An Ordering for Flexible and Finite Representation of Infinite Sequences of Proofs

David M. Cerna
Recent work with Alexander Leitsch and Anela Lolic

October 23rd, 2019

JOHANNES KEPLER UNIVERSITY LINZ

An Introduction

- This presentation is similar to the previous two in that:
  1) Motivated by the study of cut-elimination.
  2) Connected to our investigations of Herbrand's theorem in the presence of induction through schematic proof representation.
- Unlike the previous talks we are going to focus on schematic proof representation.
- We introduce an ordering developed specifically for finitely representing infinite sequences of proofs.
- Furthermore, we discuss uses beyond proof representation.
- But first let's cover some background material.

cut-elimination on inductive proofs?

- Induction is of crucial importance in mathematics.
- Tere exist forms of cut-elimination for inductive proofs.
- But Herbrand's theorem fails.
- solution: Replace inductive proofs by proof schemata and compute Herbrand systems.
- The analysis of Fürstenberg's prime proof by CERES was based on proof schemata.

Inductive proofs: extraction of Herbrand sequents?

Example: Let us consider the sequent S:

$$\forall x(P(x) \rightarrow P(f(x))) \vdash \forall n \forall x((P(\hat{g}(n,x)) \rightarrow Q(x)) \rightarrow (P(x) \rightarrow Q(x)))$$

where f is a unary function symbol and

$$\mathcal{E} = \{\hat{g}(0,x) = x, \hat{g}(s(n),x) = f(\hat{g}(n,x))\}.$$

The skolemized version:

$$S'\colon (\forall x)(P(x) \rightarrow P(f(x))) \vdash (P(\hat{g}(n_0,c)) \rightarrow Q(c)) \rightarrow (P(c) \rightarrow Q(c)).$$

Obviously, S' cannot be proven without induction - S' does not have a Herbrand sequent (w.r.t. the theory $\mathcal{E}$) .

- S' can be proven by induction - you prove the lemma

$$\psi\colon (\forall x)(P(x) \rightarrow P(f(x))) \vdash \forall n \forall x(P(x) \rightarrow P(\hat{g}(n,x)).$$

Proof Schemata Define Herbrand Schemata

$\psi(0) =$

$$\cfrac{\cfrac{\cfrac{\cfrac{P(\hat{g}(0,x_0)) \vdash P(\hat{g}(0,x_0))}{P(x_0) \vdash P(\hat{g}(0,x_0))}\ \mathcal{E}}{\vdash P(x_0) \to P(\hat{g}(0,x_0))}\ \to: r}{\vdash (\forall x)(P(x) \to P(\hat{g}(0,x)))}\ \forall: r}{(\forall x)(P(x) \to P(f(x))) \vdash (\forall x)(P(x) \to P(\hat{g}(0,x)))}\ w: l$$

and $\psi(k+1)$ is:

$$\cfrac{\dfrac{(\psi(k))}{(\forall x)(P(x) \to P(f(x))) \vdash (\forall x)(P(x) \to P(\hat{g}(\bar{k},x)))} \qquad (1)}{(\forall x)(P(x) \to P(f(x))) \vdash (\forall x)(P(x) \to P(\hat{g}(s(\bar{k}),x)))}\ cut, c: l$$

where (1) is:

$$
\begin{prooftree}
\AxiomC{$P(x_{k+1}) \vdash P(x_{k+1})$}
\AxiomC{$P(\hat{g}(\bar{k}, x_{k+1})) \vdash P(\hat{g}(\bar{k}, x_{k+1}))$}
\AxiomC{$P(\hat{g}(s(\bar{k}), x_{k+1})) \vdash P(\hat{g}(s(\bar{k}), x_{k+1}))$}
\RightLabel{$\mathcal{E}$}
\UnaryInfC{$P(f(\hat{g}(\bar{k}, x_{k+1}))) \vdash P(\hat{g}(s(\bar{k}), x_{k+1}))$}
\RightLabel{$\rightarrow: l$}
\BinaryInfC{$P(\hat{g}(\bar{k}, x_{k+1})), P(\hat{g}(\bar{k}, x_{k+1})) \rightarrow P(f(\hat{g}(\bar{k}, x_{k+1}))) \vdash P(\hat{g}(s(\bar{k}), x_{k+1}))$}
\RightLabel{$\forall: l$}
\UnaryInfC{$P(\hat{g}(\bar{k}, x_{k+1})), (\forall x)(P(x) \rightarrow P(f(x))) \vdash P(\hat{g}(s(\bar{k}), x_{k+1}))$}
\RightLabel{$\rightarrow: l$}
\BinaryInfC{$P(x_{k+1}), P(x_{k+1}) \rightarrow P(\hat{g}(\bar{k}, x_{k+1})), (\forall x)(P(x) \rightarrow P(f(x))) \vdash P(\hat{g}(s(\bar{k}), x_{k+1}))$}
\RightLabel{$\rightarrow: r$}
\UnaryInfC{$P(x_{k+1}) \rightarrow P(\hat{g}(\bar{k}, x_{k+1})), (\forall x)(P(x) \rightarrow P(f(x))) \vdash P(x_{k+1}) \rightarrow P(\hat{g}(s(\bar{k}), x_{k+1}))$}
\RightLabel{$\forall: l$}
\UnaryInfC{$(\forall x)(P(x) \rightarrow P(\hat{g}(\bar{k}, x))), (\forall x)(P(x) \rightarrow P(f(x))) \vdash P(x_{k+1}) \rightarrow P(\hat{g}(s(\bar{k}), x_{k+1}))$}
\RightLabel{$\forall: r$}
\UnaryInfC{$(\forall x)(P(x) \rightarrow P(\hat{g}(\bar{k}, x))), (\forall x)(P(x) \rightarrow P(f(x))) \vdash (\forall x)(P(x) \rightarrow P(\hat{g}(s(\bar{k}), x)))$}
\end{prooftree}
$$

Proof Schemata Define Herbrand Schemata

and define $\varphi(n) =$

$$\dfrac{\dfrac{(\psi(n))}{(\forall x)(P(x) \to P(f(x))) \vdash C(n)} \qquad \dfrac{(\chi(n))}{C(n) \vdash (P(\hat{g}(\bar{n}, c)) \to Q(c)) \to (P(c) \to Q(c))}}{(\forall x)(P(x) \to P(f(x))) \vdash (P(\hat{g}(\bar{n}, c)) \to Q(c)) \to (P(c) \to Q(c))}\ cut$$

where $C(n) = (\forall x)(P(x) \to P(\hat{g}(\bar{n}, x)))$ and $\chi(n)$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{P(c) \vdash P(c) \qquad \dfrac{P(\hat{g}(\bar{n}, c)) \vdash P(\hat{g}(\bar{n}, c)) \qquad Q(c) \vdash Q(c)}{P(\hat{g}(\bar{n}, c)) \to Q(c), P(\hat{g}(\bar{n}, c)) \vdash Q(c)}\ \to: l}{P(c), P(\hat{g}(\bar{n}, c)) \to Q(c), P(c) \to P(\hat{g}(\bar{n}, c)) \vdash Q(c)}\ \to: l}{P(\hat{g}(\bar{n}, c)) \to Q(c), P(c) \to P(\hat{g}(\bar{n}, c)) \vdash P(c) \to Q(c)}\ \to: r}{P(c) \to P(\hat{g}(\bar{n}, c)) \vdash (P(\hat{g}(\bar{n}, c)) \to Q(c)) \to (P(c) \to Q(c))}\ \to: r}{(\forall x)(P(x) \to P(\hat{g}(\bar{n}, x))) \vdash (P(\hat{g}(\bar{n}, c)) \to Q(c)) \to (P(c) \to Q(c))}\ \forall: l$$

proof schemata define Herbrand schemata

$$\dfrac{\dfrac{\psi(n)}{\forall x(P(x) \rightarrow P(f(x))) \vdash C} \qquad \dfrac{\chi(n)}{C \vdash (P(\hat{g}(\bar{n}, c)) \rightarrow Q(c)) \rightarrow (P(c) \rightarrow Q(c))}}{S_n \colon \forall x(P(x) \rightarrow P(f(x))) \vdash (P(\hat{g}(\bar{n}, c)) \rightarrow Q(c)) \rightarrow (P(c) \rightarrow Q(c))}\ cut$$

for $C = \forall x(P(x) \rightarrow P(\hat{g}(\bar{n}, x)))$

For every n we get (by cut-elimination) a Herbrand sequent S_n^* (valid in $\mathcal{E} = \{\hat{g}(0, x) = x, \hat{g}(s(n), x) = f(\hat{g}(n, x))\}$):

$$\begin{aligned} S_0^* &= \vdash (P(\hat{g}(0, c)) \rightarrow Q(c)) \rightarrow (P(c) \rightarrow Q(c)), \\ S_{n+1}^* &= (P(x) \rightarrow P(f(x)))\theta_0, \ldots, (P(x) \rightarrow P(f(x))\theta_n \vdash \\ &\quad P(\hat{g}(n+1, c)) \rightarrow Q(c)) \rightarrow (P(c) \rightarrow Q(c)). \end{aligned}$$

for $\theta_0 = \{x \leftarrow \hat{g}(n, 0)\}, \ldots, \theta_n = \{x \leftarrow \hat{g}(n, c)\}$

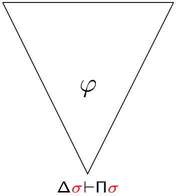
Herbrand substitution set $\rightarrow$ Herbrand system:

$$\Theta_0 = \emptyset, \quad \Theta_{n+1} = \Theta_n \cup \{\theta_n\}.$$

Beyond the Basic Example

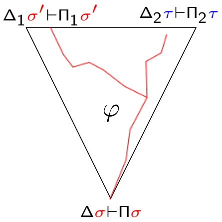
- The existing schematic version of CERES is defined for proof schema with a single free-parameter.
- Constructing proof schema which use more than one free-parameter is possible.
- It can also be shown that multi-parameter proof schemata are provability equivalent to the LK-calculus for Peano Arithmetic [Cerna,2018] (If we allow quantification of parameters).
- Though, there does not exists a corresponding CERES method. This is currently being investigated.

Quick Guide to Schematic Proof Construction



φ has free variables instantiated by σ to numerals.

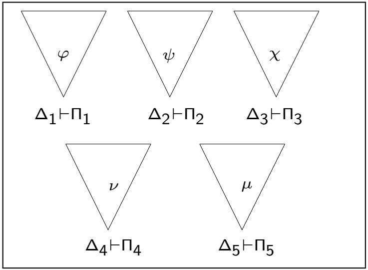
Quick Guide to Schematic Proof Construction

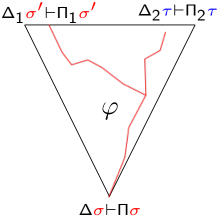


φ has free variables instantiated by σ to numerals.

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

Quick Guide to Schematic Proof Construction

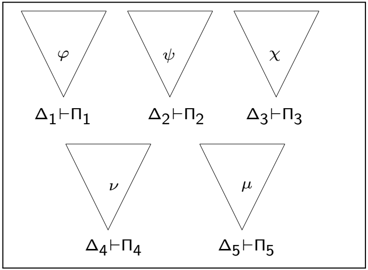


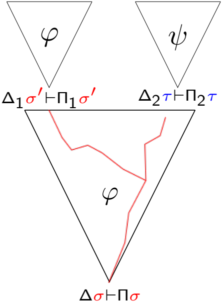


φ has free variables instantiated by σ to numerals.

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

Quick Guide to Schematic Proof Construction

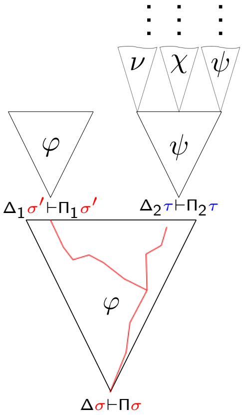




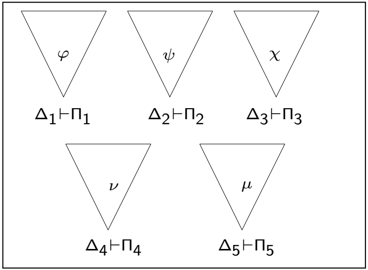
φ has free variables instantiated by σ to numerals.

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

Quick Guide to Schematic Proof Construction



Nested proof calls allowed.

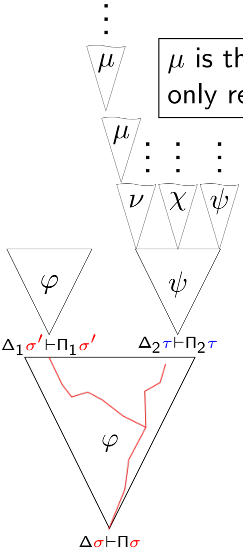


φ has free variables instantiated by σ to numerals.

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

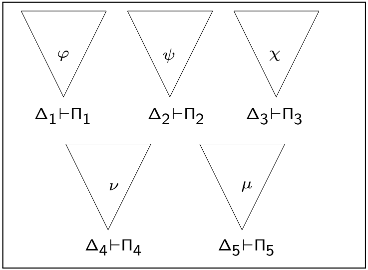
φ cannot be referenced in ψ, and ψ cannot be referenced in ν, etc.

Quick Guide to Schematic Proof Construction



μ is the least proof and only references itself.

Nested proof calls allowed.

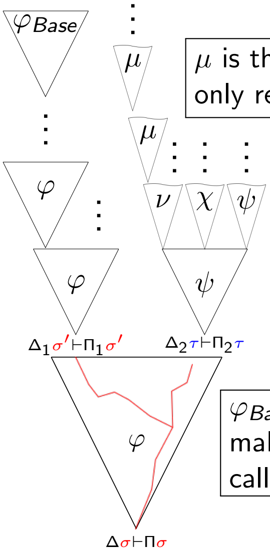


φ has free variables instantiated by σ to numerals.

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

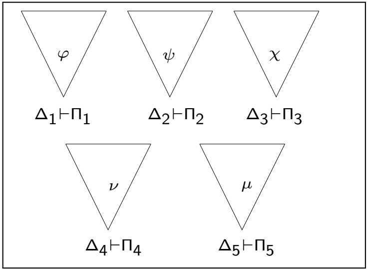
φ cannot be referenced in ψ, and ψ cannot be referenced in ν, etc.

Quick Guide to Schematic Proof Construction



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.

Problems with Schematic proof Construction

- While the above schematic proof construction works and is provability equivalent to Peano arithmetic, it does not help much with proof analysis.
- A sequence of post-transformation proofs may require a more expressive language then the one outlined above to describe it.
- Let us consider cut-elimination for schematic proofs before looking into the properties of this more expressive language.

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.

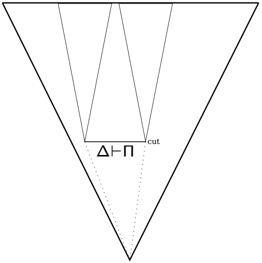
Local Cut-elimination and Schematic Proofs

- No cut reduction rules exists for links.

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

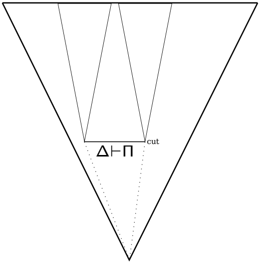
- When the call structure is non-recursive proof references can just be removed.
- However, recursive call structures block reduction of a cut formula's rank.
- Using a global approach we can extract the cut-structure as an unsatisfiable negation normal form inductive definition.

Global Cut-elimination Inductive NNF Definition Extraction

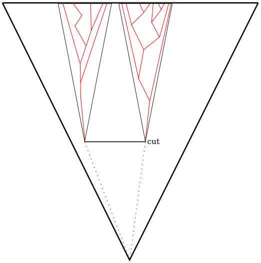


LK-Proof with cuts

Global Cut-elimination Inductive NNF Definition Extraction

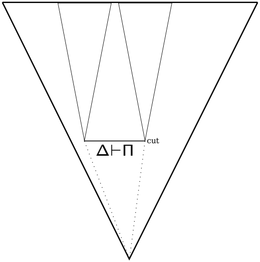


LK-Proof with cuts

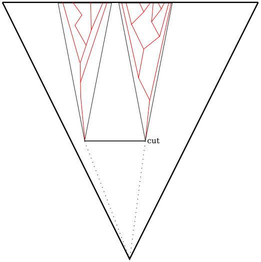


Paths to cut ancestors

Global Cut-elimination Inductive NNF Definition Extraction



LK-Proof with cuts

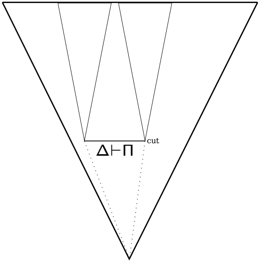


Paths to cut ancestors

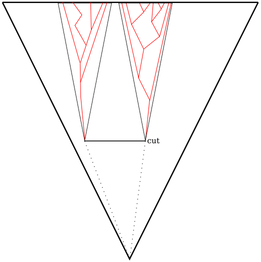
$CL(A \vdash A) \equiv \{A\}$
$CL(A \vdash A) \equiv \{\neg A\}$
$CL(A \vdash A) \equiv \{\neg A \vee A\}$

- Proof references are denoted by defined symbols.
- Such an inductive NNF definition is always unsatisfiable.
- We need to refute this inductive NNF in a finitely representable way in-order to extract Herbrand information.

Global Cut-elimination Inductive NNF Definition Extraction



LK-Proof with cuts

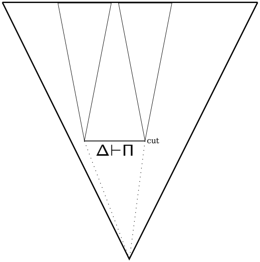


Paths to cut ancestors

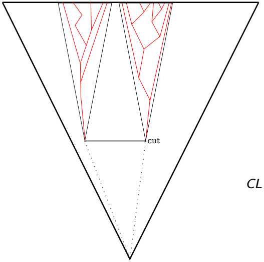
$$CL(A \vdash A) \equiv \{A\}$$
$$CL(A \vdash A) \equiv \{\neg A\}$$
$$CL(A \vdash A) \equiv \{\neg A \vee A\}$$
$$CL\left(\frac{\Delta \vdash \Pi}{\Delta' \vdash \Pi'}\ \rho\right) \equiv CL(\Delta \vdash \Pi)$$

- Proof references are denoted by defined symbols.
- Such an inductive NNF definition is always unsatisfiable.
- We need to refute this inductive NNF in a finitely representable way in-order to extract Herbrand information.

Global Cut-elimination Inductive NNF Definition Extraction



LK-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 an inductive NNF definition is always unsatisfiable.
- We need to refute this inductive NNF in a finitely representable way in-order to extract Herbrand information.

Refuting the NNF

- Note that refuting this NNF for every instance is complexity-wise equivalent to proving the inductive statement using atomic cuts only.
- Given that cut-elimination for calculi with an induction inference is usually not possible or not possible while retaining analyticity this alludes to the difficulty of the problem.
- Let us look at an example before considering how to represent it's refutation.

Eventually Constant Statement (ECS)

- Consider a proof schema indexed by n of

$$\forall x \bigvee_{i=0}^{n} f(x) = i, \Delta \vdash \exists x \forall y (x \leq y \rightarrow f(x) = f(y))$$

- using a sequence of Σ_2-cuts

$$\exists x \forall y (((x \leq y) \Rightarrow n+1 = f(y)) \vee f(y) < n+1).$$

- This example was discussed in [Cerna and Leitsch 2016] where the proof was skolemized resulting in the end sequent

$$\forall x \bigvee_{i=0}^{n} f(x) = i, \Delta \vdash \exists x (x \leq g(x) \rightarrow f(x) = f(g(x))).$$

- Note that Δ contains additional assumptions concerning f.

The 2-repetition Statement

- As an exercise we can consider what happens when we interpret g as the successor function

$$\forall x \bigvee_{i=0}^{n} f(x) = i, \Delta' \vdash \exists x(f(x) = f(suc(x))),$$

- i.e. what cut formula and which axioms are needed for this statement to hold. The cut is weaker,

$$\exists x(f(x) = k \wedge f(suc(x)) = k) \vee \forall x(f(x) < k)$$

- and the axioms concerning f are simpler

$$f(suc(x)) < s(k) \vdash f(suc(x)) = k, f(x) < k$$

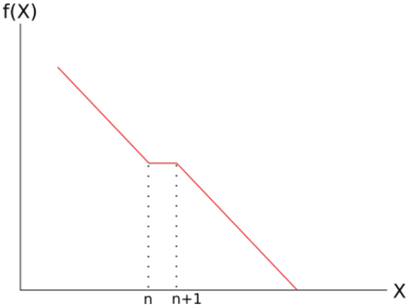
$$f(x) < s(k) \vdash f(x) = k, f(x) < k$$

The 2-repetition Statement

- A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is said to be 2-repeating if there exists at least two consecutive values $x, y \in \mathbb{N}$ s.t. $f(x) = f(y)$.

Assertion (2-repetition)

Every total monotonically decreasing function $f : \mathbb{N} \rightarrow \mathbb{N}$ is at least 2-repeating.



Analysis of the 2-repetition Statement

- Analysis of ECS was already performed in [Cerna and Leitsch 2016], the 2-repetition Statement is substantially weaker.
- While the inductive superposition prover of N. Peltier cannot find an invariant for ECS, it can find one for 2-repetition. [Leitsch et al. 2017] can be used for analysis.

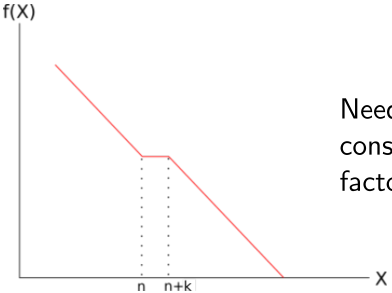
```
=============================== PROOF ==================================

% Proof 1 at 0.017 (+ 0.000) seconds.
% Given clauses 73.

% number of calls to fixpoint : 3
 S_init  :
 (51:  [ EQ(v0,f(h(v1))) | LE(f(v1),v0) if  n = s(v0) ].
50:  [ EQ(v0,f(v1)) | LE(f(v1),v0) if  n = s(v0) ].
33:  [ PHI(v0,v1) if  n = s(v0) ].
)
 S_loop  :
 (82:  [ EQ(v0,f(h(v1))) | LE(f(v1),v0) if  n = s(s(v0)) ].
80:  [ EQ(v0,f(v1)) | LE(f(v1),v0) if  n = s(s(v0)) ].
53:  [ PHI(v0,v1) if  n = s(s(v0)) ].
)
 The empty clauses  :
 (45:  [  n = 0 ].
81:  [  n = s(0) ].
112:  [  n = s(s(0)) ].
) max_rank 2
```

The k-repetition Statement

- A function $f : \mathbb{N} \to \mathbb{N}$ is said to be k-repeating if there exists at least k consecutive values $x_1, \cdots, x_k \in \mathbb{N}$ s.t. $f(x_1) = \cdots = f(x_k)$.

Assertion (k-repetition)

Every total monotonically decreasing function $f : \mathbb{N} \to \mathbb{N}$ is at least k-repeating.



Need to check all possible function constructions of which there are factorially many.

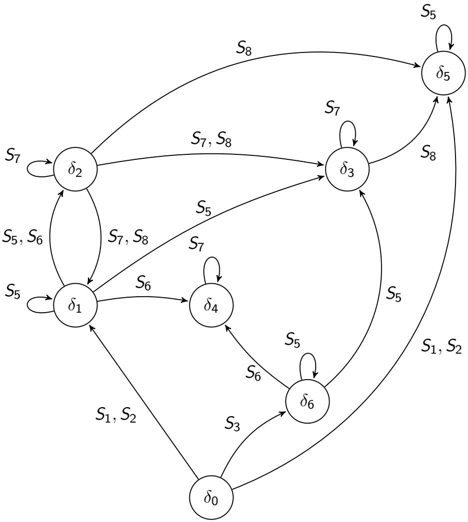
Cut Structure of k-Repetition as an Inductive Definition

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

Analysis of the k-repetition Statement

- The k-repetition statement has two independent parameters which is beyond the capabilities of N. Peltier's superposition prover and thus beyond the analysis method of [Leitsch et al. 2017]
- While two parameters need not be problematic, in the case of the k-repetition nested loops are needed to build the refutation.
- The schematic proof construction method discussed earlier cannot describe such a refutation.
- A more flexible proof construction mechanism is needed.

Call Structure of the Refutation



A more flexible formalism

- Rather than developing a formalism for proof schema in particular we provide a general scaffolding for recursive object construction.
  - For example, Proof schemata, Resolution Refutations, Schematic Unifiers, Herbrand Systems, etc.
- The scaffolding provides a "structural semantics" for finite representability.
- This scaffolding may be decorated by objects which match its structure.
- So what is this scaffolding made of?

Junctions

$$\left(\delta, \overrightarrow{t}_n\right)$$

- The above object is a junction, δ is a symbol from a set of symbols Δ, and $\overrightarrow{t}_n$ is a tuple of numeric terms which may contain parameters and primitive recursive functions.
- Junctions whose tuples of numeric terms are of the same length are well ordered by $<_n$.
- Junctions differing in tuple length must have different symbols.
- Junctions can also be well-ordered by the symbols $<_\Delta$ (by tuple size).
- We say $p \lhd q$ if either $q <_n p$ or $p <_\Delta q$.
- Note that $\lhd$ is not well founded.

Junctions

$$(\delta, n, m)$$

$$(\delta_2, n, s(m))$$

$$(\delta_2, n, s(m), k, 0)$$

$$(\delta_3, n, s(m), k, 0)$$

- The first two are valid Junctions.
- The third is not, same symbol as the second but different tuple length
- The fourth is also valid.
- Note, have an infinite set of parameters $\mathcal{P}$, a parameter assignment σ assigns a numeral to each parameter.
- The set of all parameter assignments is $\mathcal{S}$.

Flows

$$P_{S^*} = \left\{ \begin{array}{c} (A_1, S_1) \\ (A_2, S_2) \\ \dots \\ (A_n, S_n) \end{array} \right\}$$

- The A_i are sets of junctions.
- Let $\mathcal{S}$ be the set of parameter assignments, S^* is a partitioning of $\mathcal{S}$ into disjoint sets S_1 through S_n.
- If $\sigma \in S_i$ then $P_{S^*}(\sigma) = A_i$
- There is a is special junction $[P_{S^*}]$ s.t. $\forall \sigma \in \mathcal{S},\ [P_{S^*}] \in P_{S^*}(\sigma)$, The Source of the flow.
- Note, $\forall q \in A_i,\ q \neq [P_{S^*}]$ then $[P_{S^*}] \lhd q$

Example Flow

$$P_{S^*} = \left\{ \begin{array}{c} \left(\left\{ \begin{array}{c} (\delta_2, n, k), \\ (\delta_1, \hat{p}(n), s(k)), \\ (\delta_2, \hat{p}(n), s(k)), \\ (\delta_3, \hat{p}(n), s(k)) \end{array} \right\}, S_1 \right) \\ \\ \left(\left\{ \begin{array}{c} (\delta_2, n, k), \\ (\delta_3, \hat{p}(n), s(k)) \end{array} \right\}, S_2 \right) \end{array} \right.$$

- $S^* = S_1 \cup S_2$ where $S_1 = \{\sigma \in \mathcal{S} \ \& \ \sigma(n) \downarrow_\omega > 0\}$ and $S_2 = \{\sigma \in \mathcal{S} \ \& \ \sigma(n) \downarrow_\omega = 0\}$.
- If $\sigma \in S_1$ then $P_{S^*}(\sigma) = \{(\delta_2, n, k), (\delta_1, \hat{p}(n), s(k)), (\delta_2, \hat{p}(n), s(k)), (\delta_3, \hat{p}(n), s(k))\}$
- $[P_{S^*}] = (\delta_2, n, k)$
- $(\delta_2, n, k) \lhd (\delta_3, \hat{p}(n), s(k))$ where $\hat{p}$ is the predecessor function and $<_2$ is the lexicographical ordering.

Connecting flows

- Let $P_{S_1^*}$ and $P_{S_2^*}$ be flows and σ a parameter assignment such that
  - there exists $q \in P_{S_1^*}(\sigma)$ such that $\left[P_{S_1^*}\right] \neq q$ and
  - there exists a parameter assignment θ such that
  - $q\sigma = \left[P_{S_2^*}\right]\theta$ then

  we refer to the flows as linked.
- If a set of flows is completely linked together the result is a call graph.

Call graph definition

Definition

A finite set of flows $\mathcal{G}$ is referred to as a call graph if for every $P_{\mathcal{S}_1^*} \in \mathcal{G}$, $S \in \mathcal{S}^*$, $\sigma \in S$, $j \in P_{\mathcal{S}_1^*}$ there exists a unique $P_{\mathcal{S}_2^*} \in \mathcal{G}$ and $\theta \in \mathcal{S}$ s.t. $\theta([P_{\mathcal{S}_2^*}]) \downarrow_\omega = \sigma(j) \downarrow_\omega$. We write $\mathit{flow}(j, \sigma) = P_{\mathcal{S}_2^*}$ and $\mathit{subst}(j, \sigma) = \theta$.

- Essentially in a call graph every Junction under every parameter assignment is a the source of some flow.

Example Call Graph

- $\mathcal{G} = \{P_1, P_2\}$ is a call graph , where

$$P_1 = \{(\{(\delta, n), (\delta', n, p(n), n, 0)\}, \mathcal{S})\}$$

$$P_2 = \left\{ \begin{array}{c} (\{(\delta', n, m, k, w), (\delta', n, m, p(k), s(w))\}, S_1) \ , \\ (\{(\delta', n, m, k, w), (\delta', n, p(m), n, w)\}, S_2) \ , \\ (\{(\delta', n, m, k, w)\}, S_3) \end{array} \right\}$$

- $S_1 = \{\sigma | \sigma \in \mathcal{S}, \sigma(k) \downarrow_\omega > 0\}$,
- $S_2 = \{\sigma | \sigma \in \mathcal{S}, \sigma(k) \downarrow_\omega = 0 \ \& \ \sigma(m) \downarrow_\omega > 0\}$,
- $S_3 = \{\sigma | \sigma \in \mathcal{S}, \sigma(k) \downarrow_\omega = 0 \ \& \ \sigma(m) \downarrow_\omega = 0\}$
- Note that for $\sigma \in S_3$ $|P_2(\sigma)|$ such parameter assignments are referred to as sinks.

What does a call Graph define

- Starting from any parameter assignment a call graph provides a way to compute new parameter assignments until one reaches a sink.

$$[\sigma, subst(j_1,\sigma)]\,,[subst(j_1,\sigma), subst(j_2,subst(j_1,\sigma))]\,,$$

$$[subst(j_2,subst(j_1,\sigma)), subst(j_3, subst(j_2,subst(j_1,\sigma)))]\,,\cdots$$

- Let us consider the above call graph and how we may apply it to a given parameter assignment.

$$\left[\overbrace{\{n \leftarrow \alpha_1\}}^{\sigma}, \overbrace{\left\{\begin{array}{c} n \leftarrow \alpha_1\ ,\ m \leftarrow p(\alpha_1), \\ k \leftarrow \alpha_1\ ,\ w \leftarrow 0 \end{array}\right\}}^{\theta_1}\right]$$

where $j_1 = (\delta', n, p(n), n, 0)$, $flow(j_1,\sigma) = P_2$, and $Subst(j_1,\sigma) = \theta_1$.

What does a call Graph define

$$\left[\overbrace{\left\{\begin{array}{c} n \leftarrow \alpha_1 \ , \ m \leftarrow p(\alpha_1), \\ k \leftarrow \alpha_1, w \leftarrow 0 \end{array}\right\}}^{\theta_1}, \overbrace{\left\{\begin{array}{c} n \leftarrow \alpha_1 \ , \ m \leftarrow p(\alpha_1), \\ k \leftarrow p(\alpha_1), w \leftarrow s(0) \end{array}\right\}}^{\theta_2}\right]$$

where $j_2 = (\delta', n, m, p(k), s(w))$, $flow(j_2, \theta_1) = P_2$, and $Subst(j_2, \theta_1) = \theta_2$.

$$\left[\overbrace{\left\{\begin{array}{c} n \leftarrow \alpha_1 \ , \ m \leftarrow p(\alpha_1), \\ k \leftarrow p(\alpha_1) \ , \ w \leftarrow s(0) \end{array}\right\}}^{\theta_2}, \overbrace{\left\{\begin{array}{c} n \leftarrow \alpha_1 \ , \ m \leftarrow p(\alpha_1), \\ k \leftarrow p(p(\alpha_1)) \ , \ w \leftarrow s(s(0)) \end{array}\right\}}^{\theta_3}\right]$$

where $j_3 = (\delta', n, m, p(k), s(w))$, $flow(j_3, \theta_2) = P_2$, and $Subst(j_3, \theta_2) = \theta_3$.

$$\vdots$$

What does a call Graph define

$$\left[\overbrace{\left\{\begin{array}{c} n \leftarrow \alpha_1 \ , \ m \leftarrow p(\alpha_1), \\ k \leftarrow p^{\alpha_1}(\alpha_1) \ , \ w \leftarrow s^{\alpha_1}(0) \end{array}\right\}}^{\theta_3}, \overbrace{\left\{\begin{array}{c} n \leftarrow \alpha_1 \ , \ m \leftarrow p(p(\alpha_1)), \\ k \leftarrow \alpha_1 \ , \ w \leftarrow s^{\alpha_1}(0) \end{array}\right\}}^{\theta_4}\right]$$

$j_4 = (\delta', n, p(m), n, w)$, $flow(j_4, \theta_4) = P_2$, and $Subst(j_4, \theta_3) = \theta_4$.

$$\left[\overbrace{\left\{\begin{array}{c} n \leftarrow \alpha_1 \ , \ m \leftarrow p_1^{\alpha}(\alpha_1), \\ k \leftarrow 0 \ , \ w \leftarrow s^{(\alpha_1)^2}(0) \end{array}\right\}}^{\theta_4}\right]$$

Being that we have reached a sink at this Junction *flow* and *Subst* are only defined for the source of P_2.

Call Graph Traces

- The Call graph discussed above results in a simple linear structure when applied to a parameter assignment.
- In general the application of a call graph to a parameter assignment may result in branching.
- To capture this structure we defined Call Graph Traces.
- A trace is just a tree where each node is labeled by the junction the call graph passed during application to a parameter assignment.

Example: Branching Call Graph

- Consider the call graph $\mathcal{G} = \{P_1, P_2\}$ where the flows are defined as follows:

$$P_1 = \{ \ (\{(\delta, n), (\delta, p(n)), (\delta', n, n)\}, S_1) \ , \ (\{(\delta, n)\}, S_2) \ \}$$

$$P_2 = \{ \ (\{(\delta', n, m), (\delta', n, p(m))\}, S'_1) \ , \ (\{(\delta', n, m)\}, S'_2) \ \}$$

- $S_1 = \{\sigma \in \mathcal{S} \ \& \ \sigma(n) \downarrow_\omega > 0\}$
- $S_2 = \{\sigma \in \mathcal{S} \ \& \ \sigma(n) \downarrow_\omega = 0\}$
- $S'_1 = \{\sigma \in \mathcal{S} \ \& \ \sigma(m) \downarrow_\omega > 0\}$
- $S'_2 = \{\sigma \in \mathcal{S} \ \& \ \sigma(m) \downarrow_\omega = 0\}$

Trace for non-branching Call Graph

$$T(\mathcal{G}, P_1, \{n \leftarrow 2\}) = [(\delta, 2), T(\mathcal{G}, P_2, \{n \leftarrow 2, m \leftarrow 1, k \leftarrow 2, w \leftarrow 0\})]$$

$$T\left(\mathcal{G}, P_2, \left\{ \begin{array}{c} n \leftarrow 2, m \leftarrow 1, \\ k \leftarrow 2, w \leftarrow 0 \end{array} \right\}\right) = \left[(\delta', 2, 1, 2, 0), T\left(\mathcal{G}, P_2, \left\{ \begin{array}{c} n \leftarrow 2, m \leftarrow 1, \\ k \leftarrow 1, w \leftarrow 1 \end{array} \right\}\right)\right]$$

$$T\left(\mathcal{G}, P_2, \left\{ \begin{array}{c} n \leftarrow 2, m \leftarrow 1, \\ k \leftarrow 1, w \leftarrow 1 \end{array} \right\}\right) = \left[(\delta', 2, 1, 1, 1), T\left(\mathcal{G}, P_2, \left\{ \begin{array}{c} n \leftarrow 2, m \leftarrow 1, \\ k \leftarrow 0, w \leftarrow 2 \end{array} \right\}\right)\right]$$

$$T\left(\mathcal{G}, P_2, \left\{ \begin{array}{c} n \leftarrow 2, m \leftarrow 1, \\ k \leftarrow 0, w \leftarrow 2 \end{array} \right\}\right) = \left[(\delta', 2, 1, 0, 2), T\left(\mathcal{G}, P_2, \left\{ \begin{array}{c} n \leftarrow 2, m \leftarrow 0, \\ k \leftarrow 2, w \leftarrow 2 \end{array} \right\}\right)\right]$$

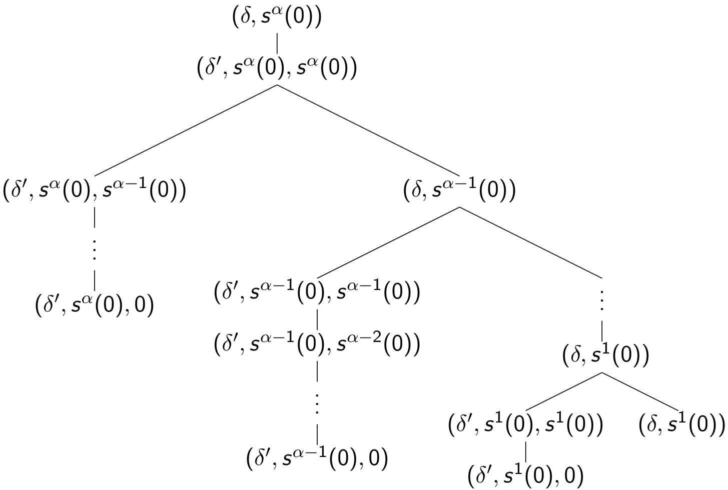
$$T\left(\mathcal{G}, P_2, \left\{ \begin{array}{c} n \leftarrow 2, m \leftarrow 0, \\ k \leftarrow 2, w \leftarrow 2 \end{array} \right\}\right) = \left[(\delta', 2, 0, 2, 2), T\left(\mathcal{G}, P_2, \left\{ \begin{array}{c} n \leftarrow 2, m \leftarrow 0, \\ k \leftarrow 1, w \leftarrow 3 \end{array} \right\}\right)\right]$$

$$T\left(\mathcal{G}, P_2, \left\{ \begin{array}{c} n \leftarrow 2, m \leftarrow 0, \\ k \leftarrow 1, w \leftarrow 3 \end{array} \right\}\right) = \left[(\delta', 2, 0, 1, 3), T\left(\mathcal{G}, P_2, \left\{ \begin{array}{c} n \leftarrow 2, m \leftarrow 0, \\ k \leftarrow 0, w \leftarrow 4 \end{array} \right\}\right)\right]$$

$$T\left(\mathcal{G}, P_2, \left\{ \begin{array}{c} n \leftarrow 2, m \leftarrow 0, \\ k \leftarrow 0, w \leftarrow 4 \end{array} \right\}\right) = [(\delta', 2, 0, 0, 4), \emptyset]$$

$$[(\delta, 2), [(\delta', 2, 1, 2, 0), [(\delta', 2, 1, 1, 1), [(\delta', 2, 1, 0, 2), [(\delta', 2, 0, 2, 2), [(\delta', 2, 0, 1, 3), [(\delta', 2, 0, 0, 4), \emptyset]]]]]]]$$

Trace for branching Call Graph



Trace Normalization

- Note that Traces seem to result in finite trees but we also mentioned that the order $\lhd$ is not well founded.
- Fortunately finite call graphs use a well founded suborder of $\lhd$

Theorem

Let $\mathcal{G}$ be a finite call graph, $P_{S^*} \in \mathcal{G}$ a flow, and $\sigma \in \mathcal{S}$ a parameter assignment. Then $T(\mathcal{G}, P_{S^*}, \sigma)$ always produces a finite trace.

- while this isn't too surprising from what we have discussed so far call graphs may contain mutually recursive calls.
- This feature is essential for the refutation of the cut structure of the k-repetition statement.

Mutual Calls

$$
C_2 = \left\{ \begin{array}{c} \left(\left\{ \begin{array}{c} (\delta_1, n, m, w, k, r, 0), \\ (\delta_1, n, m, p(w), s(k), r, 0), \\ (\delta_2, n, m, w, k, p(p(r)), 0), \\ (\delta_3, n, m, w, k, p(r), 0) \end{array} \right\}, S_5 \right) \\ \left(\left\{ \begin{array}{c} (\delta_1, n, m, w, k, r, 0), \\ (\delta_4, n, m, 0, k, p(r), 0), \\ (\delta_2, n, m, 0, k, p(p(r)), 0) \end{array} \right\}, S_6 \right) \end{array} \right\}
C_3 = \left\{ \begin{array}{c} \left(\left\{ \begin{array}{c} (\delta_2, n, m, w, k, r, q), \\ (\delta_2, n, m, w, , p(r), s(q)), \\ (\delta_1, n, m, p(w), s(k), s(r), 0), \\ (\delta_3, n, m, w, k, p(r), s(q)) \end{array} \right\}, S_7 \right) \\ \left(\left\{ \begin{array}{c} (\delta_2, n, m, w, k, r, q), \\ (\delta_1, n, m, p(w), s(k), s(m), 0), \\ (\delta_3, n, m, w, k, 0, s(q)), \\ (\delta_5, n, m, w, k, 0, s(q)) \end{array} \right\}, S_8 \right) \end{array} \right\}
$$

- $S_5 = \{\sigma | \sigma \in \mathcal{S} \ , \ \ w\sigma > 0\}$
- $S_6 = \{\sigma | \sigma \in \mathcal{S} \ , \ \ w\sigma = 0\}$
- $S_7 = \{\sigma | \sigma \in \mathcal{S} \ , \ \ r\sigma > 0\}$
- $S_8 = \{\sigma | \sigma \in \mathcal{S} \ , \ \ r\sigma = 0\}$

Mutual Calls

$$C_2 = \left\{ \begin{array}{c} \left(\left\{ \begin{array}{c} (\delta_1, n, m, w, k, r, 0), \\ (\delta_1, n, m, p(w), s(k), r, 0), \\ (\delta_2, n, m, w, k, p(p(r)), 0), \\ (\delta_3, n, m, w, k, p(r), 0) \end{array} \right\}, S_5 \right) \\ \left(\left\{ \begin{array}{c} (\delta_1, n, m, w, k, r, 0), \\ (\delta_4, n, m, 0, k, p(r), 0), \\ (\delta_2, n, m, 0, k, p(p(r)), 0) \end{array} \right\}, S_6 \right) \end{array} \right\} \quad C_3 = \left\{ \begin{array}{c} \left(\left\{ \begin{array}{c} (\delta_2, n, m, w, k, r, q), \\ (\delta_2, n, m, w, , p(r), s(q)), \\ (\delta_1, n, m, p(w), s(k), s(r), 0), \\ (\delta_3, n, m, w, k, p(r), s(q)) \end{array} \right\}, S_7 \right) \\ \left(\left\{ \begin{array}{c} (\delta_2, n, m, w, k, r, q), \\ (\delta_1, n, m, p(w), s(k), s(m), 0), \\ (\delta_3, n, m, w, k, 0, s(q)), \\ (\delta_5, n, m, w, k, 0, s(q)) \end{array} \right\}, S_8 \right) \end{array} \right\}$$

- $S_5 = \{\sigma | \sigma \in \mathcal{S} \ , \ \ w\sigma > 0\}$
- $S_6 = \{\sigma | \sigma \in \mathcal{S} \ , \ \ w\sigma = 0\}$
- $S_7 = \{\sigma | \sigma \in \mathcal{S} \ , \ \ r\sigma > 0\}$
- $S_8 = \{\sigma | \sigma \in \mathcal{S} \ , \ \ r\sigma = 0\}$

Mutual Calls

$$C_2 = \left\{ \begin{array}{c} \left(\left\{ \begin{array}{c} (\delta_1, n, m, w, k, r, 0), \\ (\delta_1, n, m, p(w), s(k), r, 0), \\ (\delta_2, n, m, w, k, p(p(r)), 0), \\ (\delta_3, n, m, w, k, p(r), 0) \end{array} \right\}, S_5 \right) \\ \left(\left\{ \begin{array}{c} (\delta_1, n, m, w, k, r, 0), \\ (\delta_4, n, m, 0, k, p(r), 0), \\ (\delta_2, n, m, 0, k, p(p(r)), 0) \end{array} \right\}, S_6 \right) \end{array} \right\} \quad C_3 = \left\{ \begin{array}{c} \left(\left\{ \begin{array}{c} (\delta_2, n, m, w, k, r, q), \\ (\delta_2, n, m, w, , p(r), s(q)), \\ (\delta_1, n, m, p(w), s(k), s(r), 0), \\ (\delta_3, n, m, w, k, p(r), s(q)) \end{array} \right\}, S_7 \right) \\ \left(\left\{ \begin{array}{c} (\delta_2, n, m, w, k, r, q), \\ (\delta_1, n, m, p(w), s(k), s(m), 0), \\ (\delta_3, n, m, w, k, 0, s(q)), \\ (\delta_5, n, m, w, k, 0, s(q)) \end{array} \right\}, S_8 \right) \end{array} \right\}$$

- $S_5 = \{\sigma | \sigma \in \mathcal{S} \ , \ w\sigma > 0\}$
- $S_6 = \{\sigma | \sigma \in \mathcal{S} \ , \ w\sigma = 0\}$
- $S_7 = \{\sigma | \sigma \in \mathcal{S} \ , \ r\sigma > 0\}$
- $S_8 = \{\sigma | \sigma \in \mathcal{S} \ , \ r\sigma = 0\}$

Mutual Calls

$$C_2 = \left\{ \begin{array}{c} \left(\left\{ \begin{array}{c} (\delta_1, n, m, w, k, r, 0), \\ (\delta_1, n, m, p(w), s(k), r, 0), \\ (\delta_2, n, m, w, k, p(p(r)), 0), \\ (\delta_3, n, m, w, k, p(r), 0) \end{array} \right\}, S_5 \right) \\ \left(\left\{ \begin{array}{c} (\delta_1, n, m, w, k, r, 0), \\ (\delta_4, n, m, 0, k, p(r), 0), \\ (\delta_2, n, m, 0, k, p(p(r)), 0) \end{array} \right\}, S_6 \right) \end{array} \right\} \quad C_3 = \left\{ \begin{array}{c} \left(\left\{ \begin{array}{c} (\delta_2, n, m, w, k, r, q), \\ (\delta_2, n, m, w, , p(r), s(q)), \\ (\delta_1, n, m, p(w), s(k), s(r), 0), \\ (\delta_3, n, m, w, k, p(r), s(q)) \end{array} \right\}, S_7 \right) \\ \left(\left\{ \begin{array}{c} (\delta_2, n, m, w, k, r, q), \\ (\delta_1, n, m, p(w), s(k), s(m), 0), \\ (\delta_3, n, m, w, k, 0, s(q)), \\ (\delta_5, n, m, w, k, 0, s(q)) \end{array} \right\}, S_8 \right) \end{array} \right\}$$

- $S_5 = \{\sigma | \sigma \in \mathcal{S} \ , \ \ w\sigma > 0\}$
- $S_6 = \{\sigma | \sigma \in \mathcal{S} \ , \ \ w\sigma = 0\}$
- $S_7 = \{\sigma | \sigma \in \mathcal{S} \ , \ \ r\sigma > 0\}$
- $S_8 = \{\sigma | \sigma \in \mathcal{S} \ , \ \ r\sigma = 0\}$

Mutual Calls

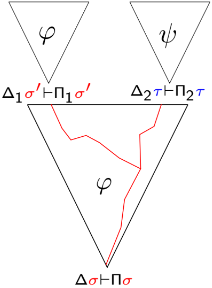
$$C_2 = \left\{ \begin{array}{c} \left(\left\{ \begin{array}{c} (\delta_1, n, m, w, k, r, 0), \\ (\delta_1, n, m, p(w), s(k), r, 0), \\ (\delta_2, n, m, w, k, p(p(r)), 0), \\ (\delta_3, n, m, w, k, p(r), 0) \end{array} \right\}, S_5 \right) \\ \left(\left\{ \begin{array}{c} (\delta_1, n, m, w, k, r, 0), \\ (\delta_4, n, m, 0, k, p(r), 0), \\ (\delta_2, n, m, 0, k, p(p(r)), 0) \end{array} \right\}, S_6 \right) \end{array} \right\} C_3 = \left\{ \begin{array}{c} \left(\left\{ \begin{array}{c} (\delta_2, n, m, w, k, r, q), \\ (\delta_2, n, m, w, , p(r), s(q)), \\ (\delta_1, n, m, p(w), s(k), s(r), 0), \\ (\delta_3, n, m, w, k, p(r), s(q)) \end{array} \right\}, S_7 \right) \\ \left(\left\{ \begin{array}{c} (\delta_2, n, m, w, k, r, q), \\ (\delta_1, n, m, p(w), s(k), s(m), 0), \\ (\delta_3, n, m, w, k, 0, s(q)), \\ (\delta_5, n, m, w, k, 0, s(q)) \end{array} \right\}, S_8 \right) \end{array} \right\}$$

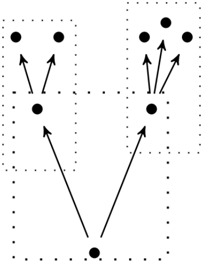
- $S_5 = \{\sigma | \sigma \in \mathcal{S} \ , \ w\sigma > 0\}$
- $S_6 = \{\sigma | \sigma \in \mathcal{S} \ , \ w\sigma = 0\}$
- $S_7 = \{\sigma | \sigma \in \mathcal{S} \ , \ r\sigma > 0\}$
- $S_8 = \{\sigma | \sigma \in \mathcal{S} \ , \ r\sigma = 0\}$

Decorating Traces with Proof Schemata

- Every derivation in a proof schema is associated with a proof symbol.
- Every derivation is associated with a set of free parameters.
- Every derivation has a set of non-trivial leaves(possibly empty)
- can use these properties to match derivations to flows and junctions contained in the flows.

Decorating Traces with Proof Schemata





Decoration of C_2 for partition S_5

$$(\delta_3, n, m, w, k, pred(r), 0)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash \hat{F}_4(\bar{\mathbf{X}}, k, 0)}{\vdash f(\hat{S}(X_3(k,0)) \not< s(k) \vee f(X_3(k,0)) < k \vee f(X_3(k,0)) = k}\; B\hat{F}_4 r}{\vdash f(X_3(k,0)) \not< s(k), f(X_3(k,0)) < k \vee f(X_3(k,0)) = k}\; \vee : r}{\vdash f(X_3(k,0)) \not< s(k), f(X_3(k,0)) < k, f(X_3(k,0)) = k}\; \vee : r}{f(X_3(k,0)) < s(k) \vdash f(X_3(k,0)) < k, f(X_3(k,0)) = k}\; \neg : r$$

(2)

$$(\delta_1, n, m, p(w), s(k), r, 0)$$

$$\dfrac{\vdash f(\alpha) < s(k) \qquad\qquad (2)}{\vdash f(\alpha) < k, f(\alpha) = k}\; \mathrm{Res}(\{\ X_3(k,0) \leftarrow \alpha\ \})$$

(1)

$$(\delta_2, n, m, w, k, p(p(r)), 0)$$

$$\dfrac{\dfrac{(1)}{\vdash f(\alpha) < k, f(\alpha) = k} \qquad \dfrac{\dfrac{\dfrac{\vdash f(\alpha) < k, \hat{F}_5(\alpha, \bar{\mathbf{X}}, k, 0)}{\vdash f(\alpha) < k, f(\hat{S}(0,\alpha)) \neq k}\; B\hat{F}_5 r}{\vdash f(\alpha) < k, f(\alpha) \neq k}\; B\hat{S} r}{f(\alpha) = k \vdash f(\alpha) < k}\; \neg : r}{\vdash f(\alpha) < k}\; \mathrm{Res}(\emptyset)$$

$$(\delta_1, n, m, w, k, r, 0)$$

Conclusion & Future work

- The formalism presented above can be used for finite representation of the resolution refutation need for proof analysis using CERES.
- Additionally, we can imagine a reverse resolution calculus
  - Given an unsatisfiable recursive clause set does there exists a refutation of the clause set using the given Call Graph.

  This is currently being investigated.

Thank you for your time.