Undecidability of Semi-Unification on a Napkin

Andrej Dudenhefner
Saarland University, Saarbrücken, Germany
dudenhefner@ps.uni-saarland.de

Abstract

Semi-unification (unification combined with matching) has been proven undecidable by Kfoury, Tiuryn, and Urzyczyn in the 1990s. The original argument reduces Turing machine immortality via Turing machine boundedness to semi-unification. The latter part is technically most challenging, involving several intermediate models of computation.

This work presents a novel, simpler reduction from Turing machine boundedness to semi-unification. In contrast to the original argument, we directly translate boundedness to solutions of semi-unification and vice versa. In addition, the reduction is mechanized in the Coq proof assistant, relying on a mechanization-friendly stack machine model that corresponds to space-bounded Turing machines. Taking advantage of the simpler proof, the mechanization is comparatively short and fully constructive.

2012 ACM Subject Classification Theory of computation → Computability

Keywords and phrases undecidability, semi-unification, mechanization

Digital Object Identifier 10.4230/LIPIcs.FSCD.2020.9

Supplementary Material `https://github.com/uds-psl/2020-fscd-semi-unification`

1 Introduction

In the 1980s it was an actively studied, long-standing open problem whether the combination of first-order unification and matching, both of which are decidable problems, is decidable. This problem, called *semi-unification*, is: given a finite set of pairs (σ, τ) of first-order terms, is there a valuation φ of term variables such that for each pair (σ, τ) we have $\psi(\varphi(\sigma)) = \varphi(\tau)$ for some valuation ψ of term variables?

Semi-unification is directly related [10, 16] to type inference in an extension of the Hindley–Milner type system [11, 19] (cf. the standard **ML** [20] programming language), which allows for polymorphic recursion [21]. Therefore, computational properties of semi-unification translate to type inference capabilities for polymorphic functional programming languages, affecting programming language design. For a broad overview over properties of semi-unification the reader is referred to [17, 13].

In the 1990s Kfoury, Tiuryn, and Urzyczyn have shown that semi-unification is undecidable [15, 17]. This negative result motivated exploration of decidable fragments of semi-unification (for an overview see [18]). The original undecidability proof is quite sophisticated, reflecting the inherent intricacy of the semi-unification problem. It involves Turing machine immortality, symmetric intercell Turing machine boundedness, path equation derivability, and termination of a redex contraction procedure for semi-unification. Therefore, it is challenging to verify the original proof down to the last detail, let alone mechanize it in a proof assistant. Additionally, the original argument uses König's lemma and it is not obvious whether it can be presented constructively.

This work contributes to a better understanding of semi-unification in three aspects. First, we present a simpler proof for the undecidability of semi-unification. The presented technical argument connects an undecidable machine property (in immediate correspondence with Turing machine boundedness) to solutions of semi-unification in a direct way. The key contribution regarding this aspect is the function ζ (Definition 41) that constructs solutions

5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020).
Editor: Zena M. Ariola; Article No. 9; pp. 9:1–9:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

for semi-unification instances. Second, we mechanize [2] (ca. 1500 lines of code in the Coq [5] proof assistant) the presented argument, leaving little room for doubt regarding its correctness. Third, König's lemma in the original argument is replaced by the fan theorem. The provided mechanization reveals full constructivity of the remaining reasoning.

Proof Synopsis

First, we reduce Turing machine immortality [12] (is there a diverging configuration?) to a uniform boundedness problem for stack machines (is there a uniform bound on the number of reachable configurations?). The considered, restricted class of stack machines, which we call *simple*, is a mechanization-friendly presentation of space-bounded Turing machines.

Second, given a simple stack machine $\mathcal{M}$, we encode each instruction of $\mathcal{M}$ as a semi-unification constraint, thereby constructing a finite set of constraints $\mathcal{C}$. Each state of $\mathcal{M}$ is a variable in $\mathcal{C}$. The resulting constraints are of restricted shape, which we also call *simple*.

Third, if $\mathcal{M}$ is uniformly bounded, then we interpret configurations of $\mathcal{M}$ as first-order terms using an uncomplicated, computable function ζ. Most importantly, the interpretation of an empty stack configuration in each state of $\mathcal{M}$ is a solution for $\mathcal{C}$.

Fourth, if $\mathcal{C}$ has a solution φ, then we construct a uniform bound for $\mathcal{M}$ from the maximal depth of the syntax trees in the range of φ.

Fifth, the above constitutes an undecidability proof of semi-unification for simple constraints and immediately implies undecidability of semi-unification.

Key aspects of all of the above points, except the third, also appear in [17]. However, the technically most challenging aspect of [17], which we are able to simplify, is to show that a solution for a constructed semi-unification instance exists. Specifically, the function ζ is the main contribution of this work towards a better understanding of semi-unification.

Organization of the Paper

Section 2 contains preliminary properties of simple semi-unification (Problem 15), which is a restriction of semi-unification that transports undecidability (Theorem 1).

Section 3 contains preliminary properties of simple stack machines (Definition 16), which are equivalent to space-bounded Turing machines. Additionally, uniform boundedness of deterministic simple stack machines (Problem 26) is shown undecidable (Theorem 2).

Section 4 contains a reduction from uniform boundedness of deterministic simple stack machines to simple semi-unification. Correctness of the reduction (Lemma 48 and Lemma 45) results in undecidability of semi-unification (Theorem 4).

Section 5 provides an overview over the mechanization [2] of the presented reduction.

Section 6 concludes and lists potential future work.

2 Semi-unification Preliminaries

This section, following [17], recollects the basic definition and properties of semi-unification (Problem 3).

▶ **Definition 1** (Terms ($\mathbb{T}$)). Let $\mathbb{V}$ be a countably infinite set of *variables* ranged over by α, β, γ. The set of *terms* $\mathbb{T}$, ranged over by σ, τ, is given by the grammar

$$\sigma, \tau \in \mathbb{T} ::= \alpha \mid \sigma \to \tau$$

▶ **Definition 2** (Valuation $(\varphi), (\psi)$). A *valuation* $\varphi : \mathbb{V} \to \mathbb{T}$ assigns terms to variables, and is tacitly lifted to terms.

▶ **Problem 3** (Semi-unification (SU))**.** Given a finite set $\{s_1 \leq_1 t_1, \ldots, s_n \leq_n t_n\}$ of *indexed inequalities*, do there exist valuations $\varphi, \psi_1, \ldots, \psi_n : \mathbb{V} \to \mathbb{T}$ such that $\psi_i(\varphi(s_i)) = \varphi(t_i)$ holds for $i = 1 \ldots n$?

Compared to first-order unification, semi-unification is non-structural. In a solvable instance, the left-hand side of an indexed inequality may even appear as subterm of the right-hand side (Example 4).

▶ **Example 4.** The indexed inequalities $\{\alpha \leq_1 \alpha \to \beta, \alpha \to \alpha \leq_2 \beta\}$ are solved by the valuations $\varphi = \{\alpha \mapsto \alpha, \beta \mapsto \alpha \to \alpha\}$, $\psi_1 = \{\alpha \mapsto \alpha \to (\alpha \to \alpha)\}$, and $\psi_2 = \{\alpha \mapsto \alpha\}$ because

$$\begin{aligned} \psi_1(\varphi(\alpha)) &= \alpha \to (\alpha \to \alpha) = \varphi(\alpha \to \beta) \\ \psi_2(\varphi(\alpha \to \alpha)) &= \alpha \to \alpha = \varphi(\beta) \end{aligned}$$

◀

Next, we introduce the notion of constraints (Definition 6) (called *path equations* in [17]). Constraints play a key role connecting (constraint-based) semi-unification to the execution of a stack machine. Intuitively, a constraint $X \doteq Y$ reflects joinability of configurations X and Y in a stack machine (cf. Section 4).

▶ **Definition 5** (Binary Words ($\mathbb{B}^*$))**.** Let $\mathbb{B} = \{0, 1\}$ be ranged over by a, b. The set $\mathbb{B}^*$ of *words* is ranged over by s, t, v, w.

▶ **Definition 6** (Constraint ($s|\alpha|t \doteq v|\beta|w$))**.** A *constraint* has the shape $s|\alpha|t \doteq v|\beta|w$, where $\alpha, \beta \in \mathbb{V}$ and $s, t, v, w \in \mathbb{B}^*$.

A constraint is *simple* if it has the shape $a|\alpha|\epsilon \doteq \epsilon|\beta|b$, where $\alpha, \beta \in \mathbb{V}$, $a, b \in \mathbb{B}$, and ϵ is the empty word.

In order to connect words with valuations, we define valuation compositions (Definition 7) and path functions on terms (Definition 8).

▶ **Definition 7** (Valuation Composition (ψ_v))**.** Let $\psi_0, \psi_1 : \mathbb{V} \to \mathbb{T}$ be valuations. For a word $v \in \mathbb{B}^*$, the *composed valuation* $\psi_v : \mathbb{T} \to \mathbb{T}$ is such that

$$\psi_\epsilon(\sigma) = \sigma \qquad \psi_{wa}(\sigma) = \psi_w(\psi_a(\sigma))$$

▶ **Definition 8** (Path Function (π_v))**.** For a word $v \in \mathbb{B}^*$, the partial *path function* $\pi_v : \mathbb{T} \rightharpoonup \mathbb{T}$ is such that

$$\pi_\epsilon(\sigma) = \sigma \quad \pi_{0w}(\sigma \to \tau) = \pi_w(\sigma) \quad \pi_{1w}(\sigma \to \tau) = \pi_w(\tau) \quad (\text{otherwise } \pi_v(\sigma) \text{ is undefined})$$

Intuitively, a simple constraint $a|\alpha|\epsilon \doteq \epsilon|\beta|b$ is satisfied by a valuation triple $(\varphi, \psi_0, \psi_1)$, if $\psi_a(\varphi(\alpha)) = \pi_b(\varphi(\beta))$. The absence of ψ_0 and ψ_1 on the right-hand side captures matching as part of semi-unification. Similarly to [17], the respective side $s|\alpha|t$ of a constraint is interpreted wrt. a valuation triple $(\varphi, \psi_0, \psi_1)$ by the term which arises when we apply ψ_s to $\varphi(\alpha)$ and then select a subterm via π_t. This interpretation is captured by the following model relation ($\models$).

▶ **Definition 9** (Model Relation ($\models$))**.** A valuation triple $(\varphi, \psi_0, \psi_1)$ *models* a constraint $s|\alpha|t \doteq v|\beta|w$, written $(\varphi, \psi_0, \psi_1) \models s|\alpha|t \doteq v|\beta|w$, if $\pi_t(\psi_s(\varphi(\alpha))) = \pi_w(\psi_v(\varphi(\beta)))$.

For a set $\mathcal{C}$ of constraints, we write $(\varphi, \psi_0, \psi_1) \models \mathcal{C}$ if $(\varphi, \psi_0, \psi_1) \models C$ for all $C \in \mathcal{C}$.

For a set $\mathcal{C}$ of constraints and a constraint C, we write $\mathcal{C} \models C$ if for all valuation triples $(\varphi, \psi_0, \psi_1)$ such that $(\varphi, \psi_0, \psi_1) \models \mathcal{C}$ we have $(\varphi, \psi_0, \psi_1) \models C$.

As a side note, path equation derivability of [17] is sound for ($\models$). The following Example 10, Example 11, and Example 13 illustrate positive and negative cases for models.

▶ **Example 10.** Let $\mathcal{C} = \{0 \shortmid \alpha \shortmid \epsilon \doteq \epsilon \shortmid \beta \shortmid 1, 1 \shortmid \gamma \shortmid \epsilon \doteq \epsilon \shortmid \beta \shortmid 1, 1 \shortmid \alpha \shortmid \epsilon \doteq \epsilon \shortmid \gamma \shortmid 0\}$ be a set of simple constraints. We have $(\varphi, \psi_0, \psi_1) \models \mathcal{C}$, where

$$\begin{aligned}
\varphi &= \{\alpha \mapsto \alpha, \beta \mapsto \beta_0 \to (\beta_{10} \to \beta_{11}), \gamma \mapsto \gamma_0 \to \gamma_1\} \\
\psi_0 &= \{\alpha \mapsto \beta_{10} \to \beta_{11}\} \\
\psi_1 &= \{\alpha \mapsto \gamma_0, \gamma_0 \mapsto \beta_{10}, \gamma_1 \mapsto \beta_{11}\}
\end{aligned}$$

▶ **Example 11.** Let $\mathcal{C} = \{0 \shortmid \alpha \shortmid \epsilon \doteq \epsilon \shortmid \beta \shortmid 1, 1 \shortmid \gamma \shortmid \epsilon \doteq \epsilon \shortmid \beta \shortmid 1, 1 \shortmid \alpha \shortmid \epsilon \doteq \epsilon \shortmid \gamma \shortmid 0\}$ be a set of simple constraints. We have $\mathcal{C} \models 0 \shortmid \alpha \shortmid 0 \doteq 11 \shortmid \alpha \shortmid \epsilon$, because for any valuations φ, ψ_0, ψ_1 such that $(\varphi, \psi_0, \psi_1) \models \mathcal{C}$ we have

$$\pi_0(\psi_0(\varphi(\alpha))) = \pi_0(\pi_1(\varphi(\beta))) = \pi_0(\psi_1(\varphi(\gamma))) = \psi_1(\pi_0(\varphi(\gamma))) = \psi_1(\psi_1(\varphi(\alpha)))$$

The *depth* of a term is the maximal depth of its syntax tree, and is non-decreasing under substitution.

▶ **Definition 12** (Term Depth (depth))**.** The function depth : $\mathbb{T} \to \mathbb{N}$ is such that

$$\mathrm{depth}(\alpha) = 0 \qquad \mathrm{depth}(\sigma \to \tau) = 1 + \max\{\mathrm{depth}(\sigma), \mathrm{depth}(\tau)\}$$

▶ **Example 13.** There is no valuation triple $(\varphi, \psi_0, \psi_1)$ that models the simple constraint $1 \shortmid \alpha \shortmid \epsilon \doteq \epsilon \shortmid \alpha \shortmid 0$. Otherwise, we would have

$$\begin{aligned}
& & \pi_\epsilon(\psi_1(\varphi(\alpha))) &= \pi_0(\psi_\epsilon(\varphi(\alpha))) & & \\
&\Longrightarrow & \psi_1(\varphi(\alpha)) &= \pi_0(\varphi(\alpha)) & & \\
&\Longrightarrow & \psi_1(\sigma \to \tau) &= \sigma & & \text{where } \varphi(\alpha) = \sigma \to \tau \\
&\Longrightarrow & \mathrm{depth}(\psi_1(\sigma \to \tau)) &= \mathrm{depth}(\sigma) & & \\
&\Longrightarrow & \mathrm{depth}(\psi_1(\sigma)) &< \mathrm{depth}(\sigma) & & \text{which is a contradiction}
\end{aligned}$$

Intuitively, the simple constraint $1 \shortmid \alpha \shortmid \epsilon \doteq \epsilon \shortmid \alpha \shortmid 0$ corresponds to an unbounded computation that transforms arbitrary many 1s on the left stack to 0s on the right stack (cf. Section 4).

The following Lemma 14 describes in which cases a simple constraint is modeled.

▶ **Lemma 14.** We have $(\varphi, \psi_0, \psi_1) \models a \shortmid \alpha \shortmid \epsilon \doteq \epsilon \shortmid \beta \shortmid b$ iff one of the following conditions holds

- $b = 0$ and $\psi_a(\varphi(\alpha)) \to \tau = \varphi(\beta)$ for some term $\tau \in \mathbb{T}$
- $b = 1$ and $\sigma \to \psi_a(\varphi(\alpha)) = \varphi(\beta)$ for some term $\sigma \in \mathbb{T}$

Finally, we identify the following semi-unification problem based on simple constraints. The importance of this restriction is pointed out in [17, Sec. 4], and its undecidability implies the undecidability of semi-unification (Theorem 1). Intuitively, we will use a simple constraint $a \shortmid \alpha \shortmid \epsilon \doteq \epsilon \shortmid \beta \shortmid b$ to represent a stack machine transition from state α to state β, removing the symbol a from the left stack and adding the symbol b to the right stack.

▶ **Problem 15** (Simple Semi-unification (SSU))**.** Given a finite set $\mathcal{C}$ of simple constraints, do there exist valuations $\varphi, \psi_0, \psi_1 : \mathbb{V} \to \mathbb{T}$ such that $(\varphi, \psi_0, \psi_1) \models \mathcal{C}$?

▶ **Theorem 1.** If simple semi-unification (Problem 15) is undecidable, then so is semi-unification (Problem 3).

Proof. Let $\mathcal{C} = \{0 \shortmid \alpha_i \shortmid \epsilon \doteq \epsilon \shortmid \beta_i \shortmid b_i \mid i = 1 \dots n\} \cup \{1 \shortmid \alpha_i \shortmid \epsilon \doteq \epsilon \shortmid \beta_i \shortmid b_i \mid i = n+1 \dots m\}$ be a set of simple constraints. We define an instance $\mathcal{D}$ of semi-unification that reflects solvability of $\mathcal{C}$ as follows.

Define $\sigma_i = \begin{cases} \alpha_i \to \gamma_i & \text{if } b_i = 0 \\ \gamma_i \to \alpha_i & \text{if } b_i = 1 \end{cases}$, where γ_i is fresh for $i = 1 \dots m$. Define $\mathcal{D}$ as (for convenience, we start indexing inequalities from 0)

$$\sigma_1 \to \dots \to \sigma_n \leq_0 \beta_1 \to \dots \to \beta_n$$
$$\sigma_{n+1} \to \dots \to \sigma_m \leq_1 \beta_{n+1} \to \dots \to \beta_m$$

First, by Lemma 14, if $\mathcal{D}$ has a solution φ, ψ_0, ψ_1, then $\psi_0(\varphi(\sigma_i)) = \varphi(\beta_i)$ for $i = 1 \dots n$, and $\psi_1(\varphi(\sigma_i)) = \varphi(\beta_i)$ for $i = n+1 \dots m$. Therefore, $(\varphi, \psi_0, \psi_1) \models \mathcal{C}$.

Second, assume $(\varphi, \psi_0, \psi_1) \models \mathcal{C}$. Define $\varphi' : \mathbb{V} \to \mathbb{T}$ such that $\varphi'(\gamma_i) = \gamma_i$ for $i = 1 \dots m$, and otherwise $\varphi'(\alpha) = \varphi(\alpha)$. For $a \in \mathbb{B}$, define $\psi'_a : \mathbb{V} \to \mathbb{T}$ such that $\psi'_a(\gamma_i) = \pi_{(1-b_i)}(\varphi(\beta_i))$ for $i = 1 \dots m$, and otherwise $\psi'_a(\alpha) = \psi_a(\alpha)$. By Lemma 14, $\varphi', \psi'_0, \psi'_1$ solve $\mathcal{D}$. ◀

3 Stack Machine Preliminaries

Instead of working with Turing machines (or symmetric intercell Turing machines of [17]), we use a more convenient computational model of simple stack machines (Definition 16). Intuitively, simple stack machines are a mechanization-friendly presentation of space-bounded Turing machines (cf. proof of Theorem 2).

▶ **Definition 16** (Simple Stack Machine ($\mathcal{M}$))**.** Let p, q range over a countably infinite set $\mathbb{S}$ of *states*. A *simple stack machine* $\mathcal{M}$ is a finite set of *instructions* of shape either $ap \longrightarrow qb$ or $pa \longrightarrow bq$, where $p, q \in \mathbb{S}$ and $a, b \in \mathbb{B}$.

A *configuration* is a triple $s \shortmid p \shortmid t$, where $p \in \mathbb{S}$ is a state, $s \in \mathbb{B}^*$ is the *left stack*, and $t \in \mathbb{B}^*$ is the *right stack*. The set of all configurations is denoted by $\mathbb{C}$.

The *step relation* $(\longrightarrow_{\mathcal{M}}) \subseteq \mathbb{C} \times \mathbb{C}$ on configurations is given by

- $sa \shortmid p \shortmid t \longrightarrow_{\mathcal{M}} s \shortmid q \shortmid bt$ if $(ap \longrightarrow qb) \in \mathcal{M}$
- $s \shortmid p \shortmid at \longrightarrow_{\mathcal{M}} sb \shortmid q \shortmid t$ if $(pa \longrightarrow bq) \in \mathcal{M}$

The *reachability relation* $(\longrightarrow^*_{\mathcal{M}}) \subseteq \mathbb{C} \times \mathbb{C}$ on configurations is the reflexive, transitive closure of $(\longrightarrow_{\mathcal{M}})$. For brevity, we say *machine* for simple stack machine.

▶ **Example 17.** Consider the machine $\mathcal{M} = \{(1p \longrightarrow p0)\}$, which pops 1s from the left stack and pushes 0s onto the right stack.

We have that from the configuration $X = 1^n \shortmid p \shortmid \epsilon$ the configurations $Y_m = 1^m \shortmid p \shortmid 0^{n-m}$ such that $m \leq n$ are reachable, i.e. $X \longrightarrow^*_{\mathcal{M}} Y_m$ for $m = 0 \dots n$.

▶ **Definition 18** (Deterministic)**.** A machine $\mathcal{M}$ is *deterministic* if for all configurations $X, Y, Z \in \mathbb{C}$ such that $X \longrightarrow_{\mathcal{M}} Y$ and $X \longrightarrow_{\mathcal{M}} Z$ we have $Y = Z$.

▶ **Remark 19.** The step relation for Turing machines is naturally connected to the step relation for simple stack machines as follows. Say a Turing machine reading a symbol a in state x writes a symbol b, transitions into a state y, and moves right. This local behavior is described by the instructions $((x,a)0 \longrightarrow b(y,0))$ and $((x,a)1 \longrightarrow b(y,1))$, where $(x,a),(y,0),(y,1) \in \mathbb{S}$. The left (resp. right) stack describes the Turing machine tape left (resp. right) of the current head position.

A distinctive machine feature is preservation of total available space under reachability (Lemma 21).

▶ **Definition 20** (Word Length (length))**.** The function length : $\mathbb{B}^* \to \mathbb{N}$ is such that

$$\mathrm{length}(\epsilon) = 0 \qquad \mathrm{length}(av) = 1 + \mathrm{length}(v)$$

▶ **Lemma 21.** If $s{\mid}p{\mid}t \longrightarrow^*_{\mathcal{M}} v{\mid}q{\mid}w$, then $\mathrm{length}(s) + \mathrm{length}(t) = \mathrm{length}(v) + \mathrm{length}(w)$.

Proof. Instructions preserve the sum of stack lengths. ◀

Since machines operate in bounded space (as opposed to Turing machines that operate on infinite tape), most machine properties, such as reachability (Lemma 22), are decidable. This is most useful for a fully constructive mechanization.

▶ **Lemma 22.** It is decidable, whether for a machine $\mathcal{M}$ and configurations $X, Y \in \mathbb{C}$, we have $X \longrightarrow^*_{\mathcal{M}} Y$.

Proof. By Lemma 21, the number of configurations reachable from X is finite and can be searched exhaustively. ◀

Although boundedness (is for any configuration X the number of configurations reachable from X finite?) is a trivially true machine property, uniform boundedness (Problem 26) is undecidable (Theorem 2).

▶ **Definition 23** (Uniformly Bounded)**.** A machine $\mathcal{M}$ is *uniformly bounded* by a natural number $n \in \mathbb{N}$ if for all configurations $X \in \mathbb{C}$ we have

$$|\{Y \in \mathbb{C} \mid X \longrightarrow^*_{\mathcal{M}} Y\}| \leq n$$

For brevity, we say that $\mathcal{M}$ is uniformly bounded if $\mathcal{M}$ is uniformly bounded by some $n \in \mathbb{N}$.

The following Example 24 illustrates a uniformly bounded machine.

▶ **Example 24.** The machine $\mathcal{M} = \{(0p \longrightarrow q1), (q1 \longrightarrow 1p), (1p \longrightarrow q0), (q0 \longrightarrow 0p)\}$ is (by case analysis) uniformly bounded by $n = 4$. For instance, in case of a configuration $X = sa{\mid}p{\mid}t$, where $a \in \mathbb{B}$ and $s, t \in \mathbb{B}^*$, we have

$$|\{Y \in \mathbb{C} \mid X \longrightarrow^*_{\mathcal{M}} Y\}| = |\{sa{\mid}p{\mid}t, s{\mid}q{\mid}(1-a)t, s(1-a){\mid}p{\mid}t, s{\mid}q{\mid}at\}| = 4 \leq n$$ ⌟

Complementarily, the following Example 25 illustrates a machine that is not uniformly bounded. As will be shown in Section 4, this is because the simple constraint $1{\mid}\alpha{\mid}\epsilon \doteq \epsilon{\mid}\alpha{\mid}0$ in Example 13 has no model.

▶ **Example 25.** The machine $\mathcal{M} = \{(1p \longrightarrow p0)\}$ from Example 17 is not uniformly bounded, because for any $n \in \mathbb{N}$ and the configuration $X = 1^n{\mid}p{\mid}\epsilon$ we have

$$|\{Y \in \mathbb{C} \mid X \longrightarrow^*_{\mathcal{M}} Y\}| = |\{1^m{\mid}p{\mid}0^{n-m} \mid 0 \leq m \leq n\}| = n + 1 > n$$ ⌟

▶ **Problem 26** (Uniform Boundedness of Deterministic Simple Stack Machines (UBDSSM))**.** Given a deterministic machine $\mathcal{M}$, is $\mathcal{M}$ is uniformly bounded?

The intuition in the above Remark 19 is used in the following Theorem 2 to connect unbounded simple stack machines to immortal Turing machines.

▶ **Theorem 2.** Uniform boundedness of deterministic simple stack machines (Problem 26) is undecidable.

Proof. Weak truth-table reduction from Turing machine mortality [12]. Let $\mathcal{T}$ be a Turing machine with moving tape over the alphabet $\mathbb{B}$ having states Q and transition function $\delta : Q \times \mathbb{B} \to Q \times \mathbb{B} \times \{L, R\}$. A generalized instantaneous description (GID)[1] of $\mathcal{T}$ is a pair $(x, T) \in Q \times \mathbb{B}^{\mathbb{Z}}$, where x is the current state and T is the current tape content with the currently scanned symbol $T(0)$.

Let $(Q \times \mathbb{B}) \subseteq \mathbb{S}$. Define a simple stack machine $\mathcal{M}$ having as instructions

- $(0(x,a) \longrightarrow (y,0)b)$ and $(1(x,a) \longrightarrow (y,1)b)$ if $\delta(x,a) = (y,b,L)$
- $((x,a)0 \longrightarrow b(y,0))$ and $((x,a)1 \longrightarrow b(y,1))$ if $\delta(x,a) = (y,b,R)$

If $\mathcal{T}$ is deterministic, then so is $\mathcal{M}$. Clearly, any finite number of $\mathcal{T}$-transitions corresponds to $\mathcal{M}$-steps for a large enough starting configuration.

We now show that if we can decide whether $\mathcal{M}$ is uniformly bounded, then we can decide whether $\mathcal{T}$ is immortal, i.e. that $\mathcal{T}$ has a GID which has no terminal successor.

First, assume that $\mathcal{M}$ is uniformly bounded by n. From a GID (x, T) we have that $\mathcal{T}$ cannot scan symbols initially positioned at i such that $i < -n$ or $i > n$. Therefore, $\mathcal{T}$ is immortal iff it loops in space $2n + 1$, which is decidable by exhaustive search.

Second, assume that every GID in $\mathcal{T}$ has a terminal successor. We use the fan theorem (as formulated by [4]) to show that $\mathcal{M}$ is uniformly bounded. Let $B = B_\top \cup B_\bot$, where $B_\top$ is the set of binary words that encode terminating computational histories (finite sequences of GIDs in bounded space) in $\mathcal{T}$, and let $B_\bot$ be the set of binary words that cannot be extended to encode a terminating computational history. Since every GID in $\mathcal{T}$ has a terminal successor, membership in B is decidable and B is a *bar*, i.e. every infinite binary sequence has a finite prefix in B. By the fan theorem, B is a uniform bar, i.e. there exists an $n \in \mathbb{N}$ such that any word in B has a prefix of length at most n that is in B. As a result, encoded terminating computational histories are of length at most n. Therefore, $\mathcal{M}$ is uniformly bounded by n. ◀

▶ **Remark 27.** In the above proof of Theorem 2, we deliberately use the fan theorem instead of König's lemma (used in [17, Corollary 5]). In constructive mathematics, the fan theorem, which is valid in Brouwer's intuitionism, is weaker than König's lemma (cf. [22]), which is valid classically.

▶ **Remark 28.** Peculiarly, for counter machines, as another model of computation, uniform boundedness is decidable (similarly to [14, Thm. 2]), whereas boundedness is not (similarly to [14, Thm. 1]). For simple stack machines it is vice versa.

3.1 Narrow Configurations

Clearly, a configuration from which no configuration with an empty left or right stack is reachable does not fully utilize the space it is provided. Therefore, key to boundedness are configurations that have an empty left or right stack, as such configurations may require additional space to reach further configurations. Extending this thought, in this section we identify a property of configurations, which we call *narrowness* (Definition 34) which plays a pivotal role in the overall argument and is part of the main contribution.

[1] An instantaneous description (ID) requires the tape content to be 0 except for finitely many positions.

We can view machine instructions as a restricted rewriting system. Such a view induces the notion of joinable configurations (Definition 29). For deterministic machines, configuration joinability is an equivalence relation (Lemma 30) with a system of representatives (Definition 31).

▶ **Definition 29** (Joinable $(\sim_{\mathcal{M}})$). Two configurations $X, Y \in \mathbb{C}$ are *joinable* in a machine $\mathcal{M}$, written $X \sim_{\mathcal{M}} Y$, if there exists a configuration Z such that $X \longrightarrow^*_{\mathcal{M}} Z$ and $Y \longrightarrow^*_{\mathcal{M}} Z$.

▶ **Lemma 30.** If a machine $\mathcal{M}$ is deterministic, then $(\sim_{\mathcal{M}})$ is an equivalence (reflexive, symmetric, transitive) relation on configurations.

Proof. Clearly, $(\sim_{\mathcal{M}})$ is reflexive and symmetric. Since $\mathcal{M}$ is deterministic, we have that $(\longrightarrow_{\mathcal{M}})$ is confluent. Therefore, for any configurations X_1, X_2, X_3, Y_1, Y_2 such that $X_1 \longrightarrow^*_{\mathcal{M}} Y_1$, $X_2 \longrightarrow^*_{\mathcal{M}} Y_1$, $X_2 \longrightarrow^*_{\mathcal{M}} Y_2$, and $X_3 \longrightarrow^*_{\mathcal{M}} Y_2$ there exists a configuration Z such that $X_1 \longrightarrow^*_{\mathcal{M}} Y_1 \longrightarrow^*_{\mathcal{M}} Z$ and $X_3 \longrightarrow^*_{\mathcal{M}} Y_2 \longrightarrow^*_{\mathcal{M}} Z$. Therefore, $(\sim_{\mathcal{M}})$ is transitive. ◀

▶ **Definition 31** (Representative $([X]_{\mathcal{M}})$). The *representative* of a configuration $X \in \mathbb{C}$ in a deterministic machine $\mathcal{M}$, written $[X]_{\mathcal{M}}$, is the lexicographically smallest configuration Y such that $X \sim_{\mathcal{M}} Y$.

▶ **Lemma 32.** For configurations $X, Y \in \mathbb{C}$, we have $[X]_{\mathcal{M}} = [Y]_{\mathcal{M}}$ iff $X \sim_{\mathcal{M}} Y$.

▶ **Remark 33.** By Lemma 21 and Lemma 22 the representative $[X]_{\mathcal{M}}$ of a configuration X in $\mathcal{M}$ is computable, and joinability $(\sim_{\mathcal{M}})$ is decidable.

Next, we identify a key property (Definition 34) of configurations, that connects machine computation with semi-unification (cf. Section 4).

▶ **Definition 34** (Narrow). A configuration X is *narrow* in a machine $\mathcal{M}$, if there exists a state $p \in \mathbb{S}$ and a word $s \in \mathbb{B}^*$ such that $X \sim_{\mathcal{M}} s\lvert p\rvert\epsilon$.

▶ **Remark 35.** For a state $p \in \mathbb{S}$, the configuration $\epsilon\lvert p\rvert\epsilon$ is narrow in any machine $\mathcal{M}$.

▶ **Remark 36.** Similarly to Lemma 22, it is decidable, whether for a machine $\mathcal{M}$ and configuration $X \in \mathbb{C}$, we have that X is narrow in $\mathcal{M}$.

▶ **Example 37.** In the machine $\mathcal{M} = \{(p1 \longrightarrow 0r), (1q \longrightarrow r1)\}$ the configuration $0\lvert p\rvert 11$ is narrow because $0\lvert p\rvert 11 \longrightarrow^*_{\mathcal{M}} 00\lvert r\rvert 1 \longleftarrow^*_{\mathcal{M}} 001\lvert q\rvert\epsilon$, that is we have $0\lvert p\rvert 11 \sim_{\mathcal{M}} 001\lvert q\rvert\epsilon$.

Narrow configurations play a pivotal role for uniform boundedness (Lemma 38 and Lemma 39). Additionally, narrowness is *the* decisive property which we use to construct solutions for semi-unification instances (Definition 41 and Definition 42).

▶ **Lemma 38.** If a machine $\mathcal{M}$ is uniformly bounded, then there exists $m \in \mathbb{N}$ such that for all narrow in $\mathcal{M}$ configurations $s\lvert p\rvert t \in \mathbb{C}$ we have $\text{length}(t) \leq m$.

Proof. If $s\lvert p\rvert t$ is narrow in $\mathcal{M}$, then there are a configurations $s'\lvert p'\rvert\epsilon$ and $v\lvert q\rvert w$ such that $s\lvert p\rvert t \longrightarrow^*_{\mathcal{M}} v\lvert q\rvert w$ and $s'\lvert p'\rvert\epsilon \longrightarrow^*_{\mathcal{M}} v\lvert q\rvert w$. If $\mathcal{M}$ is uniformly bounded by n, then we have $|\,\text{length}(t) - \text{length}(w)| \leq n$ and $|\,\text{length}(\epsilon) - \text{length}(w)| \leq n$. Therefore, $\text{length}(t) \leq 2n$. ◀

▶ **Lemma 39.** Let $\mathcal{M}$ be a deterministic machine. If there exists $m \in \mathbb{N}$ such that for all narrow in $\mathcal{M}$ configurations $\epsilon{\shortmid}p{\shortmid}t \in \mathbb{C}$ we have $\text{length}(t) \leq m$, then $\mathcal{M}$ is uniformly bounded.

Proof. Let $m \in \mathbb{N}$ be such that for all narrow in $\mathcal{M}$ configurations $\epsilon{\shortmid}p{\shortmid}t \in \mathbb{C}$ we have $\text{length}(t) \leq m$. Let $n \in \mathbb{N}$ and let $X = s{\shortmid}p{\shortmid}t$ reach at least n configurations such that $(\text{length}(s) + \text{length}(t))$ is minimal. We show that $\mathcal{M}$ is uniformly bounded by showing

$$n \leq 1 + |\{s'{\shortmid}p'{\shortmid}t' \in \mathbb{C} \mid \text{length}(s') + \text{length}(t') \leq m \text{ and } p' \text{ occurs in } \mathcal{M}\}| \qquad (\star)$$

We have $X \longrightarrow^*_{\mathcal{M}} \epsilon{\shortmid}q{\shortmid}w$ for some state $q \in \mathbb{S}$ and word $w \in \mathbb{B}^*$. Otherwise, left stacks of all configurations reachable from X would have the same prefix, which could be removed. Similarly, we have $X \longrightarrow^*_{\mathcal{M}} v{\shortmid}r{\shortmid}\epsilon$ for some state $r \in \mathbb{S}$ and word $v \in \mathbb{B}^*$. Since $\mathcal{M}$ is deterministic, $(\longrightarrow_{\mathcal{M}})$ is confluent. Therefore, the configuration $\epsilon{\shortmid}q{\shortmid}w$ is narrow in $\mathcal{M}$.

Finally, by Lemma 21, for any configuration $s'{\shortmid}p'{\shortmid}t'$ such that $X \longrightarrow^*_{\mathcal{M}} s'{\shortmid}p'{\shortmid}t'$ we have $\text{length}(s') + \text{length}(t') = \text{length}(s) + \text{length}(t) = \text{length}(w) \leq m$, showing $(\star)$. ◀

4 Undecidability of Semi-unification

In this section we fix a deterministic machine $\mathcal{M}$. Our goal is to construct a *specific instance* $\mathcal{C}_{\mathcal{M}}$ (Definition 40) of simple semi-unification such that the machine $\mathcal{M}$ is uniformly bounded if (Lemma 48) and only if (Lemma 45) $\mathcal{C}_{\mathcal{M}}$ is solvable.

For brevity, we omit $\mathcal{M}$ in notations in this section, i.e. we write $(\sim)$ for $(\sim_{\mathcal{M}})$, write $\mathcal{C}$ for $\mathcal{C}_{\mathcal{M}}$, say narrow for narrow in $\mathcal{M}$, etc. All definitions in this section tacitly depend on $\mathcal{M}$.

Let us tacitly inject $\mathbb{S}$ into $\mathbb{V}$, i.e. $\mathbb{S} \subseteq \mathbb{V}$. Additionally, for each configuration $X \in \mathbb{C}$ we fix a distinct variable $\alpha_X \in \mathbb{V}$.

▶ **Definition 40** (Specific instance $\mathcal{C}$)**.** The set $\mathcal{C}$ of simple constraints is given by

$$\mathcal{C} = \{a{\shortmid}p{\shortmid}\epsilon \doteq \epsilon{\shortmid}q{\shortmid}b \mid (ap \longrightarrow qb) \in \mathcal{M}\} \cup \{b{\shortmid}q{\shortmid}\epsilon \doteq \epsilon{\shortmid}p{\shortmid}a \mid (pa \longrightarrow bq) \in \mathcal{M}\}$$

4.1 Uniform Boundedness of $\mathcal{M}$ to Solvability of $\mathcal{C}$

In this subsection we assume that $\mathcal{M}$ is uniformly bounded and construct a solution φ, ψ_0, ψ_1 (Definition 42) for $\mathcal{C}$. Surprisingly, this can be done directly via the following function ζ (Definition 41), based on the notion of narrow configurations (Definition 34).

▶ **Definition 41** (ζ)**.** If $\mathcal{M}$ is uniformly bounded, then the function $\zeta : \mathbb{C} \to \mathbb{T}$ is given by

$$\zeta(s{\shortmid}p{\shortmid}t) = \begin{cases} \zeta(s{\shortmid}p{\shortmid}t0) \to \zeta(s{\shortmid}p{\shortmid}t1) & \text{if } s{\shortmid}p{\shortmid}t \text{ is narrow} \\ \alpha_{[s{\shortmid}p{\shortmid}t]} & \text{otherwise} \end{cases}$$

By Lemma 38, ζ is well-defined and computable (cf. Remark 36 and Remark 33). Computability of ζ is essential for a fully constructive argument.

▶ **Definition 42** (Valuations φ, ψ_0, ψ_1)**.** The valuation $\varphi : \mathbb{V} \to \mathbb{T}$ is such that

$$\varphi(p) = \zeta(\epsilon{\shortmid}p{\shortmid}\epsilon) \qquad (\text{otherwise } \varphi(\alpha) = \alpha)$$

For $a \in \mathbb{B}$, the valuation $\psi_a : \mathbb{V} \to \mathbb{T}$ is such that

$$\psi_a(\alpha_{s{\shortmid}p{\shortmid}t}) = \zeta(as{\shortmid}p{\shortmid}t) \qquad (\text{otherwise } \psi_a(\alpha) = \alpha)$$

The function ζ respects joinability (Lemma 43), i.e. it can be lifted to $(\sim)$ equivalence classes.

▶ **Lemma 43.** For configurations $X, Y \in \mathbb{C}$ such that $X \sim Y$ we have $\zeta(X) = \zeta(Y)$.

Proof. We show $\zeta(s|p|t) = \zeta(v|q|w)$ by induction on $\mathrm{depth}(\zeta(s|p|t))$.
Case $s|p|t$ is narrow: By Lemma 30, the configuration $v|q|w$ is narrow. Therefore,

$$\zeta(s|p|t) = \zeta(s|p|t0) \to \zeta(s|p|t1) \overset{\text{(IH)}}{=} \zeta(v|q|w0) \to \zeta(v|q|w1) = \zeta(v|q|w)$$

Case $s|p|t$ is not narrow: By Lemma 30, the configuration $v|q|w$ is not narrow. Therefore,

$$\zeta(s|p|t) = \alpha_{[s|p|t]} \overset{\text{Lem. 32}}{=} \alpha_{[v|q|w]} = \zeta(v|q|w)$$ ◀

Since the function ζ respects joinability, it absorbs ψ_0 and ψ_1 (Lemma 44).

▶ **Lemma 44.** For $a \in \mathbb{B}$ and configuration $s|p|t \in \mathbb{C}$, we have $\psi_a(\zeta(s|p|t)) = \zeta(as|p|t)$.

Proof. We show $\psi_a(\zeta(s|p|t)) = \zeta(as|p|t)$ by induction on $\mathrm{depth}(\zeta(s|p|t))$.
Case $s|p|t$ is narrow: We have that $as|p|t$ is narrow, and

$$\begin{aligned}\psi_a(\zeta(s|p|t)) &= \psi_a(\zeta(s|p|t0) \to \zeta(s|p|t1)) = \psi_a(\zeta(s|p|t0)) \to \psi_a(\zeta(s|p|t1))\\ &\overset{\text{(IH)}}{=} \zeta(as|p|t0) \to \zeta(as|p|t1) = \zeta(as|p|t)\end{aligned}$$

Case $s|p|t$ is not narrow: Let $v|q|w = [s|p|t]$. We have

$$\psi_a(\zeta(s|p|t)) = \psi_a(\alpha_{[s|p|t]}) = \zeta(av|q|w) \overset{\text{Lem. 43}}{=} \zeta(as|p|t)$$ ◀

As a result, the valuations φ, ψ_0, ψ_1 solve $\mathcal{C}$ (Lemma 45).

▶ **Lemma 45.** If $\mathcal{M}$ is uniformly bounded, then $(\varphi, \psi_0, \psi_1) \models \mathcal{C}$.

Proof. Configuration where both stacks are empty are trivially narrow (Remark 35).
Case $a|p|\epsilon \doteq \epsilon|q|b \in \mathcal{C}$: We have $(ap \longrightarrow qb) \in \mathcal{M}$, therefore $a|p|\epsilon \sim \epsilon|q|b$. We have

$$\psi_a(\varphi(p)) = \psi_a(\zeta(\epsilon|p|\epsilon)) \overset{\text{Lem. 44}}{=} \zeta(a|p|\epsilon) \overset{\text{Lem. 43}}{=} \zeta(\epsilon|q|b) = \pi_b(\zeta(\epsilon|q|\epsilon)) = \pi_b(\varphi(q))$$

Case $b|q|\epsilon \doteq \epsilon|p|a \in \mathcal{C}$: We have $(pa \longrightarrow bq) \in \mathcal{M}$, therefore $b|q|\epsilon \sim \epsilon|p|a$. We have

$$\psi_b(\varphi(q)) = \psi_b(\zeta(\epsilon|q|\epsilon)) \overset{\text{Lem. 44}}{=} \zeta(b|q|\epsilon) \overset{\text{Lem. 43}}{=} \zeta(\epsilon|p|a) = \pi_a(\zeta(\epsilon|p|\epsilon)) = \pi_a(\varphi(p))$$ ◀

Essentially, the function ζ interprets machine configurations as terms from which the solution $(\varphi, \psi_0, \psi_1)$ of $\mathcal{C}$ is constructed. Traditionally, this step in the overall argument [17] relies on on a more complicated path equation derivability and termination of a redex contraction procedure for semi-unification. Arguably, the function ζ is the main insight of this work, as it contributes to a simpler, fully constructive translation of machine boundedness to solvability of semi-unification.

4.2 Solvability of $\mathcal{C}$ to Uniform Boundedness of $\mathcal{M}$

In this subsection we assume that there exist valuations φ, ψ_0, ψ_1 such that $(\varphi, \psi_0, \psi_1) \models \mathcal{C}$, and we show that $\mathcal{M}$ is uniformly bounded.

Intuitively, we show that joinability is sound for constraint semantics (Corollary 47) based on soundness of the step relation for constraint semantics (Lemma 46).

▶ **Lemma 46.** For configurations $X, Y \in \mathbb{C}$ such that $X \longrightarrow Y$ we have $\mathcal{C} \models X \doteq Y$.

Proof. Let φ, ψ_0, ψ_1 be valuations such that $(\varphi, \psi_0, \psi_1) \models \mathcal{C}$.
Case $sa|p|t \longrightarrow s|q|bt$**:** We have $a|p|\epsilon \doteq \epsilon|q|b \in \mathcal{C}$. Therefore, $\psi_a(\varphi(p)) = \pi_b(\varphi(q))$ and

$$\pi_t(\psi_{sa}(\varphi(p))) = \pi_t(\psi_s(\psi_a(\varphi(p)))) = \pi_t(\psi_s(\pi_b(\varphi(q)))) = \pi_{bt}(\psi_s(\varphi(q)))$$

Case $s|p|at \longrightarrow_{\mathcal{M}} sb|q|t$**:** We have $b|q|\epsilon \doteq \epsilon|p|a \in \mathcal{C}$. Therefore, $\psi_b(\varphi(q)) = \pi_a(\varphi(p))$ and

$$\pi_{at}(\psi_s(\varphi(p))) = \pi_t(\psi_s(\pi_a(\varphi(p)))) = \pi_t(\psi_s(\psi_b(\varphi(q)))) = \pi_t(\psi_{sb}(\varphi(q)))$$ ◀

▶ **Corollary 47.** For configurations $X, Y \in \mathbb{C}$ such that $X \sim Y$ we have $\mathcal{C} \models X \doteq Y$.

As a result, narrow configurations $\epsilon|p|t$ do not admit arbitrary long right stacks t, because $\pi_t(\varphi(p))$ is undefined if $\text{length}(t)$ exceeds $\text{depth}(\varphi(p))$. The bound on depth for the range of φ immediately induces a uniform bound for $\mathcal{M}$ (Lemma 48).

▶ **Lemma 48.** If there exist valuations φ, ψ_0, ψ_1 such that $(\varphi, \psi_0, \psi_1) \models \mathcal{C}$, then $\mathcal{M}$ is uniformly bounded.

Proof. Let $\epsilon|p|t \in \mathbb{C}$ be narrow, i.e. $\epsilon|p|t \sim s|q|\epsilon$ for some state $q \in \mathbb{S}$ and word $s \in \mathbb{B}^*$. By Corollary 47, we have $\pi_t(\varphi(p)) = \psi_s(\varphi(q)) \in \mathbb{T}$. Therefore,

$$\text{length}(t) \leq \max\{\text{depth}(\varphi(r)) \mid r \in \mathbb{S} \text{ and } r \text{ occurs in } \mathcal{M}\}$$

By Lemma 39, $\mathcal{M}$ is uniformly bounded. ◀

Key to the construction of a uniform bound in the above proof is the characterization of uniform boundedness via narrow configurations (Lemma 39).

4.3 Main Result

Overall, we obtain undecidability of semi-unification (Theorem 4) via undecidability of simple semi-unification (Theorem 3).

▶ **Theorem 3.** Simple semi-unification (Problem 15) is undecidable.

Proof. By Theorem 2, uniform boundedness of deterministic machines (UBDSSM) is undecidable. Section 4 gives a reduction from UBDSSM to simple semi-unification, for which correctness is shown by Lemma 45 and Lemma 48. ◀

▶ **Theorem 4.** Semi-unification is undecidable.

Proof. By Theorem 3 and Theorem 1. ◀

Let us illustrate the construction, revisiting the uniformly bounded machine of Example 24.

▶ **Example 49.** Let $\mathcal{M} = \{(0p \longrightarrow q1), (q1 \longrightarrow 1p), (1p \longrightarrow q0), (q0 \longrightarrow 0p)\}$. Then, $\mathcal{C} = \{0_{\shortmid}p_{\shortmid}\epsilon \doteq \epsilon_{\shortmid}q_{\shortmid}1, 1_{\shortmid}p_{\shortmid}\epsilon \doteq \epsilon_{\shortmid}q_{\shortmid}1, 1_{\shortmid}p_{\shortmid}\epsilon \doteq \epsilon_{\shortmid}q_{\shortmid}0, 0_{\shortmid}p_{\shortmid}\epsilon \doteq \epsilon_{\shortmid}q_{\shortmid}0\}$. Narrow in $\mathcal{M}$ configurations are $s_{\shortmid}r_{\shortmid}\epsilon$ for words $s \in \mathbb{B}^*$ and states $r \in \mathbb{S}$, and $s_{\shortmid}q_{\shortmid}a$ for words $s \in \mathbb{B}^*$ and symbols $a \in \mathbb{B}$. Therefore (not writing out representatives), we have

$$\begin{aligned}
\varphi(p) &= \zeta(\epsilon_{\shortmid}p_{\shortmid}\epsilon) = \zeta(\epsilon_{\shortmid}p_{\shortmid}0) \to \zeta(\epsilon_{\shortmid}p_{\shortmid}1) = \alpha_{[\epsilon_{\shortmid}p_{\shortmid}0]} \to \alpha_{[\epsilon_{\shortmid}p_{\shortmid}1]}\\
\varphi(q) &= \zeta(\epsilon_{\shortmid}q_{\shortmid}\epsilon) = \zeta(\epsilon_{\shortmid}q_{\shortmid}0) \to \zeta(\epsilon_{\shortmid}q_{\shortmid}1)\\
&= \big(\zeta(\epsilon_{\shortmid}q_{\shortmid}00) \to \zeta(\epsilon_{\shortmid}q_{\shortmid}01)\big) \to \big(\zeta(\epsilon_{\shortmid}q_{\shortmid}10) \to \zeta(\epsilon_{\shortmid}q_{\shortmid}11)\big)\\
&= (\alpha_{[\epsilon_{\shortmid}q_{\shortmid}00]} \to \alpha_{[\epsilon_{\shortmid}q_{\shortmid}01]}) \to (\alpha_{[\epsilon_{\shortmid}q_{\shortmid}10]} \to \alpha_{[\epsilon_{\shortmid}q_{\shortmid}11]})\\
\psi_a(\alpha_{\epsilon_{\shortmid}p_{\shortmid}b}) &= \zeta(a_{\shortmid}p_{\shortmid}b) = \alpha_{[a_{\shortmid}p_{\shortmid}b]} && \text{for } a, b \in \mathbb{B}
\end{aligned}$$

Overall, the valuations φ, ψ_0, ψ_1 model $\mathcal{C}$, i.e. $(\varphi, \psi_0, \psi_1) \models \mathcal{C}$. For example, we have $(\varphi, \psi_0, \psi_1) \models 0_{\shortmid}p_{\shortmid}\epsilon \doteq \epsilon_{\shortmid}q_{\shortmid}1$ because $0_{\shortmid}p_{\shortmid}1 \sim \epsilon_{\shortmid}q_{\shortmid}11$ and $0_{\shortmid}p_{\shortmid}1 \sim \epsilon_{\shortmid}q_{\shortmid}11$ imply

$$\begin{aligned}
\psi_0(\varphi(p)) &= \psi_0(\alpha_{[\epsilon_{\shortmid}p_{\shortmid}0]} \to \alpha_{[\epsilon_{\shortmid}p_{\shortmid}1]}) = \psi_0(\alpha_{\epsilon_{\shortmid}p_{\shortmid}0} \to \alpha_{\epsilon_{\shortmid}p_{\shortmid}1}) = \alpha_{[0_{\shortmid}p_{\shortmid}0]} \to \alpha_{[0_{\shortmid}p_{\shortmid}1]}\\
&= \alpha_{[\epsilon_{\shortmid}q_{\shortmid}10]} \to \alpha_{[\epsilon_{\shortmid}q_{\shortmid}11]} = \pi_1(\varphi(q))
\end{aligned}$$

5 Mechanization

This section provides an overview over the mechanization [2] in the Coq proof assistant of the reduction presented in Section 4.

The mechanization can be considered self-contained code supporting the mathematical argument and its constructivity. In addition, it is compatible with the framework of *synthetic undecidability results* [9, 8, 7] in *synthetic computability theory* [3].

5.1 Semi-unification

Terms (Definition 1) are mechanized in `SemiU/SemiU_prelim.v` as the inductive type

```
Inductive term : Set :=
  | atom : nat -> term
  | arr : term -> term -> term.
```

Correspondingly, application of valuations is mechanized as

```
Definition valuation : Set := nat -> term.

Fixpoint substitute (f: valuation) (t: term) : term :=
  match t with
  | atom n => f n
  | arr s t => arr (substitute f s) (substitute f t)
  end.
```

Solvability of semi-unification inequalities is mechanized as

```
Definition inequality : Set := (term * term).

Definition solution (φ : valuation) : inequality -> Prop :=
  fun '(s, t) => exists (ψ : valuation),
    substitute ψ (substitute φ s) = substitute φ t.
```

Correspondingly, semi-unification is mechanized in `SemiU/SemiU.v` as the predicate

```
Definition SemiU (p: list inequality) := exists (φ: valuation),
  forall (c: inequality), In c p -> solution φ c.
```

5.2 Simple Stack Machines

Machines (`ssm`) are mechanized in `SM/SSM_prelim.v` as lists of instructions.

```
Definition stack : Set := list bool.
Definition state : Set := nat.
Definition config : Set := stack * state * stack.
Definition dir : Set := bool.
Definition symbol : Set := bool.
Definition instruction : Set := state * state * symbol * symbol * dir.
Definition ssm : Set := list instruction.
```

For example, `(p, q, a, b, true) : instruction` corresponds to the instruction $(ap \longrightarrow qb)$, and `(p, q, b, a, false) : instruction` corresponds to the instruction $(pb \longrightarrow aq)$. This is captured by the inductive predicate `Inductive step (M : ssm) : config -> config -> Prop`, that mechanizes the step relation.

Deterministic machines (`dssm`) admit only functional step predicates and reachability (`reachable`) is the reflexive, transitive closure of `step`.

```
Definition deterministic (M: ssm) := forall (X Y Z: config),
  step M X Y -> step M X Z -> Y = Z.

Definition dssm := { M : ssm | deterministic M }.

Definition reachable (M: ssm) : config -> config -> Prop :=
  clos_refl_trans config (step M).
```

Uniform boundedness (`bounded`) of deterministic machines (`dssm`) is mechanized in `SM/DSSM_UB.v` as the predicate `DSSM_UB`.

```
Definition bounded (M: ssm) (n: nat) : Prop :=
  forall (X: config), exists (L: list config),
    (forall (Y: config), reachable M X Y -> In Y L) /\ length L <= n.

Definition DSSM_UB (M: dssm) := exists (n: nat), bounded (proj1_sig M) n.
```

5.3 Main Result

Many-one reducibility ($\preceq$) of a predicate `p : X -> Prop` to a predicate `q : Y -> Prop` is mechanized in `Reduction.v` as

```
Definition reduces X Y (p : X -> Prop) (q : Y -> Prop) :=
  exists f : X -> Y, forall x, p x <-> q (f x).
Notation "p ⪯ q" := (reduces p q) (at level 50).
```

The main result is mechanized in `SemiU/DSSM_UB_to_SemiU.v` as

```
Theorem DSSM_UB_to_SemiU : DSSM_UB ⪯ SemiU.
Proof.
  apply (reduces_transitive DSSM_UB_to_SSemiU).
  exact SSemiU_to_SemiU.
Qed.
```

The above shows that we first reduce `DSSM_UB` to simple semi-unification (mechanized in `SemiU/SSemiU.v` as the predicate `SSemiU`) and then reduce `SSemiU` to `SemiU`. Mechanization details of `DSSM_UB_to_SSemiU` are found in `SemiU/SSemiU/DSSM_UB_to_SSemiU_argument.v`.

Informative decidability of narrowness is mechanized in `DSM/DSSM/DSSM_facts.v` as

```
Lemma narrow_dec (X: config) : decidable (narrow X).
```

Based on decidability of narrowness, the key function ζ (Definition 41) is mechanized as

```
Fixpoint ζ (n: nat) (X: config) : term :=
  match n with
  | 0 => atom (embed (nf X))
  | S n =>
      match X with
      | (A, x, B) =>
        if narrow_dec (A, x, B) then
          arr (ζ n (A, x, B++[false])) (ζ n (A, x, B++[true]))
        else atom (embed (nf X))
      end
  end.
```

where `nf X` mechanizes the representative of the mechanized configuration `X` (Definition 31). The parameter `n` is initialized with a uniform bound of the underlying machine.

Finally, Lemma 45 and Lemma 48 are mechanized as

```
Lemma soundness {M: dssm} :
  DSSM_UB M -> SSemiU (SM_to_SUcs (proj1_sig M)).

Lemma completeness {M: dssm}:
  SSemiU (SM_to_SUcs (proj1_sig M)) -> DSSM_UB M.
```

Overall, the mechanization encompasses 1500 lines of code, where two thirds show machine properties (such as decidability of narrowness) and one third is dedicated to the main argument of Section 4.

6 Conclusion

Traditionally, the association of an undecidable property for Turing machines with solvability of semi-unification is, arguably, opaque. It is established via the symmetric closure of intercell Turing machines, path equation derivability, and termination of a redex contraction procedure for semi-unification [17]. The main novelty of the presented approach is the direct association of an undecidable boundedness property with solutions of semi-unification via certain (narrow) machine configurations. As a consequence, we obtain a simpler argument for the undecidability of semi-unification. Additionally, this allows for a fully constructive mechanization of a reduction from uniform boundedness of deterministic simple stack machines (Problem 26) to semi-unification (Problem 3).

There are at least two reasonable goals to pursue next.

First, there exists a larger Coq framework [9] containing various undecidability results. The mechanization presented in Section 5 is a significant part of the ongoing effort to mechanize a reduction from the Turing machine halting problem to semi-unification. It is unclear whether a comprehensive reduction can be given fully constructively, as the presented mechanization starts with uniform boundedness. The reduction from the Turing machine halting problem (as of now) requires the fan theorem (which is part of Brouwer's constructivism, but is not considered fully constructive by Bishop). Nevertheless, it is an improvement over König's lemma used in [17]. There is reason to believe, that eliminating immortality as an intermediate step may allow for a fully constructive reduction. This is why the mechanization in Section 5 starts with boundedness as opposed to immortality.

Second, related work on semi-unification mostly follows the original approach (e.g. [1, 6]). We anticipate that the more direct argument, presented in this work, can be adapted to the related scenarios. Specifically, the presented approach seems promising to realize a fully constructive mechanization of the undecidability of unification modulo synchronous distributivity [1].

References

1 Siva Anantharaman, Serdar Erbatur, Christopher Lynch, Paliath Narendran, and Michaël Rusinowitch. Unification modulo synchronous distributivity. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *Lecture Notes in Computer Science*, pages 14–29. Springer, 2012. doi:10.1007/978-3-642-31365-3_4.

2 Andrej Dudenhefner. Mechanization of a Reduction from Uniform Boundedness of Deterministic Simple Stack Machines to Semi-unification. https://github.com/uds-psl/2020-fscd-semi-unification. Accessed: 2019-12-19.

3 Andrej Bauer. First steps in synthetic computability theory. *Electr. Notes Theor. Comput. Sci.*, 155:5–31, 2006. doi:10.1016/j.entcs.2005.11.049.

4 Josef Berger. The fan theorem and uniform continuity. In S. Barry Cooper, Benedikt Löwe, and Leen Torenvliet, editors, *New Computational Paradigms, First Conference on Computability in Europe, CiE 2005, Amsterdam, The Netherlands, June 8-12, 2005, Proceedings*, volume 3526 of *Lecture Notes in Computer Science*, pages 18–22. Springer, 2005. doi:10.1007/11494645_3.

5 The Coq Proof Assistant. https://coq.inria.fr/. Accessed: 2019-12-19.

6 Lucılia Figueiredo and Carlos Camarao. Semi-unification and periodicity of turing machines. URL: https://www.researchgate.net/publication/268426611_Semi-unification_and_Periodicity_of_Turing_Machines.

7 Yannick Forster, Edith Heiter, and Gert Smolka. Verification of PCP-related computational reductions in Coq. In *Interactive Theorem Proving - 9th International Conference, ITP 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, pages 253–269, 2018. doi:10.1007/978-3-319-94821-8_15.

8 Yannick Forster, Dominik Kirst, and Gert Smolka. On synthetic undecidability in Coq, with an application to the entscheidungsproblem. In Assia Mahboubi and Magnus O. Myreen, editors, *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, Cascais, Portugal, January 14-15, 2019*, pages 38–51. ACM, 2019. doi:10.1145/3293880.3294091.

9 Yannick Forster, Dominique Larchey-Wendling, Andrej Dudenhefner, Edith Heiter, Dominik Kirst, Fabian Kunze, Gert Smolka, Simon Spies, Dominik Wehr, and Maximilian Wuttke. A Coq Library of Undecidable Problems. In *The Sixth International Workshop on Coq for Programming Languages (CoqPL 2020)*, 2020. URL: https://github.com/uds-psl/coq-library-undecidability.

10 Fritz Henglein. Type inference with polymorphic recursion. *ACM Trans. Program. Lang. Syst.*, 15(2):253–289, 1993. doi:10.1145/169701.169692.

11 Roger Hindley. The principal type-scheme of an object in combinatory logic. *Transactions of the american mathematical society*, 146:29–60, 1969.

12 Philip K. Hooper. The Undecidability of the Turing Machine Immortality Problem. *J. Symb. Log.*, 31(2):219–234, 1966. doi:10.2307/2269811.

13 Said Jahama and Assaf J. Kfoury. A general theory of semi-unification. Technical report, Boston University Computer Science Department, 1993.

14 Jarkko Kari and Nicolas Ollinger. Periodicity and immortality in reversible computing. In Edward Ochmanski and Jerzy Tyszkiewicz, editors, *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008, Torun, Poland, August 25-29, 2008, Proceedings*, volume 5162 of *Lecture Notes in Computer Science*, pages 419–430. Springer, 2008. doi:10.1007/978-3-540-85238-4_34.

15 Assaf J. Kfoury, Jerzy Tiuryn, and Pawel Urzyczyn. The Undecidability of the Semi-Unification Problem (Preliminary Report). In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 468–476. ACM, 1990. doi:10.1145/100216.100279.

16 Assaf J. Kfoury, Jerzy Tiuryn, and Pawel Urzyczyn. Type reconstruction in the presence of polymorphic recursion. *ACM Trans. Program. Lang. Syst.*, 15(2):290–311, 1993. doi:10.1145/169701.169687.

17 Assaf J. Kfoury, Jerzy Tiuryn, and Pawel Urzyczyn. The undecidability of the semi-unification problem. *Inf. Comput.*, 102(1):83–101, 1993. doi:10.1006/inco.1993.1003.

18 Hans Leiß and Fritz Henglein. A decidable case of the semi-unification problem. In Andrzej Tarlecki, editor, *Mathematical Foundations of Computer Science 1991, 16th International Symposium, MFCS'91, Kazimierz Dolny, Poland, September 9-13, 1991, Proceedings*, volume 520 of *Lecture Notes in Computer Science*, pages 318–327. Springer, 1991. doi:10.1007/3-540-54345-7_75.

19 Robin Milner. A theory of type polymorphism in programming. *J. Comput. Syst. Sci.*, 17(3):348–375, 1978. doi:10.1016/0022-0000(78)90014-4.

20 Robin Milner, Mads Tofte, and Robert Harper. *Definition of standard ML.* MIT Press, 1990.

21 Alan Mycroft. Polymorphic type schemes and recursive definitions. In Manfred Paul and Bernard Robinet, editors, *International Symposium on Programming, 6th Colloquium, Toulouse, France, April 17-19, 1984, Proceedings*, volume 167 of *Lecture Notes in Computer Science*, pages 217–228. Springer, 1984. doi:10.1007/3-540-12925-1_41.

22 Helmut Schwichtenberg. A direct proof of the equivalence between brouwer's fan theorem and könig's lemma with a uniqueness hypothesis. *J. UCS*, 11(12):2086–2095, 2005. doi:10.3217/jucs-011-12-2086.