Unital Anti-unification: Type and Algorithms

David M. Cerna and Temur Kutsia

JOHANNES KEPLER UNIVERSITY LINZ

July 2^{nd}, 2020

What is Anti-Unification (AU)?

- Let Σ be a term alphabet and $\mathcal{V}$ a set of variables.
- By $\mathcal{T}(\Sigma, \mathcal{V})$, we refer to the set of terms inductively constructable using symbols from Σ and variables from $\mathcal{V}$.
- Substitution maps variables of $\mathcal{V}$ to terms of $\mathcal{T}(\Sigma, \mathcal{V})$.
- (Unification): Given $t_1, t_2 \in \mathcal{T}(\Sigma, \mathcal{V})$, does there exists σ such that $t_1\sigma = t_2\sigma$?
- (Anti-Unification): Does there exists a term $t_3 \in \mathcal{T}(\Sigma, \mathcal{V})$ and substitutions σ_1 and σ_2 s.t. $t_3\sigma_1 = t_1$ and $t_3\sigma_2 = t_2$?
- A generalization always exists between terms of $\mathcal{T}(\Sigma, \mathcal{V})$.
  - let $t_3 = x$, $\sigma_1 = \{x \mapsto t_1\}$, $\sigma_2 = \{x \mapsto t_2\}$
- We are interested in least general generalizations.

What is Anti-Unification (AU)?

- Let g_1 and g_2 be generalizations of $t_1, t_2 \in \mathcal{T}(\Sigma, \mathcal{V})$, then g_1 is less general than g_2, $g_2 \prec g_1$ if there exists μ s.t. $g_2\mu = g_1$.
- g_1 is least general if for every comparable term g_2, $g_2 \prec g_1$.
- Such anti-unifiers are called least general generalizations (lggs)
- In 1970, Plotkin and Reynolds independently showed that syntactic first-order AU has a unique lgg.
- May not be the case for AU modulo an equational theory.
- E-generalization considers AU where symbols of $\Sigma_E \subseteq \Sigma$ are interpreted w.r.t an equational theory E.
- Note that $=_E$ replaces $=$ and $\prec_E$ replaces $\prec$.
  - That is equality and generality modulo E.

Complete Sets of Solutions

- $\mathbf{C}_E(t,s)$ is complete for $t \triangleq s$ if for any E-generalization g, either $g \in \mathbf{C}_E(t,s)$ or there exists $g' \in \mathbf{C}_E(t,s)$ s.t. $g \prec_E g'$.
- $\mathbf{C}^{\mu}_E(t,s)$ is minimal, if every member is $\prec_E$-incomparable.
- There are four types of minimal complete sets in literature:
  - **UNITARY:** $|\mathbf{C}^{\mu}_E(t,s)| = 1$ [Plotkin & Reynolds, 1970]
    - Syntactic First-order Anti-unification ($E = \emptyset$).
  - **FINITARY:** $1 < |\mathbf{C}^{\mu}_E(t,s)| < \infty$ [Alpuente *et al.*, 2014]
    - First-order anti-unification modulo A, C, and AC theories .
  - **INFINITARY:** $|\mathbf{C}^{\mu}_E(t,s)| = \infty$ [Cerna & Kutsia, 2019]
    - First-order anti-unification modulo purely idempotent theories.
  - **NULLARY:** $\mathbf{C}^{\mu}_E(t,s)$ does not exists [Cerna & Kutsia, 2020]
    - First-order anti-unification modulo purely unital theories (multiple unit elements).

Motivation: Theories Behaving Badly

- Unit element theories were studied in [Alpuente *et al.*, 2014].
  - Known that $\mathbf{C}^{\mu}_{\mathrm{U}}(t,s)$ may be infinite.
- Similar was shown for Idempotent theories [Pottier, 1989].
  - Was not proven to be AU type infinitary in this work.
- This motivated investigating exhaustive construction of $\mathbf{C}_E(t,s)$ through grammar transformations [Burghardt, 2005].
- In [Cerna & Kutsia, 2019], a grammars based algorithm is used to prove AU modulo I is of type infinitary.
- Unital theories are collapse theories [Siekmann, 1989] as well.
  - Can a similar approach work?
- Consider the following AU problem: $g(f(a,c),a) \triangleq g(c,b)$

$$E_{\mathrm{U}} = \{f(\epsilon_f, x) = x \ , \ f(x, \epsilon_f) = x\}$$

LGG Derivation Using the Expand_{U} Inference

- In [Alpuente *et al.*, 2014], Expand_{U} extends the syntactic generalization algorithm.

$$\{x : g(f(a,c),a) \triangleq g(c,b)\}; \emptyset; x \Rightarrow_{\text{Dec}}$$
$$\{x_1 : f(a,c) \triangleq c\ ,\ x_2 : a \triangleq b\}; \emptyset; g(x_1,x_2) \Rightarrow_{\text{Expand}_{\text{U}}}$$
$$\{x_1 : f(a,c) \triangleq f(\epsilon_f, c), x_2 : a \triangleq b\}; \emptyset; g(x_1,x_2) \Rightarrow_{\text{Dec}}$$
$$\{x_3 : a \triangleq \epsilon_f\ ,\ x_4 : c \triangleq c\ ,\ x_2 : a \triangleq b\}; \emptyset; g(f(x_3,x_4),x_2) \Rightarrow_{\text{Dec}}$$
$$\{x_3 : a \triangleq \epsilon_f, x_2 : a \triangleq b\}; \emptyset; g(f(x_3,c),x_2) \Rightarrow_{\text{Solve}}$$
$$\{x_2 : a \triangleq b\}; \{x_3 : a \triangleq \epsilon_f\}; g(f(x_3,c),x_2) \Rightarrow_{\text{Solve}}$$
$$\emptyset; \{x_2 : a \triangleq b\ ,\ x_3 : a \triangleq \epsilon_f\}; \mathbf{g(f(x_3,c),x_2)}$$

- Expand_{U} introduces f allowing further decomposition.
- Finitary and finds the minimal complete set for linear variant.
- Result discussed in [Cerna & Kutsia, 2020 (MSCS)] over higher-order terms.

Motivation: Unexpected LGGs

- Expand requires f to occur as a head symbol in $s \triangleq t$.
- Reason? Infinite **cycles**.
- If we drop this restriction, what happens?

$$\{x : g(f(a,c),a) \triangleq g(c,b)\}; \emptyset; x \Rightarrow_{\text{Dec}}$$

$$\ldots$$

$$\{x_2 : a \triangleq b\}; \{x_3 : a \triangleq \epsilon_f\}; g(f(x_3,c),x_2) \Rightarrow_{\text{DH-U}}$$

$$\{x_5 : a \triangleq \epsilon_f \ , \ x_6 : \epsilon_f \triangleq b\}; \{x_3 : a \triangleq \epsilon_f\}; g(f(x_3,c),f(x_5,x_6)) \Rightarrow_{\text{Solve}}$$

$$\{x_5 : a \triangleq \epsilon_f\}; \{x_3 : a \triangleq \epsilon_f \ , \ x_6 : \epsilon_f \triangleq b\}; g(f(x_3,c),f(x_5,x_6)) \Rightarrow_{\text{Solve}}$$

$$\emptyset; \{x_3 : a \triangleq \epsilon_f \ , \ x_6 : \epsilon_f \triangleq b \ , \ x_5 : a \triangleq \epsilon_f\}; g(f(x_3,c),f(x_5,x_6)) \Rightarrow_{\text{Merge}}$$

$$\{x_3 : a \triangleq \epsilon_f \ , \ x_6 : \epsilon_f \triangleq b\}; \mathbf{g(f(x_3,c),f(x_3,x_6))}$$

- $g(f(x_3,c),x_2) \prec g(f(x_3,c),f(x_3,x_6))$
- Though, only one of infinitely many derivations.

New Rule and the consequences

- Discussed in [Cerna & Kutsia, 2020 (MSCS)] as:

$$\{x : t \triangleq s\} \uplus A \; ; \; S \; ; \; g \Longrightarrow_{\text{DH-U}}$$

$$\{x_1 : t \triangleq \epsilon_f \; , \; x_2 : \epsilon_f \triangleq s\} \uplus A \; ; \; S \; ; \; g\{x \mapsto f(x_1, x_2)\}$$

- Unnecessary for linear variant.
- Tree grammar based algorithms [Cerna & Kutsia, 2019] can capture the cyclic behavior of the DH-U inference.
- Remaining Questions:
  1) AU over $\{f(x, \epsilon_f) = x \; , \; f(\epsilon_f, x) = x\}$, finitary?
  2) Algorithm over $\{f(x, \epsilon_f) = x \; , \; f(\epsilon_f, x) = x\}$, complete?
  3) AU over $\bigcup_{i=0}^{n}\{f_i(x, \epsilon_{f_i}) = x \; , \; f_i(\epsilon_{f_i}, x) = x\}$, infinitary?
  4) Algorithm over $\bigcup_{i=0}^{n}\{f_i(x, \epsilon_{f_i}) = x \; , \; f_i(\epsilon_{f_i}, x) = x\}$, exists?

New Rule and the consequences

- Discussed in [Cerna & Kutsia, 2020 (MSCS)] as:

$$\{x : t \triangleq s\} \uplus A \; ; \; S \; ; \; g \Longrightarrow_{\text{DH-U}}$$

$$\{x_1 : t \triangleq \epsilon_f \; , \; x_2 : \epsilon_f \triangleq s\} \uplus A \; ; \; S \; ; \; g\{x \mapsto f(x_1, x_2)\}$$

- Unnecessary for linear variant.
- Tree grammar based algorithms [Cerna & Kutsia, 2019] can capture the cyclic behavior of the DH-U inference.
- Remaining Questions:
  1) AU over $\{f(x, \epsilon_f) = x \; , \; f(\epsilon_f, x) = x\}$, finitary? Yes.
  2) Algorithm over $\{f(x, \epsilon_f) = x \; , \; f(\epsilon_f, x) = x\}$, complete? Yes.
  3) AU over $\bigcup_{i=0}^{n}\{f_i(x,\epsilon_{f_i})=x \; , \; f_i(\epsilon_{f_i},x)=x\}$, infinitary? NO!
  4) Algorithm over $\bigcup_{i=0}^{n}\{f_i(x,\epsilon_{f_i})=x \; , \; f_i(\epsilon_{f_i},x)=x\}$, exists? Maybe?

New Rule and the consequences

- Discussed in [Cerna & Kutsia, 2020 (MSCS)] as:

$$\{x : t \triangleq s\} \uplus A \; ; \; S \; ; \; g \Longrightarrow_{\text{DH-U}}$$

$$\{x_1 : t \triangleq \epsilon_f \; , \; x_2 : \epsilon_f \triangleq s\} \uplus A \; ; \; S \; ; \; g\{x \mapsto f(x_1, x_2)\}$$

- Unnecessary for linear variant.
- Tree grammar based algorithms [Cerna & Kutsia, 2019] can capture the cyclic behavior of the DH-U inference.
- Remaining Questions:
  1) AU over $\{f(x, \epsilon_f) = x \; , \; f(\epsilon_f, x) = x\}$, finitary? Yes.
  2) Algorithm over $\{f(x, \epsilon_f) = x \; , \; f(\epsilon_f, x) = x\}$, complete? Yes.
  3) AU over $\bigcup_{i=0}^{n}\{f_i(x,\epsilon_{f_i})=x \; , \; f_i(\epsilon_{f_i},x)=x\}$, infinitary? NO! ◀
  4) Algorithm over $\bigcup_{i=0}^{n}\{f_i(x,\epsilon_{f_i})=x \; , \; f_i(\epsilon_{f_i},x)=x\}$, exists? Maybe?

Purely Multi-Unital AU is Nullary

- We focus on the following theory:

$$U_2 = \{f(x,\epsilon_f) = x\ ,\ f(\epsilon_f,x) = x\ ,\ g(x,\epsilon_g) = x\ ,\ g(\epsilon_g,x) = x\},$$

- and consider the anti-unification problem $\epsilon_f \triangleq \epsilon_g$.
- Obviously, x is a solution $x\{x \mapsto \epsilon_f\} = \epsilon_f$, $x\{x \mapsto \epsilon_g\} = \epsilon_g$
- What about other solutions?

Purely Multi-Unital AU is Nullary

- We focus on the following theory:

$$\mathsf{U}_2 = \{f(x,\epsilon_f) = x\,,\ f(\epsilon_f,x) = x\,,\ g(x,\epsilon_g) = x\,,\ g(\epsilon_g,x) = x\},$$

- and consider the anti-unification problem $\epsilon_f \triangleq \epsilon_g$.
- Obviously, x is a solution $x\{x \mapsto \epsilon_f\} = \epsilon_f$, $x\{x \mapsto \epsilon_g\} = \epsilon_g$
- What about other solutions? Let's apply the DH-U rule.

Purely Multi-Unital AU is Nullary

- We focus on the following theory:

$$\mathsf{U}_2 = \{f(x,\epsilon_f) = x\ ,\ f(\epsilon_f,x) = x\ ,\ g(x,\epsilon_g) = x\ ,\ g(\epsilon_g,x) = x\},$$

- and consider the anti-unification problem $\epsilon_f \triangleq \epsilon_g$.
- Obviously, x is a solution $x\{x \mapsto \epsilon_f\} = \epsilon_f$, $x\{x \mapsto \epsilon_g\} = \epsilon_g$
- What about other solutions? Let's apply the DH-U rule.

$$\{x : \epsilon_f \triangleq \epsilon_g\}; \emptyset; x \Rightarrow_{\mathsf{DH\text{-}U}}$$
$$\{x_1 : \epsilon_f \triangleq \epsilon_g\ ,\ x_2 : \epsilon_g \triangleq \epsilon_g\}; \emptyset; g(x_1, x_2) \Rightarrow_{\mathsf{DH\text{-}U}}$$
$$\{x_1 : \epsilon_f \triangleq \epsilon_g\ ,\ x_3 : \epsilon_g \triangleq \epsilon_f\ ,\ x_2 : \epsilon_f \triangleq \epsilon_g\}; \emptyset; g(x_1, f(x_3, x_4)) \Rightarrow_{Solve}$$
$$\cdots \Rightarrow_{Merge} \cdots$$
$$\{x_1 : \epsilon_f \triangleq \epsilon_g\ ,\ x_3 : \epsilon_g \triangleq \epsilon_f\}; \mathbf{g}(\mathbf{x_1}, \mathbf{f}(\mathbf{x_3}, \mathbf{x_1}))$$

- Notice, $x \prec_{\mathsf{U}_2} g(x_1, f(x_3, x_1))$. Process is repeatable on x_1 and x_3.

Purely Multi-Unital AU is Nullary

- Can generate an infinite sequence of less generality.
  - Does not guarantee Nullarity, need more general properties.

Purely Multi-Unital AU is Nullary

- Can generate an infinite sequence of less generality.
  - Does not guarantee Nullarity, need more general properties.

Theorem

Any reduced generalization of $\epsilon_f \triangleq \epsilon_g$ is either a variable or contains two distinct variables.

Theorem

For every generalization $\mathbf{g}$ of $\epsilon_f \triangleq \epsilon_g$ there exists a substitution ϑ such that $\mathbf{g}\vartheta$ is a reduced generalization of $\epsilon_f \triangleq \epsilon_g$.

Purely Multi-Unital AU is Nullary

- Can generate an infinite sequence of less generality.
  - Does not guarantee Nullarity, need more general properties.

Theorem

Any reduced generalization of $\epsilon_f \triangleq \epsilon_g$ *is either a variable or contains two distinct variables.*

Theorem

For every generalization $\mathbf{g}$ *of* $\epsilon_f \triangleq \epsilon_g$ *there exists a substitution* ϑ *such that* $\mathbf{g}\vartheta$ *is a reduced generalization of* $\epsilon_f \triangleq \epsilon_g$.

Reduced:

- $x \in var(\mathbf{g})$, $x\sigma_1 \neq_{\mathsf{U}_2} x\sigma_2$.
- $x, y \in var(\mathbf{g})$ either $x = y$, or for some $\theta \in \{\sigma_1, \sigma_2\}$, $x\theta \neq_{\mathsf{U}_2} y\theta$.

Purely Multi-Unital AU is Nullary

- Let $\mathbf{g}$ generalize $\epsilon_f \triangleq \epsilon_g$.
- We use $g(x, f(y, x))$ to construct a less general generalization.

Theorem

Let $\mathbf{g}$ *be a reduced generalization of* $\epsilon_f \triangleq \epsilon_g$*. Then there exists a reduced generalization* $\mathbf{g}'$ *of* $\epsilon_f \triangleq \epsilon_g$ *such that* $\mathbf{g} \prec_{\mathsf{U}_2} \mathbf{g}'$*.*

Proof (Sketch).

Let $\mathbf{g}' = \mathbf{g}\{x \mapsto g(x, f(x, y))\}$. If $g = x$ then obviously $\mathbf{g} \prec_{\mathsf{U}_2} \mathbf{g}'$. Thus, $Var(\mathbf{g}) = \{x, y\}$. Be reducibility, we can assume $occ(x, \mathbf{g}) = n$ and $occ(y, \mathbf{g}) = m$, for $n, m > 0$. That is $occ(x, \mathbf{g}') = 2n$ and $occ(y, \mathbf{g}') = n + m$. Assuming $\mathbf{g}' \prec_U \mathbf{g}$ contradicts that $n, m > 0$. □

Purely Multi-Unital AU is Nullary

Theorem

Let $\mathcal{C}$ be a complete set of generalizations of $\epsilon_f \triangleq \epsilon_g$. Then $\mathcal{C}$ contains $\mathbf{g}$ and $\mathbf{g}'$ such that $\mathbf{g} \prec_{\mathsf{U}_2} \mathbf{g}'$.

Proof (Sketch).

Let $\mathbf{g} \in \mathcal{C} \implies \mathbf{g}\vartheta$ is reduced $\implies$ there exists φ s.t. $\mathbf{g}\vartheta \prec_{\mathsf{U}_2} \mathbf{g}\vartheta\varphi$ $\implies$ By completeness, $\exists\mu$ such that $\mathbf{g}\vartheta\varphi\mu \in \mathcal{C} \implies \mathbf{g}' = \mathbf{g}\vartheta\varphi\mu$. □

Beyond Purely Multi-Unital Theories:

- Seems to hold adding associativity and commutativity.
- Breaks when idempotency is added (for both symbols).
- $g(x, f(x, y))\{y \mapsto x\} =_{\mathsf{UI}_2} g(x, f(x, x)) =_{\mathsf{UI}_2} g(x, x) =_{\mathsf{UI}_2} x$
- Maybe $g(x, f(x, y))$ is the wrong seed term for idempotency.
- Motivated investigation into fragments and variants.

Algorithms: Linear Variant

- Algorithm is tree grammar based à la [Cerna & Kutsia, 2019].
- Term version discussed in [Cerna & Kutsia, 2020 (MSCS)].
  - Uses $\mathsf{Expand}_{\mathsf{U}}$ [Alpuente *et al.*, 2014].
- Our algorithm consist of a set of transformation rules which are applied to configurations $\mathbf{A}; \mathbf{S}; \mathbf{L}; \mathbf{B}$.

  A - A set of anti-unification triples (AUT) $x : t \triangleq s$.
  S - A set of solved AUTs $x : t \triangleq s$.
  L - A set of cycles $(x : t \triangleq s, \{\epsilon_f, \cdots\})$.
  B - A set of Bindings $\{x \mapsto t\}$.
- The initial configuration is $\{x : t \triangleq s\}; \emptyset; \emptyset; \{x_{root} \to x\}$.
- Rules applied exhaustively following the strategy Step.

Algorithms: Example rules

Dec: **Decomposition**

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

Exp-U-Both: **Expansion for Unit, Both**

$\{x : t \triangleq s\} \mathbin{\dot\cup} A;\ S;\ L;\ B \Longrightarrow \{x_1 : g(t, \epsilon_g) \triangleq s,\ x_2 : g(\epsilon_g, t) \triangleq s,\ y_1 : t \triangleq f(s, \epsilon_f), y_2 : t \triangleq f(\epsilon_f, s)\} \cup A;\ S;\ L; B \cup \{x \mapsto x_1\} \cup \{x \mapsto x_2\} \cup \{x \mapsto y_1\} \cup \{x \mapsto y_2\}$,

where $head(t) = f \neq g = head(s)$, $\mathsf{U} \in Ax(f) \cap Ax(g)$

Algorithms: Strategy

Solve: **Solve**

$\{x : s \triangleq t\} \cup A;\ S;\ L;\ B \Longrightarrow A;\ \{x : s \triangleq t\} \cup S; L;\ B,$
where $head(s) \neq head(t)$ and $\mathsf{U} \notin Ax(head(t)) \cup Ax((head(s)))$.

Step strategy:

- Select an AUT **a** arbitrarily from **A** .
- Apply a rule applicable to **a**.
  - There is only one such rule for each **a**.
- If the rule is Exp-U-Both, apply Dec to all four new AUTs.
- If the rule is Exp-U-L or Exp-U-R, apply Dec to both AUTs.

Algorithm: Pseudocode of Step

Require: A configuration $\mathbf{C} = A; S; L; B$ and an AUT $\mathbf{a} = x : t \triangleq s \in A$.

if $head(t) = head(s)$ **then**
  Apply Dec to $\mathbf{a}$, resulting in $\mathbf{C}'$. Update $\mathbf{C} \leftarrow \mathbf{C}'$
else if $\exists f, g \in \mathcal{F} : (\mathsf{U} \in (Ax(f) \cap Ax(g)) \wedge head(s) = f \neq g = head(t))$ **then**
  Apply Exp-U-Both to $\mathbf{a}$ resulting in $\mathbf{C}' = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4\} \cup A; S; L; B'$
  Apply Dec to $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_1, \mathbf{a}_2$ resulting in $\mathbf{C}''$. Update $\mathbf{C} \leftarrow \mathbf{C}''$
else if $head(t) \neq head(s) \wedge \exists f \in \mathcal{F} : (\mathsf{U} \in Ax(f) \wedge head(s) = f)$ **then**
  Apply Exp-U-L to $\mathbf{a}$ resulting in $\mathbf{C} = \{\mathbf{a}_1, \mathbf{a}_2\} \cup A; S; L; B'$
  Apply Dec to $\mathbf{a}_1, \mathbf{a}_2$ resulting in $\mathbf{C}''$. Update $\mathbf{C} \leftarrow \mathbf{C}''$
else if $head(t) \neq head(s) \wedge \exists f \in \mathcal{F} : (\mathsf{U} \in Ax(f) \wedge head(t) = f)$ **then**
  Apply Exp-U-R to $\mathbf{a}$ resulting in $\{\mathbf{a}_1, \mathbf{a}_2\} \cup A; S; L; B'$
  Apply Dec to $\mathbf{a}_1, \mathbf{a}_2$ resulting in $\mathbf{C}''$. Update $\mathbf{C} \leftarrow \mathbf{C}''$
else
  Apply Solve to $\mathbf{a}$ resulting in $\mathbf{C}'$. Update $\mathbf{C} \leftarrow \mathbf{C}'$
end if
return $\mathbf{C}$

Algorithm: Pseudocode of $\mathfrak{G}_{\text{U-lin}}$

Require: A configuration $\mathbf{C} = A; S; L; B$
  while $A \neq \emptyset$ **do**
    $\mathbf{a} \leftarrow x : t \triangleq s \in A$
    $\mathbf{C} \leftarrow \text{Step}(\mathbf{C}, \mathbf{a})$
  end while
  return $\mathbf{C}$

Theorem (Soundness)

If $\{x : t \triangleq s\}; \emptyset; \emptyset; \{x_{\text{root}} \mapsto x\} \Longrightarrow^* \emptyset; S; L; B$ *is a transformation sequence of* $\mathfrak{G}_{\text{U-lin}}$, *then for every* $r \in \mathcal{L}(\mathcal{G}(B))$, $r \preceq_U t$ *and* $r \preceq_U s$.

Theorem (Completeness of $\mathfrak{G}_{\text{U-lin}}$)

Let s *be a linear generalization of two terms* t_1 *and* t_2. *Then there exists a transformation sequence* $\{x : t_1 \triangleq t_2\}; \emptyset; \emptyset; \{x_{\text{root}} \mapsto x\} \Longrightarrow^* \emptyset; S; L; B$ *in* $\mathfrak{G}_{\text{U-lin}}$ *such that for some term* $r \in \mathcal{L}(\mathcal{G}(B))$, $s \preceq_U r$.

Example: Using $\mathfrak{G}_{\text{U-lin}}$

$$\{x : g(f(a,c),a) \triangleq g(c,b)\};\ \emptyset;\ \emptyset;\ \{x_{\text{root}} \mapsto x\} \Longrightarrow_{\text{Dec}}$$

$$\{x_1 : f(a,c) \triangleq c, x_2 : a \triangleq b\};\ \emptyset;\ \emptyset;\ \{x_{\text{root}} \mapsto g(x_1,x_2)\} \Longrightarrow_{\text{Exp-U-L}, \text{Dec}\times 2}$$

$$\{x_3 : a \triangleq \epsilon_f, x_4 : c \triangleq c, x_5 : a \triangleq c, x_6 : c \triangleq \epsilon_f, x_2 : a \triangleq b\};\ \emptyset; \emptyset;$$
$$\{x_{\text{root}} \mapsto g(x_1,x_2), x_1 \mapsto f(x_3,x_4), x_1 \mapsto f(x_5,x_6)\} \Longrightarrow_{\text{Dec}}$$

$$\{x_3 : a \triangleq \epsilon_f, x_5 : a \triangleq c, x_6 : c \triangleq \epsilon_f, x_2 : a \triangleq b\};\ \emptyset;\ \emptyset;$$
$$\{x_{\text{root}} \mapsto g(x_1,x_2), x_1 \mapsto f(x_3,c), x_1 \mapsto f(x_5,x_6)\} \Longrightarrow_{\text{Solve}\times 4}$$

$$\emptyset;\ \{x_3 : a \triangleq \epsilon_f, x_5 : a \triangleq c, x_6 : c \triangleq \epsilon_f, x_2 : a \triangleq b\};\ \emptyset;$$
$$\{x_{\text{root}} \mapsto g(x_1,x_2),\ x_1 \mapsto f(x_3,c),\ x_1 \mapsto f(x_5,x_6)\}$$

Thus, $\mathcal{L}(\mathcal{G}(B)) \approx_U \{g(f(x_3,c),x_2), g(f(x_5,x_6),x_2)\}$.

- Note that $g(f(x_5,x_6),x_2) \prec_{\text{U}} g(f(x_3,c),x_2)$.

Algorithms: One-unital fragment

Start-Cycle-U: **Cycle introduction for Unit**

$\{x : t \triangleq s\} \uplus A;\ S;\ L;\ B \Longrightarrow$
$\{y_1 : f(t, \epsilon_f) \triangleq f(\epsilon_f, s),\ y_2 : f(\epsilon_f, t) \triangleq f(s, \epsilon_f), y_3 : t \triangleq s\} \cup$
$A;\ S;\ \{(\{x : t \triangleq s\}, \{\epsilon_f\})\} \cup L;\ B \cup \{x \mapsto y_1\} \cup \{x \mapsto y_2\}$,
where $\mathsf{U} \in Ax(f)$, $(\{y : t \triangleq s\}, Un) \notin L$, $head(t) \neq \epsilon_f$ or $head(s) \neq \epsilon_f$, $\mathsf{U} \notin Ax(head(t)) \cup Ax(head(s))$.

Sat-Cycle-U: **Cycle Saturation for Unit**

$\{x : t \triangleq s\} \uplus A;\ S;\ \{(\{y : t \triangleq s\}, Un)\} \cup L;\ B \Longrightarrow \{x : t \triangleq s\} \cup A;\ S;\ (\{y : t \triangleq s\}, Un) \cup L;\ B\{x \mapsto y\} \cup \{y \mapsto x\}$,
where $x \neq y$ and $\{y \mapsto x\} \notin B$.

Algorithms: The Cycle Procedure

Merge: **Merge**

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

where $s_1 \approx_{\mathsf{U}} s_2$ and $t_1 \approx_{\mathsf{U}} t_2$.

Require: A configuration $\mathbf{C} = A; S; L; B$, an AUT $\mathbf{a} = x : t \triangleq s$

if $\exists f \in \mathcal{F} : (\mathsf{U} \in Ax(f) \wedge (\{y : t \triangleq s\}, Un) \notin L)$ **then**

  Apply Start-Cycle-U to $\mathbf{a}$ resulting in $\mathbf{C}' = \{\mathbf{a}_1, \mathbf{a}_2, x' : t \triangleq s\} \cup A; S; L'; B'$

  Apply Dec to $\mathbf{a}_1, \mathbf{a}_2$ resulting in $\mathbf{C}''$. Update $\mathbf{C} \leftarrow \mathbf{C}''$ and $\mathbf{a} \leftarrow x' : t \triangleq s$

end if

Exhaustively apply Sat-Cycle-U to $\mathbf{C}$ resulting in $\mathbf{C}^*$. Update $\mathbf{C} \leftarrow \mathbf{C}^*$

return $(\mathbf{C}, \mathbf{a})$

Algorithm: Pseudocode of $\mathfrak{G}_{\mathsf{U(f)}}$

Require: A configuration $\mathbf{C} = A; S; L; B$
while $A \neq \emptyset$ **do**
$\mathbf{a} \leftarrow x : t \triangleq s \in A$
$(\mathbf{C}, \mathbf{a}) \leftarrow \mathsf{Cycle}(\mathbf{C}, \mathbf{a})$
$\mathbf{C} \leftarrow \mathsf{Step}(\mathbf{C}, \mathbf{a})$
Exhaustively apply Sat-Cycle-U to $\mathbf{C}$ resulting in $\mathbf{C}^*$. Update $\mathbf{C} \leftarrow \mathbf{C}^*$
end while
Exhaustively apply Merge to $\mathbf{C}$ resulting in $\mathbf{C}^*$. Update $\mathbf{C} \leftarrow \mathbf{C}^*$
return $\mathbf{C}$

- $\mathfrak{G}_{\mathsf{U(f)}}$ is terminating, sound, and complete and surprisingly:

Theorem

The set $\mathcal{L}(\mathcal{G}(B))$ computed by $\mathfrak{G}_{\mathsf{U(f)}}$ contains only finitely many incomparable generalizations.

Example: Applying $\mathfrak{G}_{\mathsf{U}(f)}$

- Let us reconsider $g(f(a,c),a) \triangleq g(c,b)$:
- The resulting grammar is

$$\mathcal{G} = \left(\{\mathbf{x}\}, \{\ \mathbf{x}\ \}, \left\{ \begin{array}{c} f, g, \epsilon_f, a, b, \\ c, y, z, y', z' \end{array} \right\}, B \right),$$

where B is the set

$$\left\{ \begin{array}{lll}
\mathbf{x} \mapsto g(f(f(y,z),y'),z') & \mathbf{x} \mapsto g(f(y,z),f(y',z')) & \mathbf{x} \mapsto g(f(f(z,y'),y),f(z,z')) \\
\mathbf{x} \mapsto g(f(f(z,y),y'),f(z,z')) & \mathbf{x} \mapsto g(f(y,y'),z') & \mathbf{x} \mapsto g(f(f(y,z),y'),f(z',z)) \\
\mathbf{x} \mapsto g(f(y,f(z,y')),z') & \mathbf{x} \mapsto g(f(z,f(y,y')),z') & \mathbf{x} \mapsto g(f(z,f(y',y)),z') \\
\mathbf{x} \mapsto g(f(f(z,y),y'),f(z',z)) & \mathbf{x} \mapsto g(f(f(z,y'),y),f(z',z)) & \mathbf{x} \mapsto f(y,z) \\
\boxed{\mathbf{x} \mapsto g(f(z,c),f(y,z))} & \mathbf{x} \mapsto g(f(y,y'),f(z',z)) & \mathbf{x} \mapsto g(f(z,f(y,y')),f(z,z')) \\
\mathbf{x} \mapsto g(f(y,f(z,y')),f(z,z')) & \mathbf{x} \mapsto g(f(z,f(y',y)),f(z,z')) & \mathbf{x} \mapsto g(f(z,c),z') \\
\mathbf{x} \mapsto g(f(f(z,y'),y),z') & \mathbf{x} \mapsto g(f(z,f(y,y')),f(z',z)) & \mathbf{x} \mapsto g(f(y,f(z,y')),f(z',z)) \\
\mathbf{x} \mapsto g(f(f(y,z),y'),f(z,z')) & \boxed{\mathbf{x} \mapsto g(f(z,c),f(z,y))} & \mathbf{x} \mapsto g(f(z,f(y',y)),f(z',z)) \\
\mathbf{x} \mapsto f(y,z) & \mathbf{x} \mapsto g(f(f(z,y),y'),z') &
\end{array} \right\}.$$

- Note that $g(f(x_3,c),x_2) \prec_{\mathsf{U}} g(f(z,c),f(z,y))$.

Algorithms: Rule for Purely Multi-unital Theories

Branch-Cycle-U: **Branching Cycle for Unit**

$$\{x : t \triangleq s\} \mathbin{\dot\cup} A;\ S;\ \{(\{y : t \triangleq s\}, Un)\} \cup L;\ B \Longrightarrow$$
$$\{y_1 : f(t, \epsilon_f) \triangleq f(\epsilon_f, s), y_2 : f(\epsilon_f, t) \triangleq f(s, \epsilon_f), y_3 : t \triangleq s\} \cup$$
$$A;\ S;\ \{(\{y : t \triangleq s\}, \{\epsilon_f\} \cup Un)\} \cup L;\ B\{x \mapsto y\} \cup$$
$$\{y \mapsto y_1\} \cup \{y \mapsto y_2\},$$

where $\mathrm{U} \in Ax(f)$, $\epsilon_f \notin Un$, $head(t) \neq \epsilon_f$ or $head(s) \neq \epsilon_f$, $\mathrm{U} \notin Ax(head(t)) \cup Ax(head(s))$.

- The general algorithm uses all previously defined rules together with Branch-Cycle-U.

Algorithms: $\mathfrak{G}_{\mathsf{U}}$ Strategy

Require: A configuration $\mathbf{C} = A; S; L; B$

while $A \neq \emptyset$ **do**
  $\mathbf{a} \leftarrow x : t \triangleq s \in A$
  $(\mathbf{C}, \mathbf{a}) \leftarrow \mathsf{Cycle}(\mathbf{C}, \mathbf{a})$
  if $\exists f \in \mathcal{A} : (\mathsf{U} \in Ax(f) \wedge (\{y : t \triangleq s\}, Un) \in L \wedge \epsilon_f \notin Un)$ **then**
    repeat
      Apply Branch-Cycle-U to $\mathbf{a}$ resulting in $\mathbf{C}' = \{\mathbf{a}_1, \mathbf{a}_2, x' : t \triangleq s\} \cup A; S; L'; B'$
      Apply Dec to $\mathbf{a}_1, \mathbf{a}_2$ resulting in $\mathbf{C}''$. Update $\mathbf{C} \leftarrow \mathbf{C}''$ and $\mathbf{a} \leftarrow x' : t \triangleq s$
      Exhaustively apply Sat-Cycle-U to $\mathbf{C}$ resulting in $\mathbf{C}^*$. Update $\mathbf{C} \leftarrow \mathbf{C}^*$
    until $\forall f \in \mathcal{A} : (\mathsf{U} \in Ax(f) \wedge (\{y : t \triangleq s\}, Un) \in L) \Rightarrow \epsilon_f \in Un)$
  end if
  $\mathbf{C} \leftarrow \mathsf{Step}(\mathbf{C}, \mathbf{a})$
  Exhaustively apply Sat-Cycle-U to $\mathbf{C}$ resulting in $\mathbf{C}^*$. Update $\mathbf{C} \leftarrow \mathbf{C}^*$
end while
Exhaustively apply Merge to $\mathbf{C}$ resulting in $\mathbf{C}^*$. Update $\mathbf{C} \leftarrow \mathbf{C}^*$
return $\mathbf{C}$

Example: Applying $\mathfrak{G}_{\mathsf{U(f)}}$

- Let us reconsider $\epsilon_f \triangleq \epsilon_g$:

$$\mathcal{G}' = \left(\{\mathbf{x}\}, \left\{ \begin{array}{c} \mathbf{x}, \\ \mathbf{y} \end{array} \right\}, \left\{ \begin{array}{c} f, g, \\ \epsilon_f, \epsilon_g, \\ y, z \end{array} \right\}, \left\{ \begin{array}{ll} \mathbf{x} \mapsto g(\mathbf{x}, f(\mathbf{x}, \mathbf{y})), & \mathbf{x} \mapsto f(\mathbf{x}, g(\mathbf{x}, \mathbf{y})) \\ \mathbf{x} \mapsto f(g(\mathbf{y}, \mathbf{x}), \mathbf{x}), & \mathbf{x} \mapsto x \\ \mathbf{x} \mapsto g(\mathbf{x}, f(\mathbf{y}, \mathbf{x})), & \mathbf{x} \mapsto f(\mathbf{x}, g(\mathbf{y}, \mathbf{x})) \\ \mathbf{x} \mapsto f(g(\mathbf{x}, \mathbf{y}), \mathbf{x}), & \mathbf{x} \mapsto g(f(\mathbf{y}, \mathbf{x}), \mathbf{x}) \\ \mathbf{x} \mapsto g(f(\mathbf{x}, \mathbf{y}), \mathbf{x}), & \mathbf{y} \mapsto f(g(\mathbf{y}, \mathbf{x}), \mathbf{y}) \\ \mathbf{y} \mapsto g(\mathbf{y}, f(\mathbf{y}, \mathbf{x})), & \mathbf{y} \mapsto f(\mathbf{y}, g(\mathbf{y}, \mathbf{x})) \\ \mathbf{y} \mapsto g(f(\mathbf{y}, \mathbf{x}), \mathbf{y}), & \mathbf{y} \mapsto y \\ \mathbf{y} \mapsto f(\mathbf{y}, g(\mathbf{x}, \mathbf{y})), & \mathbf{y} \mapsto g(\mathbf{y}, f(\mathbf{x}, \mathbf{y})) \\ \mathbf{y} \mapsto f(g(\mathbf{x}, \mathbf{y}), \mathbf{y}), & \mathbf{y} \mapsto g(f(\mathbf{x}, \mathbf{y}), \mathbf{y}) \end{array} \right\} \right).$$

- Generalizations contained in the language of this grammar are

x, $f(x, g(x, y))$, $f(g(y, x), x)$, $f(g(y, x), f(x, g(x, y)))$, $f(g(y, f(x, g(x, y))), f(x, g(x, y)))$, $f(f(x, g(x, y)), g(f(x, g(x, y)), y))$.

- Notice, $f(x, g(x, y)) \prec_{\mathsf{U}_2} f(f(x, g(x, y)), g(f(x, g(x, y)), y))$.

Future Work

- Many Open Questions and future research directions:
  - Is the procedure $\mathfrak{G}_{\mathsf{U}}$ complete for arbitrary unital theories?
  - Simplification of the one-unital procedure $\mathfrak{G}_{\mathsf{U(f)}}$.
  - Combining rules outlined in [Alpuente *et al.*, 2014] with our rules to produce procedures for restrictions of CU, AU, ACU.
  - Are unrestricted ACUI and UI infinitary or nullary?
  - Can the techniques used here and [Cerna and Kutsia, 2019] be generalized to AU for any collapse theory?
  - Are there non-trivial collapse theories of type unitary or finitary?
  - Investigating AU over algebraic structures such as Semirings.
    - Nullary in most cases [Cerna, 2020 (RISC Report)].
    - Open cases are most important.