On the Unification of Term Schemata

David M. Cerna, Alexander Leitsch, Anela Lolic

June 17^{th} 2020

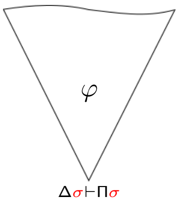
JOHANNES KEPLER UNIVERSITY LINZ

TU WIEN

Introduction

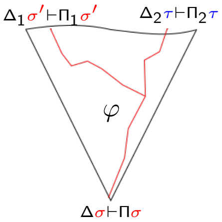
- Our motivation for studying unification over term sequences (term schemata) arose from our investigation concerning automated proof analysis in the presence of induction.
- An analysis of Fürstenburg's proof of the infinitude of primes was performed using a rudimentary schematic formalism and the first-order CERES method [Baaz *et al.*, 2008].
  - This analysis was performed without a formal framework for schematic objects.
- Since this early work several attempts have been made to develop a formal framework for proof sequences as well as their analysis [Dunchev *et al.*, 2013], [Leitsch *et al.*, 2017], [Cerna *et al.*, 2019].

Schematic Proofs in a Nutshell



φ has free variables instantiated by σ to numerals.

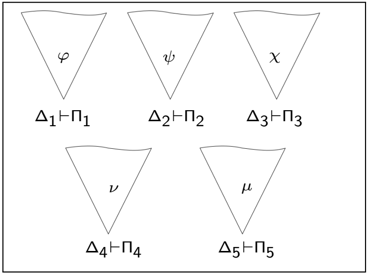
Schematic Proofs in a Nutshell

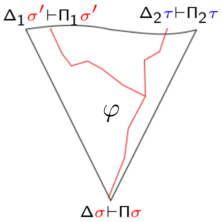


φ has free variables instantiated by σ to numerals.

$\sigma > \sigma'$ but no order relation between σ and τ.

Schematic Proofs in a Nutshell

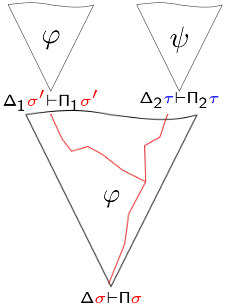


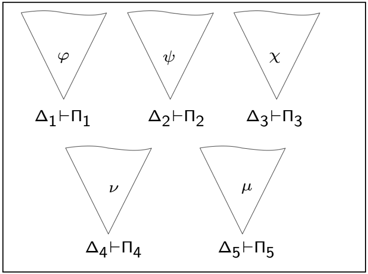


φ has free variables instantiated by σ to numerals.

$\sigma > \sigma'$ but no order relation between σ and τ.

Schematic Proofs in a Nutshell

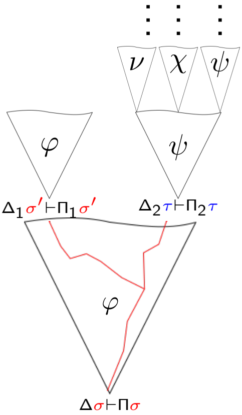




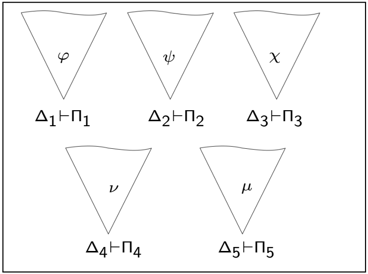
φ has free variables instantiated by σ to numerals.

$\sigma > \sigma'$ but no order relation between σ and τ.

Schematic Proofs in a Nutshell



Nested proof calls allowed.

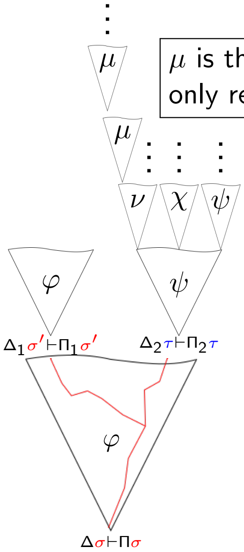


φ has free variables instantiated by σ to numerals.

$\sigma > \sigma'$ but no order relation between σ and τ.

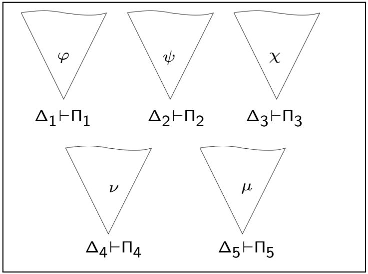
φ cannot be referenced in ψ, and ψ cannot be referenced in ν, etc.

Schematic Proofs in a Nutshell



μ is the least proof and only references itself.

Nested proof calls allowed.

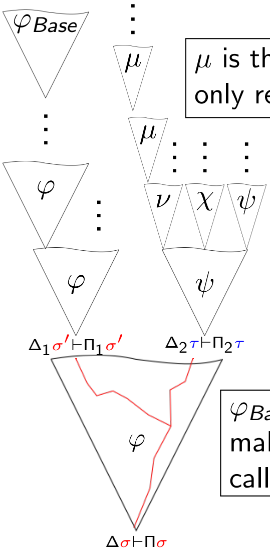


φ has free variables instantiated by σ to numerals.

$\sigma > \sigma'$ but no order relation between σ and τ.

φ cannot be referenced in ψ, and ψ cannot be referenced in ν, etc.

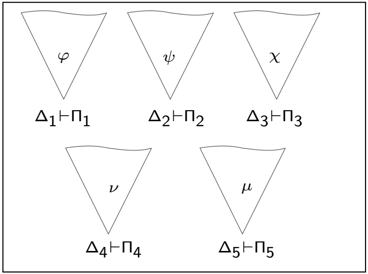
Schematic Proofs in a Nutshell



μ is the least proof and only references itself.

Nested proof calls allowed.

φ *Base* cannot make proof calls.



φ has free variables instantiated by σ to numerals.

$\sigma > \sigma'$ but no order relation between σ and τ.

φ cannot be referenced in ψ, and ψ cannot be referenced in ν, etc.

Analysis of Schematic Proofs

- With enough effort a mathematical proof can be formalized within a logical calculus such as Gentzen's sequent calculus.
- This formalization process will likely result in a proof with cut rather than an analytic proof.

$$\frac{C, \Delta \vdash \Gamma \qquad \Delta' \vdash \Gamma', C}{\Delta, \Delta' \vdash \Gamma, \Gamma'} \text{cut}$$

- Simulates lemmata and external theory introduction.
- Analysis of proofs concerns manipulation of the cut structure.
- Elimination of the cut structure results in an analytic proof.
- Such proofs provide additional mathematical understanding.

Global versus Local cut-elimination

Local cut-elimination reduces a cut formula's complexity or its distance from the leaves.

- Introduced by Gentzen as a method of proving consistency, the concept has been expanded well beyond the intended scope.

Global versus Local cut-elimination

Local cut-elimination reduces a cut formula's complexity or its distance from the leaves.

- Introduced by Gentzen as a method of proving consistency, the concept has been expanded well beyond the intended scope.

Global cut-elimination produces an intermediate representation of a formal proof's cut-structure.

- From this intermediate representation a new proof with a **trivial cut-structure** is produced.
- CERES transforms the cut structure into a NNF formula which can be refuted using resolution.

Local Cut-elimination and Schematic Proofs

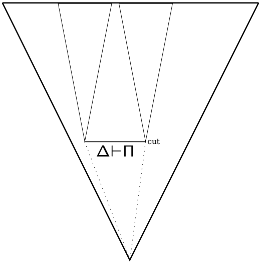
- No cut reduction rules exists for recursive calls.

$$\dfrac{\dfrac{(\varphi, \cdots)}{C, \Delta \vdash \Gamma} \qquad \dfrac{(\varphi_j, \cdots)}{\Delta' \vdash \Gamma', C}}{\Delta, \Delta' \vdash \Gamma, \Gamma'}\ \text{cut}$$

- When the call structure is non-recursive, proof references can just be removed.
- Recursive calls block reduction of a cut formula's position in the proof.
- Using a global approach we can extract the cut-structure as an unsatisfiable, recursive negation normal form formula.

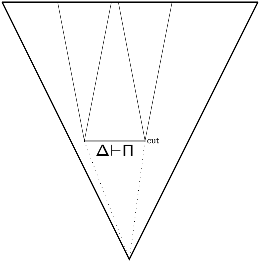
An Inductive Definition

Global Cut-elimination: Recursive NNF formula Extraction

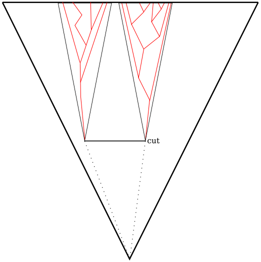


Proof with cuts

Global Cut-elimination: Recursive NNF formula Extraction

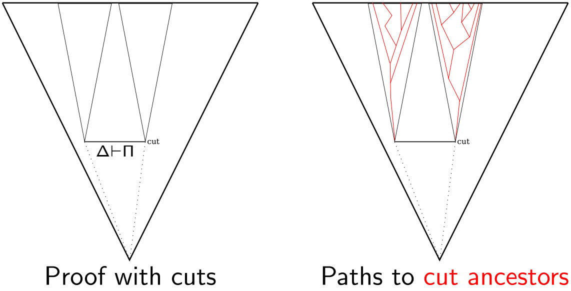


Proof with cuts



Paths to cut ancestors

Global Cut-elimination: Recursive NNF formula Extraction



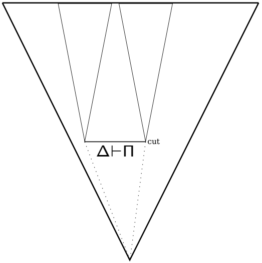
$CL(A \vdash A) \equiv \{A\}$

$CL(A \vdash A) \equiv \{\neg A\}$

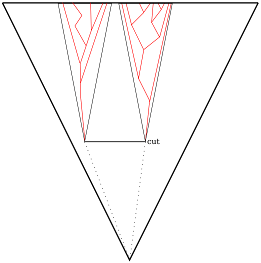
$CL(A \vdash A) \equiv \{\neg A \vee A\}$

- Proof references are denoted by defined symbols.
- Such a recursive NNF formula is always unsatisfiable.
- Proof analysis requires refuting this recursive NNF in a finitely representable way.

Global Cut-elimination: Recursive NNF formula Extraction



Proof with cuts

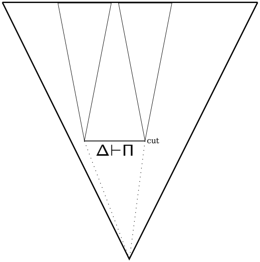


Paths to cut ancestors

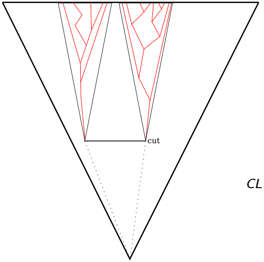
$$CL(A\vdash A)\equiv\{A\}$$
$$CL(A\vdash A)\equiv\{\neg A\}$$
$$CL(A\vdash A)\equiv\{\neg A\vee A\}$$
$$CL\left(\frac{\Delta\vdash\Pi}{\Delta'\vdash\Pi'}\ \rho\right)\equiv CL(\Delta\vdash\Pi)$$

- Proof references are denoted by defined symbols.
- Such a recursive NNF formula is always unsatisfiable.
- Proof analysis requires refuting this recursive NNF in a finitely representable way.

Global Cut-elimination: Recursive NNF formula Extraction



Proof with cuts



Paths to cut ancestors

$$CL(A \vdash A) \equiv \{A\}$$

$$CL(A \vdash A) \equiv \{\neg A\}$$

$$CL(A \vdash A) \equiv \{\neg A \vee A\}$$

$$CL\left(\frac{\Delta \vdash \Pi}{\Delta' \vdash \Pi'}\ \rho\right) \equiv CL(\Delta \vdash \Pi)$$

$$CL\left(\frac{\Delta \vdash \Pi \qquad \Delta' \vdash \Pi'}{\Delta'' \vdash \Pi''}\ \rho\right) \equiv$$

$$\begin{cases} CL(\Delta \vdash \Pi) \wedge CL(\Delta' \vdash \Pi') \\ CL(\Delta \vdash \Pi) \vee CL(\Delta' \vdash \Pi') \end{cases}$$

- Proof references are denoted by defined symbols.
- Such a recursive NNF formula is always unsatisfiable.
- Proof analysis requires refuting this recursive NNF in a finitely representable way.

Example of an Recursive NNF Formula

$$
\begin{aligned}
\hat{O}(x,y,n,m) &\Longrightarrow \hat{D}(x,n,m) \wedge\ \hat{P}(x,y,n,m) \\
\hat{D}(x,n,0) &\Longrightarrow f(x) = \hat{S}(n,a)\ \vee\ f(x) < \hat{S}(n,a) \\
\hat{D}(x,n,s(m)) &\Longrightarrow f(\hat{S}(s(m),x) = \hat{S}(n,a)\ \vee\ f(x) < \hat{S}(n,a)) \wedge\ \hat{D}(x,n,m) \\
\hat{P}(x,y,0,m) &\Longrightarrow \hat{C}(y,0,m)\ \wedge\ f(a) \not< 0 \\
\hat{P}(x,y,s(n),m) &\Longrightarrow (\hat{C}(y,s(n),m))\ \wedge (\hat{T}(x,n,m))\ \wedge\ \hat{P}(x,z,n,m) \\
\hat{C}(x,n,0) &\Longrightarrow f(x) \neq \hat{S}(n,a) \\
\hat{C}(x,n,s(m)) &\Longrightarrow f(\hat{S}(s(m),x)) \neq \hat{S}(n,a)\ \vee\ \hat{C}(x,n,m) \\
\hat{T}(x,n,0) &\Longrightarrow f(x) \not< \hat{S}(s(n),a)\ \vee\ f(x) = \hat{S}(n,a)\ \vee\ f(x) < \hat{S}(n,a) \\
\hat{T}(x,n,s(m)) &\Longrightarrow f(\hat{S}(s(m),x)) \not< \hat{S}(s(n),a)\ \vee\ f(\hat{S}(s(m),x)) = \hat{S}(n,a) \vee \\
&\qquad f(x) < \hat{S}(n,a) \wedge\ \hat{T}(x,n,m) \\
\hat{S}(0,x) &\Longrightarrow x \\
\hat{S}(s(n),x) &\Longrightarrow suc(\hat{S}(s(n),x))
\end{aligned}
$$

Refuting Recursive NNF Formula

- In [Leitsch *et al.*, 2017], the n-clause calculus of [Aravantinos *et al.*, 2013] was used to construct refutations.
  - This method is completely automated.
- However, the method is limited in scope and most problems of mathematical interest lie beyond it.
  - The above NNF is an example.
- In [Cerna *et al.*, 2019], we considered a semi-automated approach to dealing with the recursive NNF formula and designed a resolution calculus for this purpose.
- This required developing a theory of unification over schematic terms, what we will discuss for the rest of this talk.

Term Schema

- In a general sense, term schemata are nothing more than a finite representation of an infinite sequence of first-order terms s_n where n is a numeric parameter.
- Syntactically, this sequence is represented by a term $\hat{s}(n)$ where $\hat{s}$ may contain defined symbols indexed by the numeric parameter n.
- Defined symbols have associated defining equations allowing normalization to defined symbol-free first-order terms upon parameter instantiation.
- Normalization is denoted $\hat{s}(n)\downarrow_{\sigma} = s_{\alpha}$ where $\sigma = \{n \to \alpha\}$ is a parameter substitution.

Substitution Schemata

- Similarly, substitution schemata are finite representations of infinite sequences of first-order substitutions σ_n.
- Syntactically, substitution schemata are denoted by $\hat{\lambda}(n)$ and are sets of bindings $\{X(n) \rightarrow \hat{s}(n)\}$ were $\hat{s}(n)$ is a term schema.
- Normalization of a substitution schema reduces to normalization of its bindings and is denoted by $\hat{\lambda}(n)\downarrow_\sigma$.
- Index variables are referred to as global variables and will be discussed in greater detail shortly.
- The bindings used to construct a substitution schema may also be of the form $\{x \rightarrow \hat{s}(n)\}$.

Unification of Term Schema

Definition

Given two term schemata $\hat{s}(n)$ and $\hat{t}(n)$ we define $\hat{s}(n), \hat{t}(n)$ as unifiable if there exists a substitution schema $\hat{\lambda}(n)$ such that $\hat{s}(n)\hat{\lambda}(n)\downarrow_\sigma = \hat{t}(n)\hat{\lambda}(n)\downarrow_\sigma$ for all assignments σ.

- Depending on the types of variables allowed in $\hat{\lambda}(n)$ and in $\hat{s}(n)$ and $\hat{t}(n)$, we define two types of unification problems.
  - **Simple term schema unification**: Only first-order variables occur in $\hat{\lambda}(n)$, $\hat{s}(n)$, and $\hat{t}(n)$
  - **Global term schema unification**: Global variables may occur in $\hat{\lambda}(n)$, $\hat{s}(n)$, and $\hat{t}(n)$.

Simple Term Schema

- We assume a well-founded order $<$ on the defined symbols.

Definition

Let $\vec{x}$ be a tuple of first-order variables and n be a numeric parameter. A simple term schema is defined by primitive recursive definitions of the form

$$\begin{aligned} \hat{f}(\vec{x}, \mathbf{0}) &= g(\vec{x}), \\ \hat{f}(\vec{x}, s(n)) &= h(\vec{x}, n, z)\{z \leftarrow \hat{f}(\vec{x}, n)\} \end{aligned}$$

where $g(\vec{x})$ is a term over the variables $\vec{x}$ and $h(\vec{x}, n, z)$ is a term over the variables $\vec{x}, z$ and the parameter n. If $\hat{f}$ is not a $<$-minimal defined symbol then both $g(\vec{x})$ and $h(\vec{x}, n, z)$ may contain defined symbols $\hat{u}$ with $\hat{u} < \hat{f}$.

- We now provide a few examples.

Example: Simple Term Schema

- The defining equations $\hat{f}$, $\hat{f_1}$, and $\hat{\mathbf{g}}$ are as follows:

$$\begin{aligned} \hat{f}(x,0) &= h(a,a) & \hat{f}(x,s(n)) &= h(x,\hat{f}(x,n)) \\ \hat{f_1}(x,y,0) &= h(a,a) & \hat{f_1}(x,y,s(n)) &= h(x,\hat{f}(y,n)) \\ \hat{\mathbf{g}}(x,y,0) &= h(a,a) & \hat{\mathbf{g}}(x,y,s(n)) &= h(\hat{\mathbf{g}}(x,y,n),y) \end{aligned}$$

- Note that $\hat{f_1} > \hat{f}$.
- The relation between $\hat{\mathbf{g}}$ and the other symbols is irrelevant.
- Consider the parameter assignment $\sigma = \{n \rightarrow 2\}$ and the evaluation of $\hat{f_1}(x,y,n)$:

$$\hat{f_1}(x,y,n){\downarrow_\sigma} = \hat{f_1}(x,y,2){\downarrow} = h(x,\hat{f}(y,1){\downarrow}) = h(x,h(y,\hat{f}(x,0){\downarrow}))$$

$$= h(x,h(y,h(a,a))$$

Simple Term Schema Unification

- using the term schema $\hat{f}$,$\hat{f}_1$, and $\hat{\mathbf{g}}$ we can define unification problems such as:

$$\hat{f}(x,n) \stackrel{?}{=} \hat{g}(y,y,n)$$

- Note that $\sigma_0 = \{n \rightarrow 0\}$ and $\sigma_1 = \{n \rightarrow 1\}$ This problem evaluates to

$$\hat{f}(x,n)\downarrow_{\sigma_0} \stackrel{?}{=} \hat{\mathbf{g}}(y,y,n)\downarrow_{\sigma_0} \quad \Rightarrow \quad h(a,a) \stackrel{?}{=} h(a,a)$$

$$\hat{f}(x,n)\downarrow_{\sigma_1} \stackrel{?}{=} \hat{\mathbf{g}}(y,y,n)\downarrow_{\sigma_1} \quad \Rightarrow \quad h(x,h(a,a)) \stackrel{?}{=} h(h(a,a),y)$$

  Both of which are unifiable.
- However, for $\sigma_2 = \{n \rightarrow 2\}$ it evaluates to

$$h(x,h(x,h(a,a))) \stackrel{?}{=} h(h(h(a,a),y),y)$$

- After two steps unification fails due to occurs check.

Simple Term Schema Unification

- For two term schemata to be unifiable, they must be unifiable for all parameter assignments.
- The following unification problem is actually unifiable:

$$\hat{f}_1(x, y, s(n)) \stackrel{?}{=} \hat{g}(z, z, s(n))$$

- Let us evaluate the term schema for $\sigma_2 = \{n \rightarrow 2\}$:

$$h(x, h(y, h(y, h(a, a)))) = h(h(h(h(a, a), z), z), z)$$

- A unifier (also **MGU**) for this problem is as follows:

$$\theta = \{x \leftarrow h(h(h(a, a), h(y, h(y, h(a, a)))), h(y, h(y, h(a, a)))),$$
$$z \leftarrow h(y, h(y, h(a, a)))\}.$$

- The substitution schema (also the MGU**S**chema) is as follows:

$$\hat{\vartheta}(n) = \{x \leftarrow \hat{\mathbf{g}}(\hat{f}(y, n), \hat{f}(y, n), n),\ z \leftarrow \hat{f}(y, n)\}.$$

From Simple to Global Term Schemata

- Notice that simple term schemata repeat a finite set of variables arbitrarily often.
- This results in occurrence check failure in many cases.
- Usually, we do not desire all variables occurrences to be the same nor do we desire them to all be different.
- These extreme cases can be described through quantification:

$$\hat{f}(x,n) \quad \equiv \quad \forall x h(x, h(x, \cdots, h(x, h(a,a)) \cdots))$$

$$\hat{f}(x,n) \quad \equiv \quad \forall x_1, \cdots x_n h(x_1, h(x_2, \cdots, h(x_n, h(a,a)) \cdots))$$

- Global variables, variables taking numeric arguments,
- are a way of integrating indexing into the object language.
- This allows us to syntactically describe properties of the quantifier prefix.
- Additionally, it reduces unwanted occurrence check failure.

Global Term Schemata

Definition

Let $\vec{X}$ be a tuple of global variables and n be a numeric parameter. A global term schema is defined by primitive recursive definitions of the form

$$\begin{array}{rcl} \hat{f}(\vec{X}, \mathbf{0}) & = & t(\vec{X}), \\ \hat{f}(\vec{X}, s(n)) & = & s(\vec{X}, n, z)\{z \leftarrow \hat{f}(\vec{X}, n)\} \end{array}$$

where $t(\vec{X})$ is a term over the global variables $\vec{X}$ and $s(\vec{X}, n, z)$ is a term over the global variables $\vec{X}$, the individual variable z and the parameter n. If $\hat{f}$ is not a minimal defined symbol then both $t(\vec{X})$ and $s(\vec{X}, n, z)$ may contain defined symbols $\hat{u}$ with $\hat{u} < \hat{f}$.

- Notice that the domain of a unifier of global term schemata is dependent on the numeric parameter.
- We refer to such unifiers as s-unifiers (schematic unifiers).

From Simple to Global Term Schemata

- Consider the following global term schema:

$$\hat{f}(X,\mathbf{0}) = h(a, X(0)) \qquad \hat{f}(X,s(n)) = h(X(s(n)),\hat{f}(X,n))$$

$$\hat{g}(X,\mathbf{0}) = h(X(0), a) \qquad \hat{g}(X,s(n)) = h(\hat{g}(X,n),X(s(n)))$$

- Consider the unification problem $\hat{f}(X,s(n)) \stackrel{?}{=} \hat{g}(X,s(n))$.
- Evaluating this problem using $\sigma = \{n \to 1\}$ results in

$$h(X(2), h(X(1), h(a, X(0))) \stackrel{?}{=} h(h(h(X(0), a), X(1)), X(2))$$

- Note that the following is a unifier:

$$\theta = \{X(0) \to a\ ,\ X(1) \to h(a,a)\ ,\ X(2) \to h(h(a,a), h(a,a))\}$$

- The s-unifier is $\hat{\theta}(n) = \bigcup_{i=0}^{n}\{X(i) \to \hat{h}(i)\}$ where

$$\hat{h}(\mathbf{0}) = a \qquad \hat{h}(s(n)) = h(\hat{h}(n),\hat{h}(n))$$

Standard Term Schema

- The above term schema definitions do not have the full expressive power of primitive recursion.
- The following defined functions increase the expressivity.

Definition

A term schema is called a standard schema if it contains

- equations of the form $\hat{g}[\alpha, i](x_1, \ldots, x_\alpha) = x_i$ (where $1 \leq i \leq \alpha$) for every projection function $I_i^\alpha \colon \iota^\alpha \to \iota$ where $I_i^\alpha(\beta_1, \ldots, \beta_n) = \beta_i$ and
- equations of the form $\hat{h}[\alpha, c](x_1, \ldots, x_\alpha) = c$ for every constant function of type $\iota^\alpha \to \iota$.

Here $\hat{g}[\alpha, i], \hat{h}[\alpha, c]$ are α-ary defined function symbols.

Open Questions

- *Is the unification problem for simple standard schemata of recursion depth ≤ 1 decidable?*
  - Would need to show that it is reducible to the equivalence problem of LOOP-1 programs.
- *Is the unification problem for global standard schemata of recursion depth ≤ 1 decidable?*
  - Similar to first question but the unifier's domain size may vary.
- *Is the unification problem for simple and/or global free schemata decidable?*
  - While less expressive than free schemata, this type of unification shows up in the resolution calculus described in [Cerna *et al.*, 2019].
- We conjecture that all three problems are decidable.

Conclusions

- Term schemata provide an interesting unification problem motivated by computational proof analysis.
- We have yet to consider the equational variants of the above mentioned unification problems.
- It is also unclear if it is a variant of an existing unification problem such as term-graph unification or higher-order unification. Though we doubt the latter case.
- The notion of an MGU schema as well as how s-unifiers may be put in relation has not yet been formalized. Currently we are investigating this matter.