Computational content of circular proof systems.

Denis Kuperberg    Laureline Pinault    Damien Pous

LIP, ENS Lyon

Séminaire de l’équipe Méthodes Formelles
Bordeaux

Tuesday 21st January 2020
Cyclic proofs

Regular expressions

\[ e, f := 1 \mid a \in A \mid e \cdot f \mid e + f \mid e^* \]

Context: Cyclic proofs for inclusion of expressions [Das, Pous ’17]

- Infinite proof trees, with root of the form \( e \vdash f \).

\[
\begin{align*}
1 \vdash 1 & \quad \text{(Ax)} \\
1 \vdash a^* & \quad \text{(Ax)} \\
\hline
a^* \vdash a^* & \quad \text{(Ax)} \\
\hline
a, a^* \vdash a^* & \\
\hline
\end{align*}
\]

\[ a^* \vdash a^* \]
Cyclic proofs

Regular expressions

\[ e, f := 1 \mid a \in A \mid e \cdot f \mid e + f \mid e^* \]

Context: Cyclic proofs for inclusion of expressions [Das, Pous ’17]

- Infinite proof trees, with root of the form \( e \vdash f \).
  
  \[
  \begin{array}{c}
  1 \vdash 1 \\
  \hline
  (Ax) \\
  \end{array} \quad
  \begin{array}{c}
  a \vdash a \\
  \hline
  (Ax) \\
  \end{array} \quad
  \begin{array}{c}
  a^* \vdash a^* \\
  \hline
  a, a^* \vdash a^* \\
  \end{array} \quad
  \begin{array}{c}
  a^* \vdash a^* \\
  \hline
  \end{array}
  \]

- Validity condition on infinite branches
Cyclic proofs

Regular expressions

\[ e, f := 1 \mid a \in A \mid e \cdot f \mid e + f \mid e^* \]

Context: Cyclic proofs for inclusion of expressions [Das, Pous ’17]

- Infinite proof trees, with root of the form \( e \vdash f \).

\[
\begin{align*}
1 \vdash 1 \quad & (Ax) \\
1 \vdash a^* \quad & a \vdash a \quad (Ax) \\
& a^* \vdash a^* \\
& a, a^* \vdash a^*
\end{align*}
\]

- Validity condition on infinite branches

- \( \exists \) proof of \( e \vdash f \iff L(e) \subseteq L(f) \).
Computational interpretation

Proof of $e \vdash f$

Program with input from $e$ and output in $f$. 
Computational interpretation

Proof of $e \vdash f$

Program with input from $e$ and output in $f$.

Several proofs of the same statement

Several programs of the same type

Example:

$a \vdash a + a$

$in_l$ or $in_r$
Computational interpretation

Proof of $e \vdash f$

Program with input from $e$ and output in $f$.

Several proofs of the same statement

$\iff$

Several programs of the same type

Example:

\[ a \vdash a + a \]

$in_l$ or $in_r$

Curry-Howard isomorphism, typed programming,\ldots

Well-understood for finite proofs, active field for infinite proofs.
Computing languages

- Boolean type $2 = 1 + 1$

- Add *structural* rules corresponding to simple natural programs

- Study the expressive power of regular proofs (finite graphs)

- Focus on proofs for *languages*:

  Proof $\pi$ of $A^* \vdash 2 \rightarrow$ Language $L(\pi) \subseteq A^*$
Proof system

Expressions $e := A \mid A^*$

Sequents $E, F = e_1, e_2, \ldots, e_n$

Proof system with extra rules for basic data manipulation:

\[
\begin{align*}
\vdash 2 & \quad (\text{tt}) \\
E, F \vdash 2 & \quad (\text{wkn}) \\
E, e, F \vdash 2 & \quad (\text{ctr}) \\
(E, F \vdash 2)_{a \in A} & \quad (\text{A}) \\
E, A, F \vdash 2 & \quad (*) \\
E, F \vdash 2 & \quad E, A, A^*, F \vdash 2 \\
E, A^*, F \vdash 2 & \\
\end{align*}
\]
Proofs as language acceptors

What are the languages computed by cyclic proofs?

Example on alphabet \{a, b\}: b^*
Proofs as language acceptors

What are the languages computed by cyclic proofs?

Example on alphabet \{a, b\}: \(b^*\)

Lemma

\textit{Without contraction, the system captures exactly regular languages.}
With contractions: what class of language?

Example on alphabet \( \{a, b\} \): \( a^n b^n \)

1st step: create a copy of the input and delete the first \( a \)'s.
With contractions: what class of language?

Example on alphabet \{a, b\}: \(a^nb^n\)

\[
\begin{align*}
\vdash 2 & \quad \text{(tt)} \quad \vdash 2 \\
A, A^* & \vdash 2 & \text{(wkn)} \\
A^* & \vdash 2 & \text{(*)}
\end{align*}
\]

\[
\begin{align*}
\vdash 2 & \quad \text{(ff)} \\
A^* & \vdash 2 & \text{(wkn)} \\
A, A^*, A^* & \vdash 2 & \text{(*)}
\end{align*}
\]

\[
\begin{align*}
(A^*, A^* \vdash 2)_a & \\
A^*, A, A^* & \vdash 2 \\
A^*, A^* & \vdash 2 & \text{(*)}
\end{align*}
\]

\[
\begin{align*}
(A^*, A^* \vdash 2)_b & \\
A^*, A, A^* & \vdash 2 \\
A^* & \vdash 2 & \text{(ctr)}
\end{align*}
\]

1st step: create a copy of the input and delete the first \(a\)’s.
With contractions: what class of language?

Example on alphabet \( \{a, b\} \): \( a^n b^n \)

\[ \vdash 2 \]
\[ \frac{\vdash 2}{A, A^* \vdash 2} \] (wkn)
\[ \frac{A^* \vdash 2}{A^* \vdash 2} \] (wkn)

\[ \frac{A^* \vdash 2}{A, A^* \vdash 2} \] (A)

\[ \frac{A^* \vdash 2}{A^* \vdash 2} \] (ctr)

1st step: create a copy of the input and delete the first \( a \)'s.
With contractions: what class of language?

Example on alphabet \( \{a, b\} \): \( a^n b^n \)

\[
\begin{array}{c}
A^* \vdash 2 \\
\hline
\frac{A^*, A^* \vdash 2}{(A^*, A^* \vdash 2)_a} (\text{ff})
\frac{(A^*, A^* \vdash 2)_b}{(A^*, A^* \vdash 2)_b} (\text{wkn})
\end{array}
\]

\[
\begin{array}{c}
A^* \vdash 2 \\
\hline
\frac{A^*, A^*, A^* \vdash 2}{(A^*, A^*, A^* \vdash 2)_a} (\text{wkn})
\frac{(A^*, A^*, A^* \vdash 2)_b}{(A^*, A^*, A^* \vdash 2)_b} (A)
\end{array}
\]

2nd step: check that for each \( b \) of the second copy we have a \( a \) in the first one.
With contractions: what class of language?

Example on alphabet \(\{a, b\}\): \(a^n b^n\)

\[
\begin{array}{c}
A^* \vdash 2 \\
\hline
(A^*, A^* \vdash 2)_a \\
\hline
A^*, A, A^* \vdash 2 \\
\hline
(A^*, A^* \vdash 2)_a \\
\hline
A^*, A^*, A^* \vdash 2 \\
\hline
(A^*, A^* \vdash 2)_b \\
\hline
\end{array}
\]

2nd step: check that for each \(b\) of the second copy we have a \(a\) in the first one.
With contractions: what class of language?

Example on alphabet \(\{a, b\}\): \(a^n b^n\)

2nd step: check that for each \(b\) of the second copy we have a \(a\) in the first one.
With contractions: what class of language?

Example on alphabet \( \{a, b\} \): \( a^n b^n \)

\[
\begin{array}{c}
A^* \vdash 2 \\
\hline
(A^*, A^* \vdash 2)_a \quad (A^*, A^* \vdash 2)_b \\
\hline
\end{array}
\]

\[
\begin{array}{c}
A^*, A, A^* \vdash 2 \\
\hline
(A^*, A^* \vdash 2)_a \\
\hline
(A^*, A^* \vdash 2)_b \\
\hline
\end{array}
\]

\[
\begin{array}{c}
A, A^*, A^* \vdash 2 \\
\hline
(A^*, A^* \vdash 2)_b \\
\hline
\end{array}
\]

2nd step: check that for each \( b \) of the second copy we have a \( a \) in the first one.
With contractions: what class of language?

Example on alphabet \( \{a, b\} \): \( a^n b^n \)

\[
\begin{align*}
&\vdash \ (\text{ff}) \\
\frac{\vdash \ 2}{(A* \vdash \ 2)_a} & \text{(wkn)} \quad \frac{\vdash \ 2}{(A* \vdash \ 2)_b} & \text{(wkn)} \quad \frac{\vdash \ 2}{(A* \vdash \ 2)} & \text{(A)} \\
\frac{\vdash \ 2}{A, A* \vdash \ 2} & \text{(*)} \\
\frac{\vdash \ 2}{A* \vdash \ 2}
\end{align*}
\]

3rd step: checking that we have no more \( a \)'s
With contractions: what class of language?

Example on alphabet \{a, b\}: \mathit a^n \mathit b^n

\[
\begin{array}{c}
\frac{}{\vdash 2} \quad (\mathit{ff})
\hline
\frac{}{(A^* \vdash 2)_a} \quad (\mathit{wkn})
\hline
\frac{}{A, A^* \vdash 2} \quad (A)
\hline
\frac{}{A^* \vdash 2} \quad (*)
\end{array}
\]

3rd step: checking that we have no more \(a\)'s

Example on alphabet \{a, b, c\}: \mathit a^n \mathit b^n \mathit c^n
With contractions: a new automaton model

### Jumping Multihead Automata

A JMA is an automaton with \( k \) reading heads.

**Transitions:**

\[
Q \times (A \cup \{\triangleleft\})^k \rightarrow Q \times \{\uparrow, \bigcirc, J_1, \ldots, J_k\}^k
\]

- \( \uparrow \): advance one step
- \( \bigcirc \): stay in place
- \( J_i \): jump to the position of head \( i \)
With contractions: a new automaton model

**Jumping Multihead Automata**

A JMA is an automaton with $k$ reading heads.

**Transitions:**

$$Q \times (A \cup \{\leftarrow\})^k \rightarrow Q \times \{\text{//}, \text{_spinner}, J_1, \ldots, J_k\}^k$$

- ◀: advance one step
- Spinner: stay in place
- $J_i$: jump to the position of head $i$

+ Equivalent of the validity criterion
Example of JMA

Example: \( \{ a^{2n} \mid n \in \mathbb{N} \} \) is accepted by a 2-head JMA.
Example of JMA

Example: \( \{a^{2n} \mid n \in \mathbb{N}\} \) is accepted by a 2-head JMA.
Example of JMA

Example: \( \{a^{2n} \mid n \in \mathbb{N}\} \) is accepted by a 2-head JMA.
Example of JMA

Example: \( \{ a^{2^n} \mid n \in \mathbb{N} \} \) is accepted by a 2-head JMA.
Example of JMA

Example: \( \{ a^{2^n} \mid n \in \mathbb{N} \} \) is accepted by a 2-head JMA.
Example of JMA

Example: \( \{a^{2^n} \mid n \in \mathbb{N}\} \) is accepted by a 2-head JMA.
Example of JMA

Example: \( \{a^{2^n} \mid n \in \mathbb{N}\} \) is accepted by a 2-head JMA.
Example of JMA

Example: \( \{ a^{2^n} \mid n \in \mathbb{N} \} \) is accepted by a 2-head JMA.
Example of JMA

Example: \( \{a^{2n} \mid n \in \mathbb{N}\} \) is accepted by a 2-head JMA.
Example of JMA

Example: \( \{a^{2^n} \mid n \in \mathbb{N}\} \) is accepted by a 2-head JMA.
Equivalence Theorem

**Theorem**

*Cyclic proofs and JMA recognize the same class of languages.*

- States of the automaton $\sim$ Positions in the proof tree
- Accepting / Rejecting state $\sim$ True / False axiom
- Multiple heads $\sim$ Multiple copies of $A^*$
- Reading a letter $\sim$ Applying $*$ and $(A)$ rules
Expressive power of JMA

Comparison with Multihead Automata in Literature:
[Holzer, Kutrib, Malcher 2008]

1-way Multihead \[\subseteq\] JMA \[\subseteq\] 2-way Multihead

Emptiness Undecidable

\[\forall k, JMA(2) \not\subseteq 1 \text{DFA}(k)\]

e.g. Palindroms

D. Kuperberg
Computational content of circular proof systems
Tuesday 21st January 2020 11 / 17
Expressive power of JMA

Comparison with Multihead Automata in Literature:

[Holzer, Kutrib, Malcher 2008]

1-way Multihead \( \subseteq \) JMA \( \subseteq \) 2-way Multihead

Emptiness Undecidable

LogSpace

\[ \forall k, JMA(2) \not\subseteq 1 \text{DFA}(k) \]? e.g. Palindroms

D. Kuperberg
Computational content of circular proof systems
Tuesday 21st January 2020 11 / 17
Expressive power of JMA

Comparison with Multihead Automata in Literature:
[Holzer, Kutrib, Malcher 2008]

1-way Multihead $\subseteq$ JMA $\subseteq$ 2-way Multihead

Emptiness Undecidable

$\exists k, \text{JMA}(2) \not\subseteq 1\text{DFA}(k)$

$\text{LogSpace}$
Expressive power of JMA

Comparison with Multihead Automata in Literature:
[Holzer, Kutrib, Malcher 2008]

\[ \forall k, \ JMA(2) \not\subseteq 1DFA(k) \]

1-way Multihead \( \subseteq \) JMA \( \subseteq \) 2-way Multihead

\textit{Emptiness Undecidable} \hspace{1cm} \text{LogSpace}
Expressive power of JMA

Comparison with Multihead Automata in Literature:
[Holzer, Kutrib, Malcher 2008]

\[ \forall k, \ JMA(2) \not\subseteq 1DFA(k) \]

1-way Multihead \subseteq JMA \subseteq 2-way Multihead

Emptiness Undecidable

LogSpace
Expressive power of JMA

Comparison with Multihead Automata in Literature:
[Holzer, Kutrib, Malcher 2008]

\[ \forall k, \text{JMA}(2) \not\subseteq 1\text{DFA}(k) \]

1-way Multihead \( \subseteq \) JMA \( \subseteq \) 2-way Multihead

\( \text{Emptiness Undecidable} \)

LogSpace
Simulating 2-ways automata

Theorem

*JMA have same expressive power as 2-way Multihead automata.*
Theorem

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.
Simulating 2-ways automata

**Theorem**

\textit{JMA have same expressive power as 2-way Multihead automata.}

Difficulty: simulate a left move of some head.

**Example:** \textit{Palindroms} = \{\(u \in \Sigma^* \mid u = u^R\}\} is accepted by a JMA.

\[\begin{array}{c}
\text{\textgreater} \ a \ a \ b \ c \ c \ b \ a \ a \ \textless \\
\end{array}\]
Simulating 2-ways automata

Theorem

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.

Example: *Palindroms* $= \{ u \in \Sigma^* \mid u = u^R \}$ is accepted by a JMA.

![Diagram of a JMA simulation](attachment:image.png)
Simulating 2-ways automata

Theorem

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.

Example: \( \text{Palindroms} = \{ u \in \Sigma^* \mid u = u^R \} \) is accepted by a JMA.
Simulating 2-ways automata

Theorem

**JMA have same expressive power as 2-way Multihead automata.**

Difficulty: simulate a left move of some head.

Example: *Palindroms* = \( \{ u \in \Sigma^* \mid u = u^R \} \) is accepted by a JMA.

\[ \begin{array}{cccccccc}
\triangleright & a & a & b & c & c & b & a & a \\
\end{array} \]
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.

**Example:** *Palindroms* = \( \{ u \in \Sigma^* \mid u = u^R \} \) is accepted by a JMA.
Simulating 2-ways automata

Theorem

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.

Example: *Palindroms* = \( \{ u \in \Sigma^* \mid u = u^R \} \) is accepted by a JMA.

\[
\begin{array}{cccccccc}
\triangleright & a & a & b & c & c & b & a & a \\
\end{array}
\]
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

**Difficulty:** simulate a left move of some head.

**Example:** *Palindroms* \(= \{u \in \Sigma^* \mid u = u^R\}\) is accepted by a JMA.
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

**Difficulty:** simulate a left move of some head.

**Example:** $\text{Palindroms} = \{ u \in \Sigma^* \mid u = u^R \}$ is accepted by a JMA.
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

**Difficulty:** simulate a left move of some head.

**Example:** *Palindroms* $= \{ u \in \Sigma^* \mid u = u^R \}$ is accepted by a JMA.
Simulating 2-ways automata

Theorem

JMA have same expressive power as 2-way Multihead automata.

Difficulty: simulate a left move of some head.

Example: \[ \text{Palindroms} = \{ u \in \Sigma^* \mid u = u^R \} \] is accepted by a JMA.
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.

**Example:** *Palindroms* $= \{ u \in \Sigma^* \mid u = u^R \}$ is accepted by a JMA.
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.

**Example:** \( \text{Palindroms} = \{ u \in \Sigma^* \mid u = u^R \} \) is accepted by a JMA.
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.

**Example:** *Palindroms* = \{u ∈ Σ* | u = u^R\} is accepted by a JMA.
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

**Difficulty:** simulate a left move of some head.

**Example:** *Palindroms* = \( \{ u \in \Sigma^* \mid u = u^R \} \) is accepted by a JMA.
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

**Difficulty:** simulate a left move of some head.

**Example:** *Palindroms* = \( \{ u \in \Sigma^* \mid u = u^R \} \) is accepted by a JMA.
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

**Difficulty:** simulate a left move of some head.

**Example:** $\text{Palindroms} = \{u \in \Sigma^* \mid u = u^R\}$ is accepted by a JMA.
Simulating 2-ways automata

Theorem

JMA have same expressive power as 2-way Multihead automata.

Difficulty: simulate a left move of some head.

Example: \( \text{Palindroms} = \{ u \in \Sigma^* | u = u^R \} \) is accepted by a JMA.
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

**Difficulty:** simulate a left move of some head.

**Example:** $\textit{Palindroms} = \{ u \in \Sigma^* \mid u = u^R \}$ is accepted by a JMA.

![Diagram of a 2-ways automaton simulating a left move.](image-url)
Simulating 2-ways automata

Theorem

**JMA have same expressive power as 2-way Multihead automata.**

Difficulty: simulate a left move of some head.

Example: \textit{Palindroms} = \{u \in \Sigma^* \mid u = u^R\} is accepted by a JMA.

\[
\begin{array}{c c c c c c c c c c c c}
\triangleright & a & a & b & c & c & b & a & a & \triangleright \\
\end{array}
\]
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

**Difficulty:** simulate a left move of some head.

**Example:** \( \text{Palindroms} = \{u \in \Sigma^* \mid u = u^R \} \) is accepted by a JMA.

\[
\begin{array}{cccccccc}
\triangleright & a & a & b & c & c & b & a & a & \triangleright \\
\end{array}
\]
Simulating 2-ways automata

Theorem

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.

Example: *Palindroms* = \( \{ u \in \Sigma^* \mid u = u^R \} \) is accepted by a JMA.

\[
\begin{array}{cccccc}
\triangleright & a & a & b & c & c & b & a & a & \triangleleft \\
\end{array}
\]
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.

**Example:** \( \text{Palindroms} = \{ u \in \Sigma^* \mid u = u^R \} \) is accepted by a JMA.

![Diagram of a JMA accepting a palindrome](image)
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.

**Example:** *Palindroms* = \( \{ u \in \Sigma^* \mid u = u^R \} \) is accepted by a JMA.

\[
\begin{array}{cccccccc}
\triangleright & a & a & b & c & c & b & a & a & \triangleright \\
\end{array}
\]
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.

**Example:** *Palindroms = \{ u ∈ Σ* | u = u^R \} is accepted by a JMA.*
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

Difficulty: simulate a left move of some head.

**Example:** $Palindroms = \{ u \in \Sigma^* \mid u = u^R \}$ is accepted by a JMA.

Generalization of this idea ⇒ Translation from 2DFA to JMA.
Simulating 2-ways automata

**Theorem**

*JMA have same expressive power as 2-way Multihead automata.*

**Difficulty:** simulate a left move of some head.

**Example:** \( \text{Palindroms} = \{ u \in \Sigma^* \mid u = u^R \} \) is accepted by a JMA.

![Diagram](image)

Generalization of this idea \( \Rightarrow \) Translation from 2DFA to JMA.

**Corollary**

*Cyclic proofs with contraction characterize LogSpace.*
What next?

- Add the cut rule:
  \[
  \frac{E \vdash e \quad e, F \vdash g}{E, F \vdash g}
  \]

- Corresponds to composition of functions

- Enriched expressions:
  \[
  e, f ::= 1 \mid a \mid e \cdot f \mid e + f \mid e^* \mid e \rightarrow f \mid e \cap f
  \]

- Sequents \((1^*)^k \vdash 1^*\): functions \(\mathbb{N}^k \rightarrow \mathbb{N}\)
What next?

- Add the cut rule:
  \[
  \frac{E \vdash e \quad e, F \vdash g}{E, F \vdash g}
  \]

- Corresponds to composition of functions

- Enriched expressions:
  \[
  e, f := 1 \mid a \mid e \cdot f \mid e + f \mid e^* \mid e \to f \mid e \cap f
  \]

- Sequents \((1^*)^k \vdash 1^*\): functions \(\mathbb{N}^k \to \mathbb{N}\)

How does it increase the expressive power?
An extended, resource-tracking System $T$

$\lambda$-calculus extended with pairs, singletons, sums, lists, and additive pairs ($i \in \{0, 1\}$):

\[
M, N, O ::= x \mid \lambda x. M \mid MN \\
\langle M, N \rangle \mid \text{let } \langle x, y \rangle := M \text{ in } N \\
\langle \rangle \mid \text{let } \langle \rangle := M \text{ in } N \\
i_i M \mid D(M; x.N; x.O) \\
\mathbb{1} \mid M ::= N \mid R(M; N; x.y.O) \\
\langle\langle M, N \rangle \rangle \mid p_i M
\]

Comes with a type system.

Example:

\[
\Gamma \vdash L : e^* \quad \Delta \vdash M : g \quad x : e, y : g \vdash N : g \\
\text{*-e} \quad \Gamma, \Delta \vdash R(L; M; x.y.N) : g
\]
Affine version $T_{aff}$

System $T_{aff}$: Cannot use contraction in typing derivations:

$$
\frac{
   x : e, x : e, \Gamma \vdash M : f
}{
   \Gamma \vdash \lambda x. \langle \langle x, x \rangle \rangle : f
}
$$
Affine version $T_{\text{aff}}$

System $T_{\text{aff}}$: Cannot use contraction in typing derivations:

$$
\frac{x : e, x : e, \Gamma \vdash M : f}{\Gamma \vdash c x : e, \Gamma \vdash M : f}
$$

Example:

- $\lambda x. \langle x, x \rangle$ is not typable in $T_{\text{aff}}$
- $\lambda x. \langle \langle x, x \rangle \rangle$ is typable in $T_{\text{aff}}$
Results

**Theorem**

\[
\begin{align*}
\text{Cyclic proofs without contraction} & \iff \text{System } T_{\text{aff}} \\
\text{Cyclic proofs with contraction} & \iff \text{System } T \text{ functions} \iff \text{Peano}
\end{align*}
\]
Results

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Cyclic proofs without contraction</em></td>
</tr>
<tr>
<td>⇐⇒</td>
</tr>
<tr>
<td><em>System T</em> aff</td>
</tr>
<tr>
<td><em>Cyclic proofs with contraction</em></td>
</tr>
<tr>
<td>functions</td>
</tr>
<tr>
<td>⇐⇒</td>
</tr>
<tr>
<td><em>System T</em> functions</td>
</tr>
<tr>
<td>⇐⇒</td>
</tr>
<tr>
<td><em>Peano</em></td>
</tr>
</tbody>
</table>

Results

Theorem

\begin{align*}
\textit{Cyclic proofs without contraction} & \iff \textit{System } T_{\text{aff}} \\
\textit{Cyclic proofs with contraction} & \iff \textit{System } T \quad \text{functions} \quad \iff \textit{Peano}
\end{align*}

System $T_{\text{[affine]}} \rightarrow$ proofs: easy.

Affine proofs $\rightarrow T_{\text{aff}}$:

- normal form for proofs, with explicit hierarchy of cycles
- inductively build $T_{\text{aff}}$ terms, crucial use of additive pairs
Results

| Theorem |
|------------------|------------------|
| Cyclic proofs without contraction ⇐⇒ System \( T_{\text{aff}} \) |
| Cyclic proofs with contraction \( \iff \) System \( T \) \( \iff \) Peano |

System \( T \) \( [\text{affine}] \) \( \rightarrow \) proofs: easy.

Affine proofs \( \rightarrow \) \( T_{\text{aff}} \):
- normal form for proofs, with explicit hierarchy of cycles
- inductively build \( T_{\text{aff}} \) terms, crucial use of additive pairs

Proofs \( \rightarrow \) system \( T \):
Show termination in ACA0 + conservativity results.
Ongoing work

Conjecture

$T_{aff}$ computes exactly primitive recursive functions
Ongoing work

Conjecture

$T_{aff}$ computes exactly primitive recursive functions

Related result:

Theorem (Dal Lago 2009)

$T_{aff}$ without additive pairs $\iff$ primitive recursive functions

Does it still hold with additive pairs $\langle M, N \rangle$?
Ongoing work

<table>
<thead>
<tr>
<th>Conjecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{aff}$ computes exactly primitive recursive functions</td>
</tr>
</tbody>
</table>

Related result:

<table>
<thead>
<tr>
<th>Theorem (Dal Lago 2009)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{aff}$ without additive pairs $\iff$ primitive recursive functions</td>
</tr>
</tbody>
</table>

Does it still hold with additive pairs $\langle M, N \rangle$?

Other direction: More constructive proof for general System $T$?
Conjecture

$T_{\text{aff}}$ computes exactly primitive recursive functions

Related result:

**Theorem (Dal Lago 2009)**

$T_{\text{aff}}$ without additive pairs $\iff$ primitive recursive functions

Does it still hold with additive pairs $\langle M, N \rangle$?

**Other direction:** More constructive proof for general System $T$?

Thanks for your attention!