Schematic Cut Elimination and the Ordered Pigeonhole Principle

David M. Cerna
and
Alexander Leitsch

June 28, 2016

The CERES Method and Inductive Arguments

- The CERES method [Baaz & Leitsch , 2000] approaches cut elimination by refuting a clausal representation of the cut structure.

The CERES Method and Inductive Arguments

- The CERES method [Baaz & Leitsch , 2000] approaches cut elimination by refuting a clausal representation of the cut structure.
- Unlike *reductive* cut elimination á la Gentzen, which simplifies cuts locally, CERES considers the global cut structure of the proof using a clausal representation.

The CERES Method and Inductive Arguments

- The CERES method [Baaz & Leitsch , 2000] approaches cut elimination by refuting a clausal representation of the cut structure.
- Unlike *reductive* cut elimination á la Gentzen, which simplifies cuts locally, CERES considers the global cut structure of the proof using a clausal representation.
- The proof analysis of Fürstenbergs proof of the infinitude of primes [Baaz et al. , 2008] brought to our attention how a method based on the global cut structure can be beneficial for the proof analysis of inductive arguments.

The CERES Method and Inductive Arguments

- The CERES method [Baaz & Leitsch , 2000] approaches cut elimination by refuting a clausal representation of the cut structure.
- Unlike *reductive* cut elimination á la Gentzen, which simplifies cuts locally, CERES considers the global cut structure of the proof using a clausal representation.
- The proof analysis of Fürstenbergs proof of the infinitude of primes [Baaz et al. , 2008] brought to our attention how a method based on the global cut structure can be beneficial for the proof analysis of inductive arguments.
- Central to Fürstenbergs proof is an induction argument over linear progressions.

The CERES Method and Inductive Arguments

- The CERES method [Baaz & Leitsch , 2000] approaches cut elimination by refuting a clausal representation of the cut structure.
- Unlike *reductive* cut elimination á la Gentzen, which simplifies cuts locally, CERES considers the global cut structure of the proof using a clausal representation.
- The proof analysis of Fürstenbergs proof of the infinitude of primes [Baaz et al. , 2008] brought to our attention how a method based on the global cut structure can be beneficial for the proof analysis of inductive arguments.
- Central to Fürstenbergs proof is an induction argument over linear progressions.
- Though, a large portion of the proof analysis was worked out on pen and paper it lead to investigations towards the development of an inductive CERES method.

The CERES Method and Inductive Arguments

- The CERES method [Baaz & Leitsch , 2000] approaches cut elimination by refuting a clausal representation of the cut structure.
- Unlike *reductive* cut elimination á la Gentzen, which simplifies cuts locally, CERES considers the global cut structure of the proof using a clausal representation.
- The proof analysis of Fürstenbergs proof of the infinitude of primes [Baaz et al. , 2008] brought to our attention how a method based on the global cut structure can be beneficial for the proof analysis of inductive arguments.
- Central to Fürstenbergs proof is an induction argument over linear progressions.
- Though, a large portion of the proof analysis was worked out on pen and paper it lead to investigations towards the development of an inductive CERES method.
- These investigations lead to the development of the schematic CERES method [Dunchev et al. 2013], the method at the heart of the proof analysis we present today.

Reductive Cut Elimination and Induction

- Gentzen showed that Peano arithmetic is consistent by replacing instances of the induction rule by a sequence of cuts.

Reductive Cut Elimination and Induction

- Gentzen showed that Peano arithmetic is consistent by replacing instances of the induction rule by a sequence of cuts.
- This transformation was done meta-theoretically rather than algorithmically.

Reductive Cut Elimination and Induction

- Gentzen showed that Peano arithmetic is consistent by replacing instances of the induction rule by a sequence of cuts.
- This transformation was done meta-theoretically rather than algorithmically.
- In cases when the induction introduces a new eigenvariable the transformation is not semantics preserving.

Reductive Cut Elimination and Induction

- Gentzen showed that Peano arithmetic is consistent by replacing instances of the induction rule by a sequence of cuts.
- This transformation was done meta-theoretically rather than algorithmically.
- In cases when the induction introduces a new eigenvariable the transformation is not semantics preserving.
- A consequence of this issue is the loss of the <u>sub-formula property</u>.

Reductive Cut Elimination and Induction

- Gentzen showed that Peano arithmetic is consistent by replacing instances of the induction rule by a sequence of cuts.
- This transformation was done meta-theoretically rather than algorithmically.
- In cases when the induction introduces a new eigenvariable the transformation is not semantics preserving.
- A consequence of this issue is the loss of the sub-formula property.
- A Herbrand sequent cannot be extracted and the relationship between propositional logic and inductive first order is left obfuscated

Cuts don't pass over Induction

- Consider the following example:

Cuts don't pass over Induction

- Consider the following example:

- Term algebra for equational theory:

$$\varepsilon \equiv \left\{ f_I(0, x) = x, f_I(s(n), x) = f\left(f_I(n, x)\right) \right\}$$

Cuts don't pass over Induction

- Consider the following example:

- Term algebra for equational theory:

$$\varepsilon \equiv \left\{ f_I(0,x) = x, f_I(s(n),x) = f\left(f_I(n,x)\right) \right\}$$

- The consequent of the end sequent:

$$E_c \equiv \forall n \left(\left(P\left(f_I(n,c)\right) \to P\left(g(n,c)\right) \right) \to \left(P(c) \to P\left(g(n,c)\right) \right) \right)$$

Cuts don't pass over Induction

- Consider the following example:

- Term algebra for equational theory:

$$\varepsilon \equiv \left\{ f_I(0,x) = x, f_I(s(n),x) = f\left(f_I(n,x)\right) \right\}$$

- The consequent of the end sequent:

$$E_c \equiv \forall n \left(\left(P\left(f_I(n,c)\right) \rightarrow P\left(g(n,c)\right) \right) \rightarrow \left(P(c) \rightarrow P\left(g(n,c)\right) \right) \right)$$

- The cut formula:

$$C \equiv \forall n \forall x \left(P(x) \rightarrow P\left(f_I(n,x)\right) \right)$$

Cuts don't pass over Induction

- Consider the following example:

- Term algebra for equational theory:

$$\varepsilon \equiv \left\{ f_I(0,x) = x, f_I(s(n),x) = f\left(f_I(n,x)\right) \right\}$$

- The consequent of the end sequent:

$$E_c \equiv \forall n \Big(\Big(P\big(f_I(n,c)\big) \rightarrow P\big(g(n,c)\big) \Big) \rightarrow \Big(P(c) \rightarrow P\big(g(n,c)\big) \Big) \Big)$$

- The cut formula:

$$C \equiv \forall n \forall x \Big(P(x) \rightarrow P\big(f_I(n,x)\big) \Big)$$

- Inductive Lemma:

$$\forall x \Big(P(x) \rightarrow P\big(f(x)\big) \Big) \vdash \forall n \forall x \Big(P(x) \rightarrow P\big(f_I(n,x)\big) \Big)$$

Cuts don't pass over Induction

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \forall x\Big(P(x) \to P\big(f_I(0,x)\big)\Big)\end{array} \qquad \begin{array}{c}\vdots\\ \Gamma, \forall x\Big(P(x) \to P\big(f_I(\alpha,x)\big)\Big) \vdash \forall x\Big(P(x) \to P\big(f_I(s(\alpha),x)\big)\Big)\end{array}}{\forall x\Big(P(x) \to P\big(f(x)\big)\Big) \vdash \forall x\Big(P(x) \to P\big(f_I(\gamma,x)\big)\Big)}\, IND}{\color{red}{\forall x\Big(P(x) \to P\big(f(x)\big)\Big) \vdash C}}\, \forall_r$$

$$\dfrac{\color{red}{\forall x\Big(P(x) \to P\big(f(x)\big)\Big) \vdash C} \qquad \begin{array}{c}\vdots\\ C \vdash E_c\end{array}}{\forall x\Big(P(x) \to P\big(f(x)\big)\Big) \vdash E_c}\, cut$$

$C \equiv \forall n \forall x\Big(P(x) \to P\big(f_I(n,x)\big)\Big)$

- By reductive cut elimination we can eliminate the $\forall_r$.
- However, we get stuck at the induction due to the change in eigenvariables.

Avoiding Induction

- Instead of n being replaced by an eigenvariable we can consider a term t over the signature $\{0, 1, +, *\}$ s.t. $\vdash n = t$ in Peano arithmetic.
- Removing the induction and replacing it with a sequence of cuts allows reductive cut elimination to occur for any fixed term t.

$$
\dfrac{\dfrac{\begin{array}{c}\forall x\Big(P(f(x)) \rightarrow P\big(f(f_I(t,x))\big)\Big),\\ \forall x\Big(P(x) \rightarrow P\big(f(x)\big)\Big) \vdash \\ \forall x\Big(P(x) \rightarrow P\big(f_I(s(t),x)\big)\Big)\end{array} \qquad \begin{array}{c}\varphi(n) :: \forall x\Big(P(x) \rightarrow P\big(f(x)\big)\Big) \vdash \\ \forall x\Big(P(f(x)) \rightarrow P\big(f(f_I(t,x))\big)\Big)\end{array}}{\forall x\Big(P(x) \rightarrow P\big(f(x)\big)\Big), \forall x\Big(P(x) \rightarrow P\big(f(x)\big)\Big) \vdash \forall x\Big(P(x) \rightarrow P\big(f_I(s(t),x)\big)\Big)}\,cut}{\varphi(n+1) :: \forall x\Big(P(x) \rightarrow P\big(f(x)\big)\Big) \vdash \forall x\Big(P(x) \rightarrow P\big(f_I(s(t),x)\big)\Big)}\,c_I
$$

$$
\dfrac{\forall x\Big(P(x) \rightarrow P\big(f(x)\big)\Big) \vdash cut^{s(t)} \qquad cut^{s(t)} \vdash E_c^{s(t)}}{\forall x\Big(P(x) \rightarrow P\big(f(x)\big)\Big) \vdash E_c^{s(t)}}\,cut
$$

Avoiding Induction

- Instead of n being replaced by an eigenvariable we can consider a term t over the signature $\{0, 1, +, *\}$ s.t. $\vdash n = t$ in Peano arithmetic.
- Removing the induction and replacing it with a sequence of cuts allows reductive cut elimination to occur for any fixed term t.
- Essentially, we have push the eigenvariable to the metalevel for the given signature.

$$\dfrac{\dfrac{\begin{array}{c}\forall x\Big(P(f(x)) \to P(f(f_I(t,x)))\Big),\\ \forall x\Big(P(x) \to P(f(x))\Big) \vdash\\ \forall x\Big(P(x) \to P(f_I(s(t),x))\Big)\end{array} \qquad \begin{array}{c}\varphi(n) :: \forall x\Big(P(x) \to P(f(x))\Big) \vdash\\ \forall x\Big(P(f(x)) \to P(f(f_I(t,x)))\Big)\end{array}}{\forall x\Big(P(x) \to P(f(x))\Big), \forall x\Big(P(x) \to P(f(x))\Big) \vdash \forall x\Big(P(x) \to P(f_I(s(t),x))\Big)}\, cut}{\varphi(n+1) :: \forall x\Big(P(x) \to P(f(x))\Big) \vdash \forall x\Big(P(x) \to P(f_I(s(t),x))\Big)}\, c_l$$

$$\dfrac{\forall x\Big(P(x) \to P(f(x))\Big) \vdash cut^{s(t)} \qquad cut^{s(t)} \vdash E_c^{s(t)}}{\forall x\Big(P(x) \to P(f(x))\Big) \vdash E_c^{s(t)}}\, cut$$

Avoiding Induction

- Instead of n being replaced by an eigenvariable we can consider a term t over the signature $\{0, 1, +, *\}$ s.t. $\vdash n = t$ in Peano arithmetic.
- Removing the induction and replacing it with a sequence of cuts allows reductive cut elimination to occur for any fixed term t.
- Essentially, we have push the eigenvariable to the metalevel for the given signature.

$$
\dfrac{\dfrac{\begin{array}{c}\forall x\Big(P(f(x)) \rightarrow P(f(f_I(t,x)))\Big), \\ \forall x\Big(P(x) \rightarrow P(f(x))\Big) \vdash \\ \forall x\Big(P(x) \rightarrow P(f_I(s(t),x))\Big)\end{array} \qquad \begin{array}{c}\varphi(n) :: \forall x\Big(P(x) \rightarrow P(f(x))\Big) \vdash \\ \forall x\Big(P(f(x)) \rightarrow P(f(f_I(t,x)))\Big)\end{array}}{\forall x\Big(P(x) \rightarrow P(f(x))\Big), \forall x\Big(P(x) \rightarrow P(f(x))\Big) \vdash \forall x\Big(P(x) \rightarrow P(f_I(s(t),x))\Big)}\ cut}{\varphi(n+1) :: \forall x\Big(P(x) \rightarrow P(f(x))\Big) \vdash \forall x\Big(P(x) \rightarrow P(f_I(s(t),x))\Big)}\ c_l
$$

$$
\dfrac{\forall x\Big(P(x) \rightarrow P(f(x))\Big) \vdash cut^{s(t)} \qquad cut^{s(t)} \vdash E_c^{s(t)}}{\forall x\Big(P(x) \rightarrow P(f(x))\Big) \vdash E_c^{s(t)}}\ cut
$$

- Though, this method constructs a proof admitting reductive cut-elimination it does not produce a finite representation of the reductive cut elimination for all possible t.

Representing the Proof Sequence

- Let us consider a simple sequence of terms indexing a proof:

$$S \equiv 0, 0+1, (0+1)+1, ((0+1)+1)+1, \cdots$$

- If the proof for the $(n+1)^{\text{th}}$ term of the sequence S assumes the that there is a proof associated with the n^{th} term we have a structure similar to induction argument.

Representing the Proof Sequence

- Let us consider a simple sequence of terms indexing a proof:

$$S \equiv 0, 0+1, (0+1)+1, ((0+1)+1)+1, \cdots$$

- If the proof for the $(n+1)^{\mathrm{th}}$ term of the sequence S assumes the that there is a proof associated with the n^{th} term we have a structure similar to induction argument.

$$\begin{array}{c} \varphi(0) \\ \vdots \\ \varphi(t) \\ \vdots \\ \varphi(t+1) \end{array}$$

Representing the Proof Sequence

— Let us consider a simple sequence of terms indexing a proof:

$$S \equiv 0, 0+1, (0+1)+1, ((0+1)+1)+1, \cdots$$

— If the proof for the $(n+1)^{\mathrm{th}}$ term of the sequence S assumes the that there is a proof associated with the n^{th} term we have a structure similar to induction argument.

$$\begin{array}{c} \varphi(0) \\ \vdots \\ \varphi(t) \\ \vdots \\ \varphi(t+1) \end{array}$$

— This can be written more formally in the LKS-calculus using proof links .

LKS-calculus Proof Structure

$$\frac{\Sigma \vdash P(0), \Delta \qquad \Pi, P(\alpha) \vdash P(s(\alpha)), \Gamma}{\Pi, \Sigma \vdash P(\beta), \Delta, \Gamma}$$

LKS-calculus Proof Structure

$$\frac{\Sigma \vdash P(0), \Delta \qquad \Pi, P(\alpha) \vdash P(s(\alpha)), \Gamma}{\Pi, \Sigma \vdash \cancel{P(\beta)}, \Delta, \Gamma}$$

$$P(s(\alpha))$$

LKS-calculus Proof Structure

$$
\dfrac{\dfrac{\dfrac{\Psi}{\Pi' \vdash \Gamma'} \qquad \dfrac{\varphi(0)}{\Sigma \vdash P(0), \Delta}}{\vdots \qquad\qquad \vdots}}{\Pi'', \Sigma \vdash P(1), \Delta, \Gamma''}
$$

$$
\dfrac{\Sigma \vdash P(0), \Delta \qquad \Pi, \cancel{P(\alpha)} \vdash P(s(\alpha)), \Gamma}{\Pi, \Sigma \vdash \cancel{P(\beta)}, \Delta, \Gamma} \qquad \Rightarrow
$$

$$P(s(\alpha))$$

LKS-calculus Proof Structure

$$\cfrac{\Sigma \vdash P(0), \Delta \qquad \Pi, P(\alpha) \vdash P(s(\alpha)), \Gamma}{\Pi, \Sigma \vdash \underset{P(s(\alpha))}{\cancel{P(\beta)}}, \Delta, \Gamma} \quad \Rightarrow \quad \begin{array}{c} \cfrac{\cfrac{\Psi}{\Pi' \vdash \Gamma'} \qquad \cfrac{\varphi(0)}{\Sigma \vdash P(0), \Delta}}{\begin{array}{ccc} \vdots & & \vdots \\ \hline \multicolumn{3}{c}{\Pi'', \Sigma \vdash P(1), \Delta, \Gamma''} \end{array}} \\ \Downarrow \\ \cfrac{\cfrac{\Psi}{\Pi' \vdash \Gamma'} \qquad \cfrac{\varphi(1)}{\Pi'', \Sigma \vdash P(1), \Delta, \Gamma''}}{\begin{array}{ccc} \vdots & & \vdots \\ \hline \multicolumn{3}{c}{\Pi''', \Sigma \vdash P(2), \Delta, \Gamma'''} \end{array}} \end{array}$$

LKS-calculus Proof Structure

$$
\dfrac{\Sigma \vdash P(0), \Delta \qquad \Pi, P(\alpha) \vdash P(s(\alpha)), \Gamma}{\Pi, \Sigma \vdash \cancel{P(\beta)}, \Delta, \Gamma}
\qquad
\begin{array}{c} \\ P(s(\alpha)) \end{array}
$$

$$\Rightarrow$$

$$
\dfrac{\begin{array}{cc} \dfrac{\Psi}{\Pi' \vdash \Gamma'} & \dfrac{\varphi(0)}{\Sigma \vdash P(0), \Delta} \\ \vdots & \vdots \end{array}}{\Pi'', \Sigma \vdash P(1), \Delta, \Gamma''}
$$

$$\Downarrow$$

$$
\dfrac{\begin{array}{cc} \dfrac{\Psi}{\Pi' \vdash \Gamma'} & \dfrac{\varphi(1)}{\Pi'', \Sigma \vdash P(1), \Delta, \Gamma''} \\ \vdots & \vdots \end{array}}{\Pi''', \Sigma \vdash P(2), \Delta, \Gamma'''}
$$

$$\Downarrow \quad \alpha - 2 \ \textit{times}$$

$$
\dfrac{\begin{array}{cc} \dfrac{\Psi}{\Pi' \vdash \Gamma'} & \dfrac{\varphi(\alpha)}{\Pi^{(\alpha+1)}, \Sigma \vdash P(\alpha), \Delta, \Gamma^{(\alpha+1)}} \\ \vdots & \vdots \end{array}}{\Pi, \Sigma \vdash P(\alpha + 1), \Delta, \Gamma}
$$

LKS-calculus Proof Structure

$$\dfrac{\Sigma \vdash P(0), \Delta \qquad \Pi, P(\alpha) \vdash P(s(\alpha)), \Gamma}{\Pi, \Sigma \vdash \cancel{P(\beta)}, \Delta, \Gamma}$$
$$P(s(\alpha))$$

$$\Rightarrow$$

$$\dfrac{\begin{matrix}\dfrac{\Psi}{\Pi' \vdash \Gamma'} & \dfrac{\varphi(0)}{\Sigma \vdash P(0), \Delta} \\ \vdots & \vdots\end{matrix}}{\Pi'', \Sigma \vdash P(1), \Delta, \Gamma''}$$

$$\Downarrow$$

$$\dfrac{\begin{matrix}\dfrac{\Psi}{\Pi' \vdash \Gamma'} & \dfrac{\varphi(1)}{\Pi'', \Sigma \vdash P(1), \Delta, \Gamma''} \\ \vdots & \vdots\end{matrix}}{\Pi''', \Sigma \vdash P(2), \Delta, \Gamma'''}$$

$$\Downarrow \quad \alpha - 2 \ \ \textit{times}$$

$$\dfrac{\begin{matrix}\dfrac{\Psi}{\Pi' \vdash \Gamma'} & \dfrac{\varphi(\alpha)}{\Pi^{(\alpha+1)}, \Sigma \vdash P(\alpha), \Delta, \Gamma^{(\alpha+1)}} \\ \vdots & \vdots\end{matrix}}{\Pi, \Sigma \vdash P(\alpha + 1), \Delta, \Gamma}$$

- As we increase the value of α, the length of the proof increases.
- We get a cyclic proof with internal structure we refer to as configurations .
- Configurations refer to the structure of the end sequents at proof links.

Primitive Recursive Proof Representation

- More formally, what we have outlined on the previous slide is a proof schema pair φ, i.e.

  $\varphi \equiv (\pi(0), \nu(k+1))$.

- Essentially a proof written primitive recursively using only one primitive recursion operator.
- Proof schemata are a sequence of ordered proof schema pairs.
- Let us consider a proof schema $\langle \varphi, \psi \rangle$.

Primitive Recursive Proof Representation

- More formally, what we have outlined on the previous slide is a proof schema pair φ, i.e.

  $\varphi \equiv (\pi(0), \nu(k+1))$.

- Essentially a proof written primitive recursively using only one primitive recursion operator.
- Proof schemata are a sequence of ordered proof schema pairs.
- Let us consider a proof schema $\langle \varphi, \psi \rangle$.

$$\varphi(n+1)$$

Primitive Recursive Proof Representation

- More formally, what we have outlined on the previous slide is a proof schema pair φ, i.e. $\varphi \equiv (\pi(0), \nu(k+1))$.
- Essentially a proof written primitive recursively using only one primitive recursion operator.
- Proof schemata are a sequence of ordered proof schema pairs.
- Let us consider a proof schema $\langle \varphi, \psi \rangle$.

$$\begin{tikzcd} & \psi(s) \\ \varphi(n+1) \arrow[ur, bend left] \arrow[dr, bend right] & \\ & \varphi(n) \end{tikzcd}$$

Primitive Recursive Proof Representation

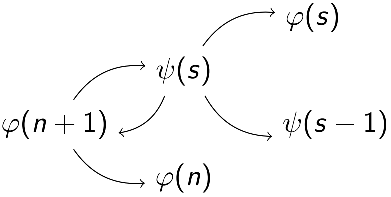
- More formally, what we have outlined on the previous slide is a proof schema pair φ, i.e.

  $\varphi \equiv (\pi(0), \nu(k+1))$.

- Essentially a proof written primitive recursively using only one primitive recursion operator.

- Proof schemata are a sequence of ordered proof schema pairs.

- Let us consider a proof schema $\langle \varphi, \psi \rangle$.



Primitive Recursive Proof Representation

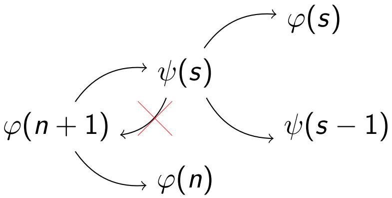
- More formally, what we have outlined on the previous slide is a proof schema pair φ, i.e.

  $\varphi \equiv (\pi(0), \nu(k+1))$.

- Essentially a proof written primitive recursively using only one primitive recursion operator.

- Proof schemata are a sequence of ordered proof schema pairs.

- Let us consider a proof schema $\langle \varphi, \psi \rangle$.



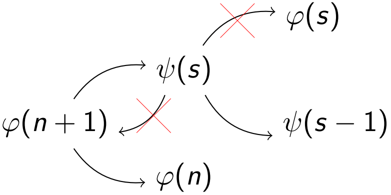
- No cyclic calls allowed.

Primitive Recursive Proof Representation

- More formally, what we have outlined on the previous slide is a proof schema pair φ, i.e.

  $\varphi \equiv (\pi(0), \nu(k+1))$.

- Essentially a proof written primitive recursively using only one primitive recursion operator.
- Proof schemata are a sequence of ordered proof schema pairs.
- Let us consider a proof schema $\langle \varphi, \psi \rangle$.



- No cyclic calls allowed. Order must be preserved $\varphi < \psi$.

Configuration of Recursive Proof Structure

- Now with a definition of proof schemata, how do we extend CERES to such objects?

Configuration of Recursive Proof Structure

- Now with a definition of proof schemata, how do we extend CERES to such objects?
- Extraction of the characteristic clause set is performed on proof schema pairs and then the parts are put together guided by the proof links. Referred to as clause set normalization, i.e. removal of intermediary clause set symbols.
- The important difference is dealing with the configurations which are relevant to the cut formulae. Consider the proof schema $\langle \varphi, \psi, \chi \rangle$:

Configuration of Recursive Proof Structure

- Now with a definition of proof schemata, how do we extend CERES to such objects?
- Extraction of the characteristic clause set is performed on proof schema pairs and then the parts are put together guided by the proof links. Referred to as clause set normalization, i.e. removal of intermediary clause set symbols.
- The important difference is dealing with the configurations which are relevant to the cut formulae. Consider the proof schema $\langle \varphi, \psi, \chi \rangle$:

$$\varphi(n+1)$$

Configuration of Recursive Proof Structure

- Now with a definition of proof schemata, how do we extend CERES to such objects?
- Extraction of the characteristic clause set is performed on proof schema pairs and then the parts are put together guided by the proof links. Referred to as clause set normalization, i.e. removal of intermediary clause set symbols.
- The important difference is dealing with the configurations which are relevant to the cut formulae. Consider the proof schema $\langle \varphi, \psi, \chi \rangle$:

$$\begin{tikzcd}
 & \chi(s) \\
 & \psi(t) \\
\varphi(n+1) \arrow[uur, bend left] \arrow[ur, bend left] \arrow[dr, bend right] & \\
 & \varphi(n)
\end{tikzcd}$$

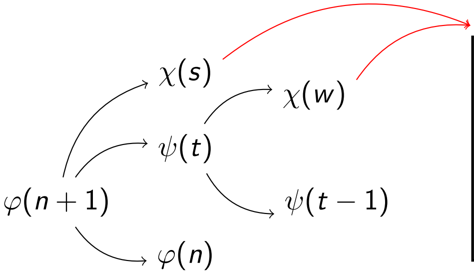
Configuration of Recursive Proof Structure

- Now with a definition of proof schemata, how do we extend CERES to such objects?
- Extraction of the characteristic clause set is performed on proof schema pairs and then the parts are put together guided by the proof links. Referred to as clause set normalization, i.e. removal of intermediary clause set symbols.
- The important difference is dealing with the configurations which are relevant to the cut formulae. Consider the proof schema $\langle \varphi, \psi, \chi \rangle$:

$$
\begin{tikzcd}
 & \chi(s) & \chi(w) \\
 & \psi(t) \arrow[ur, bend left] \arrow[dr, bend right] & \\
\varphi(n+1) \arrow[uur, bend left] \arrow[ur, bend left] \arrow[dr, bend right] & & \psi(t-1) \\
 & \varphi(n) &
\end{tikzcd}
$$

Configuration of Recursive Proof Structure

- Now with a definition of proof schemata, how do we extend CERES to such objects?
- Extraction of the characteristic clause set is performed on proof schema pairs and then the parts are put together guided by the proof links. Referred to as clause set normalization, i.e. removal of intermediary clause set symbols.
- The important difference is dealing with the configurations which are relevant to the cut formulae. Consider the proof schema $\langle\varphi, \psi, \chi\rangle$:



The two instances of χ were called by different proofs and can have different configurations concerning the ancestors of the cut formulae. The configurations must be denoted during the construction of the clause set.

Schematic CERES: Clause set Extraction & Refutation

- As with the CERES method, a clause set must be extracted, though, as expected, it is recursively defined.

Schematic CERES: Clause set Extraction & Refutation

- As with the CERES method, a clause set must be extracted, though, as expected, it is recursively defined.
- Essentially, proof schema have an infinite sequence of finite clause sets whose infinite sequence of refutations need to be described finitely.

Schematic CERES: Clause set Extraction & Refutation

- As with the CERES method, a clause set must be extracted, though, as expected, it is recursively defined.
- Essentially, proof schema have an infinite sequence of finite clause sets whose infinite sequence of refutations need to be described finitely.

  Example:

$$
\begin{array}{l}
\vdash P(c) \\
P(x_0) \vdash P(x_0) \\
P(x_0) \vdash P(f(x_0)) \\
P(f(x_1)) \vdash P(f^2(x_1)) \\
\vdots \\
P(f^n(x_n)) \vdash P(f^{n+1}(x_n)) \\
P(f^{n+1}(x_{n+1})) \vdash
\end{array}
$$

Schematic CERES: Clause set Extraction & Refutation

- As with the CERES method, a clause set must be extracted, though, as expected, it is recursively defined.
- Essentially, proof schema have an infinite sequence of finite clause sets whose infinite sequence of refutations need to be described finitely.

  Example:

$\vdash P(c)$

$P(x_0) \vdash P(x_0)$

$P(x_0) \vdash P(f(x_0))$

$P(f(x_1)) \vdash P(f^2(x_1))$

$\vdots$

$P(f^n(x_n)) \vdash P(f^{n+1}(x_n))$

$P(f^{n+1}(x_{n+1})) \vdash$

$$\frac{\vdash P(c) \qquad P(x_0) \vdash P(f(x_0))}{\vdash P(f(c))}$$

Schematic CERES: Clause set Extraction & Refutation

- As with the CERES method, a clause set must be extracted, though, as expected, it is recursively defined.
- Essentially, proof schema have an infinite sequence of finite clause sets whose infinite sequence of refutations need to be described finitely.

Example:

$$\vdash P(c)$$
$$P(x_0) \vdash P(x_0)$$
$$P(x_0) \vdash P(f(x_0))$$
$$P(f(x_1)) \vdash P(f^2(x_1))$$
$$\vdots$$
$$P(f^n(x_n)) \vdash P(f^{n+1}(x_n))$$
$$P(f^{n+1}(x_{n+1})) \vdash$$

$$\frac{\vdash P(c) \qquad P(x_0) \vdash P(f(x_0))}{\vdash P(f(c))}$$

$$\vdots$$

$$\frac{P(f^{n+1}(x_{n+1})) \vdash \qquad \dfrac{\vdash P(f^n(c)) \qquad P(f^n(x_n)) \vdash P(f^{n+1}(x_n))}{\vdash P(f^{n+1}(c))}}{\vdash}$$

Schematic Resolution Calculus

- The resolution refutation on the previous slide can be written in the schematic resolution calculus as following:

$$\rho(n+1) \Rightarrow r(P(f^{n+2}(x_{n+2}))\vdash;r(\chi(n+1);P(f^{n+1}(x_{n+1}))\vdash P(f^{n+2}(x_{n+1}));P(f^{n}(c)));P(f^{n+1}(c)))$$

$$\rho(0) \Rightarrow \chi(0)$$

$$\chi(n+1) \Rightarrow r(\chi(n);P(f^{n}(x_n))\vdash P(f^{n+1}(x_n));P(f^{n}(c)))$$

$$\chi(0) \Rightarrow r(P(c);P(x_0)\vdash P(f(x_0));P(c))$$

- Unification is not described in the resolution terms nor in the resolution schema, but rather as a substitution schema. In our case it would be $[x \setminus \lambda k.c]$, where $i \in [0, n]$.

End Result of Schematic Cut-elimination

- Unlike the CERES method for FOL, the schematic CERES method does not produce an ACNF.
  - The projections are not constructed.
- The “ACNF” is just a pairing of a resolution refutation schema with a substitution schema.
- Rather then returning a cut-free **LKS**-proof, the final result is a Herbrand system.
- To the best of our knowledge, other methods dealing with cut elimination in the presence of Induction cannot produce a Herbrand system being that they do not preserve the sub-formula property.

Weakening the Pigeonhole Principle

- Proof analysis of the full pigeonhole principle (NiA-schema) was attempted but, refuting the clause set was problematic.
- The NiA-schema considers every permutation of pigeon assignment. This translates to the clause set.

Let $f : \mathbb{N} \rightarrow \mathbb{N}_n$*, where* $n \in \mathbb{N}$*, be total, then there exists* $i, j \in \mathbb{N}$ *such that* $i < j$ *and* $f(i) = f(j)$*.*

Weakening the Pigeonhole Principle

- Proof analysis of the full pigeonhole principle (NiA-schema) was attempted but, refuting the clause set was problematic.
- The NiA-schema considers every permutation of pigeon assignment. This translates to the clause set.

Let $f : \mathbb{N} \rightarrow \mathbb{N}_n$, *where* $n \in \mathbb{N}$, *be total, then there exists* $i, j \in \mathbb{N}$ *such that* $i < j$ *and* $f(i) = f(j)$.

- Our formal proof of this statement uses the following sequence of lemmata.

$$\exists p \exists q \left(p, q \in \mathbb{N} \wedge p < q \wedge f(p) = f(q)\right) \vee$$
$$\forall x \exists y \left(x, y \in \mathbb{N} \wedge x \leq y \wedge f(y) \in \mathbb{N}_{n-1}\right)$$

- Problematically, the clause set extract from our formal proof using this lemma was too complex to analyse using our current techniques.

Adding order to PHP(ECA-schema)

- What made analysis of the NiA-schema hard was the complexity of the cut formula.
  - Essentially, we have a sequence of Π_2 cuts.
- Our solution to this problem is to modify the cut formulae to Σ_2 cuts and see how much of the NiA-schema can still be proven in the LKS-calculus.

Adding order to PHP(ECA-schema)

- What made analysis of the NiA-schema hard was the complexity of the cut formula.
  - Essentially, we have a sequence of Π_2 cuts.
- Our solution to this problem is to modify the cut formulae to Σ_2 cuts and see how much of the NiA-schema can still be proven in the LKS-calculus.
- We ended up with the following statement:

Given a total monotonically decreasing function $f : \mathbb{N} \rightarrow \{0, \cdots, n\}$, for $n \in \mathbb{N}$, there exists an $x \in \mathbb{N}$ such that for all $y \in \mathbb{N}$, where $x \leq y$, it is the case that $f(x) = f(y)$.

Adding order to PHP(ECA-schema)

- What made analysis of the NiA-schema hard was the complexity of the cut formula.
  - Essentially, we have a sequence of Π_2 cuts.
- Our solution to this problem is to modify the cut formulae to Σ_2 cuts and see how much of the NiA-schema can still be proven in the LKS-calculus.
- We ended up with the following statement:

Given a total monotonically decreasing function $f : \mathbb{N} \rightarrow \{0, \cdots, n\}$, for $n \in \mathbb{N}$, there exists an $x \in \mathbb{N}$ such that for all $y \in \mathbb{N}$, where $x \leq y$, it is the case that $f(x) = f(y)$.

- Notice that the end sequent contains a Σ_2 formula and needs to be skolemized.

Adding order to PHP(ECA-schema)

- What made analysis of the NiA-schema hard was the complexity of the cut formula.
  - Essentially, we have a sequence of Π_2 cuts.
- Our solution to this problem is to modify the cut formulae to Σ_2 cuts and see how much of the NiA-schema can still be proven in the LKS-calculus.
- We ended up with the following statement:

Given a total monotonically decreasing function $f : \mathbb{N} \rightarrow \{0, \cdots, n\}$, for $n \in \mathbb{N}$, there exists an $x \in \mathbb{N}$ such that for all $y \in \mathbb{N}$, where $x \leq y$, it is the case that $f(x) = f(y)$.

- Notice that the end sequent contains a Σ_2 formula and needs to be skolemized.
- The cut formula is as follows:

$$\exists x \forall y \left(\left((x \leq y) \rightarrow n + 1 = f(y) \right) \vee f(y) < n + 1 \right)$$

Schematic Clause Set Terms

$$cl^{\varphi,\Omega(0)}(0) \oplus \left(\{\vdash f(\alpha) < 0\} \otimes \left(\{\vdash 0 = f(\alpha)\} \otimes \{0 \leq \beta \vdash\} \right) \right) = CL_{ECA}(0)$$

$$\{f(\alpha) < 0 \vdash\} \oplus \left(\{f(g(\alpha)) < 0 \vdash\} \oplus \left(\{\vdash \alpha \leq \alpha\} \oplus \left(\{\vdash \alpha \leq g(\alpha)\} \oplus \{0 = f(\alpha), 0 = f(g(\alpha)) \vdash\} \right) \right) \right) = cl^{\varphi,\Omega(0)}(0)$$

Schematic Clause Set Terms

$$
\begin{tikzcd}[column sep=small]
 & \{0 \leq \beta \vdash\} \arrow[d] \\
\{\vdash n+1 = f(\alpha)\} \arrow[r] & \otimes \arrow[d] \\
\{\vdash f(\alpha) < n+1\} \arrow[r] & \otimes \arrow[d] \\
cl^{\varphi,\Omega(n+1)}(n+1) \arrow[r] & \oplus \arrow[d, phantom, "\shortparallel"] \\
 & {\color{red} CL_{ECA}(n+1)}
\end{tikzcd}
$$

$$
\begin{tikzcd}[column sep=small]
 & \{f(\alpha) < n+1, \alpha \leq \beta \vdash n = f(\beta), f(\beta) < n\} \arrow[d] \\
\{f(\beta) < n+1 \vdash f(\beta) < n+1\} \arrow[r] & \oplus \arrow[d] \\
\{\alpha \leq \beta \vdash \alpha \leq \beta\} \arrow[r] & \oplus \arrow[d] \\
\{n+1 = f(\beta) \vdash n+1 = f(\beta)\} \arrow[r] & \oplus \arrow[d] \\
\{\alpha \leq g(\alpha)\} \arrow[r] & \oplus \arrow[d] \\
\{\vdash \alpha \leq \alpha\} \arrow[r] & \oplus \arrow[d] \\
\{n+1 = f(\alpha), n+1 = f(g(\alpha)) \vdash\} \arrow[r] & \oplus \arrow[d] \\
cl^{\varphi,\Omega(n)}(n) \arrow[r] & \oplus \arrow[d, phantom, "\shortparallel"] \\
 & {\color{red} cl^{\varphi,\Omega(n+1)}(n+1)}
\end{tikzcd}
$$

— Note that $g(\cdot)$ is the skolem symbol. Also,

$$\Omega(n) \equiv \exists x \forall y \left(((x \leq y) \rightarrow n+1 = f(y)) \vee f(y) < n+1\right).$$

Schematic Clause Set Terms

$$
\begin{tikzcd}
 & \{0 \leq \beta \vdash\} \arrow[d] \\
\{\vdash n+1 = f(\alpha)\} \arrow[r] & \otimes \arrow[d] \\
\{\vdash f(\alpha) < n+1\} \arrow[r] & \otimes \arrow[d] \\
cl^{\varphi,\Omega(n+1)}(n+1) \arrow[r] & \oplus \arrow[d, equal] \\
 & CL_{ECA}(n+1)
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & \{f(\alpha) < n+1, \alpha \leq \beta \vdash n = f(\beta), f(\beta) < n\} \arrow[d] \\
\{f(\beta) < n+1 \vdash f(\beta) < n+1\} \arrow[r] & \oplus \arrow[d] \\
\{\alpha \leq \beta \vdash \alpha \leq \beta\} \arrow[r] & \oplus \arrow[d] \\
\{n+1 = f(\beta) \vdash n+1 = f(\beta)\} \arrow[r] & \oplus \arrow[d] \\
\{\alpha \leq g(\alpha)\} \arrow[r] & \oplus \arrow[d] \\
\{\vdash \alpha \leq \alpha\} \arrow[r] & \oplus \arrow[d] \\
\{n+1 = f(\alpha), n+1 = f(g(\alpha)) \vdash\} \arrow[r] & \oplus \arrow[d] \\
\mathit{Recursion} \Rightarrow cl^{\varphi,\Omega(n)}(n) \arrow[r] & \oplus \arrow[d, equal] \\
 & cl^{\varphi,\Omega(n+1)}(n+1)
\end{tikzcd}
$$

— Note that $g(\cdot)$ is the skolem symbol. Also,

$$\Omega(n) \equiv \exists x \forall y\, (((x \leq y) \rightarrow n+1 = f(y)) \vee f(y) < n+1).$$

schematic clause

- After extracting the schematic clause set terms, we perform normalization and tautology elimination.
- This results in the following schematic clause set:

$$C1(x,k) \equiv \ \vdash x(k) \leq x(k)$$

$$C2(x,k) \equiv \ \vdash x(k) \leq g(x(k))$$

$$C3(x,i,k) \equiv \quad i = f(x(k)), i = f(g(x(k))) \vdash$$

$$\begin{aligned} C4(x,y,i,k) \equiv \quad & y(k) \leq x(k), f(y(k)) < i+1 \vdash \\ & f(x(k)) < i, i = f(x(k)) \end{aligned}$$

$$\begin{aligned} C4'(x,y,i,k) \equiv \quad & y(k) \leq x(k+1), f(y(k)) < i+1 \vdash \\ & f(x(k+1)) < i, i = f(x(k+1)) \end{aligned}$$

$$C5(x,k) \equiv \quad f(x(k)) < 0 \vdash$$

$$C6(x,k) \equiv \quad f(g(x(k))) < 0 \vdash$$

$$C7(x,k) \equiv \quad 0 \leq x(k) \vdash f(x(k)) < n, f(x(k)) = n$$

schematic clause

- After extracting the schematic clause set terms, we perform normalization and tautology elimination.
- This results in the following schematic clause set:

$$C1(x,k) \equiv \ \vdash x(k) \leq x(k)$$

$$C2(x,k) \equiv \ \vdash x(k) \leq g(x(k))$$

$$C3(x,i,k) \equiv \ \ i = f(x(k)), i = f(g(x(k))) \vdash$$

$$C4(x,y,i,k) \equiv \ \ y(k) \leq x(k), f(y(k)) < i+1 \vdash f(x(k)) < i, i = f(x(k))$$

$$C4'(x,y,i,k) \equiv \ \ y(k) \leq x(k+1), f(y(k)) < i+1 \vdash f(x(k+1)) < i, i = f(x(k+1))$$

$$C5(x,k) \equiv \ \ f(x(k)) < 0 \vdash$$

$$C6(x,k) \equiv \ \ f(g(x(k))) < 0 \vdash$$

$$C7(x,k) \equiv \ \ 0 \leq x(k) \vdash f(x(k)) < n, f(x(k)) = n$$

- Note that x is what is referred to as a **schematic variable**, a variable of type $\omega \rightarrow \iota$.
- This is similar to a second order variable, however, instantiation of k produces a first order variable.

Refutation of the Clause set

— Once we extract the clause set we need a refutation in order to construct the ACNF.

— A problems with Schematic CERES is that both finding the substitution schema and a refutation are undecidable problems.

Refutation of the Clause set

- Once we extract the clause set we need a refutation in order to construct the ACNF.
- A problems with Schematic CERES is that both finding the substitution schema and a refutation are undecidable problems.
- However, instantiating the parameter n gives us a first order clause set.
- For example, the following is the clause set for $n = 3$.

$$\vdash x(k) \leq x(k)$$
$$\vdash x(k) \leq g(x(k))$$
$$0 = f(x(k)), 0 = f(g(x(k))) \vdash$$
$$1 = f(x(k)), 1 = f(g(x(k))) \vdash$$
$$2 = f(x(k)), 2 = f(g(x(k))) \vdash$$
$$3 = f(x(k)), 3 = f(g(x(k))) \vdash$$
$$y(k) \leq x(k), f(y(k)) < 1 \vdash$$
$$f(x(k)) < 0, 0 = f(x(k))$$
$$y(k) \leq x(k), f(y(k)) < 2 \vdash$$
$$f(x(k)) < 1, 1 = f(x(k))$$
$$y(k) \leq x(k), f(y(k)) < 3 \vdash$$
$$f(x(k)) < 2, 2 = f(x(k))$$

$$y(k) \leq x(k+1), f(y(k)) < 1 \vdash$$
$$f(x(k+1)) < 0, 0 = f(x(k+1))$$
$$y(k) \leq x(k+1), f(y(k)) < 2 \vdash$$
$$f(x(k+1)) < 1, 1 = f(x(k+1))$$
$$y(k) \leq x(k+1), f(y(k)) < 3 \vdash$$
$$f(x(k+1)) < 2, 2 = f(x(k+1))$$
$$f(x(k)) < 0 \vdash$$
$$f(g(x(k))) < 0 \vdash$$
$$0 \leq x(k) \vdash f(x(k)) < 3, f(x(k)) = 3$$

Refutation of the Clause set

- These first-order clause sets can be fed to a theorem prover.
- Though, the refutations resulting from theorem provers are not directly usable for constructing an ACNF.

Refutation of the Clause set

- These first-order clause sets can be fed to a theorem prover.
- Though, the refutations resulting from theorem provers are not directly usable for constructing an ACNF.
- We can use them to find a resolution invariant and construct a schematic resolution refutation.

Refutation of the Clause set

- These first-order clause sets can be fed to a theorem prover.
- Though, the refutations resulting from theorem provers are not directly usable for constructing an ACNF.
- We can use them to find a resolution invariant and construct a schematic resolution refutation.

310[0:MRR:309.0,306.1]	$\vdash\ f(\alpha) < 3$
311[0:MRR:10.1,310.0]	$\alpha \leq \beta \vdash\ 2 = f(\beta)\quad f(\beta) < 2$
312[0:Res:2.0,311.0]	$\vdash\ 2 = f(\alpha)\quad f(\alpha) < 2$
314[0:Res:312.0,6.1]	$2 = f(\alpha)\ \vdash\ f(g(\beta)) < 2$
315[0:Res:314.1,11.1]	$2 = f(\alpha)\quad g(\alpha) \leq \beta\ \vdash\ 1 = f(\beta)\quad f(\beta) < 1$
316[0:Res:312.0,315.0]	$g(\alpha) \leq \beta\ \vdash\ f(\alpha) < 2\ \ 1 = f(\beta)\quad f(\beta) < 1$
317[0:Res:2.0,316.0]	$\vdash\ f(\alpha) < 2\ \ 1 = f(g(\alpha))\quad f(g(\alpha)) < 1$
318[0:Res:3.0,316.0]	$\vdash\ f(\alpha) < 2\ \ 1 = f(g(g(\alpha)))\quad f(g(g(\alpha))) < 1$
321[0:Res:318.1,7.1]	$1 = f(g(\alpha))\ \vdash\ f(\alpha) < 2\ \ f(g(g(\alpha))) < 1$
322[0:Res:321.2,14.1]	$1 = f(g(\alpha))\quad g(g(\alpha)) \leq \beta\ \vdash\ f(\alpha) < 2\ \ 0 = f(\beta)$
325[0:Res:317.1,322.0]	$g(g(\alpha)) \leq \beta\ \vdash\ f(\alpha) < 2\ \ f(g(\alpha)) < 1\ \ f(\alpha) < 2\ \ 0 = f(\beta)$
327[0:Obv:325.1]	$g(g(\alpha)) \leq \beta\ \vdash\ f(g(\alpha)) < 1\ \ f(\alpha) < 2\ \ 0 = f(\beta)$
328[0:Res:2.0,327.0]	$\vdash\ f(g(\alpha)) < 1\ \ f(\alpha) < 2\ \ 0 = f(g(g(\alpha)))$
329[0:Res:3.0,327.0]	$\vdash\ f(g(\alpha)) < 1\ \ f(\alpha) < 2\ \ 0 = f(g(g(g(\alpha))))$
335[0:Res:329.2,8.1]	$0 = f(g(g(\alpha)))\ \vdash\ f(g(\alpha)) < 1\ \ f(\alpha) < 2$
336[0:MRR:335.0,328.2]	$\vdash\ f(g(\alpha)) < 1\ \ f(\alpha) < 2$
337[0:Res:336.0,14.1]	$g(\alpha) \leq \beta\ \vdash\ f(\alpha) < 2\ \ 0 = f(\beta)$
338[0:Res:2.0,337.0]	$\vdash\ f(\alpha) < 2\ \ 0 = f(g(\alpha))$
339[0:Res:3.0,337.0]	$\vdash\ f(\alpha) < 2\ \ 0 = f(g(g(\alpha)))$
344[0:Res:339.1,8.1]	$0 = f(g(\alpha))\ \vdash\ f(\alpha) < 2$
345[0:MRR:344.0,338.1]	$\vdash\ f(\alpha) < 2$

Spass output

n=5

Refutation of the Clause set

- These first-order clause sets can be fed to a theorem prover.
- Though, the refutations resulting from theorem provers are not directly usable for constructing an ACNF.
- We can use them to find a resolution invariant and construct a schematic resolution refutation.

310[0:MRR:309.0,306.1]	$\vdash f(\alpha) < 3$
311[0:MRR:10.1,310.0]	$\alpha \leq \beta \vdash 2 = f(\beta) \quad f(\beta) < 2$
312[0:Res:2.0,311.0]	$\vdash 2 = f(\alpha) \quad f(\alpha) < 2$
314[0:Res:312.0,6.1]	$2 = f(\alpha) \vdash f(g(\beta)) < 2$
315[0:Res:314.1,11.1]	$2 = f(\alpha) \quad g(\alpha) \leq \beta \vdash 1 = f(\beta) \quad f(\beta) < 1$
316[0:Res:312.0,315.0]	$g(\alpha) \leq \beta \vdash f(\alpha) < 2 \quad 1 = f(\beta) \quad f(\beta) < 1$
317[0:Res:2.0,316.0]	$\vdash f(\alpha) < 2 \quad 1 = f(g(\alpha)) \quad f(g(\alpha)) < 1$
318[0:Res:3.0,316.0]	$\vdash f(\alpha) < 2 \quad 1 = f(g(g(\alpha))) \quad f(g(g(\alpha))) < 1$
321[0:Res:318.1,7.1]	$1 = f(g(\alpha)) \vdash f(\alpha) < 2 \quad f(g(g(\alpha))) < 1$
322[0:Res:321.2,14.1]	$1 = f(g(\alpha)) \quad g(g(\alpha)) \leq \beta \vdash f(\alpha) < 2 \quad 0 = f(\beta)$
325[0:Res:317.1,322.0]	$g(g(\alpha)) \leq \beta \vdash f(\alpha) < 2 \quad f(g(\alpha)) < 1 \quad f(\alpha) < 2 \quad 0 = f(\beta)$
327[0:Obv:325.1]	$g(g(\alpha)) \leq \beta \vdash f(g(\alpha)) < 1 \quad f(\alpha) < 2 \quad 0 = f(\beta)$
328[0:Res:2.0,327.0]	$\vdash f(g(\alpha)) < 1 \quad f(\alpha) < 2 \quad 0 = f(g(g(\alpha)))$
329[0:Res:3.0,327.0]	$\vdash f(g(\alpha)) < 1 \quad f(\alpha) < 2 \quad 0 = f(g(g(g(\alpha))))$
335[0:Res:329.2,8.1]	$0 = f(g(g(\alpha))) \vdash f(g(\alpha)) < 1 \quad f(\alpha) < 2$
336[0:MRR:335.0,328.2]	$\vdash f(g(\alpha)) < 1 \quad f(\alpha) < 2$
337[0:Res:336.0,14.1]	$g(\alpha) \leq \beta \vdash f(\alpha) < 2 \quad 0 = f(\beta)$
338[0:Res:2.0,337.0]	$\vdash f(\alpha) < 2 \quad 0 = f(g(\alpha))$
339[0:Res:3.0,337.0]	$\vdash f(\alpha) < 2 \quad 0 = f(g(g(\alpha)))$
344[0:Res:339.1,8.1]	$0 = f(g(\alpha)) \vdash f(\alpha) < 2$
345[0:MRR:344.0,338.1]	$\vdash f(\alpha) < 2$

Spass output

n=5

Notice the invariant used by Spass.

Refutation of the Clause set

- After running a Spass on a few clause set instantiations a pattern emerged

Refutation of the Clause set

- After running a Spass on a few clause set instantiations a pattern emerged
- First we found the substitution schema:

$$\vartheta = [x \leftarrow \lambda k.(h(k)), y \leftarrow \lambda k.(h(k))]$$

where $h(\cdot)$ is defined as $h(0) \rightarrow 0,\ \ h(s(k)) \rightarrow g(h(k))$.

Refutation of the Clause set

- After running a Spass on a few clause set instantiations a pattern emerged
- First we found the substitution schema:

$$\vartheta = [x \leftarrow \lambda k.(h(k)), y \leftarrow \lambda k.(h(k))]$$

  where $h(\cdot)$ is defined as $h(0) \to 0,\ h(s(k)) \to g(h(k))$.
- However, concerning the refutation, we unexpectedly required computation in the ω sort.

Refutation of the Clause set

- After running a Spass on a few clause set instantiations a pattern emerged
- First we found the substitution schema:

$$\vartheta = [x \leftarrow \lambda k.(h(k)), y \leftarrow \lambda k.(h(k))]$$

  where $h(\cdot)$ is defined as $h(0) \rightarrow 0,\ \ h(s(k)) \rightarrow g(h(k))$.
- However, concerning the refutation, we unexpectedly required computation in the ω sort.
- As the induction parameter values decreases the term depth increases!!
- This was unexpected, and there was no way around it.
- We ended up having to add scratch pad memory to the definition of schematic resolution refutation.

Refutation of the Clause set

- After running a Spass on a few clause set instantiations a pattern emerged
- First we found the substitution schema:

$$\vartheta = [x \leftarrow \lambda k.(h(k)), y \leftarrow \lambda k.(h(k))]$$

  where $h(\cdot)$ is defined as $h(0) \rightarrow 0,\ h(s(k)) \rightarrow g(h(k))$.
- However, concerning the refutation, we unexpectedly required computation in the ω sort.
- As the induction parameter values decreases the term depth increases!!
- This was unexpected, and there was no way around it.
- We ended up having to add scratch pad memory to the definition of schematic resolution refutation.

A resolution proof schema $\mathcal{R}(n)$ is a structure $(\varrho_1, \cdots, \varrho_\alpha)$ together with a set of rewrite rules $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_\alpha$, where the $\mathcal{R}_i$ (*for* $1 \leq i \leq \alpha$) are pairs of rewrite rules

$$\varrho_i(0, \overline{w}, \overline{u}, \overline{X}) \rightarrow \eta_i \text{ and } \varrho_i(k+1, \overline{w}, \overline{u}, \overline{X}) \rightarrow \eta_i'$$

where, $\overline{w}$, $\overline{u}$, and $\overline{X}$ are vectors of ω, schematic, and clause variables respectively, η_i is a resolution term over terms of the form $\varrho_j(a_j, \overline{m}, \overline{t}, \overline{C})$ for $i < j \leq \alpha$, and η_i' is a resolution term over terms of the form $\varrho_j(a_j, \overline{m}, \overline{t}, \overline{C})$ and $\varrho_i(k, \overline{m}, \overline{t}, \overline{C})$ for $i < j \leq \alpha$; by a_j, we denote a term of the ω sort.

Refutation of the Clause set

— These scratch pad variables are used in our refutation as follows:

$$\varrho_4(n+1,k,x,y,Y)\Rightarrow$$

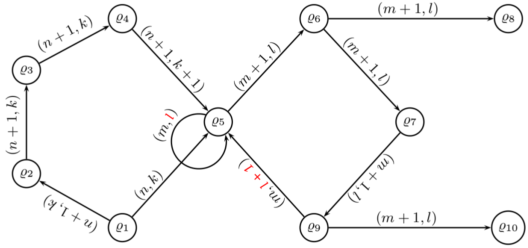
$$r(\varrho_5(n,k+1,x,y,Y\circ f(x(k+1))<n+1\vdash);r(C2(x,k);C7(x,k+1);f(x(k+1))<n+1)$$

Refutation of the Clause set

— These scratch pad variables are used in our refutation as follows:

$$\varrho_4(n+1,k,x,y,Y)\Rightarrow$$

$$r(\varrho_5(n,k+1,x,y,Y\circ f(x(k+1))<n+1\vdash);r(C2(x,k);C7(x,k+1);f(x(k+1))<n+1)$$



From Refutation to Herbrand System

- The schematic refutation is represented by the following term:

$$\rho_{ECA}(\gamma)=\varrho_1(n,k,x,y,Y)\theta\nu\vartheta[n\leftarrow\gamma]=\varrho_1(\gamma,\overline{\mu},\lambda_k.(i_s(k)),\lambda_k.(h(k)),\vdash)$$

where $\gamma \in \mathbb{N}$, ϑ is the substitution schema, the clause substitution is $\theta = \{Y \leftarrow \vdash\}$, the ω-variable substitution is $\nu = \{k \leftarrow \overline{\mu}\}$ for any $\overline{\mu} \in \mathbb{N}$, and $i_s(0) = 0,\ i_s(s(k)) = s(i_s(k))$.

From Refutation to Herbrand System

- The schematic refutation is represented by the following term:

$$\rho_{ECA}(\gamma) = \varrho_1(n,k,x,y,Y)\theta\nu\vartheta[n \leftarrow \gamma] = \varrho_1(\gamma,\overline{\mu},\lambda_k.(i_s(k)),\lambda_k.(h(k)),\vdash)$$

  where $\gamma \in \mathbb{N}$, ϑ is the substitution schema, the clause substitution is $\theta = \{Y \leftarrow \vdash\}$, the ω-variable substitution is $\nu = \{k \leftarrow \overline{\mu}\}$ for any $\overline{\mu} \in \mathbb{N}$, and $i_s(0) = 0$, $i_s(s(k)) = s(i_s(k))$.
- The pair $(\rho_{ECA}(\gamma), \vartheta)$ is the ACNF of the proof schema
- Our next step is to derive a Herbrand system based on the following sps-schema derived from the end sequent:

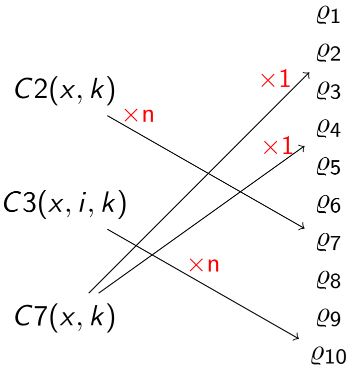
$$S(n) \equiv \left(\forall x \bigvee_{i=0}^{n} i = f(x), \forall y\, (0 \leq y \rightarrow f(y) \leq f(0))\right) \vdash \exists x\, (x \leq g(x) \rightarrow f(x) = f(g(x)))$$

Herbrand System Extraction

- To derive a Herbrand System we need to investigate how the end sequent formulae are passed down the ACNF.
- Only clauses C2, C3 and C7 have corresponding end sequent formulae.

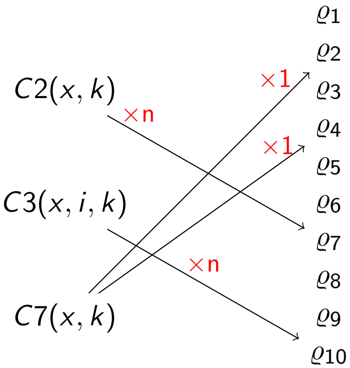
Herbrand System Extraction

- To derive a Herbrand System we need to investigate how the end sequent formulae are passed down the ACNF.
- Only clauses C2, C3 and C7 have corresponding end sequent formulae.



Herbrand System Extraction

- To derive a Herbrand System we need to investigate how the end sequent formulae are passed down the ACNF.
- Only clauses C2, C3 and C7 have corresponding end sequent formulae.



- This results in the following substitution rewrite system.

$$\mathcal{R} = \left\{ \begin{array}{l} w_1^{\varphi}(k+1) \Rightarrow [[0];[g(0)]] \\ w_1^{\varphi}(0) \Rightarrow [[0];[g(0)]] \\ \\ w_2^{\varphi}(k+1) \Rightarrow [[0];[g(0)]] \\ w_2^{\varphi}(0) \Rightarrow [[0];[g(0)]] \\ \\ w_1^{\psi}(k+1) \Rightarrow [[h(k+1)]; w_1^{\psi}(k)] \\ w_1^{\psi}(0) \Rightarrow [0] \end{array} \right.$$

Herbrand System

— Putting everything together we get the following Herbrand sequent.

$$\bigvee_{i=0}^{n} i = f(0), \bigvee_{i=0}^{n} i = f(g(0)), (0 \leq 0 \rightarrow f(0) \leq f(0)),$$

$$(0 \leq g(0) \rightarrow f(g(0)) \leq f(0)) \vdash \bigvee_{i=0}^{n} (h(i) \leq g(h(i)) \rightarrow f(h(i)) = f(g(h(i))))$$

Herbrand System

- Putting everything together we get the following Herbrand sequent.

$$\bigvee_{i=0}^{n} i = f(0), \bigvee_{i=0}^{n} i = f(g(0)), (0 \leq 0 \rightarrow f(0) \leq f(0)),$$

$$(0 \leq g(0) \rightarrow f(g(0)) \leq f(0)) \vdash \bigvee_{i=0}^{n} (h(i) \leq g(h(i)) \rightarrow f(h(i)) = f(g(h(i))))$$

- Interestingly, this sequent does not seem to be **LKE** provable.
- This occurs because we did not use tautological axioms.

Herbrand System

- Putting everything together we get the following Herbrand sequent.

$$\bigvee_{i=0}^{n} i = f(0), \bigvee_{i=0}^{n} i = f(g(0)), (0 \leq 0 \rightarrow f(0) \leq f(0)),$$

$$(0 \leq g(0) \rightarrow f(g(0)) \leq f(0)) \vdash \bigvee_{i=0}^{n} (h(i) \leq g(h(i)) \rightarrow f(h(i)) = f(g(h(i))))$$

- Interestingly, this sequent does not seem to be **LKE** provable.
- This occurs because we did not use tautological axioms.

$$f(\alpha) < n+1, \alpha \leq \beta \vdash n = f(\beta), f(\beta) < n$$

- We started from a specific axiom set that enforces the constraints placed on $f()$ and $g()$

Axioms for our Theory

— The axiom set is as follows:

$$
\begin{aligned}
&A1(i):\ \bigvee_{j=0}^{i-1} j = f(\alpha), i = f(g(\alpha)), f(g(\alpha)) < f(\alpha) \vdash \\
&A2(i):\ i = f(\alpha), \bigvee_{j=0}^{i-1} j = f(g(\alpha)), \alpha \leq g(\alpha) \vdash \\
&A3(i):\ i = f(\alpha), i = f(g(\alpha)) \vdash f(\alpha) = f(g(\alpha)) \\
&A4(i): f(g(\alpha)) = f(\alpha) \vdash f(\alpha) = f(g(\alpha)) \\
&A5(i): \vdash \alpha \leq \alpha \\
&A6(i): f(\alpha) < f(\alpha) \vdash
\end{aligned}
$$

Axioms for our Theory

- The axiom set is as follows:

$$
\begin{aligned}
&A1(i): \ \bigvee_{j=0}^{i-1} j = f(\alpha), i = f(g(\alpha)), f(g(\alpha)) < f(\alpha) \vdash \\
&A2(i): \ i = f(\alpha), \bigvee_{j=0}^{i-1} j = f(g(\alpha)), \alpha \leq g(\alpha) \vdash \\
&A3(i): \ i = f(\alpha), i = f(g(\alpha)) \vdash f(\alpha) = f(g(\alpha)) \\
&A4(i): f(g(\alpha)) = f(\alpha) \vdash f(\alpha) = f(g(\alpha)) \\
&A5(i): \vdash \alpha \leq \alpha \\
&A6(i): f(\alpha) < f(\alpha) \vdash
\end{aligned}
$$

- Interestingly enough a minimal Herbrand sequent can have the induction completely removed.

$$
\bigvee_{i=0}^{n} i = f(0), \bigvee_{i=0}^{n} i = f(g(0)), (0 \leq g(0) \rightarrow f(g(0)) \leq f(0)),
$$

$$
(0 \leq 0 \rightarrow f(0) \leq f(0)) \vdash 0 \leq g(0) \rightarrow f(0) = f(g(0)).
$$

Conclusion

- We gave a brief introduction to the important features of the schematic CERES method as well as the historical context leading to its development.

Conclusion

- We gave a brief introduction to the important features of the schematic CERES method as well as the historical context leading to its development.
    1) Our goal was to take the proof of a simple mathematical statement, a proof containing cuts, and analyse it using the method.

Conclusion

— We gave a brief introduction to the important features of the schematic CERES method as well as the historical context leading to its development.

1) Our goal was to take the proof of a simple mathematical statement, a proof containing cuts, and analyse it using the method.
2) Unfortunately, the proof chosen initially, NIA-schema way too complex for the current method.

Conclusion

- We gave a brief introduction to the important features of the schematic CERES method as well as the historical context leading to its development.
  1) Our goal was to take the proof of a simple mathematical statement, a proof containing cuts, and analyse it using the method.
  2) Unfortunately, the proof chosen initially, NIA-schema way too complex for the current method.
  3) We weakened the cuts of the NIA-schema and derived a new proof, the ECA-schema, which we can handle.

Conclusion

- We gave a brief introduction to the important features of the schematic CERES method as well as the historical context leading to its development.
  1) Our goal was to take the proof of a simple mathematical statement, a proof containing cuts, and analyse it using the method.
  2) Unfortunately, the proof chosen initially, NIA-schema way too complex for the current method.
  3) We weakened the cuts of the NIA-schema and derived a new proof, the ECA-schema, which we can handle.
- We then carried out the proof analysis of the ECA-schema and algorithmically derived a Herbrand system.

Open Problems and Future Work

- Most importantly we want to know why the ECA-schema was easy to handle while the NIA-schema was not.

Open Problems and Future Work

- Most importantly we want to know why the ECA-schema was easy to handle while the NIA-schema was not.
- Also, we have found a more expressive resolution refutation schema for the NIA-schema. We are currently investigating how general the new resolution refutation schema is.

Open Problems and Future Work

- Most importantly we want to know why the ECA-schema was easy to handle while the NIA-schema was not.
- Also, we have found a more expressive resolution refutation schema for the NIA-schema. We are currently investigating how general the new resolution refutation schema is.
- If so, what parts of the current method need to be generalized.

Open Problems and Future Work

- Most importantly we want to know why the ECA-schema was easy to handle while the NIA-schema was not.
- Also, we have found a more expressive resolution refutation schema for the NIA-schema. We are currently investigating how general the new resolution refutation schema is.
- If so, what parts of the current method need to be generalized.
- Is there a universal resolution calculus for all primitive recursive proof schemata?

Open Problems and Future Work

- Most importantly we want to know why the ECA-schema was easy to handle while the NIA-schema was not.
- Also, we have found a more expressive resolution refutation schema for the NIA-schema. We are currently investigating how general the new resolution refutation schema is.
- If so, what parts of the current method need to be generalized.
- Is there a universal resolution calculus for all primitive recursive proof schemata?
- If not, what class of proof schema is the current method complete for?

Thank you for your time.