Non-uniform complexity via non-wellfounded proofs

Turin, 11 January 2023

Gianluca Curzi

University of Birmingham

joint work with Anupam Das (University of Birmingham)

What is this presentation about?

- This talk:
  - cyclic proof systems for **FP** and **FELEMENTARY**
  - non-wellfounded proof system for **FP**/**poly**

- Some motivations:
  - **new topic**, not much about complexity-theoretic aspects of circular reasoning;
  - circular proofs **subsume** several recursion schemes;
  - **hard to tame complexity**: study conditions that identify computational and complexity-theoretic notions (uniformity, totality, safety) within cyclic proofs.

.

What is this presentation about?

- This talk:
  - cyclic proof systems for **FP** and **FELEMENTARY**
  - non-wellfounded proof system for **FP**/**poly**

- Some motivations:
  - **new topic**, not much about complexity-theoretic aspects of circular reasoning;
  - circular proofs **subsume** several recursion schemes;
  - **hard to tame complexity**: study conditions that identify computational and complexity-theoretic notions (uniformity, totality, safety) within cyclic proofs.

.

1. Implicit Computational complexity
2. Cyclic proofs
3. Cyclic proof systems for **FP** and **FELEMENTARY**
4. Non-wellfounded proof system for **FP**/**poly**

Implicit computational complexity (ICC)

- Implicit computational complexity (ICC) = characterise complexity classes by means of languages/calculi **without** explicit reference to machine models or external resource bounds.
- Originates in the 90's with the Bellantoni and Cook's paper on *safe recursion*.
- Pervasive notion of **stratification**: data are organized into *strata* (Bellantoni's safe recursion [Bellantoni and Cook 92], Leivant's predicative/ramified/tiered recursion [Leivant 95]).

Safe recursion on notation

- Function algebra B characterising **FP** [Bellantoni and Cook 92].
- Two successors: $s_0 x = 2x$ and $s_1 x = 2x + 1$.
- Function arguments partitioned into normal and safe:

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

- 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:

Safe recursion on notation

- Function algebra B characterising **FP** [Bellantoni and Cook 92].

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

- Function arguments partitioned into normal and safe:

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

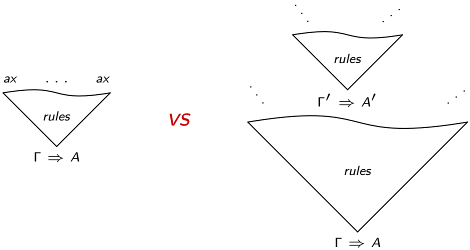
- Safe recursion on notation:

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

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

Non-wellfounded 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!

$$\mathsf{cut}\,\dfrac{\mathsf{cut}\,\dfrac{\begin{matrix}\vdots \\ \Rightarrow A\end{matrix} \qquad \mathsf{id}\,\dfrac{}{A \Rightarrow A}}{\Rightarrow A} \qquad \mathsf{id}\,\dfrac{}{A \Rightarrow A}}{\Rightarrow A}$$

- Progressiveness condition = global condition to guarantee consistency.

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

$$
\cfrac{\cfrac{\begin{array}{c}\vdots\\ \Rightarrow A\end{array} \qquad \cfrac{}{A \Rightarrow A}\,\mathsf{id}}{\Rightarrow A}\,\mathsf{cut} \qquad \cfrac{}{A \Rightarrow A}\,\mathsf{id}}{\Rightarrow A}\,\mathsf{cut}
$$

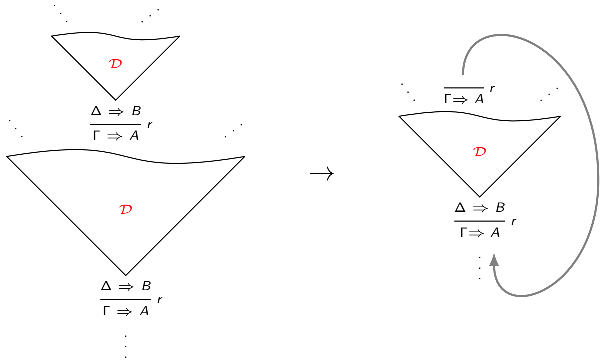
- Progressiveness condition $=$ global condition to guarantee consistency.

Cyclic proofs

- Cyclic proofs = **regular** non-wellfounded proofs
- Regular tree = only **finitely** many distinct subtrees
- Cyclic proofs admit a finite, “circular” presentation.

Cyclic proofs

- Cyclic proofs = **regular** non-wellfounded proofs
- Regular tree = only **finitely** many distinct subtrees
- Cyclic proofs admit a finite, "circular" presentation.



Cyclic proofs as programs

- Only one **formula** N corresponding to $\mathbb{N}$
- **Inference rules** correspond to algorithmic instructions
- A **cyclic proof**

$$\begin{array}{c} \mathcal{D} \\ \underbrace{N, \ldots, N}_{n} \Rightarrow N \end{array}$$

corresponds to a programs computing a number-theoretic function

$$f_{\mathcal{D}} : \underbrace{\mathbb{N} \times \ldots \times \mathbb{N}}_{n} \to \mathbb{N}$$

Cyclic proofs as programs

- Only one **formula** N corresponding to $\mathbb{N}$

- **Inference rules** correspond to algorithmic instructions

- A **cyclic proof**

$$\begin{array}{c} \mathcal{D} \\ \underbrace{N, \ldots, N}_{n} \Rightarrow N \end{array}$$

  corresponds to a programs computing a number-theoretic function

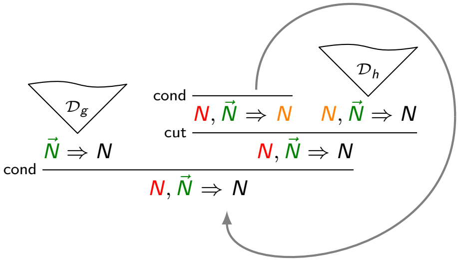
$$f_{\mathcal{D}} : \underbrace{\mathbb{N} \times \ldots \times \mathbb{N}}_{n} \to \mathbb{N}$$

An example: primitive recursion

- **Example**: the following primitive recursive definition of f

$$
\begin{aligned}
f(0, \vec{y}) &= g(\vec{y}) \\
f(x+1, \vec{y}) &= h(x, \vec{y}, f(x, \vec{y}))
\end{aligned}
$$

can be represented by



Cyclic proofs as polytime programs

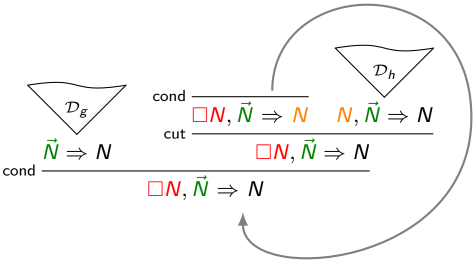
- The characterisation of **FP** is achieved in two steps:
  - introduce **modalities** $\Box N$ vs N reflecting safe/normal distinction of parameters
  - add **global proof-theoretic conditions** that induce polytime termination

- Example: safe recursion [Bellantoni&Cook, 1992]

Cyclic proofs as polytime programs

- The characterisation of **FP** is achieved in two steps:
  - introduce **modalities** $\Box N$ vs N reflecting safe/normal distinction of parameters
  - add **global proof-theoretic conditions** that induce polytime termination

- Example: safe recursion [Bellantoni&Cook, 1992]



Characterising the polynomial time (**FP**)

- Cyclic proof system CB = non-wellfounded proofs that satisfy the following global proof-theoretic conditions:
  - **Regularity** $\rightarrow$ uniformity, computability
  - **Progressiveness** $\rightarrow$ totality, termination
  - **Safety** $\rightarrow$ maintain globally the safe/normal distinction
  - **Left-leaning** $\rightarrow$ prevent nested recursion (source of exponential blow up)

- **Theorem** [Curzi&Das, 2022(a)]:
  - the functions representable in CB are exactly those in **FP**.
  - the functions representable in CB **without the left-leaning condition** are exactly those in **FELEMENTARY**.

Characterising the polynomial time (**FP**)

- Cyclic proof system CB = non-wellfounded proofs that satisfy the following global proof-theoretic conditions:
  - **Regularity** $\rightarrow$ uniformity, computability
  - **Progressiveness** $\rightarrow$ totality, termination
  - **Safety** $\rightarrow$ maintain globally the safe/normal distinction
  - **Left-leaning** $\rightarrow$ prevent nested recursion (source of exponential blow up)

- **Theorem** [Curzi&Das, 2022(a)]:
  - the functions representable in CB are exactly those in **FP**.
  - the functions representable in CB **without the left-leaning condition** are exactly those in **FELEMENTARY**.

Characterising the polynomial time (**FP**)

- Cyclic proof system CB = non-wellfounded proofs that satisfy the following global proof-theoretic conditions:
  - **Regularity** $\rightarrow$ uniformity, computability
  - **Progressiveness** $\rightarrow$ totality, termination
  - **Safety** $\rightarrow$ maintain globally the safe/normal distinction
  - **Left-leaning** $\rightarrow$ prevent nested recursion (source of exponential blow up)

- **Theorem** [Curzi&Das, 2022(a)]:
  - the functions representable in CB are exactly those in **FP**.
  - the functions representable in CB **without the left-leaning condition** are exactly those in **FELEMENTARY**.

Characterising the polynomial time (**FP**)

- Cyclic proof system CB = non-wellfounded proofs that satisfy the following global proof-theoretic conditions:
  - **Regularity** $\rightarrow$ uniformity, computability
  - **Progressiveness** $\rightarrow$ totality, termination
  - **Safety** $\rightarrow$ maintain globally the safe/normal distinction
  - **Left-leaning** $\rightarrow$ prevent nested recursion (source of exponential blow up)

- **Theorem** [Curzi&Das, 2022(a)]:
  - the functions representable in CB are exactly those in **FP**.
  - the functions representable in CB **without the left-leaning condition** are exactly those in **FELEMENTARY**.

Characterising the polynomial time (**FP**)

- Cyclic proof system CB = non-wellfounded proofs that satisfy the following global proof-theoretic conditions:
  - **Regularity** $\rightarrow$ uniformity, computability
  - **Progressiveness** $\rightarrow$ totality, termination
  - **Safety** $\rightarrow$ maintain globally the safe/normal distinction
  - **Left-leaning** $\rightarrow$ prevent nested recursion (source of exponential blow up)

- **Theorem** [Curzi&Das, 2022(a)]:
  - the functions representable in CB are exactly those in **FP**.
  - the functions representable in CB **without the left-leaning condition** are exactly those in **FELEMENTARY**.

Non-uniform polynomial time (**FP**/**poly**)

- **FP**/**poly** = class of functions computable in non-uniform polynomial time by a Turing machine

- **Theorem**: $f \in$ **FP**/**poly** iff there are polynomial size circuits computing f.

- **FP**($\mathbb{R}$) = class of functions computable in polynomial time by a Turing machine "querying bits of real numbers"

- **Theorem** [Folklore]: **FP**/**poly** = **FP**($\mathbb{R}$).

Non-uniform polynomial time ($\mathbf{FP}/\mathbf{poly}$)

- $\mathbf{FP}/\mathbf{poly}$ = class of functions computable in non-uniform polynomial time by a Turing machine
- **Theorem**: $f \in \mathbf{FP}/\mathbf{poly}$ iff there are polynomial size circuits computing f.
- $\mathbf{FP}(\mathbb{R})$ = class of functions computable in polynomial time by a Turing machine "querying bits of real numbers"
- **Theorem** [Folklore]: $\mathbf{FP}/\mathbf{poly} = \mathbf{FP}(\mathbb{R})$.

Non-wellfounded proofs as non-uniform polytime programs

- Cyclic proofs = **regular** non-wellfounded proofs

- Regular tree = only finitely many distinct subtrees

$$\boxed{\textit{regularity} \quad \approx \quad \textit{computability}}$$

- **Idea**: relaxing regularity to represent real numbers and characterise $\mathbf{FP}(\mathbb{R})$

$$\boxed{\textbf{weak } \textit{regularity} \quad \approx \quad \textit{computability} \; + \; \textbf{query on bits of real numbers}}$$

Non-wellfounded proofs as non-uniform polytime programs

- Cyclic proofs = **regular** non-wellfounded proofs
- Regular tree = only finitely many distinct subtrees

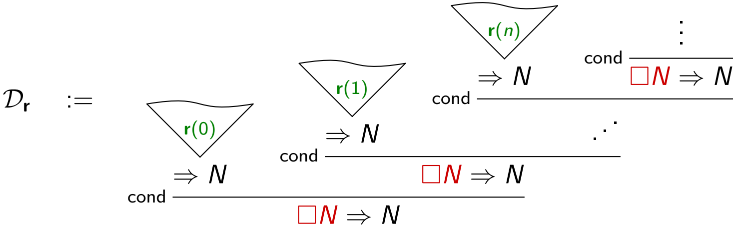
$$\boxed{\textit{regularity} \quad \approx \quad \textit{computability}}$$

- **Idea**: relaxing regularity to represent real numbers and characterise $\mathbf{FP}(\mathbb{R})$

$$\boxed{\textbf{weak } \textit{regularity} \quad \approx \quad \textit{computability} \; + \; \textbf{query on bits of real numbers}}$$

Weak regularity

- Example: representing a real number $\mathbf{r} = (\mathbf{r}(0), \mathbf{r}(1), \dots, \mathbf{r}(n), \dots)$ with non-wellfounded proofs.

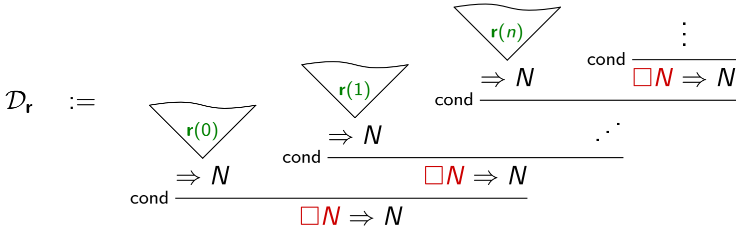


- Weakly regular proof = only finitely many distinct subproofs **containing certain inference rules**.

- **Idea:** at some point any infinite branch either "loops" or it contains the root of some $\mathcal{D}_{\mathbf{r}}$.

Weak regularity

- Example: representing a real number $\mathbf{r} = (\mathbf{r}(0), \mathbf{r}(1), \dots, \mathbf{r}(n), \dots)$ with non-wellfounded proofs.



- Weakly regular proof = only finitely many distinct subproofs **containing certain inference rules**.

- **Idea:** at some point any infinite branch either "loops" or it contains the root of some $\mathcal{D}_{\mathbf{r}}$.

Characterising **FP/poly**

- Non-wellfounded proof system nuB = weakly regular version of CB.

- **Theorem** [Curzi&Das, 2022(b)]: The functions representable in nuB are exactly those in **FP/poly**.

- **Idea of the proof**:
  - We show that nuB = CB($\mathbb{R}$), where

$$\mathrm{CB}(\mathbb{R}) := \mathrm{CB} + \left\{ \mathsf{r}\,\frac{}{\Box N \Rightarrow N} \right\}_{\mathsf{r} \in \mathbb{R}}$$

  - From CB = **FP** we infer **FP**($\mathbb{R}$) = CB($\mathbb{R}$).

  - We conclude nuB = CB($\mathbb{R}$) = **FP**($\mathbb{R}$) = **FP/poly**.

Characterising **FP/poly**

- Non-wellfounded proof system nuB = weakly regular version of CB.

- **Theorem** [Curzi&Das, 2022(b)]: The functions representable in nuB are exactly those in **FP/poly**.

- **Idea of the proof**:
  - We show that nuB = CB($\mathbb{R}$), where

$$\mathrm{CB}(\mathbb{R}) := \mathrm{CB} + \left\{ \mathbf{r}\,\frac{}{\Box N \Rightarrow N} \right\}_{\mathbf{r}\in\mathbb{R}}$$

  - From CB = **FP** we infer **FP**($\mathbb{R}$) = CB($\mathbb{R}$).

  - We conclude nuB = CB($\mathbb{R}$) = **FP**($\mathbb{R}$) = **FP/poly**.

Thank you!
Questions?

Appendix

Non-uniform complexity classes

- **FP** = class of functions computable in polynomial time on a Turing machine.
- **FP**/**poly** is an extension of **FP** that intuitively has access to a 'small' amount of *advice*, determined only by the length of the input.
- **FP**/**poly** = class of functions $f(\vec{x})$ for which there exists some strings $\alpha_{\vec{n}} \in \{0,1\}^*$ and a function $f'(x, \vec{x}) \in \mathbf{FP}$ with:
  - $|\alpha_{\vec{n}}|$ is polynomial in $\vec{n}$.
  - $f(\vec{x}) = f'(\alpha_{|\vec{x}|}, \vec{x})$.
- Note, in particular, that **FP**/**poly** admits undecidable problems. E.g. the function $f(x) = 1$ just if $|x|$ is the code of a halting Turing machine (and 0 otherwise) is in **FP**/**poly**. Indeed, the point of the class **FP**/**poly** is to rather characterise a more non-uniform notion of computation.
- **Theorem**: $f(\vec{x}) \in \mathbf{FP}/\mathbf{poly}$ iff there are poly-size circuits computing $f(\vec{x})$.

- The class **FP**($\mathbb{R}$) consists of just the functions computable in polynomial time by a Turing machine with access to oracles from:

$$\mathbb{R} := \{f(x) : \mathbb{N} \to \{0,1\} \mid |x| = |y| \implies f(x) = f(y)\}$$

- Note that the notation $\mathbb{R}$ is suggestive here, since its elements are essentially maps from lengths/positions to Booleans, and so may be identified with Boolean streams.

- **Theorem** [Folklore]: $\mathbf{FP}/\mathbf{poly} = \mathbf{FP}(\mathbb{R})$.

Rules for the non-wellfounded proof system nuB

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

$$\mathsf{w}_N\ \frac{\Gamma \Rightarrow B}{\Gamma, N \Rightarrow B} \qquad \mathsf{w}_\Box\ \frac{\Gamma \Rightarrow B}{\Box N, \Gamma \Rightarrow B} \qquad \mathsf{e}\ \frac{\Gamma, A, B, \Gamma' \Rightarrow C}{\Gamma, B, A, \Gamma' \Rightarrow C} \qquad \Box_l\ \frac{\Gamma, N \Rightarrow A}{\Box N, \Gamma \Rightarrow A} \qquad \Box_r\ \frac{\Box\Gamma \Rightarrow N}{\Box\Gamma \Rightarrow \Box N}$$

$$0\ \frac{}{\Rightarrow N} \qquad 1\ \frac{}{\Rightarrow N} \qquad \mathsf{s}_0\ \frac{\Gamma \Rightarrow A}{\Gamma \Rightarrow A} \qquad \mathsf{s}_1\ \frac{\Gamma \Rightarrow A}{\Gamma \Rightarrow A} \qquad \mathsf{srec}\ \frac{\Gamma \Rightarrow N \quad \Box N, \Gamma, N \Rightarrow N \quad \Box N, \Gamma, N \Rightarrow N}{\Box N, \Gamma \Rightarrow N}$$

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

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

Semantics of non-wellfounded proofs for nuB

$$i\ \frac{i \in \{0,1\}}{\Rightarrow N} \qquad\qquad f_{\mathcal{D}}(;) := i$$

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

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

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

$$\mathsf{cond}_{\Box}\ \frac{\overset{\mathcal{D}_0}{\Gamma \Rightarrow N} \quad \overset{\mathcal{D}_1}{\Box N, \Gamma, \Rightarrow N} \quad \overset{\mathcal{D}_2}{\Box N, \Gamma \Rightarrow N}}{\Box N, \Gamma \Rightarrow N} \qquad\qquad \begin{array}{rcl} f_{\mathcal{D}}(0, \vec{x};\vec{y}) & := & f_{\mathcal{D}_0}(\vec{x};\vec{y}) \\ f_{\mathcal{D}}(\mathsf{s}_0 x, \vec{x};\vec{y}) & := & f_{\mathcal{D}_1}(x, \vec{x};\vec{y}) \\ f_{\mathcal{D}}(\mathsf{s}_1 x, \vec{x};\vec{y}) & := & f_{\mathcal{D}_2}(x, \vec{x};\vec{y}) \end{array}$$

$$|\mathsf{cond}|_{\Box}\ \frac{\overset{\mathcal{D}_0}{\Gamma \Rightarrow N} \quad \overset{\mathcal{D}_2}{\Box N, \Gamma \Rightarrow N}}{\Box N, \Gamma \Rightarrow N} \qquad\qquad \begin{array}{rcl} f_{\mathcal{D}}(0, \vec{x};\vec{y}) & := & f_{\mathcal{D}_0}(\vec{x};\vec{y}) \\ f_{\mathcal{D}}(\mathsf{s}_i x, \vec{x};\vec{y}) & := & f_{\mathcal{D}_2}(x, \vec{x};\vec{y}) \end{array}$$

Progressiveness

- **Example.** A cyclic proof $\mathcal{D}$ representing a partial function:

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

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

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

- Progressiveness $\sim$ **totality**

Safety condition

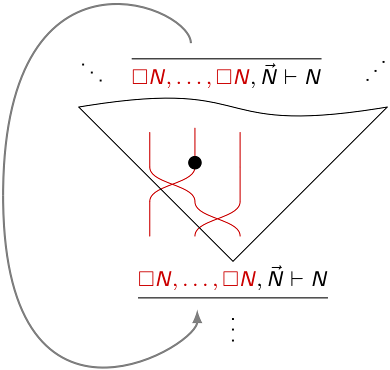
- **Example.** Modalities are not enough to enforce stratification in our setting. E.g. cyclic progressing proof $\mathcal{D}$ for primitive recursion (on notation):

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

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

- Safe proof = any infinite branch crosses finitely many $\mathrm{cut}_\Box$ rules.
- Safety condition rules out non-safe recursion schemes.

Safety condition induces a simpler $\Box$-thread structure

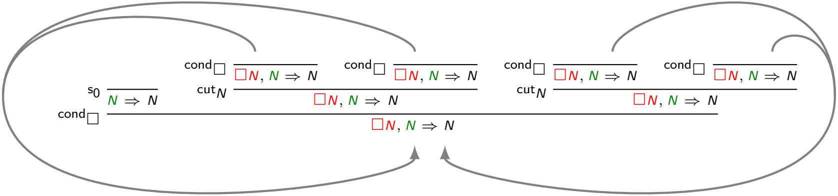


$$\mathsf{w}_{\Box}\ \frac{\Gamma \Rightarrow B}{\Box N, \Gamma \Rightarrow B} \qquad \Box_l\ \frac{\Gamma, N \Rightarrow A}{\Box N, \Gamma \Rightarrow A} \qquad \mathsf{cut}_{\Box}\ \frac{\Gamma \Rightarrow \Box N \qquad \Box N, \Gamma \Rightarrow B}{\Gamma \Rightarrow B} \qquad \mathsf{cond}_{\Box}\ \frac{\Gamma \Rightarrow N \qquad \Box N, \Gamma \Rightarrow N \qquad \Box N, \Gamma \Rightarrow N}{\Box N, \Gamma \Rightarrow N}$$

Left-leaning condition

- Safety condition is not enough! We can express **nested safe recursion**.

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



$$\exp(0; y) = \mathrm{s}_0 y$$
$$\exp(\mathrm{s}_i x; y) = \exp(x; \exp(x; y))$$

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

Hofmann's type system SLR [Hofmann 97]

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

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

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

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

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

- terms $t : (\square N)^n \to N^m \multimap N$ represent *exactly* the functions in **FP**.

Nesting and higher-order recursion

- 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].

- **Left-leaning is a linearity condition**: it prevents duplication of recursive calls, and hence their nesting.