Jérôme Fortier\textsuperscript{1,2,3,4}

Computing with Circular Proofs
Joint work with Luigi Santocanale\textsuperscript{2}

Past: Université du Québec à Montréal\textsuperscript{1}/ Aix-Marseille Université\textsuperscript{2}
Present: John Abbott College\textsuperscript{3}
Future: University of Ottawa\textsuperscript{4}
Natural numbers

**Definition (circular)**

A *natural number* is either 0, or the expression \( \text{suc}(n) \), where \( n \) is a *natural number*. 
Natural numbers

Definition (circular)

A natural number is either 0, or the expression $\text{suc}(n)$, where $n$ is a natural number.

An algebra

$$1 + \mathbb{N} \xrightarrow{\{0, \text{suc}\}} \mathbb{N}$$
A natural number is either 0, or the expression \( \text{suc}(n) \), where \( n \) is a natural number. \( \mathbb{N} \) is the least fixpoint of this definition!
Definition (circular)

A natural number is either \(0\), or the expression \(suc(n)\), where \(n\) is a natural number. \(\mathbb{N}\) is the least fixpoint of this definition!
A natural number is either 0, or the expression \( \text{suc}(n) \), where \( n \) is a natural number. \( \mathbb{N} \) is the least fixpoint of this definition!

An initial algebra

\[
\begin{align*}
1 + \mathbb{N} &\xrightarrow{1 + f} 1 + X \\
\{0, \text{suc}\} &\xrightarrow{\text{pre}} \{a, g\} \\
\mathbb{N} &\xrightarrow{\exists! f} X \\
\end{align*}
\]

\[
\begin{aligned}
f(0) &= a \\
f(\text{suc}(x)) &= g(f(x)) \\
\mathbb{N} &= \mu (1 + \mathbb{N})
\end{aligned}
\]
Some more inductive types...
Some more inductive types...

- $\mu X. (1 + A \times X) = A^* = \text{Finite words over } A$

- $*$ $\mapsto$ $\varepsilon$
- $\langle a, w \rangle$ $\mapsto$ $a \cdot w$
Some more inductive types...

- $\mu X. (1 + A \times X) = A^* =$ Finite words over $A$
  
  \[
  \begin{align*}
  * & \mapsto \varepsilon \\
  \langle a, w \rangle & \mapsto a \cdot w
  \end{align*}
  \]

- $\mu X. (1 + A \times X \times X) =$ Finite labeled binary trees
  
  \[
  * \mapsto \text{Empty tree}
  \]

\[
\langle a, T_1, T_2 \rangle \mapsto \begin{array}{c}
  \text{a} \\
  \downarrow \\
  \text{T}_1 \\
  \text{T}_2
\end{array}
\]

Jérôme Fortier  
FMCS 2015, Colgate University
Streams

**Definition (circular)**

A *stream* over an alphabet $A$ is made of a *head* $a \in A$ and a *tail*, which is itself a *stream*. 
Streams

Definition (circular)

A stream over an alphabet $A$ is made of a head $a \in A$ and a tail, which is itself a stream.
Definition (circular)

A stream over an alphabet $A$ is made of a head $a \in A$ and a tail, which is itself a stream. $A^\omega$ is the greatest fixpoint of this definition!

A final coalgebra

$$
\begin{align*}
\exists! orb \\
Z \longrightarrow A^\omega \\
\langle a, f \rangle \quad \langle \text{head, tail} \rangle \\
A \times Z \longrightarrow A \times A^\omega \\
\text{id} \times orb
\end{align*}
$$

$$
\begin{align*}
\text{head}(orb(z)) &= a(z) \\
\text{tail}(orb(z)) &= orb(f(z))
\end{align*}
$$
Definition (circular)

A stream over an alphabet $A$ is made of a head $a \in A$ and a tail, which is itself a stream. $A^\omega$ is the greatest fixpoint of this definition!

A final coalgebra

$$
\begin{align*}
Z \xrightarrow{\exists! \text{orb}} A^\omega \\
\langle a, f \rangle \downarrow \quad \text{cons} \quad \langle \text{head, tail} \rangle \\
A \times Z \xrightarrow{\text{id} \times \text{orb}} A \times A^\omega
\end{align*}
$$

$$
\begin{align*}
\text{head}(\text{orb}(z)) &= a(z) \\
\text{tail}(\text{orb}(z)) &= \text{orb}(f(z))
\end{align*}
$$

$A^\omega \simeq A \times A^\omega$
A stream over an alphabet A is made of a head $a \in A$ and a tail, which is itself a stream. $A^\omega$ is the greatest fixpoint of this definition!

A final coalgebra

$$A \times Z \xrightarrow{id \times orb} A \times A^\omega \xrightarrow{\text{cons}} \langle \text{head}, \text{tail} \rangle \xrightarrow{\exists! \text{orb}} A^\omega$$

$$\text{head}(\text{orb}(z)) = a(z)$$
$$\text{tail}(\text{orb}(z)) = \text{orb}(f(z))$$

$$A^\omega =_{\nu} A \times A^\omega$$
Another coinductive type...

\[ \nu X. (A \times X \times X) = \text{Infinite labeled binary trees} \]

Diagram:

- \( \nu X. (A \times X \times X) \)
- Infinite labeled binary trees

\[ \langle a, T_1, T_2 \rangle \]

Vertices:
- \( a \)
- \( T_1 \)
- \( T_2 \)
Mixted types
Example: Streams of natural numbers:

\[ S = \left\{ \begin{array}{c} S = \nu N \times S \\ N = \mu 1 + N \end{array} \right\} \]
Mixted types

Example: Streams of natural numbers:

\[ S = \left\{ \begin{array}{l}
S = \nu N \times S \\
N = \mu 1 + N
\end{array} \right\} \]

Definition

\( S \) is a directed system of equations (the order matters).
Mixted types

Example: Streams of natural numbers:

\[ S = \begin{cases} \ S =_2 N \times S \\ N =_1 1 + N \end{cases} \]

Definition

\( S \) is a \textit{directed system of equations} (\textit{the order matters}).

\[
\text{Priority}(X) \text{ is } \begin{cases} \text{even} , & \text{if } X =_\nu \ldots \\ \text{odd} , & \text{if } X =_\mu \ldots \end{cases}
\]
\( \mu \)-bicomplete categories

**Definition**

A category \( C \) is \( \mu \)-bicomplete if each directed system of equations can be solved within \( C \).
Definition

A category $\mathcal{C}$ is $\mu$-bicomplete if each directed system of equations can be solved within $\mathcal{C}$.

Examples:

- Any complete lattice (Tarski’s theorem).
**μ-bicomplete categories**

**Definition**

A category $C$ is **μ-bicomplete** if each directed system of equations can be solved within $C$.

**Examples:**

- Any complete lattice (Tarski’s theorem).
- $M(B)$: The free $μ$-bicocomplete category over a set $B$ of generators.
Definition

A category $\mathcal{C}$ is $\mu$-bicomp \textit{complete} if each directed system of equations can be solved within $\mathcal{C}$.

Examples:

- Any complete lattice (Tarski’s theorem).
- $\mathcal{M}(B)$: The free $\mu$-bicomp \textit{complete} category over a set $B$ of generators.
- $\text{Set}$: The category of sets.
A category $\mathcal{C}$ is $\mu$-bicomplete if each directed system of equations can be solved within $\mathcal{C}$.

Examples:

- Any complete lattice (Tarski’s theorem).
- $\mathcal{M}(B)$: The free $\mu$-bicompcomplete category over a set $B$ of generators.
- $\text{Set}$: The category of sets.

Directed systems of equations therefore provide a combinatorial description of the objects of a $\mu$-bicompcomplete category (via parity games, see [Santocanale, 2001]).
μ-bicomplete categories

Definition

A category $C$ is μ-bicomplete if each directed system of equations can be solved within $C$.

Examples:

- Any complete lattice (Tarski’s theorem).
- $\mathcal{M}(B)$: The free μ-bicomplete category over a set $B$ of generators.
- $Set$: The category of sets.

Directed systems of equations therefore provide a combinatorial description of the objects of a μ-bicomplete category (via parity games, see [Santocanale, 2001]).

What about morphisms?
Definition

A category $\mathcal{C}$ is $\mu$-bicomplete if each directed system of equations can be solved within $\mathcal{C}$.

Examples:

- Any complete lattice (Tarski's theorem).
- $\mathcal{M}(B)$: The free $\mu$-bicomplete category over a set $B$ of generators.
- $\text{Set}$: The category of sets.

Directed systems of equations therefore provide a combinatorial description of the objects of a $\mu$-bicomplete category (via parity games, see [Santocanale, 2001]).

What about morphisms? $\leadsto$ Circular proofs!
<table>
<thead>
<tr>
<th>Preliminaries</th>
<th>Fixpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circular proofs</td>
<td>$\mu$-bicomplete categories</td>
</tr>
<tr>
<td>Computing</td>
<td></td>
</tr>
</tbody>
</table>

Curry–Howard correspondence
Curry–Howard correspondence

\[ \lambda \text{-calculus} \leftrightarrow \text{Intuitionistic Logic} \]

Cartesian Closed Categories
Curry–Howard correspondence

Directed Systems ↔ Intuitionistic Logic

Cartesian Closed Categories
Curry–Howard correspondence

Directed Systems $\xrightarrow{\mu}$ Intuitionistic Logic

$\mu$-bicomplete Categories
Curry–Howard correspondence

Directed Systems $\leftrightarrow$ Circular Proofs

$\mu$-bicomplete Categories
Rules of Circular Proofs (à la Gentzen)

Given a directed system $S$:

**Axioms:**
- $\frac{0 \vdash A}{\text{LAx}}$
- $\frac{A \vdash 1}{\text{RAx}}$
- $\frac{A \vdash A}{\text{Id}}$

**Products:**
(conjunction)
- $\frac{A_i \vdash B}{A_0 \times A_1 \vdash B}$ \hspace{1cm} \text{L$x_i$}
- $\frac{A \vdash B_0 \quad A \vdash B_1}{A \vdash B_0 \times B_1}$ \hspace{1cm} \text{R$x$}

**Coproducts:**
(disjunction)
- $\frac{A_0 \vdash B \quad A_1 \vdash B}{A_0 + A_1 \vdash B}$ \hspace{1cm} \text{L$+$}
- $\frac{A \vdash B_i}{A \vdash B_0 + B_1}$ \hspace{1cm} \text{R$+$}

**Fixpoints:**
- $\frac{F(X) \vdash B}{X \vdash B}$ \hspace{1cm} \text{LF$X$}
- $\frac{A \vdash F(X)}{A \vdash X}$ \hspace{1cm} \text{RF$X$}

- “$X = p \ F(X)$” $\in S$

**Cut:**
- $\frac{A \vdash C \quad C \vdash B}{A \vdash B}$

**Assumption:**
- $\frac{A \vdash B}{A \vdash B}$
Categorical interpretation

Axioms:

\[ \begin{array}{lll}
\text{LAx} & 0 \xrightarrow{?_t} A \\
\text{RAx} & A \xrightarrow{!_t} 1 \\
\text{Id} & A \xrightarrow{id_A} A
\end{array} \]

Products:
(conjunction)

\[ \begin{array}{c}
A_i \xrightarrow{f} B \\
A_0 \times A_1 \xrightarrow{\text{pr}, f} B \\
A \xrightarrow{\{f, g\}} B_0 \times B_1
\end{array} \]

Coproducts:
(disjunction)

\[ \begin{array}{c}
A_0 \xrightarrow{f} B \\
A_1 \xrightarrow{g} B \\
A_0 + A_1 \xrightarrow{\{f, g\}} B \\
A \xrightarrow{f \cdot \text{in}_i} B_0 + B_1
\end{array} \]

Fixpoints:

\[ \begin{array}{c}
F(X) \xrightarrow{f} B \\
X \xrightarrow{\alpha_X^{-1} \cdot f} B \\
F(X) \xrightarrow{f} B \\
X \xrightarrow{\zeta_X \cdot f} B
\end{array} \]

Cut:

\[ \begin{array}{c}
A \xrightarrow{f} C, C \xrightarrow{g} B \\
A \xrightarrow{f \cdot g} B
\end{array} \]

Jérôme Fortier  
FMCS 2015, Colgate University
Example

\[ S = \{ N = \mu 1 + N \} \]
Example

\[ S = \{ N = \mu 1 + N \} \]

Solution:

\[ 1 + N \xrightarrow{\{0, \text{suc}\}} N \]
Example

\[ S = \{ N = \mu 1 + N \} \]

Solution:

\[ 1 + N \xrightarrow{\{0,\text{suc}\}} N \]

Let

\[ \text{double}(0) = 0 \]
\[ \text{double}(\text{suc}(n)) = \text{suc}(\text{suc}(\text{double}(n))) \]
Example

\[ S = \{ N = \mu 1 + N \} \]

Solution:

\[ 1 + N \xrightarrow{\{0, suc\}} N \]

Let

\[ \text{double}(0) = 0 \]
\[ \text{double}(\text{suc}(n)) = \text{suc}(\text{suc}(\text{double}(n))) \]
Example

\[ S = \{ N =_{\mu} 1 + N \} \]

Solution:

\[ 1 + N \xrightarrow{\{0,\text{suc}\}} N \]

Let

\[
\begin{align*}
\text{double}(0) &= 0 \\
\text{double}(\text{suc}(n)) &= \text{suc}(\text{suc}(\text{double}(n)))
\end{align*}
\]
Example

\[ S = \{ N =_\mu 1 + N \} \]

Solution:

\[ 1 + N \xrightarrow{\{0,\text{suc}\}} N \]

Let

\[
\begin{align*}
1 & \vdash N \\
N & \vdash N \\
\hline
1 + N & \vdash N \\
\hline
\end{align*}
\]

\[ \text{Let} \]

\[
\begin{align*}
double(0) & = 0 \\
double(\text{suc}(n)) & = \text{suc}(\text{suc}(\text{double}(n)))
\end{align*}
\]
Example

\[ S = \{ N =_\mu 1 + N \} \]

Solution:

\[
\begin{align*}
1 + N & \xrightarrow{\{0,\text{suc}\}} N \\
1 \vdash 1 + N & \quad \text{RF}_N \\
1 \vdash N & \quad \text{LF}_N \\
N \vdash N & \quad \text{L} +
\end{align*}
\]

Let

\[
\begin{align*}
\text{double}(0) & = 0 \\
\text{double}(\text{suc}(n)) & = \text{suc}(\text{suc}(\text{double}(n)))
\end{align*}
\]
Example

\[ S = \{ N = \mu 1 + N \} \]

Solution:

\[ 1 + N \xrightarrow{\{0,\text{suc}\}} N \]

Let

\[ \text{double}(0) = 0 \]
\[ \text{double}(\text{suc}(n)) = \text{suc}(\text{suc}(\text{double}(n))) \]
Example

\[ S = \{ N = \mu 1 + N \} \]

Solution:

\[ 1 + N \xrightarrow{\{0,\text{suc}\}} N \]

Let

\[ \text{double}(0) = 0 \]
\[ \text{double}(\text{suc}(n)) = \text{suc}(\text{suc}(\text{double}(n))) \]
Example

\[ S = \{ N =_\mu 1 + N \} \]

Solution:

\[ 1 + N \xrightarrow{\{0,\text{suc}\}} N \]

Let

\[
\begin{align*}
\text{double}(0) &= 0 \\
\text{double}(\text{suc}(n)) &= \text{suc}(%(\text{suc}(\text{double}(n)))
\end{align*}
\]
Example

\[ S = \{ N =_{\mu} 1 + N \} \]

Solution:

\[
1 + N \xrightarrow{\{0, \text{suc}\}} N
\]

Let

\[
\text{double}(0) = 0
\]

\[
\text{double}(\text{suc}(n)) = \text{suc}(\text{suc}(\text{double}(n)))
\]
Example

\[ S = \{ N = \mu 1 + N \} \]

Solution:

\[ 1 + N \xrightarrow{\{0, \text{suc}\}} N \]

Let

\[ \text{double}(0) = 0 \]
\[ \text{double}(\text{suc}(n)) = \text{suc}(\text{suc}(\text{double}(n))) \]
Example

\[ S = \{ N =_{\mu} 1 + N \} \]

Solution:

\[ 1 + N \xrightarrow{\{0, \text{suc}\}} N \]

Let

\begin{align*}
\text{double}(0) &= 0 \\
\text{double}(\text{suc}(n)) &= \text{suc}(\text{suc}(\text{double}(n)))
\end{align*}
Example

\[ S = \{ N = \mu 1 + N \} \]

Solution:

\[ 1 + N \xrightarrow{\{0,\text{suc}\}} N \]

Let

\[
\begin{align*}
\text{double}(0) &= 0 \\
\text{double}(\text{suc}(n)) &= \text{suc}(\text{suc}(\text{double}(n)))
\end{align*}
\]
Proofs $\Rightarrow$ Systems of equations

$\Pi :$

1. $1 \rightarrow 1$ (RAx)
2. $1 \rightarrow N$ (RF$_N$
3. $1 + N \rightarrow N$ (RF$_N$
4. $N \rightarrow N$ (R+1)
5. $N \rightarrow 1 + N$ (R+1)
6. $N \rightarrow N$ (R+1)
7. $N \rightarrow 1 + N$ (R+1)
8. $N \rightarrow N$ (R+1)

Unique solution: $f_0 = \text{double}$ J$\_\Pi$ $K := \langle f_0, f_1, ..., f_8 \rangle$. 

Jérôme Fortier  
FMCS 2015, Colgate University
Proofs $\Rightarrow$ Systems of equations

$\Pi :$

\[
\begin{align*}
N &\rightarrow N \\
\begin{array}{c}
R+1 \\
N \rightarrow 1+N \\
N \rightarrow 1+N \\
N \rightarrow 1+N \\
N \rightarrow 1+N \\
N \rightarrow 1+N \\
N \rightarrow 1+N \\
N \rightarrow 1+N
\end{array} \\
\begin{array}{c}
R+0 \\
1 \xrightarrow{f_4} 1 \\
1 \xrightarrow{f_3} 1+N \\
1 \xrightarrow{f_2} N \\
1 \xrightarrow{f_1} N \\
1 \xrightarrow{f_0} N
\end{array} \\
1 \xrightarrow{f_8} 1+N \\
\begin{array}{c}
R+1 \\
N \rightarrow 1+N \\
N \rightarrow 1+N \\
N \rightarrow 1+N \\
N \rightarrow 1+N \\
N \rightarrow 1+N \\
N \rightarrow 1+N \\
N \rightarrow 1+N
\end{array}
\end{align*}
\]

$\mapsto \begin{bmatrix} ?\Pi \end{bmatrix} =$

\[
\begin{cases}
    f_8 = f_0 \cdot \text{in}_1 \\
    f_7 = f_8 \cdot \alpha_N \\
    f_6 = f_7 \cdot \text{in}_1 \\
    f_5 = f_6 \cdot \alpha_N \\
    f_4 = !_1 \\
    f_3 = f_4 \cdot \text{in}_0 \\
    f_2 = f_3 \cdot \alpha_N \\
    f_1 = \{f_2, f_5\} \\
    f_0 = \alpha_N^{-1} \cdot f_1
\end{cases}
\]
Proofs ⇒ Systems of equations

\[ \Pi : \]

\[ \begin{align*}
N \xrightarrow{f_0} N \\
\quad \quad \xrightarrow{R+1} \\
1 \xrightarrow{f_4} 1 \\
\quad \quad \xrightarrow{R+0} \\
1 \xrightarrow{f_3} 1 + N \\
\quad \quad \xrightarrow{RF_N} \\
1 \xrightarrow{f_2} N \\
\quad \quad \xrightarrow{L+} \\
1 + N \xrightarrow{f_1} N \\
\quad \quad \xrightarrow{LF_N}
\end{align*} \]

\[ \mapsto \left\{ \begin{array}{l}
f_8 = f_0 \cdot \text{in}_1 \\
f_7 = f_8 \cdot \alpha_N \\
f_6 = f_7 \cdot \text{in}_1 \\
f_5 = f_6 \cdot \alpha_N \\
f_4 = !_1 \\
f_3 = f_4 \cdot \text{in}_0 \\
f_2 = f_3 \cdot \alpha_N \\
f_1 = \{f_2, f_5\} \\
f_0 = \alpha_N^{-1} \cdot f_1
\end{array} \right\} \]

Unique solution: \( f_0 = \text{double} \)
Proofs $\Rightarrow$ Systems of equations

**Π:**

- $N \xrightarrow{f_0} N$  \[\text{R+1} \]
- $N \xrightarrow{f_8} 1 + N$  \[\text{RF}_N \]
- $N \xrightarrow{f_7} N$  \[\text{R+1} \]
- $N \xrightarrow{f_6} 1 + N$  \[\text{RF}_N \]
- $N \xrightarrow{f_5} N$  \[\text{L+} \]
- $1 + N \xrightarrow{f_1} N$  \[\text{LF}_N \]
- $1\xrightarrow{f_4} 1$  \[\text{RAx} \]
- $1\xrightarrow{f_3} 1 + N$  \[\text{RF}_N \]
- $1\xrightarrow{f_2} N$  \[\text{} \]

$[?\Pi] = \{ f_8 = f_0 \cdot \text{in}_1, f_7 = f_8 \cdot \alpha_N, f_6 = f_7 \cdot \text{in}_1, f_5 = f_6 \cdot \alpha_N, f_4 = 1_1, f_3 = f_4 \cdot \text{in}_0, f_2 = f_3 \cdot \alpha_N, f_1 = \{f_2, f_5\}, f_0 = \alpha_N^{-1} \cdot f_1 \}$

**Unique solution:** $f_0 = \text{double}

$[!\Pi] := \langle f_0, f_1 \ldots f_8 \rangle.$
Co-example

\[
S \ =_\nu 2 \times S \\
2 \ =_\nu 1 + 1
\]
Co-example

\[ S = _\nu 2 \times S \]
\[ 2 = _\nu 1 + 1 \]

Solution:

\[ S = 2^\omega \]
\[ 2 = \{0, 1\} \]
Co-example

\[ S = \nu 2 \times S \]
\[ 2 = \nu 1 + 1 \]

Solution:

\[ S = 2^\omega \]
\[ 2 = \{0, 1\} \]

Let

\[ \text{alt} = (0, 1, 0, 1, 0, 1, \ldots) \in S \]
Co-example

\[ S =_\nu 2 \times S \]
\[ 2 =_\nu 1 + 1 \]

Solution:

\[ S = 2^\omega \]
\[ 2 = \{0, 1\} \]

Let

\[ 1 \vdash S \]
\[ \text{alt} = (0, 1, 0, 1, 0, 1, \ldots) \in S \]
Co-example

\[ S = \nu (2 \times S) \]
\[ 2 = \nu (1 + 1) \]

Solution:

\[ S = 2^\omega \]
\[ 2 = \{0, 1\} \]

Let

\[ \text{alt} = (0, 1, 0, 1, 0, 1, \ldots) \in S \]
Co-example

\[ \begin{align*}
S &= _\nu 2 \times S \\
2 &= _\nu 1 + 1
\end{align*} \]

Solution:

\[ \begin{align*}
S &= 2^\omega \\
2 &= \{0, 1\}
\end{align*} \]

Let

\[ \text{alt} = (0, 1, 0, 1, 0, 1, \ldots) \in S \]
Co-example

\[ S = \nu \ 2 \times S \]
\[ 2 = \nu \ 1 + 1 \]

Solution:

\[ S = 2^\omega \]
\[ 2 = \{0, 1\} \]

Let

\[ \text{alt} = (0, 1, 0, 1, 0, 1, \ldots) \in S \]
Co-example

\[ S = \nu 2 \times S \]
\[ 2 = \nu 1 + 1 \]

Solution:

\[ S = 2^\omega \]
\[ 2 = \{0, 1\} \]

Let

\[ \text{alt} = (0, 1, 0, 1, 0, 1, \ldots) \in S \]
Co-example

\[ S = \nu \ 2 \times S \]
\[ 2 = \nu \ 1 + 1 \]

Solution:

\[ S = 2^\omega \]
\[ 2 = \{0, 1\} \]

Let

\[ \text{alt} = (0, 1, 0, 1, 0, 1, \ldots) \in S \]
Non valid pre-proofs

1 ⊢ 1
\[ \frac{1 ⊢ 1}{1 ⊢ 1 + 1} \text{ RAx} \]
\[ \frac{1 ⊢ 1 + 1}{1 ⊢ 2} \text{ RF}_S \]
\[ \frac{1 ⊢ 2}{1 ⊢ 2 \times S} \text{ R}_\times \]
\[ \frac{1 ⊢ 2 \times S}{1 ⊢ S} \text{ RF}_S \]

1 ⊢ S
\[ \frac{1 ⊢ S}{1 ⊢ S} \text{ Id} \]
\[ \frac{S ⊢ S}{2 \times S ⊢ S} \text{ L}_\times \]
\[ \frac{S ⊢ S}{S ⊢ S} \text{ L}_F_S \]
\[ \frac{S ⊢ S}{1 ⊢ 1 + 1} \text{ C} \]

S =_\nu 2 \times S
2 =_\nu 1 + 1
Non valid pre-proofs

\[
\begin{align*}
R \text{Ax} & : 1 \vdash 1 \\
R + 0 & : 1 \vdash 1 + 1 \\
R F_S & : 1 \vdash S \\
L \times 1 & : 2 \times S \vdash S \\
L F_S & : S \vdash S \\
C & : 1 \vdash S \\
R \times & : 1 \vdash 2 \times S \\
R F_S & : 1 \vdash S \\
\end{align*}
\]

\[
\begin{align*}
\text{Id} & : S \vdash S \\
2 \times S \vdash S & : S \vdash S \\
\end{align*}
\]

\[
\begin{align*}
S \ =_\nu & \ 2 \times S \\
2 \ =_\nu & \ 1 + 1 \\
\text{head}(f) & = 0 \\
\text{tail}(f) & = \text{tail}(f) \
\end{align*}
\]
Non valid pre-proofs

\[ \frac{1 \vdash 1}{R\text{Ax}} \quad \frac{1 \vdash 1 + 1}{R + 0} \quad \frac{1 \vdash 2}{\text{RF}_S} \quad \frac{1 \vdash S}{1 \vdash S} \quad \frac{2 \times S \vdash S}{\text{L}\times_1} \quad \frac{S \vdash S}{\text{L}\text{F}_S} \quad \frac{S \vdash S}{\text{C}} \]

\[ \frac{1 \vdash 2 \times S}{\text{R}\times} \quad \frac{1 \vdash S}{1 \vdash S} \quad \frac{1 \vdash S}{\text{RF}_S} \]

Infinitely many solutions!

\[ S =_{\nu} 2 \times S \]
\[ 2 =_{\nu} 1 + 1 \]

head(f) = 0
\[ \text{tail}(f) = \text{tail}(f) \ldots \]
Non valid pre-proofs

God exists

Bible says it is the word of God

Fact

What Bible says is true

Bible is the word of God

What Bible says is true

God exists

Bible says that God exists

Fact

What Bible says is true

Bible is the word of God

God exists

Fact
We must impose some condition on the **cycles** in order to make sure they are meaningful.
We must impose some condition on the cycles in order to make sure they are meaningful.

Guard condition (informally)
Each turn in a cycle must either read a part of an inductive input (on the left) or write a part of a coinductive output (on the right).
We must impose some condition on the cycles in order to make sure they are meaningful.

**Guard condition (informally)**

Each turn in a cycle must either read a part of an inductive input (on the left) or write a part of a coinductive output (on the right).

From an analogy with parity games, we ask that any cycle:

- contains a left fixpoint rule, and the highest priority is odd,
- OR contains a right fixpoint rule, and the highest priority is even.
Guard condition

\[
\begin{array}{c}
A \vdash C \\
\hline
\end{array}
\]

\[
\begin{array}{c}
C \vdash B \\
\hline
\end{array}
\]

\[
\begin{array}{c}
A \vdash B \\
\hline
\end{array}
\]

Left

\[
\begin{array}{c}
A \vdash C \\
\hline
\end{array}
\]

\[
\begin{array}{c}
C \vdash B \\
\hline
\end{array}
\]

\[
\begin{array}{c}
A \vdash B \\
\hline
\end{array}
\]

Right

Acyclic

\[
\begin{array}{c}
A \vdash C \\
\hline
\end{array}
\]

\[
\begin{array}{c}
C \vdash B \\
\hline
\end{array}
\]

\[
\begin{array}{c}
A \vdash B \\
\hline
\end{array}
\]

Ambidextrous

Remark

In a cycle, a left cut kills the right trace and a right cut kills the left trace!

Jérôme Fortier  
FMCS 2015, Colgate University
Guard condition

In a cycle, a left cut kills the right trace and a right cut kills the left trace!
Guard condition

Definition (left $\mu$-trace)

A subset $E \subseteq \Pi$ has a left $\mu$-trace if it

- contains a left fixpoint rule, and the highest priority is odd;
- contains only left cuts.
Definition (left $\mu$-trace)

A subset $E \subseteq \Pi$ has a left $\mu$-trace if it

- contains a left fixpoint rule, and the highest priority is odd;
- contains only left cuts.

Definition (right $\nu$-trace)

A subset $E \subseteq \Pi$ has a right $\nu$-trace if it

- contains a right fixpoint rule, and the highest priority is even;
- contains only right cuts.
Guard condition (formally)

The following are equivalent.

1. Every cycle in $\Pi$ either has a left $\mu$-trace or a right $\nu$-trace.
2. Every infinite path $\Gamma$ in $\Pi$ has a tail $\Gamma'$ that has either a left $\mu$-trace or a right $\nu$-trace and every fixpoint rule in $\Gamma'$ occurs infinitely often.
3. Every nontrivial strongly connected component of $\Pi$ either has a left $\mu$-trace or a right $\nu$-trace.

Definition

A circular proof is a finite pre-proof that satisfies the guard conditions.
Guard condition

Guard condition (formally)

The following are equivalent.

1. Every cycle in $\Pi$ either has a left $\mu$-trace or a right $\nu$-trace.
Guard condition

Guard condition (formally)

The following are equivalent.

1. Every cycle in $\Pi$ either has a left $\mu$-trace or a right $\nu$-trace.
2. Every infinite path $\Gamma$ in $\Pi$ has a tail $\Gamma'$ that has either a left $\mu$-trace or a right $\nu$-trace and every fixpoint rule in $\Gamma'$ occurs infinitely often.
Guard condition

### Guard condition (formally)

The following are equivalent.

1. Every cycle in $\Pi$ either has a left $\mu$-trace or a right $\nu$-trace.

2. Every infinite path $\Gamma$ in $\Pi$ has a tail $\Gamma'$ that has either a left $\mu$-trace or a right $\nu$-trace and every fixpoint rule in $\Gamma'$ occurs infinitely often.

3. Every nontrivial strongly connected component of $\Pi$ either has a left $\mu$-trace or a right $\nu$-trace.
Guard condition

Guard condition (formally)
The following are equivalent.

1. Every cycle in \( \Pi \) either has a left \( \mu \)-trace or a right \( \nu \)-trace.
2. Every infinite path \( \Gamma \) in \( \Pi \) has a tail \( \Gamma' \) that has either a left \( \mu \)-trace or a right \( \nu \)-trace and every fixpoint rule in \( \Gamma' \) occurs infinitely often.
3. Every nontrivial strongly connected component of \( \Pi \) either has a left \( \mu \)-trace or a right \( \nu \)-trace.

Definition

A circular proof is a finite pre-proof that satisfies the guard conditions.
Soundness Theorem (F.–Santocanale, 2013)

Each circular proof denotes an unique arrow of the free \( \mu \)-bicomplete category \( \mathcal{M} \) (hence a canonical arrow of every \( \mu \)-bicomplete category \( \mathcal{C} \)).
Denotational semantics

Soundness Theorem (F.–Santocanale, 2013)

Each circular proof denotes an unique arrow of the free $\mu$-bicomplete category $\mathcal{M}$ (hence a canonical arrow of every $\mu$-bicomplete category $\mathcal{C}$).

Proof sketch:

1. For every strongly connected component $K$ of $\Pi$, we solve the system $[\Psi K]$. We can then glue the pieces together since they are well-ordered.
Denotational semantics

Soundness Theorem (F.–Santocanale, 2013)

Each circular proof denotes an unique arrow of the free $\mu$-bicomplete category $\mathcal{M}$ (hence a canonical arrow of every $\mu$-bicomplete category $\mathcal{C}$).

Proof sketch:

1. For every strongly connected component $K$ of $\Pi$, we solve the system $[?K]$. We can then glue the pieces together since they are well-ordered.
2. WLOG, $K$ has a left $\mu$-trace.
Soundness Theorem (F.–Santocanale, 2013)

Each circular proof denotes an unique arrow of the free $\mu$-bicomplete category $\mathcal{M}$ (hence a canonical arrow of every $\mu$-bicomplete category $\mathcal{C}$).

Proof sketch:

1. For every strongly connected component $K$ of $\Pi$, we solve the system $[?K]$. We can then glue the pieces together since they are well-ordered.

2. WLOG, $K$ has a left $\mu$-trace. Define $K'$ by changing all the maximal left fixpoint rules into assumptions. By induction on the directed system, $K'$ is solvable.
Soundness Theorem (F.–Santocanale, 2013)

Each circular proof denotes an unique arrow of the free $\mu$-bicomplete category $\mathcal{M}$ (hence a canonical arrow of every $\mu$-bicomplete category $\mathcal{C}$).

Proof sketch:

1. For every strongly connected component $K$ of $\Pi$, we solve the system $[?K]$. We can then glue the pieces together since they are well-ordered.

2. WLOG, $K$ has a left $\mu$-trace. Define $K'$ by changing all the maximal left fixpoint rules into assumptions. By induction on the directed system, $K'$ is solvable.

3. Use Santocanale's Fixpoint Lemma to extend that solution to $K$. □
Denotational semantics

Fullness Theorem (F.–Santocanale, 2013)

Every arrow $f : A \rightarrow B$ of $\mathcal{M}$ is the solution to some circular proof.
Denotational semantics

Fullness Theorem (F.–Santocanale, 2013)

Every arrow $f : A \rightarrow B$ of $\mathcal{M}$ is the solution to some circular proof.

Proof sketch:
We must construct a circular proof for each arrow of each diagram, namely: product, coproduct, initial algebra and final coalgebra.
Fullness Theorem (F.–Santocanale, 2013)

Every arrow $f : A \rightarrow B$ of $\mathcal{M}$ is the solution to some circular proof.

Proof sketch:
We must construct a circular proof for each arrow of each diagram, namely: product, coproduct, initial algebra and final coalgebra. Trivial in most cases.
Denotational semantics

**Fullness Theorem (F.–Santocanale, 2013)**

Every arrow $f : A \to B$ of $\mathcal{M}$ is the solution to some circular proof.

Proof sketch:
We must construct a circular proof for each arrow of each diagram, namely: **product, coproduct, initial algebra** and **final coalgebra**. Trivial in most cases. Except those!

\[
\begin{array}{c}
F(A) \xrightarrow{F(f)} F(X) \\
A \xrightarrow{f} X
\end{array}
\]

\[\alpha_A \quad x = [\Pi]\]
Denotational semantics

Fullness Theorem (F.–Santocanale, 2013)

Every arrow \( f : A \rightarrow B \) of \( \mathcal{M} \) is the solution to some circular proof.

Proof sketch:
We must construct a circular proof for each arrow of each diagram, namely: product, coproduct, initial algebra and final coalgebra. Trivial in most cases. Except those!
Fullness Theorem (F.–Santocanale, 2013)

Every arrow $f : A \rightarrow B$ of $\mathcal{M}$ is the solution to some circular proof.

Proof sketch:
We must construct a circular proof for each arrow of each diagram, namely: product, coproduct, initial algebra and final coalgebra. Trivial in most cases. Except those!

\[
\begin{align*}
F(X) &\xrightarrow{F(f)} F(Z) \\
&\xrightarrow{\zeta_Z} Z
\end{align*}
\]

\[
\begin{align*}
x = \Pi \\
\text{RF}_Z \\
\text{Copycat}
\end{align*}
\]
Theorem (Santocanale, 2001)

There is no cut-free circular proof whose interpretation in $\textbf{Set}$ is the diagonal $\Delta : \mathbb{N} \rightarrow \mathbb{N}^2$. 
Theorem (Santocanale, 2001)

There is no cut-free circular proof whose interpretation in $\text{Set}$ is the diagonal $\Delta : N \to N^2$. 
Theorem (Santocanale, 2001)

There is no cut-free circular proof whose interpretation in \( \text{Set} \) is the diagonal
\( \Delta : \mathbb{N} \rightarrow \mathbb{N}^2 \).
Diagonal map (with cuts)

\[ \Delta : \mathbb{N} \rightarrow \mathbb{N}^2 \]

\[ n \mapsto (n, n) \]

\[ N = \mu_1 + N \]

\[ M = \mu_N + M \]
Cut elimination

"ET SI ON CONSIDÈRAIT...

DES PREUVES INFINIES!"
Cut elimination

“Push ”all the cuts away from the rot in order to evacuate them at the limit.”
Pushing cuts away

If we have a left rule on the left:
Pushing cuts away

If we have a **left** rule on the **left**:

\[
\begin{array}{c}
0 \vdash C \\
\hline
\text{L} A x \\
\hline
C \vdash B \\
\hline
0 \vdash B \\
\end{array}
\]
Pushing cuts away

If we have a left rule on the left:

\[
\frac{0 \vdash C}{0 \vdash B} \quad \frac{C \vdash B}{0 \vdash B}
\]

⇒

\[
\frac{0 \vdash C}{0 \vdash B} \quad \frac{C \vdash B}{0 \vdash B}
\]
Pushing cuts away

If we have a left rule on the left:

\[
\begin{align*}
& \frac{0 \vdash C \quad C \vdash B}{0 \vdash B} \quad \text{LAx} \\
& \frac{F(X) \vdash C \quad X \vdash C \quad C \vdash B}{X \vdash B} \quad \text{LFx}
\end{align*}
\]

\[
\begin{align*}
& \frac{F(X) \vdash C \quad X \vdash C \quad C \vdash B}{X \vdash B} \quad \text{LFx}
\end{align*}
\]

\[
\begin{align*}
& \frac{F(X) \vdash C \quad X \vdash C \quad C \vdash B}{X \vdash B} \quad \text{LFx}
\end{align*}
\]
If we have a **left** rule on the **left**:

\[
\frac{0 \vdash C \quad C \vdash B}{0 \vdash B} \quad \text{LAx}
\]

\[
\frac{F(X) \vdash C \quad X \vdash C \quad C \vdash B}{X \vdash B} \quad \text{LFx}
\]

\[
\frac{F(X) \vdash B \quad X \vdash B}{X \vdash B} \quad \text{LFx}
\]
If we have a **left** rule on the **left**:

\[\begin{array}{c}
0 \vdash C \\
\hline
\hline
\hline
\hline
C \vdash B \\
\hline
0 \vdash B
\end{array}\]  \quad \Rightarrow \quad \begin{array}{c}
0 \vdash B
\end{array}

\[\begin{array}{c}
F(X) \vdash C \\
\hline
\hline
\hline
\hline
X \vdash C \\
\hline
\hline
\hline
\hline
C \vdash B \\
\hline
X \vdash B
\end{array}\]  \quad \Rightarrow \quad \begin{array}{c}
F(X) \vdash C \\
\hline
\hline
\hline
\hline
C \vdash B \\
\hline
X \vdash B
\end{array}

\[\begin{array}{c}
A_j \vdash C \\
\hline
\hline
\hline
\hline
A_0 \times A_1 \vdash C \\
\hline
\hline
\hline
\hline
C \vdash B \\
\hline
A_0 \times A_1 \vdash B
\end{array}\]  \quad \Rightarrow \quad \begin{array}{c}
A_0 \times A_1 \vdash B \\
\hline
\hline
\hline
\hline
A_0 \times A_1 \vdash C \\
\hline
\hline
\hline
\hline
C \vdash B \\
\hline
A_0 \times A_1 \vdash B
\end{array}\]
If we have a \textbf{left} rule on the \textbf{left}:

\[
\frac{0 \vdash C}{0 \vdash B} \quad \frac{C \vdash B}{C} \quad \Rightarrow \quad \frac{F(X) \vdash C}{X \vdash B} \quad \frac{X \vdash C}{C} \quad \Rightarrow \quad \frac{F(X) \vdash C}{F(X) \vdash B} \quad \frac{C \vdash B}{C} \quad \Rightarrow \quad \frac{A_i \vdash C}{A_i \vdash B} \quad \frac{C \vdash B}{C} \quad \Rightarrow \quad \frac{A_i \vdash C}{A_i \vdash B} \quad \frac{C \vdash B}{C}
\]
Pushing cuts away

If we have a **left** rule on the **left**:

\[
\frac{0 \vdash C}{\vdash B} \quad \frac{C \vdash B}{\vdash C} \quad \frac{F(X) \vdash C}{\vdash B} \quad \frac{X \vdash B}{\vdash C} \quad \frac{A_i \vdash C}{\vdash B} \quad \frac{A_0 \times A_1 \vdash C}{\vdash B} \quad \frac{A_0 + A_1 \vdash C}{\vdash B}
\]
If we have a **left** rule on the left:

1. \( \frac{0 \vdash C}{0 \vdash B} \)
   \( \frac{C \vdash B}{C} \)
   \( \frac{LAx}{0 \vdash C} \)
   \( C \vdash B \)
   \( C \)

2. \( \frac{F(X) \vdash C}{X \vdash C} \)
   \( \frac{LFx}{C \vdash B} \)
   \( X \vdash B \)
   \( C \)

3. \( \frac{A_i \vdash C}{A_0 \times A_1 \vdash C} \)
   \( \frac{L \times i}{C \vdash B} \)
   \( A_0 \times A_1 \vdash B \)
   \( C \)

4. \( \frac{A_0 \vdash C \quad A_1 \vdash C}{A_0 + A_1 \vdash C} \)
   \( \frac{L+}{C \vdash B} \)
   \( A_0 + A_1 \vdash B \)
   \( C \)
If we have a right rule on the right:

\[
\frac{A \vdash C}{A \vdash 1}
\]

\[
\frac{C \vdash 1}{A \vdash 1}
\]

\[
\frac{A \vdash C}{\frac{C \vdash 1}{A \vdash 1}}
\]

\[
\frac{C \vdash F(X)}{C \vdash X}
\]

\[
\frac{A \vdash X}{\frac{C \vdash F(X)}{A \vdash X}}
\]

\[
\frac{C \vdash B_i}{\frac{A \vdash B_0 + B_1}{A \vdash B_0 + B_1}}
\]

\[
\frac{A \vdash B_i}{\frac{C \vdash B_0 + B_1}{A \vdash B_0 + B_1}}
\]

\[
\frac{C \vdash B_0 \times B_1}{A \vdash B_0 \times B_1}
\]

\[
\frac{A \vdash B_0 \times B_1}{\frac{C \vdash B_0 \times B_1}{A \vdash B_0 \times B_1}}
\]

Jérôme Fortier  
FMCS 2015, Colgate University
Cut VS Cut

\[
\frac{A \vdash B \quad B \vdash C}{A \vdash C} \quad \frac{C \vdash D}{A \vdash D}
\]
Definition
A multicut is a finite nonempty list \( M := [u_1, u_2, \ldots, u_m] \) of composable vertices of \( \Pi \).
**Definition**

A *multicut* is a finite nonempty list $M := [u_1, u_2 \ldots u_m]$ of composable vertices of $\Pi$. 
Elimination of identities

\[
\begin{align*}
A_0 \vdash \ldots \vdash B & \quad \vdash \text{Id} \quad B \vdash B \quad B \vdash \ldots \vdash A_m \\
\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
A_0 \vdash A_m
\end{align*}
\]
Elimination of identities

\[
A_0 \vdash \ldots \vdash B \quad \frac{\text{Id}}{B \vdash B} \quad B \vdash \ldots \vdash A_m \\
\hline
A_0 \vdash A_m
\]

\[
A_0 \vdash \ldots \vdash B \quad B \vdash \ldots \vdash A_m \\
\hline
\Downarrow \text{IdElim}
\hline
A_0 \vdash A_m
\]

\text{MC}
Else, $M = [R \ldots RL \ldots L]$. 
Else, \( M = [R \ldots RL \ldots L] \).

\[
\begin{align*}
A_{i-1} &\vdash B_0 & A_{i-1} &\vdash B_1 & B_j &\vdash A_{i+1} \\
\vdots & & \times & & \times & \vdots \\
A_{i-1} &\vdash B_0 \times B_1 & B_0 \times B_1 &\vdash A_{i+1} \\
\vdots & & \times j & & \vdots \\
A_0 &\vdash A_m \\
\end{align*}
\]
Essential reductions

Else, \( M = [R \ldots RL \ldots L] \).

\[
\frac{A_{i-1} \vdash B_0 \quad A_{i-1} \vdash B_1}{A_{i-1} \vdash B_0 \times B_1} \quad \frac{B_j \vdash A_{i+1}}{B_0 \times B_1 \vdash A_{i+1}} \quad \frac{A_{i-1} \vdash B}{A_{i-1} \vdash F(X)} \quad \frac{B_j \vdash A_{i+1}}{B_j \vdash F(X)}
\]

\[
\frac{R \times}{L \times j} \quad \frac{R \times}{L \times j} \quad \frac{R \times}{L \times j} \quad \frac{R \times}{L \times j}
\]

\[
\frac{A_0 \vdash A_m}{A_0 \vdash A_m}
\]

\[
\frac{A_0 \vdash A_m}{A_0 \vdash A_m}
\]
Essential reductions

Else, $M = [R \ldots RL \ldots L]$. 

\[
\begin{align*}
A_{i-1} \vdash B_0 & \quad A_{i-1} \vdash B_1 \quad \text{R} \times \quad B_j \vdash A_{i+1} \quad \text{L} \times j \\
\quad \vdots \quad A_{i-1} \vdash B_0 \times B_1 \quad & \quad B_0 \times B_1 \vdash A_{i+1} \\
\quad A_0 \vdash A_m
\end{align*}
\]

\[
\begin{align*}
A_{i-1} \vdash B_j \quad \text{R}^+ \quad B_0 \vdash A_{i+1} & \quad B_1 \vdash A_{i+1} \quad \text{L}^+ \\
\quad \vdots \quad A_{i-1} \vdash B_0 + B_1 \quad & \quad B_0 + B_1 \vdash A_{i+1} \\
\quad A_0 \vdash A_m
\end{align*}
\]
Else, \( M = [R \ldots RL \ldots L] \).
Else, $M = [R \ldots RL \ldots L]$. 

\[
\begin{align*}
A_{i-1} \vdash B_0 &\quad A_{i-1} \vdash B_1 \\
A_{i-1} \vdash B_0 \times B_1 &\quad B_j \vdash A_{i+1} \\
\cdots &\quad B_0 \times B_1 \vdash A_{i+1} \\
&\quad A_0 \vdash A_m \\
A_{i-1} \vdash B_j &\quad B_0 \vdash A_{i+1} \quad B_1 \vdash A_{i+1} \\
A_{i-1} \vdash B_0 + B_1 &\quad B_0 + B_1 \vdash A_{i+1} \\
\cdots &\quad A_0 \vdash A_m \\
A_{i-1} \vdash F(X) &\quad F(X) \vdash A_{i+1} \\
\cdots &\quad A_{i-1} \vdash X \quad X \vdash A_{i+1} \\
&\quad A_1 \vdash A_m
\end{align*}
\]
Else, $M = [R \ldots RL \ldots L]$. 

\[
\begin{array}{c}
A_{i-1} \vdash B_0 & A_{i-1} \vdash B_1 & \quad B_j \vdash A_{i+1} \\
\vdots \\
A_{i-1} \vdash B_0 \times B_1 & B_0 \times B_1 \vdash A_{i+1} & \quad L \times j
\end{array}
\]

\[
\begin{array}{c}
A_0 \vdash A_m
\end{array}
\]

\[
\begin{array}{c}
A_{i-1} \vdash B_j \\
\vdots \\
A_{i-1} \vdash B_0 + B_1 & B_0 + B_1 \vdash A_{i+1} & \quad L +
\end{array}
\]

\[
\begin{array}{c}
A_0 \vdash A_m
\end{array}
\]

\[
\begin{array}{c}
A_{i-1} \vdash F(X) \\
\vdots \\
A_{i-1} \vdash X & F(X) \vdash A_{i+1} & X \vdash A_{i+1}
\end{array}
\]

\[
\begin{array}{c}
A_1 \vdash A_m & \quad RF_X & \quad LF_X
\end{array}
\]

\[
\begin{array}{c}
A_0 \vdash A_m
\end{array}
\]
Cut elimination procedure

Production phase: If $M$ has a left rule on the left or a right rule on the right (or just an identity), produce a chunk of the output proof-tree.

Internal phase: Else, perform an internal manipulation of $M$ (Merge, IdElim or Reduce).

Repeat forever!

Theorem (F.–Santocanale, 2013)
For any multicut $M$, the internal phase halts!
Production phase: If $M$ has a left rule on the left or a right rule on the right (or just an identity), produce a chunk of the output proof-tree.
Cut elimination procedure

- **Production phase:** If $M$ has a left rule on the left or a right rule on the right (or just an identity), produce a chunk of the output proof-tree.
- **Internal phase:** Else, perform an internal manipulation of $M$ (Merge, IdElim or Reduce).
Cut elimination procedure

- **Production phase**: If $M$ has a **left** rule on the left or a **right** rule on the right (or just an identity), produce a chunk of the output proof-tree.

- **Internal phase**: Else, perform an internal manipulation of $M$ (Merge, IdElim or Reduce).

- Repeat forever!

**Theorem (F.–Santocanale, 2013)**

For any multicut $M$, the internal phase halts!
Cut elimination procedure

- **Production phase**: If $M$ has a **left** rule on the **left** or a **right** rule on the **right** (or just an identity), produce a chunk of the output proof-tree.

- **Internal phase**: Else, perform an internal manipulation of $M$ (**Merge**, **IdElim** or **Reduce**).

- **Repeat forever!**

**Theorem (F.–Santocanale, 2013)**

*For any multicut $M$, the internal phase **halts!***
Proof sketch: Suppose it never halts...

\[ M_1 = \begin{bmatrix} u_{11} & u_{12} & u_{13} \end{bmatrix} \]

Merge ↓

\[ M_2 = \begin{bmatrix} u_{21} & u_{22} & u_{23} & u_{24} \end{bmatrix} \]

Merge ↓

\[ M_3 = \begin{bmatrix} u_{31} & u_{32} & u_{33} & u_{34} & u_{35} \end{bmatrix} \]

Reduce ↓

\[ M_4 = \begin{bmatrix} u_{41} & u_{42} & u_{43} & u_{44} & u_{45} \end{bmatrix} \]

IdElim ↓

\[ M_5 = \begin{bmatrix} u_{51} & u_{52} & u_{53} & u_{54} \end{bmatrix} \]

Reduce ↓

\[ M_6 = \begin{bmatrix} u_{61} & u_{62} & u_{63} & u_{64} \end{bmatrix} \]
Proof sketch: Suppose it never halts...

\[ T := \]

\[ M_1 = \begin{bmatrix} u_{11} & u_{12} & u_{13} \end{bmatrix} \]
Merge \[ \Downarrow \]
\[ M_2 = \begin{bmatrix} u_{21} & u_{22} & u_{23} & u_{24} \end{bmatrix} \]
Merge \[ \Downarrow \]
\[ M_3 = \begin{bmatrix} u_{31} & u_{32} & u_{33} & u_{34} & u_{35} \end{bmatrix} \]
Reduce \[ \Downarrow \]
\[ M_4 = \begin{bmatrix} u_{41} & u_{42} & u_{43} & u_{44} & u_{45} \end{bmatrix} \]
IdElim \[ \Downarrow \]
\[ M_5 = \begin{bmatrix} u_{51} & u_{52} & u_{53} & u_{54} \end{bmatrix} \]
Reduce \[ \Downarrow \]
\[ M_6 = \begin{bmatrix} u_{61} & u_{62} & u_{63} & u_{64} \end{bmatrix} \]
Proof sketch: Suppose it never halts...

\[ T \quad := \]

\[
\begin{align*}
M_1 &= \begin{bmatrix}
  u_{11} & u_{12} & u_{13} \\
\end{bmatrix} \\
\text{Merge} \downarrow & \quad \begin{array}{c}
\downarrow 0 \\
\downarrow 1 \\
\downarrow 2 \\
\downarrow 0
\end{array} \\
M_2 &= \begin{bmatrix}
  u_{21} & u_{22} & u_{23} & u_{24} \\
\end{bmatrix} \\
\text{Merge} \downarrow & \quad \begin{array}{c}
\downarrow 1 \\
\downarrow 2 \\
\downarrow 0 \\
\downarrow 0 \\
\downarrow 0
\end{array} \\
M_3 &= \begin{bmatrix}
  u_{31} & u_{32} & u_{33} & u_{34} & u_{35} \\
\end{bmatrix} \\
\text{Reduce} \downarrow & \quad \begin{array}{c}
\downarrow 0 \\
\downarrow 3 \\
\downarrow 3 \\
\downarrow 0 \\
\downarrow 0
\end{array} \\
M_4 &= \begin{bmatrix}
  u_{41} & u_{42} & u_{43} & u_{44} & u_{45} \\
\end{bmatrix} \\
\text{IdElim} \downarrow & \quad \begin{array}{c}
\downarrow 0 \\
\downarrow 0 \\
\downarrow 0 \\
\downarrow 0 \\
\downarrow 0
\end{array} \\
M_5 &= \begin{bmatrix}
  u_{51} & u_{52} & u_{53} & u_{54} \\
\end{bmatrix} \\
\text{Reduce} \downarrow & \quad \begin{array}{c}
\downarrow 0 \\
\downarrow 0 \\
\downarrow 3 \\
\downarrow 3
\end{array} \\
M_6 &= \begin{bmatrix}
  u_{61} & u_{62} & u_{63} & u_{64} \\
\end{bmatrix}
\end{align*}
\]
Proof sketch: Suppose it never halts...

\[ T := * \]

\[ M_1 = [ u_{11} \ u_{12} \ u_{13} ]; \]

Merge \[ \Downarrow \]

\[ M_2 = [ u_{21} \ u_{22} \ u_{23} \ u_{24} ]; \]

Merge \[ \Downarrow \]

\[ M_3 = [ u_{31} \ u_{32} \ u_{33} \ u_{34} \ u_{35} ]; \]

Reduce \[ \Downarrow \]

\[ M_4 = [ u_{41} \ u_{42} \ u_{43} \ u_{44} \ u_{45} ]; \]

IdElim \[ \Downarrow \]

\[ M_5 = [ u_{51} \ u_{52} \ u_{53} \ u_{54} ]; \]

Reduce \[ \Downarrow \]

\[ M_6 = [ u_{61} \ u_{62} \ u_{63} \ u_{64} ]; \]
Proof sketch: Suppose it never halts...

\[ T := \]

\[ M_1 = \begin{bmatrix} u_{11} & u_{12} & u_{13} \end{bmatrix} \]

\text{Merge} \downarrow

\[ M_2 = \begin{bmatrix} u_{21} & u_{22} & u_{23} & u_{24} \end{bmatrix} \]

\text{Merge} \downarrow

\[ M_3 = \begin{bmatrix} u_{31} & u_{32} & u_{33} & u_{34} & u_{35} \end{bmatrix} \]

\text{Reduce} \downarrow

\[ M_4 = \begin{bmatrix} u_{41} & u_{42} & u_{43} & u_{44} & u_{45} \end{bmatrix} \]

\text{IdElim} \downarrow

\[ M_5 = \begin{bmatrix} u_{51} & u_{52} & u_{53} & u_{54} \end{bmatrix} \]

\text{Reduce} \downarrow

\[ M_6 = \begin{bmatrix} u_{61} & u_{62} & u_{63} & u_{64} \end{bmatrix} \]
Proof sketch: Suppose it never halts...

\[ \tilde{T} := \begin{array}{ccc}
M_1 &=& \begin{bmatrix}
u_{11} & u_12 & u_13 \\
\end{bmatrix} \\
\text{Merge} \\
M_2 &=& \begin{bmatrix}
u_{21} & u_{22} & u_{23} & u_{24} \\
\end{bmatrix} \\
\text{Merge} \\
M_3 &=& \begin{bmatrix}
u_{31} & \text{ } & \text{ } & u_{34} & u_{35} \\
\text{Reduce} \\
M_4 &=& \begin{bmatrix}
u_{41} & u_{42} & u_{43} & u_{44} & u_{45} \\
\text{IdElim} \\
M_5 &=& \begin{bmatrix}
u_{51} & u_{52} & \text{ } & u_{53} & u_{54} \\
\text{Reduce} \\
M_6 &=& \begin{bmatrix}
u_{61} & u_{62} & u_{63} & u_{64} \\
\end{bmatrix}
\end{array} \]
Proof sketch: Suppose it never halts...

\[ \bar{T} := \ast \]

Diagram:...

\[ \overset{1}{u_{21}} \]

\[ \overset{2}{u_{12}} \]

\[ \overset{1}{u_{32}} \quad \overset{2}{u_{33}} \]

\[ \overset{3}{u_{43}} \]

\[ \overset{3}{u_{53}} \quad \overset{3}{u_{54}} \]
\( \tilde{T} \) is an infinite, finitely branching tree.
• \( \tilde{T} \) is an infinite, finitely branching tree.
• The set \( \partial_{\infty} \tilde{T} \) of infinite branches of \( \tilde{T} \) is nonempty. (Kőnig)
• \( \tilde{T} \) is an infinite, finitely branching tree.
• The set \( \partial_\infty \tilde{T} \) of infinite branches of \( \tilde{T} \) is nonempty. (Kőnig)
• \( \partial_\infty \tilde{T} \) is a complete lattice (with the lexicographical order \( \preceq \)).
\( \tilde{T} \) is an infinite, finitely branching tree.

- The set \( \partial_\infty \tilde{T} \) of infinite branches of \( \tilde{T} \) is nonempty. (Kőnig)
- \( \partial_\infty \tilde{T} \) is a complete lattice (with the lexicographical order \( \preceq \)).
- Infinite branches of \( \tilde{T} \) correspond to infinite paths of \( \Pi \).
• \( \tilde{T} \) is an infinite, finitely branching tree.
• The set \( \partial_\infty \tilde{T} \) of infinite branches of \( \tilde{T} \) is nonempty. (Kőnig)
• \( \partial_\infty \tilde{T} \) is a complete lattice (with the lexicographical order \( \triangleleft \)).
• Infinite branches of \( \tilde{T} \) correspond to infinite paths of \( \Pi \).
Therefore, they satisfy the guard condition!

\[
\partial_\infty \tilde{T} = \mu\text{-branches} \cup \nu\text{-branches}
\]
\( \tilde{T} \) is an infinite, finitely branching tree.

- The set \( \partial_{\infty} \tilde{T} \) of infinite branches of \( \tilde{T} \) is nonempty. (Kőnig)
- \( \partial_{\infty} \tilde{T} \) is a complete lattice (with the lexicographical order \( \preceq \)).
- Infinite branches of \( \tilde{T} \) correspond to infinite paths of \( \Pi \).
  Therefore, they satisfy the guard condition!

\[
\partial_{\infty} \tilde{T} = \mu\text{-branches} \cup \nu\text{-branches}
\]

- WLOG, the set of \( \nu\text{-branches} \) is nonempty.
\( \tilde{T} \) is an infinite, finitely branching tree.

- The set \( \partial_{\infty} \tilde{T} \) of infinite branches of \( \tilde{T} \) is nonempty. (Kőnig)
- \( \partial_{\infty} \tilde{T} \) is a complete lattice (with the lexicographical order \( \preceq \)).
- Infinite branches of \( \tilde{T} \) correspond to infinite paths of \( \Pi \). Therefore, they satisfy the guard condition!

\[
\partial_{\infty} \tilde{T} = \mu\text{-branches} \cup \nu\text{-branches}
\]

- WLOG, the set of \( \nu\text{-branches} \) is nonempty.

**Lemma (F.–Santocanale, 2013)**

- For each \( \nu\text{-branch} \beta \), there exists another \( \nu\text{-branch} \beta' \succ \beta \).
- Let \( E \) be a collection of \( \nu\text{-branches} \) and let \( \gamma = \sup E \). Then \( \gamma \) is a \( \nu\text{-branch} \).
\( \tilde{T} \) is an infinite, finitely branching tree.

- The set \( \partial_\infty \tilde{T} \) of infinite branches of \( \tilde{T} \) is nonempty. (Kőnig)
- \( \partial_\infty \tilde{T} \) is a complete lattice (with the lexicographical order \( \preceq \)).
- Infinite branches of \( \tilde{T} \) correspond to infinite paths of \( \Pi \).
  Therefore, they satisfy the guard condition!

\[ \partial_\infty \tilde{T} = \mu\text{-branches} \cup \nu\text{-branches} \]

- WLOG, the set of \( \nu\text{-branches} \) is nonempty.

**Lemma (F.–Santocanale, 2013)**

- For each \( \nu\text{-branch} \beta \), there exists another \( \nu\text{-branch} \beta' \succ \beta \).
- Let \( E \) be a collection of \( \nu\text{-branches} \) and let \( \gamma = \sup E \). Then \( \gamma \) is a \( \nu\text{-branch} \).

- Let \( \gamma = \sup \{ \nu\text{-branches} \} \). Then \( \gamma \prec \gamma' \preceq \gamma \)!
Let \( X \) be a set defined by a directed system of equations. Lemma: For all \( x \in X \), there is a canonical non circular (possibly infinite) proof \( \Psi_x \) of \( 1 \vdash X \), such that \( J \not\vdash \Psi_x \) on the root (conclusion).

Let \( f : X \rightarrow Y \) and \( \Pi \) such that \( J \not\vdash \Pi \) on the root.

Let \( \Psi_x \odot \Pi := \cdots \Psi_x \vdash X \cdots \Pi \vdash Y \).

Theorem (F., in thesis): Cut elimination turns \( \Psi_x \odot \Pi \) into \( \Psi_f(x) \).

It is then a generic algorithm for computing the functions that one can define using circular proofs.

Jérôme Fortier

FMCS 2015, Colgate University
Operational semantics

Let $X$ be a set defined by a directed system of equations.
Let $X$ be a set defined by a directed system of equations.

**Lemma**

For all $x \in X$, there is a canonical non-circular (possibly infinite) proof $\Psi_x$ of

$1 \vdash X$, such that $\lbrack \neg \Psi_x \rbrack = x$ on the root (conclusion).
Operational semantics

Let $X$ be a set defined by a directed system of equations.

**Lemma**

For all $x \in X$, there is a canonical non circular (possibly infinite) proof $\Psi_x$ of $1 \vdash X$, such that $[!\Psi_x] = x$ on the root (conclusion).

Let $f : X \rightarrow Y$ and $\Pi$ such that $[!\Pi] = f$ on the root.
Operational semantics

Let $X$ be a set defined by a directed system of equations.

**Lemma**

*For all* $x \in X$, *there is a canonical non circular (possibly infinite) proof* $\Psi_x$ *of* $1 \vdash X$, *such that* $\llbracket !\Psi_x \rrbracket = x$ *on the root (conclusion).*

Let $f : X \to Y$ and $\Pi$ *such that* $\llbracket !\Pi \rrbracket = f$ *on the root. Let*

$$
\Psi_x \odot \Pi := 1 \vdash X \quad X \vdash Y.
$$

$$
\Psi_x \odot \Pi \vdash 1 \vdash Y
$$

**Theorem (F., in thesis)**

Cut elimination turns $\Psi_x \odot \Pi$ into $\Psi_f(x)$. It is then a generic algorithm for computing the functions that one can define using circular proofs.
Operational semantics

Let $X$ be a set defined by a directed system of equations.

**Lemma**

For all $x \in X$, there is a canonical non circular (possibly infinite) proof $\Psi_x$ of $1 \vdash X$, such that $\llbracket ! \Psi_x \rrbracket = x$ on the root (conclusion).

Let $f : X \rightarrow Y$ and $\Pi$ such that $\llbracket ! \Pi \rrbracket = f$ on the root. Let

$$
\Psi_x \odot \Pi := 1 \vdash X \quad X \vdash Y.
$$

**Theorem (F., in thesis)**

*Cut elimination turns $\Psi_x \odot \Pi$ into $\Psi_{f(x)}$.***
Let $X$ be a set defined by a directed system of equations.

**Lemma**

For all $x \in X$, there is a canonical non circular (possibly infinite) proof $\Psi_x$ of $1 \vdash X$, such that $\langle !\Psi_x \rangle = x$ on the root (conclusion).

Let $f : X \rightarrow Y$ and $\Pi$ such that $\langle !\Pi \rangle = f$ on the root. Let

$$
\Psi_x \odot \Pi := 1 \vdash X \quad X \vdash Y \quad C
\quad 1 \vdash Y
$$

**Theorem (F., in thesis)**

Cut elimination turns $\Psi_x \odot \Pi$ into $\Psi_{f(x)}$. It is then a generic algorithm for computing the functions that one can define using circular proofs.
Expressive power

Question
What are the set-theoretic functions that one can **define** with circular proofs and/or **compute** with cut-elimination?
Expressive power

Question

What are the set-theoretic functions that one can define with circular proofs and/or compute with cut-elimination?

- Primitive recursive $\leq$ C. Definable (Paré–Román ’89);
Expressive power

Question
What are the set-theoretic functions that one can define with circular proofs and/or compute with cut-elimination?

- Primitive recursive \( \leq \) C. Definable (Paré–Román ’89);
- C. Definable \( \leq \) C. Computable \( \leq \) Computable (By cut-elimination);
Question

What are the set-theoretic functions that one can define with circular proofs and/or compute with cut-elimination?

- Primitive recursive $\leq$ C. Definable (Paré–Román ’89);
- C. Definable $\leq$ C. Computable $\leq$ Computable (By cut-elimination);
Let $\Sigma$ be a signature (a finite set with arity).

**Definition (circular)**

A $\Sigma$-tree is a tuple $t = (f, t_1 \ldots t_r)$ such that $f \in \Sigma$, $r = \text{ar}(f)$ and $t_1 \ldots t_r$ are $\Sigma$-trees.
Let $\Sigma$ be a signature (i.e., finite set with arity).

**Definition (circular)**

A $\Sigma$-tree is a tuple $t = (f, t_1 \ldots t_r)$ such that $f \in \Sigma$, $r = \text{ar}(f)$ and $t_1 \ldots t_r$ are $\Sigma$-trees.
Let $\Sigma$ be a signature ($=\$ finite set with arity$).

**Definition (circular)**

A $\Sigma$-tree is a tuple $t = (f, t_1 \ldots t_r)$ such that $f \in \Sigma$, $r = \text{ar}(f)$ and $t_1 \ldots t_r$ are $\Sigma$-trees.

\[
T = \nu \prod_{f \in \Sigma} f , \quad f = \nu \prod_{j=1}^{\text{ar}(f)} T \quad \text{(pour chaque } f \in \Sigma).\]
Example: $\Sigma = \{a, s, z\}$, $ar(a) = 2$, $ar(s) = 1$, $ar(z) = 0$. 
Example: $\Sigma = \{a, s, z\}$, $\text{ar}(a) = 2$, $\text{ar}(s) = 1$, $\text{ar}(z) = 0$.

We can encode streams of natural numbers as “combs”.
Example: $\Sigma = \{a, s, z\}$, $\text{ar}(a) = 2$, $\text{ar}(s) = 1$, $\text{ar}(z) = 0$.

We can encode streams of natural numbers as “combs”.

$$x = (0, 1, 2, 3, 4 \ldots) \implies \text{Comb}(x) = a$$
Remark:
The “comb” function is denoted by a circular proof.
Remark:
The “comb ”function is denoted by a circular proof.
We can directly encode $\Sigma$-trees into infinite proofs.

$$t = (f, t_1 \ldots t_r):$$

$$\Psi_t:\begin{array}{c}
\vdash T \\
\vdash T \\
\vdash \prod_1^r T \\
\vdash f \\
\vdash \bigwedge_{i \in \Sigma} i \\
\vdash T
\end{array}$$
Let $\Gamma$ be a finite (stack) alphabet.

**Definition**

A 0-stack is an element of $\Gamma$. For $n \geq 1$, a $n$-stack is a finite list of $(n-1)$-stacks.
Let $\Gamma$ be a finite (stack) alphabet.

**Definition**

A **0-stack** is an element of $\Gamma$. For $n \geq 1$, a **well-formed $n$-stack** is a finite nonempty list of well-formed $(n-1)$-stacks.
Stacks of stacks of stacks of . . . stacks

Let $\Gamma$ be a finite (stack) alphabet.

**Definition**

A 0-stack is an element of $\Gamma$. For $n \geq 1$, a well-formed $n$-stack is a finite nonempty list of well-formed $(n-1)$-stacks.

For well-formed $n$-stacks, let

$$
top(a) = a \quad \text{if } a \in \Gamma;$$
$$
top[s_{\ell}, s_{\ell-1} \ldots s_1] = top(s_{\ell}).$$
A directed system for $n$-stacks
A directed system for \( n \)-stacks

Evident idea: Iterate the Kleene star:

\[
S^* = \mu 1 + (S \times S^*)
\]
A directed system for $n$-stacks

Evident idea: Iterate the Kleene star:

$$S^* = \mu 1 + (S \times S^*)$$

Does not work out well with circular proofs!

$$\frac{S \vdash B}{S \times S^* \vdash B} \text{ L} \times_0 \quad \text{or} \quad \frac{S^* \vdash B}{S \times S^* \vdash B} \text{ L} \times_1$$
A directed system for $n$-stacks

Evident idea: Iterate the Kleene star:

$$S^* = \mu 1 + (S \times S^*)$$

Does not work out well with circular proofs!

$$S \vdash B \quad \text{or} \quad S^* \vdash B$$

Let's iterate the free monad instead!

$$S_0(X) = \coprod_{a \in \Gamma} X$$

$$S_n(X) = \hat{S}_{n-1}(X) = \mu Y.(X + S_n(Y))$$
A directed system for $n$-stacks

Evident idea: Iterate the Kleene star:

$$S^* = \mu \ 1 + (S \times S^*)$$

Does not work out well with circular proofs!

$$\frac{S \vdash B}{S \times S^* \vdash B} \text{L} \times 0 \quad \text{or} \quad \frac{S^* \vdash B}{S \times S^* \vdash B} \text{L} \times 1$$

Let’s iterate the free monad instead!

$$S_0(X) = \prod_{a \in \Gamma} X$$

$$S_n(X) = \hat{S}_{n-1}(X) = \mu Y.(X + S_n(Y)).$$

Lemma (F. 2015)

For all $n$, $S_n(1)$ is (isomorphic to) the set of $n$-stacks.
Operations on $n$-stacks

- $\text{spush}_1^a[a_\ell, a_{\ell-1} \ldots a_1] = [a, a_\ell, a_{\ell-1} \ldots a_1]$;
- $\text{push}_n^n[s_\ell, s_{\ell-1} \ldots s_1] = [s_\ell, s_\ell, s_{\ell-1} \ldots s_1]$;
- $\text{pop}_n^n[s_\ell, s_{\ell-1} \ldots s_1] = [s_{\ell-1} \ldots s_1]$;
- $\text{spush}_n^a[s_\ell, s_{\ell-1} \ldots s_1] = [\text{spush}_n^{a-1}(s_\ell), s_{\ell-1} \ldots s_1]$;
- $\text{push}_n^k[s_\ell, s_{\ell-1} \ldots s_1] = [\text{push}_n^{k-1}(s_\ell), s_{\ell-1} \ldots s_1]$;
- $\text{pop}_n^k[s_\ell, s_{\ell-1} \ldots s_1] = [\text{pop}_n^{k-1}(s_\ell), s_{\ell-1} \ldots s_1]$.  

Proposition (F. 2015)

Each of these operations is definable by a circular proof.
Operations on $n$-stacks

- $\text{spush}_1^a[a_\ell, a_{\ell-1} \ldots a_1] = [a, a_\ell, a_{\ell-1} \ldots a_1]$;
- $\text{push}_n^n[s_\ell, s_{\ell-1} \ldots s_1] = [s_\ell, s_\ell, s_{\ell-1} \ldots s_1]$;
- $\text{pop}_n^n[s_\ell, s_{\ell-1} \ldots s_1] = [s_{\ell-1} \ldots s_1]$;
- $\text{spush}_n^a[s_\ell, s_{\ell-1} \ldots s_1] = [\text{spush}_{n-1}^a(s_\ell), s_{\ell-1} \ldots s_1]$;
- $\text{push}_n^k[s_\ell, s_{\ell-1} \ldots s_1] = [\text{push}_{n-1}^k(s_\ell), s_{\ell-1} \ldots s_1]$;
- $\text{pop}_n^k[s_\ell, s_{\ell-1} \ldots s_1] = [\text{pop}_{n-1}^k(s_\ell), s_{\ell-1} \ldots s_1]$.

Proposition (F. 2015)

*Each of these operations is definable by a circular proof*
Higher-Order Pushdown Automata

Definition (Knapik, Niwinski, Urzyczyn, 2001)

A level $n$ pushdown automaton is a tuple $A = \langle Q, \Sigma, \Gamma, q_0, \delta \rangle$, where $Q$ is a finite set of states with an initial state $q_0 \in Q$, and $\delta : Q \times \Gamma \rightarrow \mathcal{I}_A$ is the transition function, where $\mathcal{I}_A$ is the set of admissible instructions, consisting of the expressions of one of the two following forms:

- $(q, \varphi)$, where $q \in Q$ and $\varphi$ is a level $n$ operation;
- $(f, p_1 \ldots p_r)$, where $f \in \Sigma$, $r = \text{ar}(f)$ and $p_1 \ldots p_r \in Q$. 

Jérôme Fortier  
FMCS 2015, Colgate University
A *level n pushdown automaton* is a tuple $A = \langle Q, \Sigma, \Gamma, q_0, \delta \rangle$, where $Q$ is a finite set of states with an initial state $q_0 \in Q$, and $\delta: Q \times \Gamma \to I_A$ is the transition function, where $I_A$ is the set of admissible instructions, consisting of the expressions of one of the two following forms:

- $(q, \varphi)$, where $q \in Q$ and $\varphi$ is a level $n$ operation;
- $(f, p_1 \ldots p_r)$, where $f \in \Sigma$, $r = \text{ar}(f)$ and $p_1 \ldots p_r \in Q$.

Transitions of the form $(f, p_1 \ldots p_r)$ produce a chunk of a $\Sigma$-tree;
Higher-Order Pushdown Automata

Definition (Knapik, Niwinski, Urzyczyn, 2001)

A level $n$ pushdown automaton is a tuple $A = \langle Q, \Sigma, \Gamma, q_0, \delta \rangle$, where $Q$ is a finite set of states with an initial state $q_0 \in Q$, and $\delta : Q \times \Gamma \rightarrow I_A$ is the transition function, where $I_A$ is the set of admissible instructions, consisting of the expressions of one of the two following forms:

- $(q, \varphi)$, where $q \in Q$ and $\varphi$ is a level $n$ operation;
- $(f, p_1 \ldots p_r)$, where $f \in \Sigma$, $r = \text{ar}(f)$ and $p_1 \ldots p_r \in Q$.

- Transitions of the form $(f, p_1 \ldots p_r)$ produce a chunk of a $\Sigma$-tree;
- Other transitions silently manipulate the stack;
Higher-Order Pushdown Automata

**Definition (Knapik, Niwinski, Urzyczyn, 2001)**

A *level n pushdown automaton* is a tuple $A = \langle Q, \Sigma, \Gamma, q_0, \delta \rangle$, where $Q$ is a finite set of states with an initial state $q_0 \in Q$, and $\delta : Q \times \Gamma \rightarrow I_A$ is the transition function, where $I_A$ is the set of admissible instructions, consisting of the expressions of one of the two following forms:

- $(q, \varphi)$, where $q \in Q$ and $\varphi$ is a level $n$ operation;
- $(f, p_1 \ldots p_r)$, where $f \in \Sigma$, $r = \text{ar}(f)$ and $p_1 \ldots p_r \in Q$.

- Transitions of the form $(f, p_1 \ldots p_r)$ produce a chunk of a $\Sigma$-tree;
- Other transitions silently manipulate the stack;
- Hence, $n$-PDAs are acceptors of $\Sigma$-trees.
The result

Theorem (F. 2015)

Let $A$ be a $n$-PDA and let $t$ be the $\Sigma$-tree accepted by $A$. Then $t$ is circularly computable,
Theorem (F. 2015)

Let $A$ be a $n$-PDA and let $t$ be the $\Sigma$-tree accepted by $A$. Then $t$ is circularly computable, which means that there exists a finite (not necessarily guarded) pre-proof $\Pi$ for which cut-elimination leads to $\Psi_t$. 
The result

**Theorem (F. 2015)**

Let $\mathcal{A}$ be a $n$-PDA and let $t$ be the $\Sigma$-tree accepted by $\mathcal{A}$. Then $t$ is circularly computable, which means that there exists a finite (not necessarily guarded) pre-proof $\Pi$ for which cut-elimination leads to $\Psi_t$.

**Proof:** Simulate the behaviour of $\mathcal{A}$ with cut-elimination.
The converse does not hold!

Let $h, f : \mathbb{N} \to \mathbb{N}$,

$$h_0(x) = x, \quad h_{k+1}(x) = 2^{h_k(x)} \quad \text{and} \quad f(x) = h_x(1).$$
The converse does not hold!

Let \( h_k, f : \mathbb{N} \to \mathbb{N} \),

\[
h_0(x) = x, \quad h_{k+1}(x) = 2^{h_k(x)} \quad \text{and} \quad f(x) = h_x(1).
\]

\( f \) is primitive recursive, hence, \( \text{Comb}(f) \) is circularly definable and then circularly computable.
The converse does not hold!

Let $h_k, f : \mathbb{N} \to \mathbb{N}$,

$h_0(x) = x$, $h_{k+1}(x) = 2^{h_k(x)}$ and $f(x) = h_x(1)$.

- $f$ is primitive recursive, hence, $\text{Comb}(f)$ is circularly definable and then circularly computable.

- But, if $\text{Comb}(f)$ was accepted by some $n$-PDA, then by (Damm, 1982), there would be some polynomial $p$ such that

$$f(x) \leq (h_{2n} \circ p)(x)$$

for $x$ large enough: a contradiction!
Merci beaucoup!

To learn more:

