Contents

Second-order unification and type inference for Church-style polymorphism[1]

Aleksy Schubert
alx@mimuw.edu.pl

January, 1997

Abstract

We present a proof of the undecidability of type inference for the Church-style system $\mathcal{F}$. For this, we consider the second-order unification problem. The natural reduction from unification to type inference leads to certain, quite strong, restriction on instances — arguments of variables cannot contain variables. This requires another proof of the undecidability of the second-order unification since known results use variables in arguments of other variables. We give such a proof in the present paper. Moreover, our proof uses elementary techniques, which is important from the methodological point of view, because the Goldfarb's proof [Gol81] highly relies on the undecidability of the tenth Hilbert's problem.

1 Introduction

The system $\mathcal{F}$ was independently introduced by Girard [Gir72] and Reynolds [Rey74] as an extension of the simply-typed λ-calculus introduced by H. B. Curry [Cur69]. In both of the papers, the system was presented originally in the Church paradigm. Reynolds' motivation came from the programming. From this point of view, the problem of type inference, called also "type reconstruction" (given term M, can we find a base Γ and a type τ such that $\Gamma \vdash M : \tau$?) is one of the most interesting.

The type inference problem was studied mostly for the Curry-version of the system $\mathcal{F}$. It has already been proven undecidable by Wells [Wel94]. The type inference for the Church version of the system is considered trivial (you can get such an impression reading for example p. 176 in [Wel94]) since, as normally assumed, a Church-style term contains all the relevant type information. However, the Church-style polymorphic λ-calculus is often defined as follows

$$\begin{array}{rcl} \mathbf{\Lambda}_{\mathcal{F}} & = & V \mid \mathbf{\Lambda}_{\mathcal{F}}\mathbf{\Lambda}_{\mathcal{F}} \mid \mathbf{\Lambda}_{\mathcal{F}}\mathbf{T} \mid \lambda V : \mathbf{T}\mathbf{\Lambda}_{\mathcal{F}} \mid \Lambda \mathbf{V}\mathbf{\Lambda}_{\mathcal{F}} \\ \mathbf{T} & = & \mathbf{V} \mid \mathbf{T} \to \mathbf{T} \mid \forall V \mathbf{T} \end{array}$$

[1]This work has been partly supported by KBN 8 T11C 034 10 grant.

where V is the set of type variables and $\mathbf{V}$ is the set of individual variables (see e.g. [Bar92]). Since individual variables have no type annotations, the type reconstruction problem is no longer trivial. In fact, contrary to the common belief, it occurs that the problem is undecidable.

As it is written by Wells ([Wel94]): *"Multiple stratifications of the system $\mathcal{F}$ were proposed which restrict some parameter of derivations in $\mathcal{F}$ to finite values, e.g. depth of bound type variable from binding quantifier [GR91], the "rank" of polymorphic types (introduced in [Lei83] and further studied in [KT92, KW94])), and the number of generations of instantiation of quantifiers themselves introduced by instantiation [Lei91]."* Our proof concerning type inference applies to the restricted system proposed by Leivant. It is worth noting that it is still an open question wheather the method of [Wel94] applies to this calculus in the Curry-style.

A problem similar to the type inference problem for the Church version of the system $\mathcal{F}$ has been proposed, called *partial type reconstruction* [Boe85]. The problem consist in reconstructing of type information for polymorphic terms with partially erased type annotations

$$\mathbf{\Lambda}'_{\mathcal{F}} \quad = \quad V \mid \mathbf{\Lambda}'_{\mathcal{F}}\mathbf{\Lambda}'_{\mathcal{F}} \mid \mathbf{\Lambda}'_{\mathcal{F}}\mathbf{T} \mid \mathbf{\Lambda}'_{\mathcal{F}}[\,] \mid \lambda V : \mathbf{T}\mathbf{\Lambda}_{\mathcal{F}}' \mid \lambda V : [\,]\mathbf{\Lambda}_{\mathcal{F}}' \mid \Lambda\mathbf{V}\mathbf{\Lambda}_{\mathcal{F}}'$$

This problem and related problem of type reconstruction was studied further in [Pfe88, Pfe93]. F. Pfenning in his two papers on the topic points out the connection between the higher-order unification problem and the partial type inference.

In order to show the undecidability of type inference, we deal with the second-order unification. This problem has been proven undecidable by Goldfarb in [Gol81]. The natural reduction of the unification problem to the type inference problem requires restricted form of instances of the second-order unification — arguments of variables cannot contain variables. This encouraged us to show another proof of the undecidability of the second-order unification problem since techniques used in the Goldfarb's proof require variables in arguments of variables. Moreover, the Goldfarb's proof relies on the undecidability of the tenth Hilbert's problem which uses sophisticated number-theoretical properties. Our proof is done by elementary reduction of the halting problem for two-counter automata, so the proof is methodologically simpler.

Contributions of the paper. There are two main contributions of the paper — a proof of undecidability of the type inference for the Church-style system $\mathcal{F}$ and a new elementary proof of the undecidability of the second-order unification problem. Our approach implies that the type inference problem for relevant version of Leivant's system [Lei91] is undecidable, too.

The undecidability of type inference in the system $\mathcal{F}$ gives a nice completion of known results for the various problems concerning types in the system $\mathcal{F}$. In particular, it shows surprising result of the undecidability for type inference in the system $\mathcal{F}$ and it sharpens known results on the partial type inference for the system (it implies undecidability of a very special case of the problem). From a more practical point of view, the result shows that there is no automatic method of obtaining the type for fully typed functional polymorphic program with some free variables of unknown type.

The result for the second-order unification sharpens the classical result by Goldfarb [Gol81] (we deal with simple instances) and gives a proof that uses only elementary techniques (reduction from the halting problem for two-counter automata). This is methodological simplification of the undecidability proof for second-order unification problem since the Goldfarb's proof relies on the undecidability of the tenth Hilbert's problem. The undecidability of the Hilbert's problem relies heavily on sophisticated number theoretic properties.

We conjecture that the techniques used by Farmer in [Far91] apply to our case (narrowing down the number of function variables and function constants). An appropriate modification of the approach would give the undecidability of the most restricted case of second-order unification. This issue is a work in progress, though.

Organisation of the paper. This paper is organised as follows. In Section 2, we present basic definitions and notation related to two-counter automata and second-order unification. Section 3 contains the proof of undecidability of second-order unification for simple instances. Section 4 describes a reduction of the type inference problem to the unification problem.

The full version of the paper is ftp-able by anonymous ftp from the machine `zls.mimuw.edu.pl`. The paper is contained in the file `pub/alx/simple.dvi.gz`.

Acknowledgements. First of all, I would like to thank prof. P. Urzyczyn for his encouragement to deal with these problems and for many discussions concerning the paper, G. Grudziński for sharing of his vitality, and R. Maron for discussions on the second-order unification. Thanks also for G. Dowek and T. Nipkow for remarks concerning the presentation.

2 Preliminaries

In this section, we present basic definitions and notation related to two-counter automata and second-order unification.

2.1 Two-counter automata

Let $\mathcal{O} = \{\mathrm{NOP}, \mathrm{INC}, \mathrm{DEC}\}$ and $\mathcal{T} = \{\mathrm{Z}, \mathrm{NZ}\}$. Elements of the set $\mathcal{O}$ denote operations on counters

- NOP — no operation;
- INC — increment the counter;
- DEC — decrement the counter

and elements of the set $\mathcal{T}$ denote states of a counter

- Z — the counter is zero;
- NZ — the counter is not zero.

Definition 1 `(two-counter automaton)`
A two-counter automaton $\mathcal{A}$ is a quadruple

$$\langle Q, q_s, q_f, \delta, C1, C2 \rangle$$

where Q is a set of states, q_s — a distinguished starting state, q_f — a distinguished final state which is terminal (i.e. which has no successor), $\delta \subseteq \mathcal{T}^2 \times Q \times \mathcal{O}^2 \times Q$ — a next-step relation, and $C1$ and $C2$ — two registers that may contain arbitrary natural numbers.

Remark 2.1 *For technical purposes, we restrict our two-counter automata so that they are*

1. *deterministic (δ is a function $\delta : \mathcal{T}^2 \times Q \to \mathcal{O}^2 \times Q$);*
2. *the only terminal state is q_f;*
3. *in q_f both counters are equal to zero;*
4. *there is no pair of states q_1, q_2 that $\delta \cap \mathcal{T}^2 \times \{q_1\} \times \mathcal{O}^2 \times \{q_2\}$ has the cardinality greater than 1;*
5. *for each state q we have $\delta \cap \mathcal{T}^2 \times \{q\} \times \mathcal{O}^2 \times \{q\} = \emptyset$.*

Note that these restrictions do not affect the following property

Theorem 2.2 `(undecidability of stop)`
The "halting problem" for two-counter automata is undecidable.

Proof:
See [HU79]. □

For a given automaton $\mathcal{A}$, we write

$$\mathrm{TC1}, \mathrm{TC2}, q_1 \to_\delta \mathrm{A1}, \mathrm{A2}, q_2$$

where TC1, TC2 $\in \mathcal{T}, q_1, q_2 \in Q$, and A1, A2 $\in \mathcal{O}$, to denote that the tuple $(\mathrm{TC1}, \mathrm{TC2}, q_1, \mathrm{A1}, \mathrm{A2}, q_2) \in \delta$ i.e. that if $\mathcal{A}$ ascertains in the state q_1 that TC1 is true for the first counter and TC2 for the second one then it performs action A1 on the first counter, action A2 on the second, and shifts to the state q_2.

Remark 2.3 *We assume existence of some linear order on rules in δ so that each production of the form*

$$\mathrm{TC1}, \mathrm{TC2}, q_1 \to_\delta \mathrm{A1}, \mathrm{A2}, q_2$$

has its own unique number.

We associate with each step of the automaton $\mathcal{A}$ an instantaneous description of the automaton

$$\langle q, n, n' \rangle$$

where $q \in Q$, $n, n' \in \mathbb{N}$. The intuitive meaning is that, before performing the step, the automaton is in the state q, its first counter contains n, and its second counter contains n'. We write

$$\langle q_1, n_1, n_1' \rangle \to_\delta \langle q_2, n_2, n_2' \rangle$$

to indicate that if the current ID of $\mathcal{A}$ is $\langle q_1, n_1, n_1' \rangle$ then the next ID is $\langle q_2, n_2, n_2' \rangle$.

2.2 Second-order unification

Most of the notation in the subsection is taken from the paper by Farmer [Far91].

Definition 2 `(a term language)`
Term language *is a quadruple* $L = \langle \mathrm{IC}_L, \mathrm{IV}_L, \mathrm{FC}_L, \mathrm{FV}_L \rangle$ *where:*

- IC_L *is a set of* individual constants *(denoted by a, b, c, a', etc.);*
- IV_L *is a set of* individual variables *(denoted by x, y, z, x', etc.);*
- FC_L *is a set of* functional constants *(denoted by f, g, h, f', etc.); and*
- FV_L *is a set of* functional variables *(denoted by F, G, H, F', etc.).*

Each functional item has fixed arity ≥ 1, and the sets mentioned in Definition 2 are mutually disjoint.

Let L be an arbitrary term language. We say that L *contains* a symbol s if $s \in \mathrm{IC}_L \cup \mathrm{IV}_L \cup \mathrm{FC}_L \cup \mathrm{FV}_L$.

Definition 3 `(terms)`
The terms *of* L *(*L-terms*) are defined inductively by*

- *Each* $d \in \mathrm{IC}_L \cup \mathrm{IV}_L$ *is an* L*-term.*
- *If* $d \in \mathrm{FC}_L \cup \mathrm{FV}_L$ *has arity* $n \geq 1$ *and* $t_1 \ldots t_n$ *are* L*-terms, then the term* $d(t_1, \ldots, t_n)$ *is an* L*-term.*

Let $\mathrm{PM}_L = \{w_1, w_2, w_3, \ldots\}$ be an infinite set of symbols not contained in L called *place markers*. (We shall assume that, for any other term language $L', \mathrm{PM}_L = \mathrm{PM}_{L'}$.) The language L^* differs from L by having the members of PM_L as additional symbols.

Definition 4 `(`L^*`-terms)`
L^*-terms *are defined inductively by*

- *Each* $d \in \mathrm{IC}_L \cup \mathrm{IV}_L \cup \mathrm{PM}_L$ *is an* L^**-term.*
- *If* $d \in \mathrm{FC}_L \cup \mathrm{FV}_L$ *has arity* $n \geq 1$ *and* $t_1 \ldots t_n$ *are* L^**-terms, then* $d(t_1, \ldots, t_n)$ *is an* L^**-term.*

The arity *of an* L^**-term, written* $\mathrm{rank}(t)$ *is the largest* n *such that* w_n *occurs in* t*.*

Intuitively, L^*-terms of arity n represent n-ary functions.

Definition 5 `(a simple term)`
A term s *is a* simple term *iff for each occurrence of a functional variable in* s *in a context that has the form* $Fu_1 \ldots u_n$*, for some terms* $u_1 \ldots u_n$*, none of* $v \in \mathrm{IV}_L \cup \mathrm{FV}_L$ *occurs in* u_j *for* $j \in \{1, \ldots, n\}$*.*

We denote L^*-terms (and hence L-terms) by r, s, t, r' etc. $t[t_1, \ldots, t_n]$ denotes the result of simultaneously replacing each occurrence of w_i in t with t_i for all i with $1 \leq i \leq n$. For a given term t we denote by $\mathrm{var}(t)$ the set of all variables contained in t. An L^*-term is *closed* if it contains no variables.

Definition 6 `(a substitution)`
A substitution in *L is a function σ with a finite domain* $\mathrm{dom}(\sigma) \subseteq \mathrm{IV}_L \cup \mathrm{FV}_L$ *which maps individual variables to L-terms and n-ary function variables with $n \geq 1$ to L^*-terms of arity $\leq n$.*

A substitution σ applied to $v \in \mathrm{dom}(\sigma)$ is written as $v\sigma$ instead of $\sigma(v)$. We shall assume that $x\sigma \neq x$ and $F\sigma \neq F(w_1, \ldots, w_n)$ for all substitutions σ and all $x, F \in \mathrm{dom}(\sigma)$. A substitution σ is displayed as $\{v_1 \to s_1, \ldots, v_m \to s_m\}$, where $\mathrm{dom}(\sigma) = \{v_1, \ldots, v_m\}$ and σ maps v_i to s_i for each i with $1 \leq i \leq m$. Substitutions are denoted by π, σ, τ, π', etc. A substitution σ is *closed* if $v\sigma$ is closed for all $v \in \mathrm{dom}(\sigma)$. Let $\mathrm{sub}(L)$ be the set of substitutions in L.

Definition 7 `(a result of a substitution)`
Let $\sigma = \{v_1 \to s_1, \ldots, v_m \to s_m\} \in \mathrm{sub}(L)$. The result of applying σ to an L^*-term t is defined inductively by

- *If $t \in \mathrm{IC}_L \cup \mathrm{IV}_L \cup \mathrm{PM}_L$ but $t \notin \{v_1, \ldots, v_m\}$, then $t\sigma = t$.*
- *If $t \in \mathrm{IV}_L$ and $t = v_i$ for some i with $1 \leq i \leq m$, then $t\sigma = s_i$.*
- *If $t = d(t_1, \ldots, t_n)$ where $d \in \mathrm{FC}_L \cup \mathrm{FV}_L$ but $d \notin \{v_1, \ldots, v_m\}$, then $t\sigma = d(t_1\sigma, \ldots, t_n\sigma)$.*
- *If $t = F(t_1, \ldots, t_n)$ and $F = v_i$ for some i with $1 \leq i \leq m$, then $t\sigma = s_i[t_1\sigma, \ldots, t_n\sigma]$.*

The *composition of substitutions* σ and τ is the substitution denoted by $\sigma\tau$ such that $v(\sigma\tau) = (v\sigma)\tau$ for all $v \in \mathrm{IV}_L \cup \mathrm{FV}_L$. It is easy to see that parentheses are not needed when writing $t\sigma\tau$ and $\sigma\tau\pi$.

Definition 8 `(a unification expression)`
A unification expression (u-expr) in *L is any finite set E of the form $\{\langle s_i, t_i\rangle \mid i \in I\}$ where s_i and t_i are L-terms for each $i \in I$. For $\sigma \in \mathrm{sub}(L)$, let $E\sigma$ be the u-expr $\{\langle s_i\sigma, t_i\sigma\rangle \mid i \in I\}$. Each pair $\langle s_i\sigma, t_i\sigma\rangle$ is called* unification equation.

If terms s_i and t_i for each $i \in I$ are simple termsthen we say that E is a simple instance of the unification problem *(simple u-expr).*

We associate with u-expr(L) the set of all unification expressions in L and with u-expr$_\mathrm{s}(L)$ the set of all simple unification expressions in L.

For a given $E \in \text{u-expr}(L)$ we denote by $\mathrm{var}(E)$ the set of all variables occurring in terms of E.

Remark 2.4 *It is worth mentioning that for languages which contain at least one function symbol of arity greater than 1 it is enough to consider unification expressions with only one pair.*

Definition 9 (a unifiable expression)
Let $E = \{\langle s_i, t_i\rangle \mid i \in I\} \in$ *u-expr*(L). *The expression* E *is* unifiable *if there is a substitution* σ *in* L *such that* $s_i\sigma = t_i\sigma$ *for each* $i \in I$. *Such a substitution is called a* unifier *of* E.

If $\{\langle s_i, t_i\rangle\}$ *is unifiable expression then we say that* the equation $\langle s_i, t_i\rangle$ *is unifiable.*

Definition 10 (the unification problem)
The second-order unification problem *is the problem of determining, given* L *and* $E \in$ *u-expr*(L), *whether or not* E *is unifiable.*

If $E \in$ *u-expr*$_{\mathrm{s}}(L)$ *then we call the problem* second-order unification problem for simple instances.

3 The undecidability of unification

In this section, we work with languages that contain only one function constant symbol $\rightarrow$ of arity 2 and at least one constant symbol c_0. We write this symbol as an infix operator, for example $t \rightarrow s$.

3.1 Preliminaries

To simplify operation on such terms, we introduce some notation.

Definition 11 (subterm indication)
Let $\omega \in \{L, R\}^*$. *We define by induction subterm of* t *indicated by* ω, *written* $\omega(t)$, *and the notion of applicability of* ω *as*

- $\epsilon(t) = t$ *and* ϵ *is applicable to* t;
- *if* ω *is applicable to* t_1 *then* ωL *is applicable to* $t_1 \rightarrow t_2$ *and* $\omega L(t_1 \rightarrow t_2) = \omega(t_1)$;
- *if* ω *is applicable to* t_2 *then* ωR *is applicable to* $t_1 \rightarrow t_2$ *and* $\omega R(t_1 \rightarrow t_2) = \omega(t_2)$

where ϵ *denotes the empty word.*

If it does not lead to confusion we shall omit parentheses.

Definition 12 (replacing of a subterm)
Let $\omega \in \{L, R\}^*$. *We define inductively replacing of a subterm at* ω *in* t *by the term* u, *written* $t[\omega \leftarrow u]$, *by*

- $t[\epsilon \leftarrow u] = u$;
- $(t_1 \rightarrow t_2)[\omega L \leftarrow u] = t_1[\omega \leftarrow u] \rightarrow t_2$;
- $(t_1 \rightarrow t_2)[\omega R \leftarrow u] = t_1 \rightarrow t_2[\omega \leftarrow u]$;

To apply results of Subsection 3.2 to the machinery in Subsection 3.3, we need the following fact.

Fact 3.1 (non-affecting unifier)
If σ unifies $\{\langle s, t[\omega \leftarrow u\sigma]\rangle\}$, where $\omega \in \{L, R\}^*$ and s, t, u are some terms, and $\mathrm{var}(u) \cap (\mathrm{var}(s) \cup \mathrm{var}(t[\gamma \leftarrow c_0])) = \emptyset$, where c_0 is some constant, then σ unifies $\{\langle s, t[\omega \leftarrow u]\rangle\}$

Proof:
Induction on the length of ω. □

3.2 The co-ordination equation

In this section, we present a unification equation that esures that a solution of instances of the unification problem we present later has a certain form. This particular form is such that some subterms are codes of configurations of two-counter automata.

In this and the next subsections, we deal with a language L such that

- $\{c^i_q \mid q \in Q \text{ and } i \in \{1,2\}\} \cup \{c_0, \ldots, c_7\} \subseteq \mathrm{IC}_{L_A}$ for a certain distinguished set Q;
- $G \in \mathrm{FV}_{L_A}$ where arity of G is $3|Q|+2$ ($|Q|$ is the cardinality of Q)

We assume that all the symbols above are pairwise distinct. We need a few definitions.

Definition 13 (a pre-code of a counter)
We define a pre-code of a counter $\mathrm{precntr}(n,t)$ *where n is a natural number and t is a term by*

- $\mathrm{precntr}(0,t) = t$;
- $\mathrm{precntr}(n+1,t) = c_1 \to \mathrm{precntr}(n,t)$;

Definition 14 (a pre-code and a code of ID)
A pre-code of ID *(instantaneous description) is an L-term of the form $l_1 \to l_2$ such that $l_i = c_0 \to \mathrm{precntr}(n_i, c^i_{q_i})$ for some natural numbers n_i, states q_i, and $i \in \{1,2\}$*

Given an instantaneous description $\langle q, n, n'\rangle$ we define a code of the ID *$\langle q, n, n'\rangle$ by*

$$t_{\langle q,n,n'\rangle} = (c_0 \to \mathrm{precntr}(n, c^1_q)) \to (c_0 \to \mathrm{precntr}(n', c^2_q))$$

Remark 3.2 *Of course, each code of ID corresponds to some ID.*

Definition 15 (a co-ordination equation)
The co-ordination equation *for the term t is the following pair of terms*

$$\langle s, u\rangle = \langle G f_1 \ldots f_{2r} g_1 \ldots g_r h_1 h_2,\ t \to G f'_1 \ldots f'_{2r} g'_1 \ldots g'_r h'_1 h'_2\rangle$$

where $r = |Q|$, t is L-term such that for each i if RL^i is applicable to r then $RL^i t$ is a pre-code of ID and there exists k such that $L^k t = c_5$ and for $i \in \{1, \ldots, r\}$

$$f_{2i-1} = c_1 \to c^1_{q_i}, \quad f'_{2i-1} = c^1_{q_i}, \quad f_{2i} = c_1 \to c^2_{q_i}, \quad f'_{2i} = c^2_{q_i},$$

$$g_i = (c_0 \to c^1_{q_i}) \to (c_0 \to c^2_{q_i}), \quad g'_i = c_4$$

$$h_1 = c_5 \to c_4, \quad h'_1 = c_5$$

$$h_2 = c_5 \to c_6, \quad h'_2 = c_6$$

Let σ be a unifier of the defined above co-ordination equation. In this subsection, we fix this notation and all the notation from Definition 15. In particular, we can denote the term t by Lu.

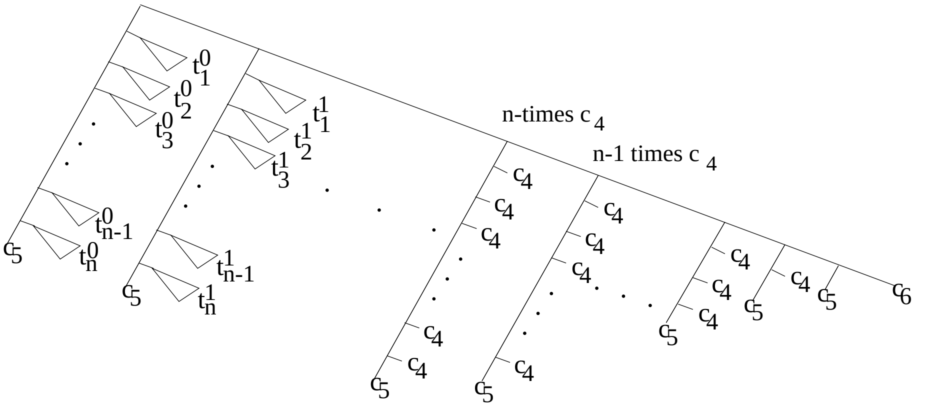
Next we introduce a reduction system.

Definition 16 (a reduction system for co-ordination)
The reduction system $\to_{Co}$ is defined by the set of rewrite rules

$$\{f_i \to_{Co} f'_i \mid i \in \{1,\dots,2r\}\} \cup \{g_i \to_{Co} g'_i \mid i \in \{1,\dots,r\}\}$$

When we write $t' \to^{\#} t''$ for some reduction system $\to$ we understand that t'' is a result of simultaneous performing of some redexes in t'.

Our intention is to show that if there exists a solution σ of the co-ordination equation then there exists one such that the term $G\sigma\vec{f}\vec{g}h_1h_2$ has the form



where for each i the sequence $\{t^j_i\}_j$ has the property that $t^j_i \to^{\#}_{Co} t^{j+1}_i$.

It is obvious that applying the reduction $\to_{Co}$ to a pre-code of ID results in either c_4 or another pre-code of ID.

Fact 3.3 (codes of ID in reduction)
Let $t_1,\dots,t_n$ be a sequence of terms such that for each $i < n$ we have $t_i \to^{\#}_{Co} t_{i+1}$ and $t_n = c_4$. There exists an element $q \in Q$ such that each pre-code of ID in the sequence is a code of ID with the state q.

Proof:
Induction on n. □

Fact 3.4 (the reduction in the result of unification)
For each i and each j if RL^jR^i is applicable to $s\sigma$ and $RL^jR^i(s\sigma)$ is a pre-code of ID then RL^jR^{i+1} is applicable to $s\sigma$ and $RL^jR^i(s\sigma) \to^{\#}_{Co} RL^jR^{i+1}(s\sigma)$.

Proof:
Let i and j be integers such that RL^jR^i is applicable to $s\sigma$ and $RL^jR^i(s\sigma)$ is a pre-code of ID. The word RL^jR^i is applicable to $G\sigma$, too, since no argument of G in s contains a pre-code of ID as a proper subterm. Since $RL^jR^i(s\sigma)$ is a pre-code of ID and no argument of G in s is neither c_0 nor c_1, there are three possibilities

1. $RL^jR^i(G\sigma)$ contains no place marker. Then $RL^jR^i(s\sigma) = RL^jR^i(G\sigma)$. Since $u = t \to G\ldots$, we get $RL^jR^i(G\sigma) = RL^jR^{i+1}(u\sigma)$ and then as $s\sigma = u\sigma$, we have $RL^jR^{i+1}(u\sigma) = RL^jR^{i+1}(s\sigma)$ so indeed in this case $RL^jR^i(s\sigma) \to^{\#}_{Co} RL^jR^{i+1}(s\sigma)$

2. $RL^jR^i(G\sigma)$ is a place marker. Since $RL^jR^i(s\sigma)$ is a pre-code of ID, there is only one possibility — $RL^jR^i(G\sigma) = w_l$ for $l \in \{2r+1, \ldots, 3r\}$. The latter implies that $RL^jR^i(G\sigma\vec{f}\vec{g}h_1h_2) = g_l$ and $RL^jR^i(G\sigma\vec{f}'\vec{g}'h'_1h'_2) = g'_l$, so $RL^jR^i(G\sigma\vec{f}\vec{g}h_1h_2) \to^{\#}_{Co} RL^jR^i(G\sigma\vec{f}'\vec{g}'h'_1h'_2)$ and then $RL^jR^i(s\sigma) \to^{\#}_{Co} RL^jR^{i+1}(u\sigma)$ and then since $RL^jR^{i+1}(u\sigma) = RL^jR^{i+1}(s\sigma)$ we get that $RL^jR^i(s\sigma) \to^{\#}_{Co} RL^jR^{i+1}(s\sigma)$.

3. $RL^jR^i(G\sigma)$ contains some place markers as proper subterms. We have that $RL^jR^i(s\sigma) = (c_0 \to \mathrm{precntr}(l_1, x^1_{m_1})) \to (c_0 \to \mathrm{precntr}(l_2, y^2_{m_2}))$ where x, y are either f or g and $m_1 \in \{1, 3, \ldots, 2r-1\}$ and $m_2 \in \{2, 4, \ldots, 2r\}$. This implies $R^{l_1}LRL^jR^i(G\sigma) = w_{m_1}$ or $R^{l_2}RRL^jR^i(G\sigma) = w_{m_2}$. We proceed for the case $R^lLRL^jR^i(G\sigma) = w_{m_1}$. The equality $R^lLRL^jR^i(G\sigma) = w_{m_1}$ implies that the subterm $R^lLRL^jR^i(G\sigma\vec{f}\vec{g}h_1h_2) = f_{m_1}$ and the subterm $R^lLRL^jR^i(G\sigma\vec{f}'\vec{g}'h'_1h'_2) = f'_{m_1}$, so
$$LRL^jR^i(G\sigma\vec{f}\vec{g}h_1h_2) \to^{\#}_{Co} LRL^jR^i(G\sigma\vec{f}'\vec{g}'h'_1h'_2)$$
and then $LRL^jR^i(s\sigma) \to^{\#}_{Co} LRL^jR^{i+1}(u\sigma)$ and then since the equality $RL^jR^{i+1}(u\sigma) = RL^jR^{i+1}(s\sigma)$ holds we get
$$LRL^jR^i(s\sigma) \to^{\#}_{Co} LRL^jR^{i+1}(s\sigma).$$
There exists possibility that $R^{l_2}RRL^jR^i(G\sigma) = f_{m_1}$, too. We proceed in the case analogously to the (1) case of the proof and get $LRL^jR^i(s\sigma) \to^{\#}_{Co} LRL^jR^{i+1}(s\sigma)$, too.

   The case for $R^lRRL^jR^i(G\sigma) = w_{m_2}$ is similar and we get the reduction $RRL^jR^i(s\sigma) \to^{\#}_{Co} RRL^jR^{i+1}(s\sigma)$ holds. This combined with the result for $LRL^jR^i(s\sigma)$ gives $RL^jR^i(s\sigma) \to^{\#}_{Co} RL^jR^{i+1}(s\sigma)$.

□

Fact 3.5 (behaviour for c_4 and c_5)
Let i and j be integers such that L^jR^i is applicable to $s\sigma$. If $L^jR^i(s\sigma) = c_5 \to c_4$ then either $L^jR^{i+1}(s\sigma) = c_5 \to c_4$ or $L^jR^{i+1}(s\sigma) = c_5$.

Moreover, if $RL^jR^i(s\sigma) = c_4$ and $LL^jR^i(s\sigma) \neq c_5$ then $RL^jR^{i+1}(s\sigma) = c_4$.

Proof:
A priori there are two cases — either the subterm $L^jR^i(G\sigma) = w_m$ for some m or the subterm $L^jR^i(G\sigma) = s_1 \to s_2$.

The former case is possible only for $m = 3r + 1$ since only that argument of G in s contains occurrence of c_4. As this argument is $c_5 \to c_4$, we get that $LL^jR^i(s\sigma) = c_5$. Since $3r + 1$st argument of G in u is c_5, we get that $L^jR^{i+1}(u\sigma) = c_5$, so $L^jR^{i+1}(s\sigma) = c_5$.

If $L^jR^i(G\sigma) = s_1 \to s_2$ and $s_1 = c_5$ then, since there is no argument of G in s equal to c_4, we get that $s_2 = c_4$. We get then that $L^jR^i(G\sigma) = L^jR^{i+1}(u\sigma)$ since $u = t \to (G\ldots)$. As $s\sigma = u\sigma$, we have $L^jR^{i+1}(u\sigma) = L^jR^{i+1}(s\sigma)$ so indeed in this case $L^jR^i(s\sigma) = c_5 \to c_4$.

If $L^jR^i(G\sigma) = s_1 \to s_2$ and $s_1 \neq c_5$ then, since there is no argument of G in s equal to c_4, we have $s_2 = c_4$. We get then that $RL^jR^i(G\sigma) = RL^jR^{i+1}(u\sigma)$ since $u = t \to (G\ldots)$. As $s\sigma = u\sigma$, we have $RL^jR^{i+1}(u\sigma) = RL^jR^{i+1}(s\sigma)$ so indeed in this case $RL^jR^i(s\sigma) = c_4$. □

Fact 3.6 `(behaviour for` c_5 `and pre-codes)`
For each i and j such that $L^{j+1}R^i(s\sigma) = c_5$ and $RL^jR^i(s\sigma)$ is a pre-code of ID, we have $L^{j+1}R^{i+1}(s\sigma) = c_5$.

Proof:
Since there is no argument e of G in s such that c_5 and a pre-code of ID are simultanoulsly contained in e as subterms, we have $L^jR^i(G\sigma) = t_1 \to t_2$ for some terms t_1, and t_2. Since no argument of G in s is c_5, we have $t_1 = c_5$. Then $L^{j+1}R^{i+1}(u\sigma) = c_5$ and then $L^{j+1}R^{i+1}(s\sigma) = c_5$. □

Fact 3.7 `(`c_4 `is obligatory)`
For each i such that RL^i is applicable there exist j_1 such that $L^{i+1}R^{j_2}(s\sigma) = c_5$, and $j_2 \geq j_1$ such that $RL^iR^{j_2}(s\sigma) = c_4$.

Proof:
By the definition of co-ordinating equation, there exists k such that $L^{k+1}(s\sigma) = c_5$. We prove this fact by descending induction on i beginning at k.

1. $i = k$. Obviously, $j_1 = 0$. Since $L^k(s\sigma)$ is a pre-code of ID, we get by Fact 3.5 and finiteness of $s\sigma$ that there exists m such that we have the reduction $RL^kR^j(s\sigma) \to^{\#}_{Co} RL^kR^{j+1}(s\sigma)$ for $j < m$ and the reduction $RL^kR^j(s\sigma) \not\to^{\#}_{Co} RL^kR^{j+1}(s\sigma)$ for $j = m$. Fact 3.6 implies that for $j \leq m$ we have $L^{k+1}R^j(s\sigma) = c_5$. Moreover, $L^kR^m(s\sigma)$ must be c_4, because otherwise by properties of reduction $\to_{Co}$, this subterm must be a pre-code of ID and then by Fact 3.4 we have $L^kR^m(s\sigma) \to^{\#}_{Co} L^kR^{m+1}(s\sigma)$ which contradicts our assumption on m.

2. $i < k$. By the induction hypothesis, there exists j_1' such that $L^{i+2}R^{j_1'}(s\sigma) = c_5$ and $j_2' \geq j_1'$ such that $RL^{i+1}R^{j_2'}(s\sigma) = c_4$. By Fact 3.5 and finitness of $s\sigma$, we get that there exists $j_1 > j_2'$ such that $L^{i+1}R^{j_1}(s\sigma) = c_5$. If $L^iR^{j_1}(s\sigma) = c_4$ then $j_2 = j_1$. Otherwise, since $RL^i(s\sigma)$ is a pre-code of ID, so $RL^iR^{j_1}(s\sigma)$ is. By reasoning similar to the one in the case for $i = k$, we get that there exists $j_2 \geq j_1$ such that $RL^iR^{j_2}(s\sigma) = c_4$.

□

Now we prove two main facts of this subsection.

Lemma 3.8 `(pre-codes are codes in a solution)`
For each i, if RL^i is applicable to t then $RL^i t$ is a code of ID.

Proof:
Let i be an integer such that RL^i is applicable to t. Then by Fact 3.4, RL^{i+1} is applicable to $s\sigma$ and $RL^{i+1}(s\sigma)$ is a pre-code of ID. By Fact 3.7, there exists j_2 such that $RL^{i+1}R^{j_2}(s\sigma) = c_4$. Then by Fact 3.4 and Fact 3.5 sequence $\{RL^{i+1}R^l(s\sigma)\}_{0\leq l\leq j_2}$ has the property that there occurs the reduction

$$RL^{i+1}R^l(s\sigma) \rightarrow^{\#}_{Co} RL^{i+1}R^{l+1}(s\sigma)$$

for $l < j_2$ and $RL^{i+1}R^{j_2}(s\sigma) = c_4$. By Fact 3.3, all pre-codes in the sequence are codes of ID for some state q, so the subterm $RL^{i+1}(s\sigma)$ is, and then $RL^i t$, too. □

Fact 3.9 `(the co-ordination equation is unifiable)`
Let τ be a substitution. If the subterm $RL^i t$ is a code of ID for each i such that RL^i is applicable to t then there exists a unifier τ' of the co-ordination equation for t such that $x\tau = x\tau'$ for $x \neq G$.

Proof:
If $t = c_5$ then it is easy to see that $\tau' = \tau[w_{3r+2}/G]$ with $r = |Q|$ is our solution.

If $t \neq c_5$ then, by our assumptions, there exists i_0 such that $L^{i_0}t = c_5$ and for each $i < i_0$ the subterm $RL^i t$ is a code of ID. This implies that there exists j_0 such that for each i there exists a sequence $\{t^i_j\}_{0\leq j\leq j_0}$ such that $t^i_0 = RL^i t$, $t^i_{j_0} = c_4$, and $t^i_j \rightarrow^{\#}_{Co} t^i_{j+1}$. We build another sequence $\{s^i_j\}_{0\leq j\leq j_0}$. Let $t^i_j \rightarrow^{\#}_{Co} t^i_{j+1}$ then there exists some cotext $C[\ldots]$ such that

$$t^i_j = C[f_1, \ldots, f_{2r}, g_1, \ldots, g_r]$$

$$t^i_{j+1} = C[f'_1, \ldots, f'_{2r}, g'_1, \ldots, g'_r]$$

where $r = |Q|$. We put $s^i_j = C[w_1, \ldots, w_{3r}]$.

We construct a term t'

1. for $j \leq j_0$ and $i < i_0$ we put $RL^{i+1}R^j t' = s^i_j$;
2. $R^{j_0+1}t' = t^{i_0}$ where t^{i_0} is recursively defined by
   - $t^1 = w_{3r+1} \rightarrow w_{3r+2}$
   - $t^{j+1} = ((Lt^j) \rightarrow c_4) \rightarrow t^j$

We put $\tau' = \tau[t'/G]$. We shall show that $s\tau' = u\tau'$.

- For $j \leq j_0$ and $i < i_0$ we have $RL^{i+1}R^j(s\tau') = t^i_j$ and, similarly, the term $RL^{i+1}R^j(u\tau') = t^i_j$.

- For $j \leq j_0$ we have $L^{i_0+1}R^j(s\tau') = c_5$, because $L^{i_0+1}R^j(G\tau') = c_5$ and similarly, $L^{i_0+1}R^j(u\tau') = c_5$, because for $j > 0$ we have $L^{i_0+1}R^{j-1}(G\tau') = c_5$ and for $j = 0$ we have $L^{i_0+1}(t) = c_5$.

- For each $l < i_0$ and $i < i_0 - l$ we have $RL^{i+1}R^{j_0+l}(s\tau') = c_4$, because the subterm $RL^{i+1}R^{j_0+l}(G\tau') = c_4$ for $i < i_0 - l - 1$ and the subterm $RL^{i+1}R^{j_0+l}(G\tau') = w_{3r+1}$ for $i = i_0 - l - 1$ and $3r + 1$st argument of G in s is $c_5 \to c_4$. Similarly, $RL^{i+1}R^{j_0+l}(u\tau') = c_4$, because the subterm $RL^{i+1}R^{j_0+l}(G\tau') = c_4$ for $i < i_0 - l$.

- $L^{i_0-l+1}R^{j_0+l}(s\tau') = c_5$, because $L^{i_0-l}R^{j_0+l}(G\tau') = w_{3r+1}$ and $3r + 1$st argument of G in s is $c_5 \to c_4$. Similarly, $L^{i_0-l+1}R^{j_0+l}(u\tau') = c_5$, because $L^{i_0-l}R^{j_0+l-1}(G\tau') = w_{3r+1} \to c_4$ and $3r + 1$st argument of G in u is c_5.

- At last, $R^{j_0+i_0}(s\tau') = c_5 \to c_6$, because $R^{j_0+i_0}(G\tau') = w_{3r+2}$ and $3r+2$nd argument of G in s is $c_5 \to c_6$. Similarly, $R^{j_0+i_0}(u\tau') = c_5 \to c_6$, because $R^{j_0+i_0-1}(G\tau') = w_{3r+1} \to w_{3r+2}$ and $3r + 1$st argument of G in u is c_5 and $3r + 2$nd argument of G in u is c_6.

□

3.3 The instance of unification

Now we are ready to present the instances of unification problem for which we shall show undecidability.

Definition 17 (`a unification generated by an automaton`)
Let $\mathcal{A} = \langle Q, q_s, q_f, \delta, C1, C2\rangle$ be a two-counter automaton. For the automaton we define a language $L_{\mathcal{A}}$ and an instance of second-order unification $E_{\mathcal{A}} \in$ u-expr$_s(L_{\mathcal{A}})$. $L_{\mathcal{A}} = \langle \mathrm{IC}_{L_{\mathcal{A}}}, \mathrm{IV}_{L_{\mathcal{A}}}, \mathrm{FC}_{L_{\mathcal{A}}}, \mathrm{FV}_{L_{\mathcal{A}}}\rangle$ where

- $\mathrm{IC}_{L_{\mathcal{A}}} = \{c_q^i \mid q \in Q and i \in \{1,2\}\} \cup \{c_0, \ldots, c_6, c_7\}$;
- $\mathrm{IV}_{L_{\mathcal{A}}} = \{x_1\}$;
- $\mathrm{FC}_{L_{\mathcal{A}}} = \{\to\}$ *and* $\to$ *has the arity 2;*
- $\mathrm{FV}_{L_{\mathcal{A}}} = \{F, G\}$, *where F has the arity $2 * |\delta| + 2$ and G has the arity $3|Q| + 2$*

The set $E_{\mathcal{A}}$ has four elements

$$\langle s_1, u_1\rangle = \langle Fc_2c_0\ldots c_0, x_1 \to c_2\rangle$$

$$\langle s_2, u_2\rangle = \langle Fc_3c_0\ldots c_0, x_1 \to c_3\rangle$$

$$\langle s_3, u_3\rangle = \langle Fc_7b_1^1\ldots b_k^1 b_1^2\ldots b_k^2 e, (Fab_1'^1\ldots b_k'^1 b_1'^2\ldots b_k'^2 e') \to c_7\rangle$$

$$\langle s_4, u_4\rangle =$$
$$\langle Gf_1\ldots f_{2r}g_1\ldots g_r h_1 h_2, (Fab_1'^1\ldots b_k'^1 b_1'^2\ldots b_k'^2 e') \to Gf_1'\ldots f_{2r}'g_1'\ldots g_r' h_1' h_2'\rangle$$

where $r = |Q|$, $k = |\delta|$, *terms* f_i, f'_i *for* $i \in \{1, \ldots, 2r\}$, g_j, g'_j *for* $j \in \{1, \ldots r\}$, *and terms* h_1, h_2, h'_1, h'_2 *are as in Definition 15. Remaining terms are*

$$a = (c_0 \to c^1_{q_s}) \to (c_0 \to c^2_{q_s})$$

where q_s *is a starting state, and for each element of* δ *of the form*

$$\mathrm{TC1}, \mathrm{TC2}, q_1 \to_\delta \mathrm{A1}, \mathrm{A2}, q_2$$

we put for $i \in \{1, 2\}$ *and* $j \in \{1, \ldots, r\}$

$$b^i_j = c_\epsilon \to c_{q_1}$$

where $\epsilon = 0$ *if* $TCi = Z$ *and* $\epsilon = 1$ *if* $TCi = NZ$*. If* Ai *is DEC then*

$$b'^i_j = c_{q_2}$$

if Ai *is NOP then*

$$b'^i_j = L(b^i_j) \to c_{q_2}$$

if Ai *is INC then*

$$b'^i_j = L(b^i_j) \to (c_1 \to c_{q_2})$$

and at last

$$e = c_5 \to ((c_0 \to c^1_{q_f}) \to (c_0 \to c^2_{q_f}))$$

$$e' = c_5$$

Remark 3.10 *Terms* $b^1_i, b^2_i, b'^1_i, b'^2_i$ *correspond to the i-th rule in* δ*.*

Remark 3.11 *We shall apply the reasoning from Section 3.2 to the fourth equation of* $E_{\mathcal{A}}$*.*

In the remaining part of the paper, we fix an automaton $\mathcal{A}$. We need a few facts.

Fact 3.12 `(the first argument of` F`)`
If $E_{\mathcal{A}}$ *has a unifier* σ *then* $F\sigma = t_1 \to w_1$ *for some* $L^*_{\mathcal{A}}$*-term* t_1*.*

Proof:
Consider the first equation of $E_{\mathcal{A}}$. Unifiers of the equation have property that either $F\sigma = t_1 \to c_2$ or $F\sigma = t'_1 \to w_1$ where t_1 and t'_1 are some $L^*_{\mathcal{A}}$-terms. Similarly for the second equation of $E_{\mathcal{A}}$ — unifiers of the equation have property that either $F\sigma = t_2 \to c_3$ or $F\sigma = t'_2 \to w_1$ where t_2 and t'_2 are some $L^*_{\mathcal{A}}$-terms. As both equations must be unified, the only possibility is that $F\sigma$ has the form $t_1 \to w_1$. □

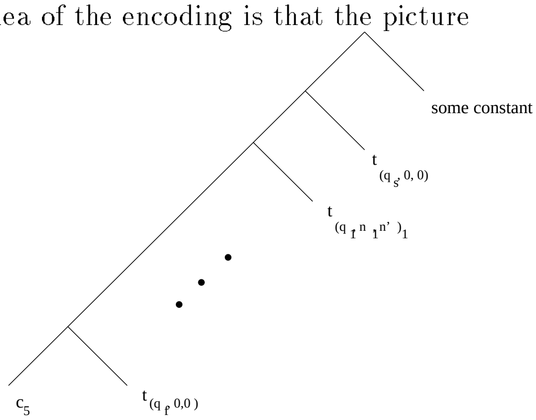
Fact 3.13 `(first two equations)`
If σ *is a substitution such that* $F\sigma = t_1 \to w_1$ *for some* $L^*_{\mathcal{A}}$*-term* t_1 *then there exists a substitution* σ' *such that* $x\sigma = x\sigma'$ *for* $x \neq x_1$ *and* σ' *unifies first two equations of* $E_{\mathcal{A}}$*.*

Proof:
It is easy to see that $\sigma' = \sigma[L((Fw_1c_0 \dots c_0)\sigma)/x_1]$ meets our requirements. $\square$

Lemma 3.14 `(stop gives a unifier)`
If the automaton $\mathcal{A}$ stops then there exists a unifier σ of the unification expression $E_{\mathcal{A}}$.

Proof:
Let $\langle q_0, 0, 0\rangle, \langle q_1, n_1^1, n_1^2\rangle, \dots, \langle q_n, 0, 0\rangle$, where $q_s = q_0$ and $q_f = q_n$, be a run of the automaton $\mathcal{A}$. We have a sequence of corresponding codes of ID's — $\{t_i\}_{0\leq i\leq n}$. We construct another sequence of terms — $\{s_i\}_{0\leq i<n}$. For $i < n$, if $\langle q_i, n_i^1, n_i^2\rangle \rightarrow_\delta \langle q_{i+1}, n_{i+1}^1, n_{i+1}^2\rangle$ according to the m-th rule of δ (see Remark 2.3) then $t_i = u_1 \rightarrow u_2$ where $u_l = c_0 \rightarrow \mathrm{precntr}(n_i^l, c_{q_i}^l)$. This implies that either $u_l = c_0 \rightarrow \mathrm{precntr}(n_i^l - 1, b_m^l)$ if $n_i^l \neq 0$ or $u_l = b_m^l$ if $n_i^l = 0$. We put $s_i = u_1' \rightarrow u_2'$ where $u_l' = c_0 \rightarrow \mathrm{precntr}(n_i^l - 1, w_{(l-1)k+m+1})$ if $n_i^l \neq 0$ and $u_l' = w_{(l-1)k+m+1}$ if $n_i^l = 0$ where $k = |\delta|$

The general idea of the encoding is that the picture



presents the term $Fc_7b_1^1 \dots b_k^1b_1^2 \dots b_k^2e\sigma$. In order to define a substitution, we construct t' as follows

- $Rt' = w_1$;
- $RL^{i+1}t' = s_i$ for $i < n$;
- $L^{n+1}t' = w_{2k+2}$.

We take σ such that $\sigma(F) = t'$ as a candidate for a unifier of the third equation in $E_{\mathcal{A}}$

This substitution unifies third equation of $E_{\mathcal{A}}$ since by induction on j, such that $0 < j \leq n$, we get that $RL^j((Fc_7b_1^1 \dots b_k^1b_1^2 \dots b_k^2e)\sigma) = t_j$ and, similarly for $0 < j \leq n+1$, the term $RL^{j-1}((Fab_1'^1 \dots b_k'^1b_1'^2 \dots b_k'^2e')\sigma) = t_j$. Moreover, $R((Fc_7b_1^1 \dots b_k^1b_1^2 \dots b_k^2e)\sigma) = c_7$ by Fact 3.12.

Since, we have $RL^{j-1}((Fab_1'^1 \dots b_k'^1b_1'^2 \dots b_k'^2e')\sigma) = t_j$ for $0 < j \leq n+1$, we can, using Fact 3.13 and Fact 3.9, get a substitution that unifies the rest of equations in $E_{\mathcal{A}}$ while retaining unifiability of the third equation. $\square$

In order to prove the other side lemma we have to introduce another reduction system and prove some properties of it.

Definition 18 (the reduction system $\to_{Sim}$)
The reduction system $\to_{Sim}$ is defined by the set of rewrite rules

$$\{b_i^j \to_{Sim} b_i'^j \mid i,j \in \{1,\ldots,k\}\}$$

where notation is taken from Definition 17.

Fact 3.15 (a property of $\to_{Sim}$)
Let σ be a unifier of $E_{\mathcal{A}}$. If $RL^i(u_3^{\mathcal{A}}\sigma)$ is a pre-code of ID and RL^{i+1} is applicable to $u_3^{\mathcal{A}}\sigma$ then

$$RL^i(u_3^{\mathcal{A}}\sigma) \to_{Sim}^{\#} RL^{i+1}(u_3^{\mathcal{A}}\sigma)$$

Proof:
Let $\phi : \{1,2\} \to \{L,R\}$ be defined so that $\phi(1) = L$, and $\phi(2) = R$. We prove that for $l = 1,2$

$$\phi(l)RL^i(u_3^{\mathcal{A}}\sigma) \to_{Sim}^{\#} \phi(l)RL^{i+1}(u_3^{\mathcal{A}}\sigma).$$

Since $RL^i(u_3^{\mathcal{A}}\sigma) = LRL^i(u_3^{\mathcal{A}}\sigma) \to RRL^i(u_3^{\mathcal{A}}\sigma)$, this implies that

$$RL^i(u_3^{\mathcal{A}}\sigma) \to_{Sim}^{\#} RL^{i+1}(u_3^{\mathcal{A}}\sigma).$$

We have that $\phi(l)RL^i(u_3^{\mathcal{A}}\sigma) = c_0 \to \mathrm{precntr}(j, c_q)$ with $q \in Q$. There are two cases

- $j > 0$. The form of arguments of F in $u_3^{\mathcal{A}}\sigma$ implies that $\phi(l)RL^i(F\sigma) = c_0 \to \mathrm{precntr}(j_l, s_l)$ where $j_l \in \mathbb{N}$ and $s_l \in \{w_{(i-1)k+2}, \ldots, w_{lk+2}\} \cup \{c_q^l \mid q \in Q\}$.
  - If $\phi(l)RL^i(F\sigma) = c_0 \to \mathrm{precntr}(j_l, w_{i_l})$, where the symbol $w_{i_l} \in \{w_{(i-1)k+2}, \ldots, w_{lk+2}\} \cup \{c_q^l \mid q \in Q\}$, then $\phi(l)RL^i(u_3^{\mathcal{A}}\sigma) = c_0 \to \mathrm{precntr}(j_l, b_{i_l}^l)$ and $\phi(l)RL^{i+1}(u_3^{\mathcal{A}}\sigma) = c_0 \to \mathrm{precntr}(j_l, b_{i_l}'^l)$, so we have $\phi(l)RL^i(t_3^{\mathcal{A}}\sigma) \to_{Sim} \phi(l)RL^{i+1}(u_3^{\mathcal{A}}\sigma)$ according to the reduction rule $b_{i_l}^l \to_{Sim} b_{i_l}'^l$.
  - If $\phi(l)RL^i(F\sigma) = c_0 \to \mathrm{precntr}(j_l, c_q^l)$, where $q \in Q$, then the term $\phi(l)RL^{i+1}(u_3^{\mathcal{A}}\sigma)$ is the same, thus the hypothesis holds trivially.
- $j = 0$. This implies that $\phi(l)RL^i(F\sigma) = w_{j_l}$, where $j_l \in \mathbb{N}$ and $w_{j_l} \in \{w_{(i-1)k+2}, \ldots, w_{lk+2}\}$, then $\phi(l)RL^i(t_3^{\mathcal{A}}\sigma) = b_{i_l}^l$ and $\phi(l)RL^{i+1}(u_3^{\mathcal{A}}\sigma) = b_{i_l}'^l$, so $\phi(l)RL^i(t_3^{\mathcal{A}}\sigma) \to_{Sim}^{\#} \phi(l)RL^{i+1}(u_3^{\mathcal{A}}\sigma)$ according to the rule $b_{i_l}^l \to_{Sim} b_{i_l}'^l$.

□

Fact 3.16 (a code of ID gives a step of $\mathcal{A}$)
Let $t_{\langle q_1,n_1,n_1'\rangle}, t_{\langle q_2,n_2,n_2'\rangle}$ be different codes of IDs $\langle q_1,n_1,n_1'\rangle, \langle q_2,n_2,n_2'\rangle$ such that

$$t_{\langle q_1,n_1,n_1'\rangle} \to_{Sim}^{\#} t_{\langle q_2,n_2,n_2'\rangle}.$$

We have

$$\langle q_1,n_1,n_1'\rangle \to_{\delta} \langle q_2,n_2,n_2'\rangle.$$

where δ is the next-step relation of $\mathcal{A}$.

Proof:
By restriction (5) in Remark 2.1, we have that $q_1 \neq q_2$. This ensures that $Lt_{\langle q_1,n_1,n'_1\rangle}$ and $Rt_{\langle q_1,n_1,n'_1\rangle}$ contain redexes of $\to_{Sim}$. Otherwise, either the term $t_{\langle q_2,n_2,n'_2\rangle}$ is not a code of ID or $q_1 = q_2$. Since the reduction $t_{\langle q_1,n_1,n'_1\rangle} \to^{\#}_{Sim} t_{\langle q_2,n_2,n'_2\rangle}$ holds, we have four cases

- $t_{\langle q_1,n_1,n'_1\rangle} = (c_0 \to \mathrm{precntr}(j_1, s_1)) \to (c_0 \to \mathrm{precntr}(j_2, s_2))$
- $t_{\langle q_1,n_1,n'_1\rangle} = s_1 \to (c_0 \to \mathrm{precntr}(j_2, s_2))$
- $t_{\langle q_1,n_1,n'_1\rangle} = (c_0 \to \mathrm{precntr}(j_1, s_1)) \to s_2$
- $t_{\langle q_1,n_1,n'_1\rangle} = s_1 \to s_2$

where $j_1, j_2 \in \mathbb{N}$ and $s_l \in \{b^l_1, \ldots, b^l_k\}$ for $k = |\delta|$ and $l \in \{1, 2\}$ (the condition for s_l holds, because the term $Lt_{\langle q_1,n_1,n'_1\rangle}$ does not contain c^2_q for $q \in Q$ and the term $Rt_{\langle q_1,n_1,n'_1\rangle}$ does not contain c^1_q). We shall present the proof for the first case. In other cases, the proof is similar.

It is enough to prove that $s_1 = b^1_j$ and $s_2 = b^2_j$ for some j. Indeed, as $b^1_j \to_{Sim} b'^1_j$ and $b^2_j \to_{Sim} b'^2_j$ and no other reduction is possible (see restriction (1) in Remark 2.1 and the way b^i_j is defined), we have

$$\langle q_1, n_1, n'_1\rangle \to_\delta \langle q_2, n_2, n'_2\rangle$$

by the j-th rule of δ.

We shall prove that $s_1 = b^1_j$ and $s_2 = b^2_j$ for some j. The restriction (4) in Remark 2.1 ensures that there is only one way to shift from q_1 to q_2, so there exists only one pair of rewrite rules of $\to_{Sim}$ that can be applied to s_1 and s_2. This implies that $s_1 = b^1_j$ and $s_2 = b^2_j$ for some j. □

Lemma 3.17 `(a unifier implies stop)`
If the unification expression $E_{\mathcal{A}}$ has a unifier then the automaton $\mathcal{A}$ stops.

Proof:
Let i_0 be the greatest number such that RLi_0 is applicable to $u^{\mathcal{A}}_3\sigma$. By Fact 3.13 and the way $u^{\mathcal{A}}_3\sigma$ is constructed, $RL(u^{\mathcal{A}}_3\sigma) = (c_0 \to c^1_{q_s}) \to (c_0 \to c^2_{q_s})$. This implies by Fact 3.15 that, for all i such that $1 \leq i \leq i_0$, we have the reduction

$$RL^i(u^{\mathcal{A}}_3\sigma) \to^{\#}_{Sim} RL^{i+1}(u^{\mathcal{A}}_3\sigma). \tag{1}$$

Since $\to_{Sim}$ applied to a pre-code of ID gives a pre-code of ID, we have the conclusion that $RL^i(u^{\mathcal{A}}_3\sigma)$ is a pre-code of ID for all i such that $1 \leq i \leq i_0$. Since the fourth equation of $E_{\mathcal{A}}$ is the co-ordination equation for the term $L(u^{\mathcal{A}}_3\sigma)$, we can conclude by Fact 3.1 and Lemma 3.8 that $RL^i(u^{\mathcal{A}}_3\sigma)$ is a code of ID. This implies that the sequence $\{RL^i(u^{\mathcal{A}}_3\sigma)\}_{1\leq i\leq i_0}$ corresponds to the sequence of ID's

$$\langle q_1, n_1, n'_1\rangle, \langle q_2, n_2, n'_2\rangle, \ldots, \langle q_{i_0}, n_{i_0}, n'_{i_0}\rangle.$$

Since RL^{i_0+1} is not applicable to $u^{\mathcal{A}}_3\sigma$, we have that $L^{i_0}(F\sigma) = w_{j_0}$ for some j_0. Since $L^{i_0}(u^{\mathcal{A}}_3\sigma)$ contains a code of ID as a proper subterm, we have $j_0 = 2k + 2$

where $k = |\delta|$. This means $\langle q_{i_0}, n_{i_0}, n'_{i_0}\rangle = \langle q_f, 0, 0\rangle$. Moreover, by Fact 3.13, $\langle q_1, n_1, n'_1\rangle = \langle q_s, 0, 0\rangle$.

We take into account the subsequence of $\{\langle q_i, n_i, n'_i\rangle\}_{1\le i\le i_0}$ indexed by l_j — $\{\langle q_{l_j}, n_{l_j}, n'_{l_j}\rangle\}_j$ such that $\langle q_{l_j}, n_{l_j}, n'_{l_j}\rangle \neq \langle q_{l_{j+1}}, n_{l_{j+1}}, n'_{l_{j+1}}\rangle$. By Fact 3.16 and (1), we get that $\{\langle q_{l_j}, n_{l_j}, n'_{l_j}\rangle\}_j$ is a sequence of subsequent steps of $\mathcal{A}$ beginning at $\langle q_s, 0, 0\rangle$ and ending at $\langle q_f, 0, 0\rangle$, so this sequence is the successful run of $\mathcal{A}$. This means that $\mathcal{A}$ halts. □

Now we can give the main result about the unification.

Theorem 3.18 `(the undecidability of the unification)`
The second-order unification problem for simple instances is undecidable.

Proof:
Given an automaton $\mathcal{A}$ we construct a unification problem instance $E_\mathcal{A}$ such that according to Lemma 3.14 and Lemma 3.17, $\mathcal{A}$ stops iff $E_\mathcal{A}$ has a unifier, so if the second-order unification problem was decidable then the halting problem for two-counter automata would be decidable, too. □

4 The system $\mathcal{F}$

We deal with the Church-style system $\mathcal{F}$ as defined in [Bar92] (called there $\lambda 2$). The problem in question is

Definition 19 `(the type inference problem)`
The type inference problem *is the problem to decide whether for a given term M of the system, there exists a base Γ and a type τ such that*

$$\Gamma \vdash M : \tau$$

is correct in the system $\mathcal{F}$. Such a term M is called an instance of the type inference problem.

Now we aim for the proof of undecidability of the type inference problem for the Church-style system $\mathcal{F}$. We shall present a coding of (simple) instances of the second-order unification problem into terms of the system $\mathcal{F}$ such that $E \in \text{u-expr}_\text{s}(L)$ for some L is unifiable iff for $\text{code}(E)$ there exists Γ and τ such that $\Gamma \vdash \text{code}(E) : \tau$ is correct in the system $\mathcal{F}$. Our coding applies to each language L containing at most one constant function symbol $\rightarrow$ (the arity of $\rightarrow$ is 2). Moreover, we assume we have at least one constant c_0.

By Remark 2.4, we can assume E has only one equation. From now on we fix one such an equation.

Definition 20 `(a nice subterm)`
A subterm s of a term t is nice *iff it is not a variable and has at least one occurrence in a position which is not contained in an argument of a functional variable occuring in t.*

Example 1 *The term $c_0 \to c_0$ is a nice subterm of the term $(c_0 \to c_0) \to F(c_0 \to c_0)$ and is not a nice subterm of the term $G(c_0 \to c_0) \to F(c_0 \to c_0)$.*

Definition 21 `(coding of unification expressions)`
Let L be a language and $E = \{\langle s,t\rangle\} \in u\text{-}expr_{\mathrm{s}}(L)$. We introduce two disjoint sets of type variables

$$G_L = \{\gamma_c \mid c \in \mathrm{IC}_L\}$$

and

$$A_L = \{\alpha_i \mid i \in \mathbb{N}\}$$

and define a coding of L^*-terms without variables to types

- $\mathrm{ctype}(t_1 \to t_2) = \mathrm{ctype}(t_1) \to \mathrm{ctype}(t_2)$*;*
- $\mathrm{ctype}(c) = \gamma_c$ *for* $c \in \mathrm{IC}_L$*.*
- $\mathrm{ctype}(w_i) = \alpha_i$ *for* $w_i \in \mathrm{PM}_L$*.*

We take two sets of distinct individual variables

$$V^E_{\mathrm{eq}} = \{v^u_{\mathrm{eq}} \mid u \textit{ is a nice subterm of } s \textit{ or } t \textit{ or a variable}\}$$

$$V^E_{\mathrm{check}} = \{v^u_{\mathrm{ch}} \mid u \textit{ is a nice subterm of } s \textit{ or } t\}$$

such that $V^E_{eq} \cap V^E_{check} = \emptyset$. We introduce three more individual variables z, v_{main}, and v_{pair}. We can define recursively a coding of nice subterms of terms in E now

- $\mathrm{coding}^E(Fu_1 \ldots u_n, v) =$

  $$v_{\mathrm{pair}}(v^{Fu_1\ldots u_n}_{\mathrm{eq}}(v^F_{\mathrm{eq}}\mathrm{ctype}(u_1)\ldots\mathrm{ctype}(u_n))z)(v^{Fu_1\ldots u_n}_{\mathrm{eq}}vz)$$

  where $F \in \mathrm{FV}_L \cup \mathrm{IV}_L$ and has arity n;

- $\mathrm{coding}^E(c, v) = v_{\mathrm{pair}}(v^c_{\mathrm{eq}}v^c_{\mathrm{ch}}z)(v^c_{\mathrm{eq}}vz)$ *where* $c \in \mathrm{IC}_L$*;*
- $\mathrm{coding}^E(u_1 \to u_2, v) =$

  $$v_{\mathrm{pair}}(v^{u_1\to u_2}_{\mathrm{eq}}(vv^{u_1}_{\mathrm{ch}})\mathrm{coding}^E(u_1, v^{u_1}_{\mathrm{ch}}))(v^{u_1\to u_2}_{\mathrm{eq}}v^{u_2}_{\mathrm{ch}}\mathrm{coding}^E(u_2, v^{u_2}_{\mathrm{ch}})).$$

Finally, we define the code for E

$$\begin{aligned}\mathrm{code}(E) = \Lambda\alpha\gamma_{c_1}\ldots\gamma_{c_m}.\lambda v_{\mathrm{pair}} : \alpha \to \alpha \to \alpha.\lambda z : \alpha.\lambda v^{c_1}_{\mathrm{ch}} : \gamma_{c_1}\ldots v^{c_m}_{\mathrm{ch}} : \gamma_{c_m}.\\ v_{\mathrm{pair}}(v_{\mathrm{main}}v^s_{\mathrm{ch}}\mathrm{coding}^E(s, v^s_{\mathrm{ch}}))(v_{\mathrm{main}}v^t_{\mathrm{ch}}\mathrm{coding}^E(t, v^t_{\mathrm{ch}}))\end{aligned}$$

where $\{c_1, \ldots, c_m\} = \mathrm{IC}_L$.

Generally, our intention is that if $\mathrm{code}(E)$ is typable then the type of the term is

$$\forall\alpha\gamma_{c_1}\ldots\gamma_{c_m}.(\alpha \to \alpha \to \alpha) \to \alpha \to \gamma_{c_1} \to \ldots \to \gamma_{c_m} \to \alpha$$

We shall call the type τ_L.

Definition 22 `(coding of a substitution)`
For a given substitution σ such that $\mathrm{var}(E) \subseteq \mathrm{dom}(\sigma)$*, we define*

$$\begin{array}{rcl}\Gamma_\sigma & = & \{v_{\mathrm{eq}}^{u_1 \to u_2} : \mathrm{ctype}(u_2\sigma) \to \alpha \to \alpha \mid u_1 \to u_2 \textit{ is a nice subterm} \\ & & \textit{of } s \textit{ or } t\} \cup \\ & \cup & \{v_{\mathrm{eq}}^{F u_1 \ldots u_n} : \mathrm{ctype}(F u_1 \ldots u_n \sigma) \to \alpha \to \alpha \mid F \in \mathrm{FV}_L \cup \mathrm{IV}_L \textit{ and} \\ & & \textit{has arity } n \textit{ and } u_1, \ldots, u_n \textit{ are terms without variables}\} \cup \\ & \cup & \{v_{\mathrm{eq}}^{F} : \forall \alpha_1 \ldots \alpha_n . \mathrm{ctype}(F\sigma) \mid F \in \mathrm{FV}_L \cup \mathrm{IV}_L\} \cup \\ & \cup & \{v_{\mathrm{eq}}^{c} : \mathrm{ctype}(c) \to \alpha \to \alpha \mid c \in \mathrm{IC}_L\} \cup \\ & \cup & \{v_{\mathrm{ch}}^{u} : \mathrm{ctype}(u\sigma) \mid u \textit{ is a nice subterm of } s \textit{ or } t\} \cup \\ & \cup & \{v_{\mathrm{pair}} : \alpha \to \alpha \to \alpha, z : \alpha\} \cup \\ & \cup & \{v_{\mathrm{main}} : \mathrm{ctype}(s\sigma) \to \alpha \to \alpha\}.\end{array}$$

The coding has the following property

Fact 4.1 `(property of the coding)`
For each substitution σ such that $\mathrm{var}(E) \subseteq \mathrm{dom}(\sigma)$*, we have that for each nice subterm u of terms in E*

$$\Gamma_\sigma \vdash \mathrm{coding}^E(u, v_{\mathrm{ch}}^u) : \alpha$$

is correct in the system $\mathcal{F}$.

Proof:
Induction on the structure of the term u.

- $u = c$ for some $c \in \mathrm{IC}_L$. In this case, we have that $\mathrm{coding}^E(c, v_{\mathrm{ch}}^c) = v_{\mathrm{pair}}(v_{\mathrm{eq}}^c v_{\mathrm{ch}}^c z)(v_{\mathrm{eq}}^c v_{\mathrm{ch}}^c z)$. We have $\Gamma_\sigma(v_{\mathrm{ch}}^c) = \gamma_c$ and $\Gamma_\sigma(z) = \alpha$. Since $\Gamma_\sigma(v_{\mathrm{eq}}^c) = \gamma_c \to \alpha \to \alpha$, we get

  $$\Gamma_\sigma \vdash v_{\mathrm{eq}}^c v_{\mathrm{ch}}^c z : \alpha$$

  This implies

  $$\Gamma_\sigma \vdash v_{\mathrm{pair}}(v_{\mathrm{eq}}^c v_{\mathrm{ch}}^c z)(v_{\mathrm{eq}}^c v_{\mathrm{ch}}^c z) : \alpha$$

- $u = F u_1 \ldots u_n$ for some $F \in \mathrm{FV}_L \cup \mathrm{IV}_L$ of arity n and $u_1, \ldots, u_n$ terms without variables. In this case,

  $$\mathrm{coding}^E(F u_1 \ldots u_n, v_{\mathrm{ch}}^{F u_1 \ldots u_n}) = v_{\mathrm{pair}}(v_{\mathrm{eq}}^{F u_1 \ldots u_n}(v_{\mathrm{eq}}^F \mathrm{ctype}(u_1) \ldots \mathrm{ctype}(u_n))z)(v_{\mathrm{eq}}^{F u_1 \ldots u_n} v_{\mathrm{ch}}^{F u_1 \ldots u_n} z))$$

  We have $\Gamma_\sigma(v_{\mathrm{eq}}^F) = \forall \alpha_1 \ldots \alpha_n . \mathrm{ctype}(F\sigma)$. This means

  $$\Gamma_\sigma \vdash v_{\mathrm{eq}}^F \mathrm{ctype}(u_1) \ldots \mathrm{ctype}(u_n) : \mathrm{ctype}(F)[\vec{u_i}/\vec{\alpha_i}]$$

  Since arguments of $v_{\mathrm{eq}}^{F u_1 \ldots u_n}$ are $\mathrm{ctype}(F)[\vec{u_i}/\vec{\alpha_i}]$ and z has the type α we get

  $$\Gamma_\sigma \vdash v_{\mathrm{eq}}^{F u_1 \ldots u_n}(v_{\mathrm{eq}}^F \mathrm{ctype}(u_1) \ldots \mathrm{ctype}(u_n))z : \alpha$$

Since $\Gamma_\sigma(v_{\text{ch}}^{Fu_1\ldots u_n}) = \text{ctype}(F)[\vec{u_i}/\vec{\alpha_i}]$, we get

$$\Gamma_\sigma \vdash v_{\text{eq}}^{Fu_1\ldots u_n} v_{\text{ch}}^{Fu_1\ldots u_n} z : \alpha,$$

too. This implies

$$\Gamma_\sigma \vdash v_{\text{pair}}(v_{\text{eq}}^{Fu_1\ldots u_n}(v_{\text{eq}}^{F}\text{ctype}(u_1)\ldots\text{ctype}(u_n))z)(v_{\text{eq}}^{Fu_1\ldots u_n} v_{\text{ch}}^{Fu_1\ldots u_n} z)) : \alpha$$

- $u = u_1 \to u_2$ for some terms u_1, u_2. In this case,

$$\begin{aligned}\text{coding}^E(u_1 \to u_2, v_{\text{ch}}^{u_1\to u_2}) = v_{\text{pair}}&(v_{\text{eq}}^{u_1\to u_2}(v_{\text{ch}}^{u_1\to u_2} v_{\text{ch}}^{u_1})\text{coding}^E(u_1, v_{\text{ch}}^{u_1}))\\ &(v_{\text{eq}}^{u_1\to u_2} v_{\text{ch}}^{u_2}\text{coding}^E(u_2, v_{\text{ch}}^{u_2}))\end{aligned}$$

By induction hypothesis

$$\Gamma_\sigma \vdash \text{coding}^E(u_i, v_{\text{ch}}^{u_i}) : \alpha$$

for $i \in \{1, 2\}$. By the definition of Γ_σ, we get that $\Gamma_\sigma \vdash v_{\text{ch}}^{u_1\to u_2} v_{\text{ch}}^{u_1}$: $\text{ctype}(u_2)$ and $\Gamma_\sigma \vdash v_{\text{ch}}^{u_2} : \text{ctype}(u_2)$. Since the type of $v_{\text{eq}}^{u_1\to u_2}$ in Γ_σ is $\text{ctype}(u_2) \to \alpha \to \alpha$, we get that

$$\begin{aligned}\Gamma_\sigma \vdash v_{\text{pair}}&(v_{\text{eq}}^{u_1\to u_2}(v_{\text{ch}}^{u_1\to u_2} v_{\text{ch}}^{u_1})\text{coding}^E(u_1, v_{\text{ch}}^{u_1}))\\ &(v_{\text{eq}}^{u_1\to u_2} v_{\text{ch}}^{u_2}\text{coding}^E(u_2, v_{\text{ch}}^{u_2}))\end{aligned}$$

□

As a corollary, we get

Lemma 4.2 `(a unifier implies typability)`
If σ is a unifier of E then

$$\Gamma_\sigma \vdash \text{code}(E) : \tau_L$$

is correct in the system $\mathcal{F}$.

Proof:
By Fact 4.1 we get that

$$\Gamma_\sigma \vdash \text{coding}^E(s, v_{\text{ch}}^s) : \alpha$$

and

$$\Gamma_\sigma \vdash \text{coding}^E(t, v_{\text{ch}}^t) : \alpha$$

Then by the definition of Γ_σ, we get

$$\Gamma_\sigma \vdash v_{\text{main}} v_{\text{ch}}^s \text{coding}^E(s, v_{\text{ch}}^s) : \alpha. \tag{2}$$

and

$$\Gamma_\sigma \vdash v_{\text{main}} v_{\text{ch}}^t \text{coding}^E(t, v_{\text{ch}}^t) : \alpha$$

This implies that

$$\Gamma_\sigma \vdash \text{code}(E) : \tau_L$$

□

Now we shall make some preparation for the other side lemma.

Definition 23 (from a type to an L-term)
First, we define recursively two codings of types

$$\begin{array}{rcl}
\mathrm{tcode}'(\langle\alpha_1\ldots\alpha_n\rangle,\gamma_c) & = & c \textit{ where } c\in \mathrm{IC}_L\\
\mathrm{tcode}'(\langle\alpha_1\ldots\alpha_n\rangle,\alpha_i) & = & w_i \textit{ for } i\in\{1,\ldots,n\}\\
\mathrm{tcode}'(\langle\alpha_1\ldots\alpha_n\rangle,\forall\alpha.\tau) & = & c_0 \textit{ where } c_0 \textit{ is a distinguished constant}\\
\mathrm{tcode}'(\langle\alpha_1\ldots\alpha_n\rangle,u_1\to u_2) & = & \mathrm{tcode}'(\langle\alpha_1\ldots\alpha_n\rangle,u_1)\to\mathrm{tcode}'(\langle\alpha_1\ldots\alpha_n\rangle,u_2)
\end{array}$$

and

$$\begin{array}{rcl}
\mathrm{tcode}(\forall\alpha_1\ldots\alpha_n.\tau) & = & \mathrm{tcode}'(\langle\alpha_1\ldots\alpha_n\rangle,\tau) \textit{ where } \tau \textit{ does not begin with}\\
 & & \quad \textit{a quantifier;}\\
\mathrm{tcode}(\tau) & = & \mathrm{tcode}'(\langle\rangle,\tau) \textit{ where } \tau \textit{ does not begin with a quantifier.}
\end{array}$$

Definition 24 (from a base to a substitution)
For a given base Γ, we define a substitution σ_Γ as

$$\sigma_\Gamma(F)=\mathrm{tcode}(\Gamma(v^F_{\mathrm{eq}}))$$

For notational convenience we put

$$\Delta=\{v_{\mathrm{pair}}:\alpha\to\alpha\to\alpha, z:\alpha\}\cup\{v^c_{\mathrm{eq}}:\gamma_c\to\alpha\to\alpha \mid c\in\mathrm{IC}_L\}$$

Fact 4.3 (a property of the other side coding)
Let u be a nice subterm of t or s. For each base Γ and a type α such that

$$\Gamma,\Delta\vdash \mathrm{coding}^E(u,v^u_{\mathrm{ch}}):\alpha$$

is correct in the system $\mathcal{F}$, we have $\mathrm{tcode}(\Gamma(v^u_{\mathrm{ch}}))=u\sigma_\Gamma$.

Proof:
Induction on the structure of u.

- $u=c$ for some $c\in\mathrm{IC}_L$. Since v^c_{eq} has a type $\gamma c\to\alpha\to\alpha$ in Δ, we get by the shape of $\mathrm{coding}^E(c,v^c_{\mathrm{ch}})$ that v^c_{eq} has the type γ_c in Γ. This implies $\mathrm{tcode}(\Gamma(v^c_{\mathrm{ch}}))=c=c\sigma_\Gamma$.

- $u=u_1\to u_1$ for some terms u_1 and u_2. Since $\mathrm{coding}^E(u_i,v^{u_i}_{\mathrm{ch}})$ for $i\in\{1,2\}$ are subterms of $\mathrm{coding}^E(u,v^u_{\mathrm{ch}})$, we have that there exist τ_i such that
$$\Gamma,\Delta\vdash\mathrm{coding}^E(u_i,v^{u_i}_{\mathrm{ch}}):\tau_i.$$
Since $\mathrm{coding}^E(t,v)=v_{\mathrm{pair}}t_1t_2$ for any t and v and for some t_1,t_2, and since $\Delta(v_{\mathrm{pair}})=\alpha\to\alpha\to\alpha$, we have $\tau_i=\alpha$ for $i\in\{1,2\}$. By the induction hypothesis, we get that $\mathrm{tcode}(\Gamma(v^{u_i}_{\mathrm{ch}}))=u_i\sigma_\Gamma$ for $i\in\{1,2\}$, so indeed, $\mathrm{tcode}(\Gamma(v^u_{\mathrm{ch}}))=u\sigma_\Gamma$ in this case.

- $u = Fu_1 \dots u_n$ for some n-ary symbol $F \in \mathrm{FV}_L \cup \mathrm{IV}_L$. Since v^F_{eq} is applied to the sequence $\mathrm{ctype}(u_1) \dots \mathrm{ctype}(u_n)$ in $\mathrm{coding}^E(u, v^u_{\mathrm{ch}})$, we get that the type of v^F_{eq} has the form $\forall\alpha_1 \dots \alpha_n.\tau$ for some type τ. Since v^u_{ch} is the same argument of v^u_{eq} in $\mathrm{coding}^E(u, v^u_{\mathrm{ch}})$ and in $v^F_{\mathrm{eq}}\mathrm{ctype}(u_1) \dots \mathrm{ctype}(u_n)$, we get that type of v^u_{ch} is equal to $\tau[\mathrm{ctype}(u_i)/\alpha i]$. It is easy to see that $\mathrm{tcode}(\tau[\overrightarrow{\mathrm{ctype}}(u_i)/\vec{\alpha_i}])$ is equal to $F\sigma_\Gamma[\vec{u_i}/w_i]$ which gives our result, $\mathrm{tcode}(\Gamma(v^u_{\mathrm{ch}})) = u\sigma_\Gamma$.

□

Again, we get as a corollary

Lemma 4.4 `(typability gives a unifier)`
If there exists a base Γ and a type τ such that

$$\Gamma \vdash \mathrm{code}(E) : \tau$$

is correct in the system $\mathcal{F}$ then σ_Γ is a unifier of E.

Proof:
By the form of $\mathrm{code}(E)$ we get that types of v^s_{ch} and v^t_{ch} are equal. Since by Fact 4.3, we have that $\mathrm{tcode}(\Gamma(v^s_{\mathrm{ch}})) = s\sigma_\Gamma$ and $\mathrm{tcode}(\Gamma(v^t_{\mathrm{ch}})) = t\sigma_\Gamma$, we get that σ_Γ unifies E. □

Theorem 4.5 `(the undecidability of the type inference)`
The problem of the type inference is undecidable for the Church-style system $\mathcal{F}$.

Proof:
By Lemma 4.2 and Lemma 4.4, we have that for a given instance E of the unification problem there exists a term $\mathrm{code}(E)$ of the system $\mathcal{F}$ such that E has a unifier iff there exists a base Γ and a term τ such that

$$\Gamma \vdash \mathrm{code}(E) : \tau$$

is correct in the system $\mathcal{F}$. This and Theorem 3.18 imply that the problem of type inference for the Church-style system $\mathcal{F}$ is undecidable. □

Remark 4.6 *It is worth mentioning that the same proof works for the Leivant's [Lei91] predicative version of the system $\mathcal{F}$. The types substituted for variables are all of the zero generation.*

5 Conclusions

We have presented an alternative proof for the classical result by Goldfarb — the undecidability of the second-order unification problem. The Goldfarb's proof reduces the tenth Hilbert's problem to the second-order unification problem. The undecidability of the tenth Hilbert's problem requires non-trivial number theoretic properties, though. In our approach, we use a reduction to elementary undecidable problem — the halting problem for two-counter automata. This is a methodological simplification of the undecidability proof for the second-order unification problem.

Another interesting conclusion of the paper is that the type inference problem is undecidable for the Church-style system $\mathcal{F}$. This is surprising since this problem is usually considered easy in the context of Church-style λ-calculi with types.

Moreover, our proof works for the predicative restriction of the system $\mathcal{F}$ introduced by Leivant [Lei91].

References

[Bar92] H. P. Barendregt, *Lambda calculi with types*, Handbook of Logic in Computer Science (S. Abramsky, D. M. Gabbay, and T. S. E. Mainbaum, eds.), vol. 2, Oxford University Press, 1992, pp. 117–309.

[Boe85] H.-J. Boehm, *Partial polymorphic type inference is undecidable*, 26th Annual Symposium on Foundations of Computer Science, IEEE, October 1985, pp. 339–345.

[Cur69] H. B. Curry, *Modified basic functionality in combinatory logic*, Dialectica (1969).

[Far91] William M. Farmer, *Simple second-order languages for which unification is undecidable*, TCS (1991), no. 87, 25–41.

[Gir72] J.-Y. Girard, *Interprétation fonctionelle et élimination des coupures dans l'arithmetique d'ordre supérieur*, Ph.D. thesis, Université Paris VII, 1972.

[Gol81] W. D. Goldfarb, *The undecidability of the second-order unification problem*, TCS (1981), no. 13, 225–230.

[GR91] P. Gianini and S. Ronchi Della Rocca, *Type inference in polymorphic type discipline*, Theoretical Aspects of Comput. Soft.: Int'l Conf., LNCS, vol. 526, Springer-Verlag, September 1991, pp. 18–37.

[HU79] J.E. Hopcroft and J.D. Ullman, *Introduction to automata theory, language, and computation*, Addison-Wesley Publishing Company, 1979.

[KT92] A. J. Kfoury and J. Tiuryn, *Type reconstruction in finite-rank fragments of the second-order λ-calculus*, Information and Computation **2** (1992), no. 98, 228–257.

[KW94] A. J. Kfoury and J. B. Wells, *A direct algorithm for type inference in the rank-2 fragment of the second-order λ-calculus*, Proc. 1994 ACM Conf. LISP Funct. Program., 1994.

[Lei83] D. Leivant, *Polymorphic type inference*, Conf. Rec. 10th Ann. ACM Symp. Princ. Program. Lang., January 1983, pp. 88–98.

[Lei91] D. Leivant, *Finitely stratified polymorphism*, Information and Computation **1** (1991), no. 93, 93–113.

[Pfe88] F. Pfenning, *Partial polymorphic type inference and higher-order unification*, Proceedings of the 1988 ACM Conference on Lisp and Functional Programming, ACM Press, July 1988, pp. 153–163.

[Pfe93] F. Pfenning, *On the undecidability of partial polymorphic type reconstruction*, Fundamenta Informaticae **19** (1993), no. 1,2, 185–199.

[Rey74] J. C. Reynolds, *Mathematical foundations of software development*, (Ehring et al., ed.), Lecture Notes in Computer Science, vol. 19, Springer, 1974, pp. 408–425.

[Wel94] J. B. Wells, *Typability and type checking in the second-order λ-calculus are equivalent and undecidable*, Proceedings of Ninth Annual IEEE Symposium on Logic in Computer Science, July 1994, pp. 176–185.