Cyclic Implicit Complexity

SCALP 2021

Fontainebleau

Gianluca Curzi

University of Birmingham

joint work with Anupam Das (University of Birmingham)

What is this presentation about?

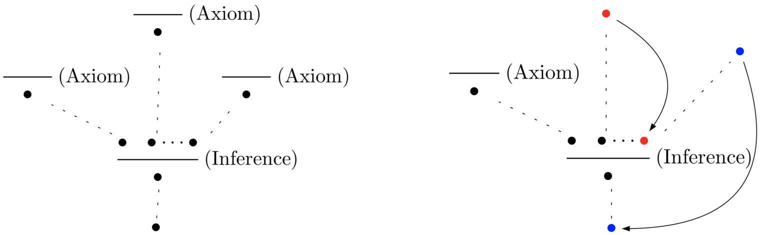


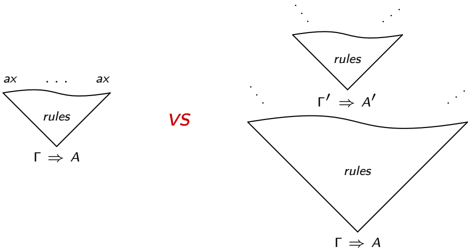
Figure: From "Introduction to cyclic proofs" (Brotherston 2008).

- Goal: cyclic proof systems to capture complexity classes in the style of ICC.
- Some motivations:
  - cyclic proofs **subsume** various recursion schemes;
  - relatively **new topic**, not much about complexity-theoretic aspects of cyclic proofs;
  - hard to **tame complexity**, criteria to weaken loop structure.

.

Non-wellfounded proofs

- Inductive vs non-wellfounded proofs:



- **Non-wellfounded proofs** to reason about μ-calculus (e.g. [Dax, Hofmann and Lange 06], [Niwinski and Walukiewicz 96]), (co)induction (e.g. [Brotherston and Simpson 11]), Kleene algebra (e.g. [Das and Pous 17, 18]), linear logic (e.g. [Baelde, Doumane and Saurin 16]), continuous cut-elimination (e.g. [Mints 75] and [Fortier and Santocanale 13]).

- **Problem**. Any formula is derivable!

$$\text{cut}\ \dfrac{\text{cut}\ \dfrac{\begin{array}{c}\vdots\\ \Rightarrow A\end{array} \qquad \text{id}\ \dfrac{}{A \Rightarrow A}}{\Rightarrow A} \qquad \text{id}\ \dfrac{}{A \Rightarrow A}}{\Rightarrow A}$$

- Progressiveness criterion = global criterion to guarantee consistency.

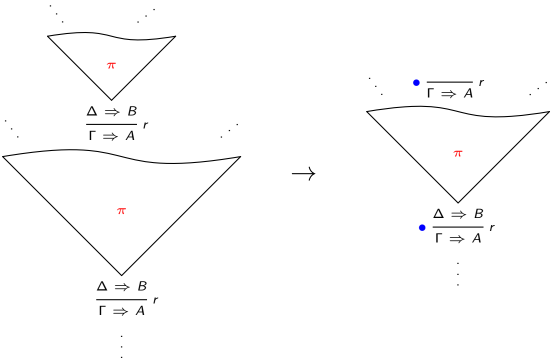
- **Problem**. Any formula is derivable!

$$
\mathrm{cut}\;\dfrac{\mathrm{cut}\;\dfrac{\begin{array}{c}\vdots\\ \Rightarrow A\end{array} \qquad \mathrm{id}\;\dfrac{}{A \Rightarrow A}}{\Rightarrow A} \qquad \mathrm{id}\;\dfrac{}{A \Rightarrow A}}{\Rightarrow A}
$$

- Progressiveness criterion = global criterion to guarantee consistency.

Cyclic proofs

- Cyclic proofs = only **finitely** many distinct subproofs.
- Cycle normal form = finite, “circular” presentation of a cyclic proof.



ICC and safe recursion on notation

- Function algebra B characterizing **FPTIME** [Bellantoni and Cook 92].
- Two successors: $s_0x = 2x$ and $s_1x = 2x + 1$.
- Function arguments partitioned into normal and safe:

$$f(x_1, \ldots, x_n \, ; \, y_1, \ldots, y_m)$$

- Safe recursion on notation:

$$\begin{aligned} f(0, \vec{x}; \vec{y}) &= g(\vec{x}; \vec{y}) \\ f(s_0x, \vec{x}; \vec{y}) &= h_0(x, \vec{x}; \vec{y}, f(x, \vec{x}; \vec{y})) \\ f(s_1x, \vec{x}; \vec{y}) &= h_1(x, \vec{x}; \vec{y}, f(x, \vec{x}; \vec{y})) \end{aligned}$$

Idea. Recursive calls **only** in the safe zone.

ICC and safe recursion on notation

- Function algebra B characterizing **FPTIME** [Bellantoni and Cook 92].

- Two successors: $s_0 x = 2x$ and $s_1 x = 2x + 1$.

- Function arguments partitioned into normal and safe:

$$f(x_1, \ldots, x_n\,;\, y_1, \ldots, y_m)$$

- Safe recursion on notation:

$$\begin{aligned} f(0, \vec{x}; \vec{y}) &= g(\vec{x}; \vec{y}) \\ f(s_0 x, \vec{x}; \vec{y}) &= h_0(x, \vec{x}; \vec{y}, f(x, \vec{x}; \vec{y})) \\ f(s_1 x, \vec{x}; \vec{y}) &= h_1(x, \vec{x}; \vec{y}, f(x, \vec{x}; \vec{y})) \end{aligned}$$

Idea. Recursive calls **only** in the safe zone:

ICC and safe recursion on notation

- Function algebra B characterizing **FPTIME** [Bellantoni and Cook 92].

- Two successors: $s_0 x = 2x$ and $s_1 x = 2x + 1$.

- Function arguments partitioned into normal and safe:

$$f(x_1, \ldots, x_n \,;\, y_1, \ldots, y_m)$$

- Safe recursion on notation:

$$\begin{aligned} f(0, \vec{x}; \vec{y}) &= g(\vec{x}; \vec{y}) \\ f(s_0 x, \vec{x}; \vec{y}) &= h_0(x, \vec{x}; \vec{y}, f(x, \vec{x}; \vec{y})) \\ f(s_1 x, \vec{x}; \vec{y}) &= h_1(x, \vec{x}; \vec{y}, f(x, \vec{x}; \vec{y})) \end{aligned}$$

Idea. Recursive calls **only** in the safe zone:

ICC and safe recursion on notation

- Function algebra B characterizing **FPTIME** [Bellantoni and Cook 92].

- Two successors: $s_0 x = 2x$ and $s_1 x = 2x + 1$.

- Function arguments partitioned into normal and safe:

$$f(x_1, \ldots, x_n \,;\, y_1, \ldots, y_m)$$

- Safe recursion on notation:

$$\begin{aligned} f(0, \vec{x}; \vec{y}) &= g(\vec{x}; \vec{y}) \\ f(s_0 x, \vec{x}; \vec{y}) &= h_0(x, \vec{x}; \vec{y}, f(x, \vec{x}; \vec{y})) \\ f(s_1 x, \vec{x}; \vec{y}) &= h_1(x, \vec{x}; \vec{y}, f(x, \vec{x}; \vec{y})) \end{aligned}$$

Idea. Recursive calls **only** in the safe zone:

Non-wellfounded version of B

- Formulas $A, B, C \in \{N, \Box N\}$ and contexts $\Gamma, \Delta = A_1, \ldots, A_n$.

- Non-wellfounded proofs generated by the following rules:

$$\text{id}\ \frac{}{N \Rightarrow N} \qquad \text{w}_N\ \frac{\Gamma \Rightarrow B}{\Gamma, N \Rightarrow B} \qquad \text{w}_\Box\ \frac{\Gamma \Rightarrow B}{\Box N, \Gamma \Rightarrow B} \qquad \text{e}\ \frac{\Gamma, A, B, \Gamma' \Rightarrow C}{\Gamma, B, A, \Gamma' \Rightarrow C}$$

$$\Box_l\ \frac{\Gamma, N \Rightarrow A}{\Box N, \Gamma \Rightarrow A} \qquad \Box_r\ \frac{\Box N, \ldots, \Box N \Rightarrow N}{\Box N, \ldots, \Box N \Rightarrow \Box N} \qquad 0\ \frac{}{\Rightarrow N} \qquad \text{s}_0\ \frac{}{N \Rightarrow N} \qquad \text{s}_1\ \frac{}{N \Rightarrow N}$$

$$\text{cut}_N\ \frac{\Gamma \Rightarrow N \quad \Gamma, N \Rightarrow B}{\Gamma \Rightarrow B} \qquad \text{cut}_\Box\ \frac{\Gamma \Rightarrow \Box N \quad \Box N, \Gamma \Rightarrow B}{\Gamma \Rightarrow B}$$

$$\text{cond}_N\ \frac{\Gamma \Rightarrow N \quad \Gamma, N \Rightarrow N \quad \Gamma, N \Rightarrow N}{\Gamma, N \Rightarrow N} \qquad \text{cond}_\Box\ \frac{\Gamma \Rightarrow N \quad \Box N, \Gamma \Rightarrow N \quad \Box N, \Gamma \Rightarrow N}{\Box N, \Gamma \Rightarrow N}$$

Semantics of non-wellfounded proofs for B

$$0\ \frac{}{\Rightarrow N} \qquad f_{\mathcal{D}}(;) := 0$$

$$\mathsf{s}_i\ \frac{}{N \Rightarrow N} \qquad f_{\mathcal{D}}(;x) := \mathsf{s}_i x$$

$$\text{cut}\ \frac{\overset{\mathcal{D}_0}{\Gamma \Rightarrow N} \quad \overset{\mathcal{D}_1}{\Gamma, N \Rightarrow A}}{\Gamma \Rightarrow A} \qquad f_{\mathcal{D}}(\vec{x};\vec{y}) := f_{\mathcal{D}_1}(\vec{x};\vec{y}, f_{\mathcal{D}_0}(\vec{x};\vec{y}))$$

$$\text{cut}_\Box\ \frac{\overset{\mathcal{D}_0}{\Gamma \Rightarrow \Box N} \quad \overset{\mathcal{D}_1}{\Box N, \Gamma \Rightarrow A}}{\Gamma \Rightarrow A} \qquad f_{\mathcal{D}}(\vec{x};\vec{y}) := f_{\mathcal{D}_1}(f_{\mathcal{D}_0}(\vec{x};\vec{y}), \vec{x};\vec{y})$$

$$\text{cond}_\Box\ \frac{\overset{\mathcal{D}_0}{\Gamma \Rightarrow N} \quad \overset{\mathcal{D}_1}{\Box N, \Gamma, \Rightarrow N} \quad \overset{\mathcal{D}_2}{\Box N, \Gamma \Rightarrow N}}{\Box N, \Gamma \Rightarrow N}$$

$$\begin{aligned} f_{\mathcal{D}}(0, \vec{x};\vec{y}) &:= f_{\mathcal{D}_0}(\vec{x};\vec{y}) \\ f_{\mathcal{D}}(\mathsf{s}_0 x, \vec{x};\vec{y}) &:= f_{\mathcal{D}_1}(x, \vec{x};\vec{y}) \\ f_{\mathcal{D}}(\mathsf{s}_1 x, \vec{x};\vec{y}) &:= f_{\mathcal{D}_2}(x, \vec{x};\vec{y}) \end{aligned}$$

Cyclicity

- Cyclic proof = finitely many distinct subproofs.

- **Idea**. Cyclicity = computability.

- **Example.** A cyclic proof $\mathcal{D}$:

$$\dfrac{\mathsf{s}_0\,\dfrac{}{N \Rightarrow N} \qquad \mathsf{cut}_N\,\dfrac{}{\Box N, N \Rightarrow N}\,\bullet}{\Box N, N \Rightarrow N}\,\mathsf{cut}_N\,\bullet$$

$$f_{\mathcal{D}}(x; y) \quad := \quad f_{\mathcal{D}}(x; \mathsf{s}_0 y)$$

Progressiveness

- Progressive proof = every infinite branch contains a $\Box N$-*thread* with infinitely many principal formulas of the rule $\text{cond}_\Box$.

- **Example.** A progressing proof:

$$
\text{cond}_\Box\,\dfrac{
\text{id}\,\dfrac{}{N \Rightarrow N}
\qquad
\text{cut}_N\,\dfrac{\text{cond}_\Box\,\dfrac{}{\Box N, N \Rightarrow N}\bullet \qquad s_0\,\dfrac{}{N \Rightarrow N}}{\Box N, N \Rightarrow N}
\qquad
\text{cut}_N\,\dfrac{\text{cond}_\Box\,\dfrac{}{\Box N, N \Rightarrow N}\bullet \qquad s_1\,\dfrac{}{N \Rightarrow N}}{\Box N, N \Rightarrow N}
}{\Box N, N \Rightarrow N}\bullet
$$

$$
\begin{aligned}
f_{\mathcal{D}}(0; y) &= y \\
f_{\mathcal{D}}(s_0 x; y) &= s_0(f_{\mathcal{D}}(x; y)) \\
f_{\mathcal{D}}(s_1 x; y) &= s_1(f_{\mathcal{D}}(x; y))
\end{aligned}
$$

- **Idea**. Progressiveness = totality.

Safety

- **Problem**. Modalities are not enough to enforce stratification in an non-wellfounded setting.

- **Example.** A cyclic progressing proof $\mathcal{D}$ for primitive recursion (on notation):

$$\text{cond}_\Box\ \dfrac{\dfrac{\mathcal{D}_0}{\Gamma \Rightarrow N} \qquad \text{cut}_\Box\ \dfrac{\dfrac{}{\Box N, \Gamma \Rightarrow N}\,\bullet \qquad \dfrac{\mathcal{D}_1}{\Box N, \Gamma, \Box N \Rightarrow N}}{\Box N, \Gamma \Rightarrow N} \qquad \text{cut}_\Box\ \dfrac{\dfrac{}{\Box N, \Gamma \Rightarrow N}\,\bullet \qquad \dfrac{\mathcal{D}_2}{\Box N, \Gamma, \Box N \Rightarrow N}}{\Box N, \Gamma \Rightarrow N}}{\Box N, \Gamma \Rightarrow N}\,\bullet$$

$$\begin{aligned} f_{\mathcal{D}}(0, \vec{x};\,) &= f_{\mathcal{D}_0}(\vec{x};\,) \\ f_{\mathcal{D}}(s_0 x, \vec{x};\,) &= f_{\mathcal{D}_1}(x, \vec{x}, f(x, \vec{x});\,) \\ f_{\mathcal{D}}(s_1 x, \vec{x};\,) &= f_{\mathcal{D}_2}(x, \vec{x}, f(x, \vec{x};\,);\,) \end{aligned}$$

- Safe proof = any branch crosses finitely many $\text{cut}_\Box$-steps.

- Cyclic proof system NCB = cyclic progressing safe proofs.

Safety

- **Problem**. Modalities are not enough to enforce stratification in an non-wellfounded setting.

- **Example.** A cyclic progressing proof $\mathcal{D}$ for primitive recursion (on notation):

$$\mathrm{cond}_\Box\ \dfrac{\dfrac{\mathcal{D}_0}{\Gamma \Rightarrow N} \qquad \mathrm{cut}_\Box\ \dfrac{\overline{\Box N, \Gamma \Rightarrow N}\ \bullet \qquad \dfrac{\mathcal{D}_1}{\Box N, \Gamma, \Box N \Rightarrow N}}{\Box N, \Gamma \Rightarrow N} \qquad \mathrm{cut}_\Box\ \dfrac{\overline{\Box N, \Gamma \Rightarrow N}\ \bullet \qquad \dfrac{\mathcal{D}_2}{\Box N, \Gamma, \Box N \Rightarrow N}}{\Box N, \Gamma \Rightarrow N}}{\Box N, \Gamma \Rightarrow N}\ \bullet$$

$$f_{\mathcal{D}}(0, \vec{x};\,) = f_{\mathcal{D}_0}(\vec{x};\,)$$
$$f_{\mathcal{D}}(s_0 x, \vec{x};\,) = f_{\mathcal{D}_1}(x, \vec{x}, f(x, \vec{x});\,)$$
$$f_{\mathcal{D}}(s_1 x, \vec{x};\,) = f_{\mathcal{D}_2}(x, \vec{x}, f(x, \vec{x};\,);\,)$$

- Safe proof = any branch crosses finitely many $\mathrm{cut}_\Box$-steps.

- Cyclic proof system NCB = cyclic progressing safe proofs.

Nesting

- **Problem**. NCB can express **nested recursion**.

- **Example.** A cyclic progressing safe proof for the **exponential** function $\exp(x)(y) = 2^{2^{|x|}} \cdot y$:

$$\mathrm{cond}_{\square}\dfrac{\mathrm{s}_0\dfrac{}{N \Rightarrow N} \qquad \mathrm{cut}_N\dfrac{\mathrm{cond}_{\square}\dfrac{}{\square N, N \Rightarrow N}\bullet \qquad \mathrm{cond}_{\square}\dfrac{}{\square N, N \Rightarrow N}\bullet}{\square N, N \Rightarrow N} \qquad \mathrm{cut}_N\dfrac{\mathrm{cond}_{\square}\dfrac{}{\square N, N \Rightarrow N}\bullet \qquad \mathrm{cond}_{\square}\dfrac{}{\square N, N, \Rightarrow N}\bullet}{\square N, N \Rightarrow N}}{\square N, N \Rightarrow N}\bullet$$

$$\begin{aligned}
\exp(0; y) &= \mathrm{s}_0 y \\
\exp(\mathrm{s}_0 x; y) &= \exp(x; \exp(x; y)) \\
\exp(\mathrm{s}_1 x; y) &= \exp(x; \exp(x; y))
\end{aligned}$$

- Left-leaning proof = any branch goes right at a cut_N-step only finitely often.

- Cyclic proof system CB = cyclic progressing safe left-leaning proofs.

Nesting

- **Problem**. NCB can express **nested recursion**.

- **Example.** A cyclic progressing safe proof for the **exponential** function $\exp(x)(y) = 2^{2^{|x|}} \cdot y$:

$$\mathrm{cond}_{\Box}\,\dfrac{\mathrm{s}_0\,\dfrac{}{N \Rightarrow N} \qquad \mathrm{cut}_N\,\dfrac{\mathrm{cond}_{\Box}\,\dfrac{}{\Box N, N \Rightarrow N}\,\bullet \qquad \mathrm{cond}_{\Box}\,\dfrac{}{\Box N, N \Rightarrow N}\,\bullet}{\Box N, N \Rightarrow N} \qquad \mathrm{cut}_N\,\dfrac{\mathrm{cond}_{\Box}\,\dfrac{}{\Box N, N \Rightarrow N}\,\bullet \qquad \mathrm{cond}_{\Box}\,\dfrac{}{\Box N, N, \Rightarrow N}\,\bullet}{\Box N, N \Rightarrow N}}{\Box N, N \Rightarrow N}\,\bullet$$

$$\begin{aligned}
\exp(0; y) &= \mathrm{s}_0 y \\
\exp(\mathrm{s}_0 x; y) &= \exp(x; \exp(x; y)) \\
\exp(\mathrm{s}_1 x; y) &= \exp(x; \exp(x; y))
\end{aligned}$$

- Left-leaning proof = any branch goes right at a cut_N-step only finitely often.
- Cyclic proof system CB = cyclic progressing safe left-leaning proofs.

Results and perspectives

Characterization results:

- **Theorem**. NCB captures exactly **FELEMENTARY**.
- **Theorem**. CB captures exactly **FPTIME**.

Conclusions and future directions:

- CB = circular version of B
- NCB = generalization of B to nested safe recursion schemes
- Higher-order version of cyclic proof systems based on Hofmann's SLR?
- Cyclic proof systems to characterize other complexity classes, like **FPSPACE**, **ALOGTIME**, **NC**?

Thank you!
Questions?

Appendix

Hofmann's type system SLR

- Two function spaces: $\Box A \to B$ (*modal*) and $A \multimap B$ (*linear*).

- Safe linear recursion operator (with A $\Box$-*free*):

$$\mathrm{rec}_A : \underset{x}{\Box N} \to \underbrace{(\Box N \to A \multimap A)}_{h} \to \underset{g}{A} \to A$$

where $f(x) = \mathrm{rec}_A(x, h, g)$ means:

$$\begin{array}{rcl} f(0) & = & g \\ f(\mathsf{s}_0 x) & = & h(x, f(x)) \\ f(\mathsf{s}_1 x) & = & h(x, f(x)) \end{array}$$

- SLR captures exactly **FPTIME**.

Nesting and abstraction complexity

- Nested recursion in SLR if higher-order types are not handled **linearly**:

$$
\begin{array}{lcll}
A & = & N \to N & \\
g & = & \mathsf{s}_0 & : A \\
h & = & \lambda x : \Box N . \lambda u : N \to N . \lambda y : N . u(uy) & : \Box N \to A \to A \to A
\end{array}
$$

$$\exp(x; y) = \mathrm{rec}_A(x, h, g)(y)$$

- **Takeaway**. Type n cyclic proofs can represent type n+1 recursion [Das 21].

Nesting and abstraction complexity

- Nested recursion in SLR if higher-order types are not handled **linearly**:

$$\begin{array}{rcll} A & = & N \to N & \\ g & = & \mathsf{s}_0 & : A \\ h & = & \lambda x : \Box N . \lambda u : N \to N . \lambda y : N . u(uy) & : \Box N \to A \to A \to A \end{array}$$

$$\exp(x; y) = \mathrm{rec}_A(x, h, g)(y)$$

- **Takeaway**. Type n cyclic proofs can represent type n+1 recursion [Das 21].