Idempotent Anti-Unification

David Cerna

RISC, Johannes Kepler University Linz, Austria
Institute for Formal Models and Verification, Johannes Kepler University Linz, Austria

Temur Kutsia

RISC, Johannes Kepler University Linz, Austria

Abstract

In this paper we address two problems related to idempotent anti-unification. First, we show that there exists an anti-unification problem with a single idempotent symbol which has an infinite minimal complete set of generalizations. It means that anti-unification with a single idempotent symbol has infinitary or nullary generalization type, similar to anti-unification with two idempotent symbols, shown earlier by Loïc Pottier. Next, we develop an algorithm, which takes an arbitrary idempotent anti-unification problem and computes a representation of its solution set in the form of a regular tree grammar. The algorithm does not depend on the number of idempotent function symbols in the input terms. The language generated by the grammar is the minimal complete set of generalizations of the given anti-unification problem, which implies that idempotent anti-unification is infinitary.

Keywords: Anti-unification, generalization, idempotence, regular tree grammar.

1. Introduction

The equational theory of idempotence is defined by the axiom $f(x, x) \approx x$, stating that the function symbol f is idempotent. Many interesting algebraic structures have this property. It is an example of a regular collapse theory (Siekmann, 1989), which means that the variable sets of both sides of the defining axiom(s) are the same (the regularity property), and it contains an axiom of the form $t \approx x$, where t is a non-variable term and x is a variable (the collapse property). In fact, it is the simplest such theory. It makes idempotence interesting from the unification theory point of view. There are several works that studied unification modulo idempotence alone (Raulefs and Siekmann, 1978; Kühner et al., 1977; Herold, 1987; Siekmann, 1978; Hullot, 1980) or in combination with some other properties, see, e.g., (Livesey et al., 1979; Baader, 1986; Baader and Büttner, 1988; Schmidt-Schauß, 1986; Schmidt-Schauß, 1986; Siekmann, 1978; Livesey and Siekmann, 1976). They show that in the unification hierarchy (Siekmann, 1978; Baader

Email addresses: David.Cerna@risc.jku.at (David Cerna), kutsia@risc.jku.at (Temur Kutsia)
URL: https://risc.jku.at/m/david-cerna/ (David Cerna), https://www.risc.jku.at/people/tkutsia/ (Temur Kutsia)

and Snyder, 2001), idempotent unification is finitary (i.e., the minimal complete set of unifiers always exists and is finite), it remains finitary if idempotence is combined with commutativity, but becomes of type zero in combination with associativity. Type zero means that for some unification problems the minimal complete set of unifiers does not exist: minimality conflicts with completeness. However, if we put all three properties together, associativity, commutativity, and idempotence, then it is again finitary.

Anti-unification with idempotent functions has been studied with far less intensity. The work of Pottier (1989) shows that in a theory with two idempotent symbols, there exists an anti-unification problem with infinitely many incomparable generalizations. Anti-unification in description logic $\mathcal{EL}$ (Konev and Kutsia, 2016) uses a combination of idempotence with associativity and commutativity, plus some other special properties.

In general, unification and anti-unification types do not correlate. For instance, associative unification is infinitary (minimal complete set of unifiers always exists and for some problems is it infinite) (Plotkin, 1972; Siekmann, 1975), but associative anti-unification is finitary (Alpuente et al., 2014). Nominal unification is unitary (Urban et al., 2004), but nominal anti-unification (with unrestricted number of atoms, as in nominal unification) is of type zero (Baumgartner, 2015; Baumgartner et al., 2015). There are similarities as well: both syntactic unification (Robinson, 1965) and anti-unification (Plotkin, 1970; Reynolds, 1970) are unitary. The same is true for higher-order patterns (Miller, 1991; Baumgartner et al., 2017, 2013). For commutative theories, both unification and anti-unification are finitary (Alpuente et al., 2014).

In this context, it is interesting to see how idempotent anti-unification behaves. Pottier's result implies that anti-unification with two idempotent symbols is at least infinitary, i.e., it is either infinitary or of type zero. But which one is actually the case? Moreover, what happens if we allow any number of idempotent symbols?

These were the questions that motivated our work, and we showed that, in fact, two idempotent symbols are not necessary to have a problem with infinitely many least general generalizations: one suffices. Moreover, we showed that idempotent anti-unification is infinitary, not of type zero: the minimal complete set of generalizations always exists. This is true for any number of idempotent symbols permitted in the theory. It gives yet another dichotomy between unification and anti-unification: this time, finitary vs infinitary. Moreover, we have constructed an algorithm, which computes a finite representation of such sets in the form of regular tree grammar, for any number of idempotent symbols in the input.

The results of this paper contribute into the understanding anti-unification in one of the most basic, yet nontrivial equational theories. In general, equational anti-unification problem is studied less intensively than its unification counterpart. (We review some of the work related to this subject below in a separate section.) Besides the theoretical importance, our results have a practical merit as well. Most importantly, showing that idempotent anti-unification is (at least) infinitary for one idempotent function symbol is encouraging from the perspective of the development of a combination method for anti-unification algorithms (in the style of such results for unification in the union of signature-disjoint theories (Baader and Schulz, 1996)). If the theory with one idempotent symbol had the finitary generalization type, it would mean that finitary anti-unification algorithms can not be combined, since from (Pottier, 1989) it is known that anti-unification with two idempotent symbols is infinitary or of type zero. Also, the idea of representing generalization sets by regular tree grammars (which resembles to the idea of E-generalizations using grammars from (Burghardt, 2005)), on which we elaborate in this paper, might be worth to extend to other theories as well, aiming at representing preferably minimal, complete sets of generalizations compactly and developing terminating algorithms to compute them.

From the applicability perspective, idempotent generalization, in a restricted form or in combination with the other equational properties, can be potentially useful for proof transformation methods such as schematic cut elimination (Leitsch et al., 2017), cut introduction (Hetzl et al., 2014, 2012) and induction theorem proving based on tree grammar construction and minimization (Eberhard and Hetzl, 2015).

The paper is organized as follows: In Section 2 we introduce the main notions and establish the terminology. In Section 3, we give an example of an anti-unification problem with one idempotent symbol, which has an infinite complete set of generalizations. We prove that this set is also minimal, which implies that such a theory is either infinitary or of type zero. In Section 4 an algorithm for anti-unification with arbitrary number idempotent function symbols is described. The algorithm accepts two terms and computes a regular tree grammar, for which we prove that it generates a complete set of idempotent generalizations of the input terms. In Section 5 we prove that the algorithm is also minimal, i.e., the computed grammar generates a minimal complete set of generalizations, which implies that anti-unification with arbitrary number of idempotent function symbols is infinitary. Section 6 concludes.

1.1. Related Work

As we have already mentioned, equational anti-unification has not been studied as intensively as equational unification. Pottier (1989) gave not only an example of idempotent anti-unification with an infinite set of solutions, but also an algorithm for anti-unification with one associative-commutative symbol. Baader (1991) studied the effect which different instantiation preorders have on E-generalization problems, and addressed generalization in the class called commutative equational theories. Examples of such theories are commutative monoids, commutative idempotent monoids, and abelian groups. Generalization in commutative theories is unitary.

Biere (1993) studied word anti-unification (anti-unification in free monoids) together with its special case, so called ϵ-free generalization. It is finitary. Generalization for sequences of unranked terms has been investigated in (Yamamoto et al., 2001) for a restricted case and in (Kutsia et al., 2014; Baumgartner and Kutsia, 2017) for the general first- and second-order cases.

Alpuente et al. (2014) described a modular rule-based generalization algorithm for order-sorted theories with associative, commutative, and unit function symbols and for their combinations, and showed that all of them the finitary generalization type. Associative, commutative, and associative-commutative anti-unification for higher-order patterns has been discussed in (Cerna and Kutsia, 2018). They are finitary as well.

The problem of generalization for clauses studied in (Plotkin, 1970) can also be seen as a special equational anti-unification problem. Considering the disjunction as an associative, commutative, idempotent (ACI) symbol, taking the empty clause as the unit element and the subclause relation for comparison, one can say that the anti-unification problem for clauses is an ACIU anti-unification problem for terms where the ACI symbol appears only in the topmost position.

Burghardt (2005) and Heinz (1995) (see also (Burghardt and Heinz, 2014)) proposed an interesting idea to use regular tree grammars to compute a finite representation of a complete set of generalizations, provided that the equational theory leads to regular congruence classes. They first construct grammars for congruence classes of the input terms, and then compute a grammar representation of the solution set. We could have applied this technique to idempotent anti-unification considered in this paper, but we were interested in obtaining a description of a *minimal* complete set of generalizations. Computing first a grammar for a complete set of generalizations, and then trying to obtain from there a description of a minimal complete set

seems to be a non-trivial task. Therefore, we developed an algorithm that directly computes the desired representation for the idempotent equational theory.

2. Preliminaries

We assume familiarity with the basic notions of unification theory, see, e.g., (Baader and Snyder, 2001).

We consider a ranked alphabet $\mathcal{A}$, consisting of the set $\mathcal{F}$ of function symbols with fixed arity and the set of variables $\mathcal{V}$. A term t over $\mathcal{A}$ is defined as $t ::= x \mid f(t_1, \ldots, t_n)$, where $x \in \mathcal{V}$ and $f \in \mathcal{F}$ with the arity $n \geq 0$. The set of terms over the alphabet $\mathcal{A}$ is denoted by $\mathcal{T}(\mathcal{A})$. Nullary function symbols are called constants. We denote variables by x, y, z, u, v, constants by a, b, c, d, function symbols f, g, h, and terms by s, t, r. We denote the set of variables appearing in a term t by $var(t)$. The depth of a term t is defined inductively as $dep(x) = 1$ for variables and $dep(f(t_1, \cdots, t_n)) = \max\{dep(t_1), \cdots, dep(t_1))\} + 1$ otherwise.

The set of *positions* of a term t, denoted by $pos(t)$, is the set of stings of positive integers, defined as $pos(x) = \{\epsilon\}$ and $pos(f(t_1, \ldots, t_n)) = \{\epsilon\} \cup \bigcup_{i=1}^{n} \{i.p \mid p \in pos(t_i)\}$, where ϵ stands for the empty string. If p is a position in a term s and t is a term, then $s|_p$ denotes the subterm of s at position p and $s[t]_p$ denotes the term obtained from s by replacing the subterm $s|_p$ with t. The *head* of a term t is defined as $head(x) = x$ and $head(f(t_1, \ldots, t_n)) = f$.

A *substitution* is a mapping from variables to terms such that all but finitely many variables are mapped to themselves. Lower case Greek letters are used to denote them, except the identity substitution, which is denoted by Id. They are extended to terms in the usual way and we use the postfix notation for that, writing $t\sigma$ for an *instance* of a term t under a substitution σ. The *composition* of substitutions σ and ϑ, written as juxtaposition $\sigma\vartheta$, is the substitution defined as $x(\sigma\vartheta) = (x\sigma)\vartheta$ for all variables x.

The *domain* of a substitution σ is the set of variables which are not mapped to themselves by σ: $dom(\sigma) := \{x \mid x\sigma \neq x\}$. The restriction of σ to a set of variables X, denoted $\sigma|_X$, is the substitution defined as $x(\sigma|_X) = x\sigma$ if $x \in X$ and $x(\sigma|_X) = x$ otherwise.

A *binding* is a pair of a variable and a term, written as $x \mapsto t$. To explicitly write substitutions, we use the standard convention representing a substitution σ as a finite set of bindings $\{x \mapsto t\sigma \mid x \in dom(\sigma)\}$. *Application* of σ to a set of bindings B, written $B\sigma$, is defined as $B\sigma = \{x \mapsto t\sigma \mid x \mapsto t \in B\}$.

An *idempotent equational theory* is generated (in the usual way) from the axiom of idempotence for some function symbol, i.e., from a equality of the form $f(x, x) \approx x$, which states that f is idempotent. We denote this equational theory by $\approx_I$, or by $\approx_{I(f,g,\ldots)}$, if we want to make explicit the idempotent symbols.

We say that a term is in *idempotent normal form* (*I-normal form*) if it does not contain a subterm of the form $f(t, t)$ for any idempotent symbol f. To get an I-normal form of a term, all the subterms of the form $f(t, t)$ are replaced by t repeatedly as long as possible, for each idempotent symbol f. We write $nf_I(s)$ for the I-normal form of s, and for a set of terms S, $nf_I(S)$ denotes the set $nf_I(S) := \{nf_I(s) \mid s \in S\}$.

A term r is *more general* than s modulo I (r is an *I-generalization* of s) if there exists a substitution σ such that $r\sigma \approx_I s$. It is written as $r \preceq_I s$. The relation $\preceq_I$ is a quasi-ordering. Its strict part is denoted by $\prec_I$, and the equivalence relation it induces by $\simeq_I$.

Given two terms t and s, we say that a term r is their *least general generalization* modulo idempotence (I-lgg or just lgg in short), if $r \preceq_I s$ and $r \preceq_I t$ hold and for all r' with the property $r' \preceq_I t$, $r' \preceq_I s$, and $r \preceq_I r'$ we have $r \simeq_I r'$.

A *minimal and complete set of idempotent generalizations* of two terms t and s is the set $\mathcal{G}$ with the following three properties:

1. Each element of $\mathcal{G}$ is an I-generalization of t and s (soundness of $\mathcal{G}$).
2. For each I-generalization r' of t and s, there exists $r \in \mathcal{G}$ such that $r' \preceq_I r$, i.e., r is less general than r' modulo I (completeness of $\mathcal{G}$).
3. If $r_1, r_2 \in \mathcal{G}$ such that $r_1 \preceq_I r_2$, then $r_1 = r_2$ (minimality of $\mathcal{G}$).

We write $mcsg_I(t, s)$ for the minimal complete set of I-generalizations of t and s. An *anti-unification problem* is a pair of terms for which an *mcsg* is to be computed.

Often we just say generalization, lgg, etc. instead of I-generalization, I-lgg and so on.

Anti-unification type of equational theories are defined similarly (but dually) to unification type, based on the existence and cardinality of a minimal complete set of generalizations. We assume here no restriction on the signature, i.e., the problems and generalizations may contain arbitrary function symbols. Then the types are defined as follows:

- Unary type: Any anti-unification problem in the theory has a singleton *mcsg*.
- Finitary type: Any anti-unification problem in the theory has an *mcsg* of finite cardinality, for at least one problem having it greater than 1.
- Infinitary type: For any anti-unification problem in the theory there exists an *mcsg*, and for at least one problem this set is infinite.
- Nullary type (or type zero): There exists an anti-unification problem in the theory which does not have an *mcsg*, i.e., every complete set of generalizations for this problem contains elements such that one is more general than the other.

For each of these types, there exists a corresponding instance of an equational theory. The syntactic first-order anti-unification (Plotkin, 1970; Reynolds, 1970) is unitary; commutative anti-unification (Alpuente et al., 2014) is finitary; nominal anti-unification with infinitely many atoms is nullary (Baumgartner, 2015; Baumgartner et al., 2015). In this paper we illustrate that idempotent anti-unification is infinitary. This holds for arbitrary number of idempotent function symbols involved in anti-unification problems.

We represent anti-unification problems in the form of *anti-unification triples* (AUTs). An AUT is a triple of a variable and two terms, written as $x : t \triangleq_I s$. Here x is a fresh variable which stands for the most general common I-generalization of t and s. Any I-generalization r of t and s is then an instance of x, witnessed by a substitution σ such that $x\sigma \approx_I r$.

In all the notations, we omit the subscript I when it is clear from the context.

A *regular tree grammar* is a tuple $\langle \alpha, N, T, R \rangle$, where the symbol α is called the *axiom*, N is the set of *non-terminal* symbols with arity 0 such that $\alpha \in N$, T is the set of terminal symbols with $T \cap N = \emptyset$, and R is the set of production rules of the form $\beta \mapsto t$ where $\beta \in N$ and $t \in \mathcal{T}(T \cup N)$. Given a regular tree grammar $G = \langle \alpha, N, T, R \rangle$, the *derivation relation* $\rightarrow_G$ is a relation on pairs of terms of $\mathcal{T}(T \cup N)$ such that $s \rightarrow_G t$ if and only if there exists a position p in s and a rule $\nu \rightarrow r \in R$ such that $s|_p = \nu$ and $t = s[r]_p$. The *language generated by* G is the set of terms $L(G) := \{t \mid t \in \mathcal{T}(T) \text{ and } \alpha \rightarrow_G^+ t\}$, where $\rightarrow_G^+$ is the transitive closure of the relation $\rightarrow_G$. When the grammar is clear from the context, we write $\rightarrow$ instead of $\rightarrow_G$. Given a grammar G, the set of nonterminals of G that appear in a syntactic object (term, rule, AUT, etc.) O is denoted by $nter(G, O)$. For a grammar G, the set of nonterminals that *can be reached* from a nonterminal

ν, denoted by $reach(G, \nu)$, is defined as $reach(G, \nu) := \{\mu \mid \nu \to_G^+ t \text{ and } \mu \in nter(G, t)\}$. When the grammar is clear from the context, we write $\to$ instead of $\to_G$.

Given a set U and an equivalence relation $\equiv$ on the elements of U, we denote an $\equiv$-equivalence class of $u \in U$ by $[u]$. We assume to have a function $rep_{\equiv}$ which for each $\equiv$-equivalence class gives its representative. For a set $M \subseteq U$, we denote by $Rep_{\equiv}(M)$ the set of representatives of $\equiv$-equivalence classes of elements of M: $Rep_{\equiv}(M) := \{rep_{\equiv}([m]) \mid m \in M\}$.

We extend the relation $\preceq_I$ and $\approx_I$ to bindings: $x \mapsto t \preceq_I y \mapsto s$ iff $x = y$ and $t \preceq_I s$. Then we have $x \mapsto t \approx_I y \mapsto s$ iff $x = y$ and $t \approx_I s$. Obviously, it is an equivalence relation. The strict relation $\prec_I$ is extended to bindings straightforwardly.

Given a set M of terms (resp. bindings) and the corresponding equivalence relation $\approx_I$, the function *Minimize* keeps only those representatives of $\approx_I$-equivalence classes of elements of M, which are not more general than the other representatives:

$$Minimize(M) := Rep_{\approx_I}(M) \setminus \{m \in Rep_{\approx_I}(M) \mid \text{there exists } m' \in Rep_{\approx_I}(M) \text{ such that } m \prec_I m'\}.$$

Definition 1 (Regular Tree Grammar Corresponding to a Set of Bindings)**.** Given a set of bindings B, the *corresponding regular tree grammar* $G(B) = \langle \alpha, N, T, R \rangle$ is defined by the following construction:

- α is the root of B.
- $N = \{x \mid x \mapsto r \in B \text{ for some } r\}$.
- $T = F \cup V$, where F is the set of all function symbols that appear in terms of the right hand sides of B, and $V = \{var(r) \mid x \mapsto r \in B \text{ for some } x\} \setminus N$.
- The set of rules R is defined in several steps. First, we minimize the set B by applying the *Minimize* function to it.

  In matching needed to decide $\prec_I$ for minimization, only variables from T can match. Variables from N are treated as constants. Next, we construct the base set of rules R_b from $Minimize(B)$:

  $$R_b := \{x \to r \mid x \mapsto r \in Minimize(B)\}.$$

  It gives the *base grammar* $G_b(B) = \langle \alpha, N, T, R_b \rangle$. Note that language generated by $G_b(B)$ is finite. Finally, we define R by adding to R_b some recursive duplication rules with idempotent symbols:

  $$R = R_b \cup \bigcup_{x, \exists r. x \to r \in R_b} Duplicate(x, G_b(B)),$$

  where $Duplicate(x, G_b(B))$ is defined as

  $$Duplicate(x, G_b(B)) := \{x \to f(x, x) \mid f \text{ is an idempotent symbol and } R_b \text{ contains two different rules } y \to r_1 \text{ and } y \to r_2 \text{ for } y \in reach(G_b(B), x)\}.$$

Note that *Duplicate* uses all idempotent function symbols. Usually we assume that they are those that appear in B.

3. Infinitely Many Generalizations with One Idempotent Symbol

In this section we give an example that shows that idempotent anti-unification problems might have infinitely many anti-unifiers even if the theory contains only one idempotent symbol. The investigations by Pottier (1989) provided an example of an idempotent anti-unification problem using two idempotent function symbols which has infinitely many incomparable generalizations.

His example is as follows: consider an alphabet $\Sigma = \{f, g, a, b\}$ where both f and g are binary idempotent function symbols and a and b are constants. Now we consider the following generalization problem $f(a, b) \triangleq g(a, b)$. Notice that $g(f(a, x), f(y, b))$ and $f(g(a, x), g(y, b))$ are lggs of the given problem (we will refer to them as G_1 and G_2, respectively). Using them we can develop the following recursive construction:

$$S_0 = G_1,$$
$$S_{n+1} = f(G_1, g(S_n, G_2)).$$

Notice that S_{n+1} is always incomparable to S_n. However, this is not the minimal complete set, because the construction is limited to repeated use of $\{G_1, G_2\}$. Later in this section we provide a minimal complete construction, but first we address the issue of infinitely many generalizations with one idempotent symbol by encoding Pottier's example in a simpler language.

Consider the alphabet $\Sigma = \{h, a, b\}$ where h is an idempotent function symbol and a and b are constants. Note that we can encode Pottier's example, using this signature, by encoding $f(\cdot, \cdot)$ as $h(a, h(\cdot, \cdot))$ and $g(\cdot, \cdot)$ as $h(b, h(\cdot, \cdot))$. One can think of this as an encoding of two binary functions using a ternary function and two constants. Though the precise motivation does not matter for the results which follow. Thus, the generalization problem is now $h(a, h(a, b)) \triangleq h(b, h(a, b))$. Also like the previous case there are at least two incomparable generalizations for this anti-unification problem which are incomparable to their syntactic generalization $h(x, h(a, b))$:

$$G'_1 = h(h(x, h(x, b)), h(a, h(x, b))),$$
$$G'_2 = h(h(x, h(a, x)), h(h(x, b), h(a, b))).$$

Lemma 2. *The terms G'_1 and G'_2 are lggs of $h(a, h(a, b)) \triangleq h(b, h(a, b))$.*

Proof. We prove the lemma for G'_1. For G'_2 it is similar. Let us denote $s_1 = h(a, h(a, b))$ and $s_2 = h(b, h(a, b))$.

We need to show that for any substitution σ, if $G'_1 \prec_I G'_1\sigma$, then $G'_1\sigma$ is not a generalization of both s_1 and s_2. We can assume without loss of generality that the domain of σ is $\{x\}$, i.e., it has the form $\{x \mapsto t\}$, and prove the statement by induction on the depth of t. The I-normal form of t is assumed.

The assumption $G'_1 \prec_I G'_1\sigma$ suggests that we should consider only t which is not $\simeq_I$-equivalent to a variable. Otherwise we will have $G'_1 \simeq_I G'_1\sigma$, not $G'_1 \prec_I G'_1\sigma$.

Note that our alphabet contains only h, a, and b. Any term built over these symbols and variables is $\simeq_I$-equivalent to a variable iff the term is constructed only by the idempotent symbol h and variables. For instance, $h(x_1, h(x_2, x_3)) \simeq_I x_4$, because $h(x_1, h(x_2, x_3))\{x_1 \mapsto x_4, x_2 \mapsto x_4, x_3 \mapsto x_4\} \approx_I x_4$ and $x_4\{x_4 \mapsto h(x_1, h(x_2, x_3))\} \approx_I h(x_1, h(x_2, x_3))$.

Now we proceed by induction. The base case is when the depth is 1. This can happen only if t is a, t is b, or t is a variable. In the first two cases $G'_1\sigma$ is not a generalization of both $h(a, h(a, b))$ and $h(b, h(a, b))$. The third case is excluded because t is $\simeq_I$-equivalent to a variable. Hence, the base case is proved.

As the induction hypothesis, assume that if $G'_1 \prec_I G'_1\{x \mapsto t\}$, then $G'_1\{x \mapsto t\}$ is not a generalization of both s_1 and s_2 for any t of depth n, and prove the same statement for any t' of depth $n+1$. For this, we take t' of depth $n+1$ and assume $G'_1 \prec_I G'_1\{x \mapsto t'\}$. The depth $n+1$ suggests that t' should have a form $h(t_1, t_2)$, where the depths of t_1, t_2 are at most n, and at least one of them, say t_1, has the depth necessarily n.

If t' is ground, then we should have $t' \approx_I a$ in order $G'_1\{x \mapsto t'\}$ to be a generalization of $h(a, h(a, b))$, and $t' \approx_I b$ in order $G'_1\{x \mapsto t'\}$ to be a generalization of $h(b, h(a, b))$. But $t' \approx_I a$ and $t' \approx_I b$ at the same time is not possible. Hence, $G'_1\{x \mapsto t'\}$ can not be a generalization of both s_1 and s_2 for a ground t'.

For a non-ground t', it should contain at least one a or b. Otherwise it would be $\simeq_I$-equivalent to a variable, which is forbidden by the assumption. Now, assume that there exist substitutions σ'_1 and σ'_2 such that $G'_1\{x \mapsto h(t_1, t_2)\}\sigma'_1 \approx_I h(a, h(a, b))$ and $G'_1\{x \mapsto h(t_1, t_2)\}\sigma'_2 \approx_I h(b, h(a, b))$. Then it must be the case that $h(t_1, t_2)\sigma'_1 \approx_I a$ and $h(t_1, t_2)\sigma'_2 \approx_I b$. But this would imply that $t_1\sigma'_1 \approx_I a$, $t_2\sigma'_1 \approx_I a$, $t_1\sigma'_2 \approx_I b$, $t_2\sigma'_2 \approx_I b$ and thus $G'_1\{x \mapsto t_1\}$ and $G'_1\{x \mapsto t_2\}$ would be also generalizations. However, this contradicts the induction hypothesis. Hence, we get that $G'_1\{x \mapsto t'\}$ is not a generalization of s_1 and s_2 for a non-ground t' either.

□

We can use the same recursive construction as Pottier's to produce an infinite set of incomparable generalizations. Though, once again, this construction does not result into a minimal complete set of generalizations. However, it turns out that we can come up even with a simpler example, which has infinitely many incomparable generalizations and we could even construct an $mcsg_I$. Such a generalization problem is $h(a, b) \triangleq h(b, a)$, which has at least the following two lggs, which we denote by H_1 and H_2:

$$H_1 = h(h(x, b), h(a, y)),$$
$$H_2 = h(h(x, a), h(b, y)).$$

Notice that $H_1\{x \mapsto a, y \mapsto b\} = h(a, b)$ and that $H_1\{x \mapsto b, y \mapsto a\} = h(b, a)$. Similar to the example handled by Lemma 2, to prove that H_1 is an lgg we need to show that a substitution σ with domain $\{x, y\}$, when applied to H_1, does not result in a generalization of $h(a, b) \triangleq h(b, a)$. This requires a slightly more complex induction but essentially the same proof of the base and step case would be used. The heart of the argument of Lemma 2 is the fact that substitution by a depth one term results in the terms of the initial AUP.

Now we modify Pottier's recursive construction to obtain infinite set of generalizations for $h(a, b) \triangleq h(b, a)$:

$$S_0 = \{H_1,\ H_2\}.$$
$$S_k = \{h(s_1, s_2) \mid s_1, s_2 \in S_{k-1}, s_1 \neq s_2\} \cup S_{k-1},\ k > 0.$$

For example, we have

$$S_1 = \{h(H_1, H_2),\ h(H_2, H_1)\} \cup S_0.$$
$$S_2 = \{h(H_1, h(H_1, H_2)),\ h(H_1, h(H_2, H_1)),\ h(h(H_1, H_2), H_1),\ h(h(H_2, H_1), H_1)),$$
$$h(H_2, h(H_1, H_2)),\ h(H_2, h(H_2, H_1)),\ h(h(H_1, H_2), H_2),\ h(h(H_2, H_1), H_2),$$
$$h(h(H_1, H_2), h(H_2, H_1)),\ h(h(H_2, H_1), h(H_1, H_2))\} \cup S_1.$$
$$\ldots$$

By S_∞ we denote the limit of this construction. Its elements are generalizations of $h(a,b)$ and $h(b,a)$, because for any $g \in S_\infty$ we have $g\{x \mapsto a, y \mapsto b\} \simeq_I h(a,b)$ and $g\{x \mapsto b, y \mapsto a\} \simeq_I h(b,a)$.

The theorem below shows that S_∞, in fact, is a minimal complete set of generalizations for $h(a,b) \triangleq h(b,a)$. However, enumerating the minimal complete set of generalizations for arbitrary AUPs is much more involved, as we will see in the next sections.

Theorem 3. *$S_\infty = mcsg_I(h(a,b), h(b,a))$.*

Proof. We have already shown soundness above, when we proved that S_∞ is a set of generalizations of $h(a,b)$ and $h(b,a)$.

Minimality follows from the fact that the definition of S_∞ guarantees that no two terms in the constructed set is comparable with respect to $\prec_I$. We show only completeness: Take an arbitrary generalization G of $h(a,b)$ and $h(b,a)$ and show that S_∞ contains G' such that $G \preceq_I G'$.

Let use consider an arbitrary generalization G in I-normal form of $h(a,b) \triangleq h(b,a)$ over the variables $x_1, \ldots, x_m$. The generalization G must be of the form $h(g', g'')$, since that our term signature is $\{h(\cdot,\cdot), a, b\}$. Let σ_1 and σ_2 be substitutions with domain $x_1, \ldots, x_m$ such that $h(g', g'')\sigma_1 \approx_I h(a,b)$ and $h(g', g'')\sigma_2 \approx_I h(b,a)$. Then one of the following cases must hold:

(a) $g'\sigma_1 \approx_I a$ and $g''\sigma_1 \approx_I b$ and $g'\sigma_2 \approx_I b$ and $g''\sigma_2 \approx_I a$.

(b) $g'\sigma_1 \approx_I a$ and $g''\sigma_1 \approx_I b$ and $g'\sigma_2 \approx_I g''\sigma_2 \approx_I h(b,a)$.

(c) $g'\sigma_1 \approx_I g''\sigma_1 \approx_I h(a,b)$ and $g'\sigma_2 \approx_I b$ and $g''\sigma_2 \approx_I a$.

(d) $g'\sigma_1 \approx_I g''\sigma_1 \approx_I h(a,b)$ and $g'\sigma_2 \approx_I g''\sigma_2 \approx_I h(b,a)$.

Case (a):. If we think back to Lemma 2, there are restrictions on the construction of g' and g''. Note that if g' (without loss of generality) contains an occurrence of a, then $g'\sigma_2 \not\approx_I b$ because the occurrence will remain, even after idempotent normalization. The same goes for occurrences of b. Thus, the only non-variable symbol occurring in g' and g'' would be $h(\cdot,\cdot)$. Now let us perform an induction on the term depth of g' and g''.

If both subterms have term depth 1, then neither contains an instance of $h(\cdot,\cdot)$ and thus both g' and g'' must be variables. Furthermore, they must be different variables implying that $G \simeq_I h(x_1, x_2)$. However, it is obvious that $h(x_1, x_2)$ is more general than H_1 and H_2 from S_0. Hence, we have in S_∞ elements which are less general than $h(g', g'')$.

Now let us assume, without loss of generality, that if g' has term depth at most n, g'' has an arbitrary term depth, then there exists $G' \in S_\infty$ such that $h(g', g'') \preceq_I G'$. We should show that if g' has term depth $n+1$ and g'' has an arbitrary term depth, then there exists $G' \in S_\infty$ such that $h(g', g'') \preceq_I G'$.

Let us consider a subterm of g' which has term depth 2 and thus is of the form $h(x_1, x_2)$. We can transform $h(g', g'')$ to $h(g^*, g'')$ by applying the substitution $\{x_1 \mapsto y, x_2 \mapsto y\}$ to $h(g', g'')$ and I-normalize it. This decreases the term depth at least by 1 and by the induction hypothesis there exists a $G' \in S_\infty$ such that $h(g', g'') \preceq_I h(g^*, g'') \preceq_I G'$. If $h(g^*, g'')$ is a generalization of $h(a,b)$ and $h(b,a)$, we are done. Now assume that it is not the case. But then σ_1 and σ_2 would have had to map x_1 and x_2 to different values, but this would imply that both a and b would show up in $g'\sigma_1$ contradicting our assumption for case (a).

Case (*b*)*:*. Note that case (b) with minor changes proves case (c) as well. In this case g' can contain instances of a and g'' of b. If they do not contain instances of a or b then G is trivially more general than generalizations in S_0. Also, the variables of g' are distinct from the variables of g'' by the assumptions of case (b). This can be proven by a simple argument: assume that g' and g'' share a variable y, and σ_1 maps y to a term containing a then $g''\sigma_1$ will necessarily contain a contradicting our construction. The same argument holds for a term containing b.

By the assumptions of case (b) $g'\sigma_2 \approx_I h(b, a)$. Let the notation $s[t \leftarrow z]$ stand for the term obtained from s by replacing all occurrences of t in it by the variable z, i.e., abstracting the occurrences of the subterm t in s by z. Let us instead of considering σ_2 consider a substitution θ' defined as follows:

$$x\theta' = (x\sigma_2)[b \leftarrow z'], \text{ for all } x,$$

where z' is a fresh variable. Notice that $g'\theta' \approx_I h(z', a)$ which does not violate the assumptions of case (b). The same construction can be carried out for g'' with respect to a and z'', resulting in $g''\theta'' \approx_I h(b, z'')$. Thus, $G \approx_I h(g'\theta', g''\theta'') \approx_I h(h(z', a), h(b, z'')) \in S_0$.

Case (*d*)*:*. We see that g' and g'' both are common generalizations of $h(a, b)$ and $h(b, a)$ and thus we can perform an induction over the construction of S_∞. In particular, we will prove by induction that for each n, if the depth of G is at most $n + 3$, then the set S_n contains a G' such that $G \preceq_I G'$.

Base case: $n = 0$ and G has depth three or less. One can construct all terms of depth three or less and easily show that: either the term is not a generalization or it is more general than one of the terms of S_0. For the induction hypothesis we assume that if G is a term of depth at most $n + 3$ then there exists $G' \in S_n$ such that $G \preceq_I G'$. We show that if G is a term of depth at most $n + 4$ then there exists $G' \in S_{n+1}$ such that $G \preceq_I G'$. We know that $G \approx_I h(g', g'')$ and that g' and g'' are also generalizations of $h(a, b)$ and $h(b, a)$. The term depth of g' and g'' is at most $n + 3$ and thus by the induction hypothesis there exists $H', H'' \in S_n$ such that $g' \preceq_I H'$ and $g'' \preceq_I H''$. Furthermore $S_n \subseteq S_{n+1}$. This implies $G \approx_I h(g', g'') \preceq_I h(H', H'')$, which, by the definition of the S-hierarchy, is a member of S_{n+1}. We take $G' = h(H', H'')$ and thus, the case (d) is proved.

Thus we have shown that for all n, S_n is a minimal complete set of generalizations of $h(a, b)$ and $h(b, a)$ up to term depth $n + 3$. Being that S_∞ is the union of all S_n, the set S_∞ is a minimal complete set of generalizations of $h(a, b)$ and $h(b, a)$.

□

Hence, the S_n construction provides an enumeration of $mcsg_I(h(a, b), h(b, a))$ and can be seen as a finite algorithmic description of this set. We can describe this growth rate as a recurrence for an generalization of the S_n construction where S_0 can have an arbitrary size.

Theorem 4. *Let A be a finite set, $P(S, S') = \{(a, b) \mid a \in S,\ b \in S',\ a \neq b\}$, and S_n the following recursive set construction:*

$$F_0 = 1,$$
$$F_1 = |S_0|,$$
$$F_{n+1} = |S_n| + |P(S_n, S_n)|.$$

Then for $n \geq 1$, $F_{n+1} = F_n^2 - F_{n-1}^2 + F_{n-1}$.

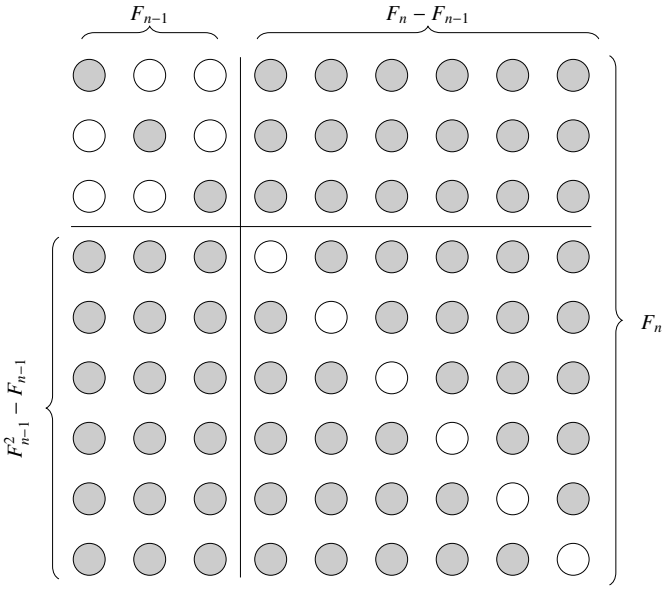

Figure 1: Geometric proof of Theorem 4 for $F_1 = 3$.

Proof. We use induction on n. Let $n = 1$. We have $F_2 = |S_1| \cup |P(S_1, S_1)|$. We know that $|F_1| = |S_0|$ and that $|P(S_1, S_1)| = F_1^2 - F_1$, because $(a, a) \notin P(S_1, S_1)$ for $a \in A$. Thus, $F_2 = |A|^2$ which is precisely given by the formula in the theorem statement, i.e. $F_2 = F_1^2 - F_0^2 + F_0 = |S_0|^2 - 1 + 1 = |S_0|^2$.

For the induction hypothesis, let us assume the theorem holds for F_n and show that it holds for F_{n+1}. We know that F_n is at least as large as F_{n-1} by definition and thus we can consider the subsets of S_n, S_{n-1} and $S_n \setminus S_{n-1}$ when computing F_{n+1}. Note that the elements of $P(S_{n-1}, S_{n-1})$ are already members of $S_n \setminus S_{n-1}$ and thus do not need to be considered. But we do need to consider the following cases $P(S_{n-1}, S_n \setminus S_{n-1})$, $P(S_n \setminus S_{n-1}, S_{n-1})$, $P(S_n \setminus S_{n-1}, S_n \setminus S_{n-1})$ which have size $F_{n-1}(F_n - F_{n-1})$, $F_{n-1}(F_n - F_{n-1})$, $(F_n - F_{n-1})^2 - (F_n - F_{n-1})$, respectively. Thus, we can compute the value of F_{n+1} as the following:

$$
\begin{aligned}
&2 \cdot F_{n-1}(F_n - F_{n-1}) + (F_n - F_{n-1})^2 - (F_n - F_{n-1}) + F_n = \\
&2F_{n-1}F_n - 2F_{n-1}^2 + (F_n^2 - 2F_nF_{n-1} + F_{n-1}^2) + F_{n-1} = \\
&F_n^2 - F_{n-1}^2 + F_{n-1}.
\end{aligned}
$$

It proves the induction step. See Figure 1 for a geometric proof of the theorem. □

This construct can be pushed a little further by considering the ratio between the large and small square in Figure 1 Given that the area of the small square is F_{n-1}^2 and the area of the large square is $(F_{n-1}^2 - F_{n-1})^2 = F_{n-1}^4 - 2F_{n-1}^3 + F_{n-1}^2$.

Theorem 5. *Let F_n be the construction of Theorem 4. Then $F_n \approx O(2^{2^n})$.*

Proof. Consider the ratio between the large and small square found in Figure 1:

$$\frac{(F_{n-1}^2 - F_{n-1})^2}{F_{n-1}^2} = F_{n-1}^2 - 2 \cdot F_{n-1} + 1 = O(n^2) \tag{1}$$

Furthermore, the large square for case n is the small square for case $n+1$, this the ratio between the small square of case n and large square of case $n+1$ must be $O(n^4)$. The construction is a quadratic recurrence which are known to have a complexity of $O(2^{2^n})$.

□

To summarize the results of this section, we note that one idempotent symbol is sufficient to construct an anti-unification problem which has infinitely many incomparable generalizations. A simple example of such a problem is $h(a, b) \triangleq h(b, a)$, for which we showed how to construct a minimal complete set of generalizations.

However, the construction can not help us to construct minimal complete set of generalizations in general case. Therefore, this example only says that idempotent generalization with one idempotent function symbol is either infinitary or of type zero. Still, it is an improvement over result from (Pottier, 1989), which says the same for two idempotent symbols.

In the next sections we prove that I-generalization is indeed infinitary, not only for one idempotent symbol, but for arbitrary number of them. Instead of the S_n construction, tree grammars are used for finite algorithmic description of the minimal complete set of generalizations.

4. Generalization Algorithm

In this section we present an idempotent pre-generalization algorithm for construction of a finite representation of the infinite set of their generalizations.

Our algorithm is given in a rule-based manner, where the rules transform configurations into configurations. A *configuration* is a triple $A,\ S\,;\ B$, where A is a set of anti-unification problems to be solved, S is a set of already solved anti-unification problems (called the store), and B is a set of bindings, representing the generalization "computed so far". The intuitive idea is to collect such B's at the end and construct from them a regular tree grammar, from which one can read off each generalization. We elaborate on the details later, after the rules are formulated.

It is assumed that all terms in A and S are in I-normal form. The rules are defined as follows:

Dec: **Decomposition**

$$\{x : f(s_1, \ldots, s_n) \triangleq f(t_1, \ldots, t_n)\} \cup A;\ S\,;\ B \Longrightarrow$$
$$\{y_1 : s_1 \triangleq t_1, \ldots, y_n : s_n \triangleq t_n\} \cup A;\ S\,;\ B\{x \mapsto f(y_1, \ldots, y_n)\}$$

where f is a free function symbol, $n \geq 0$, and $y_1, \ldots, y_n$ are fresh variables.

Solve: **Solve**

$$\{x : s \triangleq t\} \cup A;\ S\,;\ B \Longrightarrow A;\ \{x : s \triangleq t\} \cup S\,;B,$$

where $head(s) \neq head(t)$ and neither $head(s)$ nor $head(t)$ is an idempotent symbol.

Id-Left: **Idempotent symbol in the left**

$$\{x : f(s_1, s_2) \triangleq t\} \cup A;\ S\,;\ B \Longrightarrow$$
$$\{y_1 : s_1 \triangleq t,\ y_2 : s_2 \triangleq t\} \cup A;\ S\,;\ B\{x \mapsto f(y_1, y_2)\},$$

where f is an idempotent function symbol, $head(t)$ is not idempotent, and y_1 and y_2 are fresh variables.

Id-Right: **Idempotent symbol in the right**

$\{x : s \triangleq f(t_1, t_2)\} \cup A;\ S;\ B \Longrightarrow$
$\{y_1 : s \triangleq t_1, y_2 : s \triangleq t_2\} \cup A;\ S;\ B\{x \mapsto f(y_1, y_2)\}$,

where f is an idempotent function symbol, $head(s)$ is not idempotent, and y_1 and y_2 are fresh variables.

Id-Both 1: **Idempotent symbol in both sides 1**

$\{x : f(s_1, s_2) \triangleq f(t_1, t_2)\} \cup A;\ S;\ B \Longrightarrow$
$\{y_1 : s_1 \triangleq t_1,\ y_2 : s_2 \triangleq t_2\} \cup A;\ S;\ B \cup \{x \mapsto f(y_1, y_2)\}$,

where f is an idempotent function symbol and y_1 and y_2 are fresh variables.

Id-Both 2: **Idempotent symbol in both sides 2**

$\{x : f(s_1, s_2) \triangleq t\} \cup A;\ S;\ B \Longrightarrow$
$\{y_1 : s_1 \triangleq t,\ y_2 : s_2 \triangleq t\} \cup A;\ S;\ B \cup \{x \mapsto f(y_1, y_2)\}$,

where f is an idempotent function symbol, $head(t)$ is idempotent, and y_1 and y_2 are fresh variables.

Id-Both 3: **Idempotent symbol in both sides 3**

$\{x : s \triangleq f(t_1, t_2)\} \cup A;\ S;\ B \Longrightarrow$
$\{y_1 : s \triangleq t_1, y_2 : s \triangleq t_2\} \cup A;\ S;\ B \cup \{x \mapsto f(y_1, y_2)\}$,

where f is an idempotent function symbol, $head(s)$ is idempotent, and y_1 and y_2 are fresh variables.

Merge: **Merge**

$\emptyset;\ \{x_1 : s_1 \triangleq t_1,\ x_2 : s_2 \triangleq t_2\} \cup S;\ B \Longrightarrow \emptyset;\ \{x_1 : s_1 \triangleq t_1\} \cup S;\ B\{x_2 \mapsto x_1\}$,

where $s_1 \approx_I s_2$ and $t_1 \approx_I t_2$.

The idempotent pre-generalization algorithm I-PreGen applies the rules exhaustively, starting from the initial state $\{x : s \triangleq t\};\ \emptyset;\ \{x_{\text{gen}} \mapsto x\}$ for the given terms s and t.

Theorem 6 (Termination)**.** I-PreGen *terminates on any input.*

Proof. All rules strictly reduce the number of symbols in the set of anti-unification problems to be solved. Hence, the rules can be applied finitely many times and the algorithm terminates. □

The idempotent generalization algorithm I-Gen takes as input two terms s and t and performs the following steps:

1. Apply I-PreGen on s and t. After I-PreGen terminates, collect the S's and B's computed in the branches of the derivation tree. Assume that there were n derivation trees, and S_i's and B_i, $1 \leq i \leq n$, are respectively the sets of stores and bindings computed in the leaves of those branches.
2. The variable x_{gen} is the root of each B_i. By construction, in each B_i there are no two rules with the same left-hand side. There is no recursion either. It means that $L(G(B_i))$ consists of a single term. Let it be denoted by $T(B_i)$. Let J be the maximal set of indices $J \subseteq \{1, \ldots, n\}$ such that $rep(T(B_j)) \in Minimize(\cup_{i=1}^{n}\{T(B_i)\})$ for all $j \in J$, i.e., they are terms retained after the minimization of the set $\cup_{i=1}^{n}\{T(B_i)\}$.

3. Apply the Merge rule starting from the configuration $\emptyset; \cup_{j\in J} S_j; \cup_{j\in J} B_j$ as long as possible. In this process, different choices of selecting AUTs to merge might give different final results, but they all will be the same modulo idempotence and variable renaming. It justifies to define the sets STORE and B as the store and binding sets, respectively, in the final configuration of such an exhaustive application of the Merge rule:

$$\emptyset; \cup_{j\in J} S_j; \cup_{j\in J} B_j \Longrightarrow^*_{\mathsf{Merge}} \emptyset;\ \mathsf{STORE};\ \mathsf{B}$$

and Merge does not apply to $\emptyset;\ \mathsf{STORE};\ \mathsf{B}$ anymore.

4. Return STORE and $G(\mathsf{B})$, which is a regular tree grammar that corresponds to B. For the given s and t, we will refer to this grammar as $G(s,t)$. Below we will also use $G_b(s,t)$, which is the base grammar $G_b(\mathsf{B})$.

Based on Theorem 6 and the fact that the construction of $G(\mathsf{B})$ stops, we can conclude that I-Gen is terminating.

By the construction of stores, it is easy to see that STORE does not contain two AUTs $x : s_1 \triangleq t_1$ and $x : s_2 \triangleq t_2$ with the same generalization variable x.

Note that by the construction, if B contains a binding $x \mapsto t$ where $x \neq x_{\text{gen}}$, then there exists at least one other rule $x \mapsto s \in \mathsf{B}$ for the same x and both $head(t)$ and $head(s)$ are idempotent. This happens because only the rules Id-Both 1, Id-Both 2, and Id-Both 3 add new elements to the set of bindings. In such cases, for the same variable we may have two (when only Id-Both 2 and Id-Both 3 apply) or three bindings (when also Id-Both 1 applies) in B. The variable x_{gen} is the *root* of B.

Each variable that appears in the set of terminals of $G(\mathsf{B})$ is a generalization variable which occurs in the store in one of the leaves of the derivation tree of the algorithm above. To distinguish these variables and nonterminals syntactically, we use the bold face font for the latter.

Example 7. Applying I-PreGen to $f(a,b) \triangleq f(b,a)$, where f is idempotent, results in the following sets of bindings:

$$\{x_{\text{gen}} \mapsto x,\ x \mapsto f(y_1, y_2)\},$$
$$\{x_{\text{gen}} \mapsto x,\ x \mapsto f(f(z_1, b), f(a, z_2))\},$$
$$\{x_{\text{gen}} \mapsto x,\ x \mapsto f(f(v_1, a), f(b, v_2))\},$$

from which I-Gen constructs the regular tree grammar $\langle \boldsymbol{x}_{\text{gen}}, N, T, R\rangle$, where

$$N = \{\boldsymbol{x}_{\text{gen}}, \boldsymbol{x}\},$$
$$T = \{f, a, b, z_1, z_2\},$$
$$R = \{\boldsymbol{x}_{\text{gen}} \to \boldsymbol{x},\ \boldsymbol{x}_{\text{gen}} \to f(\boldsymbol{x}_{\text{gen}}, \boldsymbol{x}_{\text{gen}}),$$
$$\boldsymbol{x} \to f(f(z_1, b), f(a, z_2)),\ \boldsymbol{x} \to f(f(z_1, a), f(b, z_2)),\ \boldsymbol{x} \to f(\boldsymbol{x}, \boldsymbol{x})\}.$$

Note that $\boldsymbol{x} \to f(y_1, y_2)$ is omitted, since $f(y_1, y_2)$ is more general than, e.g., $f(f(z_1, b), f(a, z_2))$.
$\mathsf{STORE} = \{z_1 : a \triangleq b,\ z_2 : b \triangleq a\}$. ▶

Example 8. Applying I-PreGen to $f(a, f(a,b)) \triangleq f(a, f(b,a))$, where f is idempotent, gives the following sets of bindings:

$$\{x_{\text{gen}} \mapsto x_1,\ x_1 \mapsto f(a, x_2),\ x_2 \mapsto f(f(y_1, a), f(b, y_2))\},$$

$\{x_{\text{gen}} \mapsto x_1,\ x_1 \mapsto f(a, x_2),\ x_2 \mapsto f(f(y_3, b), f(a, y_4))\},$
$\{x_{\text{gen}} \mapsto x_1,\ x_1 \mapsto f(f(a, f(z_1, a)), x_3),\ x_3 \mapsto f(a, f(b, z_2))\},$
$\{x_{\text{gen}} \mapsto x_1,\ x_1 \mapsto f(f(a, f(z_1, a)), x_3),\ x_3 \mapsto f(f(a, f(z_1, a)), f(z_3, f(b, z_3)))\},$
$\{x_{\text{gen}} \mapsto x_1,\ x_1 \mapsto f(f(a, f(z_1, a)), x_3),\ x_3 \mapsto f(f(a, z_4), x_5),\ x_5 \mapsto f(z_1, z_4)\},$
$\{x_{\text{gen}} \mapsto x_1,\ x_1 \mapsto f(f(a, f(z_1, a)), x_3),\ x_3 \mapsto f(f(a, z_4), x_5),\ x_5 \mapsto f(f(z_1, a), f(b, z_4))\},$
$\{x_{\text{gen}} \mapsto x_1,\ x_1 \mapsto f(f(a, f(z_1, a)), x_3),\ x_3 \mapsto f(f(a, z_4), x_5),\ x_5 \mapsto f(f(z_1, b), f(a, z_4))\},$
$\{x_{\text{gen}} \mapsto x_1,\ x_1 \mapsto f(f(a, f(a, v_1)), x_4),\ x_4 \mapsto f(v_2, f(a, v_1))\},$
$\{x_{\text{gen}} \mapsto x_1,\ x_1 \mapsto f(f(a, f(a, v_1)), x_4),\ x_4 \mapsto f(f(v_3, a), x_6),\ x_6 \mapsto f(v_3, v_1)\},$
$\{x_{\text{gen}} \mapsto x_1,\ x_1 \mapsto f(f(a, f(a, v_1)), x_4),\ x_4 \mapsto f(f(v_3, a), x_6),\ x_6 \mapsto f(f(v_3, a), f(b, v_1))\},$
$\{x_{\text{gen}} \mapsto x_1,\ x_1 \mapsto f(f(a, f(a, v_1)), x_4),\ x_4 \mapsto f(f(v_3, a), x_6),\ x_6 \mapsto f(f(v_3, b), f(a, v_1))\},$
$\{x_{\text{gen}} \mapsto x_1,\ x_1 \mapsto f(f(a, f(a, v_1)), x_4),\ x_4 \mapsto f(f(v_4, f(v_4, b)), f(a, f(a, v_1)))\}$

from which I-Gen constructs the grammar $\langle \boldsymbol{x}_{\text{gen}}, N, T, R\rangle$, where

$N = \{\boldsymbol{x}_{\text{gen}}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_6\},$
$T = \{f, a, b, y_1, y_2\},$
$R = \{\boldsymbol{x}_{\text{gen}} \to \boldsymbol{x}_1,\ \boldsymbol{x}_{\text{gen}} \to f(\boldsymbol{x}_{\text{gen}}, \boldsymbol{x}_{\text{gen}}),$
$\boldsymbol{x}_1 \to f(a, \boldsymbol{x}_2),\ \boldsymbol{x}_1 \to f(f(a, f(y_1, a)), \boldsymbol{x}_3),\ \boldsymbol{x}_1 \to f(f(a, f(a, y_2)), \boldsymbol{x}_4),$
$\boldsymbol{x}_1 \to f(\boldsymbol{x}_1, \boldsymbol{x}_1),$
$\boldsymbol{x}_2 \to f(f(y_1, a), f(b, y_2)),\ \boldsymbol{x}_2 \to f(f(y_1, b), f(a, y_2)),\ \boldsymbol{x}_2 \to f(\boldsymbol{x}_2, \boldsymbol{x}_2),$
$\boldsymbol{x}_3 \to f(a, f(b, y_2)),\ \boldsymbol{x}_3 \to f(f(a, f(y_1, a)), f(y_2, f(b, y_2))),\ \boldsymbol{x}_3 \to f(f(a, y_2), \boldsymbol{x}_5),$
$\boldsymbol{x}_3 \to f(\boldsymbol{x}_3, \boldsymbol{x}_3),$
$\boldsymbol{x}_4 \to f(f(y_1, a), \boldsymbol{x}_6),\ \boldsymbol{x}_4 \to f(f(y_1, f(y_1, b)), f(a, f(a, y_2))),\ \boldsymbol{x}_4 \to f(\boldsymbol{x}_4, \boldsymbol{x}_4),$
$\boldsymbol{x}_5 \to f(f(y_1, a), f(b, y_2)),\ \boldsymbol{x}_5 \to f(f(y_1, b), f(a, y_2)),\ \boldsymbol{x}_5 \to f(\boldsymbol{x}_5, \boldsymbol{x}_5),$
$\boldsymbol{x}_6 \to f(f(y_1, a), f(b, y_2)),\ \boldsymbol{x}_6 \to f(f(y_1, b), f(a, y_2)),\ \boldsymbol{x}_6 \to f(\boldsymbol{x}_6, \boldsymbol{x}_6)\}.$

$\mathsf{STORE} = \{y_1 : a \triangleq b, y_2 : b \triangleq a\}$. The derivation below illustrates that $f(a, f(f(f(y_1, a), f(b, y_2)), f(f(y_1, b), f(a, y_2))) \in L(G(f(a, f(a, b)), f(a, f(b, a))))$:

$$\begin{aligned}\boldsymbol{x}_{\text{gen}} &\to \boldsymbol{x}_1 \to f(a, \boldsymbol{x}_2) \to f(a, f(\boldsymbol{x}_2, \boldsymbol{x}_2)) \to f(a, f(f(f(y_1, a), f(b, y_2)), \boldsymbol{x}_2))\\ &\to f(a, f(f(f(y_1, a), f(b, y_2)), f(f(y_1, b), f(a, y_2)))).\end{aligned}$$

▶

Example 9. Applying I-PreGen to $g(a, f(a, b)) \triangleq g(a, f(b, a))$, where g is free and f is idempotent, gives the following sets of bindings:

$\{x_{\text{gen}} \mapsto g(a, x),\ x \mapsto f(y_1, y_2)\},$
$\{x_{\text{gen}} \mapsto g(a, x),\ x \mapsto f(f(z_1, b), f(a, z_2))\},$
$\{x_{\text{gen}} \mapsto g(a, x),\ x \mapsto f(f(v_1, a), f(b, v_2))\},$

from which I-Gen constructs the grammar $\langle \boldsymbol{x}_{\text{gen}}, N, T, R\rangle$, where

$N = \{\boldsymbol{x}_{\text{gen}}, \boldsymbol{x}\},$

$$T = \{f, g, a, b, z_1, z_2\},$$
$$R = \{\boldsymbol{x}_{\text{gen}} \to g(a, \boldsymbol{x}),\ \boldsymbol{x}_{\text{gen}} \to f(\boldsymbol{x}_{\text{gen}}, \boldsymbol{x}_{\text{gen}}),$$
$$\boldsymbol{x} \to f(f(z_1, b), f(a, z_2)),\ \boldsymbol{x} \to f(f(z_1, a), f(b, z_2)),\ \boldsymbol{x} \to f(\boldsymbol{x}, \boldsymbol{x})\}.$$

As for STORE, it is $\{z_1 : a \triangleq b,\ z_2 : b \triangleq a\}$. We have, for instance, $f(g(a, f(f(z_1, b), f(a, z_2))), g(a, f(f(z_1, a), f(b, z_2)))) \in L(G(g(a, f(a, b)), g(a, f(b, a))))$. To see that this term is indeed a common I-generalization of $g(a, f(a, b))$ and $g(a, f(b, a))$, we can read off the matching substitutions from STORE: $\sigma_1 = \{z_1 \mapsto a,\ z_2 \mapsto b\}$ and $\sigma_2 = \{z_1 \mapsto b,\ z_2 \mapsto a\}$ and check:

$$f(g(a, f(f(z_1, b), f(a, z_2))), g(a, f(f(z_1, a), f(b, z_2))))\sigma_1 =$$
$$f(g(a, f(f(a, b), f(a, b))), g(a, f(f(a, a), f(b, b)))) \approx_I$$
$$f(g(a, f(a, b)), g(a, f(a, b))) \approx_I g(a, f(a, b)).$$
$$f(g(a, f(f(z_1, b), f(a, z_2))), g(a, f(f(z_1, a), f(b, z_2))))\sigma_2 =$$
$$f(g(a, f(f(b, b), f(a, a))), g(a, f(f(b, a), f(b, a)))) \approx_I$$
$$f(g(a, f(b, a)), g(a, f(b, a))) \approx_I g(a, f(b, a)).$$

▶

Example 10. Let f and g be idempotent symbols. Then for $f(a, b) \triangleq g(a, c)$ the regular tree grammar computed by I-PreGen is $\langle \boldsymbol{x}_{\text{gen}}, N, T, R\rangle$, where

$$N = \{\boldsymbol{x}_{\text{gen}}, \boldsymbol{x}\},$$
$$T = \{f, g, a, y_1, y_2, y_3\},$$
$$R = \{\boldsymbol{x}_{\text{gen}} \to \boldsymbol{x},\ \boldsymbol{x}_{\text{gen}} \to f(\boldsymbol{x}_{\text{gen}}, \boldsymbol{x}_{\text{gen}}),\ \boldsymbol{x}_{\text{gen}} \to g(\boldsymbol{x}_{\text{gen}}, \boldsymbol{x}_{\text{gen}}),$$
$$\boldsymbol{x} \to f(g(a, y_1), g(y_2, y_3)),\ \boldsymbol{x} \to g(f(a, y_2), f(y_1, y_3)),\ \boldsymbol{x} \to f(\boldsymbol{x}, \boldsymbol{x}),\ \boldsymbol{x} \to g(\boldsymbol{x}, \boldsymbol{x})\}.$$

STORE is $\{y_1 : a \triangleq c,\ y_2 : b \triangleq a,\ y_3 : b \triangleq c\}$.

I-Gen is sound, as the following theorem shows:

Theorem 11 (Soundness)**.** *Let s and t be two terms and $G(s, t)$ be the regular tree grammar computed by* I-Gen *for them. Then any term in $L(G(s, t))$ is a common I-generalization of s and t.*

Proof. First, notice that if $r_1, r_2 \in L(G(s, t))$ are I-generalizations of s and t, then $f(r_1, r_2)$ is also an I-generalization of s and t. Indeed, assume r_1 and r_2 generalize s. Then there exist substitutions σ_1 and σ_2 such that $r_i\sigma_i \approx_I s$, $i = 1, 2$. From the algorithm rules it follows that we can actually read off these σ's from STORE. It implies that if $x \in dom(\sigma_1) \cap dom(\sigma_2)$, then $x\sigma_1 = x\sigma_2$. Let ϑ be a substitution defined as $x\vartheta = x\sigma_1$ if $x \in dom(\sigma_1)$ and $x\vartheta = x\sigma_2$ otherwise. Then we have $f(r_1, r_2)\vartheta = f(r_1\sigma_1, r_2\sigma_2) \approx_I f(s, s) \approx_I s$, which shows that $f(r_1, r_2)$ is an I-generalization of s. For t it is analogous.

Hence, it remains to prove that every term in $L(G_b(s, t))$ is a common I-generalization of s and t. For the elements of $L(G(s, t))$ generated with the help of duplication rules, the theorem follows from the what we just showed.

The terms of $L(G_b(s, t))$ are precisely those derived using I-PreGen (i.e., the right hand sides of bindings in binding sets computed by I-PreGen) and thus we can prove the theorem by well-founded induction over the length of the derivation of terms in $L(G_b(s, t))$.

Let as assume that the proposition holds for each r, which is derived by I-PreGen in at most n steps and prove it for the case of $n+1$ steps. We consider different cases based what rule can apply to the initial configuration $\{x : s \triangleq t\};\ \emptyset;\ \{x_{\text{gen}} \mapsto x\}$.

Dec: If $s = a$ and $t = a$, the decomposition rule gives the final configuration $\emptyset;\ \emptyset,\ \{x_{\text{gen}} \mapsto a\}$, $L(G_b(s,t)) = \{a\}$ and the theorem is true.

Now assume $s = f(s_1, \ldots, s_m)$, $t = f(t_1, \ldots, t_m)$. Take one of the derivations:

$$\{x : f(s_1, \ldots, s_m) \triangleq f(t_1, \ldots, t_m)\};\ \emptyset;\ \{x_{\text{gen}} \mapsto x\} \Longrightarrow_{\mathsf{Dec}}$$
$$\{x_1 : s_1 \triangleq t_1, \ldots, x_m : s_m \triangleq t_m\};\ \emptyset;\ \{x_{\text{gen}} \mapsto f(x_1, \ldots, x_m)\} \Longrightarrow^n \emptyset;\ S;\ B.$$

We have $L(G_b(B)) \subseteq L(G_b(s,t))$. The part of this derivation after the first step can be represented as a combination of m derivations of length at most n:

$$\{x_i : s_i \triangleq t_i\};\ \emptyset;\ \{y_i \mapsto x_i\} \Longrightarrow_{R^i_1} \cdots \Longrightarrow_{R^i_{n_i}} \emptyset;\ S_i;\ \{y_i \mapsto l'_i\} \cup B'_i, \tag{2}$$

where $1 \le i \le m$, $n_1 + \cdots + n_m \le n$, and R^i_k is the rule name used in the ith derivation at kth step. The mentioned combination can be expressed, e.g., like this:

$$\{x_1 : s_1 \triangleq t_1, \ldots, x_m : s_m \triangleq t_m\};\ \emptyset;$$
$$\quad \{x_{\text{gen}} \mapsto f(y_1, \ldots, y_m), y_1 \mapsto x_1, \ldots, y_m \mapsto x_m\} \Longrightarrow_{R^1_1} \cdots \Longrightarrow_{R^1_{n_1}}$$
$$\{x_2 : s_2 \triangleq t_2, \ldots, x_m : s_m \triangleq t_m\};\ S_1;$$
$$\quad \{x_{\text{gen}} \mapsto f(y_1, \ldots, y_m), y_1 \mapsto l'_1, \ldots, y_m \mapsto x_m\} \cup B'_1 \Longrightarrow_{R^2_1} \cdots \Longrightarrow_{R^2_{n_2}}$$
$$\cdots \Longrightarrow_{R^m_1} \cdots \Longrightarrow_{R^m_{n_m}}$$
$$\emptyset;\ S_1 \cup \cdots \cup S_m;$$
$$\quad \{x_{\text{gen}} \mapsto f(y_1, \ldots, y_m), y_1 \mapsto l'_1, \ldots, y_m \mapsto l'_m\} \cup B'_1 \cup \cdots \cup B'_m \Longrightarrow^*_{\mathsf{Merge}}$$
$$\emptyset;\ S;\ \{x_{\text{gen}} \mapsto f(y_1, \ldots, y_m), y_1 \mapsto l_1, \ldots, y_m \mapsto l_m\} \cup B_1 \cup \cdots \cup B_m.$$

Although this derivation and the derivation $\{x_1 : s_1 \triangleq t_1, \ldots, x_m : s_m \triangleq t_m\};\ \emptyset;\ \{x_{\text{gen}} \mapsto f(x_1, \ldots, x_m)\} \Longrightarrow^n \emptyset;\ S;\ B$ syntactically slightly differ from each other (the presence of y's in the first one is the difference), they generate equivalent tree grammars: the languages generated by them are the same (up to renaming of store variables).

By the induction hypothesis, the elements of the language $L(G_b(s_i, t_i)) = L(G_b(\{y_i \mapsto l'_1\} \cup B'_i))$ are common I-generalizations of s_i and t_i. By the tree grammar definition, it implies the elements of the language $L(G_b(\{x_{\text{gen}} \mapsto f(y_1, \ldots, y_m), y_1 \mapsto l'_1, \ldots, y_m \mapsto l'_m\} \cup B'_1 \cup \cdots \cup B'_m))$ are common I-generalizations of $s = f(s_1, \ldots, s_m)$ and $t = f(t_1, \ldots, t_m)$. If there are two AUTs in $S_1 \cup \cdots \cup S_m$ which are subject of Merge, it indicates that in s there are two positions p'_s and p''_s containing I-equivalent subterms, in t there are two positions p'_t and p''_t containing I-equivalent subterms, the subterms $s|_{p'_s}$ and $t|_{p'_t}$ are generalized by a variable v', and $s|_{p''_s}$ and $t|_{p''_t}$ are generalized by another variable v''. However, since $s|_{p'_s} \approx_I s|_{p''_s}$ and $t|_{p'_t} \approx_I t|_{p''_t}$, one can just use v' in place of v'' in a generalization of s and t and still have their generalization. Hence, the sequence of Merge rule applications transform generalizations into generalizations, meaning that every element of the set $L(G_b(f(y_1, \ldots, y_m), y_1 \mapsto l_1, \ldots, y_m \mapsto l_m\} \cup B_1 \cup \cdots \cup B_m)) = L(G_b(B))$ is a common I-generalization of s and t. Since the derivation ending with B was chosen arbitrarily, by construction of $L(G_b(s,t))$ we get that this set consists of common I-generalizations of s and t.

Solve: If the AUT for s and t is transformed by this rule, the final configuration is $\emptyset$; $\{x : s \triangleq t\}$, $\{x_{\text{gen}} \mapsto x\}$, $L(G_b(s,t)) = \{x\}$ and the theorem holds trivially.

The rules for idempotent heads: Id-Left, Id-Right, Id-Both 1, Id-Both 2, Id-Both 3. Reasoning for these cases is similar to the reasoning for the case with the decomposition rule above. There is an extra detail to be taken into account: for Id-Left, for instance, we should observe that if $s = f(s_1, s_2)$ and r_i is a common I-generalization for s_i and t, $i = 1, 2$, then $f(r_1, r_2)$ is a common I-generalization of s and t.

Merge: This rule transforms a common I-generalization of s and t into another, less general common I-generalization of s and t. It finishes the proof.

□

The algorithm I-Gen is also complete:

Theorem 12 (Completeness)**.** *Let t_1 and t_2 be two terms and $G(t_1, t_2)$ be the regular tree grammar computed by* I-Gen *for them. Let r be a common I-generalization of t_1 and t_2. Then there exists $s \in L(G(t_1, t_2))$ such that $r \preceq_I s$.*

Proof. By structural induction on r. We assume that t_1 and t_2 are in I-normal form and do not share variables. The latter is not a restriction since we can treat variables in t_1 and t_2 as constants. Note also that the simplification involved in the construction of $G(t_1, t_2)$ by *Minimize* does not violate the claim of the theorem, since its effect is to remove a more general term from $L(G(t_1, t_2))$ while keeping less general ones there.

Case 1. r is a variable.. It implies that $head(t_1) \neq head(t_2)$ and none of those heads are idempotent. We use Solve and get that $L(G(t_1, t_2))$ contains a variable, which proves this case.

Case 2. r is a constant.. let r be a. Then both t_1 and t_2 should be a and by Dec rule we obtain $a \in L(G(t_1, t_2))$.

Case 3. $r = f(r_1, \ldots, r_n)$ and f is free.. Then t_1 and t_2 both have f as the head. Let $t_1 = f(t'_1, \ldots, t'_n)$ and $s_2 = f(t''_1, \ldots, t''_n)$. Then r_i is a common I-generalization of t'_i and t''_i. It might happen that for some $1 \leq j, k, \leq n$, the terms r_j and r_k share some (terminal) variables, e.g., z. It indicates that t'_j and t'_k have a common subterm l', and t''_j and t''_k have a common subterm l''. The occurrence of l' in t'_j (resp. in t'_k) corresponds to the occurrence of l'' in t''_j (resp. in t''_k). These are those subterms that are generalized by the variable z shared between r_j and r_k.

The original anti-unification problem between t_1 and t_2 can be transformed by the Dec rule, which gives rise to new anti-unification problems $y_i : t'_i \triangleq t''_i$, $1 \leq i \leq n$. If we assume that each $y_i : t'_i \triangleq t''_i$ is a new AUT and run I-Gen for each of them, we obtain $G(t'_i, t''_i)$, $1 \leq i \leq n$.

By the induction hypothesis, there exists $s_i \in L(G(t'_i, t''_i))$ such that $r_i \preceq_I s_i$. Hence, each $G(t'_i, t''_i)$ contains a rule $y_{\text{gen}_i} \to q_i$, such that y_{gen_i} is the axiom and s_i is obtained from q_i, where q_i is either y_i or an instance of y_i. Then s_j contains a subterm l_j and s_k contains a subterm l_k, which are instances of z. Since at this stage s_j and s_k are independent, those subterms are not necessarily the same, but each of them is a computed generalization of l' and l''. We can choose the computing derivations in such a way that l_j and l_k are obtained by exactly the same applications of the rules. This is possible because they generalize the same terms, and the derivations are guided by those terms. Therefore, if l_j and l_k are not the same, they would differ by the names of fresh (terminal) variables generated during the two derivations.

Now we need to put those separate derivations together to show that there exists the desired $s \in L(G(t_1, t_2))$. Essentially, the only required addition would be to consider the necessity of the Merge rule, which would make sure that the subterms such as l_j and l_k above are generalized in the same way. Then the grammar $G(t_1, t_2)$ will contain a rule $x_{\text{gen}} \to f(q_1, \ldots, q_n)$, where the q_i's are as above with the difference that terms s_j and s_k they generate may have shared (terminal) variables. Let $\hat{s}_i$, $1 \leq i \leq n$ be the instance of the corresponding s_i under such variable-sharing if there exists one. Otherwise, $\hat{s}_i = s_i$. Hence, for all terms in $L(G(t_i', t_i''))$, and, in particular, for $\hat{s}_i$, $1 \leq i \leq n$, we have $s = f(\hat{s}_1, \ldots, \hat{s}_n) \in L(G(t_1, t_2))$. On the other hand, we have $r = f(r_1, \ldots, r_n) \preceq_I f(\hat{s}_1, \ldots, \hat{s}_n)$, which finishes the proof of this case.

Case 3. $r = f(r_1, r_2)$ and f is idempotent.. Then we have the following subcases:

1. $t_1 = f(t_1', t_2')$, $t_2 = f(t_1'', t_2'')$ and r_i is a common I-generalization of t_i' and t_i'', $i = 1, 2$. In this case we use Id-Both 1 rule and proceed similarly to Case 3 above.
2. $t_1 = f(t_1', t_2')$ and r_i is a common generalization of t_i' and t_2, $i = 1, 2$. Depending whether $head(t_2)$ is idempotent or not, we proceed either by Id-Left or Id-Both 1 and reason similarly to Case 3.
3. $t_2 = f(t_1'', t_2'')$ and r_i is a common generalization of t_1 and t_i'', $i = 1, 2$. Depending whether $head(t_1)$ is idempotent or not, we proceed either by Id-Right or Id-Both 2 and reason similarly to Case 3.
4. Both r_1 and r_2 are common I-generalizations of t_1 and t_2. (It means that there exist substitutions σ_1 and σ_2 such that $r_1\sigma_1 \approx_I r_2\sigma_1 \approx_I t_1$ and $r_1\sigma_2 \approx_I r_2\sigma_2 \approx_I t_2$.) By the induction hypothesis, we have $s_1, s_2 \in L(G(t_1, t_2))$ such that $r_i \preceq_I s_i$, $i = 1, 2$. By the construction of $G(t_1, t_2)$, if $s_1, s_2 \in G(t_1, t_2)$, then $f(s_1, s_2) \in G(t_1, t_2)$. Since $f(r_1, r_2) \preceq_I f(s_1, s_2)$, this case is also proved.

□

Let σ_{t_1} and σ_{t_2} be substitutions obtained from STORE after I-Gen has been applied to t_1 and t_2:

$$\sigma_{t_1} := \{x \mapsto l \mid x : l \triangleq r \in \text{STORE for some } r\},$$
$$\sigma_{t_2} := \{x \mapsto r \mid x : l \triangleq r \in \text{STORE for some } l\}.$$

Then the following theorem holds:

Theorem 13. *$L(G(t_1, t_2))\sigma_{t_i}$ is the I-equivalence class for the term t_i, $i = 1, 2$.*

Proof. By Theorem 11, every element $t \in L(G(t_1, t_2))$ is an I-generalization of t_1 and t_2. The only instantiable variables in t are those which are the generalization variables in STORE, i.e., those in the domain of σ_{t_i}. By the Solve rule, if $x : l \triangleq r$ is in the store, x generalizes an occurrence of l in t_1 and an occurrence of r from t_2. Then we get $t\sigma_{t_i} \approx_I t_i$, and, hence, $L(G(t_1, t_2))\sigma_{t_i}$ is a subset of the I-equivalence class for t_i, $i = 1, 2$. The other inclusion follows from the Completeness Theorem (Theorem 12).

□

5. Idempotent Generalization is Infinitary

To show that idempotent generalization is infinitary we must prove that for any terms s and t the language $L(G(s,t))$ is complete and minimal with respect to a given idempotent equational theory.

We assume that $L(G(s,t))$ is I-normalized (i.e., $nf_I(L(G(s,t)) = L(G(s,t))$ and proceed as follows: First, we show that $L(G_b(s,t))$ is minimal (i.e., it does not contain distinct terms that are comparable by $\preceq_I$). Next, to each term r in $L(G(s,t))$ we associate a natural number $dra(r)$ which indicates the maximal number of applications of the duplicating rules (rules from $R \setminus R_b$) needed to construct r, and define $L_n(G(s,t)) = \{r \mid r \in L(G(s,t)) \text{ and } dra(r) \leq n\}$. These are finite languages for all $n \geq 0$, and $L_0(G(s,t)) = L(G_b(s,t))$. Then we prove that for all $n \geq 0$, $L_n(G(s,t))$ is minimal. Since $L_n(G(s,t)) \subseteq L_{n+1}(G(s,t))$ for all $n \geq 0$ and $\cup_{n=0}^{\infty} L_n(G(s,t)) = L(G(s,t))$, it implies that $L(G(s,t))$ is not only complete, but also minimal.

Minimality of $L(G_b(s,t))$ follows from the construction, since the *Simplify* function makes sure that no two terms generated by the same nonterminal are comparable with respect to $\preceq_I$. Before presenting our proof of minimality we provide the following helpful lemma

Lemma 14. *Let $s \triangleq t$ be an AUP over a signature Σ which may contain idempotent function symbols. Let t_1 and t_2 be $\preceq_I$-incomparable generalizations of $s \triangleq t$ and σ be a substitution such that $t_1\sigma$ and $t_2\sigma$ are not generalizations of $s \triangleq t$. Then for any idempotent function symbol $f \in \Sigma$, $f(t_1\sigma, t_2\sigma)$ is not a generalization of $s \triangleq t$.*

Proof. We perform induction over the term depth of t_1 and t_2.

Case: ($\mathbf{BC}_1$) If $dep(t_1) = dep(t_2) = 1$ then one of the following must hold:

- They are the same constant, i.e. $t_1 = t_2$.
- They are both variables.
- one is a constant and one is a variable.

In each case the terms are not $\preceq_I$-incomparable and thus trivially satisfying the lemma.

Case: ($\mathbf{BC}_2$) Let us assume, w.l.o.g, that the Lemma holds when $dep(t_1) = n$ and $dep(t_2) = 1$. We show that the Lemma holds when $dep(t_1) = n + 1$ and $dep(t_2) = 1$. Once again the terms are not $\preceq_I$-incomparable and thus trivially satisfy the Lemma.

Case: ($\mathbf{IH}_1$) Let us assume that the Lemma holds for $dep(t_1) = n$ and $dep(t_2) = m$ and show that it holds for $dep(t_1) = n + 1$ and $dep(t_2) = m + 1$.

Case 1: Assume that $t_1 = g(s_1, \cdots, s_k)$ and $t_2 = g(r_1, \cdots, r_k)$ for a non-idempotent function symbol g. This implies that $s = g(s'_1, \cdots, s'_k)$ and $t = g(r'_1, \cdots, r'_k)$. Thus, for $f(t_1\sigma, t_2\sigma)$ to be a generalization it is necessarily the case that $t_1\sigma = t_2\sigma$ implying, without loss of generality, that $f(t_1\sigma, t_2\sigma) = t_1\sigma$ and thus contradicting our assumption that $t_1\sigma$ is not a generalization.

Case 2: Assume that $t_1 = g(s_1, s_2)$ and $t_2 = g(r_1, r_2)$ where g is idempotent. This results in two distinct cases dependent on s and t.

Case 2a: Let us further assume that $s = g(s'_1, s'_2)$ and $t = g(r'_1, r'_2)$. We want to show that $g(t_1\sigma, t_2\sigma)$ is a generalization and reach a contradiction. In order to show a contradiction we consider the four possible ways t_1 and t_2 generalizes $s \triangleq t$. Either s_1 generalizes $s'_1 \triangleq r'_1$ and s_2 generalizes $s'_2 \triangleq r'_2$ ((2aa) and (2ab)) or both s_1 and s_2 generalize $s \triangleq t$ ((2ac) and (2ad)). Also, Either r_1 generalizes $s'_1 \triangleq r'_1$ and r_2 generalizes $s'_2 \triangleq r'_2$ ((2aa) and (2ac)) or both r_1 and r_2 generalize $s \triangleq t$ ((2ab) and (2ad)).

Case 2aa: In this case, given that $t_1\sigma$ and $t_2\sigma$ do not generalize $s \triangleq t$ it is necessarily the case that $s_1\sigma$ does not generalize $s'_1 \triangleq r'_1$, $s_2\sigma$ does not generalize $s'_2 \triangleq r'_2$, $r_1\sigma$ does not generalize $s'_1 \triangleq r'_1$, and $r_2\sigma$ does not generalize $s'_2 \triangleq r'_2$. This implies that there does not exists θ_1 and θ_2 such that $g(t_1\sigma, t_2\sigma)\theta_1 = g(s, s) = s$ and $g(t_1\sigma, t_2\sigma)\theta_2 = g(t, t) = t$. Now let us assume there exists a θ_1 and θ_2 such that $g(t_1\sigma, t_2\sigma)\theta_1 = g(g(s'_1, s'_1), g(s'_2, s'_2)) = s$ and $g(t_1\sigma, t_2\sigma)\theta_2 = g(g(r'_1, r'_1), g(r'_2, r'_2)) = t$. This would imply $t_1\sigma$ is a generalization of $s'_1 \triangleq r'_1$ and that $t_2\sigma$ is a generalization of $s'_2 \triangleq r'_2$ contradicting out assumption. Thus showing that this case cannot occur.

Case 2ab: In this case, given that $t_1\sigma$ and $t_2\sigma$ do not generalize $s \triangleq t$ it is necessarily the case that $s_1\sigma$ does not generalize $s'_1 \triangleq r'_1$, $s_2\sigma$ does not generalize $s'_2 \triangleq r'_2$, $r_1\sigma$ does not generalize $s \triangleq t$, and $r_2\sigma$ does not generalize $s \triangleq t$. Using a similar argument as the one used for (2aa) we can get a contradiction by just considering $t_1\sigma$, thus showing that this case leads to a contradiction.

Case 2ac: In this case, given that $t_1\sigma$ and $t_2\sigma$ do not generalize $s \triangleq t$ it is necessarily the case that $s_1\sigma$ does not generalize $s \triangleq t$, $s_2\sigma$ does not generalize $s \triangleq t$, $r_1\sigma$ does not generalize $s'_1 \triangle r'_1$, and $r_2\sigma$ does not generalize $s'_2 \triangleq r'_2$. Using a similar argument as the one used for (2aa) we can get a contradiction by just considering $t_2\sigma$, thus showing that this case leads to a contradiction.

Case 2ad: In this case, given that $t_1\sigma$ and $t_2\sigma$ do not generalize $s \triangleq t$ it is necessarily the case that $s_1\sigma$ does not generalize $s \triangleq t$, $s_2\sigma$ does not generalize $s \triangleq t$, $r_1\sigma$ does not generalize $s \triangle t$, and $r_2\sigma$ does not generalize $s \triangleq r$. This case is an instance of the induction hypothesis and thus contradicts our assumption.

Case 2b: Let us assume that $s = g(s'_1, s'_2)$ and $t = g(r'_1, r'_2)$. We want to show that $f(t_1\sigma, t_2\sigma)$, s.t. $f \neq g$, is a generalization and reach a contradiction. This is similar to (2a) except there are less cases to consider, i.e. if θ is a substitution which when applied to $f(t_1\sigma, t_2\sigma)$ results in, without loss of generality, s then the outer most occurrence of f does not occur in s after idempotent normalization of $f(t_1\sigma, t_2\sigma)\theta$.

Case 3: Assume that w.l.o.g $t_1 = g(s_1, \cdots, s_k)$ and $t_2 = f(r_1, r_2)$ such that g is a syntactic function symbol and f is idempotent. This implies that $s = g(s'_1, \cdots, s'_k)$ and $t = g(r'_1, \cdots, r'_k)$, and furthermore, it implies that r_1 and r_2 must be generalizations of s and t, i.e. there exists θ_1 and θ_2 s.t. $f(r_1, r_2)\theta_1 = f(s, s) = s$ and $f(r_1, r_2)\theta_2 = f(t, t) =$

t. Thus, for $f(t_1\sigma, t_2\sigma)$ to be a generalization it is necessarily the case that $r_1\sigma$ and $r_2\sigma$ to be generalizations of s and t. However, by the induction hypothesis this results in a contradiction.

We have covered all possible cases. and thus have proven the Lemma. □

Lemma 15. *Let $s \triangleq t$ be an AUP over a signature Σ which may contains idempotent function symbols s.t. there exists $n \in \mathbb{N}$ up to which $L_n(G(s,t))$ is both minimal and complete. Let $w \in L_{n+1}(G(s,t)) \setminus L_n(G(s,t))$, $r \in L_n(G(s,t))$, and p a position be s.t. $w = r\left[f(t_1,t_2)\right]_p$ for $t_1, t_2 \in L_n(G(s,t))$. Then it is necessarily the case that neither $t_1 \prec_I f(t_1,t_2)$ nor $t_2 \prec_I f(t_1,t_2)$.*

Proof. Without loss of generality, we can focus on proving $t_1 \not\prec_I f(t_1,t_2)$. we do so by considering two cases:

Case 1: Assume that $\exists p$ s.t. $r|_p = t_1$ and $r \in L_0(G(s,t))$. This case implies that t_1 is constructed without the use of duplication rules of $R \setminus R_b$. Now let us consider $[x \to t^*(\bar{x})] \in R_b$, where $\bar{x}$ is a list of the non-terminals occurring in t^*, s.t. $t_1 = t^*(\bar{x})$. In particular let us consider $\bar{x}$ to be empty. This means that I-PreGen did not use any of the id-both rules when constructing t_1. Furthermore, as a final assumption, t_1 does not contain idempotent function symbols. Under these constraints t_1 is a syntactic part of the generalization and thus $t_1 \not\prec_I f(t_1,t_2)$ ($\mathbf{BC}_1$).

Assume that t_1 contains at most m idempotent function symbols and $t_1 \not\prec_I f(t_1,t_2)$. We refer to this induction hypothesis as ($\mathbf{IH}_2$). We show that $t_1 \not\prec_I f(t_1,t_2)$ when t_1 contains at most $m+1$ idempotent function symbols.

There are only two interesting cases to consider, when $t_1 = f(t'_1,t'_2)$ and when $t_1 = g(t'_1,t'_2)$ where g is idempotent and $f \neq g$. The statement $t_1 \not\prec_I f(t_1,t_2)$ trivial holds when $head(t_1)$ is non-idempotent. Let us now assume that there exists a substitution σ such that $t_1\sigma =_I f(t_1,t_2)$, $t'_1\sigma =_I f(t_1,t_2)$ and $t'_2\sigma =_I f(t_1,t_2)$. Notice that the existence of such a σ allows us to reformulate the statement we are proving as $t'_1 \not\prec_I f(t_1,t_2)$. Being that t'_1 has m idempotent symbols, by $\mathbf{IH}_2$, $t'_1 \not\prec_I f(t_1,t_2)$ holds. If instead there exists σ such that $t_1\sigma =_I f(t_1,t_2)$ and $t'_1\sigma =_I t_1$ and $t'_2\sigma =_I t_2$ then we reformulate the statement we are proving as $t'_1 \not\prec_I f(t'_1,t'_2)$ which once again has fewer idempotent symbols and by $\mathbf{IH}_2$, $t'_1 \not\prec_I f(t'_1,t'_2)$. If instead $t_1 = g(t'_1,t'_2)$ where g is idempotent (not equivalent to f) there exists a substitution σ such that $t_1\sigma =_I f(t_1,t_2)$, $t'_1\sigma =_I f(t_1,t_2)$ and $t'_2\sigma =_I f(t_1,t_2)$. This is equivalent to the case when we reformulate the statement we are proving as $t'_1 \not\prec_I f(t_1,t_2)$. This proves the case when the list of non-terminals in $x \to t^*(\bar{x})$ is empty and thus provides a second base case ($\mathbf{BC}_2$).

Now let us assume that the $t_1 \not\prec_I f(t_1,t_2)$ holds when the longest derivation in the tree grammar starting from $x \to t^*(\bar{x})$ is of length at most k. We show that the statements hold if the longest derivation has length at most $k+1$. We refer to this induction hypothesis as ($\mathbf{IH}_3$). Now once again we have to consider the number of idempotent symbols, but this time the number of non-terminals occurring in $t^*(\bar{x})$ is non-zero.

Once again, let us consider the case when $t^*(\bar{x})$ does not contain idempotent function symbols. This implies that their construction is syntactic and we can go directly to the non-terminals $\bar{x}$ and look at the rules containing them on the left hand side. Let $x_i \leftarrow r_i^*(\bar{z})$, where $x_i \in \bar{x}$, be such a rule. For each rule of this form we can add a new rule to the grammar $x \to t^*(x_1, \cdots, r^*(\bar{z}), x_{i+1}, \cdots, x_w)$ where $r^*(\bar{z})$ replaces the proper non-terminal.

Once this is done for all rules $x_i \leftarrow r_i^*(\bar{z})$ we remove $x \rightarrow t^*(\bar{x})$ from the grammar. This reduces the path length but does not change the language of the grammar. Thus, by ($\mathbf{IH}_3$) the statement holds in this case and provides us with ($\mathbf{BC}_3$).

We now assume $t_1 \not\prec_I f(t_1, t_2)$ when the number of idempotent symbols occurring in $t^*(\bar{x})$ is at most m ($\mathbf{IH}_4$) and show it holds when number of idempotent symbols occurring is at most $m+1$. This is exactly them same as the proof of ($\mathbf{BC}_2$). This ends the proof of case 1.

Case 2: For some $n + 1 > m > 0$, $\exists p$ s.t. $r|_p = t_1$ and $r \in L_m(G(s,t))$. Let us assume that other then grammar rules of the form $x \rightarrow g(x,x)$ where g is an idempotent function symbol, the rules used for constructing t_1 are of the form $x \rightarrow t^*(\bar{x})$ where $\bar{x}$ is an empty set of non-terminals. If t_1 is constructed from a single rule of the form $t^*(\bar{x})$ then by our idempotent equational theory is equivalent to no applications of duplication rules, i.e. $R \setminus R_b$, at all and is considered by Case 1. Thus, we assume that at least two rules of the form $x \rightarrow t^*(\bar{x})$ are used for the construction of t_1. Furthermore we assume that none of the $t^*(\bar{x})$ terms contain idempotent function symbols.

By our assumptions so far, if p' is a position of t_1 and $k \geq 1$ s.t. $t_1|_{p'} = g(s_1, \cdots, s_k)$ and there exists $s_i = x$, where x is a variable, then g must be a non-idempotent function symbol. if g where idempotent then $s_i \preceq s_{((i+1) \mod 2)}$ and would have been removed during simplification of $G_b(s,t)$. This implies that the immediate scope of any variable occurring in t_1 is not idempotent and that the argument from ($\mathbf{BC}_2$) concerning non idempotent symbols can be applied.

Let us assume that $t_1 \not\prec_I f(t_1, t_2)$ holds when the rules of the form $x \rightarrow t^*(\bar{x})$ used to construct t_1 contains at most q idempotent function symbols. We show that the statement also holds when $x \rightarrow t^*(\bar{x})$ contains at most $q+1$ idempotent function symbols ($\mathbf{IH}_5$). This is essentially the same argument perform for ($\mathbf{IH}_2$) except we have the additional condition that we have to differentiate between idempotent function symbols which occur as part of the rules $x \rightarrow t^*(\bar{x})$ and those which occur as part of $R \setminus R_b$. Note that $q+1$ concerns the former idempotent symbols ($\mathbf{BC}_4$). This requires the same argument used for ($\mathbf{BC}_2$).

Now we have to repeat the argument used in Case 1 concerning longest derivation in the tree grammar ignoring the duplication rules of $R \setminus R_b$. We are referring to the argument proving ($\mathbf{BC}_3$), and thus proving case 2.

Thus we have shown $t_1 \not\prec_I f(t_1, t_2)$. □

Theorem 16. *Let $s \triangleq t$ be an AUP over a signature Σ which may contains idempotent function symbols and $G(s,t)$ be the grammar computed by* I-Gen. *Then for all $n \geq 0$, $L_n(G(s,t))$ is minimal.*

Proof. We consider the minimality of $L_0(G(s,t))$ as a base case ($\mathbf{BC}_1$). Let us assume as an induction hypothesis that the theorem holds for $L_n(G(s,t))$ and show that it holds for $L_{n+1}(G(s,t))$. We will refer to this induction hypothesis as ($\mathbf{IH}_1$).

Since $L_n(G(s,t))$ is minimal, proving minimality of $L_{n+1}(G(s,t))$ means that we should prove the following three propositions:

P1. For all $s_1 \in L_{n+1}(G(s,t)) \setminus L_n(G(s,t))$ and $s_2 \in L_n(G(s,t))$, it is not the case that $s_2 \preceq_I s_1$.

P2. For all $s_1 \in L_{n+1}(G(s,t)) \setminus L_n(G(s,t))$ and $s_2 \in L_n(G(s,t))$, it is not the case that $s_1 \preceq_I s_2$.

P3. For all $s_1, s_2 \in L_{n+1}(G(s,t)) \setminus L_n(G(s,t))$, if $s_1 \neq s_2$, then $s_1 \not\preceq_I s_2$.

Proving P1. In order to prove **P1** we perform induction over the minimum $0 \leq k$ s.t. $s_2 \in L_k(G(s,t))$. The following assertions play an integral role in the majority of the cases outlined below:

(a) There exists positions p, q s.t. $s_2|_q\sigma = f(t_1,t_2)$ where $t_1, t_2 \in L_n(G(s,t))$, $s_1|_p = f(t_1,t_2)$ and at least one of t_1 and t_2 is in the language $L_n(G(s,t)) \setminus L_{n-1}(G(s,t))$. The existence of such positions is implied by the construction of s_1 from a term $t \in L_n(G(s,t)) \setminus L_{n-1}(G(s,t))$ by replacing the term at position p by $f(t_1,t_2)$

(b) There exists positions l_1, l_2 such that $f(t_1,t_2)$, t_1, and t_2 are generalizations of $s|_{l_1} \triangleq t|_{l_2}$. This is a consequence of grammar construction and the definition of our duplication rules.

Case 1: ($\mathbf{BC}_2$) We assume that $k = 0$ and thus, $s_2 \in L_0(G(s,t))$. Even under this restriction we need to perform an induction on the structure of $s_2|_q$.

Case:($\mathbf{BC}_3$) We assume $s_2|_q = x$ where x is a variable. A consequence of this assumption is $s_2\{x \leftarrow f(t_1,t_2)\} = s_1$. By assertion (a), $s_1 = t[f(t_1,t_2)]_p$ and $t[t_1]_p, t[t_2]_p \in L_n(G(s,t))$. This implies that either $s_2\{x \leftarrow t_1\} = t[t_1]_p$ or $s_2\{x \leftarrow t_2\} = t[t_2]_p$ contradicting the minimality of $L_n(G(s,t))$ assumed by ($\mathbf{IH}_1$). Therefore, $s_2|_q$ cannot be a variable nor a term of depth 1.

Case:($\mathbf{IH}_3$) We assume that $s_2|_q$ cannot be a term of depth n and show that it cannot be a term of depth $n+1$. By construction, $s_2|_q \neq g(w_1, \cdots, w_s)$ for a syntactic function symbol g. Therefore, we only consider the following cases:

Case 1:($\mathbf{IH}_3$) Let $s_2|_q = f(r_1,r_2)$. This implies that $r_1\sigma = f(t_1,t_2)$ and $r_2\sigma = f(t_1,t_2)$ or $r_1\sigma = t_1$ and $r_2\sigma = t_2$. In the former case $s_2|_q$ can be replaced by either r_1 or r_2 which cannot be the case by ($\mathbf{IH}_3$). In the latter case, r_1 and r_2 must also be solutions to $s|_{l_1} \triangleq t|_{l_2}$, but this would implies that $L_n(G(s|_{l_1}, t|_{l_2}))$ is not minimal and thus results in a contradiction with ($\mathbf{IH}_1$).

Case 2:($\mathbf{IH}_3$) Let $s_2|_q = g(r_1,r_2)$, for $g \neq f$. This implies that $r_1\sigma = f(t_1,t_2)$ and $r_2\sigma = f(t_1,t_2)$. In this case $s_2|_q$ can be replaced by either r_1 or r_2 and by ($\mathbf{IH}_3$) the statement holds.

Case:($\mathbf{IH}_2$) We assume, for $s_2 \in L_k(G(s,t)) \setminus L_{k-1}(G(s,t))$, s.t. $0 < k < n$, that no position in s_2 can result in $f(t_1,t_2)$ after applying σ. We show that the same holds for $k = n$. It must be the case that $s_2|_q = f(r_1,r_2)$ for idempotent function symbol f and $r_1, r_2 \in L_{n-1}(G(s,t))$ and at least one of r_1, r_2 is in $L_n(G(s,t)) \setminus L_{n-1}(G(s,t))$ or else the case is covered by ($\mathbf{IH}_2$). Thus as before $r_1\sigma = f(t_1,t_2)$ and $r_2\sigma = f(t_1,t_2)$ or $r_1\sigma = t_1$ and $r_2\sigma = t_2$. The former is an instance of ($\mathbf{IH}_3$) and the latter contradicts ($\mathbf{IH}_1$).

The only case we have not considered is $t_1 \prec_I f(t_1,t_2)$ or $t_2 \prec_I f(t_1,t_2)$. However, this is not possible by Lemma 15.

Proving P2. Let us consider a position p s.t. $s_1|_p = f(t_1, t_2)$, where f is idempotent and either $dra(t_1) = n$ and/or $dra(t_2) = n$, w.l.o.g we assume t_1. Furthermore, let σ be the substitution such that $s_1\sigma = s_2$. Let us now consider positions l_1, l_2 such that $f(t_1, t_2)$, t_1, and t_2 are generalizations of $s|_{l_1} \triangleq t|_{l_2}$. We now consider the sub-grammar of $G(s, t)$, namely $G(s|_{l_1}, t|_{l_2})$. If $t_1\sigma$ and/or $t_2\sigma$ are generalizations of $s|_{l_1} \triangleq t|_{l_2}$ then we contradict the minimality of $L_n(G(s|_{l_1}, t|_{l_2}))$. Otherwise if neither is a generalization then $f(t_1\sigma, t_2\sigma)$ cannot be a generalization by Lemma 14.

Proving P3. Let $s_1 = f(r_1, r_2)$ and assume by contradiction that $s_1 \preceq_I s_2$. Let σ be a substitution such that $f(r_1\sigma, r_2\sigma) \approx_I s_2$. We proceed by case distinction on s_2.

Case 1: s_2 has the form $f(t_1, t_2)$. Then we have two subcases:

- Subcase 1. $r_i\sigma \approx_I t_i$, $i = 1, 2$. But it contradicts the minimality of $L_n(G(s, t))$, because $r_1, r_2, t_1, t_2 \in L_n(G(s, t))$.
- Subcase 2. $r_1\sigma = r_2\sigma \approx_I f(t_1, t_2) = s_2$. But since $r_1, r_2 \in L_n(G(s, t))$, $s_2 \in L_{n+1}(G(s, t)) \setminus L_n(G(s, t))$, we get a contradiction with the already proved **P1**.

Case 2: s_2 has the form $g(t_1, t_2)$, $g \neq f$. The only possibility is $r_1\sigma = r_2\sigma \approx_I g(t_1, t_2)$, and the argument is like in Subcase 2 above.

It proves **P3**.

□

A simple corollary of Theorem 16 is that for all $n, m \geq 0$ and $n \leq m$, if $r \in L_n(G(s, t))$ then $r \in L_m(G(s, t))$. Essentially, the size of the languages is strictly increasing, i.e. $L_n(G(s, t)) < L_{n+1}(G(s, t))$. The result is the following theorem

Theorem 17. *$L(G(s, t))$ is a minimal complete set of I-generalizations of s and t.*

Proof. Essentially $nf_I(L(G(s, t))) \equiv \bigcup_{n=0}^{\infty} nf_I(L_n(G(s, t)))$ where each $L_n(G(s, t))$ is minimal complete with respect to a restriction on the application of the grammar rules. The infinite union of the languages is unrestricted. □

Corollary 18. *Idempotent generalization is infinitary.*

Proof. Theorem 17 says that the minimal complete set of idempotent generalizations of two terms always exists. The example in Section 3 shows that it is infinitary. □

6. Conclusion

The main contributions of this paper are twofold: first, we show that anti-unification with a single idempotent function symbol is at least infinitary. It was known from Pottier (1989) that the same holds for anti-unification with two idempotent symbols. Therefore, our result has a practical implication: If it were not true (i.e., if anti-unification with a single idempotent symbol were unitary or finitary), one could not hope to get a general combination method for anti-unification in disjoint theories, because finitary algorithms can not be combined into an infinitary procedure. Now, when this potential obstacle is removed (at least for the idempotent case), it makes sense to look into the details of the combination of anti-unification algorithms, and it is a part of our future work.

The second contribution is the algorithm for solving idempotent anti-unification problems. Although the solution sets of those problems might be infinite, it turns out that they form regular tree languages, and our algorithm computes their finite representation in the form of regular tree grammars. It should be remarked that the languages generated by those grammars are not only complete, but also minimal sets of generalizations. The algorithm does not depend on the number of different idempotent symbols in the theory. This result implies that idempotent anti-unification is infinitary for any number of idempotent function symbols.

References

Alpuente, M., Escobar, S., Espert, J., Meseguer, J., 2014. A modular order-sorted equational generalization algorithm. Inf. Comput. 235, 98–136.
URL https://doi.org/10.1016/j.ic.2014.01.006

Baader, F., 1986. The theory of idempotent semigroups is of unification type zero. J. Autom. Reasoning 2 (3), 283–286.
URL https://doi.org/10.1007/BF02328451

Baader, F., 1991. Unification, weak unification, upper bound, lower bound, and generalization problems. In: Book, R. V. (Ed.), Rewriting Techniques and Applications, 4th International Conference, RTA-91, Como, Italy, April 10-12, 1991, Proceedings. Vol. 488 of Lecture Notes in Computer Science. Springer, pp. 86–97.
URL https://doi.org/10.1007/3-540-53904-2_88

Baader, F., Büttner, W., 1988. Unification in commutative idempotent monoids. Theor. Comput. Sci. 56, 345–353.
URL https://doi.org/10.1016/0304-3975(88)90140-5

Baader, F., Schulz, K. U., 1996. Unification in the union of disjoint equational theories: Combining decision procedures. J. Symb. Comput. 21 (2), 211–243.
URL https://doi.org/10.1006/jsco.1996.0009

Baader, F., Snyder, W., 2001. Unification theory. In: Robinson, J. A., Voronkov, A. (Eds.), Handbook of Automated Reasoning (in 2 volumes). Elsevier and MIT Press, pp. 445–532.

Baumgartner, A., 2015. Anti-unification algorithms: Design, analysis, and implementation. Ph.D. thesis, Johannes Kepler University Linz, available from http://www.risc.jku.at/publications/download/risc_5180/phd-thesis.pdf.

Baumgartner, A., Kutsia, T., 2017. Unranked second-order anti-unification. Inf. Comput. 255, 262–286.
URL https://doi.org/10.1016/j.ic.2017.01.005

Baumgartner, A., Kutsia, T., Levy, J., Villaret, M., 2013. A variant of higher-order anti-unification. In: van Raamsdonk, F. (Ed.), 24th International Conference on Rewriting Techniques and Applications, RTA 2013, June 24-26, 2013, Eindhoven, The Netherlands. Vol. 21 of LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 113–127.
URL https://doi.org/10.4230/LIPIcs.RTA.2013.113

Baumgartner, A., Kutsia, T., Levy, J., Villaret, M., 2015. Nominal anti-unification. In: Fernández, M. (Ed.), RTA 2015, June 29 to July 1, 2015, Warsaw, Poland. Vol. 36 of LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 57–73.
URL https://doi.org/10.4230/LIPIcs.RTA.2015.57

Baumgartner, A., Kutsia, T., Levy, J., Villaret, M., 2017. Higher-order pattern anti-unification in linear time. J. Autom. Reasoning 58 (2), 293–310.
URL https://doi.org/10.1007/s10817-016-9383-3

Biere, A., 1993. Normalisation, unification and generalisation in free monoids. Master's thesis, University of Karlsruhe, (in German).

Burghardt, J., 2005. E-generalization using grammars. Artif. Intell. 165 (1), 1–35.
URL https://doi.org/10.1016/j.artint.2005.01.008

Burghardt, J., Heinz, B., 2014. Implementing anti-unification modulo equational theory. CoRR abs/1404.0953.
URL http://arxiv.org/abs/1404.0953

Cerna, D. M., Kutsia, T., 2018. Higher-order equational pattern anti-unification. In: Kirchner, H. (Ed.), 3rd International Conference on Formal Structures for Computation and Deduction, FSCD 2018, July 9-12, 2018, Oxford, UK. Vol. 108 of LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 12:1–12:17.
URL https://doi.org/10.4230/LIPIcs.FSCD.2018.12

Eberhard, S., Hetzl, S., 2015. Inductive theorem proving based on tree grammars. Ann. Pure Appl. Logic 166 (6), 665–700.
URL https://doi.org/10.1016/j.apal.2015.01.002

Heinz, B., 1995. Anti-unifikation modulo gleichungstheorie und deren anwendung zur lemmagenerierung. Ph.D. thesis, TU Berlin.

Herold, A., 1987. Narrowing techniques applied to idempotent unification. In: Morik, K. (Ed.), GWAI-87, 11th German Workshop on Artificial Intelligence, Geseke, Germany, September 28 - October 2, 1987, Proceedings. Vol. 152 of Informatik-Fachberichte. Springer, pp. 231–240.
URL https://doi.org/10.1007/978-3-642-73005-4_25

Hetzl, S., Leitsch, A., Reis, G., Tapolczai, J., Weller, D., 2014. Introducing quantified cuts in logic with equality. In: IJCAR 2014, Vienna, Austria, July 19-22, 2014. Proceedings. pp. 240–254.

Hetzl, S., Leitsch, A., Weller, D., 2012. Towards algorithmic cut-introduction. In: LPAR-18, Mérida, Venezuela, March 11-15, 2012. Proceedings. pp. 228–242.

Hullot, J., 1980. Canonical forms and unification. In: Bibel, W., Kowalski, R. A. (Eds.), 5th Conference on Automated Deduction, Les Arcs, France, July 8-11, 1980, Proceedings. Vol. 87 of Lecture Notes in Computer Science. Springer, pp. 318–334.
URL https://doi.org/10.1007/3-540-10009-1_25

Konev, B., Kutsia, T., 2016. Anti-unification of concepts in description logic EL. In: Baral, C., Delgrande, J. P., Wolter, F. (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016. AAAI Press, pp. 227–236.
URL http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12880

Kühner, S., Mathis, C., Raulefs, P., Siekmann, J. H., 1977. Unification of idempotent functions. In: Reddy, R. (Ed.), Proceedings of the 5th International Joint Conference on Artificial Intelligence. Cambridge, MA, USA, August 22-25, 1977. William Kaufmann, p. 528.
URL http://ijcai.org/Proceedings/77-1/Papers/092.pdf

Kutsia, T., Levy, J., Villaret, M., 2014. Anti-unification for unranked terms and hedges. J. Autom. Reasoning 52 (2), 155–190.

Leitsch, A., Peltier, N., Weller, D., 2017. CERES for first-order schemata. J. Log. Comput. 27 (7), 1897–1954.
URL https://doi.org/10.1093/logcom/exx003

Livesey, M., Siekmann, J., 1976. Unification of sets and multisets. Internal report MEMO SEKI-76-II, Universität Karlsruhe.

Livesey, M., Siekmann, J., Szabó, P., Unvericht, E., 1979. Unification problems for combinations of associativity, commutativity, distributivity and idempotence axioms. In: Joyner, W. (Ed.), Proceedings of the 4th Workshop on Automated Deduction, CADE-4. University of Texas at Austin, USA.

Miller, D., 1991. A logic programming language with lambda-abstraction, function variables, and simple unification. J. Log. Comput. 1 (4), 497–536.
URL https://doi.org/10.1093/logcom/1.4.497

Plotkin, G. D., 1970. A note on inductive generalization. Machine Intell. 5 (1), 153–163.

Plotkin, G. D., 1972. Building in equational theories. Machine Intell. 7, 7390.

Pottier, L., 1989. Generalisation de termes en theorie equationnelle. Cas associatif-commutatif. Rapports de Recherche 1056, INRIA Sophia Antipolis.

Raulefs, P., Siekmann, J., 1978. Unification of idempotent functions. MEMO SEKI-78-II-KL, Universität Karlsruhe.

Reynolds, J. C., 1970. Transformational systems and the algebraic structure of atomic formulas. Machine Intel. 5 (1), 135–151.

Robinson, J. A., 1965. A machine-oriented logic based on the resolution principle. J. ACM 12 (1), 23–41.
URL http://doi.acm.org/10.1145/321250.321253

Schmidt-Schauß, M., 1986. Unification properties of idempotent semigroups. SEKI Report SR-86-07, Universität Kaiserslautern.

Schmidt-Schauß, M., 1986. Unification under associativity and idempotence is of type nullary. J. Autom. Reasoning 2 (3), 277–281.
URL https://doi.org/10.1007/BF02328450

Siekmann, J., 1975. String-unification. Internal Report Memo CSM-7, Essex University.

Siekmann, J., 1978. Unification and matching problems. Memo csa-4-78, Essex University.

Siekmann, J. H., 1989. Unification theory. J. Symb. Comput. 7 (3/4), 207–274.
URL https://doi.org/10.1016/S0747-7171(89)80012-4

Urban, C., Pitts, A. M., Gabbay, M., 2004. Nominal unification. Theor. Comput. Sci. 323 (1-3), 473–497.
URL https://doi.org/10.1016/j.tcs.2004.06.016

Yamamoto, A., Ito, K., Ishino, A., Arimura, H., 2001. Modelling semi-structured documents with hedges for deduction and induction. In: Rouveirol, C., Sebag, M. (Eds.), ILP. Vol. 2157 of Lecture Notes in Computer Science. Springer, pp. 240–247.