Algebraic anti-unification

Christian Antic

Journal of Artificial Intelligence Research 1 (submitted) Submitted ; published

Algebraic anti-unification

Christian Antić CHRISTIAN.ANTIC@ICLOUD.COM *Vienna University of Technology*
Vienna, Austria

Abstract

Abstraction is key to human and artificial intelligence as it allows one to see common structure in otherwise distinct objects or situations and as such it is a key element for generality in AI. Anti-unification (or generalization) is *the* part of theoretical computer science and AI studying abstraction. It has been successfully applied to various AI-related problems, most importantly inductive logic programming. Up to this date, anti-unification is studied from a syntactic perspective in the literature. The purpose of this paper is to *initiate* an *algebraic* (i.e. semantic) theory of anti-unification within general algebras. This is motivated by recent applications to analogical reasoning in the form of similarity and analogical proportions.

1. Introduction

Abstraction is key to human and artificial intelligence (AI) as it allows one to see common structure in otherwise distinct objects or situations (Giunchiglia & Walsh, 1992; Saitta & Zucker, 2013) and as such it is a key element for generality in computer science and artificial intelligence (McCarthy, 1987; Kramer, 2007). It has been studied in the fields of theorem proving (e.g. Plaisted, 1981) and knowledge representation and reasoning (KR&R) by a number of authors (e.g. Knoblock, 1994; Sacerdoti, 1974), and it has recently gained momentum in answer set programming (Saribatur & Eiter, 2020; Saribatur, Eiter, & Schüller, 2021), one of the most prominent formalisms in the field of KR&R (see e.g. Brewka, Eiter, & Truszczynski, 2011; Lifschitz, 2019). Saribatur et al. (2021) contain a very rich bibliography for readers interested in the literature on abstraction.

Anti-unification (or generalization) (cf. Cerna & Kutsia, 2023) is *the* field of mathematical logic and theoretical computer science studying abstraction. It is the "dual" operation to the well-studied unification operation (cf. Baader & Snyder, 2001). More formally, given two terms s and t, unification searches for a substitution — the *most general unifier* — σ satisfying $s\sigma = t\sigma$, whereas syntactic anti-unification searches for the *least general generalization* u such that $s = u\sigma$ and $t = u\theta$, for some substitutions σ, θ. Notice the difference between the two operations: while unification computes a substitution, anti-unification computes a term. Therefore saying that the two operations are "dual" — as it is often done in the literature — is a bit misleading.

Syntactic anti-unification has been introduced by Plotkin (1970) and Reynolds (1970), and it has found numerous applications in theoretical computer science and artificial intelligence, as for example in inductive logic programming (Muggleton, 1991) (cf. Cropper & Morel, 2021; Cropper, 2022), programming by example (Gulwani, 2016), library learning and compression (Cao et al., 2023), and, in the form of E-generalization (i.e. anti-unification modulo theory) (Heinz, 1995; Burghardt, 2005), and in analogy-making (Weller & Schmid, 2007; Schmidt et al., 2014) (for further applications, see e.g. Barwell et al., 2018; de Sousa et al., 2021; Vanhoof & Yernaux, 2019).

The **purpose of this paper** is to *initiate* an algebraic theory of anti-unification within general algebras. More formally, given two algebras $\mathfrak{A}$ and $\mathfrak{B}$ in the sense of universal algebra (cf. Burris & Sankappanavar, 2000, §II), we shall define the set of ***minimally general generalizations*** (or ***mggs***) of two *elements* of algebras (instead of terms). That is, given a in $\mathfrak{A}$ and b in $\mathfrak{B}$, the set $a \Uparrow_{\mathfrak{AB}} b$ of mggs

will consist of all terms s such that a and b are within the range of the term function induced by s in $\mathfrak{A}$ and $\mathfrak{B}$, respectively, and s is minimal with respect to a suitable algebraic generalization ordering (see §3). Notice that this operation has *no* "dual" in the theory of unification since it makes no sense to try to "unify" two elements (constant symbols) a and b by finding a substitution σ such that $a\sigma = b\sigma$ (which holds iff $a = b$).

The initial **motivation** for studying algebraic anti-unification as proposed in this paper are two recent applications to AI, which we shall now briefly recall:

Similarity Detecting and exploiting similarities between seemingly distant objects is at the core of artificial general intelligence utilized for example in analogical transfer (Badra, Sedki, & Ugon, 2018; Badra & Lesot, 2023). The author has recently introduced an abstract algebraic notion of similarity in the general setting of universal algebra, the same mathematical context underlying this paper (Antić, 2023a). There, the set $\uparrow_{\mathfrak{A}} a$ of all generalizations of an element a in $\mathfrak{A}$ naturally occurs since similarity is roughly defined as follows: two elements a in $\mathfrak{A}$ and b in $\mathfrak{B}$ are called ***similar*** in $\mathfrak{AB}$ iff either $(\uparrow_{\mathfrak{A}} a) \cup (\uparrow_{\mathfrak{B}} b)$ consists only of trivial generalizations generalizing all elements of $\mathfrak{A}$ and $\mathfrak{B}$; or $a \uparrow_{\mathfrak{AB}} b$ is $\subseteq$-maximal with respect to a and b (separately). The intuition here is that generalizations in $\uparrow_{\mathfrak{A}} a$ encode *properties* of a; for example, the term $2x$ is a generalization of a natural number $a \in \mathbb{N}$ with respect to multiplication — that is, $2x \in \uparrow_{(\mathbb{N},\cdot,\mathbb{N})} a$ — iff a is even. While the role of the set of *all* generalizations of elements is immanent from the definition of similarity, the role of the set of *minimally general generalizations* as defined here is more mysterious and the content of ongoing research.

Analogical proportions Analogical proportions are expressions of the form "a is to b what c is to d" — written $a : b :: c : d$ — at the core of analogical reasoning with numerous applications to artificial intelligence such as computational linguistics (e.g. Lepage, 1998, 2001, 2003), image processing (e.g. Lepage, 2014), recommender systems (e.g. Hug et al., 2019), and program synthesis (Antić, 2023c), to name a few (cf. Prade & Richard, 2021). The author has recently introduced an abstract algebraic framework of analogical proportions in the general setting of universal algebra, the same setting as the notion of algebraic anti-unification of this paper (Antić, 2022). It is formulated in terms of arrow proportions of the form "a transforms into b as c transforms into d" — written $a \to b :\cdot c \to d$ — and justifications of the form $s \to t$ generalizing the arrows $a \to b$ and $c \to d$. Thus, the sets of generalizations $\uparrow_{\mathfrak{A}} (a \to b)$ and $\uparrow_{\mathfrak{B}} (c \to d)$ naturally occur in the framework and its role is evident from the definition of proportions. However, the role of the set of *minimally general generalizations* is more mysterious and the content of ongoing research — it has already been shown by Antić (2023d), however, that least general generalizations are key to term proportions in free term algebras which is a strong indication that algebraic anti-unification as proposed in this paper ought to play a key role for analogical proportions in the general setting (the role of anti-unification for analogical proportions has been recognized for E-anti-unification in other frameworks of analogical proportions by Weller and Schmid (2007)).

Antić (2023c) has recently studied (directed) ***logic program proportions*** of the form $P \to Q :\cdot R \to S$ for automated logic programming. In the process, so-called ***logic program forms*** (Antić, 2023b) where introduced as proper generalizations of logic programs. Given two programs P and R, computing the all common forms $P \uparrow R$ and the minimally general forms $P \Uparrow R$ appears challenging given the rich algebraic structure of logic programs and thus challenging. As in the abstract setting of analogical proportions above, the role of $\uparrow_{\mathfrak{P}} (P \to Q)$ is evident

from the definition of logic program proportions, whereas the role of minimally general forms is the content of ongoing research.

We do not claim to obtain any deep results — the purpose of this paper is rather to *introduce* the framework of algebraic anti-unification and to *initiate* its study. Moreover, We focus here primarily on **foundational issues**, not on applications (but see the potential applications to similarity and analogical proportions briefly discusses above).

Algebraic anti-unification as proposed in this paper is related to *E-generalization* or *anti-unification modulo equational theory* (Burghardt, 2005). In fact, if the underlying algebra has an equational axiomatization E, then algebraic and E-generalization are two sides of the same coin. This line of work is not pursued in this paper and therefore remains an interesting line for future research.

2. Preliminaries

We expect the reader to be fluent in basic universal algebra as it is presented for example in Burris and Sankappanavar (2000, §II).

A ***language*** L of algebras is a set of ***function symbols***[1] together with a ***rank function*** $r : L \to \mathbb{N}$, and a denumerable set X of ***variables*** distinct from L. Terms are formed as usual from variables and function symbols. The set of variables occurring in a term s are denoted by Xs and s has ***rank*** k iff $Xs = \{x_1, \ldots, x_k\}$. The rank of s is denoted by rs.

An ***L-algebra*** $\mathfrak{A}$ consists of a non-empty set A, the ***universe*** of $\mathfrak{A}$, and for each function symbol $f \in L$, a function $f^{\mathfrak{A}} : A^{rf} \to A$, the ***functions*** of $\mathfrak{A}$ (the ***distinguished elements*** of $\mathfrak{A}$ are the 0-ary functions). We will not distinguish between distinguished elements and their 0-ary function symbols. An algebra is ***injective*** iff each of its function is injective. Notice that every distinguished element a of $\mathfrak{A}$ is a 0-ary function $a : A^0 \to A$ which maps some single dummy element in A^0 to a and thus *is* injective.

Every term s induces a ***term function*** $s^{\mathfrak{A}}$ on $\mathfrak{A}$ in the usual way. In this paper, we shall not distinguish between terms inducing the same function on the underlying algebra, which is common practice in mathematics where one does usually not distinguish, for example, between the terms $2x^2$ and $(1 + 1)x^2$ in arithmetic.

Fact 1. *Every term function induced by injective functions is injective.*

The ***term algebra*** $\mathfrak{T}_{LX}$ over L and X is the algebra we obtain by interpreting each function symbol $f \in L$ by

$$f^{\mathfrak{T}_{LX}} : T_{LX}^{rf} \to T_{LX} : s_1 \ldots s_{rf} \mapsto f s_1 \ldots s_{rf}.$$

A ***substitution*** is a mapping $\sigma : X \to T_{LX}$, where $\sigma x := x$ for all but finitely many variables, extended from X to terms in T_{LX} inductively as usual, and we write $s\sigma$ for the application of σ to the term s. For two terms $s, t \in T_{LX}$, we define the ***syntactic generalization ordering*** by

$$s \lesssim t \quad :\Leftrightarrow \quad s = t\sigma, \quad \text{for some substitution } \sigma.$$

A ***homomorphism*** is a mapping $H : \mathfrak{A} \to \mathfrak{B}$ such that for any function symbol $f \in L$ and elements $a_1, \ldots, a_{rf} \in A$,

$$Hf^{\mathfrak{A}} a_1 \ldots a_{rf} = f^{\mathfrak{B}} Ha_1 \ldots Ha_{rf}.$$

An ***isomorphism*** is a bijective homomorphism.

1. We omit constant symbols since constants are identified with 0-ary functions.

3. Algebraic anti-unification

This is the main section of the paper. Here we shall introduce algebraic anti-unification and derive some elementary observations.

In the rest of the paper, let $\mathfrak{A}$ and $\mathfrak{B}$ be L-algebras over some joint language of algebras L, and let $\mathfrak{AB}$ be a pair of L-algebras. We will always write $\mathfrak{A}$ instead of $\mathfrak{AA}$.

Definition 2. Given an L-term s, define

$$\downarrow_{\mathfrak{A}} s := \left\{ s^{\mathfrak{A}}\mathbf{o} \in A \;\middle|\; \mathbf{o} \in A^{rs} \right\}.$$

In case $a \in \downarrow_{\mathfrak{A}} s$, we say that s is a ***generalization*** of a and a is an ***instance*** of s.

Fact 3. *Every distinguished element $a \in A$ is an instance of itself*[2] *since*

$$a \in \downarrow_{\mathfrak{A}} a = \{a\}. \tag{1}$$

Definition 4. Define the ***(semantic) generalization ordering*** for two L-terms s and t in $\mathfrak{AB}$ by

$$s \sqsubseteq_{\mathfrak{AB}} t \quad :\Leftrightarrow \quad \downarrow_{\mathfrak{A}} s \subseteq \downarrow_{\mathfrak{A}} t \quad \text{and} \quad \downarrow_{\mathfrak{B}} s \subseteq \downarrow_{\mathfrak{B}} t,$$

and

$$s \equiv_{\mathfrak{AB}} t \quad :\Leftrightarrow \quad s \sqsubseteq_{\mathfrak{AB}} t \quad \text{and} \quad t \sqsubseteq_{\mathfrak{AB}} s.$$

Example 5. $0x \equiv_{(\mathbb{N},\cdot,0)} 0$.

Fact 6. *The generalization ordering $\sqsubseteq_{\mathfrak{AB}}$ is a pre-order between L-terms, for any pair of L-algebras $\mathfrak{AB}$, that is, it is reflexive and transitive.*

Fact 7. *For any distinguished element $a \in A$, we have by (1):*

$$a \sqsubseteq_{\mathfrak{A}} s \quad \Leftrightarrow \quad a \in \downarrow_{\mathfrak{A}} s$$

and

$$s \sqsubseteq_{\mathfrak{A}} a \quad \Leftrightarrow \quad s \equiv_{\mathfrak{A}} a \quad \Leftrightarrow \quad s^{\mathfrak{A}} \text{ is a constant function with value } a.$$

For two distinguished elements $a, b \in A$, we have

$$a \sqsubseteq_{\mathfrak{A}} b \quad \Leftrightarrow \quad \downarrow_{\mathfrak{A}} a \subseteq \downarrow_{\mathfrak{A}} b \quad \Leftrightarrow \quad \{a\} \subseteq \{b\} \quad \Leftrightarrow \quad a = b \quad \Leftrightarrow \quad a \equiv_{\mathfrak{A}} b.$$

Remark 8. Notice that our semantic generalization ordering $\sqsubseteq$ differs from the usual one in *syntactic* anti-unification $\lesssim$ where a term s is said to be ***more general than*** a term t iff there is a substitution σ such that $t = s\sigma$.

Definition 9. For any element $a \in A$, we define

$$\uparrow_{\mathfrak{A}} a := \{s \in T_{LX} \mid a \in \downarrow_{\mathfrak{A}} s\},$$

extended to elements $a \in A$ and $b \in B$ by

$$a \uparrow_{\mathfrak{AB}} b := (\uparrow_{\mathfrak{A}} a) \cap (\uparrow_{\mathfrak{B}} b).$$

2. More precisely, of the constant symbol denoting it.

We shall now introduce the **main notion of the paper**:

Definition 10. Define the set of ***minimally general generalizations*** (or ***mggs***) of two elements $a \in A$ and $b \in B$ in $\mathfrak{AB}$ by

$$a \Uparrow_{\mathfrak{AB}} b := \min_{\sqsubseteq_{\mathfrak{AB}}}(a \uparrow_{\mathfrak{AB}} b),$$

In case $a \Uparrow_{\mathfrak{AB}} b = \{s\}$ contains a single generalization, s is called the ***least general generalization*** (or ***lgg***) of a and b in $\mathfrak{AB}$.

Here $\min_{\sqsubseteq_{\mathfrak{AB}}}$ is the function computing the *set* of minimal generalizations with respect to $\sqsubseteq_{\mathfrak{AB}}$ (not a single minimal generalization), which means that in case there are *no* minimal generalizations, the computed set is empty. Notice that in case $a \in A$ is a distinguished element, we always have[3]

$$a \in a \Uparrow_{\mathfrak{A}} a.$$

Example 11. Let $\mathfrak{BOOL} := (\{0,1\}, \vee, \neg, \{0,1\})$ be the 2-element boolean algebra with disjunction and negation (and therefore with all boolean functions) where both truth values are distinguished elements. Notice that terms in $T_{\{\vee,\neg,0,1\}}(X)$ are propositional formulas with variables from X in the usual sense. We have

$$\begin{aligned}
\uparrow_{\mathfrak{BOOL}} 0 &= \{s \in T_{\{\vee,\neg,0,1\}}(X) \mid s \text{ is falsifiable}\},\\
\uparrow_{\mathfrak{BOOL}} 1 &= \{s \in T_{\{\vee,\neg,0,1\}}(X) \mid s \text{ is satisfiable}\},\\
0 \uparrow_{\mathfrak{BOOL}} 1 &= \{s \in T_{\{\vee,\neg,0,1\}}(X) \mid s \text{ is satisfiable and falsifiable}\}.
\end{aligned}$$

We clearly have

$$0 \Uparrow_{\mathfrak{BOOL}} 1 = 0 \uparrow_{\mathfrak{BOOL}} 1.$$

4. Generalization type

The following definition is an adaptation of the nomenclature in Cerna and Kutsia (2023, Definition 5) (triviality is new):

Definition 12. The ***generalization type*** of a pair of L-algebras $\mathfrak{AB}$ is called

- ***nullary*** iff $a \Uparrow_{\mathfrak{AB}} b = \emptyset$, for some $a \in A$ and $b \in B$;
- ***unitary*** iff $|a \Uparrow_{\mathfrak{AB}} b| = 1$, for all $a \in A$ and $b \in B$;
- ***finitary*** iff
  - $|a \Uparrow_{\mathfrak{AB}} b| < \infty$, for all $a \in A$ and $b \in B$;
  - $|a \Uparrow_{\mathfrak{AB}} b| > 1$, for some $a \in A$ and $b \in B$;
- ***infinitary*** iff $|a \Uparrow_{\mathfrak{AB}} b| = \infty$, for some $a \in A$ and $b \in B$;
- ***trivial*** iff $a \Uparrow_{\mathfrak{AB}} b = T_{LX}$, for all $a \in A$ and $b \in B$.

3. More precisely, the constant symbol for the element a is in $a \Uparrow_{\mathfrak{A}} a$.

Fact 13. $\uparrow_{\mathfrak{A}} a = \{x\}$ *implies* $a \Uparrow_{\mathfrak{AB}} b = \{x\}$*, for all* $b \in B$*. Hence, algebras containing elements having only the trivial generalization* x *cannot be nullary.*

Fact 14. $a \uparrow_{\mathfrak{AB}} b = \{x\}$ *implies* $a \Uparrow_{\mathfrak{AB}} b = a \uparrow_{\mathfrak{AB}} b \neq \emptyset$*.*

Fact 15. *In any pair of algebras* $\mathfrak{AB}$ *with* $|A| = |B| = 1$*, we have* $a \Uparrow_{\mathfrak{AB}} b = a \uparrow_{\mathfrak{AB}} b \neq \emptyset$*. Hence, algebras consisting of a single element cannot be nullary.*

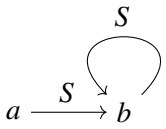
Proof. Every generalization $s \in a \uparrow_{\mathfrak{AB}} b$ has to satisfy $\downarrow_{\mathfrak{A}} s = \{a\}$ and $\downarrow_{\mathfrak{B}} s = \{b\}$, which means that there cannot be a generalization t such that $\downarrow_{\mathfrak{AB}} t \subsetneq \downarrow_{\mathfrak{AB}} s$. □

Fact 16. *The algebra* (A) *consisting only of its universe is unitary and trivial.*

Proof. A direct consequence of $a \Uparrow_{(A)} b = \{x\}$, for all $a, b \in A$ (notice that $a \Uparrow_{(A)} a \neq \{a\}$ since a is not a distinguished element of (A) and thus cannot be used to form terms). □

Fact 17. $s \in a \Uparrow_{\mathfrak{AB}} b$ *for any term* s *satisfying* $\downarrow_{\mathfrak{A}} s = \{a\}$ *and* $\downarrow_{\mathfrak{B}} s = \{b\}$*.*

Example 18. In the monounary algebra $\mathfrak{A} = (\{a, b\}, S)$ given by



we have

$$\uparrow_{\mathfrak{A}} a = \{x\}$$

and thus by Fact 13 we have

$$a \Uparrow_{\mathfrak{A}} b = \{x\}.$$

This shows that $\mathfrak{A}$ is unitary.

Fact 19. *The generalization type of* $(\mathbb{N}, S)$*, where* $S\,x := x + 1$ *is the successor function, is unitary by the forthcoming Corollary 35.*

Notice that in term algebras, syntactic and semantic generalization orderings (and thus anti-unification) coincide:

Lemma 20. *For any L-terms* $s, t \in T_{LX}$*, we have*

$$s \lesssim t \quad \Leftrightarrow \quad s \sqsubseteq_{\mathfrak{T}_{LX}} t.$$

Proof. We have the following equivalences:

$$\begin{aligned} s \lesssim t \quad &\Leftrightarrow \quad s = t\mathbf{o}, \quad \text{for some } \mathbf{o} \in T^{rt}_{LX} \\ &\Leftrightarrow \quad \downarrow_{\mathfrak{T}_{LX}} s \subseteq \downarrow_{\mathfrak{T}_{LX}} t \\ &\Leftrightarrow \quad s \sqsubseteq_{\mathfrak{T}_{LX}} t. \end{aligned}$$

□

Fact 21. *Term algebras are unitary.*

Proof. A direct consequence of Lemma 20 and the well-known fact that term algebras are unitary by Plotkin's (1970) and Reynolds's (1970) classical results, and their original algorithms (plus Huet's (1976) simple algorithm) can be used to compute the least general generalization of two terms. □

Theorem 22. *In every injective algebra $\mathfrak{A}$, we have $a \Uparrow_{\mathfrak{A}} b \neq \emptyset$, for all $a, b \in A$. Consequently, injective algebras cannot be nullary.*

Proof. We have $a \Uparrow_{\mathfrak{A}} b = \emptyset$ iff for each $s \in a \uparrow_{\mathfrak{A}} b$ there is some $t \in a \uparrow_{\mathfrak{A}} b$ such that $t \sqsubset s$, which is equivalent to $\downarrow_{\mathfrak{A}} t \subset \downarrow_{\mathfrak{A}} s$. Since $t^{\mathfrak{A}}$ is injective by assumption (Fact 1), we must have $t^{\mathfrak{A}}\mathbf{o} \neq t^{\mathfrak{A}}\mathbf{u}$ for every $\mathbf{o}, \mathbf{u} \in A^{rt}$, which means that

$$| \downarrow_{\mathfrak{A}} t | = |\{t^{\mathfrak{A}}\mathbf{o} \mid \mathbf{o} \in A^{rt}\}| = |A^{rt}|$$

and, analogously,

$$| \downarrow_{\mathfrak{A}} s | = |A^{rs}|.$$

Since $\downarrow_{\mathfrak{A}} t \subsetneq \downarrow_{\mathfrak{A}} s$, we have

$$| \downarrow_{\mathfrak{A}} t | < | \downarrow_{\mathfrak{A}} s |,$$

thus

$$|A^{rt}| < |A^{rs}|,$$

and hence

$$rt < rs.$$

Since the rank of every function is finite, we have $rs < \infty$ and thus there can be only finitely many t's with $rt < rs$. Hence, there must be some $t' \in a \uparrow b$ for which there can be no $t'' \in a \uparrow b$ with $t'' \sqsubset t'$ — we thus conclude $t' \in a \Uparrow_{\mathfrak{A}} b$ and $a \Uparrow_{\mathfrak{A}} b \neq \emptyset$. □

5. Characteristic generalizations

Anti-unifying an element with itself yields the set of minimal general generalizations of that element which, in a sense, "characterize" that element since generalizations encode properties of elements (for example, $2x$ is a genearlization of a natural number a iff a is even). This motivates the following definition:

Definition 23. Define the set of ***characteristic generalizations*** of $a \in A$ in $\mathfrak{A}$ by

$$\Uparrow_{\mathfrak{A}} a := \min_{\sqsubseteq_{\mathfrak{A}}} \uparrow_{\mathfrak{A}} a = a \Uparrow_{\mathfrak{A}} a.$$

Example 24. Recall the situation in Example 11. We have

$$\begin{aligned} \Uparrow_{\mathfrak{BOOL}} 0 &= \{s \in T_{\{\vee,\neg,0,1\}}(X) \mid s \text{ is unsatisfiable}\}, \\ \Uparrow_{\mathfrak{BOOL}} 1 &= \{s \in T_{\{\vee,\neg,0,1\}}(X) \mid s \text{ is valid}\}, \\ 0 \Uparrow_{\mathfrak{BOOL}} 1 &= \emptyset. \end{aligned}$$

Definition 25. We call a set G of generalizations a ***characteristic set of generalizations*** of a in $\mathfrak{A}$ iff

1. $G \subseteq \Uparrow_{\mathfrak{A}} a \in A$;
2. $G \not\subseteq \Uparrow_{\mathfrak{A}} b$ for all $b \neq a$.

In case $G = \{s\}$ is a singleton, we call s a ***characteristic generalization*** of a in $\mathfrak{A}$.

Example 26. $X \cup X^c$ is a characteristic generalization of U in $(2^U, \cup)$.

Example 27. $x + (-x)$ is a characteristic generalization of 0 in $(\mathbb{Z}, +)$.

6. Homomorphisms

In this section, we show that algebraic anti-unification is compatible with structure-preserving mappings.

Lemma 28 (Homomorphism Lemma)**.** *For any homomorphism $H : \mathfrak{A} \to \mathfrak{B}$, any L-term s, and any elements $a, b \in A$,*

$$H \downarrow_{\mathfrak{A}} s \subseteq \downarrow_{\mathfrak{B}} s, \tag{2}$$
$$a \uparrow_{\mathfrak{A}} b \subseteq Ha \uparrow_{\mathfrak{B}} Hb. \tag{3}$$

In case H is an isomorphism, we have

$$H \downarrow_{\mathfrak{A}} s = \downarrow_{\mathfrak{B}} s, \tag{4}$$
$$a \uparrow_{\mathfrak{A}} b = Ha \uparrow_{\mathfrak{B}} Hb. \tag{5}$$

Proof. Since H is a homomorphism by assumption, we have

$$H \downarrow_{\mathfrak{A}} s = \left\{ Hs^{\mathfrak{A}}\mathbf{o} \;\middle|\; \mathbf{o} \in A^{rs} \right\} = \left\{ s^{\mathfrak{B}}H\mathbf{o} \;\middle|\; \mathbf{o} \in A^{rs} \right\} \subseteq \left\{ s^{\mathfrak{B}}\mathbf{o} \;\middle|\; \mathbf{o} \in B^{rs} \right\} = \downarrow_{\mathfrak{B}} s.$$

In case H is an isomorphism, we have "=" instead of "$\subseteq$" in the above computation.

We shall now prove (3) by showing that every term $s \in a \uparrow_{\mathfrak{A}} b$ is in $Ha \uparrow_{\mathfrak{A}} Hb$. Since $s \in a \uparrow_{\mathfrak{A}} b$ holds by assumption, we have

$$a = s^{\mathfrak{A}}\mathbf{o} \quad \text{and} \quad b = s^{\mathfrak{A}}\mathbf{u},$$

for some $\mathbf{o}, \mathbf{u} \in A^{rs}$. Since H is a homomorphism, we thus have

$$Ha = Hs^{\mathfrak{A}}\mathbf{o} = s^{\mathfrak{B}}H\mathbf{o} \quad \text{and} \quad Hb = Hs^{\mathfrak{A}}\mathbf{u} = s^{\mathfrak{B}}H\mathbf{u},$$

which shows (3).

It remains to show that in case H is an isomorphism, we have

$$Ha \uparrow_{\mathfrak{B}} Hb \subseteq a \uparrow_{\mathfrak{A}} b.$$

For this, let

$$s \in Ha \uparrow_{\mathfrak{B}} Hb$$

be a generalization of Ha and Hb in $\mathfrak{B}$ which by definition means that there are some $\mathbf{o}, \mathbf{u} \in B^{rs}$ such that

$$Ha = s^{\mathfrak{B}}\mathbf{o},$$
$$Hb = s^{\mathfrak{B}}\mathbf{u}.$$

Since H is an isomorphism, its inverse H^{-1} is an isomorphism as well, which yields

$$a = H^{-1}s^{\mathfrak{B}}\mathbf{o} = s^{\mathfrak{A}}H^{-1}\mathbf{o},$$
$$b = H^{-1}s^{\mathfrak{B}}\mathbf{u} = s^{\mathfrak{A}}H^{-1}\mathbf{u}.$$

This shows

$$s \in a \uparrow_{\mathfrak{A}} b.$$

□

Theorem 29 (Isomorphism Theorem)**.** *For any isomorphism $H : \mathfrak{A} \to \mathfrak{B}$ and elements $a, b \in A$,*

$$a \Uparrow_{\mathfrak{A}} b = Ha \Uparrow_{\mathfrak{B}} Hb. \tag{6}$$

Proof. ($\subseteq$) We show that each

$$s \in a \Uparrow_{\mathfrak{A}} b \tag{7}$$

is contained in $Ha \Uparrow_{\mathfrak{B}} Hb$ for the isomorphism H. By Lemma 28, we have $a \uparrow_{\mathfrak{A}} b \subseteq Ha \uparrow_{\mathfrak{B}} Hb$. It thus remains to show that s is $\sqsubseteq_{\mathfrak{B}}$-minimal.

Suppose there is some $t \in Ha \uparrow_{\mathfrak{B}} Hb$ such that $t \sqsubset_{\mathfrak{B}} s$, that is,

$$\downarrow_{\mathfrak{B}} t \subsetneq \downarrow_{\mathfrak{B}} s. \tag{8}$$

Since $t \in Ha \uparrow_{\mathfrak{B}} Hb$, we have

$$Ha = t^{\mathfrak{B}}\mathbf{o},$$
$$Hb = t^{\mathfrak{B}}\mathbf{u},$$

for some $\mathbf{o}, \mathbf{u} \in B^{rt}$. Now since H is an isomorphism, its inverse H^{-1} is an isomorphism as well, and we thus have

$$a = H^{-1}t^{\mathfrak{B}}\mathbf{o} = t^{\mathfrak{A}}H^{-1}\mathbf{o}$$
$$b = H^{-1}t^{\mathfrak{B}}\mathbf{u} = t^{\mathfrak{A}}H^{-1}\mathbf{u},$$

which shows

$$t \in a \uparrow_{\mathfrak{A}} b.$$

Now we have by (4) and (8),

$$\downarrow_{\mathfrak{A}} t = H^{-1} \downarrow_{\mathfrak{B}} t \subsetneq H^{-1} \downarrow_{\mathfrak{A}} s = \downarrow_{\mathfrak{A}} s,$$

which is equivalent to

$$t \sqsubset_{\mathfrak{A}} s,$$

a contradiction to (7) and thus to the $\sqsubseteq_{\mathfrak{A}}$-minimality of s.

($\supseteq$) Analogous. □

7. The (k, ℓ)-fragments

Since computing the set of *all* generalizations is rather difficult in general, it is reasonable to study fragments of algebraic anti-unification. For this, we introduce in this section the (k, ℓ)-fragments:

Definition 30. Let $X_k := \{x_1, \ldots, x_k\}$, for some $k, \ell \in \mathbb{N} \cup \{\infty\}$ so that $X_\infty = X$. Define

$$\uparrow_{\mathfrak{A}}^{(k,\ell)} a := (\uparrow_{\mathfrak{A}} a) \cap \{s(x_1, \ldots, x_k) \in T_{LX_k} \mid \text{each of the } k \text{ variables in } X_k \text{ occurs at most } \ell \text{ times in } s\}.$$

We write k instead of (k, ∞) so that

$$\uparrow_{\mathfrak{A}}^{k} a = (\uparrow_{\mathfrak{A}} a) \cap T_{LX_k}.$$

The simplest fragment — namely, the $(1, 1)$-fragment — contains only ***monolinear generalizations*** containing exactly one occurrence of a single variable x. We denote the ***monolinear generalization*** and ***instantiation operations*** in $\mathfrak{A}$ by $\uparrow_{\mathfrak{A}}^{m}$ and $\downarrow_{\mathfrak{A},m}$, respectively, and the so-obtained anti-unification relation by $\Uparrow_{\mathfrak{A},m}$.

As a simple demonstration of fragments, we compute monolinear algebraic anti-unification in the set domain:

Proposition 31. *For any universe U and $A, B \subseteq U$, we have*

$$A \Uparrow^{m}_{(2^U,\cup,2^U)} B = \begin{cases} \{X \cup (A \cap B)\} & A \neq B, \\ \{A\} & A = B, \end{cases}$$

$$A \Uparrow^{m}_{(2^U,\cap,2^U)} B = \begin{cases} \{X \cap (A \cup B)\} & A \neq B, \\ \{A\} & A = B, \end{cases}$$

$$A \Uparrow^{m}_{(2^U,{}^c,2^U)} B = \begin{cases} \{X, X^c\} & A \neq B, \\ \{A\} & A = B. \end{cases}$$

Proof. Given two sets $C, D \subseteq U$, we define

$$[C, D] := \{E \subseteq U \mid C \subseteq E \subseteq D\}.$$

1. First, we work in $(2^U, \cup, 2^U)$ and omit the explicit reference to the algebra. We have

   $$A \uparrow^m B = \{X \cup C \mid \emptyset \subseteq C \subseteq A \cap B\} \cup \{A \mid \text{if } A = B\}$$

   and

   $$\downarrow_m (X \cup C) = [C, U].$$

   This implies

   $$X \cup C \sqsubseteq X \cup D \quad \Leftrightarrow \quad [C, U] \subseteq [D, U] \quad \Leftrightarrow \quad D \subseteq C.$$

   Hence

   $$A \Uparrow^m B = \begin{cases} \{X \cup (A \cap B)\} & A \neq B \\ \{A\} & A = B. \end{cases}$$

2. Second, we work in $(2^U, \cap, 2^U)$ and omit the explicit reference to the algebra. We have

$$A \uparrow^m B = \{X \cap C \mid A, B \subseteq C\} \cup \{A \mid \text{if } A = B\}$$

and

$$\downarrow_m (X \cap C) = [\emptyset, C].$$

This implies

$$X \cap C \sqsubseteq X \cap D \quad \Leftrightarrow \quad [\emptyset, C] \subseteq [\emptyset, D] \quad \Leftrightarrow \quad C \subseteq D.$$

Hence

$$A \Uparrow^m B = \begin{cases} \{X \cap (A \cup B)\} & A \neq B \\ \{A\} & A = B. \end{cases}$$

3. Third, we work in $(2^U, .^c, 2^U)$ and omit the explicit reference to the algebra. We have

$$A \uparrow^m B = \{X, X^c\} \cup \{A \mid \text{if } A = B\}$$

and

$$\downarrow_m X^c = 2^U = \downarrow X.$$

This implies

$$X^c \equiv X.$$

Hence

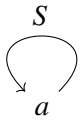
$$A \Uparrow^m B = \begin{cases} \{X, X^c\} & A \neq B \\ \{A\} & A = B. \end{cases}$$

□

8. Monounary algebras

In the rest of this section, let $\mathfrak{A} = (A, S)$ be a monounary algebra with $S : A \to A$ the only unary function (we can imagine S to be a generalized "successor" function).

Example 32. We show that the generalization type of the monounary algebra



is infinitary and trivial. We have

$$\uparrow_{\mathfrak{A}} a = \{S^m x \mid k \geq 0\}.$$

Since

$$\downarrow_{\mathfrak{A}} S^m x = \downarrow_{\mathfrak{A}} S^n x = \{a\}, \quad \text{for all } m, n \geq 0,$$

we have

$$S^m x \equiv_{\mathfrak{A}} S^n x, \quad \text{for all } m, n \geq 0,$$

and thus

$$a \Uparrow_{\mathfrak{A}} a = \{S^m x \mid k \geq 0\} = T_{\{S\}}(\{x\}).$$

This shows that the generalization type of $\mathfrak{A}$ is infinitary and trivial.

Example 33. We show that the generalization type of the monounary algebra

$$a \overset{S}{\longleftrightarrow} b$$

is infinitary and trivial. Since

$$\uparrow_{\mathfrak{B}} a = \{S^m x \mid k \geq 0\} = \uparrow_{\mathfrak{B}} b$$

and

$$\downarrow_{\mathfrak{B}} S^m x = \downarrow_{\mathfrak{B}} S^n x = \{a, b\}, \quad \text{for all } m, n \geq 0,$$

implies

$$S^m x \equiv_{\mathfrak{B}} S^n x, \quad \text{for all } m, n \geq 0,$$

and thus

$$a \Uparrow_{\mathfrak{B}} a = b \Uparrow_{\mathfrak{B}} b = a \Uparrow_{\mathfrak{B}} b = \{S^m x \mid k \geq 0\} = T_{\{S\}}(\{x\}).$$

We define

$$m(a) := \begin{cases} \max\{m \geq 0 \mid S^m x \in \uparrow_{\mathfrak{A}} a\} & \text{if the maximum exists,} \\ \infty & \text{otherwise,} \end{cases}$$

$$m(a, b) := \min(m(a), m(b)) \in \mathbb{N} \cup \{\infty\}.$$

Theorem 34. *Let $\mathfrak{A} = (A, S)$ be a monounary algebra. For any $a, b \in A$, we have*

$$a \Uparrow_{\mathfrak{A}} b = \begin{cases} \left\{S^{m(a,b)} x\right\} & m(a, b) < \infty \\ \emptyset & \textit{otherwise.} \end{cases}$$

Proof. Suppose $m(a, b) < \infty$, which means that either $m(a) < \infty$ or $m(b) < \infty$, that is, there is some maximal k such that $S^m x \in \uparrow_{\mathfrak{A}} a$ or $S^m x \in \uparrow_{\mathfrak{A}} b$. We then have

$$S^{k+\ell} x \notin (\uparrow_{\mathfrak{A}} a) \cap (\uparrow_{\mathfrak{A}} b) = a \uparrow b, \quad \text{for all } \ell \geq 1,$$

which implies $a \Uparrow_{\mathfrak{A}} b = \{S^m x\}$ — notice that $k = m(a, b)$ by definition.

Now suppose $m(a, b) = \infty$, which means that $m(a) = m(b) = \infty$. In that case, we clearly have $\max_{\leq}(a \uparrow_{\mathfrak{A}} b) = \emptyset$. □

Corollary 35. *Let $(\mathbb{N}, S)$ be the infinite monounary algebra where $S x := x + 1$ denotes the successor function. For any $a, b \in \mathbb{N}$, we have $a \Uparrow_{(\mathbb{N},S)} b = \left\{S^{\min(a,b)} x\right\}$. This means that the generalization type of $(\mathbb{N}, S)$ is unitary.*

9. Finite unary algebras aka semiautomata

In this section, we study algebraic anti-unification in finite unary algebras, which can be seen as semiautomata, and show that we can use well-known methods from the theory of finite automata to compute sets of (minimally general) generalizations.

In the rest of this section, let

$$\mathfrak{A} = (A, \Sigma := \{f_1, \ldots, f_n\}),$$

for some $n \geq 1$, be a finite unary algebra with finite universe A. We shall now recall that every such algebra is essentially a semiautomaton.

Recall that a (***finite deterministic***) ***semiautomaton*** (see e.g. Holcombe, 1982, §2.1) is a construct

$$\mathfrak{S} = (S, \Sigma, \delta),$$

where S is a finite set of ***states***, Σ is a finite ***input alphabet***, and $\delta : S \times \Sigma \to S$ is a ***transition function***. Every semiautomaton can be seen as a finite unary algebra in the following well-known way: every symbol $\sigma \in \Sigma$ induces a unary function $\sigma^{\mathfrak{S}} : S \to S$ via $\sigma^{\mathfrak{S}} := \delta(x, \sigma)$. We can now omit δ and define

$$\mathfrak{S}' := (S, \Sigma^{\mathfrak{S}} := \{\sigma^{\mathfrak{S}} \mid \sigma \in \Sigma\}).$$

It is immediate from the construction that $\mathfrak{S}$ and $\mathfrak{S}'$ represent essentially the same semiautomaton and that every semiautomaton can be represented in that way — the difference is that $\mathfrak{S}'$ is a finite unary algebra!

Recall that a (***finite deterministic***) ***automaton*** (see e.g. Sipser, 2013, §1.1) is a construct

$$\mathcal{A} := (Q, \Sigma, \delta, q_0, F),$$

where (Q, Σ, δ) is a semiautomaton, $q_0 \in Q$ is the ***initial state***, and $F \subseteq Q$ is a set of ***final states***. The ***behavior*** of $\mathcal{A}$ is given by

$$\|\mathcal{A}\| := \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\},$$

where $\delta^* : Q \times \Sigma^* \to Q$ is defined recursively as follows, for $q \in Q, a \in \Sigma, w \in \Sigma^*$:

$$\begin{aligned} \delta^*(q, \varepsilon) &:= q, \\ \delta^*(q, aw) &:= \delta^*(\delta(q, a), w). \end{aligned}$$

Notice that since automata are built from semiautomata by adding an initial state and a set of final states, and since every semiautomaton $\mathfrak{S} = (S, \Sigma, \delta)$ can be represented in the form of a finite unary algebra $\mathfrak{S}' = (S, \Sigma^{\mathfrak{S}})$ as above, we can reformulate every automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ as $\mathcal{A}' = (Q, \Sigma^{\mathcal{A}}, q_0, F)$, where $\Sigma^{\mathcal{A}} := \{\sigma^{\mathcal{A}} \mid \sigma \in \Sigma\}$ and $\sigma^{\mathcal{A}} := \delta(\,.\,, \sigma) : Q \to Q$. In other words, given a finite unary algebra (semiautomaton)

$$\mathfrak{A} = (A, \Sigma),$$

we can construct a finite automaton

$$\mathfrak{A}_{a \to F} = (A, \Sigma, a, F)$$

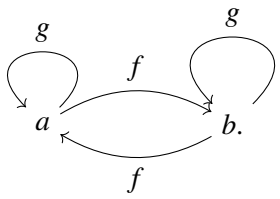
by designating a state $a \in A$ as the initial state, and by designating a set of states $F \subseteq A$ as final states.

We want to compute the set of generalizations $\uparrow_{\mathfrak{A}} a$. Notice that we can identify each term in $T_{\{f,g\}}(\{x\})$ with a word over the alphabet $\Sigma = \{f, g\}$: for example, the term $fgfx$ can be identified with the word $fgf \in \Sigma^*$ since the variable x contains no information. We denote the function induced by a word $w \in \Sigma^*$ in $\mathfrak{A}$ by $w^{\mathfrak{A}}$ — for example, $(fg)^{\mathfrak{A}}$ is the function on A which first applies g and then f. We shall now show that in any finite unary algebra (semiautomaton) $\mathfrak{A} = (A, \Sigma)$, the set of generalizations $\uparrow_{\mathfrak{A}} a$ can be computed by some finite automaton as illustrated by the following example:

Example 36. Consider the finite unary algebra (semiautomaton)

$$\mathfrak{A} = (\{a, b\}, \Sigma := \{f, g\})$$

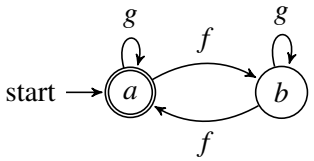
given by



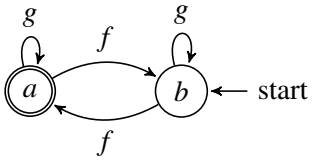
We can identify the set of all generalizations of a in $\mathfrak{A}$ with

$$\uparrow_{\mathfrak{A}} a = \{w \in \Sigma^* \mid \delta^*(a, w) = a\} \cup \{u \in \Sigma^* \mid \delta^*(b, u) = a\}.$$

Now define the automaton $\mathfrak{A}_{a\to\{a\}}$ by adding to the semiautomaton $\mathfrak{A}$ the initial state a and the set of final states $\{a\}$ (we use here the standard pictorial notation for automata)



and the automaton $\mathfrak{A}_{b\to\{a\}}$ by



We then clearly have

$$\uparrow_{\mathfrak{A}} a = ||\mathfrak{A}_{a\to\{a\}}|| \cup ||\mathfrak{A}_{b\to\{a\}}||.$$

It is straightforward to generalize the construction in Example 36:

Definition 37. Given a finite unary algebra (semiautomaton) $\mathfrak{A} = (A, \Sigma)$, the automaton $\mathfrak{A}_{b\to\{a\}}$ is the automaton induced by the functions in Σ with start state b and single final state a given by

$$\mathfrak{A}_{b\to\{a\}} := (A, \Sigma, b, \{a\}).$$

Fact 38. *Given any finite*[4] *unary algebra (semiautomaton)* $\mathfrak{A} = (A, \Sigma)$ *and* $a \in A$*, we have*

$$\uparrow_{\mathfrak{A}} a = \bigcup_{b \in A} ||\mathfrak{A}_{b \to \{a\}}||. \tag{9}$$

We are now ready to prove the main result of this section:

Theorem 39. *Let* $\mathfrak{A} = (A, \Sigma^{\mathfrak{A}})$ *and* $\mathfrak{B} = (B, \Sigma^{\mathfrak{B}})$ *be finite unary algebras (semiautomata) over the same set of function symbols (input alphabet)* Σ*. We have the following:*

1. *For any element (state)* $a \in A$*,* $\uparrow_{\mathfrak{A}} a$ *is a regular language.*
2. *For any elements (states)* $a \in A$ *and* $b \in B$*,* $a \uparrow_{\mathfrak{AB}} b$ *is a regular language.*
3. *For any elements (states)* $a \in A$ *and* $b \in B$*,* $a \Uparrow_{\mathfrak{AB}} b$ *is computable.*

Proof. Since A is finite and since regular languages are known to be closed under finitely many unions, we conclude by (9) that $\uparrow_{\mathfrak{A}} a$ is a regular language.

Since regular languages are known to be closed under finitely many intersections and since we already know that $\uparrow_{\mathfrak{A}} a$ and $\uparrow_{\mathfrak{B}} b$ are regular languages, we conclude that

$$a \uparrow_{\mathfrak{AB}} b = (\uparrow_{\mathfrak{A}} a) \cap (\uparrow_{\mathfrak{B}} b)$$

is a regular language.

Given some word $w \in a \uparrow_{\mathfrak{AB}} b$ (recall that we identify the term $wx \in T_\Sigma(\{x\})$ with the word $w \in \Sigma^*$ since x bears no information), we have $w \in a \Uparrow_{\mathfrak{AB}} b$ iff the term $wx \in T_\Sigma(\{x\})$ is $\sqsubseteq_{\mathfrak{A}}$-minimal and $\sqsubseteq_{\mathfrak{B}}$-minimal, which amounts to deciding whether there is some term $ux \in T_\Sigma(\{x\})$ such that $u \in a \uparrow_{\mathfrak{AB}} b$ and

$$\{u^{\mathfrak{A}} a \in A \mid a \in A\} \subsetneq \{w^{\mathfrak{A}} a \in A \mid a \in A\} \quad \text{or} \quad \{u^{\mathfrak{B}} b \in B \mid b \in B\} \subsetneq \{w^{\mathfrak{B}} b \in B \mid b \in B\}.$$

In a finite algebra, this is clearly a computable relation. Hence, $a \Uparrow_{\mathfrak{A}} b$ is computable in finite unary algebras (semiautomata). □

10. Finite algebras

In this section, let $\mathfrak{A}$ be a *finite* algebra which means that its underlying universe A is a finite set and it contains finitely many functions. For $k \geq 1$, let $X_k := \{x_1, \ldots, x_k\}$.

Recall that a (***frontier-to-root***) ***tree automaton*** (see e.g. Gécseg & Steinby, 2015)

$$\mathcal{T}_{k,\alpha,F}(\mathfrak{A}) := (\mathfrak{A}, L, X_k, \alpha, F)$$

consists of

- a finite L-algebra $\mathfrak{A}$,
- an ***initial assignment*** $\alpha : X_k \to A$, and
- a set $F \subseteq A$ of ***final states***.

4. Finiteness is required since regular languages are not closed under *infinite* union.

The ***regular tree language*** recognized by $\mathcal{T}_{k,\alpha,F}(\mathfrak{A})$ is given by

$$\|\mathcal{T}_{k,\alpha,F}(\mathfrak{A})\| := \left\{ s \in T_{LX_k} \;\middle|\; s^{\mathfrak{A}}\alpha \in F \right\}.$$

We have (recall the definition of $\uparrow^k_{\mathfrak{A}} a$ from §7)

$$\uparrow^k_{\mathfrak{A}} a = \bigcup_{\alpha \in A^{X_k}} \|\mathcal{T}_{k,\alpha,\{a\}}(\mathfrak{A})\|.$$

Since A^{X_k} is a finite set and tree automata are closed under finite union, the set $\uparrow^k_{\mathfrak{A}} a$ is a regular tree language. Moreover, since tree automata are closed under finite intersection, there is some tree automaton $\mathfrak{T}_{k,a,b}(\mathfrak{A})$ such that

$$a \uparrow^k_{\mathfrak{A}} b = (\uparrow^k_{\mathfrak{A}} a) \cap (\uparrow^k_{\mathfrak{A}} b) = \|\mathfrak{T}_{k,a,b}(\mathfrak{A})\|.$$

For the computation the set of ***minimally general*** k***-generalizations*** $a \Uparrow^k_{\mathfrak{A}} b$ it therefore remains to check for each $s \in a \uparrow^k_{\mathfrak{A}} b = \|\mathfrak{T}_{k,a,b}(\mathfrak{A})\|$ whether s is $\sqsubseteq_{\mathfrak{A}}$-minimal among the k-generalizations of a and b in $\mathfrak{A}$.

11. Generalized algebraic anti-unification

In this section, we introduce the following generalization of element-wise anti-unification from above to set-wise anti-unification. In the rest of this section, C is a subset of the universe A of $\mathfrak{A}$, and D is a subset of the universe B of $\mathfrak{B}$.

Definition 40. Define

$$\begin{aligned}
\uparrow_{\mathfrak{A}} C &:= \bigcap_{a \in C} \uparrow_{\mathfrak{A}} a \\
C \uparrow_{\mathfrak{AB}} D &:= (\uparrow_{\mathfrak{A}} C) \cap (\uparrow_{\mathfrak{B}} D) \\
C \Uparrow_{\mathfrak{AB}} D &:= \min_{\sqsubseteq_{\mathfrak{AB}}}(C \uparrow_{\mathfrak{AB}} D) \\
\Uparrow_{\mathfrak{A}} C &:= C \Uparrow_{\mathfrak{A}} C.
\end{aligned}$$

Proposition 41. $C \uparrow_{\mathfrak{AB}} D = \bigcap_{a \in C, b \in D} [a \uparrow_{\mathfrak{AB}} b]$.

Proof.

$$\begin{aligned}
C \uparrow_{\mathfrak{AB}} D &= (\uparrow_{\mathfrak{A}} C) \cap (\uparrow_{\mathfrak{B}} D) \\
&= \left[\bigcap_{a \in C} \uparrow_{\mathfrak{A}} a\right] \cap \left[\bigcap_{b \in D} \uparrow_{\mathfrak{B}} b\right] \\
&= \bigcap_{a \in C} \bigcap_{b \in D} [(\uparrow_{\mathfrak{A}} a) \cap (\uparrow_{\mathfrak{B}} b)] \\
&= \bigcap_{a \in C, b \in D} [a \uparrow_{\mathfrak{AB}} b].
\end{aligned}$$

□

Fact 42. *We have the following:*

- $\uparrow_{\mathfrak{A}} \{a\} = \uparrow_{\mathfrak{A}} a$.
- $C \subseteq \downarrow_{\mathfrak{A}} s \quad \Leftrightarrow \quad s \in \uparrow_{\mathfrak{A}} C$.
- $C = \downarrow_{\mathfrak{A}} s \quad \Rightarrow \quad s \in \Uparrow_{\mathfrak{A}} C$.
- $s \in C \uparrow_{\mathfrak{A}} D \quad \Leftrightarrow \quad C \cup D \subseteq \downarrow_{\mathfrak{A}} s$.

Example 43. We wish to compute $\Uparrow_{\mathfrak{N}} 2\mathbb{N}$ in $\mathfrak{N} = (\mathbb{N}, +, \cdot, 1)$. The generalization $2x$ defines exactly the even numbers in the sense that

$$\downarrow_{\mathfrak{N}} 2x = 2\mathbb{N},$$

which means that

$$2x \in \Uparrow_{\mathfrak{N}} 2\mathbb{N}.$$

Every other generalization $s \in \Uparrow_{\mathfrak{N}} 2\mathbb{N}$ thus has to satisfy

$$\downarrow_{\mathfrak{N}} s = \downarrow_{\mathfrak{N}} 2x = 2\mathbb{N}.$$

12. Conclusion

This paper introduced algebraic anti-unification in the general setting of universal algebra thus complementing the purely syntactic theory of anti-unification from the literature as initiated in the seminal works of Reynolds (1970) and Plotkin (1970).

The framework of this paper is unilingual in the sense that the underlying language of the algebras involved in anti-unification are the same. This is common practice in universal algebra. However, one can easily imagine practical scenarios in theoretical computer science and artificial intelligence, where different underlying languages are desired. A major line of future theoretical research therefore is to generalize the notions and results of this paper from a unilingual to a *bilingual* setting where the underlying languages $L_{\mathfrak{A}}$ and $L_{\mathfrak{B}}$ of $\mathfrak{A}$ and $\mathfrak{B}$ may differ. One possibility is to use the well-known notion of *interpretability* of one theory into another (see e.g. Hinman, 2005, §2.6) (recall that an algebra is a structure in the logical sense without relations other than equality).

From a practical point of view, the main line of future research is to study computability and complexity issues. Recall that in §9, we have shown that in finite unary algebras (aka semiautomata), minimally general generalizations can be computed using standard techniques from the theory of finite automata. Moreover, in §10, we have shown that more generally, in *any* finite algebra, we can use tree automata to compute minimally general k-generalizations, for any k. This is closely related to finite model theory (see e.g. Ebbinghaus & Flum, 1999; Libkin, 2012). However, since in practice we often encounter finitely representable *infinite* structures (Blumensath & Grädel, 2000, 2004), obtaining analous computability results is mandatory.

Another important line of applied future research is to develop *efficient* algorithms for the computation of minimally general generalizations in finite and infinite structures and to provide implementations which can be used in practice.

References

Antić, C. (2022). Analogical proportions. *Annals of Mathematics and Artificial Intelligence*, *90*(6), 595–644. https://doi.org/10.1007/s10472-022-09798-y.

Antić, C. (2023a). Generalization-based similarity. https://arxiv.org/pdf/2302.10096.pdf.

Antić, C. (2023b). Logic program forms. https://hal.science/hal-04261236.

Antić, C. (2023c). Logic program proportions. *Annals of Mathematics and Artificial Intelligence*, accepted. https://arxiv.org/pdf/1809.09938.pdf.

Antić, C. (2023d). Tree proportions. https://hal.science/hal-04247007.

Baader, F., & Snyder, W. (2001). Unification theory. In *Handbook of Automated Reasoning*. Elsevier.

Badra, F., & Lesot, M.-J. (2023). Case-based prediction — a survey. *International Journal of Approximate Reasoning*, 108920.

Badra, F., Sedki, K., & Ugon, A. (2018). On the role of similarity in analogical transfer. In Cox, M. T., Funk, P., & Begum, S. (Eds.), *ICCBR 2018*, LNAI 11156, pp. 499–514. Springer-Verlag.

Barwell, A. D., Brown, C., & Hammond, K. (2018). Finding parallel functional pearls: Automatic parallel recursion scheme detection in Haskell functions via anti-unification. *Future Generation Computer Systems*, *79*(2), 669–686.

Blumensath, A., & Grädel, E. (2000). Automatic structures. In *LICS 2000*, pp. 51–62. IEEE Computer Society.

Blumensath, A., & Grädel, E. (2004). Finite presentations of infinite structures: Automata and interpretations. *Theory of Computing Systems*, *37*, 641–674.

Brewka, G., Eiter, T., & Truszczynski, M. (2011). Answer set programming at a glance. *Communications of the ACM*, *54*(12), 92–103.

Burghardt, J. (2005). *E*-generalization using grammars. *Artificial Intelligence*, *165*(1), 1–35.

Burris, S., & Sankappanavar, H. (2000). *A Course in Universal Algebra*. http://www.math.hawaii.edu/~ralph/Classes/619/univ-algebra.pdf.

Cao, D., Kunkel, R., Nandi, C., Willsey, M., Tatlock, Z., & Polikarpova, N. (2023). babble: Learning better abstractions with e-graphs and anti-unification. *Proceedings of the ACM on Programming Languages*, *7*(POPL), 396–424.

Cerna, D. M., & Kutsia, T. (2023). Anti-unification and generalization: a survey. In *IJCAI 2023*, pp. 6563–6573.

Cropper, A. (2022). Inductive logic programming at 30. *Machine Learning*, *111*(1), 147–172.

Cropper, A., & Morel, R. (2021). Learning programs by learning from failures. *Machine Learning*, *110*(4), 801–856.

de Sousa, R. R., Soares, G., Gheyi, R., Barik, T., & D'Antoni, L. (2021). Learning quick fixes from code repositories. In *SBES 2021*, pp. 74–83.

Ebbinghaus, H.-D., & Flum, J. (1999). *Finite Model Theory* (2 edition). Springer Monographs in Mathematics. Springer-Verlag, Berlin/Heidelberg.

Gécseg, F., & Steinby, M. (2015). *Tree Automata* (2 edition). https://arxiv.org/pdf/1509.06233.pdf.

Giunchiglia, F., & Walsh, T. (1992). A theory of abstraction. *Artificial Intelligence*, *57*, 323–389.

Gulwani, S. (2016). Programming by examples — and its applications in data wrangling. In *NATO Science for Peace and Security Series*, Vol. 45, pp. 137–158. IOS Press.

Heinz, B. (1995). *Anti-Unifikation modulo Gleichungstheorie und deren Anwendung zur Lemmagenerierung*. Ph.D. thesis, TU Berlin.

Hinman, P. G. (2005). *Fundamentals of Mathematical Logic*. A K Peters, Wellesley, MA.

Holcombe, W. M. (1982). *Algebraic Automata Theory*. Cambridge Studies in Advanced Mathematics 1. Cambridge University Press, New York.

Huet, G. (1976). *Résolution d'équations dans des langages d'ordre 1,2,...,ω*. Ph.D. thesis, Université Paris VII.

Hug, N., Prade, H., Richard, G., & Serrurier, M. (2019). Analogical proportion-based methods for recommendation — first investigations. *Fuzzy Sets and Systems*, *366*, 110–132.

Knoblock, C. A. (1994). Automatically generating abstractions for planning. *Artificial Intelligence*, *68*(2), 243–302.

Kramer, J. (2007). Is abstraction the key to computing?. *Communications of the ACM*, *50*(4), 37–42.

Lepage, Y. (1998). Solving analogies on words: an algorithm. In Boitet, C., & Whitelock, P. (Eds.), *COLING-ACL 1998*, pp. 728–735. Morgan Kaufmann Publishers.

Lepage, Y. (2001). Analogy and formal languages. *Electronic Notes in Theoretical Computer Science*, *53*, 180–191.

Lepage, Y. (2003). *De L'Analogie. Rendant Compte de la Commutation en Linguistique*. Habilitation à diriger les recherches, Université Joseph Fourier, Grenoble.

Lepage, Y. (2014). Analogies between binary images: application to Chinese characters. In Prade, H., & Richard, G. (Eds.), *Computational Approaches to Analogical Reasoning: Current Trends*, Studies in Computational Intelligence 548, pp. 25–57. Springer-Verlag.

Libkin, L. (2012). *Elements of Finite Model Theory*. Springer-Verlag, Berlin/Heidelberg.

Lifschitz, V. (2019). *Answer Set Programming*. Springer Nature Switzerland AG, Cham, Switzerland.

McCarthy, J. (1987). Generality in artificial intelligence. *Communications of the ACM*, *30*(12), 1030–1035.

Muggleton, S. (1991). Inductive logic programming. *New Generation Computing*, *8*(4), 295–318.

Plaisted, D. A. (1981). Theorem proving with abstraction. *Artificial Intelligence*, *16*(1), 47–108.

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

Prade, H., & Richard, G. (2021). Analogical proportions: why they are useful in AI. In Zhou, Z.-H. (Ed.), *IJCAI 2021*, pp. 4568–4576.

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

Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, *5*(2), 115–135.

Saitta, L., & Zucker, J.-D. (2013). *Abstraction in Artificial Intelligence and Complex Systems*. Springer.

Saribatur, Z. G., & Eiter, T. (2020). Omission-based abstraction for answer set programs. *Theory and Practice of Logic Programming*, 1–51.

Saribatur, Z. G., Eiter, T., & Schüller, P. (2021). Abstraction for non-ground answer set programs. *Artificial Intelligence*, *300*, 103563.

Schmidt, M., Krumnack, U., Gust, H., & Kühnberger, K.-U. (2014). Heuristic-driven theory projection: an overview. In Prade, H., & Richard, G. (Eds.), *Computational Approaches to Analogical Reasoning: Current Trends*, Vol. 548 of *Studies in Computational Intelligence*, pp. 163–194. Springer-Verlag, Berlin/Heidelberg.

Sipser, M. (2013). *Introduction to the Theory of Computation* (3 edition). Cengage Learning, Boston.

Vanhoof, W., & Yernaux, G. (2019). Generalization-driven semantic clone detection in CLP. In Gabbrielli, M. (Ed.), *LOPSTR 2019*, pp. 228–242. Springer-Verlag.

Weller, S., & Schmid, U. (2007). Solving proportional analogies using E-generalization. In Freksa, C., Kohlhase, M., & Schmill, K. (Eds.), *KI 2006, LNAI 4314*, pp. 64–75. Springer-Verlag.