Correctness of Second Order Multiplicative Linear Logic

Adrien Ragot [1,2]
Lorenzo Tortora de Falco [2]
Thomas Seiller [1]

[1]: Université Sorbonne Paris Nord. LIPN
[2]: Università Degli Studi Roma tre

AILA XXVIII

Udine, September 2024

PLAN

PLAN

1 - Correctness: Mathematics & Computer Science analogies

PLAN

1 - Correctness: Mathematics & Computer Science analogies

2 - Parsing criterion : Recipe

PLAN

1 - Correctness: Mathematics & Computer Science analogies

2 - Parsing criterion : Recipe

3 - Obtaining more criterions

I. Correctness

Mathematics x Computer Science

Correctness

Correctness

in Mathematics

Correctness

in Mathematics

Proofs

Correctness

in Mathematics

Proofs

is my proof Correct ?

Correctness

in Mathematics

in Computer Science

Proofs

is my proof Correct ?

Correctness

in Mathematics

Proofs

is my proof Correct ?

in Computer Science

Programs

Correctness

in Mathematics

Proofs

is my proof Correct ?

in Computer Science

Programs

is my program Correct ?

Correctness

in Mathematics

Proofs

(of a proposition A)

is my proof Correct ?

in Computer Science

Programs

is my program Correct ?

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?

in Computer Science

Programs (of type A)

is my program Correct ?

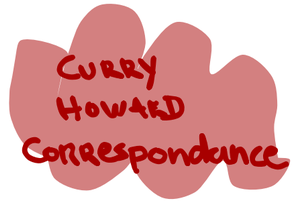
Correctness

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?



in Computer Science

Programs (of type A)

is my program Correct ?

(dialogical logic / Game Semantics)

Does my proof convinces any opponent

Correctness

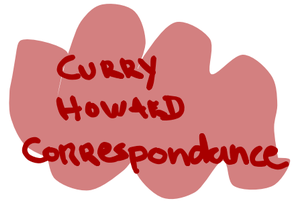
in Mathematics

Proofs (of a proposition A)

is my proof Correct ?

(dialogical logic / Game Semantics)

Does my proof convinces any opponent



in Computer Science

Programs (of type A)

is my program Correct ?

My program behaves as expected

in an evaluation context

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?

Curry Howard Correspondance

in Computer Science

Programs (of type A)

is my program Correct ?

(dialogical logic / Game Semantics)

Does my proof convinces any opponent

colleague

Realisability

My program behaves as expected

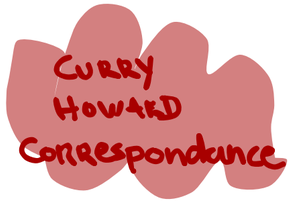
in any evaluation context

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?



in Computer Science

Programs (of type A)

is my program Correct ?

(dialogical logic / Game Semantics)

Does my proof convinces any opponent

colleague

↓

= correct =

Realisability

My program behaves as expected

in any evaluation context

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?

CURRY HOWARD Correspondance

in Computer Science

Programs (of type A)

is my program Correct ?

(dialogical logic / Game Semantics)

Does my proof convinces any opponent

colleague

↓

= correct =

Realisability

My program behaves as expected

in any evaluation context

other programs

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?

CURRY HOWARD Correspondance

in Computer Science

Programs (of type A)

is my program Correct ?

(dialogical logic / Game Semantics)

Does my proof convinces any opponent

colleague

↓

« correct »

Realisability

My program behaves as expected

in any evaluation context

other programs

↓

« correct »

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?

CURRY HOWARD Correspondance

in Computer Science

Programs (of type A)

is my program Correct ?

(dialogical logic / Game Semantics)

Does my proof convinces any opponent

colleague

↓

« correct »

My program behaves as expected (Realisability)

in any evaluation context

other programs

↓

« correct »

infinitely many

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?

Curry Howard Correspondance

in Computer Science

Programs (of type A)

is my program Correct ?

(dialogical logic / Game Semantics)

Does my proof convinces any opponent

colleague

↓

« correct »

My program behaves as expected (Realisability)

in any evaluation context

other programs

↓

« correct »

infinitely many

↓

Naively undecidable

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?

CURRY HOWARD Correspondance

in Computer Science

Programs (of type A)

is my program Correct ?

(dialogical logic / Game Semantics)

Does my proof convinces any opponent

colleague → - correct -

(in)finitely many?

Realisability

My program behaves as expected in any evaluation context

other programs → - correct -

infinitely many → Naively undecidable

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?

Curry Howard Correspondance

in Computer Science

Programs (of type A)

is my program Correct ?

(dialogical logic / Game Semantics)

Does my proof convinces any opponent

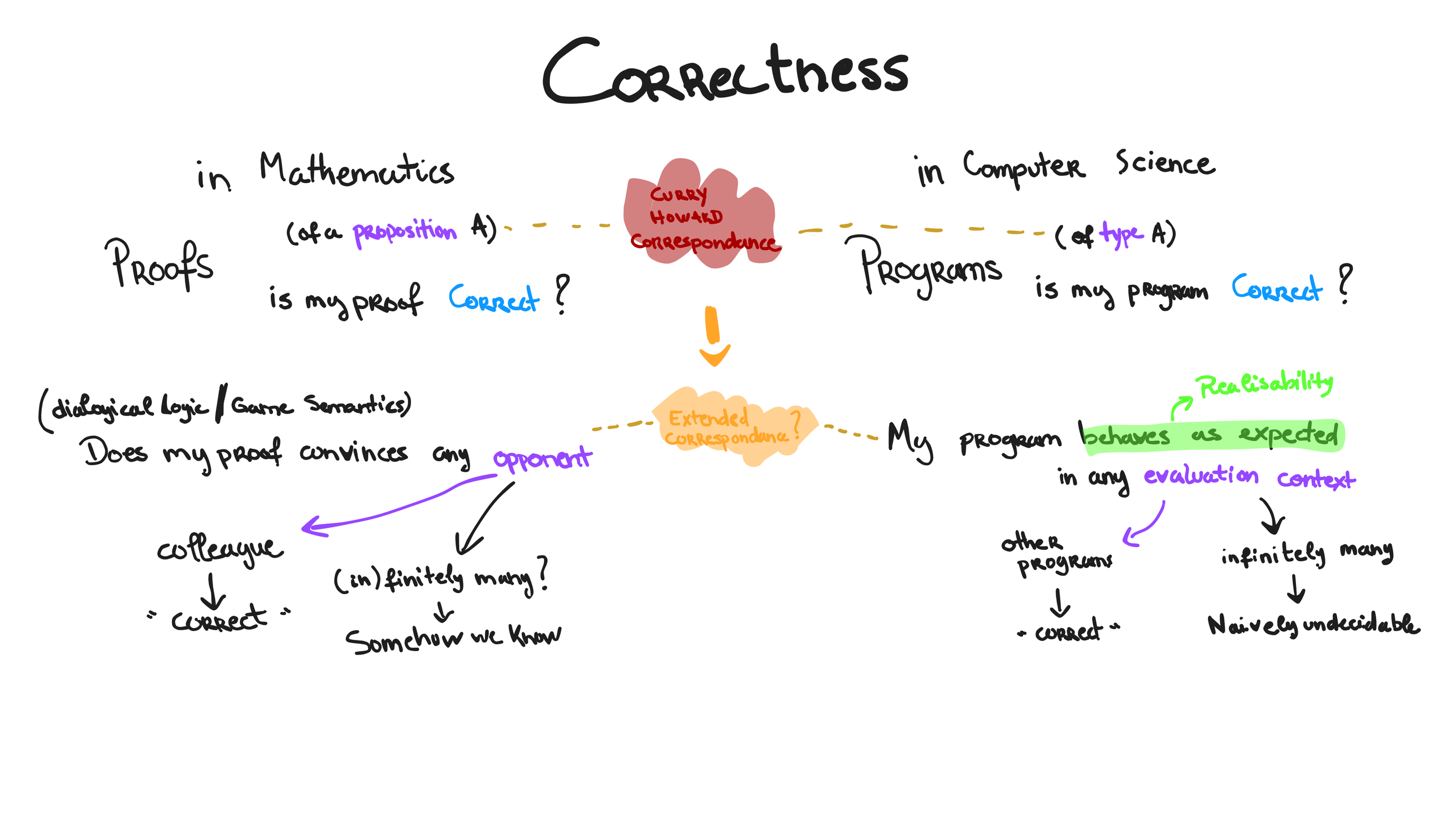
- colleague → - correct -
- (in)finitely many? → Somehow we know

My program behaves as expected (Realisability)

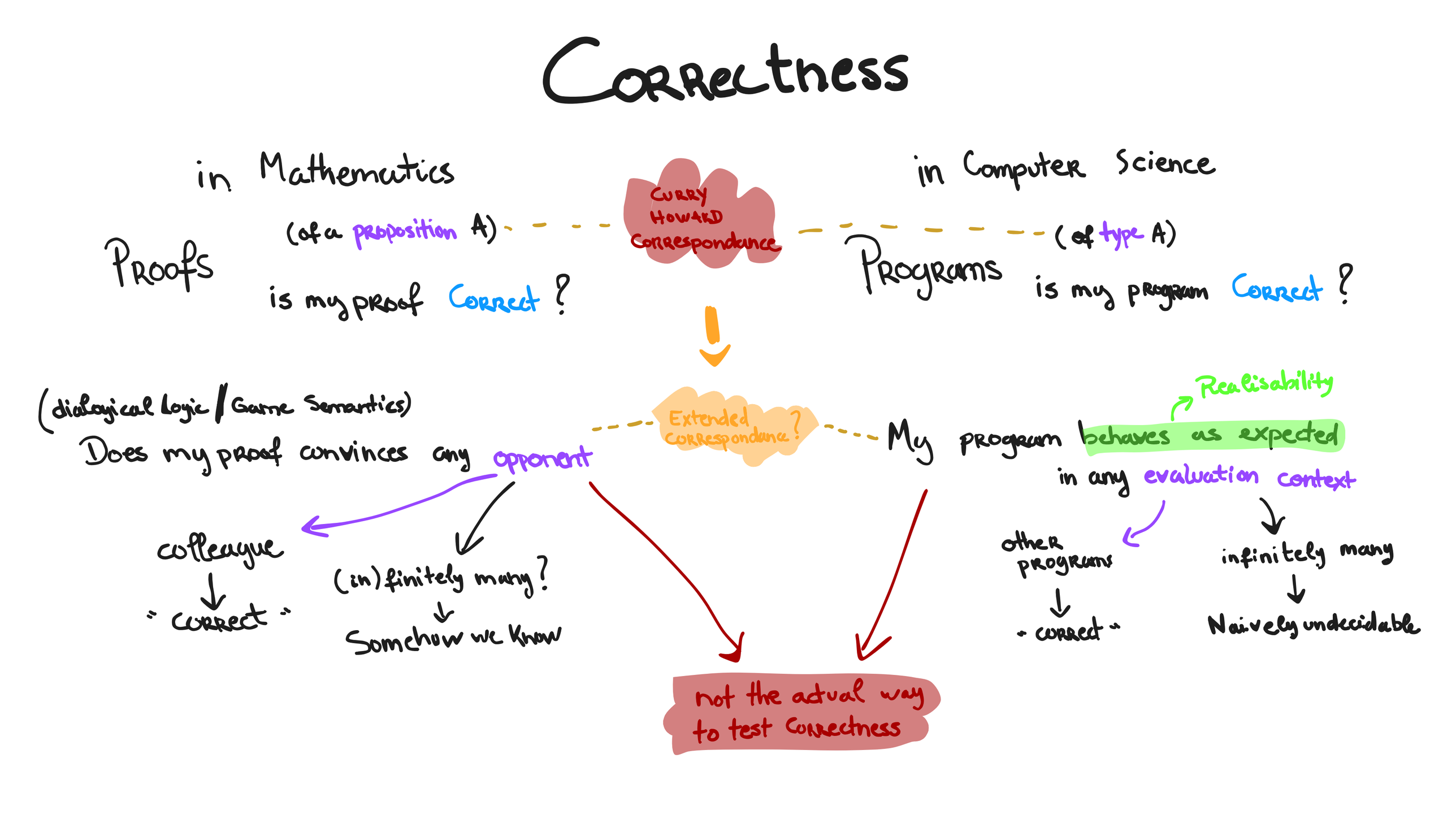
in any evaluation context

- other programs → - correct -
- infinitely many → Naively undecidable

Correctness



Correctness

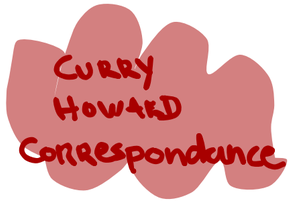


Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?



in Computer Science

Programs (of type A)

is my program Correct ?

Correctness

in Mathematics

Proofs (of a proposition A) - - - - CURRY HOWARD CORRESPONDANCE - - - - - - - - - - (of type A) Programs

in Computer Science

is my proof Correct ?

is my program Correct ?

Example (inclusion)

A ::= " if n is an integer, n is a real "

Correctness

in Mathematics

Proofs (of a proposition A) - - - - CURRY HOWARD CORRESPONDANCE - - - - - - - - - - (of type A) Programs

in Computer Science

is my proof Correct ?

is my program Correct ?

Example (inclusion)

A ::= " if n is an integer, n is a real "

$P_1 : A$ ::= let $n \in \mathbb{N}$, since $\mathbb{N} \subseteq \mathbb{R}$
one concludes

Correctness

in Mathematics

Proofs (of a proposition A) - - - - CURRY HOWARD CORRESPONDANCE - - - - - - - - - - (of type A) Programs

in Computer Science

Proofs: is my proof Correct ?

Programs: is my program Correct ?

Example (inclusion)

A ::= " if n is an integer, n is a Real "

P_1 : A ::= let $n \in \mathbb{N}$, since $\mathbb{N} \subseteq \mathbb{R}$ one concludes

P_2 : A ::= Say $n = 1$ because $1 \in \mathbb{R}$ then $n \in \mathbb{R}$.
We conclude that any integer belongs to $\mathbb{R}$.

Example (Cast)

A ::= INPUT int n
OUTPUT float n (corresponding to the integer in input)

Correctness

in Mathematics

(of a proposition A)

Proofs

is my proof Correct ?

Curry Howard Correspondance

in Computer Science

(of type A)

Programs

is my program Correct ?

Example (inclusion)

A ::= " if n is an integer, n is a Real "

$P_1 : A ::=$ let $n \in \mathbb{N}$, since $\mathbb{N} \subseteq \mathbb{R}$ one concludes

$P_2 : A ::=$ Say $n = 1$ because $1 \in \mathbb{R}$ then $n \in \mathbb{R}$.
We conclude that any integer belongs to $\mathbb{R}$.

Example (Cast)

A ::= INPUT int n
OUTPUT float n (corresponding to the integer in input)

$P_1 : A ::=$ function $x \rightarrow$ float_of_int (x)

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?

Curry Howard Correspondance

in Computer Science

Programs (of type A)

is my program Correct ?

Example (inclusion)

A ::= " if n is an integer, n is a Real "

P_1 : A ::= let $n \in \mathbb{N}$, since $\mathbb{N} \subseteq \mathbb{R}$ one concludes

correct

P_2 : A ::= Say $n = 1$ because $1 \in \mathbb{R}$ then $n \in \mathbb{R}$.
We conclude that any integer belongs to $\mathbb{R}$.

Example (Cast)

A ::= INPUT int n
OUTPUT float n (corresponding to the integer in input)

P_1 : A ::= function x → float_of_int (x)

P_2 : A ::= function 1 → float_of_int (1)

Correctness

in Mathematics

(of a proposition A)

Proofs

is my proof Correct ?

Curry Howard Correspondance

in Computer Science

(of type A)

Programs

is my program Correct ?

Example (inclusion)

A ::= " if n is an integer, n is a Real "

$P_1 : A ::=$ let $n \in \mathbb{N}$, since $\mathbb{N} \subseteq \mathbb{R}$ one concludes — correct

$P_2 : A ::=$ Say $n = 1$ because $1 \in \mathbb{R}$ then $n \in \mathbb{R}$. We conclude that any integer belongs to $\mathbb{R}$. — incorrect (partial)

Example (Cast)

A ::= INPUT int n
OUTPUT float n (corresponding to the integer in input)

$P_1 : A ::=$ function x → float_of_int (x)

$P_2 : A ::=$ function 1 → float_of_int (1)

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?

CURRY HOWARD Correspondance

in Computer Science

Programs (of type A)

is my program Correct ?

Example (inclusion)

A ::= " if n is an integer, n is a Real "

$P_1 : A ::=$ let $n \in \mathbb{N}$, since $\mathbb{N} \subseteq \mathbb{R}$ one concludes — correct

$P_2 : A ::=$ Say $n = 1$ because $1 \in \mathbb{R}$ then $n \in \mathbb{R}$. We conclude that any integer belongs to $\mathbb{R}$. — incorrect (partial)

↳ P_2 behaves as expected when given 1 as input, i.e. in the context $\lambda f . (f)1 \sim (\bullet)1$

Example (Cast)

A ::= INPUT int n
OUTPUT float n (corresponding to the integer in input)

$P_1 : A ::=$ function x → float_of_int (x)

$P_2 : A ::=$ function 1 → float_of_int (1)

Correctness

in Mathematics

Proofs (of a proposition A)

is my proof Correct ?

Curry Howard Correspondance

in Computer Science

Programs (of type A)

is my program Correct ?

Example (inclusion)

A ::= " if n is an integer, n is a Real "

$P_1 : A ::=$ let $n \in \mathbb{N}$, since $\mathbb{N} \subseteq \mathbb{R}$ one concludes — correct

$P_2 : A ::=$ Say $n = 1$ because $1 \in \mathbb{R}$ then $n \in \mathbb{R}$. We conclude that any integer belongs to $\mathbb{R}$. — incorrect (partial)

↳ P_2 behaves as expected when given 1 as input, i.e. in the context $\lambda f.\,(f)\,1$ $\tilde{}$ $(\bullet)\,1$

P_2 fails against $\lambda f.\,(f)\,2$

Example (Cast)

A ::= INPUT int n
OUTPUT float n (corresponding to the integer in input)

$P_1 : A ::=$ function x → float_of_int (x)

$P_2 : A ::=$ function 1 → float_of_int (1)

Incorrectness

in Mathematics

Mathematicians want to eliminate incorrectness

in Computer Science

Computer scientists / Programmers are used to incorrectness (BUGS)

In Correctness

in Mathematics

Mathematicians want to eliminate incorrectness

in Computer Science

Computer Scientists / Programmers are used to incorrectness (BUGS)

(• In practice, programmers seek to eliminate bugs.)

Incorrectness

in Mathematics

Mathematicians want to eliminate incorrectness

(• In practice, mathematicians produce a lot of incorrectness)

in Computer Science

Computer scientists / Programmers are used to incorrectness (BUGS)

(• In practice, programmers seek to eliminate bugs.)

Incorrectness

in Mathematics

Mathematicians want to eliminate incorrectness

(• In practice, mathematicians produce a lot of incorrectness)

in Computer Science

Computer scientists / Programmers are used to incorrectness (BUGS)

(• In practice, programmers seek to eliminate bugs.)

• Infinite loops e.g. $\lambda x.(x)x$

Incorrectness

in Mathematics

Mathematicians want to eliminate incorrectness

(• In practice, mathematicians produce a lot of incorrectness)

• No corresponding proof attempt

in Computer Science

Computer scientists / Programmers are used to incorrectness (BUGS)

(• In practice, programmers seek to eliminate bugs.)

• Infinite loops e.g. $\lambda x.(x)x$

Incorrectness

in Mathematics

Mathematicians want to eliminate incorrectness

(• In practice, mathematicians produce a lot of incorrectness)

• No corresponding proof attempt

Formalisms:
- Hilbert System, Natural Deduction, Sequent Calculus, ...
- proof = tree

in Computer Science

Computer scientists / Programmers are used to incorrectness (BUGS)

(• In practice, programmers seek to eliminate bugs.)

• Infinite loops e.g. $\lambda x.(x)x$

Incorrectness

in Mathematics

Mathematicians want to eliminate incorrectness

(• In practice, mathematicians produce a lot of incorrectness)

• No corresponding proof attempt

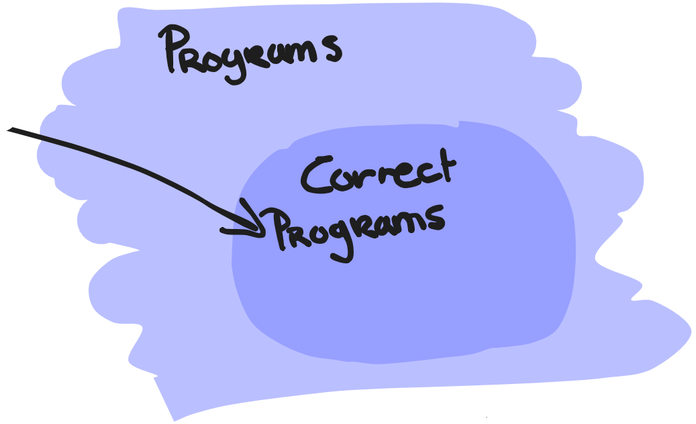
Formalisms:
- Hilbert System, Natural Deduction, Sequent Calculus, ...
- proof = tree

in Computer Science

Computer scientists / Programmers are used to incorrectness (BUGS)

(• In practice, programmers seek to eliminate bugs.)

• Infinite loops e.g. $\lambda x.(x)x$



Incorrectness

in Mathematics

Mathematicians want to eliminate incorrectness

(• In practice, mathematicians produce a lot of incorrectness)

• No corresponding proof attempt

in Computer Science

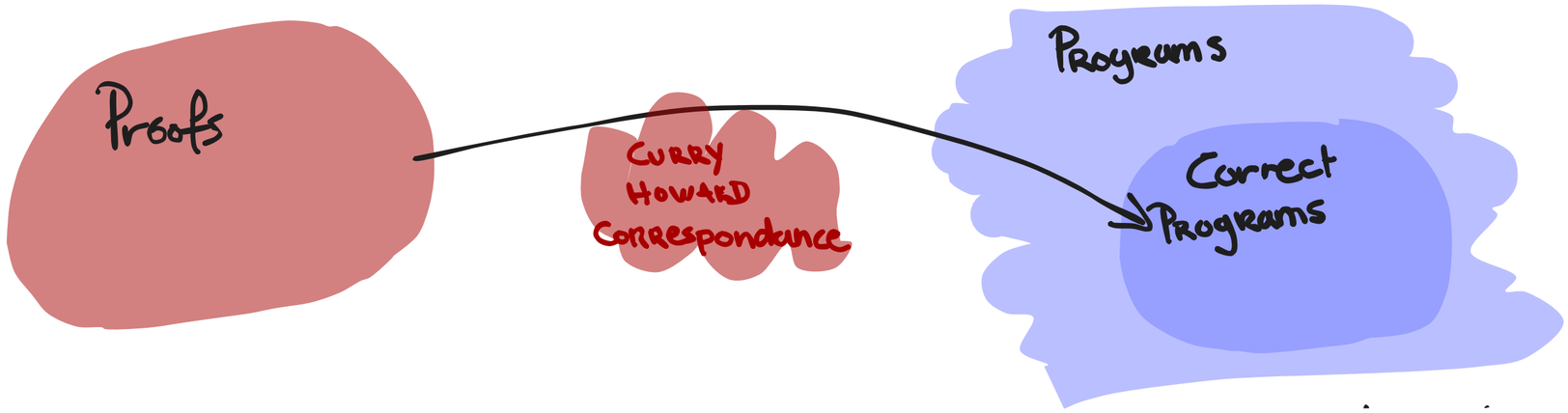
Computer scientists / Programmers are used to incorrectness (BUGS)

(• In practice, programmers seek to eliminate bugs.)

• Infinite loops e.g. $\lambda x.(x)x$

Formalisms:

- Hilbert System, Natural Deduction, Sequent Calculus, ...
- proof = tree



CORRECT-BY-DESIGN

In Correctness

in Mathematics

Mathematicians want to eliminate incorrectness

(• In practice, mathematicians produce a lot of incorrectness)

• No corresponding proof attempt

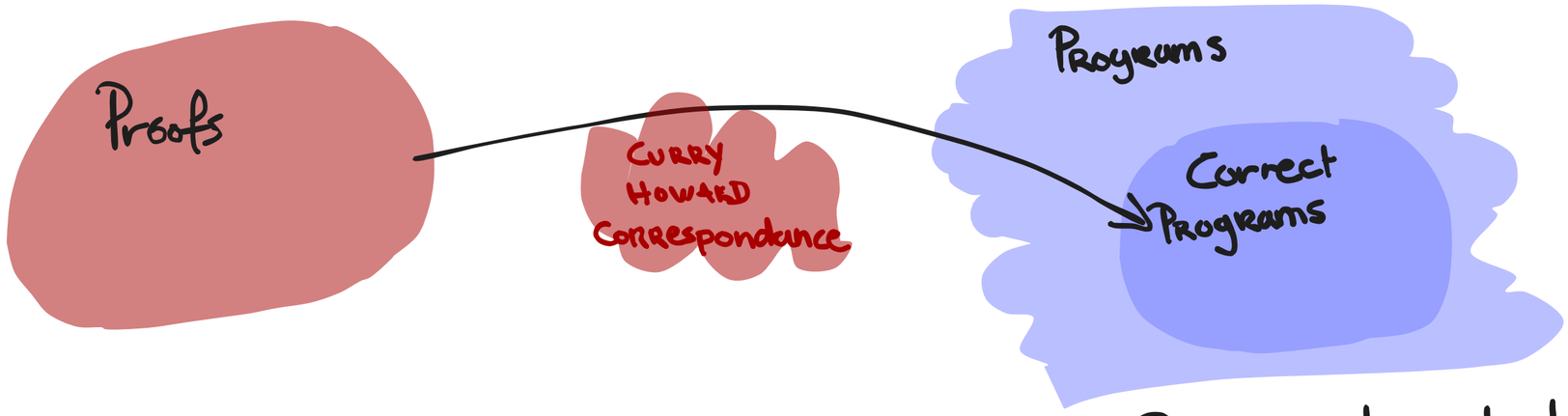
in Computer Science

Computer scientists / Programmers are used to incorrectness (BUGS)

(• In practice, programmers seek to eliminate bugs.)

• Infinite loops e.g. $\lambda x.(x)x$

Formalisms:
- Hilbert System, Natural Deduction, Sequent Calculus, ...
- proof = tree



CORRECT-BY-DESIGN

Correct Programs have to be identified
↳ typing inference, Realisability, semantics, ...

In Correctness

in Mathematics

Mathematicians want to eliminate incorrectness

(• In practice, mathematicians produce a lot of incorrectness)

• No corresponding proof attempt

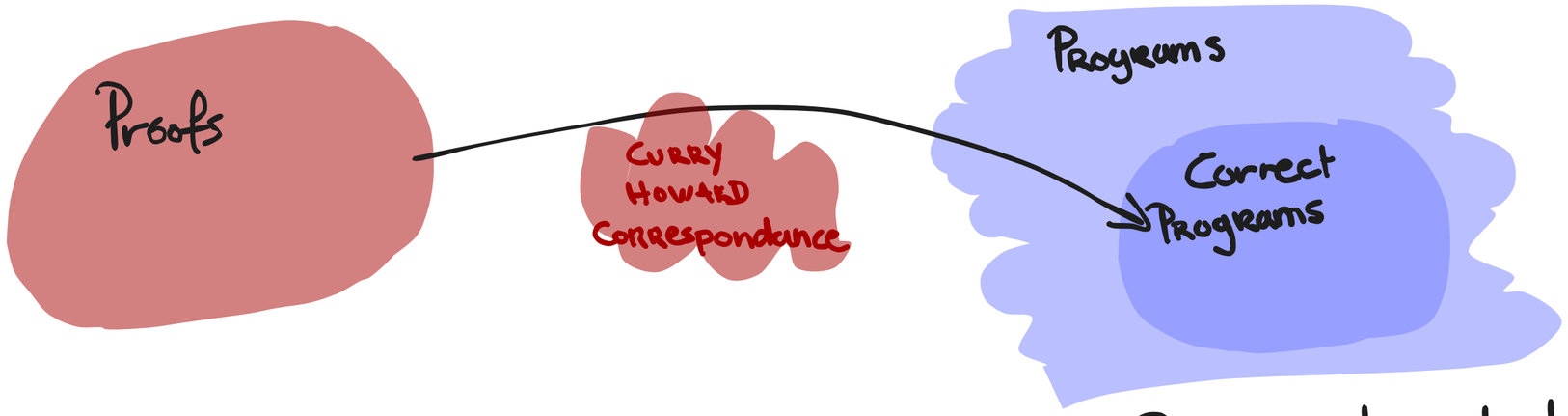
in Computer Science

Computer scientists / Programmers are used to incorrectness (BUGS)

(• In practice, programmers seek to eliminate bugs.)

• Infinite loops e.g. $\lambda x.(x)x$

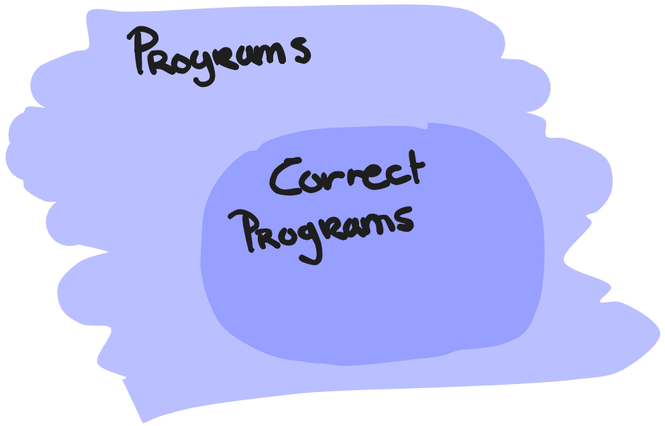
Formalisms:
- Hilbert System, Natural Deduction, Sequent Calculus, ...
- proof = tree



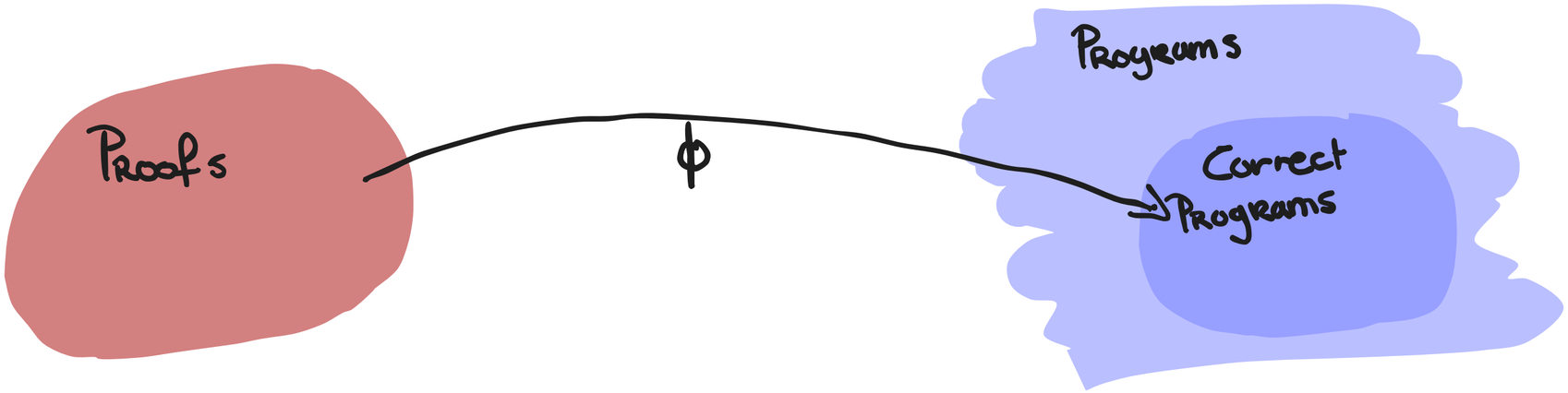
CORRECT-BY-DESIGN
↘ poorer, less expressive

Correct Programs have to be identified
↳ typing inference, Realisability, semantics, ...

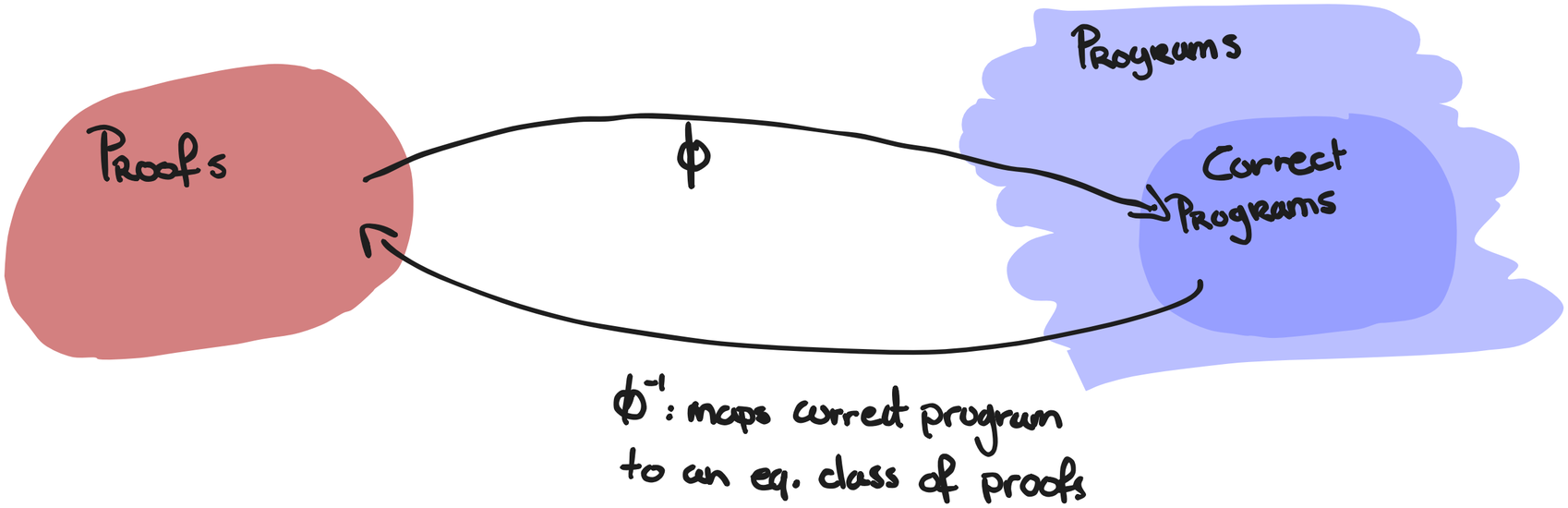
The "Philosophy" / Framework



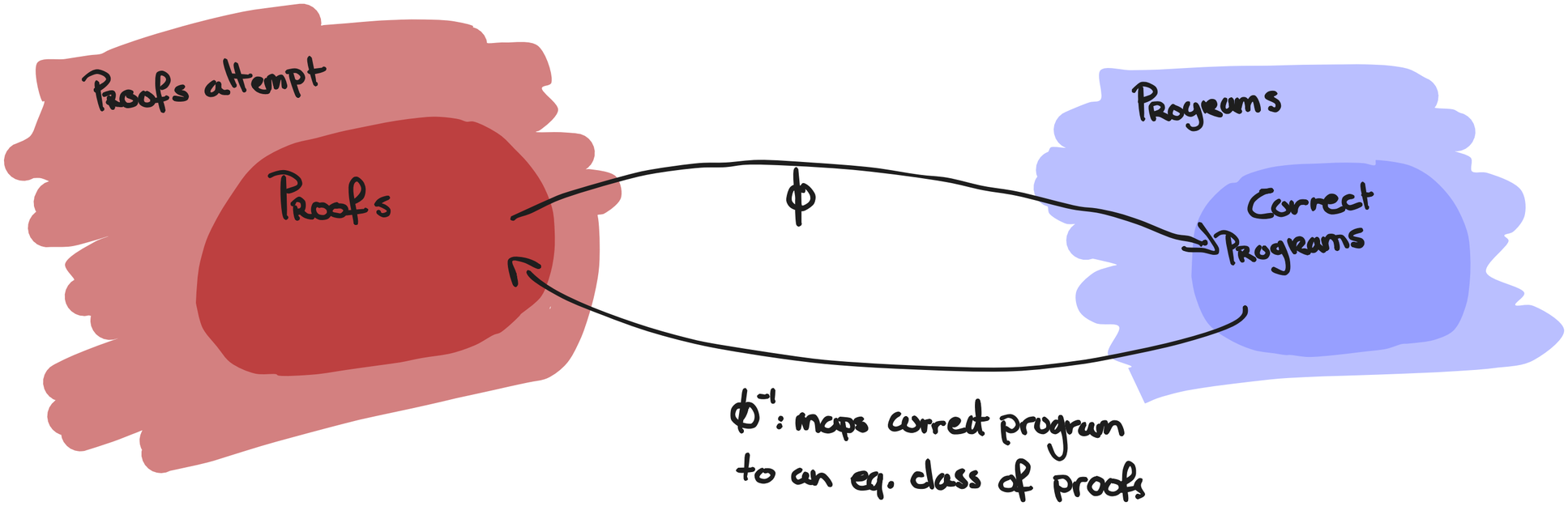
The "Philosophy" / Framework



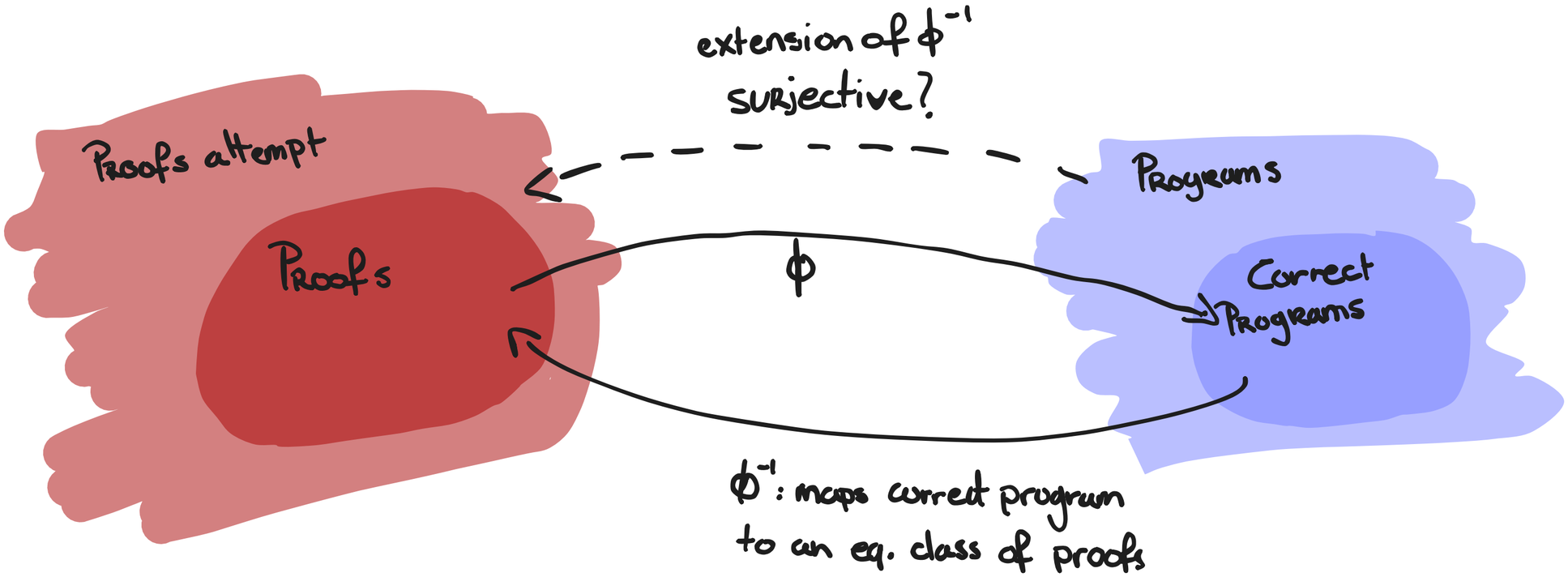
The "Philosophy" / Framework



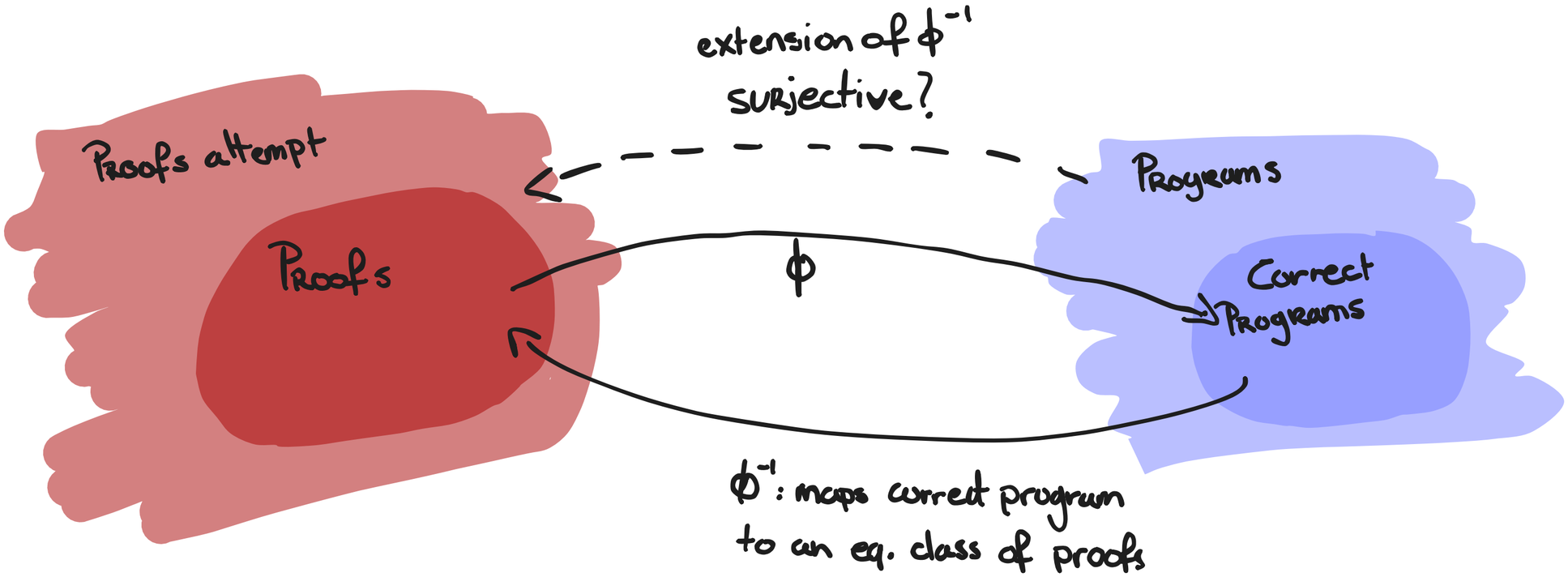
The "Philosophy" / Framework



The "Philosophy" / Framework



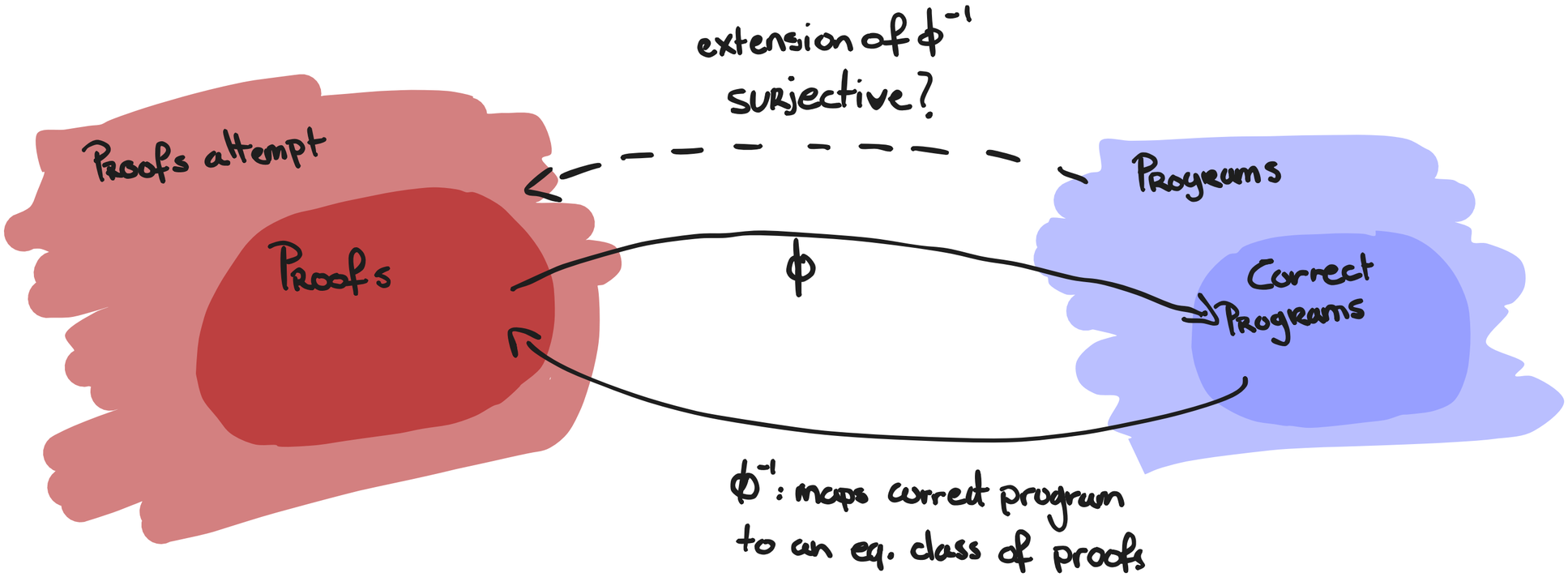
The "Philosophy" / Framework



Evaluate Correctness

- semantics, type inference, realisability

The "Philosophy" / Framework



~~correct by design~~

Evaluate correctness

Evaluate Correctness

- semantics, type inference, realisability

Linear Logic Proof nets

Linear Logic Proof nets

- We restrict ourselves to Linear Logic (Girard, 1987)

Linear Logic Proof nets

- We restrict ourselves to Linear Logic (Girard, 1987)
  - ↳ specifically MLL2: multiplicative second order Linear Logic.

Linear Logic Proof nets

- We restrict ourselves to Linear Logic (Girard, 1987)
  - ↳ specifically MLL2: multiplicative second order Linear Logic.

- We implement the program in proof nets

Linear Logic Proof nets

- We restrict ourselves to Linear Logic (Girard, 1987)
  - ↳ specifically MLL2: multiplicative second order Linear Logic.

- We implement the program in proof nets
  - → graph-like structures

Linear Logic Proof nets

- We restrict ourselves to Linear Logic (Girard, 1987)
  - ↳ specifically MLL2: multiplicative second order Linear Logic.

- We implement the program in proof nets → graph-like structures

This provides the desired framework:

Linear Logic Proof nets

- We restrict ourselves to Linear Logic (Girard, 1987)
  - ↳ specifically MLL2: multiplicative second order Linear Logic.

- We implement the program in proof nets
  → graph-like structures

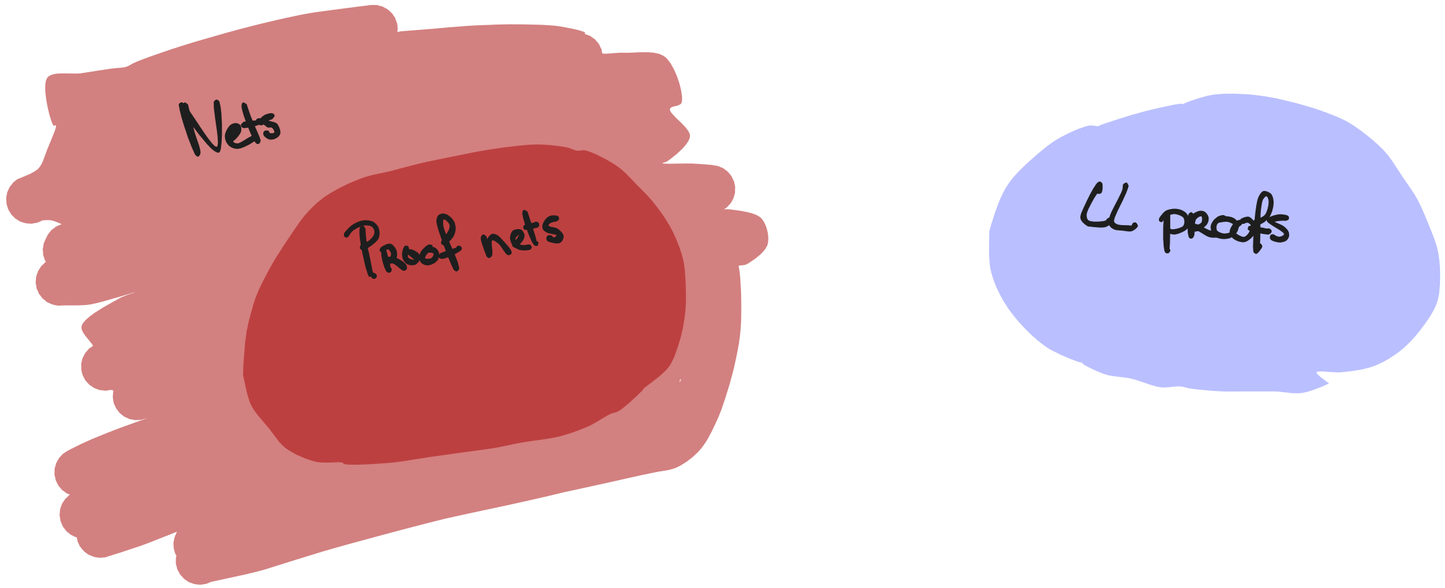
This provides the desired framework:

Linear Logic Proof nets

- We restrict ourselves to Linear Logic (Girard, 1987)
  - ↳ specifically MLL2: multiplicative second order Linear Logic.

- We implement the program in proof nets
  → graph-like structures

This provides the desired framework:

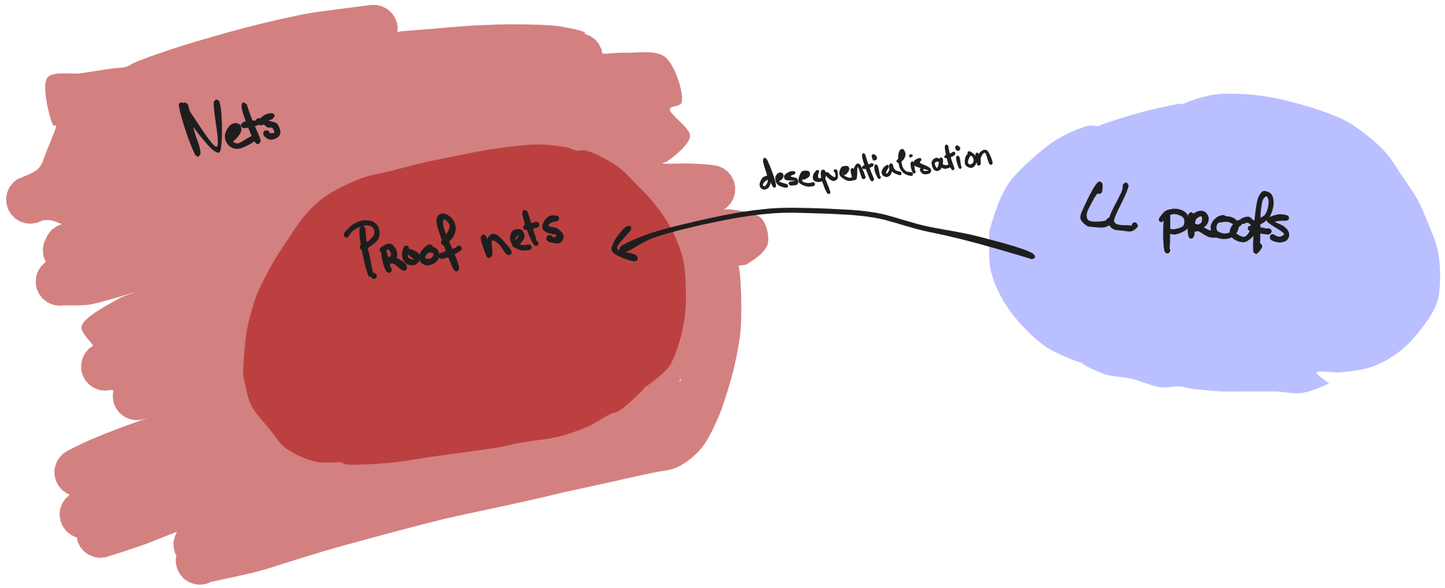


Linear Logic Proof nets

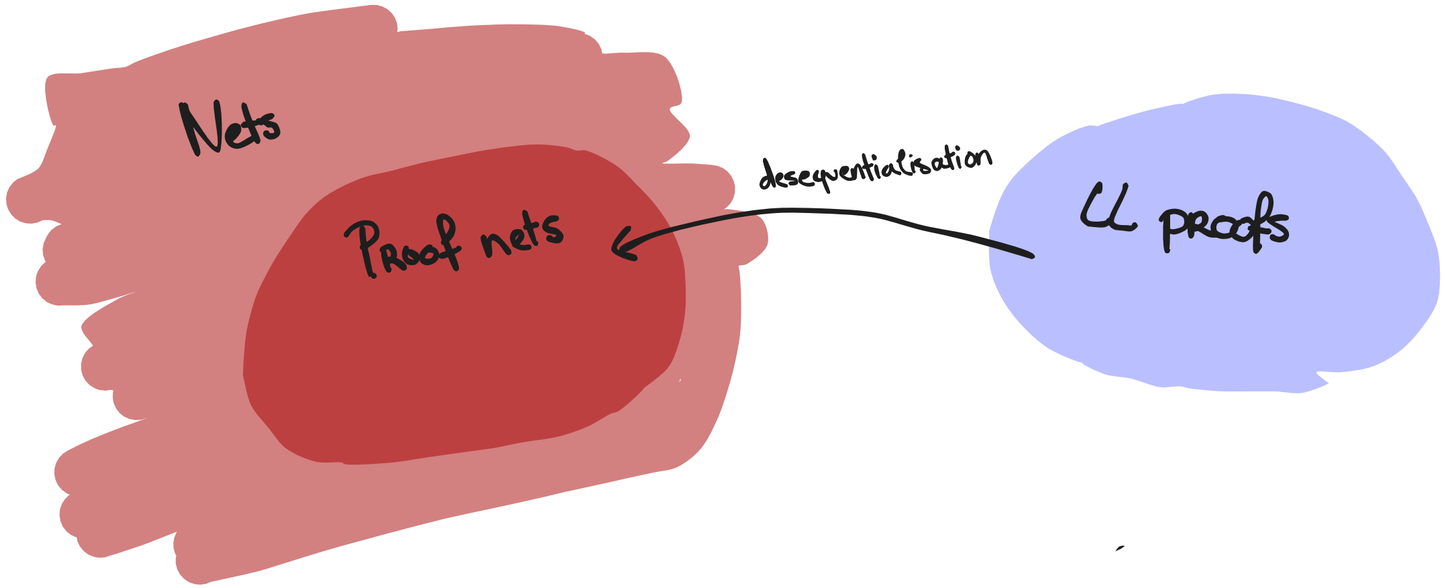
- We restrict ourselves to Linear Logic (Girard, 1987)
  - ↳ specifically MLL2: multiplicative second order Linear Logic.

- We implement the program in proof nets
  - → graph-like structures

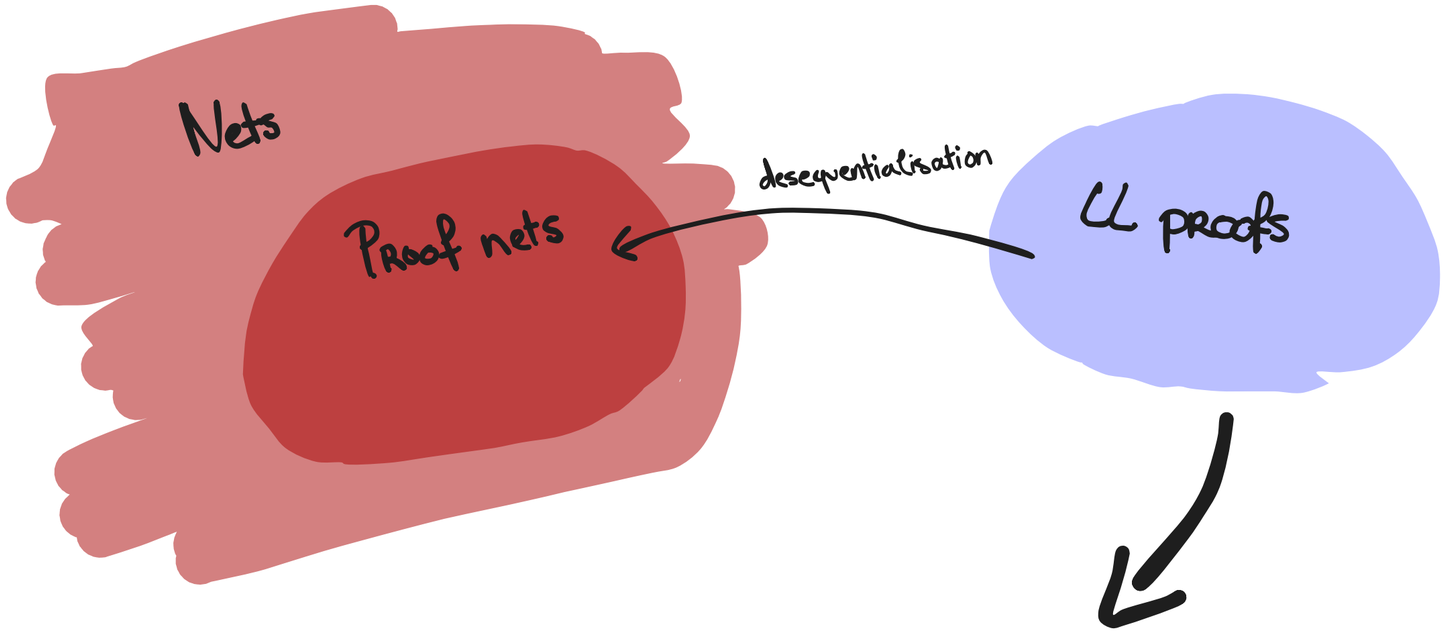
This provides the desired framework:



MLL2 Sequent Calculus



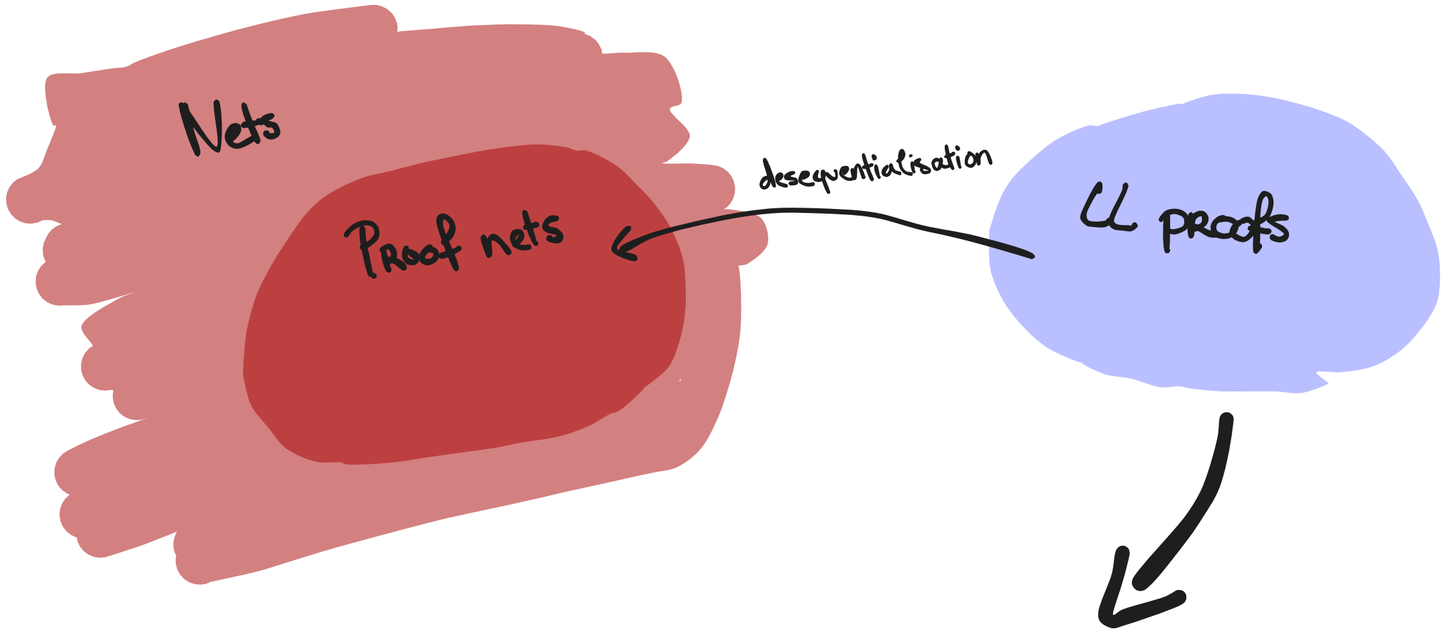
MLL2 Sequent Calculus



$$\frac{}{A, A^{\perp}}\,ax$$

$$\frac{}{\vdash \Gamma}\,\top$$

MLL2 Sequent Calculus

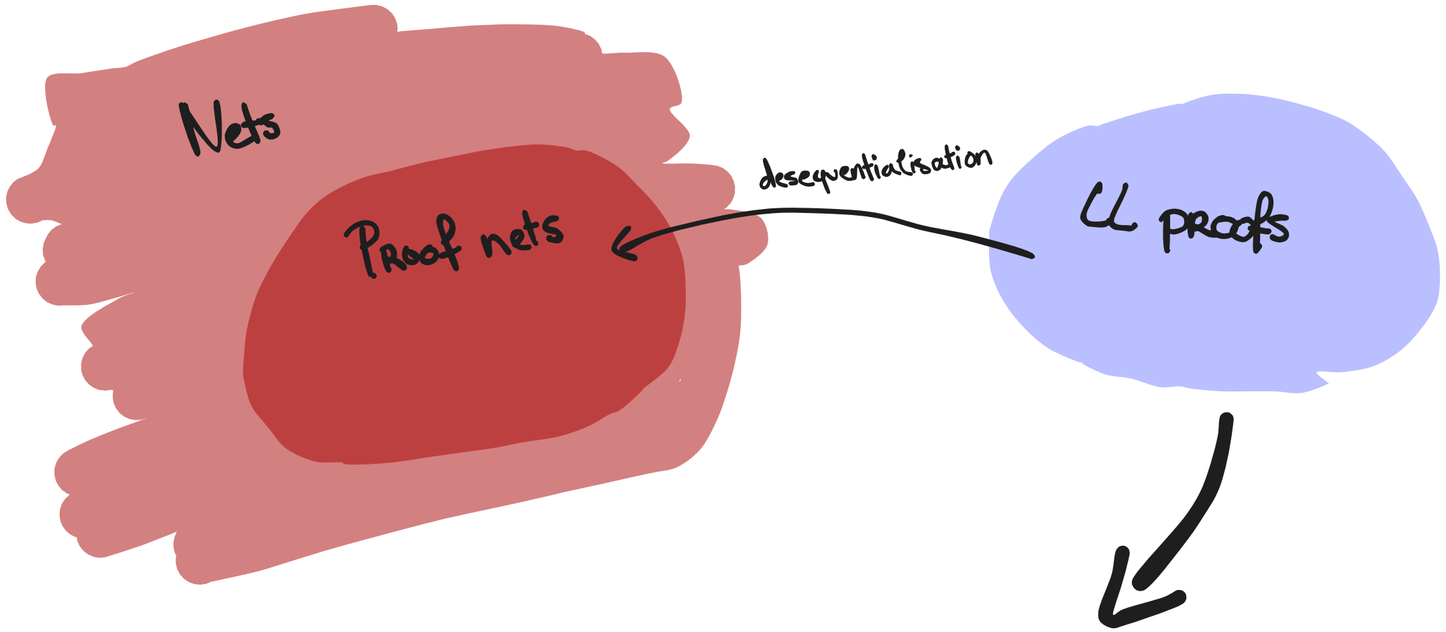


$$\frac{}{A, A^{\perp}}\,ax$$

$$\frac{}{\Gamma}\,+$$

$$\frac{\Gamma, A, B, \Delta}{\Gamma, B, A, \Delta}\,ex \qquad \frac{\Gamma, A \quad \Delta, A^{\perp}}{\Gamma, \Delta}\,cut$$

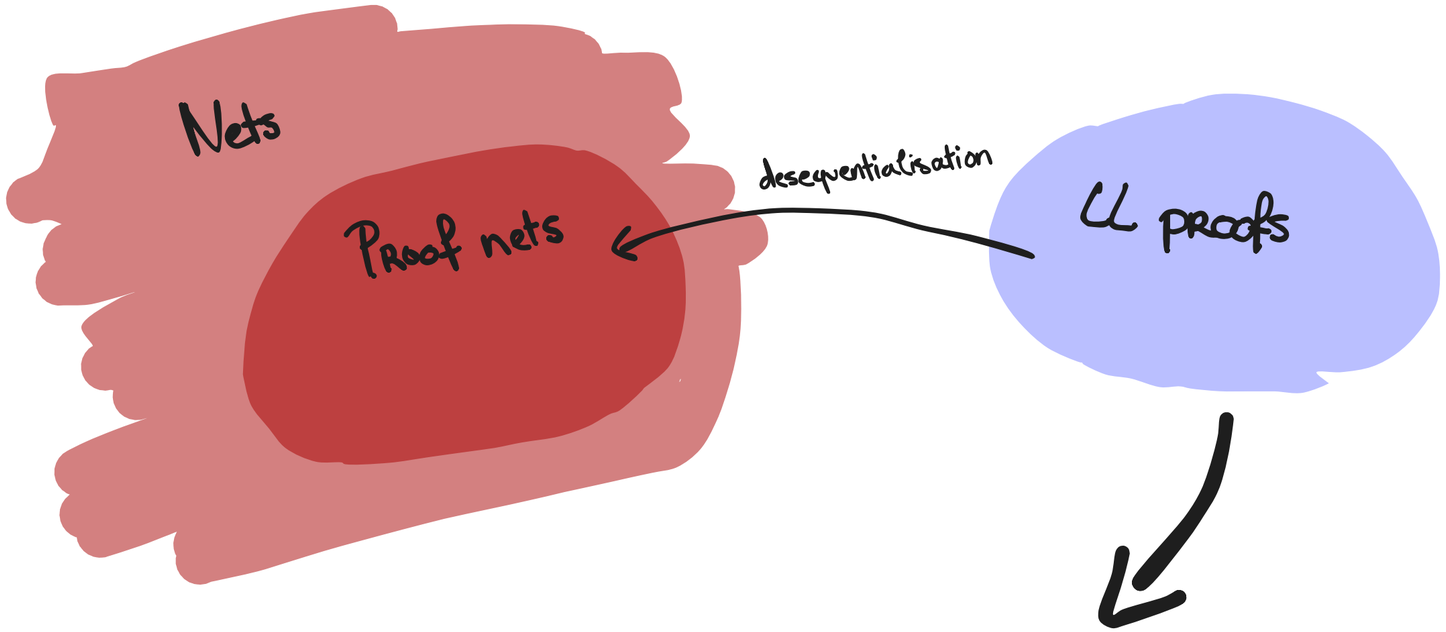
MLL2 Sequent Calculus



$$\frac{}{A, A^{\perp}}\,ax \qquad \frac{\Gamma, A, B}{\Gamma, A \parr B}\,\parr \qquad \frac{\Gamma, A \qquad \Delta, B}{\Gamma, A \otimes B}\,\otimes$$

$$\frac{}{\Gamma}\,\top \qquad \frac{\Gamma, A, B, \Delta}{\Gamma, B, A, \Delta}\,ex \qquad \frac{\Gamma, A \qquad \Delta, A^{\perp}}{\Gamma, \Delta}\,cut$$

MLL2 Sequent Calculus



$$\frac{}{A, A^{\perp}}\,ax \qquad \frac{\Gamma, A, B}{\Gamma, A \parr B}\,\parr \qquad \frac{\Gamma, A \qquad \Delta, B}{\Gamma, A \otimes B}\,\otimes \qquad \frac{\Gamma, A[B/X]}{\Gamma, \exists X\, A}\,\exists$$

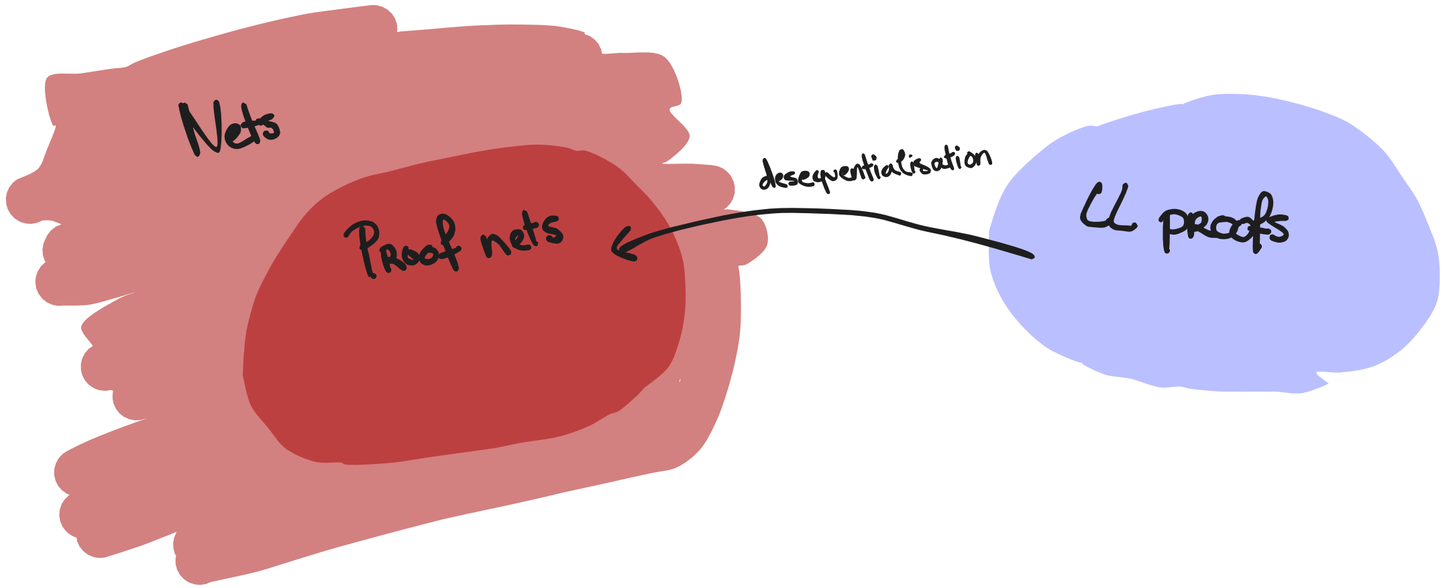
$$\frac{}{\Gamma}\,+ \qquad \frac{\Gamma, A, B, \Delta}{\Gamma, B, A, \Delta}\,ex \qquad \frac{\Gamma, A \qquad \Delta, A^{\perp}}{\Gamma, \Delta}\,cut \qquad \frac{\Gamma, A}{\Gamma, \forall X A}\,\forall \quad (X \text{ does not occur free in } \Gamma)$$

II Parsing criterion

(Recipe)

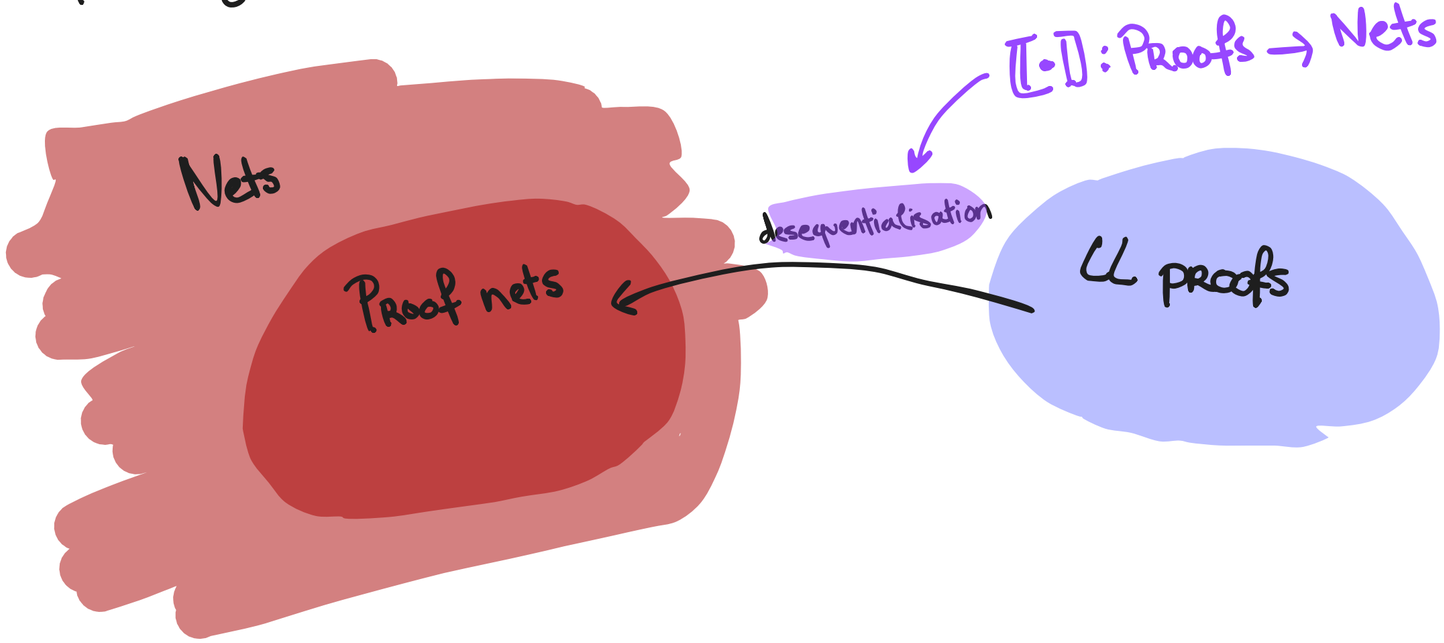
PARSING

Ingredients for parsing:



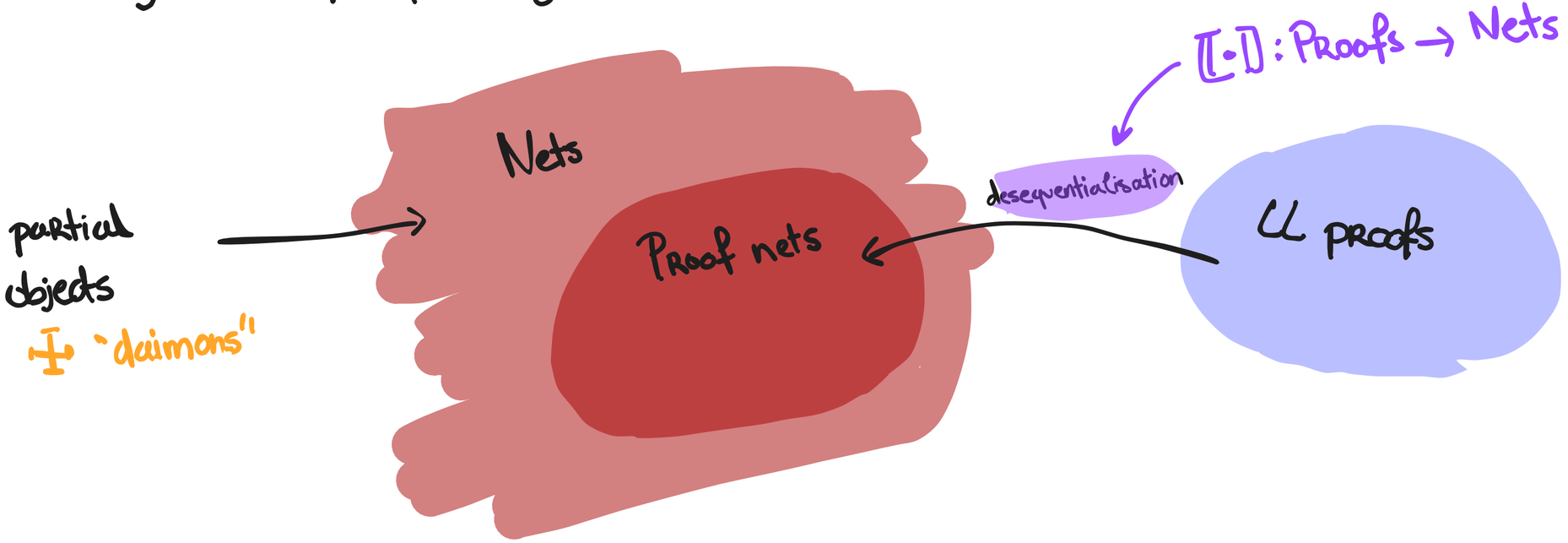
PARSING

Ingredients for parsing:



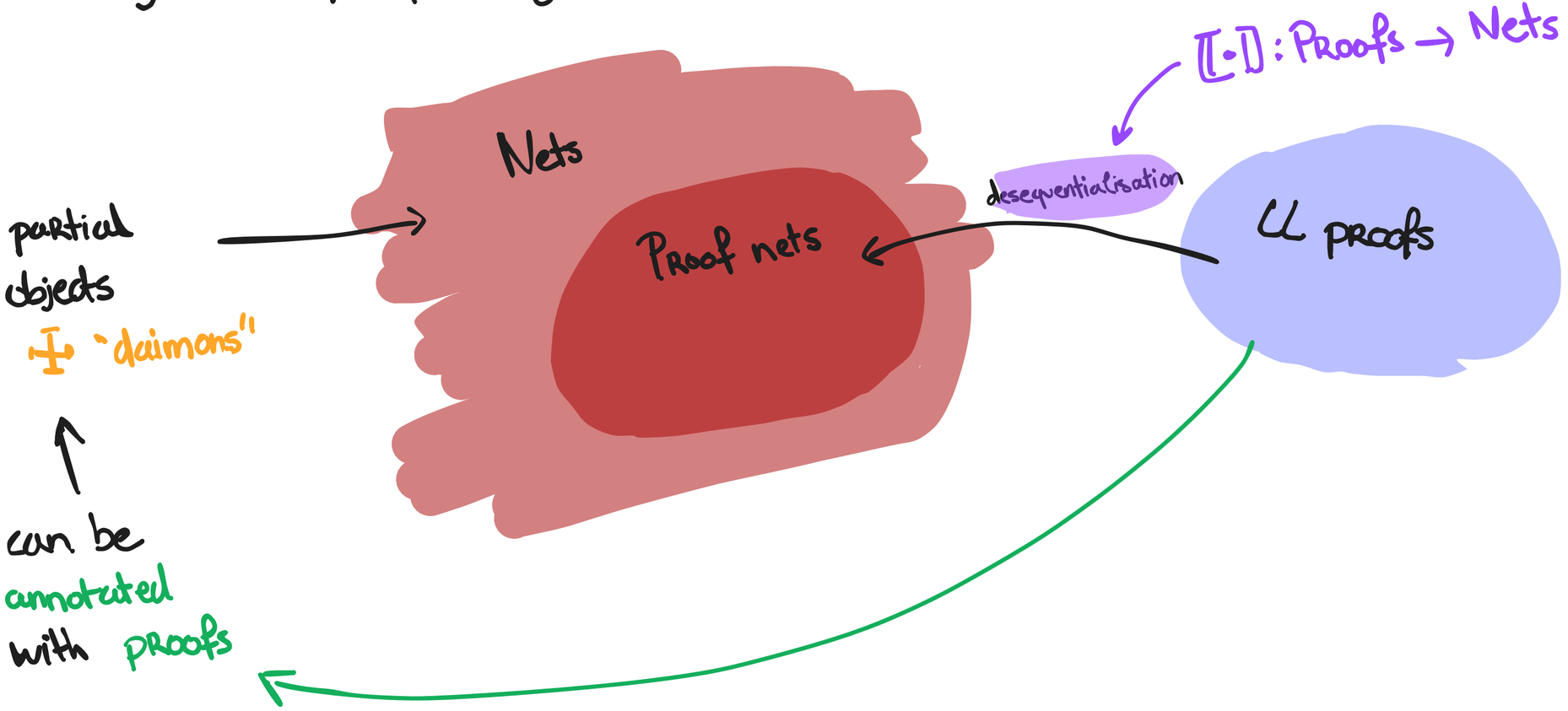
PARSING

Ingredients for parsing:



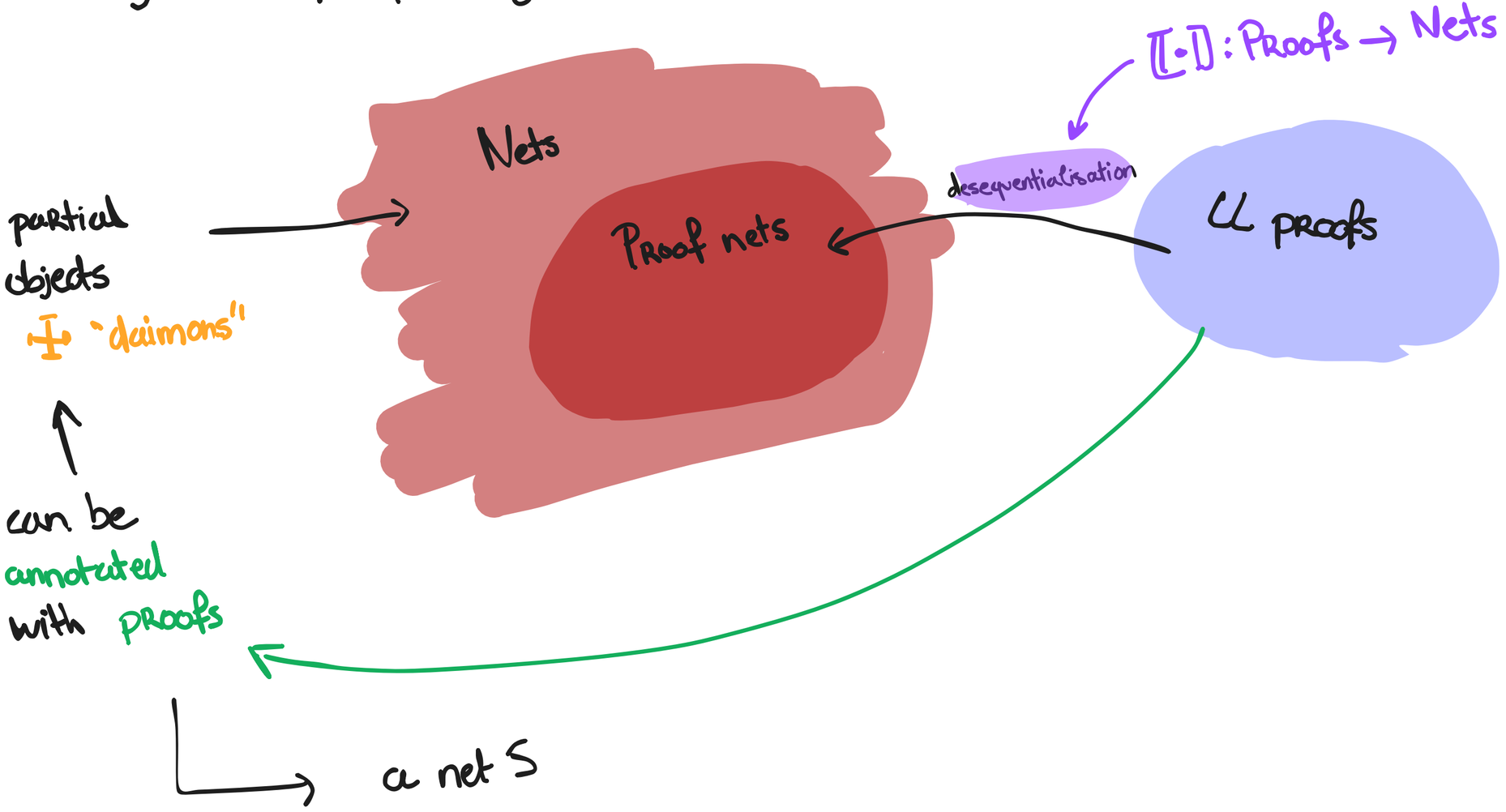
PARSING

Ingredients for parsing:



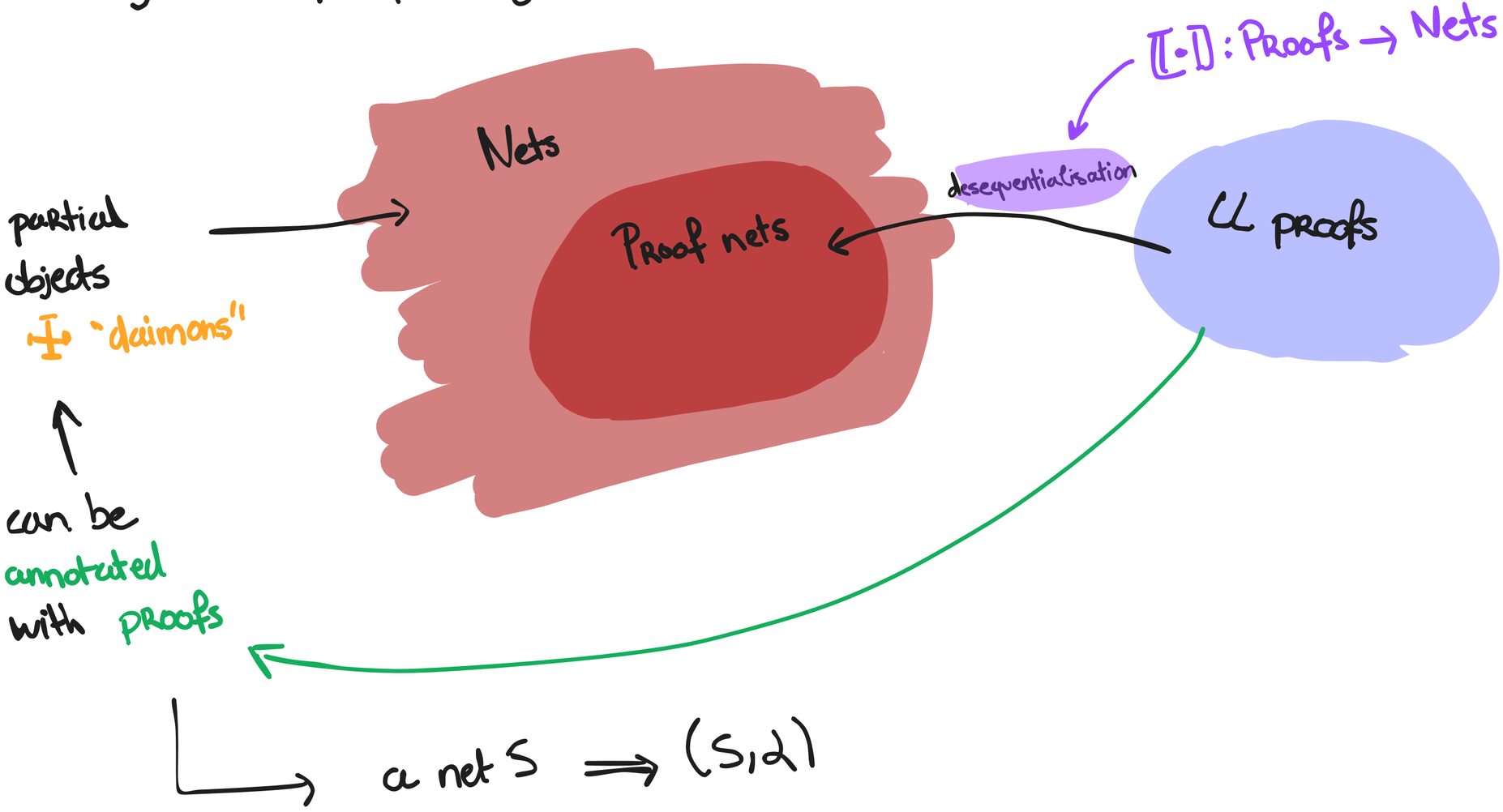
PARSING

Ingredients for parsing:



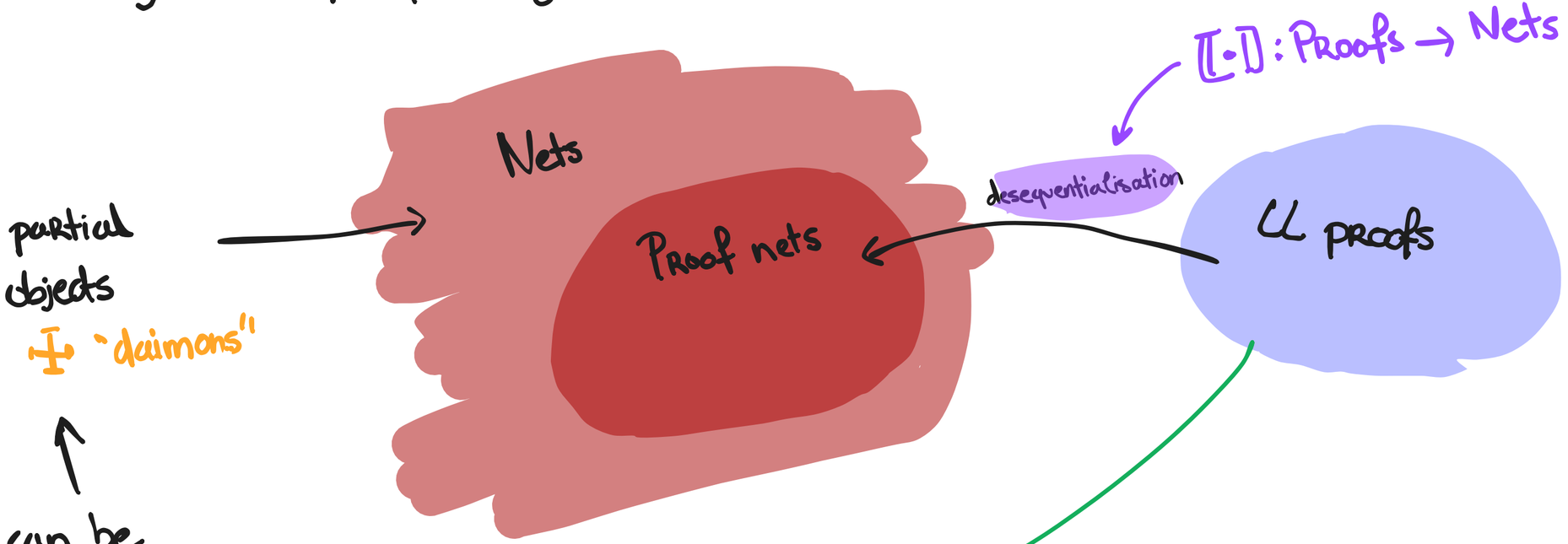
PARSING

Ingredients for parsing:



PARSING

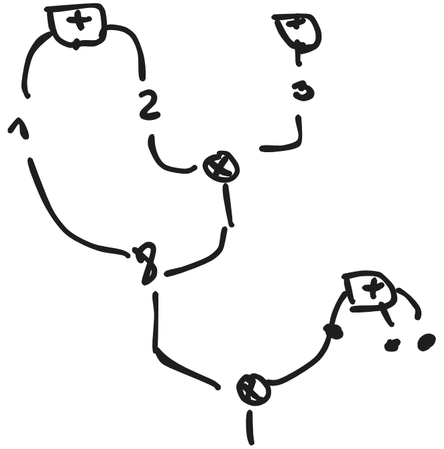
Ingredients for parsing:



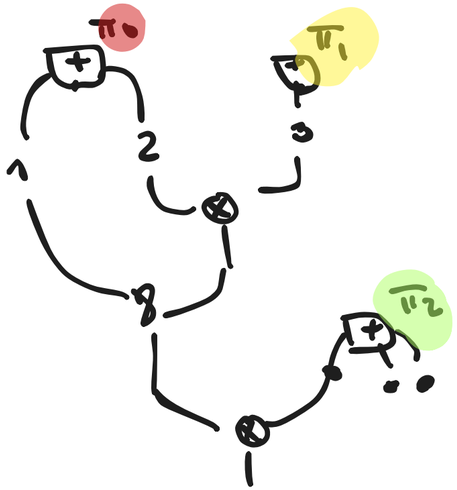
↳ a net $S \Longrightarrow (S, \alpha)$

annotation: $S^{\text{fin}} \to$ Proofs (many)

a net S



an annotated version (S, λ)



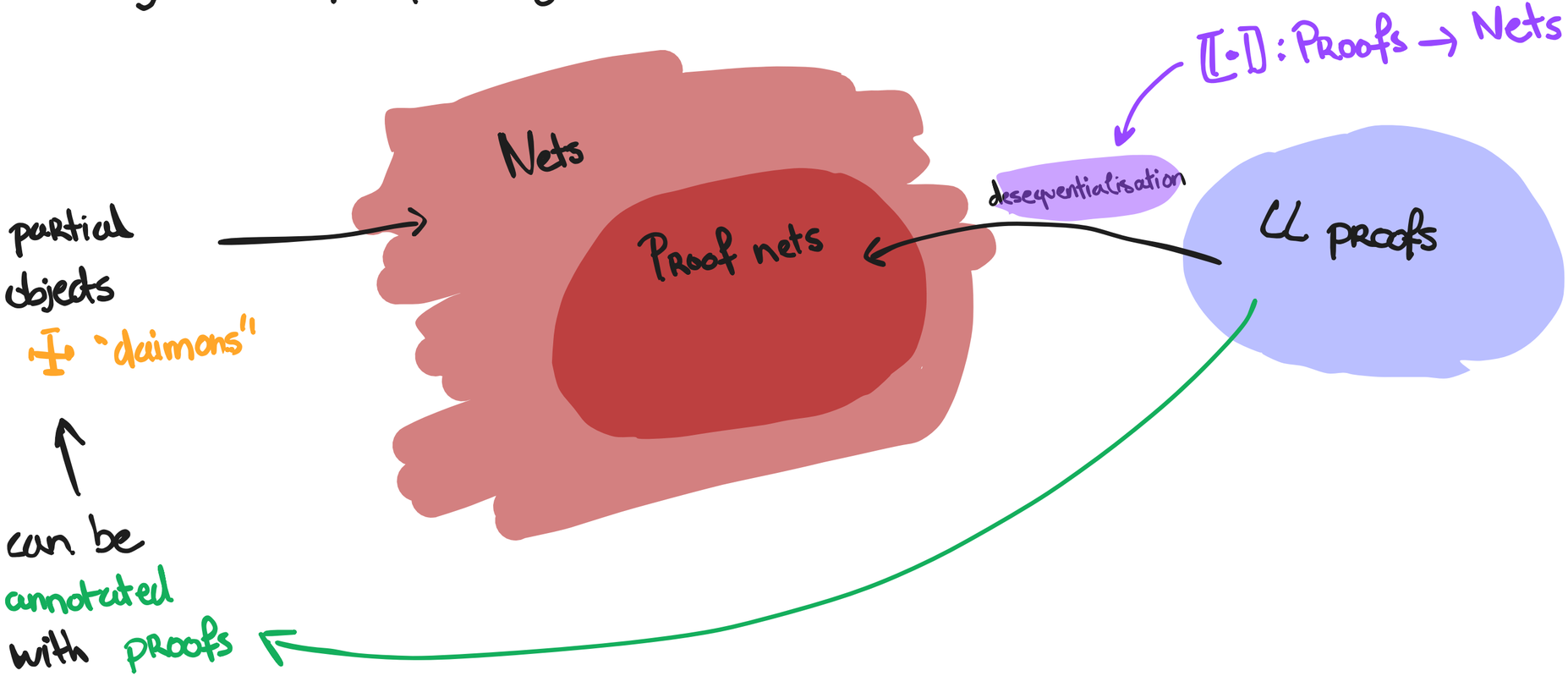
$\pi_0 = \dfrac{}{A, A^\perp}\,+$

$\pi_1 = \dfrac{\dfrac{}{A}\,+ \qquad \dfrac{}{B, C}\,+}{A \otimes B, C}\,\otimes \Big/ \; (A \otimes B) \parr C$

$\pi_2 = \dfrac{}{A, B, C, D}\,+ \Big/ \; A \parr B, C, D$

PARSING

Ingredients for parsing:



PARSING = REWRITING ON NETS

Two PARSINGS
- for annotated nets → P^{a}
- for regular nets → P

PARSING CRITERION

Theorem

$S \rightarrow_p \Vdash \quad \Leftrightarrow \quad S$ is correct

PARSING CRITERION

a net with a single link

Theorem

$S \to p \vdash$ $\Leftrightarrow$ S is correct

PARSING CRITERION

a net with a single link

Theorem

$S \to p$ ⊢ $\iff$ S is correct

$(S = [\![\pi]\!]$)

π ↑ proof

PARSING CRITERION

a net with a single link

Theorem

$S \to p$ ⊢ $\Leftrightarrow$ S is correct

$(S = [\![\pi]\!]$)

proof

$\Leftarrow$ inductively on π.

Should be made easy:

PARSING CRITERION

a net with a single link

Theorem

$S \to p$ ⊢ $\iff$ S is correct

$(S = [\![\pi]\!]$)

proof

$\Leftarrow$ inductively on π.

Should be made easy:

$$\left[\frac{\Gamma, A, B}{\Gamma, A \parr B}^{+}\right] \to p \left[\frac{}{\Gamma, A \parr B}^{+}\right]$$

PARSING CRITERION

a net with a single link

Theorem

$S \to_P \vdash \Leftrightarrow$ S is correct

$(S = [\![\pi]\!])$

π : proof

$\Leftarrow$ inductively on π.

Should be made easy:

$$\left[\!\!\left[\dfrac{\Gamma, A, B}{\Gamma, A \parr B}\, + \right]\!\!\right] \to_P \left[\!\!\left[\dfrac{}{\Gamma, A \parr B}\, + \right]\!\!\right]$$

$$\left[\!\!\left[\dfrac{\vdash \Gamma, A^+ \quad \vdash \Delta, B^+}{\vdash \Gamma, A \otimes B, \Delta}\, \otimes \right]\!\!\right] \to_P \left[\!\!\left[\dfrac{}{\vdash \Gamma, A \otimes B, \Delta}\, + \right]\!\!\right]$$

PARSING CRITERION

a net with a single link

Theorem

$S \to_P \pi$ $\Leftrightarrow$ S is correct

$(S = [\![\pi]\!])$

π ← proof

$\Leftarrow$ inductively on π.

Should be made easy:

$$\left[\!\!\left[\dfrac{\Gamma, A, B}{\Gamma, A \parr B}\,{}^{+} \right]\!\!\right] \to_P \left[\!\!\left[\dfrac{}{\Gamma, A \parr B}\,{}^{+} \right]\!\!\right]$$

$$\left[\!\!\left[\dfrac{\dfrac{}{\Gamma, A}\,{}^{+} \quad \dfrac{}{\Delta, B}\,{}^{+}}{\Gamma, A \otimes B, \Delta}\otimes \right]\!\!\right] \to_P \left[\!\!\left[\dfrac{}{\Gamma, A \otimes B, \Delta}\,{}^{+} \right]\!\!\right]$$

$$\left[\!\!\left[\dfrac{\dfrac{}{\Gamma, A}\,{}^{+}}{\Gamma, \forall X A}\forall \right]\!\!\right] \to_P \left[\!\!\left[\dfrac{}{\Gamma, \forall X A}\,{}^{+} \right]\!\!\right]$$

PARSING CRITERION

a net with a single link

Theorem

$S \to_P \pi$ $\iff$ S is correct

$(S = [\![\pi]\!])$

π : proof

$[\![\Leftarrow]\!]$ inductively on π.

Should be made easy:

$$[\![\dfrac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\, \parr^+]\!] \to_P [\![\vdash \Gamma, A \parr B \,^+]\!]$$

$$[\![\dfrac{\vdash \Gamma, A^+ \quad \vdash \Delta, B^+}{\vdash \Gamma, A \otimes B, \Delta}\, \otimes]\!] \to_P [\![\vdash \Gamma, A \otimes B, \Delta \,^+]\!]$$

$$[\![\dfrac{\vdash \Gamma, A^+}{\vdash \Gamma, \forall X A}\, \forall]\!] \to_P [\![\vdash \Gamma, \forall X A \,^+]\!]$$

$$[\![\dfrac{\vdash \Gamma, A[B/X]^+}{\vdash \Gamma, \exists X A}]\!] \to_P [\![\vdash \Gamma, \exists X A \,^+]\!]$$

PARSING

PARSING = REWRITING ON NETS

Two PARSINGS
- for annotated nets → P^a
- for regular nets → P

PARSING

PARSING = REWRITING ON NETS

Two PARSINGS
- for annotated nets → P^a
- for regular nets → P

KEY PROPERTIES:

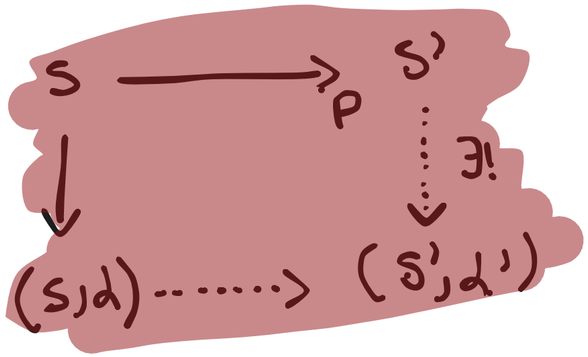
PARSING

PARSING = REWRITING ON NETS

Two PARSINGS
- for annotated nets → $\mathcal{P}^a$
- for regular nets → $\mathcal{P}$

KEY PROPERTIES:

D1



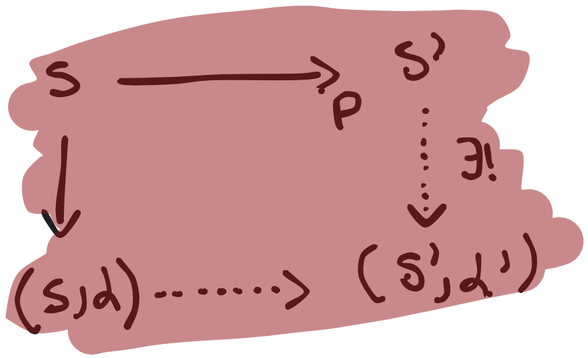
PARSING

PARSING = REWRITING ON NETS

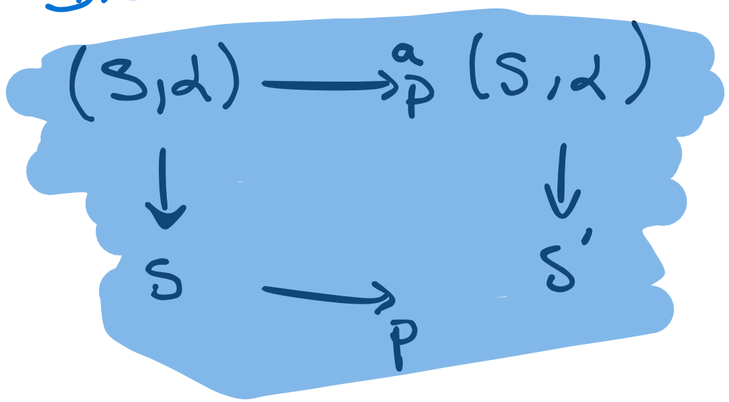
Two PARSINGS
- for annotated nets → $\xrightarrow{a}_P$
- for regular nets → $\to_P$

KEY PROPERTIES:

D1



D2



PARSIN CRITERION: LEMMAS

(S, α) valid $\overset{\text{def}}{\iff}$ α maps daimons to proofs

PARSIN CRITERION: LEMMAS

(S, d) valid $\overset{\text{def}}{\iff}$ d maps daimons to proofs

(S, d) elementary $\overset{\text{def}}{\iff}$ d maps daimons to proofs of the form $\dfrac{}{\pi^+}$

PARSIN CRITERION: LEMMAS

(S, α) valid $\overset{\text{def}}{\iff}$ α maps daimons to proofs

(S, α) elementary $\overset{\text{def}}{\iff}$ α maps daimons to proofs of the form $\dfrac{}{\pi^+}$

PROPOSITIONS

PARSIN CRITERION: LEMMAS

(S, d) valid $\overset{\text{def}}{\iff}$ d maps daimons to proofs

(S, d) elementary $\overset{\text{def}}{\iff}$ d maps daimons to proofs of the form $\dfrac{}{\pi^+}$

PROPOSITIONS

P1. $\to_P$ preserves validity.

PARSIN CRITERION: LEMMAS

(S, d) valid $\overset{\text{def}}{\iff}$ d maps daimons to proofs

(S, d) elementary $\overset{\text{def}}{\iff}$ d maps daimons to proofs of the form $\dfrac{}{\pi^+}$

PROPOSITIONS

P1. $\to_P$ preserves validity.

P2. $P \leftarrow$ preserves validity

PARSIN CRITERION: LEMMAS

(S, α) valid $\overset{\text{def}}{\iff}$ α maps daimons to proofs

(S, α) elementary $\overset{\text{def}}{\iff}$ α maps daimons to proofs of the form $\dfrac{}{\pi^+}$

PROPOSITIONS

P1. $\to_P$ preserves validity.

P2. $P\leftarrow$ preserves validity

P3. $(\dagger, \lambda : \dagger \mapsto \pi) \overset{*}{\underset{P}{\leftarrow}} (\llbracket \pi \rrbracket, \beta)$

proof

elementary

PARSIN CRITERION: LEMMAS

(S, α) valid $\overset{\text{def}}{\iff}$ α maps daimons to proofs

(S, α) elementary $\overset{\text{def}}{\iff}$ α maps daimons to proofs of the form $\dfrac{}{\pi^+}$

PROPOSITIONS

P1. $\to_P$ preserves validity.

P2. $P \leftarrow$ preserves validity

P3. $(\ddagger, \alpha : \ddagger \mapsto \pi) \overset{*}{\underset{P}{\leftarrow}} (\llbracket \pi \rrbracket, \beta)$ (proof, elementary)

AP1. valid antiparsing normal forms = elementary annotated net.

PARSIN CRITERION: LEMMAS

(S, α) valid $\overset{\text{def}}{\iff}$ α maps daimons to proofs

(S, α) elementary $\overset{\text{def}}{\iff}$ α maps daimons to proofs of the form $\overline{\pi^+}$

PROPOSITIONS

P1. $\to_p$ preserves validity.

P2. $p\leftarrow$ preserves validity

P3. $(\mathcal{H}, \alpha : \mathcal{H} \mapsto \pi) \overset{*}{\underset{p}{\leftarrow}} (\llbracket \pi \rrbracket, \beta)$ (proof, elementary)

AP1. valid antiparsing normal forms = elementary annotated net.

AP2. antiparsing is SN

PARSING CRITERION

Theorem

$S \rightarrow_p \Vdash \quad \Leftrightarrow \quad S$ is correct

$\Rightarrow$

PARSING CRITERION

Theorem

$S \to p \vdash \quad \Leftrightarrow \quad S$ is correct

$\Rightarrow$

Assume $S \to p \vdash$

PARSING CRITERION

Theorem

$S \to_p \Pi \iff S$ is correct

$\boxed{\Rightarrow}$

Assume $S \to_p \Pi$

- by recursively using (P1), $(S, \alpha) \to_p (\Pi, \alpha_0 : \Pi \mapsto \pi)$

PARSING CRITERION

Theorem

$S \to_p \Pi \iff S$ is correct

$\boxed{\Rightarrow}$

Assume $S \to_p \Pi$

- by recursively using (P1), $(S, \alpha) \to_p (\Pi, \alpha_0 : \Pi \mapsto \pi)$

↑ valid + elementary

PARSING CRITERION

Theorem

$S \to_p \Psi \iff S$ is correct

$\Rightarrow$

Assume $S \to_p \Psi$

- by recursively using (P1), $(S, \alpha) \to_p (\Psi, \alpha_0 : \Psi \mapsto \pi)$

  (S, α): valid + elementary

  $\alpha_0 : \Psi \mapsto \pi$: valid (P1)

PARSING CRITERION

Theorem

$S \to_p \Psi \iff S$ is correct

$\boxed{\Rightarrow}$

Assume $S \to_p \Psi$

- by recursively using (P1), $(S, \alpha) \to_p (\Psi, \alpha_0 : \Psi \mapsto \pi)$
  - (S, α): valid + elementary
  - Ψ: valid (P1)
  - $\alpha_0 : \Psi \mapsto \pi$: proof

PARSING CRITERION

Theorem

$S \to_p \Psi \iff S$ is correct

$\Rightarrow$

Assume $S \to_p \Psi$

- by recursively using (P1), $(S, \alpha) \to_p (\Psi, \alpha_0 : \Psi \mapsto \pi)$
  - α: valid + elementary
  - α_0: valid (P1)
  - π: proof

- $(\Psi, \alpha_0 : \Psi \mapsto \pi) \xleftarrow{*}_p (\llbracket \pi \rrbracket, \beta)$ (P3)

PARSING CRITERION

Theorem

$S \to_p \Psi \iff S$ is correct

$\boxed{\Rightarrow}$

Assume $S \to_p \Psi$

- by recursively using (P1), $(S, \alpha) \to_p (\Psi, d_0 : \Psi \mapsto \pi)$
  - (S, α): valid + elementary
  - $(\Psi, d_0 : \Psi \mapsto \pi)$: valid (P1); $d_0 : \Psi \mapsto \pi$: proof
- $(\Psi, d_0 : \Psi \mapsto \pi) \overset{*}{\leftarrow}_p (\llbracket \pi \rrbracket, \beta)$ (P3)
- Thus conj $(\tau, \beta) \to_p^* (\Psi, d_0)$ (AP1 + AP2)

  is equal to $(\llbracket \pi \rrbracket, \beta)$

PARSING CRITERION

Theorem

$S \to_p \Pi \iff S$ is correct

$\Rightarrow$

Assume $S \to_p \Pi$

- by recursively using (P1), $(S, \alpha) \to_p (\Pi, \alpha_0 : \Pi \mapsto \pi)$
  - (S, α): valid + elementary
  - $(\Pi, \alpha_0 : \Pi \mapsto \pi)$: valid (P1); proof
- $(\Pi, \alpha_0 : \Pi \mapsto \pi) \overset{*}{\leftarrow}_p (\llbracket \pi \rrbracket, \beta)$ (P3)
- Thus conf $(T, \beta) \to_p^* (\Pi, \alpha_0)$ (AP1 + AP2)
  is equal to $(\llbracket \pi \rrbracket, \beta)$
- hence $(S, \alpha) = (\llbracket \pi \rrbracket, \beta)$ and so $S = \Pi \pi \rrbracket$

□

III Deriving more criterions

New criterions

New criterions

(a) S is P and $S \rightarrow_p S'$ then S' is P

New criterions

(a) S is P and $S \to_p S'$ then S' is P

(b) S' is P and $S \to_p S'$ then S is P

New criterions

(a) S is $\mathcal{P}$ and $S \to_p S'$ then S' is $\mathcal{P}$

(b) S' is $\mathcal{P}$ and $S \to_p S'$ then S is $\mathcal{P}$

(c) S is contracted and $\mathcal{P} \Rightarrow S = \mathbb{H}$.

New criterions

(a) S is $\mathcal{P}$ and $S \to_p S'$ then S' is $\mathcal{P}$

(b) S' is $\mathcal{P}$ and $S \to_p S'$ then S is $\mathcal{P}$

(c) S is contracted and $\mathcal{P} \Rightarrow S = \mathbb{H}$.

THEOREM

S is correct
$\Leftrightarrow$
S is $\mathcal{P}$

New criterions

(a) S is $\mathcal{P}$ and $S \to_p S'$ then S' is $\mathcal{P}$

(b) S' is $\mathcal{P}$ and $S \to_p S'$ then S is $\mathcal{P}$

(c) S is contracted and $\mathcal{P} \Rightarrow S = H$.

THEOREM

S is correct
$\Leftrightarrow$
S is $\mathcal{P}$

$\Rightarrow$ Assume $S \to_p^* H$,

New criterions

(a) S is $\mathcal{P}$ and $S \to_p S'$ then S' is $\mathcal{P}$

(b) S' is $\mathcal{P}$ and $S \to_p S'$ then S is $\mathcal{P}$

(c) S is contracted and $\mathcal{P}$ $\Rightarrow$ $S = \mathcal{H}$.

THEOREM

S is correct

$\Leftrightarrow$

S is $\mathcal{P}$

$\Rightarrow$ Assume $S \to_p^* \mathcal{H}$, since $\mathcal{H}$ is $\mathcal{P}$

New criterions

(a) S is $\mathcal{P}$ and $S \to_p S'$ then S' is $\mathcal{P}$

(b) S' is $\mathcal{P}$ and $S \to_p S'$ then S is $\mathcal{P}$

(c) S is contracted and $\mathcal{P} \Rightarrow S = \mathbb{H}$.

THEOREM

S is correct
$\Leftrightarrow$
S is $\mathcal{P}$

$\Rightarrow$ Assume $S \to_p^* \mathbb{H}$, since $\mathbb{H}$ is $\mathcal{P}$

by (b) we derive S is $\mathcal{P}$

New criterions

(a) S is $\mathcal{P}$ and $S \to_p S'$ then S' is $\mathcal{P}$

(b) S' is $\mathcal{P}$ and $S \to_p S'$ then S is $\mathcal{P}$

(c) S is contracted and $\mathcal{P} \Rightarrow S = \mathbb{H}$.

THEOREM

S is correct
$\Leftrightarrow$
S is $\mathcal{P}$

$\Rightarrow$ Assume $S \to_p^* \mathbb{H}$, since $\mathbb{H}$ is $\mathcal{P}$
by (b) we derive S is $\mathcal{P}$

$\Leftarrow$ Assume S is $\mathcal{P}$,

New criterions

(a) S is $\mathcal{P}$ and $S \to_p S'$ then S' is $\mathcal{P}$

(b) S' is $\mathcal{P}$ and $S \to_p S'$ then S is $\mathcal{P}$

(c) S is contracted and $\mathcal{P} \Rightarrow S = \mathcal{H}$.

THEOREM

S is correct
$\Leftrightarrow$
S is $\mathcal{P}$

$\Rightarrow$ Assume $S \to_p^* \mathcal{H}$, since $\mathcal{H}$ is $\mathcal{P}$

by (b) we derive S is $\mathcal{P}$

$\Leftarrow$ Assume S is $\mathcal{P}$, compute the contracted form of S

$S \to_p^* S'$,

New criterions

(a) S is $\mathcal{P}$ and $S \to_p S'$ then S' is $\mathcal{P}$

(b) S' is $\mathcal{P}$ and $S \to_p S'$ then S is $\mathcal{P}$

(c) S is contracted and $\mathcal{P} \Rightarrow S = \mathcal{H}$.

THEOREM

S is correct
$\Leftrightarrow$
S is $\mathcal{P}$

$\Rightarrow$ Assume $S \to_p^* \mathcal{H}$, since $\mathcal{H}$ is $\mathcal{P}$

by (b) we derive S is $\mathcal{P}$

$\Leftarrow$ Assume S is $\mathcal{P}$, compute the contracted form of S

$S \to_p^* S'$, by (a) then S' is $\mathcal{P}$ (and contracted)

New criterions

(a) S is $\mathcal{P}$ and $S \to_p S'$ then S' is $\mathcal{P}$

(b) S' is $\mathcal{P}$ and $S \to_p S'$ then S is $\mathcal{P}$

(c) S is contracted and $\mathcal{P}$ $\Rightarrow$ $S = \bot\!\!\!\bot$.

THEOREM

S is correct
$\Leftrightarrow$
S is $\mathcal{P}$

$\Rightarrow$ Assume $S \to_p^* \bot\!\!\!\bot$, since $\bot\!\!\!\bot$ is $\mathcal{P}$

by (b) we derive S is $\mathcal{P}$

$\Leftarrow$ Assume S is $\mathcal{P}$, compute the contracted form of S

$S \to_p^* S'$, by (a) then S' is $\mathcal{P}$ (and contracted)

by (c), $S' = \bot\!\!\!\bot$.

New criterions

(a) S is $\mathcal{P}$ and $S \to_p S'$ then S' is $\mathcal{P}$

(b) S' is $\mathcal{P}$ and $S \to_p S'$ then S is $\mathcal{P}$

(c) S is contracted and $\mathcal{P}$ $\Rightarrow$ $S = \mathcal{H}$.

THEOREM

S is correct

$\Leftrightarrow$

S is $\mathcal{P}$

$\Rightarrow$ Assume $S \to_p^* \mathcal{H}$, since $\mathcal{H}$ is $\mathcal{P}$

by (b) we derive S is $\mathcal{P}$

$\Leftarrow$ Assume S is $\mathcal{P}$, compute the contracted form of S

$S \to_p S'$, by (a) then S' is $\mathcal{P}$ (and contracted)

by (c), $S' = \mathcal{H}$. Thus $S \to \mathcal{H}$. □ .

New criterions

(a) S is ACC and $S \to_p S'$ then S' is ACC

(b) S' is ACC and $S \to_p S'$ then S is ACC

(c) S is contracted and ACC $\Rightarrow$ $S = \dagger$.

THEOREM

S is correct
$\Leftrightarrow$
S is ACC

$\Rightarrow$ Assume $S \to_p^* \dagger$, since $\dagger$ is ACC

by (b) we derive S is ACC

$\Leftarrow$ Assume S is ACC, compute the contracted form of S

$S \to_p^* S'$, by (a) then S' is ACC (and contracted)

by (c), $S' = \dagger$. Thus $S \to \dagger$. □ .

EXTRAS

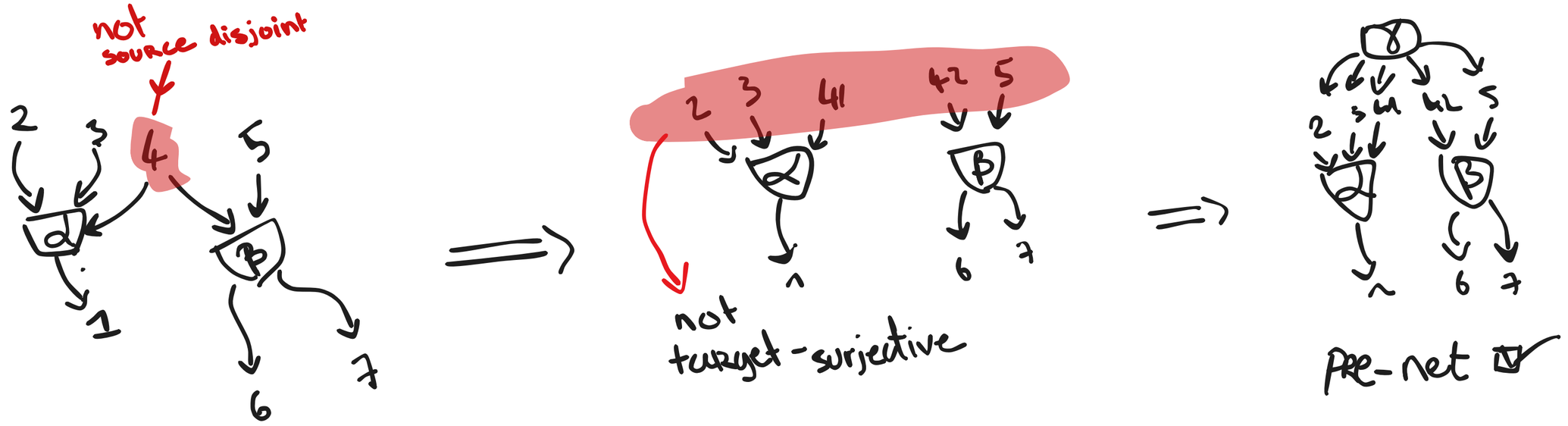
Properties of hypergraphs

(1) target surjective $\overset{def}{\iff} V = \bigcup_{e \in E} t(e)$

(2) target disjoint $\overset{def}{\iff} \forall e, e' \in E,\ t(e) \cap t(e') = \emptyset$

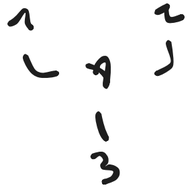
(3) source disjoint $\overset{def}{\iff} \forall e, e' \in E,\ s(e) \cap s(e') = \emptyset$

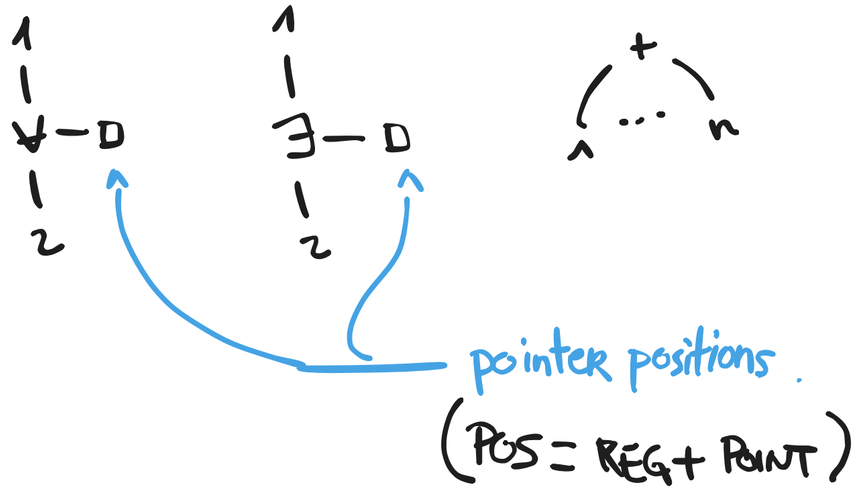
Pre-Net = target disjoint + source disjoint + target surjective Hypergraph



CONVENTION

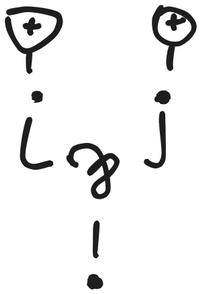
with LABELS = $\{\otimes, \parr, +, \forall, \exists\}$ and the following hyperedges (links)

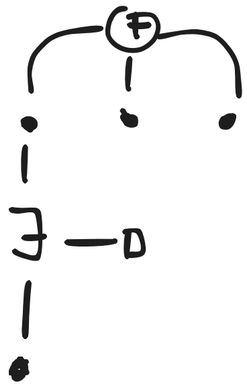




(POS = REG + POINT)

examples of pre-net



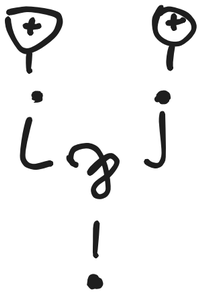


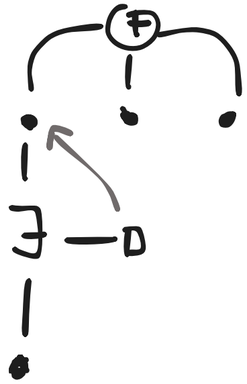
pointer link = link with { one source, one target } ↔ one of these two is a pointer position.

a pointer e is
- alive if s(e) is POINT-POS
- dead if s(e) isn't POINT-POS

- net = PRE-net + pointers + ζ
- alive net = pre-net + alive pointers + ζ

examples of ~~pre~~-net





regular position •

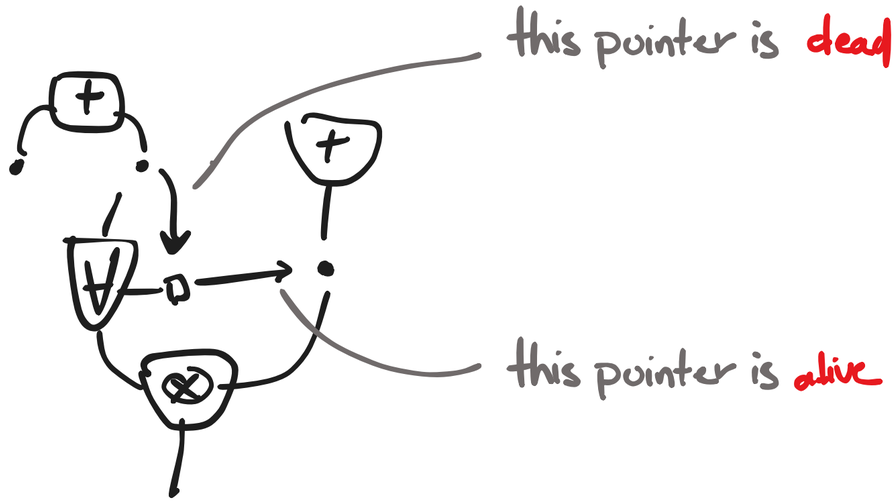
pointer position □

pointer →

hyperedge

alive pointer □→○

dead pointer ○→□



Additional convention

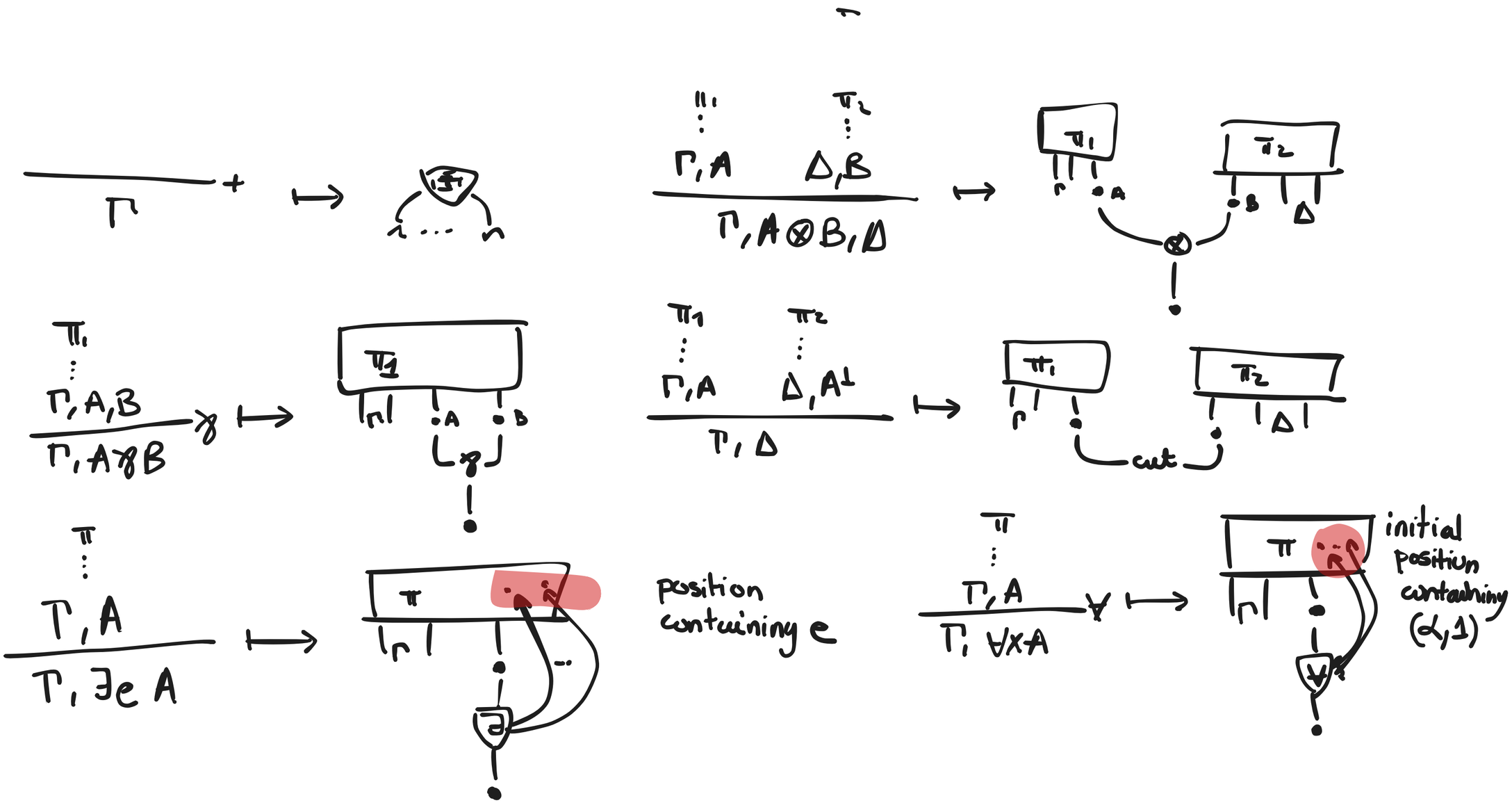
- ∀ pointers { □→•P OR •P→□ } imply P is initial

∀-pointer → =  OR

- ∃ pointers are only alive

- ∃ pointers □→•P are such that

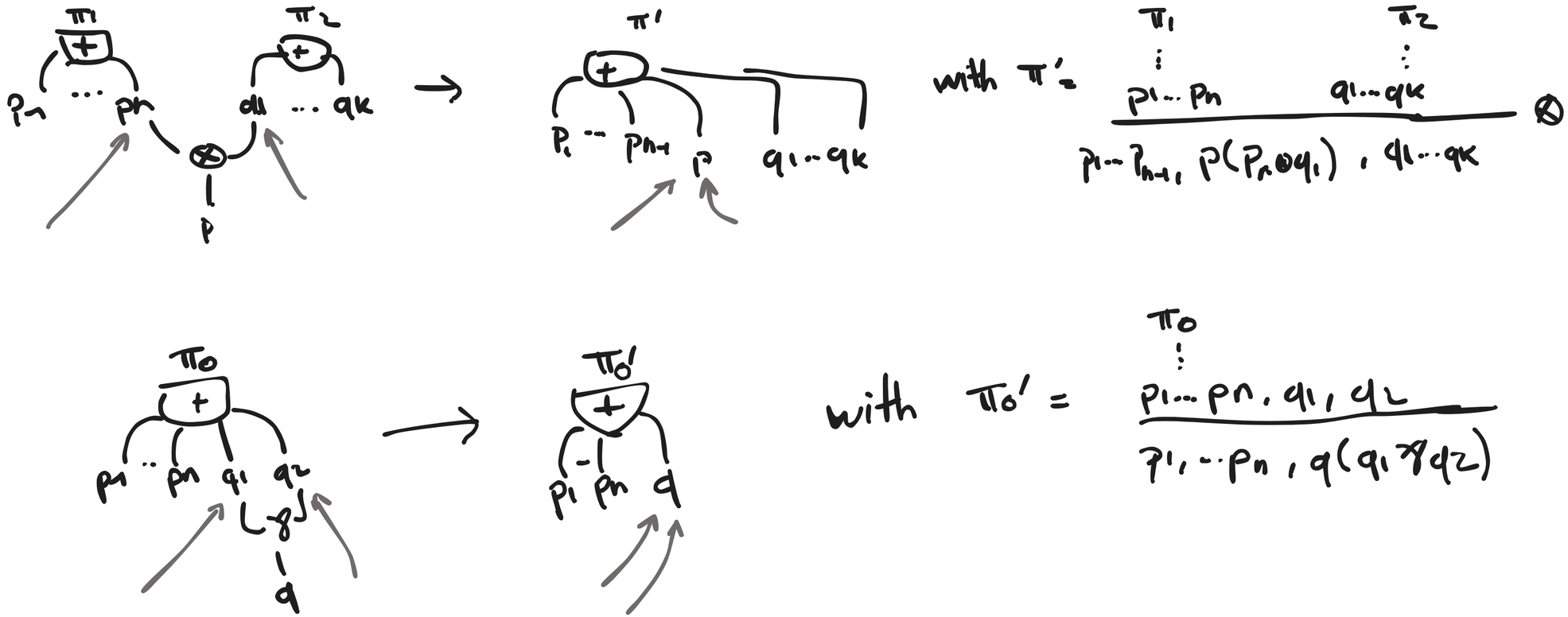
∃-pointer → =  OR 



- ALL POINTERS ARE ALIVE

PARSING (1/2)

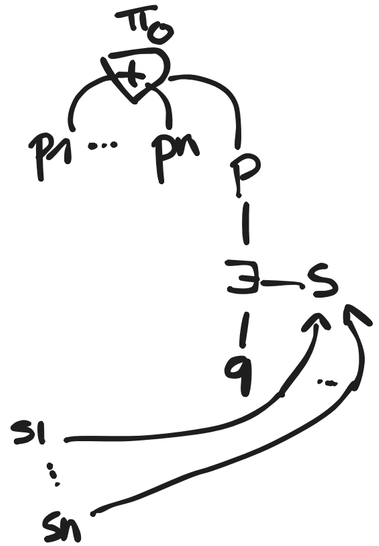
Key idea : parsing = inverse of desequentialisation



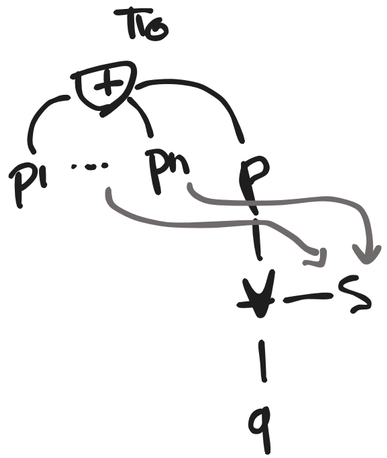
PARSING (2|2)

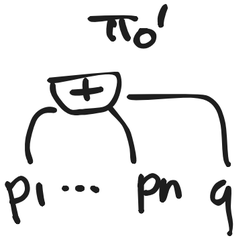
Key idea : Parsing = inverse of desequentialisation

Parsing is performed on REROUTED nets
(for confluence of $\to_p$)
only $\forall$-pointers remain



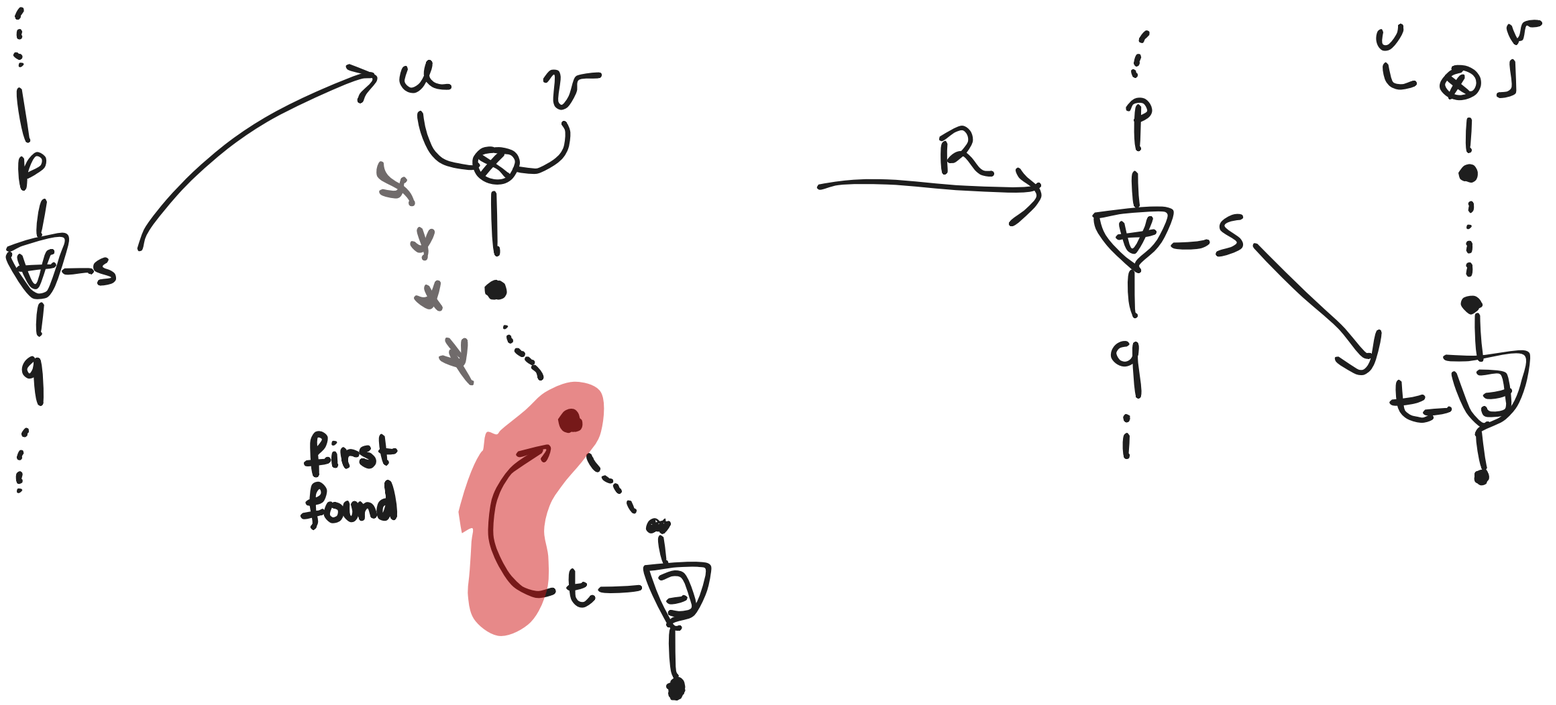
with $\pi_0' = \dfrac{\pi_0 \;\vdots\; p_1, \dots, p_n, p}{p_1, \dots, p_n, q\,(\exists c\, q)}$



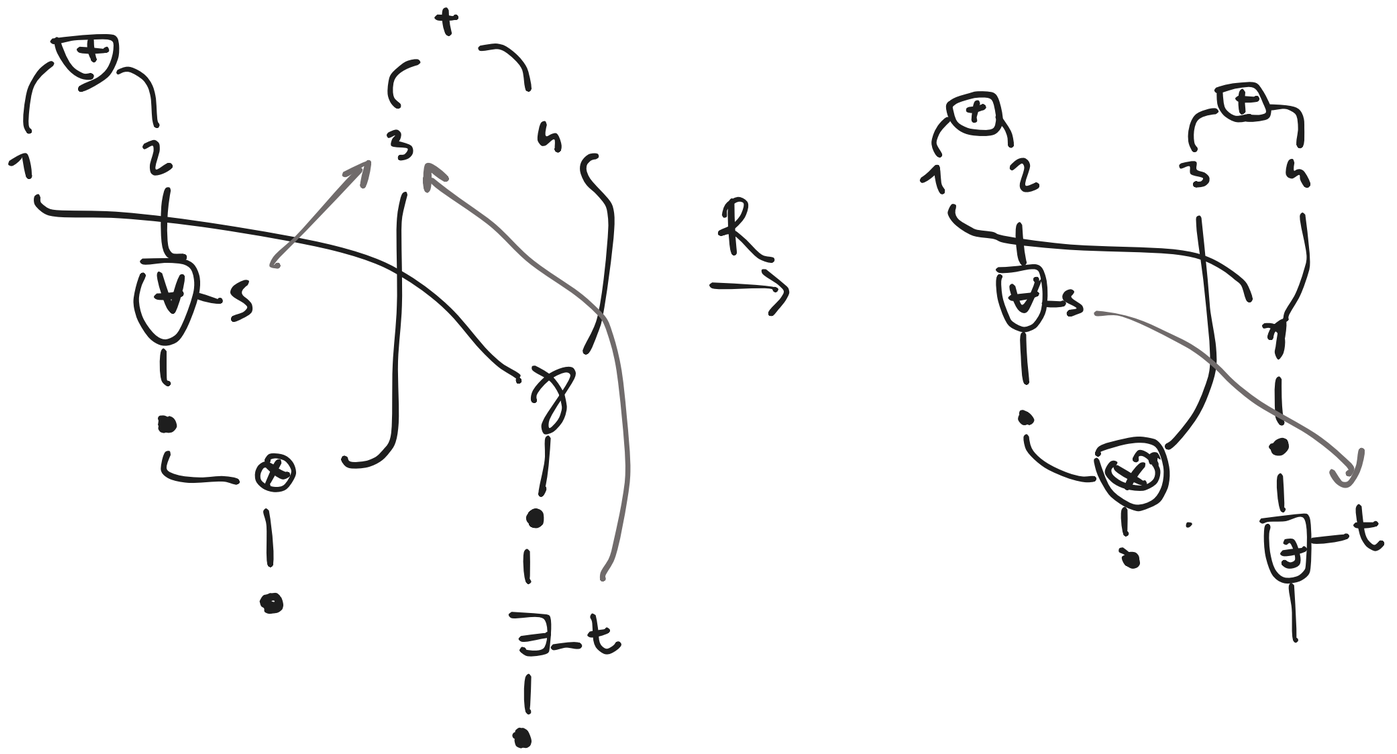


with $\pi_0' = \dfrac{\pi_0 \;\vdots\; p_1, \dots, p_n, p}{p_1, \dots, p_n, q\,(\forall_c p)}$

Rerouting of ∀-pointers



Rerouting of a net S = normalize for $\xrightarrow{R}$
+ forget ∃-pointers



cannot be parsed

Thank You.