a canonical version $S^*$ of $S$ s.t. $S^*$ is PFR and left($S^*$) is right($S^*$)-distinct.

4.10. Example. 0. Let $S$ as in 4.3. Then $S$ is not $\beta$-solvable because left($S$) is not right($S$)-distinct.

1. $S = ((x \quad \lambda a. z = z, x \quad \Omega \quad y = y), \{x\})$ is not PFR, hence it is not $\beta$-solvable.

5. Regularity

If $S$ is a canonical SL-system and $M = z \in S$ and head$(M_a)$ = $z$ then the smaller is the order of $M_a$ the more cleverness we need to find a solution for $S$ (if any). For this reason the $\beta$-solvability problem for SL-systems is, in general, undecidable. To get a decidable class of SL-systems we ask that the order of $M_a$ is large enough. The functions in $Q(X)$ (5.0) describe the skeleton of the $\lambda$-terms that will be substituted for the unknowns in an SL-system (7.0).

5.0. Notation. Let $X \subseteq_f Var$. We define:

$$Q(X) = \{e : X \times \{0, 1, 2, 3\} \rightarrow \mathbb{N} \}$$

$$\forall x \in X \quad \exists e(x, 0) \leq e(x, 1) \text{ and } e(x, 2) \geq 1$$

$$\forall x, x' \in X \quad e(x, 2) - e(x', 2) = e(x, 1) - e(x, 0) + e(x', 0) - e(x', 1) \Rightarrow x = x' \}.$$

If $M = z \in S$ and head$(M_a)$ = $z$ then using the function rad (5.1.1) we can check if ord$(M_a)$ is big enough (5.2.0.2) to allow the use of the Böhm-out technique in our systems solving algorithm.

5.1. Definition. Let $X \subseteq_f Var, e \in Q(X), \mathcal{F} \subseteq_f \Lambda, M \in \mathcal{F}$ and $\alpha \in Seq$.

$$0. \quad \text{rad}(X, e, \mathcal{F}, M, \alpha) =$$

$$\begin{cases} 0 & \text{if } M \uparrow \alpha \uparrow \text{ then } 0; \\ \text{head}(M_a) \in X & \text{then } e(\text{head}(M_a), 1) - \deg(M_a) + \text{ord}(M_a); \\ \exists \beta < \alpha \quad \text{head}(M_\beta) = \text{head}(M_a) & \text{then max} \{\deg(N_a) | \beta \leq \alpha \text{ and } \text{head}(N_a) = \text{head}(M_a) \text{ and } N \in \mathcal{F} \} + 1 - \deg(M_a) + \text{ord}(M_a); \\ \neg \exists \beta < \alpha \quad \text{head}(M_\beta) = \text{head}(M_a) & \text{then ord}(M_a); \end{cases}$$

Notation.

$$\mathcal{F} \subseteq_f \Lambda, e \in Q(X).$$

Taking into account what our systems solving algorithm can do ($Z, X, e$)-distinction (5.2.1) yields $\beta$-solvability (whereas $Z$-distinction (4.5) does not).

5.2. Definition. Let $X, Z \subseteq_f Var, \mathcal{F} \subseteq_f \Lambda, e \in Q(X).$

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

$$\exists M, N \in \mathcal{F} \quad [\text{head}(M_a) \equiv x \in X \text{ and } \text{head}(N_a) \in \text{FV}(N_a) \Rightarrow [e(x, 2) > e(x, 1) - e(x, 0) + \max \{\deg(Q_a) | \text{head}(Q_a) \in \text{FV}(Q_a) \text{ and } Q \in \mathcal{F} \}]].$$

1. $\forall \beta \leq \alpha \forall M \in \mathcal{F}$

$$[\text{head}(M_a) \in X \Rightarrow \deg(M_a) < \deg(\text{head}(M_a)), 0]].$$

2. $\forall M \in \mathcal{F} \quad [\text{head}(M_a) \in Z \Rightarrow [\deg(M_a) = 0 \text{ and } \text{ord}(M_a) = \text{rad}(X, e, \mathcal{F}, \alpha)].]$$

We say that $\mathcal{F}$ is ($Z, X, e$)-distinct iff the following conditions are satisfied:

1.0. If $\text{Card}(\mathcal{F}) = 1$ then $\exists \alpha \text{ usf and agt for } \mathcal{F}$ s.t. $[\alpha \text{ is (Z, X, e)-safe in } \mathcal{F} \text{ and } \forall M \in \mathcal{F} \text{ head}(M_a) \in Z].$

1.1. If $\text{Card}(\mathcal{F}) > 1$ then $\exists \alpha \text{ usf and agt for } \mathcal{F}$ s.t. $[\alpha \text{ is (Z, X, e)-safe in } \mathcal{F} \text{ and } \forall P \in \mathcal{F} / \sim \alpha P \text{ is } (Z, X, e)-\text{distinct}].$

The more severe are the constraints we set on the code to be generated by the compiler the more cleverness is required from the compiler to satisfy these constraints. If we ask for too much cleverness such a compiler does not exist. This is the meaning of the undecidability of the $\beta$-solvability problem for SL-systems (3.1). To get a class of SL-systems for which the $\beta$-solvability problem is decidable we prevent the user from writing too severe constraints on the code to be generated by the compiler. This lead us to the definition of regular SL-systems.

An SL-system is said to be regular iff the following conditions are satisfied:
5.3. Definition. Let $S = (\Gamma, X)$ be a system and $e \in Q(X)$.
0. A canonical SL-system $S$ is said to be e-regular iff whenever $S$ is PFR (4.7.1) and $\text{left}(S)$ is right(S)-
distinct (4.5) the following conditions are satisfied:
0.0. $\forall x \in (X \cap \text{head(\text{left}(S)))}$
\[ [e(x, 1) = \min \{m \mid x M_1 \ldots M_m = z \in S]\}].
0.1. $\forall x M_1 \ldots M_m = z \in S$ $[\text{head}(M_{\ell_0}, 0) \neq z \implies$ (*)
\[ e(x, 2) = e(x, 1) - m + \text{ord} (M_{\ell_0})].
0.2. $\forall x M_1 \ldots M_m = z \in S$ $[\text{head}(M_{\ell_0}, 0) \neq \text{FV}(M_{\ell_0})$ \[ \implies e(x, 2) \geq e(x, 1) - \text{deg}(M_{\ell_0}) + \text{ord} (M_{\ell_0})].
0.3. $\forall x M_1 \ldots M_m = z \in S$ $[\text{head}(M_{\ell_0}, 0) \neq \text{FV}(M_{\ell_0})$ \[ \implies e(x, 2) \geq e(x, 1) - \text{deg}(M_{\ell_0}) + \text{ord} (M_{\ell_0})].
0.4. $\forall x M_1 \ldots M_m = z \in S$ $[\text{head}(M_{\ell_0}, 0) \neq \text{FV}(M_{\ell_0}) \implies$ (*)
\[ e(x, 2) \geq e(x, 1) - \text{deg}(M_{\ell_0}) + \text{ord} (M_{\ell_0})].
0.5. Let (see 1.1.5)
\[ G_x = \{(x M_1 \ldots M_m = z \in S) \}
\[ G_x = \{(x M_1 \ldots M_m = z \in S) \}
\[ \exists k > 0 \}
\[ x M_1 \ldots M_m + k = z \in S \}
\[ \ast = \{(x, \lambda t \Omega_1 \ldots \Omega_k) / \Omega \in \text{BT}(x M_1 \ldots M_m + k)
\[ \text{and } \text{head}(x M_1 \ldots M_m + k) = z \}
\[ \text{and } \text{head}(x M_1 \ldots M_m + k) = z \}
\[ (x M_1 \ldots M_m + k)^* = x M_1' \ldots M_m + k].
\[ (x M_1 \ldots M_m + k)^* = x M_1' \ldots M_m + k].
\[ \text{Then:}$
\[ \text{Then:}$
0.5.0. $\forall x \in (X \cap \text{head(\text{left}(S)))}$
\[ [\text{left}(S) \rightarrow \text{right}(S) \rightarrow \text{e}-\text{safe in } G_x].
0.5.1. $\forall x \in (X \cap \text{head(\text{left}(S)))}$
\[ [(G_x \cup G_x) \rightarrow \text{right}(S) \rightarrow \text{e}-\text{distinct}].
1. A canonical SL-system $S = (\Gamma, X)$ is said to be regular iff \[ \exists e \in (Q(X)) \text{ s.t. } S \text{ is } e-\text{regular}.
2. An SL-system $S$ is said to be regular iff there exists a canonical version $S'$ of $S$ s.t. $S'$ is PFR and \[ \text{left}(S')$ is right(S')-\text{distinct there exists also a canonical version $S''$ of $S$ s.t. $S''$ is regular, PFR and \[ \text{left}(S'')$ is right(S'')-\text{distinct}.
3. Let $S = (\Gamma, X)$ be a system with equations having form \[ x M = x M_1 \ldots y Q_1, \text{where } x \in X \text{ and } y \in \text{FV}(x M_1 \ldots y Q_1 - X).$
$S$ is said to be regular iff $S_{\Omega}$ is regular (see 1.3.6). \[ S_{\Omega}$ is regular (see 1.3.6).
5.4. Example. 0. The systems in 0.1-4, 2.2 and 4.2 are regular.
1. $S = (x (\lambda a. b. z) = z, x (\lambda a. b. a) \Omega = y), (x))$ is regular.
2. $S = ((x (\lambda a. a) z = z, x (\lambda a. a (a u)) = u), (x))$ is regular (5.3.0.5.1 fails).
3. A separability problem (2.1.0) ([ICDR 78], [Bar 84, 10.4.4]) is a regular SL-system.
4. All the systems studied in [BT 87, 91] are regular systems. However there are regular systems (e.g. those in 0.2-4, 5.4.1, 8.2) that are not in the classes studied in [BT 87, 91], [BPT 88] or [PT 90]. On the other hand not all the systems studied in [BPT 88] or [PT 90] are regular systems because in those classes there are systems having proper subterms (of an LHS term) with head an unknown
and with degree too large.

6. Regular HSL-Systems
In 6.0 is a sufficient condition for the existence of a common $\beta$-left-inverse for $S \subseteq \Gamma A^0$ which yields a characterization of the $\beta$-solubility problem for regular HSL-systems (6.1).

6.0. Lemma. Let $S = (\Gamma, (x))$ be a canonical HSL-system with equations having form $x M = z$.
If left(S) is (right(S), $\emptyset$, $\emptyset$)-distinct then $S$ is $\beta$-solvable.
Proof. (sketch) The proof can be carried out by induction on Card(S) endowing the Buchout technique with a local
memory (see 2.2) and using only safe nodes (which is possible because left(S) is (right(S), $\emptyset$, $\emptyset$)-distinct).

The $\beta$-solubility problem for regular HSL-systems is decidable.
6.1. Theorem. Let $S = (\Gamma, (x))$ be a canonical and regular HSL-system. Then $S$ is $\beta$-solvable iff $S$ is PFR and left(S) is right(S)-\text{distinct}.
Proof. (sketch) (\Rightarrow). By 4.9. (\Leftarrow). By induction on Card(\text{\text{degree}(M)} | M \in \text{left}(S))) and using 6.0.

6.2. Example. 0. Let $S = (x (\lambda a. b. z) = z,$
\[ x (\lambda a. b. a) \Omega = y), (x)).
$S$ is regular, PFR and left(S) is right(S)-\text{distinct}.
Hence by 6.1 $S$ is $\beta$-solvable. A possible $\beta$-solution for
$S$ is: $D = \lambda t. t (\lambda a. t \Omega) \Omega$. \[ D = \lambda t. t (\lambda a. t \Omega) \Omega$.
1. Let $S' = (x (\lambda a. b. z) = z, x (\lambda a. b. c. e) \Omega = y), (x)).$
$S'$ is regular, PFR and left(S') is right(S')-\text{distinct}.
Hence by 6.1 $S'$ is $\beta$-solvable. A possible $\beta$-solution for $S'$ is: $D = \lambda t. t \Omega \Omega$.
2. The system $S'' = (x (\lambda a. a) z = z, x (\lambda a. a (a u)) = u,$
\[ x (\lambda a. a (a I y)) = y), (x))$ is not regular (see
5.4.2, 4.11 and 3.1), but it is $\beta$-solvable.

A possible $\beta$-solution for $S^*$ is: $G = \lambda t_1 t_1 I$.

7. Regular SL-systems

We characterize the $\beta$-solvability for regular SL-systems. To construct a $\beta$-solution for such systems we introduce a particular context.

7.0. Notation. Let \((\vec{x}, x_1, \ldots, x_n) \subseteq \Gamma\) Var, and \(e \in Q(\vec{x})\).

1. \(D_{x,e} = (\lambda x_1 \ldots x_n . \{ I \} \mathcal{D}_{x_1} \ldots \mathcal{D}_{x_n} . e\).

We break self-application finding a common solution for an infinite class of systems.

7.1. Lemma. Let \(S = (\Gamma, (\vec{x}, u))\) be a canonical and regular SL-system with equations having form \(u M = z\) where \(u \in \mathcal{F}(\vec{M})\). If \(S\) is PFR and \(\text{left}(S)\) is right($S$)-distinct then \(\exists e \in Q(\vec{x}) \exists F \in A^+ \forall G[ ] \exists \mathcal{D}_{x,e} [ ]\)

\(\forall u M_1 \ldots M_m = z \in S \in \mathcal{F}[G[M_1] \ldots G[M_m] = z].\)

Proof. (sketch) Let \(e' \in Q(\vec{x}, u)\) s.t. \(S\) is $e'$-regular and \(e \in Q(\vec{x})\) be the restriction of \(e'\) to \(\{\vec{x}\}\). In this situation it is possible to show that \(S_1 = ((u \mathcal{D}_{x,e} M_1) \ldots \mathcal{D}_{x,e} M_m) = z\) \(\forall u M_1 \ldots M_m = z \in S\), \(u\) is regular and PFR and \(\text{left}(S_1)\) is right($S_1$)-distinct. Hence, by 6.1, \(S_1\) is $\beta$-solvable. Let \(F\) be a $\beta$-solution for \(S_1\). Then \(G[ ] \exists \mathcal{D}_{x,e} [ ]\).

7.2. Theorem. Let \(S = (\Gamma, X)\) be a canonical and regular SL-system.

Then \(S\) is $\beta$-solvable iff \(S\) is PFR and \(\text{left}(S)\) is right($S$)-distinct.

Proof. (sketch) \((\Rightarrow)\). By 4.9.

\((\Leftarrow)\). Let \(e \in Q(X)\) s.t. \(0. S\) is $e$-regular;

1. \(d = \max\{\max(\deg(M_{e(x,0)})) \mid x M_1 \ldots M_m = z \in \text{left}(S)\}, \max(\deg(M_{e(x,1)})) \mid x \in X\};\)

2. \(\forall x \in X e(x, 3) = d + 1 + \max(\deg(M_{e(x,0)}), \deg(M_{e(x,0)})) \mid x M_1 \ldots M_m = z \in \text{left}(S)\}.

By 5.3 such an \(e\) exists. Let \(x \in X = (\vec{x}) = (x_1, \ldots, x_n).\)

\(\forall x \in X\) let \(S(x) = \{(u M_1 \ldots M_m = z \mid x M_1 \ldots M_m = z \in S\}, \{x, \vec{x}\}\) (\(u\) fresh). Then \(\forall x \in X S(x)\) is regular and PFR and \(\text{left}(S(x))\) is right($S(x)$)-distinct. Hence by 7.1

\(\forall x \in X \exists F_x \in A^+ \forall G[ ] \exists \mathcal{D}_{x,e} [ ]\)

\(\forall u M_1 \ldots M_m = z \in S(x) \in \mathcal{F}[G[M_1] \ldots G[M_m] = z].\)

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

\(A(x) = (\{i, d + 1 + \max(\deg(M_{e(x,0)}), \deg(M_{e(x,0)})) \mid x M_1 \ldots M_m = z \in \text{left}(S)\})\)

\(M_{e(x,0)} = \lambda \vec{t}_1 \vec{t}_2 \vec{t}_3 \vec{t}_4 \vec{b}_1 \vec{b}_2 \vec{b}_3 \vec{b}_4 \) and \(\vec{b}_1 \in \vec{S}\),

\(B(x) = \{(e(x,0) + \max(\deg(M_{e(x,0)}), \deg(M_{e(x,0)})) + (d - 1 + \max(\deg(M_{e(x,0)}, \deg(M_{e(x,0)}))) + (e(x,1) - e(x', 2)) \mid x M_1 \ldots M_m = z \in \text{left}(S)\}\)

\(\mathcal{F}[G[M_1] \ldots G[M_m] = z].\)

7.3. Corollary. Let \(S = (\Gamma, X)\) be a regular SL-system. Then \(S\) is $\beta$-solvable iff there exists a canonical version $S^*$ of $S$ s.t. $S^*$ is PFR and \(\text{left}(S^*)\) is right($S^*$)-distinct.

Proof. By 4.1, 4.9 and 7.2.

7.4. Example. 0. Let \(S = ((x y \Omega, (\lambda a x \Omega) = y, x \Omega \Omega, (\lambda a b z) = z), \{x\})\). \(S\) is regular and \(\text{left}(S)\) is right($S$)-distinct. Hence by 7.2 \(S\) is $\beta$-solvable. A possible $\beta$-solution for $S$ is:

\(D = \lambda y \tau_1 \tau_2 \tau_3 \Gamma(y)\).

1. The system in 4.2 is $\beta$-solvable.
8. Applications

Using SL-systems it is possible to solve systems like those in Ex. 0.1-4.

8.0. Theorem. Let $S = (\Gamma_1 \cup \Gamma_2, (\bar{x}))$ be a system and $\bar{y} \in (\text{Var} - (\bar{x}))$ s.t.:

1. For each $M \in \text{left}(S)$ there exists $Q \in \Lambda$ s.t. $M = (\lambda \bar{x}. \bar{y}) (\bar{x}; \bar{y})$ and $\bar{y} \notin FV(Q)$.
2. Each equation in $\Gamma_1$ has form $x \bar{M} = y (\bar{x}; \bar{y}) \bar{z}$, where $x \in (\bar{x})$, $y \in (\bar{y})$, $\bar{z} \in (\bar{x}, \bar{y})$ and the variables in head(right($\Gamma_1$)) are pairwise distinct.
3. Each equation in $\Gamma_2$ has form $x \bar{M} = z$, where $x \in (\bar{x})$ and $z \in (\bar{x}, \bar{y})$.

Then:

0. If $S$ is $\beta$-solvable iff there exists a canonical version $S^+$ of $S^+$ s.t.: $S^+$ is PFR and left($S^+$) is right($S^+$)-distinct.
1. If $S$ is $\beta$-solvable then it has a $\beta$-solution having nf.

Proof. (sketch) Transforming $S$ into an SL-system (note that $S^+$ is a step in this direction).

8.1. Example. By 8.0 the systems in 0.1-4 are $\beta$-solvable.

8.2. Example. It is well known that many interesting data structures can be represented using (heterogeneous) term algebras (e.g. [BB 85]).

Let $U_1 = \{ \langle A_{j, 1}, \ldots, A_{j, n(j)} \rangle, (i_j, j; A_{j, k(0)}, i, z_j) \rightarrow A_{j, k(0)} \mid i = 1, \ldots, m(j) \}$ be term algebras ($j = 1, 2$).

A partial recursive function from $U_1$ to $U_2$ can be represented in the $\lambda$-calculus solving the system $S = (\Gamma_1 \cup \Gamma_2 \cup \Gamma_3, (f, \bar{g}, \bar{h}, \bar{d}))$, where:

1. (left-invertibility of the constructors)
   $\Gamma_1 = \{ P_j, h \bar{y} (g_{j, i} z_j 1 \ldots z_{k(0), i}(j)) = z_{h(0)} \mid i = 1, \ldots, m(j) \}$ and $h = 1, \ldots, k(j, a(j, i))$ and $j = 1, 2, \ldots$.
   (recognizability of the constructors)
   $\Gamma_2 = \{ d_j \bar{y} (g_{j, i} z_j 1 \ldots z_{k(j, a(j, i))}(j)) = y_0, j, i \mid j = 1, 2 \}$ and $i = 1, \ldots, m(j)$.
   (specification of the function $f$)
   $\Gamma_3 = \{ f \bar{y} (b_0, \bar{y} z_j 1 \ldots z_{k(1, a(1, i))}(1)) = y_{1, i} (f, \bar{g}, \bar{h}, \bar{d}; \bar{y}) 1 \ldots z_{k(1, a(1, i))}(1) \mid i = 1, \ldots, m(1) \}$.

9. Complexity Analysis

Testing the regularity of an SL-system $S$, testing the $\beta$-solubility of $S$ and constructing a $\beta$-solution for $S$ are Polynomial Time task (9.0.0-2). On the other hand we can easily define a class of non regular SL-systems for which the $\beta$-solubility problem is NP-complete (9.2).

We only consider $\lambda$-terms having a finite Böhm-tree. Moreover we assume of having an oracle that given a $\lambda$-term $M$ computes $BT(M)$ (or equivalently we assume that $\lambda$-terms are given in $\beta\Omega$-normal form).

9.0. Proposition. Let $S = (\Gamma, \bar{x})$ be an SL-system.

0. To test if there exists a canonical version $S^*$ of $S$ s.t. $S^*$ is PFR and left($S^*$) is right($S^*$)-distinct (4.7.1 and 4.5) is a Polynomial Time task.

1. To test if $S$ is regular (5.3.1) is a Polynomial Time task.

2. The solution in 7.2 can be constructed in Polynomial Time.

Proof. (sketch) 0.1. All the algorithms involved are Polynomial Time.

2. The construction in 7.2 imitates the checking of the conditions $S$ is regular, PFR and left($S$) is right($S$)-distinct.

To solve systems like those in 8.0 is a Polynomial Time task.
9.1. Proposition. Let $S = (\Gamma_1 \cup \Gamma_2, (\mathcal{R}))$ satisfying the hypotheses of 8.0. To test if $S$ is $\beta$-solvable and to construct a $\beta$-solution for $S$ (if any) are Polynomial Time task.

**Proof.** (sketch) $S$ can be transformed into an SL-system in Polynomial Time. Then the thesis follows from 9.0. □

Even simple looking extensions of the class of the regular SL-systems yield an NP-complete $\beta$-solvability problem.

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

**Proof.** We codify the satisfiability problem for propositional formulas with SL-systems. Let PropForm (PropVar) the set of Propositional Formulas (Variables).

Let $L : \text{PropForm} \rightarrow A$ defined as follows: $L(x) = x$, $L(\neg A) = L(A)^2 U^2_1$, $L(A \lor B) = L(A) L(B) U^2_2$, $L(A \land B) = L(A) U^2_1 L(B)$

(we are representing true ($T$) with $U^2_1$ and false ($F$) with $U^2_2$).

Let $A \in \text{PropForm}$ s.t. $FV(A) = \{x_1, \ldots, x_n\}$. We define:

- $\text{Transl}(A) = \{((L(A) z \in F = z) \cup \{x_1 z z = z \mid i = 1, \ldots, n\}, \{x_1, \ldots, x_n\})\}.$
- $\text{Transl}(A)$ is an SL-system and is $\beta$-solvable iff $A$ is satisfiable. In fact:

$(\Rightarrow).$ Let $D[\_ = (\lambda x_1 \ldots x_n \_ ) D_1 \ldots D_n$ a $\beta$-solution for $\text{Transl}(A)$. Then $\forall \ i \in \{1, \ldots, n\}$ [$D_i = U^2_1$ or $D_i = U^2_2$] and $D[L(A)] = U^2_2$. Choosing $* = \{x_i \mid$ if $D_i = U^2_1$ then $T$ else $F \mid i = 1, \ldots, n\}$ we have $A^* = T$.

$(\Leftarrow).$ Let $* $ s.t. $A^* = T, \forall i \in \{1, \ldots, n\}$ define:

$D_i = \text{if } x_i^* = T$ then $U^2_1$ else $U^2_2$. Then a $\beta$-solution for $\text{Transl}(A)$ is $D[\_ = (\lambda x_1 \ldots x_n \_ ) D_1 \ldots D_n$.

Define: $\text{SLP} = \{\text{Transl}(A) \mid A \in \text{PropForm}\}$. Then the $\beta$-solvability problem for the systems in SLP is NP-complete.

Note that the systems in SLP are not regular. □

10. Conclusions

Though the $\beta$-solvability problem for SL-systems (2.0) is undecidable (3.1) there is an interesting class of SL-systems (5.3) definable in Polynomial Time (9.0.1) for which the $\beta$-solvable problem is decidable (7.3) in Polynomial Time (9.0.2). This class yields (8.0, 9.1) an equational programming language in which constraints (e.g. like those in 0.2-4, 8.2) on the code generated by the compiler can be specified by the user, properties of data structures can be described in an abstract way (e.g. as in 0.2-4, 8.2), the $\lambda$-terms representing the programs have normal form and the inverse functions of the constructors (of a data structure) are always one shot (e.g. as in 0.3, 8.2). To widen the language introduced seems to be next step.

Acknowledgements

I am grateful to Rick Statman for the many helpful discussions on the topics of this paper and for his valuable suggestions. Without some of the questions I received from Rick Statman Section 9 would not be here. I am also grateful to Mariangela Dezani for her questions and comments about a preliminary version of this paper.

I thank the referees for their useful remarks and suggestions.

References

[Bar 84] Barendregt, H.P., The lambda calculus, North Holland, 1984


[BD 74] Böhm, C. and Dezani-Ciancaglini, M., Combinatorial problems, combinator equations and normal forms, LNCS 14, 1974, 185-199

[Ber 83] Berarducci, A., Program. funz. e rappres. in alcuni sistemi di logica combinatoria, Tesi di Laurea in Matematica, 1985 (supervisor C. Böhm)


[BT 87] Böhm, C. and Tronci, E., X-Separability and Left-Invertibility in Lambda-Calculus, LICS 87, Computer Soc.of the IEEE, pp.320-328

[BT 91] Böhm, C., Tronci, E., About systems of equations, X-separability and left-invertibility in the $\lambda$-calculus, Inf. and Comp. 90, 1-32 (1991)


[MZ 83] Margaria, I., Zacchi, M., Right and Left Invertibility in $\lambda$-$\beta$-calculus, RAIRO Th.Inf. 17, 1983, 71-88


[Sta 87] Statman, R., Private communication