Logical Methods in Computer Science
Volume 22, Issue 2, 2026, pp. 3:1–3:23
https://lmcs.episciences.org/

Submitted Jun. 02, 2025
Published Apr. 09, 2026

ONE IS ALL YOU NEED: ASSOCIATIVE SECOND-ORDER UNIFICATION WITHOUT FIRST-ORDER VARIABLES

DAVID M. CERNA [a,b] AND JULIAN PARSERT [c]

[a] Institute of Computer Science, Czech Academy of Sciences
e-mail address: dcerna@cs.cas.cz

[b] Dynatrace Research
e-mail address: david.cerna@dynatrace.com

[c] RPTU Kaiserslautern
e-mail address: julian.parsert@gmail.com

Abstract. We introduce a fragment of second-order unification, referred to as *Second-Order Ground Unification (SOGU)*, with the following properties: (i) only one second-order variable is allowed and (ii) first-order variables do not occur. We study an equational variant of SOGU where the signature contains *associative* binary function symbols (ASOGU) and show that Hilbert's 10^{th} problem is reducible to ASOGU unifiability, thus proving undecidability. Our reduction provides a deeper understanding of the decidability boundary for (equational) second-order unification, as previous results required first-order variable occurrences, multiple second-order variables, and/or equational theories involving *length-reducing* rewrite systems. Furthermore, our reduction holds even in the case when associativity of the binary function symbol is restricted to *power associative*, i.e. f(f(x,x),x)= f(x,f(x,x)).

1. Introduction

In general, unification is the process of equating symbolic expressions. Second-order unification concerns symbolic expressions containing function variables, i.e., variables that take expressions as arguments. Such processes are fundamental to mathematics and computer science and are central to formal methods, verification, automated reasoning, interactive theorem proving, and various other areas. In addition, formal verification methods based on *satisfiability modulo theories (SMT)* exploit various forms of unification within the underlying theories and their implementations.

In this paper, we reduce Hilbert's 10^{th} problem to a fragment of second-order unification restricted as follows: (i) only one second-order variable occurs, (ii) no first-order variables occur, and (iii) an associative binary function symbol is allowed to occur.

Essentially, our encoding maps unknowns of the given polynomial to argument positions of the unique second-order variable occurring in the unification problem. Replacement of an unknown by a non-negative integer n is equivalent to applying a substitution to the

Funded by Czech Science Foundation Grant No. 22-06414L and Cost Action CA20111 EuroProofNet and the Austrian Science Fund (FWF) project AUTOSARD (36623), the European Research Council (ERC) project FormalWeb3 (Grant ID 101156734) and LASD (Grant ID: 101089343).

DOI:10.46298/LMCS-22(2:3)2026

unification problem, which replaces $\mathbf{F}$ by a term containing n occurrences of the associated bound variable.

Such unification problems are related to recent investigations aiming to increase the expressive power of SMT by adding higher-order features [BRO+19, Tou24]. Some methods for finding SMT models use synthesis techniques [PBJK23] such as *Syntax-Guided Synthesis* (SyGuS) [ABD+15], a common approach for function synthesis problems. In some cases, SyGuS can be considered a form of equational second-order unification where (i) only one second-order variable is allowed and (ii) first-order variables do not occur. Often, enumerative SyGuS solvers use *Counterexample Guided Inductive Synthesis* (CEGIS) [ADK+18] to speed up the synthesis procedure by leveraging ground instances of the problem. In the synthesis domain of *Programming-By-Example* (PBE), the goal is to find functions that satisfy a given set of concrete input-output examples where no variables (other than the synthesis target) are present [GPS17]. Combining these developments motivates the investigation of second-order unification, including *ground* cases without first-order variables.

Already in the 1970s, Huet and Lucchesi proved the undecidability of higher-order logic [Hue73, Luc72]. Concerning the equational variant of higher-order unification [Sny90, NQ91, DJ95], undecidability follows from Huet's result. Early investigations [Gou94] discovered decidable fragments of higher-order E-unification. However, these are not interesting to the work presented in this paper. Goldfarb [Gol81] strengthened Huet's undecidability result by proving second-order unification undecidable. Both results only concern the general unification problem, thus motivating the search for decidable fragments and honing the undecidability results (See [Lev14] for a comprehensive survey). Known decidable classes include *Monadic Second-order* [Far88, LSV08], *Second-order Linear* [Lev96], *Bounded Second-order* [Sch04], and *Context Unification* [Jez14]. Concerning second-order E-unification, the authors of [OND98] present several decidable and undecidable fragments using a reduction from word unification problems and length-reducing equational theories. Undecidability of Second-order unification has been shown for the following fragments:

- two second-order variables, no first-order variables, [LV00]
- one second-order variable *with at least eight* first-order variables [GJV98],
- one second-order variable with only ground arguments and first-order variables [LV00], and
- two second-order variables over a *length-reducing* equational theory [OND98].

Interestingly, Levy [Lev14] notes that the number of second-order variables only play a minor role in the decidability of fragments, as Levy and Veanes [LV00] provide a reduction translating arbitrary second-order equations to equations containing only one second-order variable and *additional first-order variables*. These results immediately lead to the following question:

> How important are first-order variables for the undecidability of second-order unification?

To address this question, we investigate (*associative*) *second-order ground unification* where only one second-order variable (arbitrary occurrences) is allowed, *no first-order variables* occur, and some binary function symbols are interpreted as associative. To this end, we introduce two functions related to the *multiplicity operator* [dV85, AK10], the n-counter and the n-multiplier, that allow us to reason about the multiplicity of function symbols with respect to a given substitution and of function symbols introduced by the substitution, respectively. These functions allow us to describe the properties of the unification problem

number theoretically. As a result, we can reduce finding solutions to *Diophantine equations* to a *unification condition* involving the structure of the substitution and n-counter, thus proving undecidability. In particular, our contributions are as follows:

- We introduce the n-counter and n-multiplier and prove essential properties of both.
- We prove the undecidability of associative second-order unification with *one* function variable and *no* first-order variables by showing undecidability over a restricted signature: a single constant and a binary function symbol that is interpreted as power associative, i.e., f(x,f(x,x))=f(f(x,x),x).

To illustrate our encoding, let us consider a few simple examples. First, consider the polynomial $p(x) = x - 2$. We encode this polynomial as the following *unification problem* where $\mathbf{F}$ is a second-order variable, a is a constant, g is an associative binary function symbol:

$$\mathbf{F}(g(a, a)) \stackrel{?}{=} g(a, a, \mathbf{F}(a))$$

Observe that we *flattened* nested associative function symbols, i.e., intermediary occurrences of associative function symbols are dropped. Furthermore, the arity of $\mathbf{F}$ is equivalent to the number of unknowns in $p(x)$. The consecutive $\mathbf{a}$'s in the term $g(a, a, \mathbf{F}(a))$ denote the 2 in $p(x)$. The minus sign is denoted by the consecutive $\mathbf{a}$'s occurring on the right-hand side of the unification problem and the absence of any prefix on the left-hand side. The consecutive $\mathbf{a}$'s in the term $\mathbf{F}(g(a, a))$ denote the coefficient of x in $p(x)$ (i.e. 1), since subtracting the number of occurrences of $\mathbf{a}$'s in $\mathbf{F}(a)$ from occurrences in $\mathbf{F}(g(a, a))$ results in 1.

The obvious solution to the equation $p(x) = 0$ is $x = 2$. Observe that the substitution $\{F \mapsto \lambda y.g(y, y)\}$ where the bound variable occurs twice is a solution to the unification problem we derived from $p(x)$. In our encoding, the number of occurrences of a bound variable in the solution is precisely the value that should be substituted into the associated unknown.

$$\mathbf{F}(g(a, a))\{F \mapsto \lambda y.g(y, y)\} = (\lambda y.g(y, y))g(a, a) \rightarrow_\beta g(a, a, a, a)$$
$$g(a, a, \mathbf{F}(a))\{F \mapsto \lambda y.g(y, y)\} = g(a, a, (\lambda y.g(y, y))a) \rightarrow_\beta g(a, a, a, a)$$

Let us consider a slightly more complex example: $p(x) = x^2 - 4x + 3$. Note that we will only consider natural number solutions to the polynomial; thus, we have selected a quadratic with two natural number solutions for illustration. We encode this polynomial as the following *unification problem* where $\mathbf{F}$ is a second-order variable, a is a constant, g is an associative binary function symbol:

$$g(a, a, a, \mathbf{F}(\mathbf{F}(g(a, a)))) \stackrel{?}{=} \mathbf{F}(g(a, a, a, a, \mathbf{F}(a)))$$

Observe that we now have nested function variables as the degree of x is 2 in one of the monomials. Furthermore, consider the arguments to the outermost occurence of $\mathbf{F}$, $\mathbf{F}(g(a, a))$ and $g(a, a, a, a, \mathbf{F}(a))$. Together, these terms form the unification problem for the polynomial $x - 4$. When dealing with polynomials of higher degree, we group the monomials together by variable, reduce their degree by 1, and recursively call the procedure.

It is easily derived that $p(x)$ has two solutions, namely, $x = 3$ and $x = 1$. From these solutions, we can derive substitutions for the unification problem with the appropriate number of bound variables, i.e., $\{F \mapsto \lambda y.y\}$ and $\{F \mapsto \lambda y.g(y, y, y)\}$. Observe the following:

$$g(a, a, a, \mathbf{F}(\mathbf{F}(g(a, a)))\{F \mapsto \lambda y.y\} = g(a, a, a, (\lambda y.y)((\lambda y.y)g(a, a)) \rightarrow_\beta g(a, a, a, a, a)$$
$$\mathbf{F}(g(a, a, a, a, \mathbf{F}(a)))\{F \mapsto \lambda y.y\} = (\lambda y.y)(g(a, a, a, a, ((\lambda y.y))a) \rightarrow_\beta g(a, a, a, a, a)$$

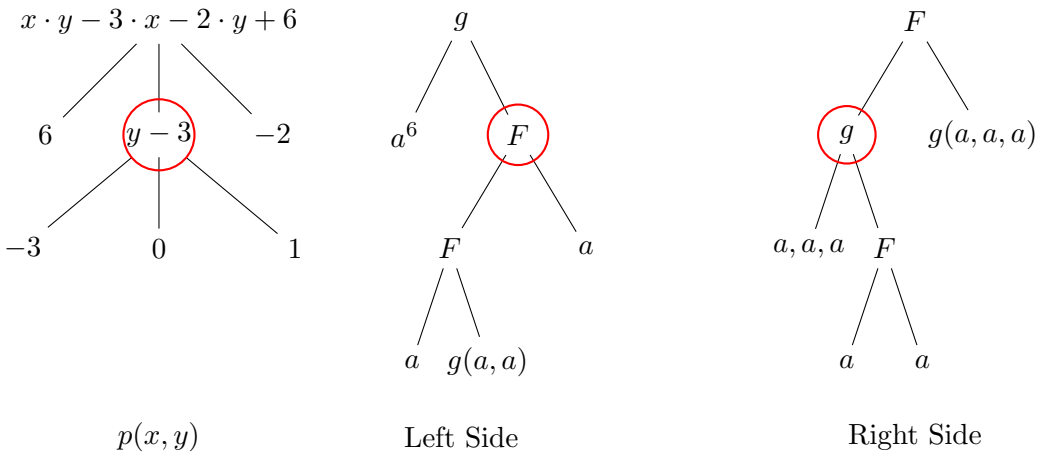

Figure 1: Construction of terms from a polynomial. The red circles denote the start of terms representing the encircled polynomial.

We leave it as an exercise to verify the other solution, as it yields a term containing 21 occurrences of a. To illustrate how the number of unknowns influences the function variable, let us consider the polynomial $p(x, y) = xy - 3x - 2y + 6$ or $(x - 2)(y - 3)$ when factored. The result unification problem is as follows (See Figure 1 for a graphical illustration):

$$g(a^6, \mathbf{F}(\mathbf{F}(a, g(a, a)), a)) \stackrel{?}{=} \mathbf{F}(g(a, a, a, \mathbf{F}(a, a)), g(a, a, a))$$

where a^n abbreviates a sequence of n occurrences of a. Observe that, within the outermost occurrence of $\mathbf{F}$, each position contains a representation of a recursively derived polynomial. Furthermore, the groupings are not independent. Let use consider the nested terms in the x position, $\mathbf{F}(a, g(a, a))$ and $g(a, a, a, \mathbf{F}(a, a))$ denoting the polynomial $y - 3$. Even though $y - 3$ does not contain the unknown x, we must include the x position as the initial problem has two unknowns. We put a single $\mathbf{a}$ in the position of the second-order variable to denote no information. Now consider the nested problem in the y position, a and $g(a, a, a)$ denoting the polynomial -2. Observe that even though y occurs in both xy and $-2y$, we cannot consider the xy monomial as it was "used up" by the first problem. Thus, we only need to consider the representation of -2. This is the interdependence mentioned above.

From the solution $x = 2$ and $y = 1$, we can build the substitution $\{F \mapsto \lambda y, z.g(y, y, z)\}$ that, when applied to the terms in the unification problem, results in a term with 15 occurrences of a. A more complex example is presented in Section 4 after we introduce the encoding and prove important properties of it.

Observe that through our encoding, any decidable class of Diophantine equations provides a decidable fragment of the second-order unification problem presented in this work. Furthermore, our reduction uses a simple encoding that guarantees the equation presented in Lemma 3.8 directly reduces to $0 = p(\overline{x_n})$ where $p(\overline{x_n})$ denotes a polynomial with integer coefficients over the variables $x_1, \cdots, x_n$.

There are likely more intricate encodings based on the n-counter and the n-multiplier, which map polynomials to more interesting unification problems. Furthermore, it remains open whether second-order ground unification is decidable, i.e., without function symbols interpreted as *power associative*.

The rest of the paper is as follows:

- In section 3, we introduce the concepts of *n-multiplier* (Definition 3.2) and *n-counter* (Definition 3.4). These concepts provide a method for counting the number of occurrences of a given constant within a term after applying a substitution with a fixed number of occurrences of each bound variable.
- We go on to show properties of these functions, specifically, how these functions relate to one another (Lemma 3.6) and how the functions relate to unifiability of two terms (Lemma 3.8).
- In section 4, we first focus on functions for manipulating polynomials. First, we introduce *monomial groups* (Definition 4.1), which group monomials by the unknowns occurring within them. Specifically, we order the unknowns and place the monomials into the group associated with the smallest unknown that occurs within it. This restructuring of the polynomials allows us to recursively descend the structure and build terms from it.
- The function recursively building terms from the polynomials is the *n-converter* (Definition 4.3), which has two modes depending on which side of the unification problem the term comes from.
- Theorem 4.9 illustrates the importance of these terms, computing the counter of the terms and then taking the difference results in the input polynomial. We use this result to complete the reduction (Lemma 4.11).

2. PRELIMINARIES

Let Σ be a *signature* of function symbols with a fixed arity. For $f \in \Sigma$, the arity of f is denoted by $\mathit{arity}(f) \geq 0$ and if $\mathit{arity}(f) = 0$ we refer to f as a *constant* (For the rest of the paper a and b are the only distinct constant symbols and f and g are the only distinct function symbols).

By $\mathcal{V}$, we denote a countably infinite set of *variables*. Furthermore, let $\mathcal{V}_i, \mathcal{V}_f \subset \mathcal{V}$ such that $\mathcal{V}_i \cap \mathcal{V}_f = \emptyset$. We refer to members of $\mathcal{V}_i$ as *individual variables*, denoted by $x, y, z, \ldots$ and members of $\mathcal{V}_f$ as *function variables*, denoted by $F, G, H, \ldots$. Members of $\mathcal{V}_f$ have an arity > 0 which we denote by $\mathit{arity}(F)$ where $F \in \mathcal{V}_f$. By $\mathcal{V}_f^n$, where $n > 0$, we denote the set of all function variables with arity n.

We refer to members of the term algebra $\mathcal{T}(\Sigma, \mathcal{V})$ as *terms*. By $\mathcal{V}_i(t)$ and $\mathcal{V}_f(t)$ ($\mathcal{V}_f^n(t)$ for $n \geq 0$), we denote the set of individual variables and function variables (with arity $= n$) occurring in t, respectively. We refer to a term t as *n-second-order ground* (n-SOG) if $\mathcal{V}_i(t) = \emptyset$, $\mathcal{V}_f(t) \neq \emptyset$ with $\mathcal{V}_f(t) \subset \mathcal{V}_f^n$, *first-order* if $\mathcal{V}_f(t) = \emptyset$, and *ground* if t is first-order and $\mathcal{V}_i(t) = \emptyset$. When possible, without causing confusion, we will abbreviate a sequence of terms $t_1, \ldots, t_n$ by $\overline{t_n}$ where $n > 0$. The *set of subterms of a term t* is denoted $\mathit{sub}(t)$. The *number of occurrences of a symbol (or variable) f in a term t* is denoted $\mathit{occ}_\Sigma(f, t)$.

A *n-second-order ground (n-SOG) unification equation* has the form $u \stackrel{?}{=}_F v$ where u and v are n-SOG terms and $F \in \mathcal{V}_f^n$ such that $\mathcal{V}_f(u) = \{F\}$ and $\mathcal{V}_f(v) = \{F\}$. A *n-second-order ground unification problem* (n-SOGU problem) is a pair $(\mathcal{U}, F)$ where $\mathcal{U}$ is a set of n-SOG unification equations and $F \in \mathcal{V}_f^n$ such that for all $u \stackrel{?}{=}_G v \in \mathcal{U}$, $G = F$. Recall from the

definition of n-SOG that $\mathcal{V}_i(u) = \mathcal{V}_i(v) = \emptyset$. When possible, without causing confusion, we will write SOG, SOGU, ... (i.e., drop the n- prefix)

We define the depth of a term t, denoted $dep(t)$ inductively as follows: (i) if $t \in \mathcal{V}_i$ or $arity(t) = 0$, then $dep(t) = 1$, (ii) $F \in \mathcal{V}_f$ and $t = F(t_1, \cdots, t_n)$, then $dep(t) = 1 + \max\{dep(t_i) \mid 1 \leq i \leq n\}$, and (iii) if $t = f(t_1, \cdots, t_n)$, then $dep(t) = 1 + \max\{dep(t_i) \mid 1 \leq i \leq n\}$

A *substitution* is a set of bindings of the form $\{F_1 \mapsto \lambda\overline{y_{l_1}}.t_1, \ldots F_k \mapsto \lambda\overline{y_{l_k}}.t_k, x_1 \mapsto s_1, \ldots, x_w \mapsto s_w\}$ where $k, w \geq 0$, for all $1 \leq i \leq k$, t_i is first-order and $\mathcal{V}_i(t_i) \subseteq \{y_1, \ldots, y_{l_i}\}$, $arity(F_i) = l_i$, and for all $1 \leq i \leq w$, s_i is ground. Given a substitution σ, $dom_f(\sigma) = \{F \mid \{F\sigma = \lambda\overline{x_n}.t \in \sigma \wedge F \in \mathcal{V}_f^n\}$ and $dom_i(\sigma) = \{x \mid x\sigma \neq x \wedge x \in \mathcal{V}_i\}$. We refer to a substitution σ as second-order when $dom_i(\sigma) = \emptyset$ and first-order when $dom_f(\sigma) = \emptyset$. We use postfix notation for substitution applications, writing $t\sigma$ instead of $\sigma(t)$. Substitutions are denoted by lowercase Greek letters. As usual, the application $t\sigma$ affects only the free variable occurrences of t whose free variable is found in $dom_i(\sigma)$ and $dom_f(\sigma)$. Furthermore, we assume $t\sigma$ to be in β-normal form unless otherwise stated. A substitution σ is a *unifier* of an SOGU problem $(\mathcal{U}, F)$, if $dom_f(\sigma) = \{F\}$, $dom_i(\sigma) = \emptyset$, and for all $u \stackrel{?}{=}_F v \in \mathcal{U}$, $u\sigma = v\sigma$.

The main result presented in the paper concerns *second-order ground E-unification* where E is a set of equational axioms.

Definition 2.1 (Equational theory [BN98])**.** *Let E be a set of equational axioms. The relation*

$$\approx_E = \{(s,t) \in \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V}) \mid E \models s \approx t\}$$

is called the equational theory *induced by E.*

The relation $\{(s,t) \in \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V}) \mid E \models (s,t)\}$ induced by a set of equalities E gives the set of equalities satisfied by all structures in the theory of E. We will use the notation $s \approx_E t$ for (s,t) belonging to this set.

In particular, we consider *associative* theories

$$\mathcal{A} = \{f(x, f(y,z)) = f(f(x,y), z) \mid f \in \Sigma_{\mathcal{A}}\}$$

where $\Sigma_{\mathcal{A}} \subset \Sigma$ such that $\Sigma_{\mathcal{A}}$ is finite, possibly empty, and for all $f \in \Sigma_{\mathcal{A}}$, $arity(f) = 2$. When possible, without causing confusion, we will use *flattened* notation for function symbols in $\Sigma_{\mathcal{A}}$, i.e. we write $f(t_1, \ldots, t_n)$ dropping intermediate occurences of f.

A *n-second-order ground $\mathcal{A}$-unification problem*, abbreviated n-ASOGU problem, is a triple $(\mathcal{U}, \mathcal{A}, F)$ where $(\mathcal{U}, F)$ is a n-second-order ground unification problem and $\mathcal{A}$ is a non-empty set of associativity axioms. A substitution σ is a *unifier* of an ASOGU problem $(\mathcal{U}, \mathcal{A}, F)$, if $dom_f(\sigma) = \{F\}$, $dom_i(\sigma) = \emptyset$, and for all $u \stackrel{?}{=}_F v \in \mathcal{U}$, $u\sigma \approx_{\mathcal{A}} v\sigma$.

In later sections, we will use the following theorem due to Matiyasevich, Robinson, Davis, and Putnam:

Theorem 2.2 (Matiyasevich–Robinson–Davis–Putnam theorem or Hilbert's 10^{th} problem [Mat93])**.** *Let $p(\overline{x})$ be a polynomial with integer coefficients. Then whether $p(\overline{x}) = 0$ has an integer solution is undecidable.*

Importantly, Theorem 2.2 also holds if we restrict the solutions to non-negative integers (natural numbers with 0).

3. n-Multipliers and n-Counters

In this section, we define and discuss the n-multiplier and n-counter functions, which allow us to encode number-theoretic problems in second-order unification. These concepts are related to the *multiplicity operator* used for finiteness results concerning the lambda calculus [dV85, AK10]. The results in this section hold for the associative variant of the unification problem, but we use the syntactic variant below to simplify the presentation. The n-multiplier and n-counter are motivated by the following simple observation about SOGU.

Observation 3.1. *Let $(\mathcal{U}, F)$ be a unifiable ASOGU problem, and σ a unifier of $(\mathcal{U}, F)$. Then for all $f \in \Sigma$ and $u \stackrel{?}{=}_F v \in \mathcal{U}$, $occ_\Sigma(f, u\sigma) = occ_\Sigma(f, v\sigma)$.*

Proof. Follows from the definition of unifier over an associative theory. □

With this observation, we now seek to relate the number of occurrences of a symbol f in a term t and a substitution σ with the number of occurrences of f in the term $t\sigma$. The *n-multiplier* counts the multiplicative effect of nested variables, while the *n-counter* counts how the multiplicative effect of nested variables affects the multiplicity of a particular symbol occurring in t.

Definition 3.2 (n-Mutiplier)**.** *Let t be an SOG term such that $\mathcal{V}_f(t) = \{F\}$, $F \in \mathcal{V}_f^n$, and $h_1, \ldots, h_n \geq 0$ are non-negative integers. Then we define the n-multiplier for t at $\overline{h_n}$, denoted $\mathtt{mul}[F, \overline{h_n}, t]$, recursively as follows:*

- $\mathtt{mul}[F, \overline{h_n}, a] = 0$.
- $\mathtt{mul}[F, \overline{h_n}, f(\overline{t_n})] = \sum_{j=1}^{l} \mathtt{mul}[F, \overline{h_n}, t_j]$
- $\mathtt{mul}[F, \overline{h_n}, F(\overline{t_n})] = 1 + \sum_{i=1}^{n} h_i \cdot \mathtt{mul}[F, \overline{h_n}, t_i]$

Furthermore, let $(\mathcal{U}, F)$ be an SOGU problem. Then

$$\mathtt{mul}[F, \overline{h_n}, \mathcal{U}]_l = \sum_{u \stackrel{?}{=}_F v \in \mathcal{U}} \mathtt{mul}[F, \overline{h_n}, u] \qquad \mathtt{mul}[F, \overline{h_n}, \mathcal{U}]_r = \sum_{u \stackrel{?}{=}_F v \in \mathcal{U}} \mathtt{mul}[F, \overline{h_n}, v].$$

The *n-multiplier* captures the following property of a term: let t be an SOG term such that $\mathcal{V}_f(t) = \{F\}$, $arity(F) = n$, $f \in \Sigma$, and $\sigma = \{F \mapsto \lambda\overline{x_n}.s\}$ a substitution where

- $occ_\Sigma(f, s) \geq 1$ and $occ_\Sigma(f, t) = 0$,
- $\mathcal{V}_i(s) \subseteq \{\overline{x_n}\}$, and
- for all $1 \leq i \leq n$, $occ(x_i, s) = h_i$.

Then $occ_\Sigma(f, t\sigma) = occ_\Sigma(f, s) \cdot \mathtt{mul}[F, \overline{h_n}, t]$. Observe that the $\overline{h_n}$ captures duplication of the arguments to F within the term t. Also, it is not a requirement that $occ_\Sigma(f, t) = 0$, but making this assumption simplifies the relationship between t and $t\sigma$ for illustrative purposes. See the following for a concrete example:

Example 3.3. *Consider the term*

$$t = g\Big(F\big(g(a, F(s(a)))\big), g\Big(F(a), F(F(F(b)))\Big)\Big)$$

Then the n-multiplier of t is $3 + 2 \cdot h_1 + h_1^2$ and is derived as follows:

$$\mathtt{mul}[F, h_1, t] = \mathtt{mul}[F, h_1, F\big(g(a, F(s(a)))\big)] +$$

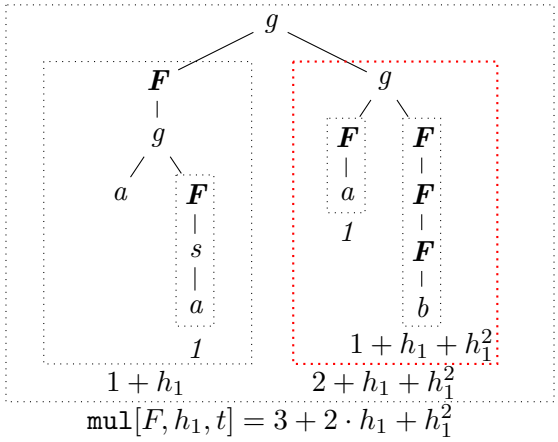

Figure 2: Computation of $\texttt{mul}[F, h_1, t]$ *(See Example 3.3).*

$$
\begin{aligned}
\texttt{mul}[F, h_1, g\big(F(a), F(F(F(b)))\big)] &\\
\texttt{mul}[F, h_1, F\big(g(a, F(s(a)))\big)] &= 1 + h_1 \cdot \texttt{mul}[F, h_1, g(a, F(s(a)))]\\
\texttt{mul}[F, h_1, g\big(F(a), F(F(F(b)))\big)] &= \texttt{mul}[F, h_1, F(a)] + \texttt{mul}[F, h_1, F(F(F(b)))]\\
\texttt{mul}[F, h_1, g(a, F(s(a)))] &= 1\\
\texttt{mul}[F, h_1, F(a)] &= 1\\
\texttt{mul}[F, h_1, F(F(F(b)))] &= 1 + h_1 \cdot \texttt{mul}[F, h_1, F(F(b))]\\
\texttt{mul}[F, h_1, F(F(b))] &= 1 + h_1
\end{aligned}
$$

Thus, when $h_1 = 2$ *we get* $\texttt{mul}[F, h_1, t] = 11$*. Observe* $occ_\Sigma(g', t\{F \mapsto \lambda x.g'(x, x)\}) = 11$*, i.e.*

$$
\begin{aligned}
F(g(a, F(s(a))))\{F \mapsto \lambda x.g'(x, x)\} &= \mathbf{g}'(t', t')\\
t' &= g(a, \mathbf{g}'(s(a), s(a)))\\
F(a)\{F \mapsto \lambda x.g'(x, x)\} &= \mathbf{g}'(a, a)\\
F(F(F(b)))\{F \mapsto \lambda x.g'(x, x)\} &= \mathbf{g}'(t'', t'')\\
t'' &= \mathbf{g}'(\mathbf{g}'(b, b), \mathbf{g}'(b, b))
\end{aligned}
$$

Given that $occ_\Sigma(g', t) = 0$*, all occurrences of* g' *are introduced by* σ*. See Figure 2 for a tree representation of the computation.*

Next, we introduce the n-counter function. Informally, given an SOG term t such that $\mathcal{V}_f(t) = \{F\}$, a symbol $g \in \Sigma$, and a substitution σ with $dom_f(\sigma) = \{F\}$, the n-counter captures the number of occurrences of g in $t\sigma$. The substitution can change the occurrences of g in two ways: (i) by introducing additional occurrences of g and (ii) by duplicating occurrences of g that occur as arguments to F. The n-counter function captures both of these changes. In particular, we are interested in occurrences of the latter type as these are exploited by the reduction presented in Section 4.

Definition 3.4 (n-Counter)**.** *Let* $g \in \Sigma$*,* t *be an SOG term such that* $\mathcal{V}_f(t) = \{F\}$ *and* $F \in \mathcal{V}_f^n$*, and* $h_1, \ldots, h_n \geq 0$ *are non-negative integers. Then we define the* n*-counter of* g *for* t *at* $\overline{h_n}$*, denoted* $\texttt{cnt}[F, \overline{h_n}, g, t]$*, recursively as follows:*

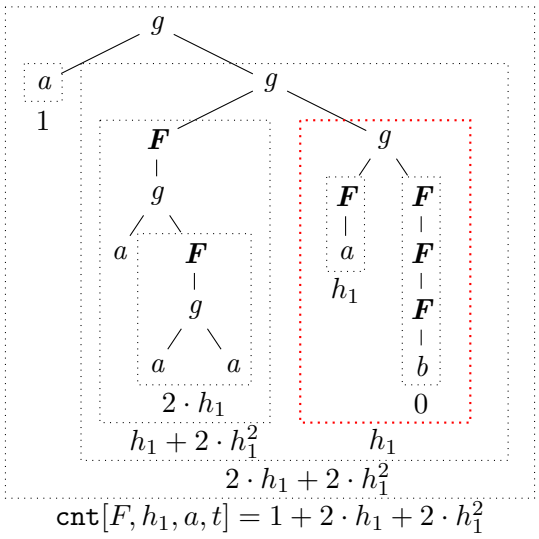

Figure 3: Computation of $\texttt{cnt}[F, h_1, a, t]$ *(See Example 3.5).*

- $\texttt{cnt}[F, \overline{h_n}, a, b] = 0$
- $\texttt{cnt}[F, \overline{h_n}, a, a] = 1$
- $\texttt{cnt}[F, \overline{h_n}, f, g(\overline{t_l})] = \sum_{j=1}^{l} \texttt{cnt}[F, \overline{h_n}, f, t_j]$
- $\texttt{cnt}[F, \overline{h_n}, f, f(\overline{t_l})] = 1 + \sum_{j=1}^{l} \texttt{cnt}[F, \overline{h_n}, f, t_j]$
- $\texttt{cnt}[F, \overline{h_n}, f, F(\overline{t_n})] = \sum_{i=1}^{n} h_i \cdot \texttt{cnt}[F, \overline{h_n}, f, t_i]$

Furthermore, let $(\mathcal{U}, F)$ *be an SOGU problem. Then*

$$\texttt{cnt}[F, \overline{h_n}, g, \mathcal{U}]_l = \sum_{u \overset{?}{=}_F v \in \mathcal{U}} \texttt{cnt}[F, \overline{h_n}, g, u] \qquad \texttt{cnt}[F, \overline{h_n}, g, \mathcal{U}]_r = \sum_{u \overset{?}{=}_F v \in \mathcal{U}} \texttt{cnt}[F, \overline{h_n}, g, v].$$

The *n-counter* captures the following property of a term: let t be an SOG term such that $\mathcal{V}_f(t) = \{F\}$, $arity(F) = n$, $f \in \Sigma$, and $\sigma = \{F \mapsto \lambda \overline{x_n}.s\}$ a substitution where

- $occ_\Sigma(f, s) = 0$,
- $\mathcal{V}_i(s) \subseteq \{\overline{x_n}\}$, and
- for all $1 \leq i \leq n$, $occ_\Sigma(x_i, s) = h_i$.

Then $occ_\Sigma(f, t\sigma) = \texttt{cnt}[F, \overline{h_n}, f, t]$. Observe that the $\overline{h_n}$ captures duplication of the arguments to F within the term t. Also, it is not a requirement that $occ_\Sigma(f, s) = 0$, but making this assumption simplifies the relationship between t and $t\sigma$ for illustrative purposes. See the following for a concrete example:

Example 3.5. *Consider the term* $t =$

$$g\Big(a, g\Big(F\big(g(a, F(g(a, a)))\big), g\big(F(a), F(F(F(b)))\big)\Big)\Big).$$

The counter of t *over the symbol* a *is* $1 + 2 \cdot h_1 + 2 \cdot h_1^2$ *and is derived as follows:*

$$\texttt{cnt}[F, h_1, a, t] \; = \texttt{cnt}[F, h_1, a, a] +$$

$$\begin{aligned}
&\texttt{cnt}[F, h_1, a, F\big(g(a, F(g(a, a)))\big)] + \\
&\texttt{cnt}[F, h_1, a, g\big(F(a), F(F(F(b)))\big)] \\
\texttt{cnt}[F, h_1, a, a] &= 1 \\
\texttt{cnt}[F, h_1, a, F\big(g(a, F(g(a, a)))\big)] &= h_1 \cdot \texttt{cnt}[F, h_1, a, g\big(F(a), F(F(F(b)))\big)] \\
\texttt{cnt}[F, h_1, a, g(F(a), F(F(F(b))))] &= \texttt{cnt}[F, h_1, a, F(a)] + \texttt{cnt}[F, h_1, a, F(F(F(b)))] \\
\texttt{cnt}[F, h_1, a, g(a, F(g(a, a)))] &= 1 + 2 \cdot h_1 \\
\texttt{cnt}[F, h_1, a, F(a)] &= h_1 \\
\texttt{cnt}[F, h_1, a, F(F(F(b)))] &= 0
\end{aligned}$$

Thus, when $h_1 = 2$ we get $\texttt{cnt}[F, h_1, a, t] = 13$ (the sum of the three main components as illustrated above). Observe $occ_\Sigma(a, t\{F \mapsto \lambda x.g(x, x)\}) = 13$. Applying the substitution $\sigma = \{F \mapsto \lambda x.g(x, x)\}$ to t results in the following

$$\begin{aligned}
F(g(a, F(g(a, a))))\sigma &= g(t', t') \\
t' &= g(\boldsymbol{a}, g(g(\boldsymbol{a}, \boldsymbol{a}), g(\boldsymbol{a}, \boldsymbol{a}))) \\
F(a)\sigma &= g(\boldsymbol{a}, \boldsymbol{a}) \\
F(F(F(b)))\sigma &= g(t'', t'') \\
t'' &= g(g(b, b), g(b, b))
\end{aligned}$$

See Figure 3 for a tree representation of the computation.

The n-multiplier and n-counter operators count the occurrences of symbols in $t\sigma$ by only considering t. Observe that $h_1, \ldots, h_n$ denote the multiplicity of the arguments of the function variable F within a substitution σ with domain F. For any symbol c occurring in t, the n-counter predicts the occurences of c in $t\sigma$ by only considering t and $h_1, \ldots, h_n$. For any symbol c occurring in the range of a substitution σ, the n-multiplier predicts the occurences of c in $t\sigma$ by only considering t and $h_1, \ldots, h_n$. We now describe the relationship between the n-multiplier, n-counter, and the total occurrences of a given symbol within the term $t\sigma$.

Lemma 3.6. *Let $g \in \Sigma$, $n \geq 0$, t be an SOG term such that $\mathcal{V}_f(t) = \{F\}$, $h_1, \ldots, h_n \geq 0$ are non-negative integers, and $\sigma = \{F \mapsto \lambda\overline{x_n}.s\}$ a substitution such that $\mathcal{V}_i(s) \subseteq \{\overline{x_n}\}$ and for all $1 \leq i \leq n$ $occ_\Sigma(x_i, s) = h_i$. Then*

$$occ_\Sigma(g, t\sigma) = occ_\Sigma(g, s) \cdot \texttt{mul}[F, \overline{h_n}, t] + \texttt{cnt}[F, \overline{h_n}, g, t].$$

Proof. We prove the lemma by induction on $dep(t)$.

Base case: When $dep(t) = 1$, then either (i) t is a constant or (ii) $t = F$ and $arity(F) = 0$.

(i) Observe that $t = t\sigma$ and $\texttt{mul}[F, \overline{h_n}, t] = 0$. If $t = g$ then $\texttt{cnt}[F, \overline{h_n}, g, t] = 1$, otherwise $\texttt{cnt}[F, \overline{h_n}, g, t] = 0$. In either case, we get $occ(g, t\sigma)_\Sigma = 0 + \texttt{cnt}[F, \overline{h_n}, g, t]$.

(ii) Observe that $t\sigma = s$, $\texttt{mul}[F, t] = 1$, and $\texttt{cnt}[F, g, t] = 0$. Thus, $occ(g, t\sigma)_\Sigma = occ_\Sigma(g, s) \cdot \texttt{mul}[F, t]$ which reduces to $occ_\Sigma(g, s) = occ_\Sigma(g, s)$.

Hence, in either case, we obtain the desired result.

Step case: Now, for the induction hypothesis, we assume the lemma holds for all terms t such that $dep(t) < w + 1$ and prove the lemma for a term t' such that $dep(t) = w + 1$. Consider the following three cases:

- $t = f(t_1, \ldots, t_k)$ and $f = g$. We know by the induction hypothesis that for $1 \leq i \leq k$, $occ_\Sigma(g, t_i\sigma) = occ_\Sigma(g, s) \cdot \mathtt{mul}[F, \overline{h_n}, t_i] + \mathtt{cnt}[F, \overline{h_n}, g, t_i]$. Thus,

$$occ_\Sigma(g, t\sigma) = 1 + \sum_{i=1}^{k} occ_\Sigma(g, t_i\sigma) =$$
$$1 + occ_\Sigma(g, s) \cdot \sum_{i=1}^{k} \mathtt{mul}[F, \overline{h_n}, t_i] + \sum_{i=1}^{k} \mathtt{cnt}[F, \overline{h_n}, g, t_i] =$$
$$occ_\Sigma(g, s) \cdot \mathtt{mul}[F, \overline{h_n}, t] + \mathtt{cnt}[F, \overline{h_n}, g, t]$$

  where, by the definition of the n-multiplier and the n-counter, $\mathtt{mul}[F, \overline{h_n}, t] = \sum_{i=1}^{k} \mathtt{mul}[F, \overline{h_n}, t_i]$, and $\mathtt{cnt}[F, \overline{h_n}, g, t] = 1 + \sum_{i=1}^{k} \mathtt{cnt}[F, \overline{h_n}, g, t_i]$.
- $t = f(t_1, \ldots, t_k)$ and $f \neq g$: This case follows from the previous case except, we do not count f when counting occurrences of g and $\mathtt{cnt}[F, \overline{h_n}, g, t] = \sum_{i=1}^{k} \mathtt{cnt}[F, \overline{h_n}, g, t_i]$.
- $t = F(r_1, \ldots, r_n)$. By the induction hypothesis, we have that for all $1 \leq i \leq n$,

$$occ_\Sigma(g, r_i\sigma) = occ_\Sigma(g, s) \cdot \mathtt{mul}[F, \overline{h_n}, r_i] + \mathtt{cnt}[F, \overline{h_n}, g, r_i].$$

  We can derive the following equality and conclude the proof using the above assumption.

$$occ_\Sigma(g, t\sigma) = occ_\Sigma(g, s) + \sum_{i=1}^{n} h_i \cdot occ_\Sigma(g, r_i\sigma) =$$
$$occ_\Sigma(g, s) + occ_\Sigma(g, s) \cdot \left(\sum_{i=1}^{n} h_i \cdot \mathtt{mul}[F, \overline{h_n}, r_i] \right) + \left(\sum_{i=1}^{n} h_i \cdot \mathtt{cnt}[F, \overline{h_n}, g, r_i] \right) =$$
$$occ_\Sigma(g, s) \cdot \left(1 + \sum_{i=1}^{n} h_i \cdot \mathtt{mul}[F, \overline{h_n}, r_i] \right) + \left(\sum_{i=1}^{n} h_i \cdot \mathtt{cnt}[F, \overline{h_n}, g, r_i] \right) =$$
$$occ_\Sigma(g, s) \cdot \mathtt{mul}[F, \overline{h_n}, F(\overline{r_n})] + \mathtt{cnt}[F, \overline{h_n}, g, F(\overline{r_n})] =$$
$$occ_\Sigma(g, s) \cdot \mathtt{mul}[F, \overline{h_n}, t] + \mathtt{cnt}[F, \overline{h_n}, g, t] \qquad \square$$

When we use Lemma 3.6 in Section 4, the term s in $\sigma = \{F \mapsto \lambda\overline{x_n}.s\}$ we will not contain any occurences of g, i.e. $occ_\Sigma(g, s) = 0$. Furthermore, Lemma 3.6 captures an essential property of the n-multiplier and n-counter, which we illustrate in the following example.

Example 3.7. *Consider the term* $t =$

$$g\Big(a, g\Big(F\big(g(a, F(g(a, a)))\big), g\big(F(a), F(F(F(b)))\big)\Big)\Big).$$

and the substitution $\{F \mapsto \lambda x.g(a, g(x, x))\}$. *The* n*-counter of* a *for* t *at 2 is* $\mathtt{cnt}[F, 2, a, t] = 13$ *and the* n*-multiplier for* t *at 2 is* $\mathtt{mul}[F, 2, t] = 11$. *Observe* $occ_\Sigma(a, t\{F \mapsto \lambda x.g(a, g(x, x))\}) = 24$ *and* $occ_\Sigma(a, s) \cdot \mathtt{mul}[F, 2, t] + \mathtt{cnt}[F, 2, a, t] = 24$. *Applying the substitution* $\{F \mapsto \lambda x.g(a, g(x, x))\}$ *to* t *results in the following:*

$$F(g(a, F(g(a, a))))\{F \mapsto \lambda x.g(a, g(x, x))\} = g(\boldsymbol{a}, g(t', t'))$$
$$t' = g(\boldsymbol{a}, g(\boldsymbol{a}, g(g(\boldsymbol{a}, \boldsymbol{a}), g(\boldsymbol{a}, \boldsymbol{a})))))$$
$$F(a)\{F \mapsto \lambda x.g(a, g(x, x))\} = g(\boldsymbol{a}, g(\boldsymbol{a}, \boldsymbol{a}))$$
$$F(F(F(b)))\{F \mapsto \lambda x.g(a, g(x, x))\} = g(\boldsymbol{a}, g(t'', t''))$$

$$t'' = g(\boldsymbol{a}, g(g(\boldsymbol{a}, g(b,b)), g(\boldsymbol{a}, g(b,b))))$$

So far, we have only considered arbitrary terms and substitutions. We now apply the above results to unification problems and their solutions. In particular, a corollary of Lemma 3.6 is that there is a direct relation between the n-multiplier and n-counter of an unifiable unification problem. The following lemma describes this relation.

Lemma 3.8 (Unification Condition)**.** *Let $(\mathcal{U}, F)$ be an unifiable SOGU problem such that $\mathcal{V}_f(\mathcal{U}) = \{F\}$, $h_1, \ldots, h_n \geq 0$ are non-negative integers, and $\sigma = \{F \mapsto \lambda\overline{x_n}.s\}$ an unifier of $(\mathcal{U}, F)$ such that $\mathcal{V}_i(s) \subseteq \{\overline{x_n}\}$ and for all $1 \leq i \leq n$,$occ_\Sigma(x_i, s) = h_i$. Then for all $g \in \Sigma$,*

$$occ_\Sigma(g,s) \cdot (\mathtt{mul}[F, \overline{h_n}, \mathcal{U}]_l - \mathtt{mul}[F, \overline{h_n}, \mathcal{U}]_r) = \mathtt{cnt}[F, \overline{h_n}, g, \mathcal{U}]_r - \mathtt{cnt}[F, \overline{h_n}, g, \mathcal{U}]_l. \quad (3.1)$$

Proof. By Lemma 3.1, for any $g \in \Sigma$ and $u \stackrel{?}{=}_F v \in \mathcal{U}$, we have $occ_\Sigma(g, u\sigma) = occ_\Sigma(g, v\sigma)$ and by Lemma 3.6 we also have

$$occ_\Sigma(g, u\sigma) = occ_\Sigma(g,s) \cdot \mathtt{mul}[F, \overline{h_n}, u] + \mathtt{cnt}[F, \overline{h_n}, g, u],$$
and
$$occ_\Sigma(g, v\sigma) = occ_\Sigma(g,s) \cdot \mathtt{mul}[F, \overline{h_n}, v] + \mathtt{cnt}[F, \overline{h_n}, g, v].$$

Thus, for any $g \in \Sigma$ and $u \stackrel{?}{=}_F v \in \mathcal{U}$,

$$occ_\Sigma(g,s) \cdot \mathtt{mul}[F, \overline{h_n}, u] + \mathtt{cnt}[F, \overline{h_n}, g, u] = occ_\Sigma(g,s) \cdot \mathtt{mul}[F, \overline{h_n}, v] + \mathtt{cnt}[F, \overline{h_n}, g, v].$$

From this equation, we can derive the following:

$$occ_\Sigma(g,s) \cdot (\mathtt{mul}[F, \overline{h_n}, u] - \mathtt{mul}[F, \overline{h_n}, v]) = \mathtt{cnt}[F, \overline{h_n}, g, v] - \mathtt{cnt}[F, \overline{h_n}, g, u] \quad (3.2)$$

We can generalize Equation 3.2 to $\mathcal{U}$ by computing Equation 3.1 for each $u \stackrel{?}{=}_F v \in \mathcal{U}$ and adding the results together and thus deriving the following

$$occ_\Sigma(g,s) \cdot (\mathtt{mul}[F, \overline{h_n}, \mathcal{U}]_l - \mathtt{mul}[F, \overline{h_n}, \mathcal{U}]_r) = \mathtt{cnt}[F, \overline{h_n}, g, \mathcal{U}]_r - \mathtt{cnt}[F, \overline{h_n}, g, \mathcal{U}]_l. \quad \square$$

The *unification condition* provides a necessary condition for unifiability that we use in the undecidability proof presented in Section 4, specifically the relationship between the left and right sides of the unification equation presented in Equation 3.1. Sufficiency requires an additional assumption, namely, the signature contains at least one *associative function symbol*. The following example shows an instance of this property.

Example 3.9. *Consider the SOGU problem $F(g(a,a)) \stackrel{?}{=}_F g(F(a), F(a))$ and the unifier $\sigma = \{F \mapsto \lambda x.g(x,x)\}$. Observe that*

$$occ_\Sigma(a, g(x,x)) \cdot (\ \mathtt{mul}[F, 2, F(g(a,a))]_l - \mathtt{mul}[F, 2, g(F(a), F(a))]_r\ ) = 0 \cdot (1-2) = 0$$

and for the right side of Equation 3.1 we get

$$\mathtt{cnt}[F, h, a, g(F(a), F(a))]_r - \mathtt{cnt}[F, h, a, F(g(a,a))]_l = 4 - 4 = 0.$$

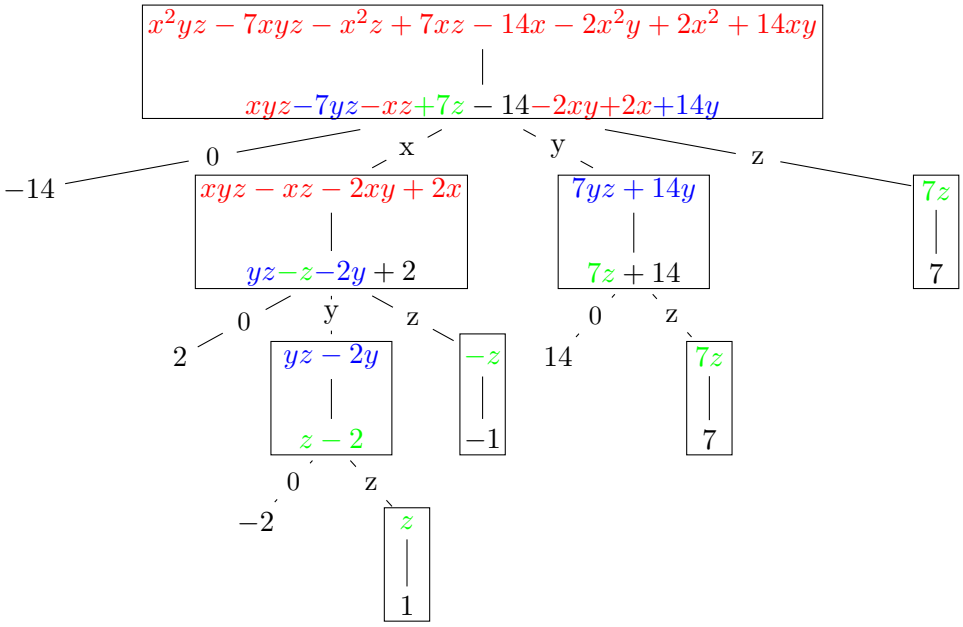

Figure 4: We recursively apply reduction and monomial grouping decomposition (Definition 4.1) to the polynomial at the root of the tree. In each box, the lower polynomial is the reduction of the upper polynomial by the unknown labeling the edge to the parent box. By 0, we denote the monomial grouping 0, and x,y, and z denote the groupings associated with unknowns.

4. Undecidability of ASOGU Unification

We now use the machinery we built in the previous section to encode *Diophantine equations* in unification problems. As a result, we can transfer undecidability results for Diophantine equations to ASOGU unification problems. Our undecidability result hinges on Lemma 3.1. Observe that for SOGU, two terms might have an equal number of occurrences of a symbol without being syntactically equal. Hence, we introduce an associative binary function symbol g to solve this issue. For the remainder of this section, we consider a finite signature Σ such that $\{g, a\} \subseteq \Sigma$, $arity(g) = 2$, $arity(a) = 0$, and $\mathcal{A} = \{g(x, g(y, z)) = g(g(x, y), z)\}$. We will write g in *flattened form* (see Section 2). Note that since our signature only consists of a single constant, strictly speaking, we only require the *weaker* property of *power associativity*.

We now introduce the basic definitions needed to describe our translation from polynomials to terms. By $p(\overline{x_n})$ we denote a polynomial in reduced form[1] with integer coefficients over the unknowns $x_1, \ldots, x_n$ ranging over the natural numbers and by $mono(p(\overline{x_n}))$ we denote the set of monomials of $p(\overline{x_n})$. Given a polynomial $p(\overline{x_n})$ and $1 \leq i \leq n$ by $div(p(\overline{x_n}), x_i)$ we denote that x_i *divides* $p(\overline{x_n})$. Furthermore, $deg(p(\overline{x}_n)) = \max\{k \mid k \geq 0 \wedge m = x_i^k \cdot q(\overline{x_n}) \wedge 1 \leq i \leq n \wedge m \in mono(p(\overline{x_n}))\}$. Given a polynomial $p(\overline{x_n})$, a polynomial $p'(\overline{x_n})$ is a sub-polynomial of $p(\overline{x_n})$ if $mono(p'(\overline{x_n})) \subseteq mono(p(\overline{x_n}))$. Using the above definition, we define distinct sub-polynomials based on divisibility by one of the input unknowns

[1] multiplication is fully distributed over addition and combining like terms.

(Definition 4.1). See Figure 4 for an illustration of the procedure defined in Definition 4.1 recursively applied to a polynomial.

Definition 4.1 (monomial groupings)**.** *Let $p(\overline{x_n}) = q(\overline{x_n}) + c$ be a polynomial where $q(\overline{x_n})$ does not have a constant term and $c \in \mathbb{Z}$, $<_{p(\overline{x_n})}$ be a total linear ordering on $\overline{x_n}$, and for all $0 \leq j \leq n$, $S_{x_j} = \{m \mid m \in mono(p(\overline{x_n})) \wedge \forall i(1 \leq i \leq n \wedge x_i <_{p(\overline{x_n})} x_j \Rightarrow \neg div(m, x_i))\}$, that is the set of monomials not divisible by any variable smaller than x_j. Then*

- $p(\overline{x_n})_0 = c$,
- $p(\overline{x_n})_{x_j} = 0$ *if there does not exist* $m \in S_{x_j}$ *such that* $div(m, x_j)$,
- *otherwise,* $p(\overline{x_n})_{x_j} = p'(\overline{x_n})$, *where* $p'(\overline{x_n})$ *is the sub-polynomial of* $p(\overline{x_n})$ *such that* $mono(p'(\overline{x_n})) = \{m \mid m \in S_{x_j} \wedge div(m, x_j)\}$.

Furthermore, let $p(\overline{x_n})_{x_j} = x_j \cdot p'(\overline{x_n})$. Then $p(\overline{x_n})_{x_j} \downarrow= p'(\overline{x_n})$ is the reduction *of $p(\overline{x_n})_{x_j}$, i.e., the degree of x_j in $p(\overline{x_n})_{x_j}$ is reduced by one.*

Essentially, monomial groupings are a way to partition a given polynomial $p(\overline{x_n})$ with respect to an ordering $<_{p(\overline{x_n})}$ on the unknowns $\overline{x_n}$. This partition results in a set of subpolynomials of $p(\overline{x_n})$ and for each subpolynomial, its monomials share a common unknown, thus implying that a reduction of these subpolynomials always exists.

Example 4.2. *Consider the polynomial*

$$\begin{aligned} p(x,y,z) = {} & x^2yz - 7xyz - x^2z + 7xz - 12z \\ & - 14x - 2x^2y + 2x^2 + 14xy + 12yz - 24y + 24 \end{aligned}$$

Assuming the unknowns are ordered $x <_{p(x,y,z)} y <_{p(x,y,z)} z$, its monomial grouping are as follows:

$$\begin{aligned} p(x,y,z)_0 &= 24 \\ p(x,y,z)_x &= x^2yz - 7xyz - x^2z + 7xz - 14x - 2x^2y + 2x^2 + 14xy \\ p(x,y,z)_y &= 12yz - 24y \\ p(x,y,z)_z &= -12z \end{aligned}$$

The reduction of these groupings is as follows:

$$\begin{aligned} p(x,y,z)_x \downarrow &= xyz - 7yz - xz + 7z - 14 - 2xy + 2x + 14y \\ p(x,y,z)_y \downarrow &= 12z - 24 \\ p(x,y,z)_z \downarrow &= -12 \end{aligned}$$

In Figure 4, we recursively apply reduction and grouping decomposition to $p(x,y,z)_x$. The illustrated procedure is at the heart of our encoding (Definition 4.3).

We now define a second-order term representation for arbitrary polynomials as follows:

Definition 4.3 (n-Converter)**.** *Let $p(\overline{x_n})$ be a polynomial and $F \in \mathcal{V}_f^n$. The positive and negative second-order term representation of $p(\overline{x_n})$ denoted as $\texttt{cvt}[F, p(\overline{x_n})]^+$ and $\texttt{cvt}[F, p(\overline{x_n})]^-$ respectively are recursively defined as follows:*

- *if $p(\overline{x_n}) = p(\overline{x_n})_0 = 0$, then*

$$\texttt{cvt}[F, p(\overline{x_n})]^+ = \texttt{cvt}[F, p(\overline{x_n})]^- = a$$

- *if* $p(\overline{x_n}) = p(\overline{x_n})_0 = c \geq 1$, *then*
  - $\texttt{cvt}[F, p(\overline{x_n})]^+ = g(\overbrace{a, \ldots, a}^{|p(\overline{x_n})_0|+1 \text{ occurences}})$
  - $\texttt{cvt}[F, p(\overline{x_n})]^- = a$.
- *if* $p(\overline{x_n}) = p(\overline{x_n})_0 < 0$, *then*
  - $\texttt{cvt}[F, p(\overline{x_n})]^+ = a$.
  - $\texttt{cvt}[F, p(\overline{x_n})]^- = g(\overbrace{a, \ldots, a}^{|p(\overline{x_n})_0|+1 \text{ occurences}})$
- *if* $p(\overline{x_n}) \neq p(\overline{x_n})_0$ *and* $p(\overline{x_n})_0 = 0$, *then for* $\star \in \{+, -\}$, $\texttt{cvt}[F, p(\overline{x_n})]^\star = F(t_1, \ldots, t_n)$
  - *where for all* $1 \leq i \leq n$, $t_i = \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^\star$.
- *if* $p(\overline{x_n}) \neq p(\overline{x_n})_0$ *and* $p(\overline{x_n})_0 \geq 1$, *then*
  - $\texttt{cvt}[F, p(\overline{x_n})]^+ = g(t, F(t_1, \ldots, t_n))$ *where*
    - $t = \overbrace{a, \ldots, a}^{|p(\overline{x_n})_0| \text{ occurences}}$ *and* [2]
    - *for all* $1 \leq i \leq n$, $t_i = \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^+$.
  - $\texttt{cvt}[F, p(\overline{x_n})]^- = F(t_1, \ldots, t_n)$ *where*
    - *for all* $1 \leq i \leq n$, $t_i = \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^-$.
- *if* $p(\overline{x_n}) \neq p(\overline{x_n})_0$ *and* $p(\overline{x_n})_0 < 0$, *then*
  - $\texttt{cvt}[F, p(\overline{x_n})]^+ = F(t_1, \ldots, t_n)$ *where*
    - *for all* $1 \leq i \leq n$, $t_i = \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^+$.
  - $\texttt{cvt}[F, p(\overline{x_n})]^- = g(t, F(t_1, \ldots, t_n))$ *where*
    - $t = \overbrace{a, \ldots, a}^{|p(\overline{x_n})_0| \text{ occurences}}$ *and*
    - *for all* $1 \leq i \leq n$, $t_i = \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^-$.

Intuitively, the n-converter takes a polynomial in n unknowns and separates it into $n+1$ subpolynomials disjoint with respect to monomial groupings. Each of these subpolynomials is assigned to one of the arguments of the second-order variable (except the subpolynomial representing an integer constant), and the n-converter is called recursively on these sub-polynomials. The process stops when all the sub-polynomials are integers. This procedure is terminating as polynomials have a maximum degree. Example 4.4 illustrates the construction of a term from a polynomial. Example 4.5 & 4.6 construct the n-multiplier and n-counter of the resulting term, respectively.

Example 4.4. *Consider the polynomial* $p(x, y) = 3 \cdot x^3 + xy - 2 \cdot y^2 - 2$. *The positive and negative terms representing this polynomial are as follows (See Figure 5 for a tree representation):*

Positive* n*-Converter

[2] Remember that nested associative functions are written in flattened form.

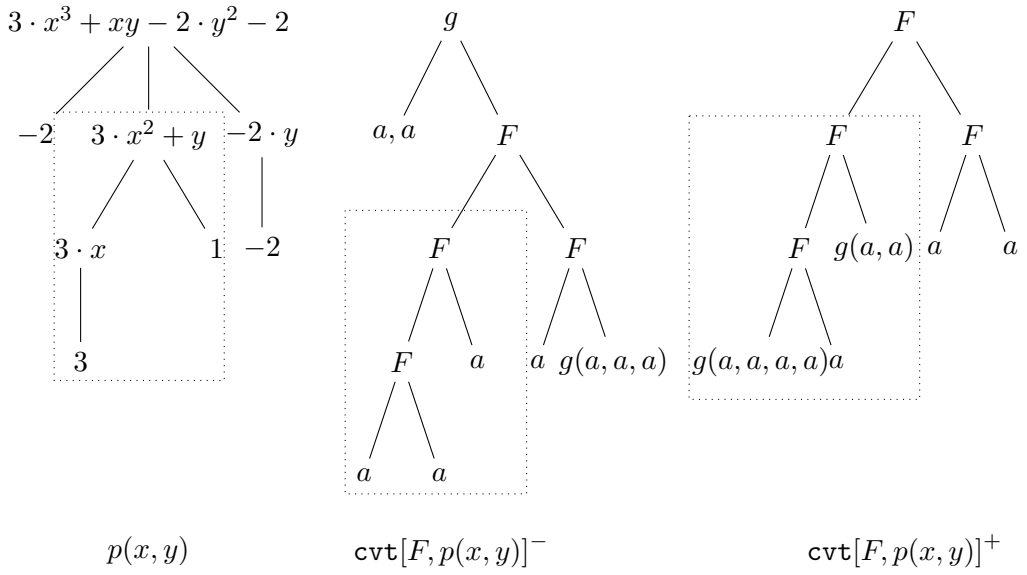

Figure 5: Applying Definition 4.3 to the polynomial of Example 4.4, we derive $\texttt{cvt}[F, p(x, y)]^-$ and $\texttt{cvt}[F, p(x, y)]^+$. The boxed section of the polynomial tree results in the boxed sections of the two term trees. The precise construction is described in Example 4.4.

$$
\begin{aligned}
\texttt{cvt}[F, 3 \cdot x^3 + xy - 2 \cdot y^2 - 2]^+ &= F(\texttt{cvt}[F, 3 \cdot x^2 + y]^+, \texttt{cvt}[F, -2 \cdot y]^+) \\
\texttt{cvt}[F, 3 \cdot x^2 + y]^+ &= F(\texttt{cvt}[F, 3 \cdot x]^+, \texttt{cvt}[F, 1]^+) \\
\texttt{cvt}[F, 3 \cdot x]^+ &= F(\texttt{cvt}[F, 3]^+, a) \\
\texttt{cvt}[F, 3]^+ &= g(a, a, a, a) \\
\texttt{cvt}[F, 1]^+ &= g(a, a) \\
\texttt{cvt}[F, -2 \cdot y]^+ &= F(a, a)
\end{aligned}
$$

Negative n-Converter

$$
\begin{aligned}
\texttt{cvt}[F, 3 \cdot x^3 + xy - 2 \cdot y^2 - 2]^- &= g(a, a, F(\texttt{cvt}[F, 3 \cdot x^2 + y]^-, \texttt{cvt}[F, -2 \cdot y]^-)) \\
\texttt{cvt}[F, 3 \cdot x^2 + y]^- &= F(\texttt{cvt}[F, 3 \cdot x]^-, \texttt{cvt}[F, 1]^-) \\
\texttt{cvt}[F, 3 \cdot x]^- &= F(a, a) \\
\texttt{cvt}[F, 1]^- &= a \\
\texttt{cvt}[F, -2 \cdot y]^- &= F(a, g(a, a, a))
\end{aligned}
$$

The result is the following two terms:

$$
\begin{aligned}
\texttt{cvt}[F, 3 \cdot x^3 + xy - 2 \cdot y^2 - 2]^+ &= F(F(F(g(a, a, a, a), a), g(a, a)), F(a, a)) \\
\texttt{cvt}[F, 3 \cdot x^3 + xy - 2 \cdot y^2 - 2]^- &= g(a, a, F(F(F(a, a), a), F(a, g(a, a, a))))
\end{aligned}
$$

Example 4.5. *Consider the terms constructed in Example 4.4, that is, let* $s = \mathtt{cvt}[F, 3 \cdot x^3 + xy - 2 \cdot y^2 - 2]^+$ *and* $t = \mathtt{cvt}[F, 3 \cdot x^3 + xy - 2 \cdot y^2 - 2]^-$*. The* n*-multiplier is computed as follows:*

Positive n***-multiplier***

$$\begin{aligned}
\mathtt{mul}[F, x, y, s] &= 1 + x \cdot \mathtt{mul}[F, x, y, \mathtt{cvt}[F, 3 \cdot x^2 + y]^+] + \\
&\quad\ y \cdot \mathtt{mul}[F, x, y, \mathtt{cvt}[F, -2 \cdot y]^+] \\
\mathtt{mul}[F, x, y, \mathtt{cvt}[F, 3 \cdot x^2 + y]^+] &= 1 + x \cdot \mathtt{mul}[F, x, y, \mathtt{cvt}[F, 3 \cdot x]^+] \\
\mathtt{mul}[F, x, y, \mathtt{cvt}[F, 3 \cdot x]^+] &= 1 \\
\mathtt{mul}[F, x, y, \mathtt{cvt}[F, -2 \cdot y]^+] &= 1
\end{aligned}$$

Negative n***-multiplier***

$$\begin{aligned}
\mathtt{mul}[F, x, y, t] &= 1 + x \cdot \mathtt{mul}[F, x, y, \cdot \mathtt{cvt}[F, 3 \cdot x^2 + y]^-] + \\
&\quad\ y \cdot \mathtt{mul}[F, x, y, \mathtt{cvt}[F, -2 \cdot y]^-] \\
\mathtt{mul}[F, x, y, \mathtt{cvt}[F, 3 \cdot x^2 + y]^-] &= 1 + x \cdot \mathtt{mul}[F, x, y, \mathtt{cvt}[F, 3 \cdot x]^-] \\
\mathtt{mul}[F, x, y, \mathtt{cvt}[F, 3 \cdot x]^-] &= 1 \\
\mathtt{mul}[F, x, y, \mathtt{cvt}[F, -2 \cdot y]^-] &= 1
\end{aligned}$$

Thus, $\mathtt{mul}[F, x, y, s] = \mathtt{mul}[F, x, y, t] = 1 + x + x^2 + y$

Example 4.6. *Consider the terms constructed in Example 4.4. The* n*-counter is computed as follows:*

Positive n***-counter***

$$\begin{aligned}
\mathtt{cnt}[F, x, y, a, s] &= x \cdot \mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, 3 \cdot x^2 + y]^+] + \\
&\quad\ y \cdot \mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, -2 \cdot y]^+] \\
\mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, 3 \cdot x^2 + y]^+] &= x \cdot \mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, 3 \cdot x]^+] + 2 \cdot y \\
\mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, 3 \cdot x]^+] &= x \cdot \mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, 3]^+] + y \\
\mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, 3]^+] &= 4 \\
\mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, -2 \cdot y]^+] &= x + y
\end{aligned}$$

Negative n***-counter***

$$\begin{aligned}
\mathtt{cnt}[F, x, y, a, t] &= 2 + x \cdot \mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, 3 \cdot x^2 + y]^-] + \\
&\quad\ y \cdot \mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, -2 \cdot y]^-] \\
\mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, 3 \cdot x^2 + y]^-] &= x \cdot \mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, 3 \cdot x]^-] + \\
&\quad\ y \\
\mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, 3 \cdot x]^-] &= x + y \\
\mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, -2 \cdot y]^-] &= x + y \cdot \mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, -2]^-] \\
\mathtt{cnt}[F, x, y, a, \mathtt{cvt}[F, -2]^-] &= 3
\end{aligned}$$

Thus,

$$\begin{aligned} p(x,y) &= \texttt{cnt}[F,x,y,a,s] = 4\cdot x^3 + x^2y + 3\cdot xy + y^2 \\ q(x,y) &= \texttt{cnt}[F,x,y,a,t] = x^3 + x^2y + 2\cdot xy + 3\cdot y^2 + 2 \\ p(x,y) - q(x,y) &= 3x^3 + xy - 2\cdot y^2 - 2 \end{aligned}$$

Using the operator introduced in Definition 4.3, we can transform a polynomial with integer coefficients into an ASOGU problem. The next definition describes the process:

Definition 4.7. *Let $p(\overline{x_n})$ be a polynomial and $F \in \mathcal{V}_f^n$. Then $(\mathcal{U}, \mathcal{A}, F)$ is the ASOGU problem induced by $p(\overline{x_n})$ where $\mathcal{U} = \{\texttt{cvt}[F, p(\overline{x_n})]^- \stackrel{?}{=}_F \texttt{cvt}[F, p(\overline{x_n})]^+\}$.*

Example 4.8. *For the polynomial presented in Example 4.4, we can build the unification problem $\mathcal{U} = \{s \stackrel{?}{=} t\}$ where*

$$\begin{aligned} s &= F\Big(F\big(F(g(a,a,a,a),a), g(a,a)\big), F(a,a)\Big) \\ t &= g\Big(a, a, F\big(F(F(a,a),a), F(a,g(a,a,a))\big)\Big) \end{aligned}$$

Observe that the n-converter will always produce a flex-rigid unification equation as long as the input polynomial is of the form $p(\overline{x_n}) = p'(\overline{x_n}) + c$ where $c \neq 0$. When $c = 0$, we get a flex-flex unification equation, and there is always a solution for both the polynomial and the unification equation.

The result of this translation is that the n-counter captures the structure of the polynomial, and the n-multipliers cancel out.

Theorem 4.9. *Let $n \geq 1$, $p(\overline{x_n})$ be a polynomial, and $(\mathcal{U}, \mathcal{A}, F)$ an ASOGU problem induced by $p(\overline{x_n})$ where $\mathcal{U} = \{\texttt{cvt}[F, p(\overline{x_n})]^- \stackrel{?}{=}_F \texttt{cvt}[F, p(\overline{x_n})]^+\}$. Then*

$$p(\overline{x_n}) = \texttt{cnt}[F, \overline{x_n}, a, \mathcal{U}]_r - \texttt{cnt}[F, \overline{x_n}, a, \mathcal{U}]_l \quad \text{and} \quad 0 = \texttt{mul}[F, \overline{x_n}, \mathcal{U}]_l - \texttt{mul}[F, \overline{x_n}, \mathcal{U}]_r.$$

Proof. We proceed by induction on $deg(p(\overline{x_n}))$.
Base case: When $deg(p(\overline{x_n})) = 0$, then $p(\overline{x_n}) = c$ for $c \in \mathbb{Z}$. We have the following cases:

- if $c = 0$, then $\mathcal{U} = \{a \stackrel{?}{=}_F a\}$. This implies that $\texttt{cnt}[F, \overline{x_n}, a, a] = \texttt{cnt}[F, \overline{x_n}, a, a] = 1$ and $\texttt{mul}[F, \overline{x_n}, a] = \texttt{mul}[F, \overline{x_n}, a] = 0$, i.e., the counter and multiplier of the left and right side are equivalent,
- if $c > 0$, then $\mathcal{U} = \{s \stackrel{?}{=}_F t\}$ where $occ_\Sigma(a, t) = c + 1$ and $occ_\Sigma(a, s) = 1$. This implies that $\texttt{cnt}[F, \overline{x_n}, a, t] - \texttt{cnt}[F, \overline{x_n}, a, s] = c$ and $\texttt{mul}[F, \overline{x_n}, s] = \texttt{mul}[F, \overline{x_n}, t] = 0$, i.e., the counter of the left and right side cancel out to c and the multipliers are equivalent.
- if $c < 0$, then $\mathcal{U} = \{s \stackrel{?}{=}_F t\}$ where $occ_\Sigma(a, t) = 1$ and $occ_\Sigma(a, s) = c + 1$. This implies that $\texttt{cnt}[F, \overline{x_n}, a, t] - \texttt{cnt}[F, \overline{x_n}, a, s] = c$ and $\texttt{mul}[F, \overline{x_n}, s] = \texttt{mul}[F, \overline{x_n}, t] = 0$, i.e., the counter of the left and right side cancel out to c and the multipliers are equivalent.

These arguments complete the base case.
Step case: Let $\mathcal{U} = \{s \stackrel{?}{=}_F t\}$. We assume for our induction hypothesis that for all polynomials $p(\overline{x_n})$ with $deg(p(\overline{x_n})) \leq k$ the statement holds, and show the statement holds for polynomials $p(\overline{x_n})$ with $deg(p(\overline{x_n})) = k + 1$. observe that

$$\texttt{cnt}[F, \overline{x_n}, a, t] = \texttt{cnt}[F, \overline{x_n}, a, \texttt{cvt}[F, p(\overline{x_n})]^+] = \texttt{cnt}[F, \overline{x_n}, a, \texttt{cvt}[F, p(\overline{x_n})_0]^+] +$$

$$\sum_{i=1}^{n} x_i \cdot \texttt{cnt}[F, \overline{x_n}, a, \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^+]$$

$$\texttt{cnt}[F, \overline{x_n}, a, s] = \texttt{cnt}[F, \overline{x_n}, a, \texttt{cvt}[F, p(\overline{x_n})]^-]_l = \texttt{cnt}[F, \overline{x_n}, a, \texttt{cvt}[F, p(\overline{x_n})_0]^-] +$$

$$\sum_{i=1}^{n} x_i \cdot \texttt{cnt}[F, \overline{x_n}, a, \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^-]$$

Observe that a is an arity zero constant, thus we do not increment the sum during the recursion (See Definition 3.4, bullets 3 and 4). Furthermore, when $p(\overline{x_n})_0 = 0$, both $\texttt{cnt}[F, \overline{x_n}, a, \texttt{cvt}[F, p(\overline{x_n})_0]^+]$ and $\texttt{cnt}[F, \overline{x_n}, a, \texttt{cvt}[F, p(\overline{x_n})_0]^-]$ are spuriously equivalent to 1 rather then 0. However, the terms cancel and do not influence the results). By the induction hypothesis, we know that for all $0 \leq i \leq n$,

$$p(\overline{x_n})_{x_i} = \texttt{cnt}[F, \overline{x_n}, a, \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^+] - \texttt{cnt}[F, \overline{x_n}, a, \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^-]$$

Thus, we can derive the following:

$$\texttt{cnt}[F, \overline{x_n}, a, t] - \texttt{cnt}[F, \overline{x_n}, a, s] = p(\overline{x_n})_0 + \sum_{i=1}^{n} x_i \cdot p(\overline{x_n})_{x_i} = p(\overline{x_n})$$

Similarly,

$$\texttt{mul}[F, \overline{x_n}, t] = \texttt{mul}[F, \overline{x_n}, \texttt{cvt}[F, p(\overline{x_n})]^+] = \texttt{mul}[F, \overline{x_n}, \texttt{cvt}[F, p(\overline{x_n})_0]^+] +$$

$$\sum_{i=1}^{n} x_i \cdot \texttt{mul}[F, \overline{x_n}, \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^+]$$

$$\texttt{mul}[F, \overline{x_n}, s] = \texttt{mul}[F, \overline{x_n}, \texttt{cvt}[F, p(\overline{x_n})]^-] = \texttt{mul}[F, \overline{x_n}, \texttt{cvt}[F, p(\overline{x_n})_0]^-] +$$

$$\sum_{i=1}^{n} x_i \cdot \texttt{mul}[F, \overline{x_n}, \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^-]$$

By the induction hypothesis, we know that for all $0 \leq i \leq n$,

$$0 = \texttt{mul}[F, \overline{x_n}, \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^+] - \texttt{mul}[F, \overline{x_n}, \texttt{cvt}[F, p(\overline{x_n})_{x_i} \downarrow]^-]$$

Thus, we can derive the following:

$$\texttt{mul}[F, \overline{x_n}, t] - \texttt{mul}[F, \overline{x_n}, s] = 0 + \sum_{i=1}^{n} 0 \cdot x_i = 0$$

□

A simple corollary of Theorem 4.9 concerns commutativity of unification equations:

Corollary 4.10. *Let $n \geq 1$, $p(\overline{x_n})$ be a polynomial, and $(\{s \stackrel{?}{=}_F t\}, \mathcal{A}, F)$ an ASOGU problem induced by $p(\overline{x_n})$. Then $-p(\overline{x_n}) = \texttt{cnt}[F, \overline{x_n}, a, s] - \texttt{cnt}[F, \overline{x_n}, a, t]$.*

Proof. Same as Theorem 4.9 but swapping terms. □

Both $p(\overline{x_n})$ and $-p(\overline{x_n})$ have the same roots, and the induced unification problem is flex-rigid (unless the polynomial is a constant); thus, the induced unification problem uniquely captures the polynomial $p(\overline{x_n})$.

We now prove that the unification condition introduced in Lemma 3.8 is equivalent to finding the solutions to polynomial equations. The following shows how a solution to a polynomial can be obtained from the unification condition and vice versa.

Lemma 4.11. *Let $p(\overline{x_n})$ be a polynomial, $s = \mathtt{cvt}[F, p(\overline{x_n})]^+$, $t = \mathtt{cvt}[F, p(\overline{x_n})]^-$ and $(\{s \stackrel{?}{=}_F t\}, \mathcal{A}, F)$ the ASOGU problem induced by $p(\overline{x_n})$ (Definition 4.7). Then there exists non-negative integers $h_1, \ldots, h_n \geq 0$ such that*

$$\sigma = \left\{ F \mapsto \lambda \overline{x_n}.g(\overbrace{x_1, \ldots, x_1}^{h_1}, \ldots, \overbrace{x_n, \ldots, x_n}^{h_n}) \right\} \tag{4.1}$$

is a unifier of $\{s \stackrel{?}{=}_F t\}$ if and only if $\{x_i \mapsto h_i \mid 1 \leq i \leq n\}$ is a solution to $p(\overline{x_n}) = 0$.

Proof. Observe, by Theorem 4.9, $\mathtt{mul}[F, \overline{h_n}, s] - \mathtt{mul}[F, \overline{h_n}, t] = 0$, thus we can ignore the n-multiplier. We can prove the two directions as follows:
$\Longrightarrow$: If σ unifies $\{s \stackrel{?}{=}_F t\}$, then $\mathtt{cnt}[F, \overline{h_n}, a, s] = \mathtt{cnt}[F, \overline{h_n}, a, t]$ and by Lemma 3.8, $0 = \mathtt{cnt}[F, \overline{h_n}, a, t] - \mathtt{cnt}[F, \overline{h_n}, a, s]$. By Theorem 4.9, we know that $p(\overline{h_n}) = \mathtt{cnt}[F, \overline{h_n}, a, t] - \mathtt{cnt}[F, \overline{h_n}, a, s]$. Thus, we derive $p(\overline{h_n}) = 0$ by transitivity.
$\Longleftarrow$: If $\{x_i \mapsto h_i \mid 1 \leq i \leq n\}$ is a solution to $p(\overline{x_n}) = 0$, we can derive, using Theorem 4.9, that $0 = \mathtt{cnt}[F, \overline{h_n}, a, t] - \mathtt{cnt}[F, \overline{h_n}, a, s]$. Furthermore, we derive that $\mathtt{cnt}[F, \overline{h_n}, a, s] = \mathtt{cnt}[F, \overline{h_n}, a, t]$. Now, let σ be the substitution defined in Equation 4.1. It then follows from the definition of the n-counter (Definition 3.4), that $occ_\Sigma(a, s\sigma) = occ_\Sigma(a, t\sigma)$. Given that g is the only other symbol occurring in $s\sigma$ and $t\sigma$ and g is associative, it follows that $s\sigma \approx_{\mathcal{A}} t\sigma$. Thus, σ is a unifier of $\{s \stackrel{?}{=}_F t\}$. □

We have proven that there is a reduction from Hilbert's 10th problem over the non-negative integers to ASOGU. We can now state the main result of this paper.

Theorem 4.12. *There exists $n \geq 1$ such that n-ASOGU is undecidable.*

Proof. The statement follows from the reduction presented in Lemma 4.11 and Theorem 2.2. □

The following example illustrates this encoding.

Example 4.13. *Consider the following polynomial:*

$$p(x, y) = (x - 1)(x - 2)(y - 1)(y - 2)$$

Expansion and reduction result in the following polynomial:

$$x^2y^2 - 3x^2y - 3xy^2 + 9xy + 2x^2 + 2y^2 - 6y - 6x + 4$$

We encode this polynomial as the following unification problem where $\boldsymbol{F}$ is a second-order variable, a is a constant, g is an associative binary function symbol:

$$\boldsymbol{F}\Big(g\Big(a^6, \boldsymbol{F}\Big(\boldsymbol{F}(a, g(a^3, \boldsymbol{F}(a, a))\Big), \boldsymbol{F}(a, g(a^4)))\Big), g(a^6, \boldsymbol{F}(a, a))\Big)$$
$$\stackrel{?}{=}$$
$$g\Big(a^4, \boldsymbol{F}\Big(\boldsymbol{F}\Big(g(a^2, \boldsymbol{F}(a, \boldsymbol{F}(a, g(a^2)))), g(a^9, \boldsymbol{F}(a, a))\Big), \boldsymbol{F}(a, g(a^3))\Big)\Big)$$

where a^n abbreviates a sequence of n occurrences of a, and terms are written in flattened form, i.e., intermediary occurrences of associative function symbols are dropped. Observe that the arity of $\boldsymbol{F}$ is equivalent to the number of unknowns in $p(x, y)$. We can derive solutions for this unification problem from the zeros of $p(x, y)$:

- $p(1,0) = 0$. *The substitution* $\{F \mapsto \lambda x, y.x\}$ *unifies the above terms resulting in* $g(a^7) \stackrel{?}{=} g(a^7)$.
- $p(0,2) = 0$. *The substitution* $\{F \mapsto \lambda x, y.g(y,y)\}$ *unifies the above terms resulting in* $g(a^{16}) \stackrel{?}{=} g(a^{16})$.
- $p(1,2) = 0$. *The substitution* $\{F \mapsto \lambda x, y.g(x,y,y)\}$ *unifies the above terms resulting in* $g(a^{55}) \stackrel{?}{=} g(a^{55})$.
- $p(2,1) = 0$. *The substitution* $\{F \mapsto \lambda x, y.g(x,x,y)\}$ *unifies the above terms resulting in* $g(a^{65}) \stackrel{?}{=} g(a^{65})$.

As described above, each unknown is associated with an argument of $\boldsymbol{F}$.

Theorem 4.12 partially answers the question posed in Section 1 by demonstrating that occurrences of first-order variables within the unification problem do not impact the decidability of second-order unification over an associative theory. Furthermore, this holds when only *one* second-order variable is present. However, it remains open whether SOGU is decidable over the empty theory. Compared to the work of Otto, Narendran, and Dougherty [OND98], we show that the undecidability of second-order E-unification is not contingent on the equational theory being a length-reducing rewrite system. As a final point, it is known that the construction used to demonstrate the undecidability of Diophantine equation solving over $\mathbb{N}$ requires a polynomial with at least 9 unknowns [Mat93]. This result extends to our work, implying that the arity of the function variable F needs to be at least 9. Thus, the decidability of ASOGU remains open for function variables of lower arity.

As mentioned earlier, Theorem 4.12 holds when associativity is replaced with *power associativity*. Furthermore, it is possible to adjust Definition 4.3 such that Theorem 4.9 and Corollary 4.10 hold for n-SOGU. However, for the sake of clarity, we omitted these details in the presentation of our results. Observe that associativity is only required for the *only if* part of Lemma 4.11.

5. Conclusion and Future Work

We show that associative second-order ground unification is undecidable using a reduction from Hilbert's 10^{th} problem over non-negative integers. The reduction required two novel occurrence counting functions related to the *multiplicity operator* [dV85, AK10] and their relationship to the existence of a unifier. Using these operators, we show how to encode a polynomial in reduced form in a single unification equation that only contains occurrences of an associative binary function symbol and a single constant. The reduction holds even in the case when the binary function symbol is interpreted as *power associative* only, i.e., $f(x, f(x,x)) = f(f(x,x),x)$. It remains open whether second-order ground unification (the non-associative case) is decidable. We plan to investigate how our encoding can be used to discover decidable fragments of both second-order ground unification and its equational extension. Additionally, we plan to investigate variants of the presented encoding, e.g., other ways to encode polynomials in unification equations and their associated unification problems.

References

[ABD+15] Rajeev Alur, Rastislav Bodík, Eric Dallal, Dana Fisman, Pranav Garg, Garvit Juniwal, Hadas Kress-Gazit, P. Madhusudan, Milo M. K. Martin, Mukund Raghothaman, Shambwaditya Saha,

Sanjit A. Seshia, Rishabh Singh, Armando Solar-Lezama, Emina Torlak, and Abhishek Udupa. Syntax-guided synthesis. In Maximilian Irlbeck, Doron A. Peled, and Alexander Pretschner, editors, *Dependable Software Systems Engineering*, volume 40 of *NATO Science for Peace and Security Series, D: Information and Communication Security*, pages 1–25. IOS Press, 2015. doi:10.3233/978-1-61499-495-4-1.

[ADK+18] Alessandro Abate, Cristina David, Pascal Kesseli, Daniel Kroening, and Elizabeth Polgreen. Counterexample guided inductive synthesis modulo theories. In Hana Chockler and Georg Weissenbacher, editors, *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I*, volume 10981 of *Lecture Notes in Computer Science*, pages 270–288. Springer, 2018. doi:10.1007/978-3-319-96145-3_15.

[AK10] Beniamino Accattoli and Delia Kesner. The structural *lambda*-calculus. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 381–395. Springer, 2010. doi:10.1007/978-3-642-15205-4_30.

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.

[BRO+19] Haniel Barbosa, Andrew Reynolds, Daniel El Ouraoui, Cesare Tinelli, and Clark W. Barrett. Extending SMT solvers to higher-order logic. In Pascal Fontaine, editor, *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, volume 11716 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2019. doi:10.1007/978-3-030-29436-6_3.

[DJ95] Daniel J. Dougherty and Patricia Johann. A combinatory logic approach to higher-order e-unification. volume 139, pages 207–242, 1995. doi:10.1016/0304-3975(94)00210-A.

[dV85] Roel C. de Vrijer. A direct proof of the finite developments theorem. *J. Symb. Log.*, 50(2):339–343, 1985. doi:10.2307/2274219.

[Far88] William M. Farmer. A unification algorithm for second-order monadic terms. *Ann. Pure Appl. Log.*, 39(2):131–174, 1988. doi:10.1016/0168-0072(88)90015-2.

[GJV98] Harald Ganzinger, Florent Jacquemard, and Margus Veanes. Rigid reachability. In Jieh Hsiang and Atsushi Ohori, editors, *Advances in Computing Science - ASIAN '98, 4th Asian Computing Science Conference, Manila, The Philippines, December 8-10, 1998, Proceedings*, volume 1538 of *Lecture Notes in Computer Science*, pages 4–21. Springer, 1998. doi:10.1007/3-540-49366-2_2.

[Gol81] Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theor. Comput. Sci.*, 13:225–230, 1981. doi:10.1016/0304-3975(81)90040-2.

[Gou94] Jean Goubault. Higher-order rigid e-unification. In Frank Pfenning, editor, *Logic Programming and Automated Reasoning, 5th International Conference, LPAR'94, Kiev, Ukraine, July 16-22, 1994, Proceedings*, volume 822 of *Lecture Notes in Computer Science*, pages 129–143. Springer, 1994. doi:10.1007/3-540-58216-9_34.

[GPS17] Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. Program synthesis. *Found. Trends Program. Lang.*, 4(1-2):1–119, 2017. doi:10.1561/2500000010.

[Hue73] Gérard P. Huet. The undecidability of unification in third order logic. *Inf. Control.*, 22(3):257–267, 1973. doi:10.1016/S0019-9958(73)90301-X.

[Jez14] Artur Jez. Context unification is in PSPACE. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 244–255. Springer, 2014. doi:10.1007/978-3-662-43951-7_21.

[Lev96] Jordi Levy. Linear second-order unification. In Harald Ganzinger, editor, *Rewriting Techniques and Applications, 7th International Conference, RTA-96, New Brunswick, NJ, USA, July 27-30, 1996, Proceedings*, volume 1103 of *Lecture Notes in Computer Science*, pages 332–346. Springer, 1996. doi:10.1007/3-540-61464-8_63.

[Lev14] Jordi Levy. On the limits of second-order unification. In Temur Kutsia and Christophe Ringeissen, editors, *Proceedings of the 28th International Workshop on Unification, UNIF 2014, Vienna, Austria, July 13, 2014*, pages 5–14, 2014. URL: https://www3.risc.jku.at/publications/download/risc_5001/proceedings-UNIF2014.pdf#page=10.

[LSV08] Jordi Levy, Manfred Schmidt-Schauß, and Mateu Villaret. The complexity of monadic second-order unification. *SIAM J. Comput.*, 38(3):1113–1140, 2008. doi:10.1137/050645403.

[Luc72] Claudio L Lucchesi. The undecidability of the unification problem for third order languages. *Report CSRR*, 2059:129–198, 1972.

[LV00] Jordi Levy and Margus Veanes. On the undecidability of second-order unification. *Inf. Comput.*, 159(1-2):125–150, 2000. doi:10.1006/inco.2000.2877.

[Mat93] Yuri V. Matiyasevich. *Hilbert's tenth problem*. MIT Press, Cambridge, MA, USA, 1993.

[NQ91] Tobias Nipkow and Zhenyu Qian. Modular higher-order E-unification. In Ronald V. Book, editor, *Rewriting Techniques and Applications, 4th International Conference, RTA-91, Como, Italy, April 10-12, 1991, Proceedings*, volume 488 of *Lecture Notes in Computer Science*, pages 200–214. Springer, 1991. doi:10.1007/3-540-53904-2_97.

[OND98] Friedrich Otto, Paliath Narendran, and Daniel J. Dougherty. Equational unification, word unification, and 2nd-order equational unification. *Theoretical Computer Science*, 198(1):1–47, 1998. URL: https://www.sciencedirect.com/science/article/pii/S0304397597001308, doi:10.1016/S0304-3975(97)00130-8.

[PBJK23] Julian Parsert, Chad E. Brown, Mikolas Janota, and Cezary Kaliszyk. Experiments on infinite model finding in SMT solving. In Ruzica Piskac and Andrei Voronkov, editors, *LPAR 2023: Proceedings of 24th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Manizales, Colombia, 4-9th June 2023*, volume 94 of *EPiC Series in Computing*, pages 317–328. EasyChair, 2023. URL: https://doi.org/10.29007/slrm, doi:10.29007/SLRM.

[Sch04] Manfred Schmidt-Schauß. Decidability of bounded second order unification. *Inf. Comput.*, 188(2):143–178, 2004. URL: https://doi.org/10.1016/j.ic.2003.08.002, doi:10.1016/J.IC.2003.08.002.

[Sny90] Wayne Snyder. Higher order e-unification. In Mark E. Stickel, editor, *10th International Conference on Automated Deduction, Kaiserslautern, FRG, July 24-27, 1990, Proceedings*, volume 449 of *Lecture Notes in Computer Science*, pages 573–587. Springer, 1990. doi:10.1007/3-540-52885-7_115.

[Tou24] Sophie Tourret. Invited talk: The hows and whys of higher-order SMT. In Giles Reger and Yoni Zohar, editors, *Proceedings of the 22nd International Workshop on Satisfiability Modulo Theories co-located with the 36th International Conference on Computer Aided Verification (CAV 2024), Montreal, Canada, July, 22-23, 2024*, volume 3725 of *CEUR Workshop Proceedings*, page 1. CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3725/invited1.pdf.