Realizability in Cyclic Proof

Extracting Ordering Information for Infinite Descent

Reuben N. S. Rowe [1] James Brotherston [2]

UCL PPLV Seminar, Thursday 14th December 2017

[1]School of Computing, University of Kent, Canterbury, UK

[2]Department of Computer Science, UCL, London, UK

Outline

- Part I: Introduction to cyclic proofs
- Part II: Realizability results — how to extract semantic ordering information from cyclic proofs

Outline

- Part I: Introduction to cyclic proofs
  - What are they?
  - Some examples:
    - first order logic, separation logic, Hoare logic
  - General principles and results
- Part II: Realizability results — how to extract semantic ordering information from cyclic proofs

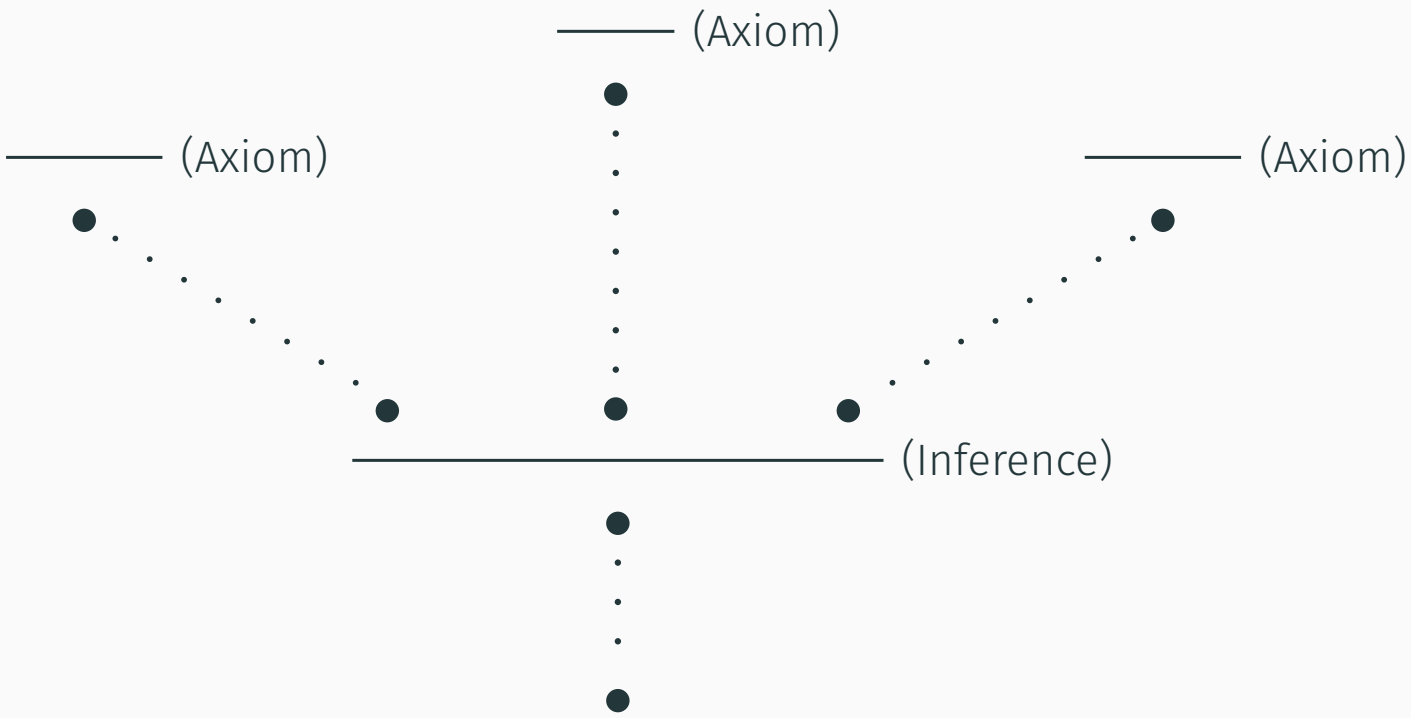
Outline

- Part I: Introduction to cyclic proofs
  - What are they?
  - Some examples:
    - first order logic, separation logic, Hoare logic
  - General principles and results
- Part II: Realizability results — how to extract semantic ordering information from cyclic proofs
  - ordering information = inclusion between semantic approximations
  - structural *realizability* property for cyclic entailment proofs
  - equivalence with weighted automata inclusion

Part I

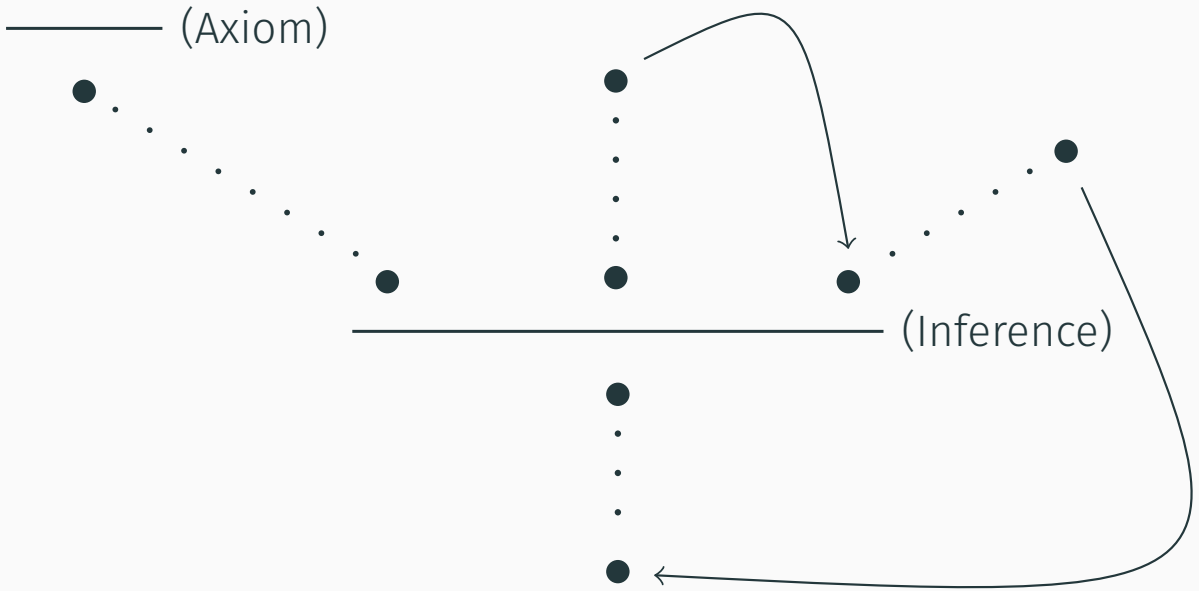
Cyclic Proofs

What is Cyclic Proof?



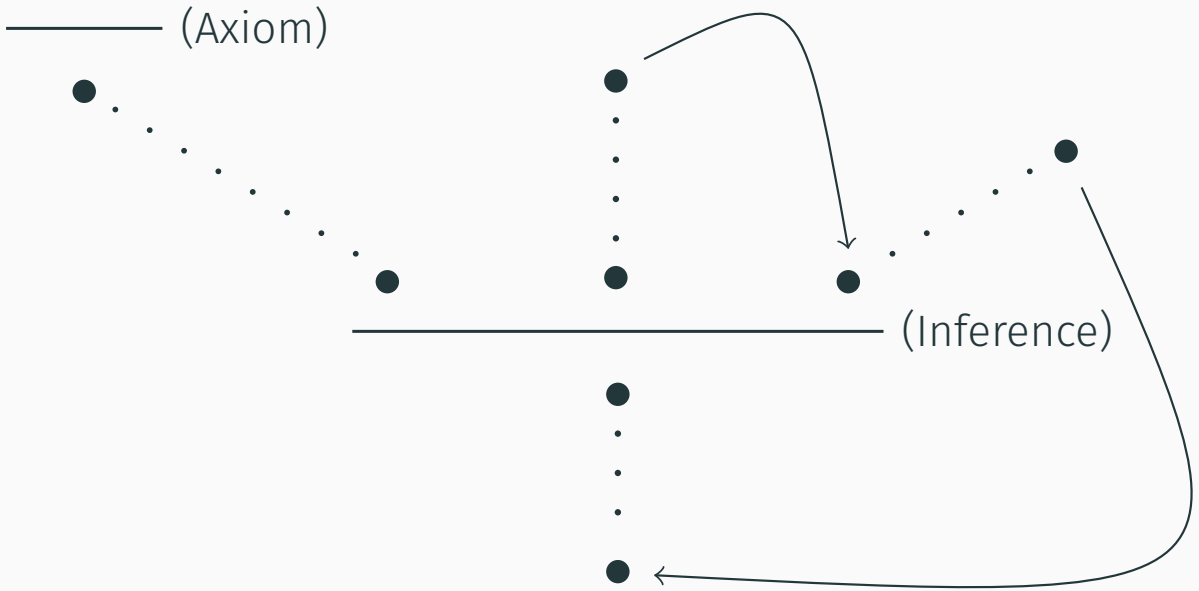
- We are all familiar with proofs as finite trees

What is Cyclic Proof?



- We are all familiar with proofs as finite trees
- But what if we allow proofs to be cyclic graphs instead?

What is Cyclic Proof?



- We are all familiar with proofs as finite trees
- But what if we allow proofs to be cyclic graphs instead?
- Cyclic proofs must satisfy a syntactic global trace property

Why Cyclic Proof?

- It subsumes standard induction
- It can help discover inductive hypotheses
- Termination arguments can often be extracted from cyclic proofs

First Order Logic: The Sequent Calculus $\mathcal{LK}$

$$(\vee\mathrm{L}):\ \frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta} \qquad (\vee\mathrm{R}):\ \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}$$

$$(\wedge\mathrm{L}):\ \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \qquad (\wedge\mathrm{R}):\ \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}$$

$$(\to\mathrm{L}):\ \frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \to B \vdash \Delta} \qquad (\neg\mathrm{R}):\ \frac{\Gamma, F \vdash \Delta}{\Gamma \vdash \neg F, \Delta}$$

$$(\neg\mathrm{L}):\ \frac{\Gamma \vdash F, \Delta}{\Gamma, \neg F \vdash \Delta} \qquad (\to\mathrm{R}):\ \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \to B, \Delta}$$

$$(=\mathrm{L}):\ \frac{\Gamma[u/x, t/y] \vdash \Delta[u/x, t/y]}{\Gamma[t/x, u/y], t = u \vdash \Delta[t/x, u/y]} \qquad (=\mathrm{R}):\ \frac{}{\Gamma \vdash t = t, \Delta}$$

First Order Logic: The Sequent Calculus $\mathcal{LK}$

$$\text{(Axiom)}: \frac{}{F \vdash F} \qquad \text{(Subst)}: \frac{\Gamma \vdash \Delta}{\Gamma\theta \vdash \Delta\theta} \qquad \text{(Cut)}: \frac{\Gamma \vdash F, \Delta \quad \Sigma, F \vdash \Pi}{\Gamma, \Sigma \vdash \Delta, \Pi}$$

$$\text{(WL)}: \frac{\Gamma \vdash \Delta}{\Gamma, F \vdash \Delta} \qquad \text{(WR)}: \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, F} \qquad \text{(PL)}: \frac{\Gamma, A, B, \Sigma \vdash \Delta}{\Gamma, B, A, \Sigma \vdash \Delta}$$

$$\text{(CL)}: \frac{\Gamma, F, F \vdash \Delta}{\Gamma, F \vdash \Delta} \qquad \text{(CR)}: \frac{\Gamma \vdash F, F, \Delta}{\Gamma \vdash F, \Delta} \qquad \text{(PR)}: \frac{\Gamma \vdash \Delta, A, B, \Sigma}{\Gamma \vdash \Delta, B, A, \Sigma}$$

First Order Logic with Inductive Predicate Definitions ($\mathrm{FOL}_{\mathrm{ID}}$)

- Assume signature with zero, successor, and equality
- Allow inductive predicate definitions, e.g.

$$\frac{}{\mathrm{N}\,0} \qquad \frac{\mathrm{N}\,x}{\mathrm{N}\,\mathrm{s}x} \qquad\qquad \frac{}{\mathrm{E}\,0} \qquad \frac{\mathrm{E}\,x}{\mathrm{O}\,\mathrm{s}x} \qquad \frac{\mathrm{O}\,x}{\mathrm{E}\,\mathrm{s}x}$$

First Order Logic with Inductive Predicate Definitions ($\mathrm{FOL}_{\mathrm{ID}}$)

- Assume signature with zero, successor, and equality
- Allow inductive predicate definitions, e.g.

$$\frac{}{\mathrm{N}\,0} \quad \frac{\mathrm{N}\,x}{\mathrm{N}\,sx} \qquad \frac{}{\mathrm{E}\,0} \quad \frac{\mathrm{E}\,x}{\mathrm{O}\,sx} \quad \frac{\mathrm{O}\,x}{\mathrm{E}\,sx}$$

- These induce unfolding rules for the sequent calculus, e.g.

$$\frac{\Gamma, t = 0 \vdash \Delta \quad \Gamma, t = sx, \mathrm{N}\,x \vdash \Delta}{\Gamma, \mathrm{N}\,t \vdash \Delta}\ (\text{Case N}) \quad (\text{where x fresh})$$

$$\frac{}{\Gamma \vdash \Delta, \mathrm{N}\,0}\ (\mathrm{NR}_1) \qquad \frac{\Gamma \vdash \Delta, \mathrm{N}\,t}{\Gamma \vdash \Delta, \mathrm{N}\,st}\ (\mathrm{NR}_2)$$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$

$$\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$

$$\dfrac{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x \qquad\qquad x = \mathsf{s}y, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(\text{Case N})$$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$

$$\dfrac{\dfrac{}{\vdash \mathsf{E}\,0, \mathsf{O}\,0}\,(=\mathsf{L})}{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x} \qquad \dfrac{}{x = \mathsf{s}y, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}$$
$$\dfrac{\qquad\qquad\qquad\qquad}{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(\text{Case N})$$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathsf{E}\,0, \mathsf{O}\,0}\,(\mathsf{ER}_1)}{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L}) \qquad x = \mathsf{s}y, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(\text{Case N})$$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathsf{E}\,0, \mathsf{O}\,0}\,(\mathsf{ER}_1)}{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L}) \qquad \dfrac{\mathsf{N}\,y \vdash \mathsf{E}\,sy, \mathsf{O}\,sy}{x = sy, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L})}{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(\text{Case N})$$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathsf{E}\,0, \mathsf{O}\,0}\,(\mathsf{ER}_1)}{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L}) \qquad \dfrac{\dfrac{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{O}\,sy}{\mathsf{N}\,y \vdash \mathsf{E}\,sy, \mathsf{O}\,sy}\,(\mathsf{ER}_2)}{x = sy, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L})}{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(\text{Case N})$$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$

$$\dfrac{\dfrac{}{\vdash \mathsf{E}\,0, \mathsf{O}\,0}\,(\mathsf{ER}_1)}{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L}) \qquad \dfrac{\dfrac{\dfrac{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{E}\,y}{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{O}\,sy}\,(\mathsf{OR}_1)}{\mathsf{N}\,y \vdash \mathsf{E}\,sy, \mathsf{O}\,sy}\,(\mathsf{ER}_2)}{x = sy, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L})}{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(\text{Case N})$$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$

$$\dfrac{\dfrac{}{\vdash \mathsf{E}\,0, \mathsf{O}\,0}\,(\mathsf{ER}_1)}{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L}) \qquad \dfrac{\dfrac{\dfrac{\mathsf{N}\,y \vdash \mathsf{E}\,y, \mathsf{O}\,y}{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{E}\,y}\,(\mathsf{PR})}{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{O}\,\mathsf{s}y}\,(\mathsf{OR}_1)}{\mathsf{N}\,y \vdash \mathsf{E}\,\mathsf{s}y, \mathsf{O}\,\mathsf{s}y}\,(\mathsf{ER}_2)}{x = \mathsf{s}y, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L})$$

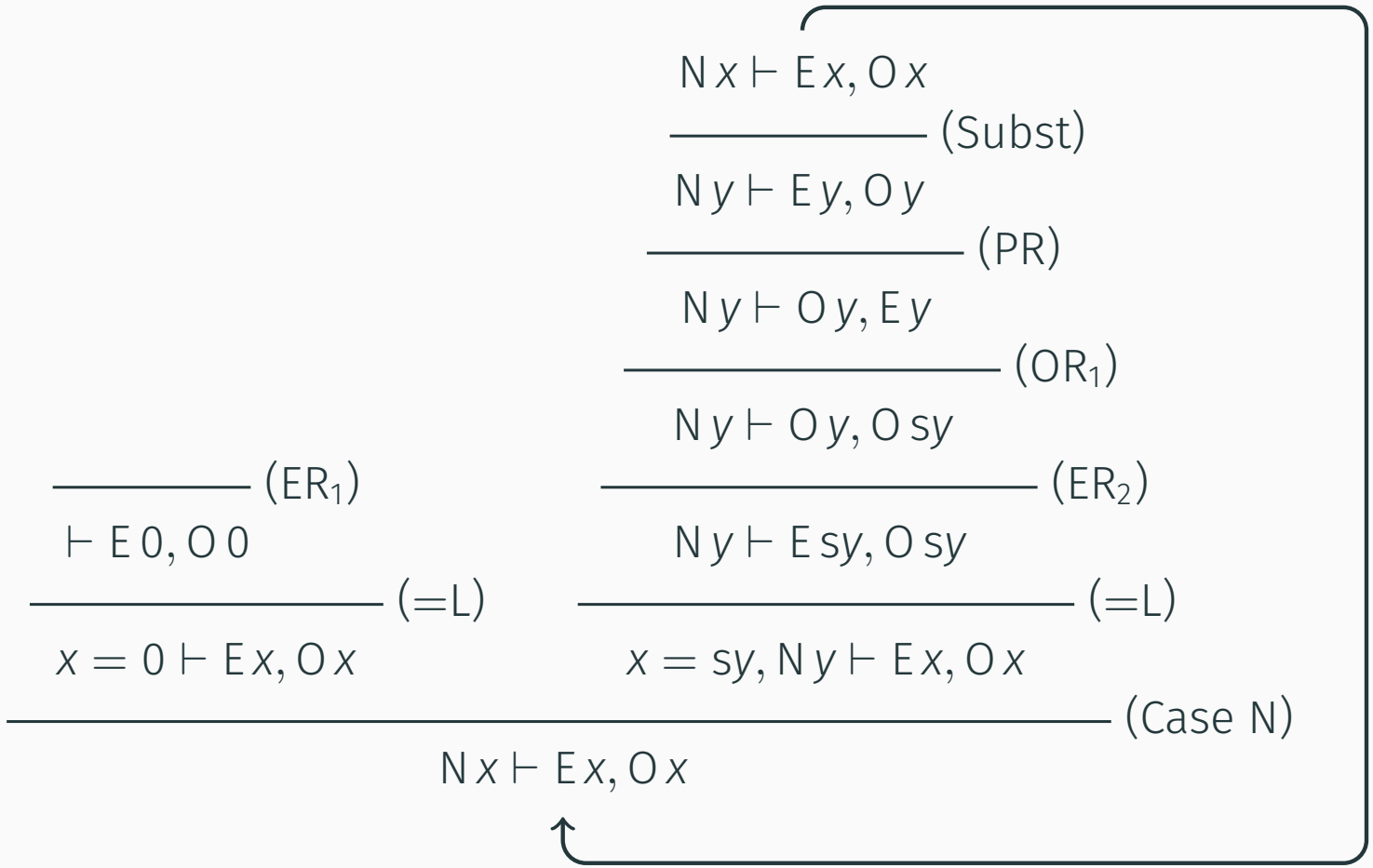
$$\dfrac{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x \qquad x = \mathsf{s}y, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(\mathsf{Case}\ \mathsf{N})$$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$

$$\dfrac{\dfrac{}{\vdash \mathsf{E}\,0, \mathsf{O}\,0}\,(\mathsf{ER}_1)}{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\!\mathsf{L}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}{\mathsf{N}\,y \vdash \mathsf{E}\,y, \mathsf{O}\,y}\,(\mathsf{Subst})}{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{E}\,y}\,(\mathsf{PR})}{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{O}\,sy}\,(\mathsf{OR}_1)}{\mathsf{N}\,y \vdash \mathsf{E}\,sy, \mathsf{O}\,sy}\,(\mathsf{ER}_2)}{x = sy, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\!\mathsf{L})$$

$$\dfrac{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x \qquad x = sy, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(\mathsf{Case\ N})$$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$



A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$

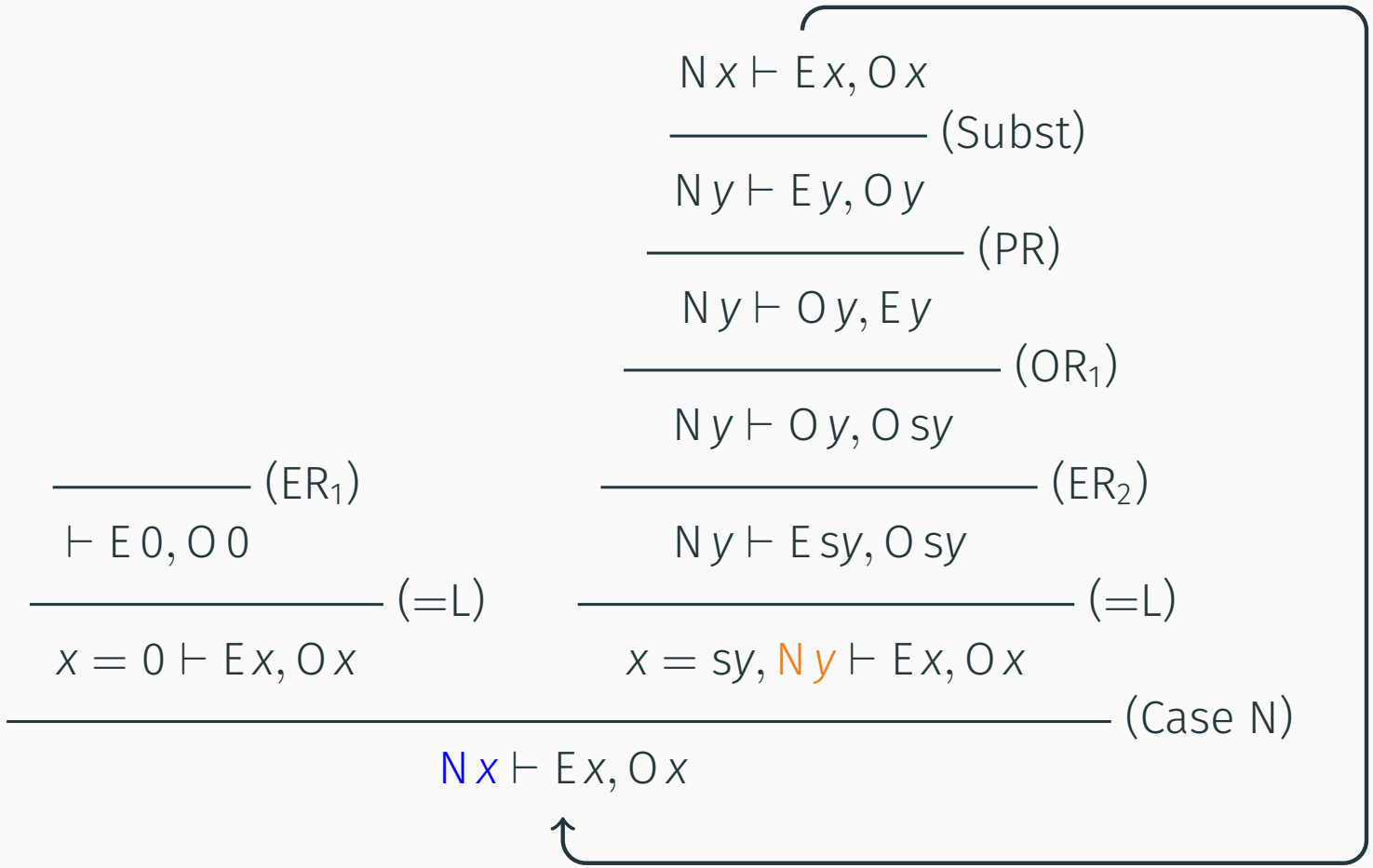
$$\dfrac{\dfrac{}{\vdash \mathsf{E}\,0, \mathsf{O}\,0}\,(\mathsf{ER}_1)}{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}{\mathsf{N}\,y \vdash \mathsf{E}\,y, \mathsf{O}\,y}\,(\mathsf{Subst})}{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{E}\,y}\,(\mathsf{PR})}{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{O}\,sy}\,(\mathsf{OR}_1)}{\mathsf{N}\,y \vdash \mathsf{E}\,sy, \mathsf{O}\,sy}\,(\mathsf{ER}_2)}{x = sy, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L})$$

$$\dfrac{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x \qquad x = sy, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(\mathsf{Case\ N})$$

- Suppose $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$ is not valid:

  $[\![x]\!]_{m_1}$

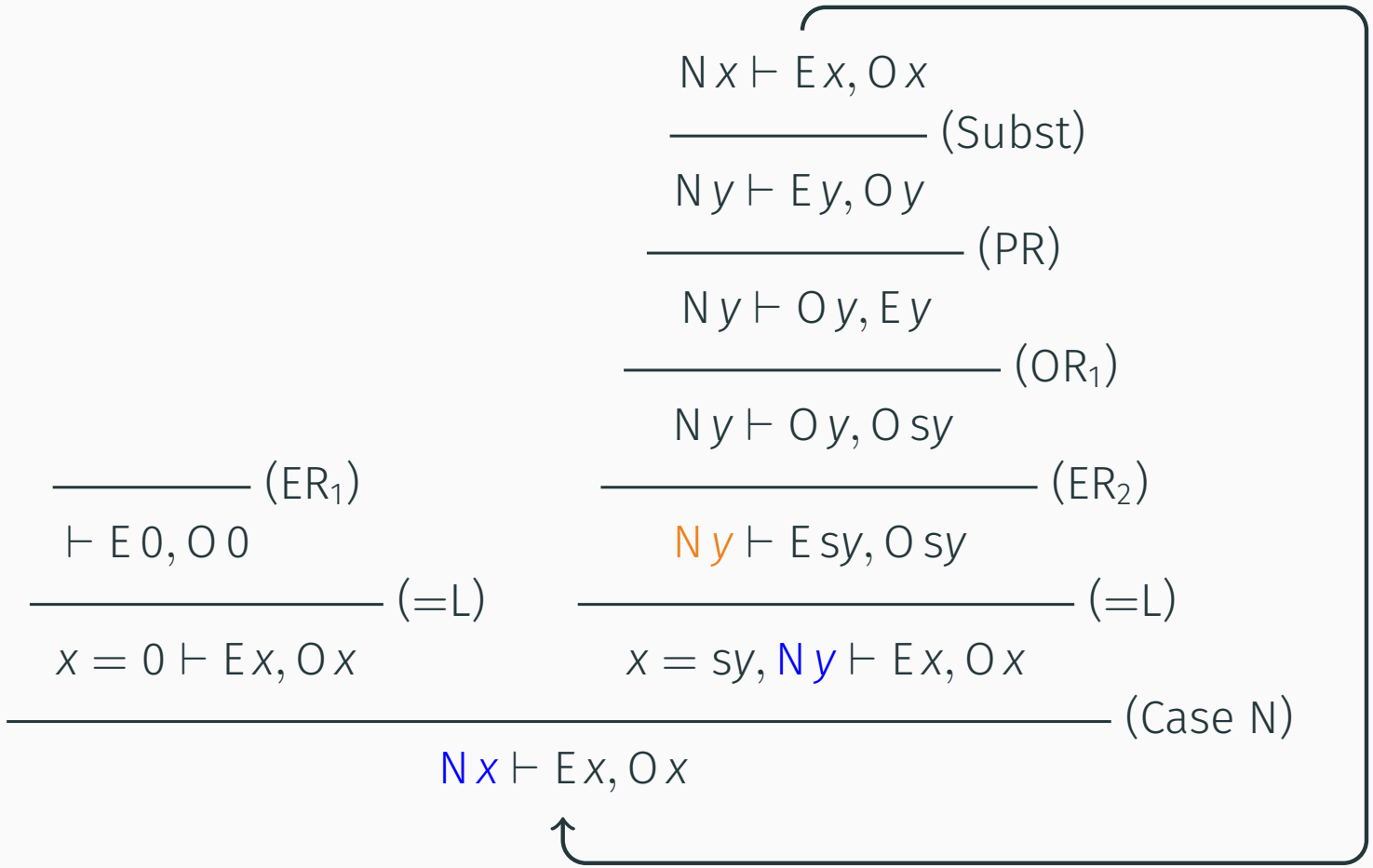
A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$



- Suppose $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$ is not valid:

  $[\![x]\!]_{m_1} > [\![y]\!]_{m_2}$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$



- Suppose $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$ is not valid:

  $[\![x]\!]_{m_1} > [\![y]\!]_{m_2} = [\![y]\!]_{m_3}$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$

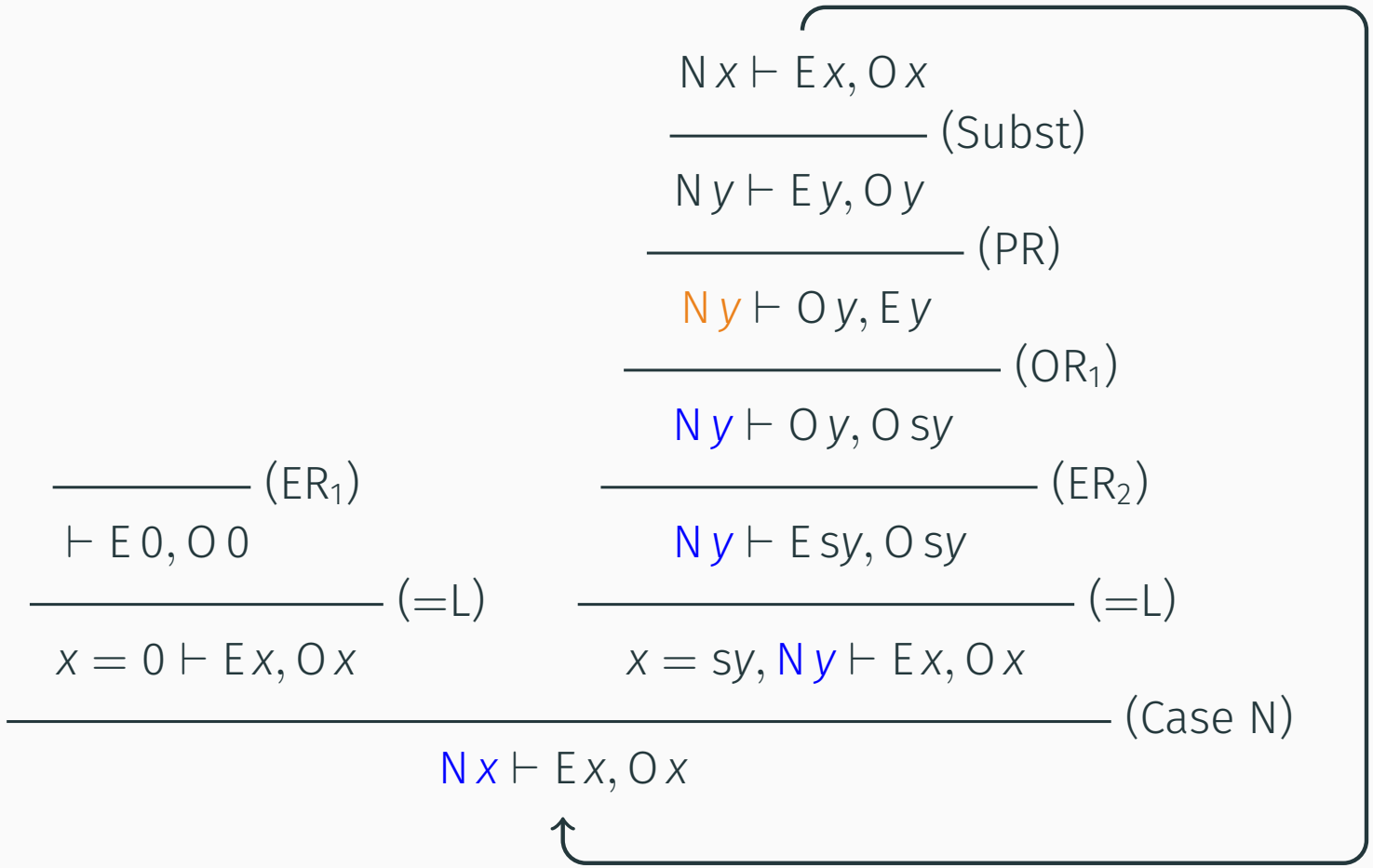
$$\dfrac{\dfrac{}{\vdash \mathsf{E}\,0, \mathsf{O}\,0}\,(\mathsf{ER}_1)}{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}{\mathsf{N}\,y \vdash \mathsf{E}\,y, \mathsf{O}\,y}\,(\mathsf{Subst})}{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{E}\,y}\,(\mathsf{PR})}{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{O}\,sy}\,(\mathsf{OR}_1)}{\mathsf{N}\,y \vdash \mathsf{E}\,sy, \mathsf{O}\,sy}\,(\mathsf{ER}_2)}{x = sy, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathsf{L})$$

$$\dfrac{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x \qquad x = sy, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(\mathsf{Case\ N})$$

- Suppose $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$ is not valid:

$$[\![x]\!]_{m_1} > [\![y]\!]_{m_2} = [\![y]\!]_{m_3} = [\![y]\!]_{m_4}$$

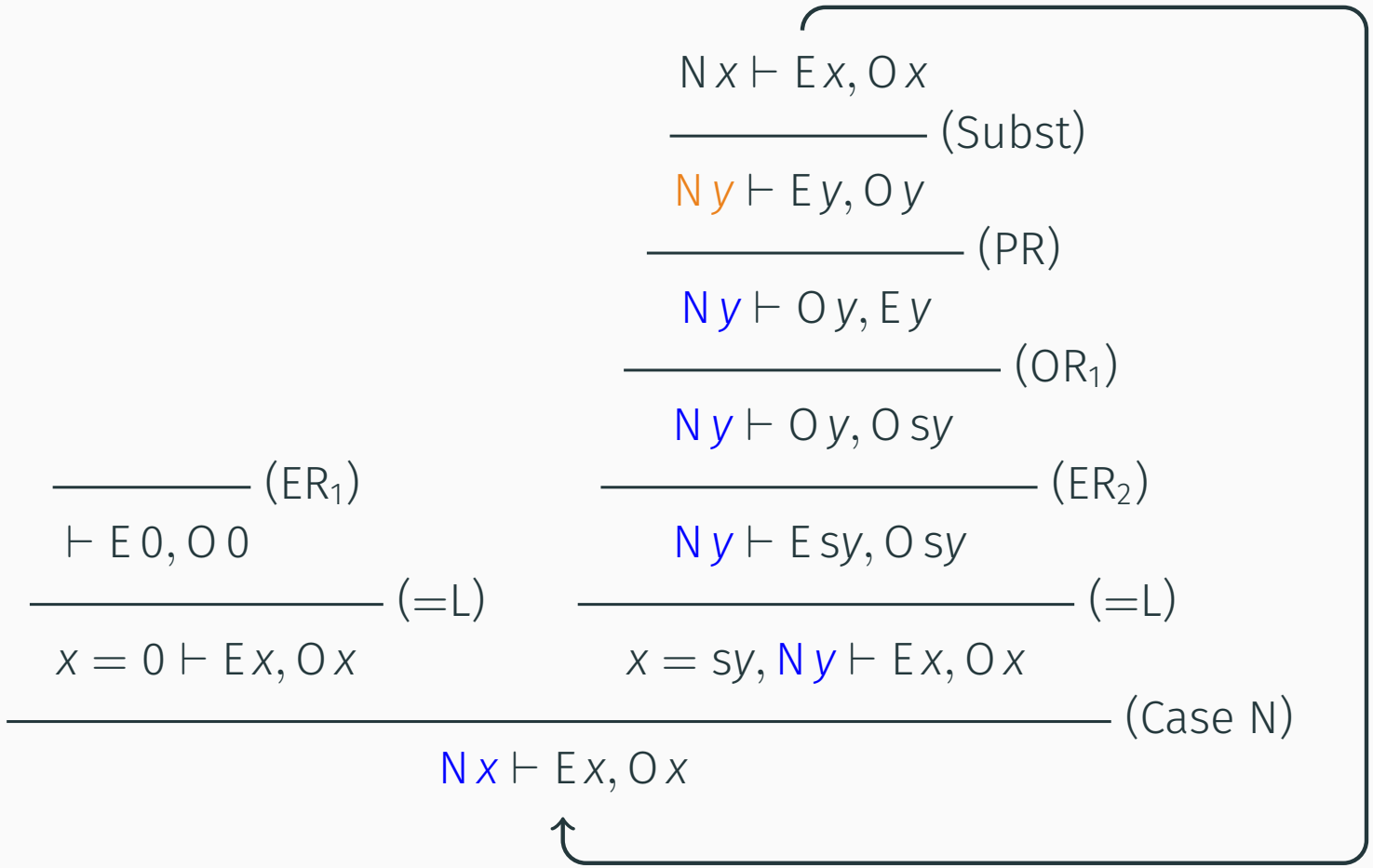
A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$



- Suppose $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$ is not valid:

$$[\![x]\!]_{m_1} > [\![y]\!]_{m_2} = [\![y]\!]_{m_3} = [\![y]\!]_{m_4} = [\![y]\!]_{m_5}$$

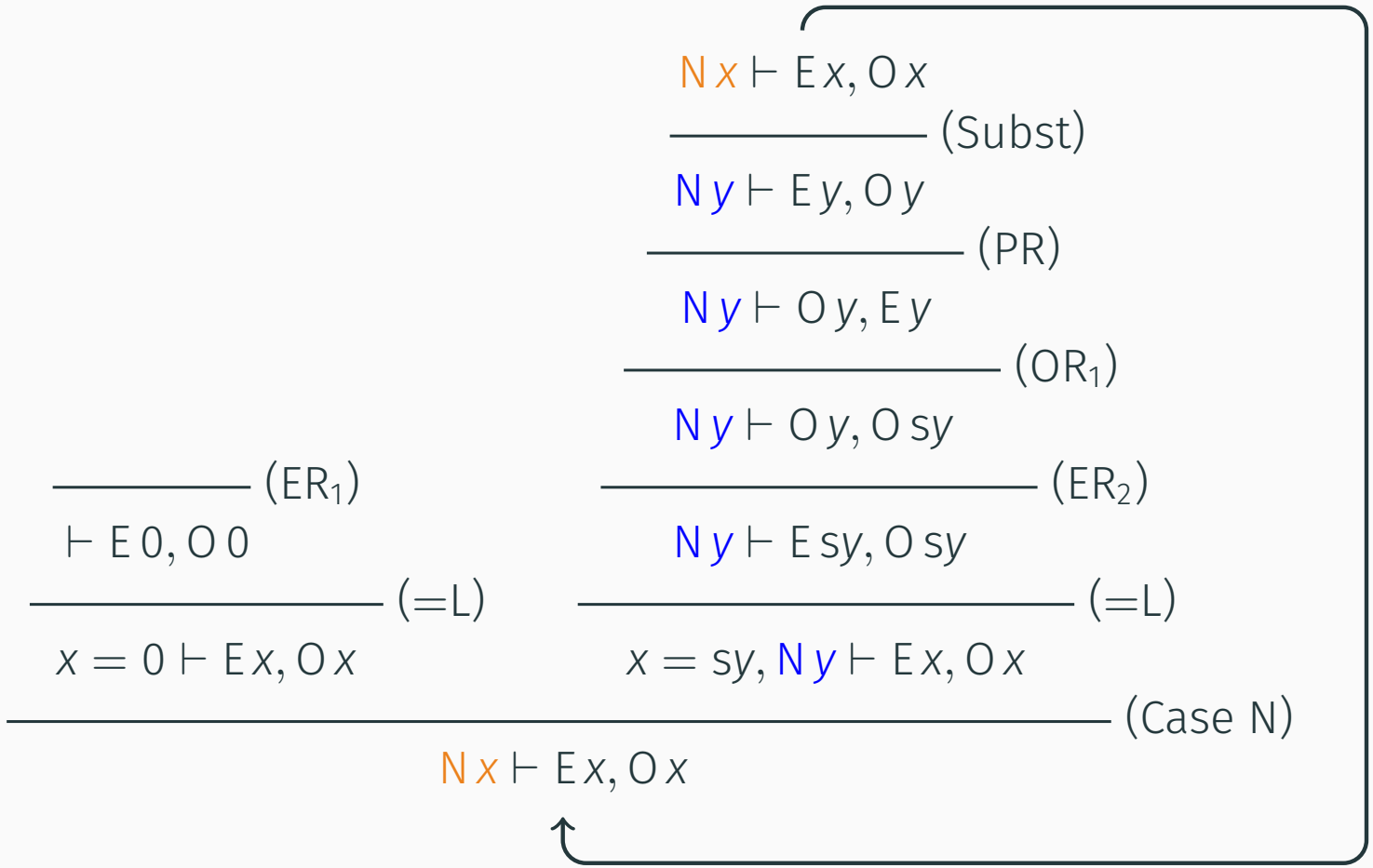
A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$



- Suppose $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$ is not valid:

$$[\![x]\!]_{m_1} > [\![y]\!]_{m_2} = [\![y]\!]_{m_3} = [\![y]\!]_{m_4} = [\![y]\!]_{m_5} = [\![y]\!]_{m_6}$$

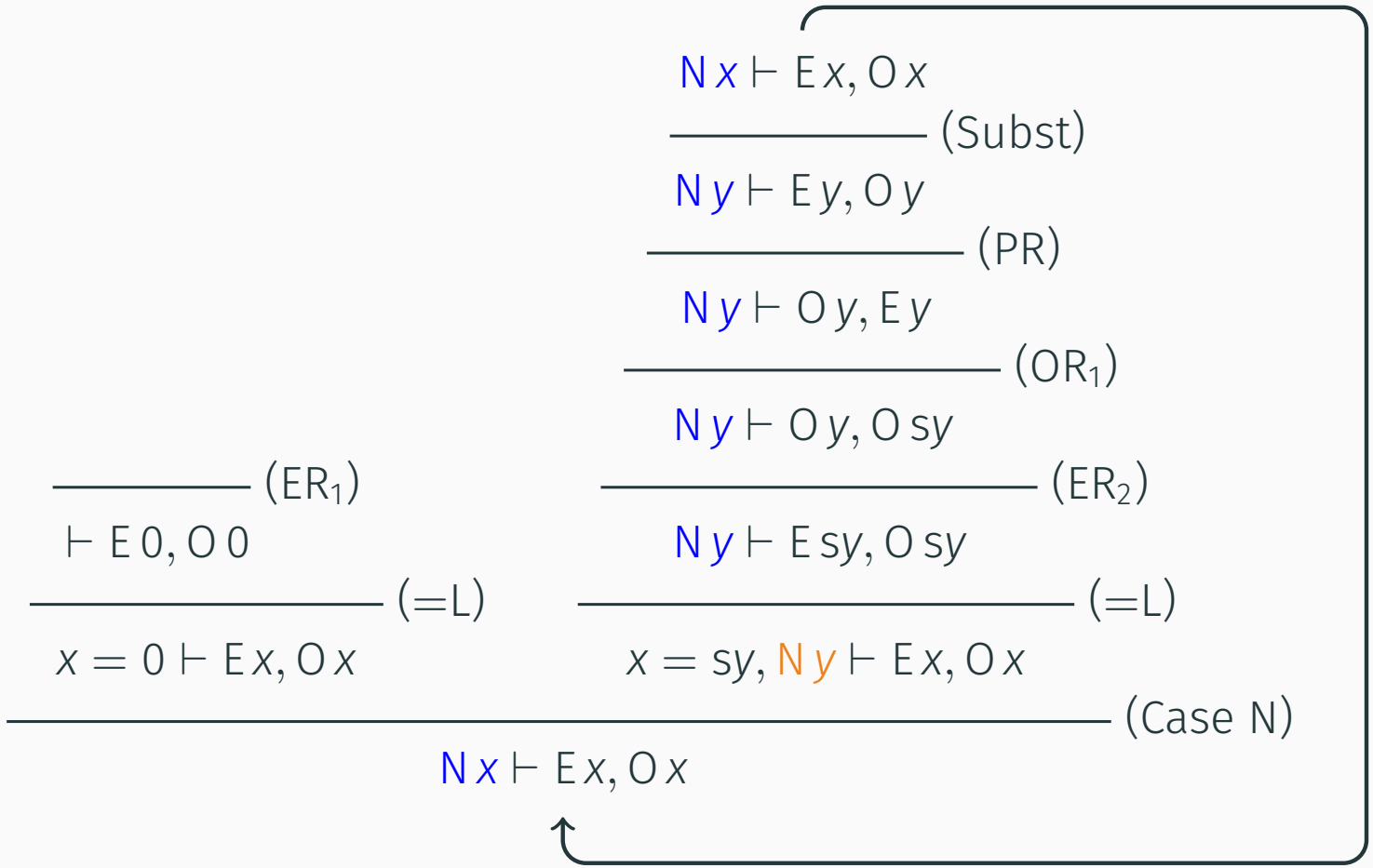
A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$



- Suppose $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$ is not valid:

$$[\![x]\!]_{m_1} > [\![y]\!]_{m_2} = [\![y]\!]_{m_3} = [\![y]\!]_{m_4} = [\![y]\!]_{m_5} = [\![y]\!]_{m_6} = [\![x]\!]_{m_7}$$

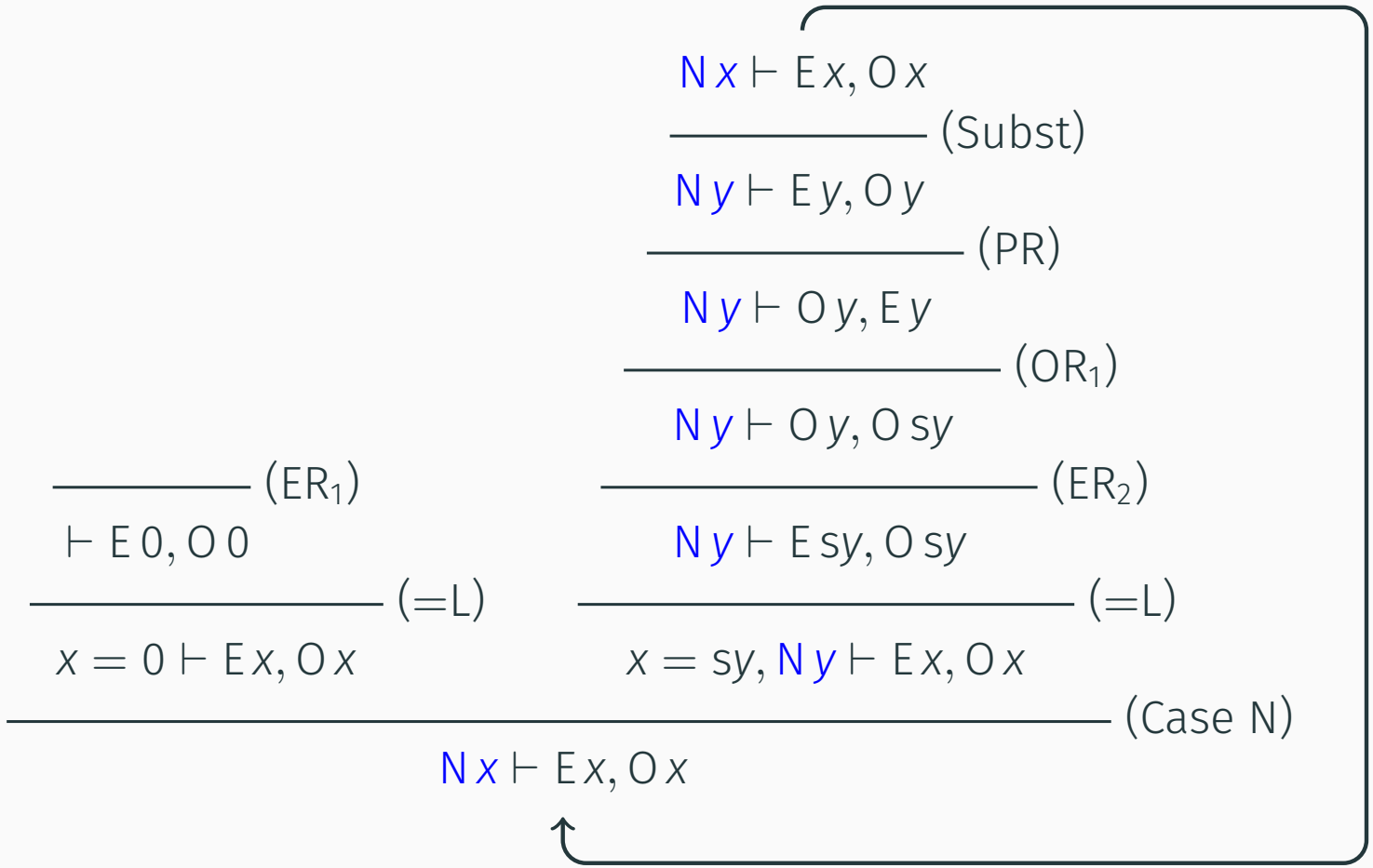
A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$



- Suppose $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$ is not valid:

$$[\![x]\!]_{m_1} > [\![y]\!]_{m_2} = [\![y]\!]_{m_3} = [\![y]\!]_{m_4} = [\![y]\!]_{m_5} = [\![y]\!]_{m_6} = [\![x]\!]_{m_7} > [\![y]\!]_{m_8} \cdots$$

A Cyclic Proof of $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$



- Suppose $\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$ is not valid:

$$n_1 > n_2 > n_3 > \ldots \qquad (n_i \in \mathbb{N} \text{ for all } i)$$

Example: Separation Logic

- Separation Logic incorporates formulas for representing heap memory:

Example: Separation Logic

- Separation Logic incorporates formulas for representing heap memory:
  - **emp** denotes the empty heap

Example: Separation Logic

- Separation Logic incorporates formulas for representing heap memory:
  - **emp** denotes the empty heap
  - $x \mapsto \vec{v}$ is the single-cell heap containing values $\vec{v}$ at memory location x

Example: Separation Logic

- Separation Logic incorporates formulas for representing heap memory:
  - **emp** denotes the empty heap
  - $x \mapsto \vec{v}$ is the single-cell heap containing values $\vec{v}$ at memory location x
  - $F * G$ denotes a heap h that can be split into disjoint sub-heaps h_1 and h_2 which model F and G respectively

Example: Separation Logic

- Separation Logic incorporates formulas for representing heap memory:
  - **emp** denotes the empty heap
  - $x \mapsto \vec{v}$ is the single-cell heap containing values $\vec{v}$ at memory location x
  - $F * G$ denotes a heap h that can be split into disjoint sub-heaps h_1 and h_2 which model F and G respectively
- Inductive predicates now represent data-structures, e.g. linked-list segments:

$$\frac{x = y \wedge \mathbf{emp}}{\mathsf{ls}(x, y)} \qquad \frac{x \mapsto z * \mathsf{ls}(z, y)}{\mathsf{ls}(x, y)}$$

A Cyclic Proof of List Segment Concatenation

$$\mathsf{ls}(x, y) * \mathsf{ls}(y, z) \vdash \mathsf{ls}(x, z)$$

A Cyclic Proof of List Segment Concatenation

$$\dfrac{(x = y \wedge \mathsf{emp}) * \mathsf{ls}(y, z) \vdash \mathsf{ls}(x, z) \qquad x \mapsto v * \mathsf{ls}(v, y) * \mathsf{ls}(y, z) \vdash \mathsf{ls}(x, z)}{\mathsf{ls}(x, y) * \mathsf{ls}(y, z) \vdash \mathsf{ls}(x, z)}\ (\text{Case } \mathsf{ls})$$

A Cyclic Proof of List Segment Concatenation

$$\dfrac{\dfrac{\mathsf{emp} * \mathsf{ls}(x, z) \vdash \mathsf{ls}(x, z)}{(x = y \wedge \mathsf{emp}) * \mathsf{ls}(y, z) \vdash \mathsf{ls}(x, z)}\,(=\mathrm{L}) \qquad \vdots \qquad x \mapsto v * \mathsf{ls}(v, y) * \mathsf{ls}(y, z) \vdash \mathsf{ls}(x, z)}{\mathsf{ls}(x, y) * \mathsf{ls}(y, z) \vdash \mathsf{ls}(x, z)}\,(\text{Case } \mathsf{ls})$$

A Cyclic Proof of List Segment Concatenation

$$\dfrac{\dfrac{\dfrac{\mathsf{ls}(x,z) \vdash \mathsf{ls}(x,z)}{\mathsf{emp} * \mathsf{ls}(x,z) \vdash \mathsf{ls}(x,z)}\,(\equiv)}{(x = y \wedge \mathsf{emp}) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(=\mathsf{L}) \qquad \vdots \qquad x \mapsto v * \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}{\mathsf{ls}(x,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(\text{Case } \mathsf{ls})$$

A Cyclic Proof of List Segment Concatenation

$$\dfrac{\dfrac{\dfrac{}{\mathsf{ls}(x,z) \vdash \mathsf{ls}(x,z)}\,(\mathsf{Id})}{\mathbf{emp} * \mathsf{ls}(x,z) \vdash \mathsf{ls}(x,z)}\,(\equiv)}{(x = y \wedge \mathbf{emp}) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(=\mathsf{L}) \qquad \vdots \qquad x \mapsto v * \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}{\mathsf{ls}(x,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(\text{Case } \mathbf{ls})$$

A Cyclic Proof of List Segment Concatenation

$$\dfrac{\dfrac{\dfrac{}{\mathsf{ls}(x,z) \vdash \mathsf{ls}(x,z)}\,(\mathsf{Id})}{\mathsf{emp} * \mathsf{ls}(x,z) \vdash \mathsf{ls}(x,z)}\,(\equiv)}{(x = y \wedge \mathsf{emp}) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(=\mathsf{L}) \qquad \dfrac{x \mapsto v * \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash x \mapsto v * \mathsf{ls}(v,z)}{x \mapsto v * \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(\mathsf{ls}\mathsf{R}_2)}{\mathsf{ls}(x,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(\text{Case } \mathsf{ls})$$

A Cyclic Proof of List Segment Concatenation

$$\dfrac{\dfrac{\dfrac{}{\mathsf{ls}(x,z) \vdash \mathsf{ls}(x,z)}\,(\mathsf{Id})}{\mathsf{emp} * \mathsf{ls}(x,z) \vdash \mathsf{ls}(x,z)}\,(\equiv)}{(x = y \wedge \mathsf{emp}) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(=\mathsf{L}) \qquad \dfrac{\dfrac{x \mapsto v \vdash x \mapsto v \qquad \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(v,z)}{x \mapsto v * \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash x \mapsto v * \mathsf{ls}(v,z)}\,(*)}{x \mapsto v * \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(\mathsf{ls}\mathsf{R}_2)$$

$$\dfrac{\vdots \qquad\qquad \vdots}{\mathsf{ls}(x,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(\text{Case } \mathsf{ls})$$

A Cyclic Proof of List Segment Concatenation

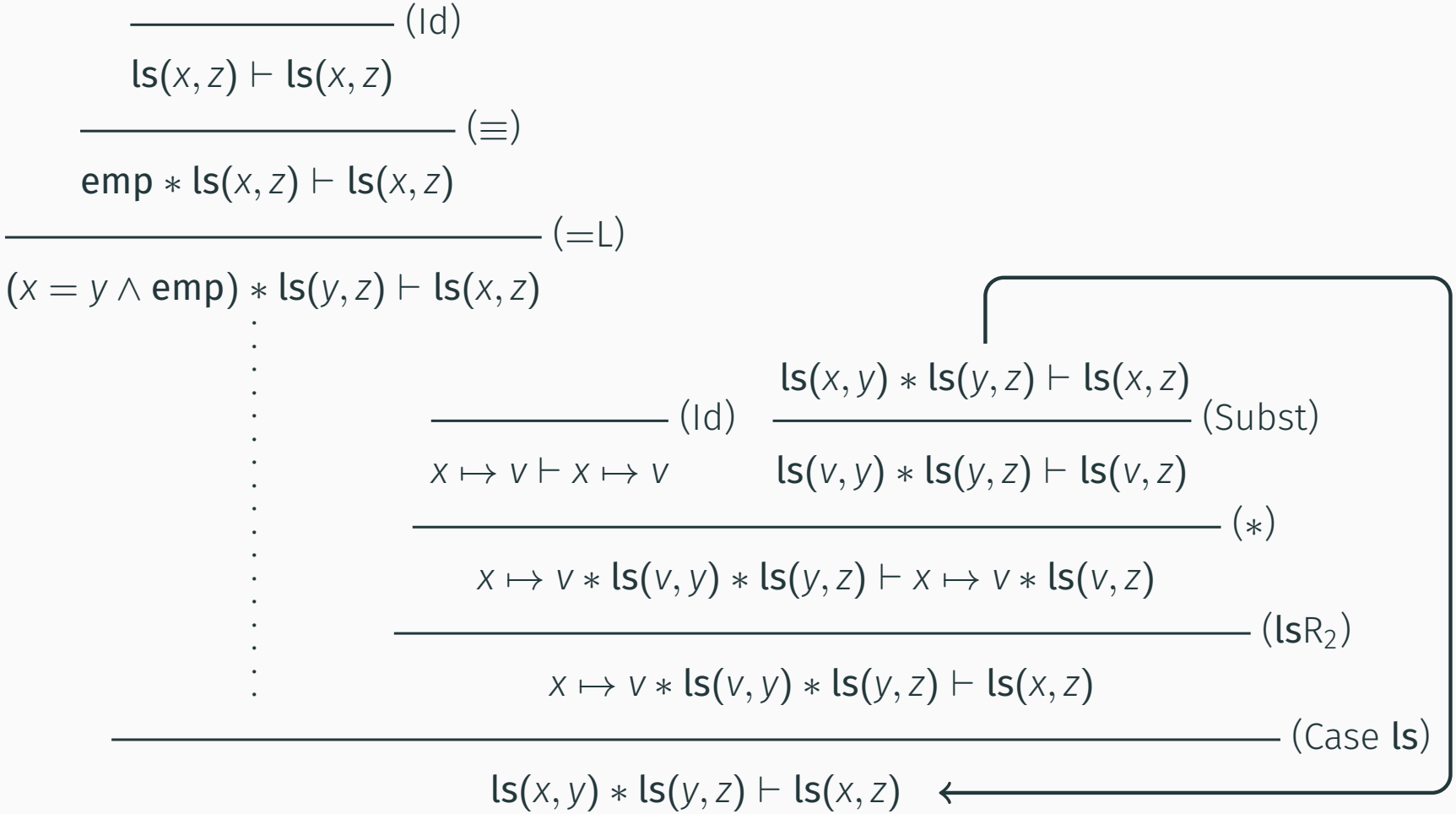
$$\dfrac{\dfrac{\dfrac{}{\mathsf{ls}(x,z) \vdash \mathsf{ls}(x,z)}\,(\mathsf{Id})}{\mathbf{emp} * \mathsf{ls}(x,z) \vdash \mathsf{ls}(x,z)}\,(\equiv)}{(x = y \wedge \mathbf{emp}) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(=\mathsf{L}) \qquad \dfrac{\dfrac{\dfrac{}{x \mapsto v \vdash x \mapsto v}\,(\mathsf{Id}) \qquad \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(v,z)}{x \mapsto v * \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash x \mapsto v * \mathsf{ls}(v,z)}\,(*)}{x \mapsto v * \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(\mathsf{ls}\mathrm{R}_2)$$

$$\dfrac{(x = y \wedge \mathbf{emp}) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z) \qquad x \mapsto v * \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}{\mathsf{ls}(x,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(\text{Case } \mathsf{ls})$$

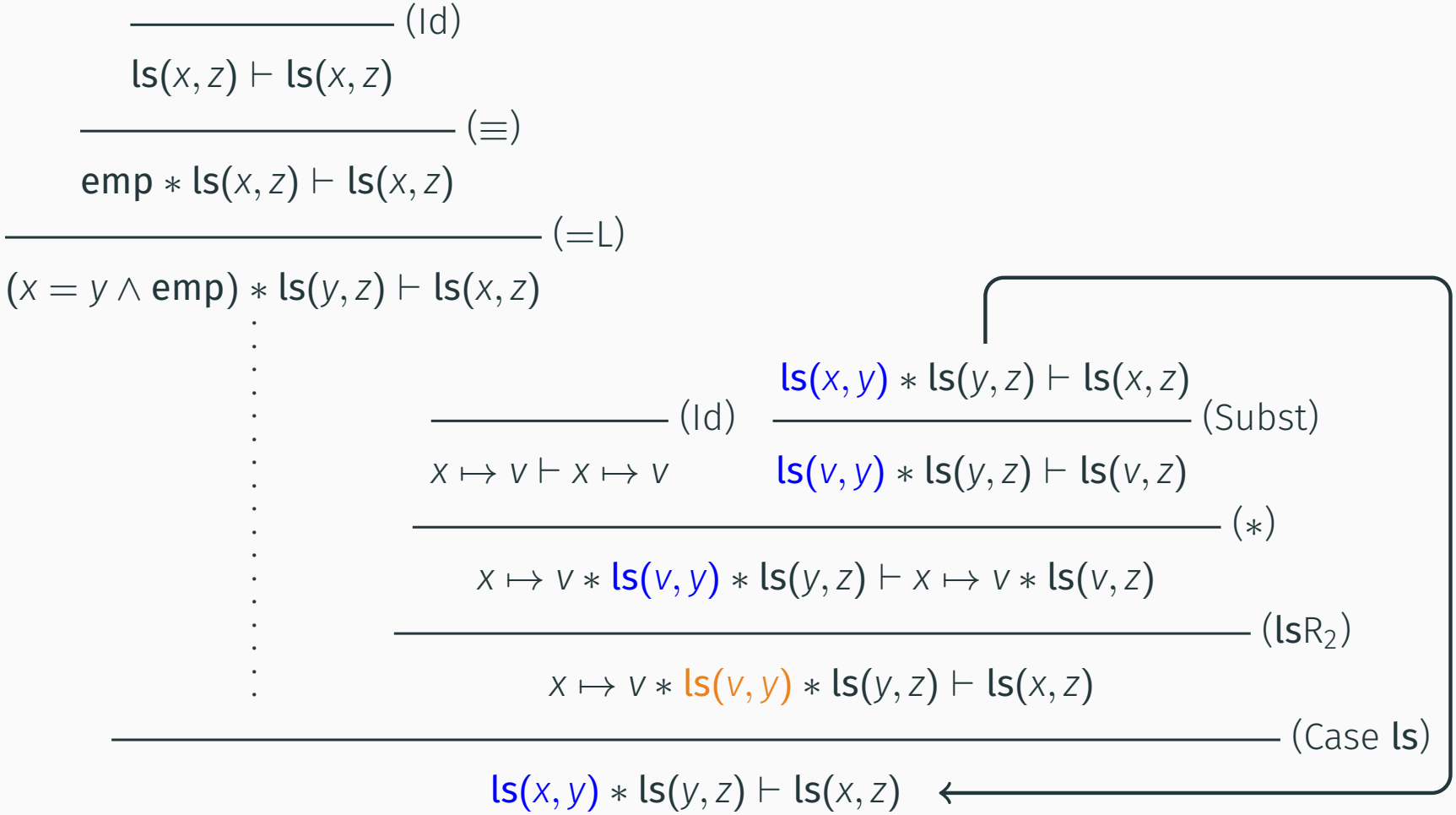
A Cyclic Proof of List Segment Concatenation

$$\dfrac{\dfrac{\dfrac{}{\mathsf{ls}(x,z) \vdash \mathsf{ls}(x,z)}\,(\mathrm{Id})}{\mathbf{emp} * \mathsf{ls}(x,z) \vdash \mathsf{ls}(x,z)}\,(\equiv)}{(x = y \wedge \mathbf{emp}) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(=\mathrm{L}) \qquad \vdots \qquad \dfrac{\dfrac{\dfrac{}{x \mapsto v \vdash x \mapsto v}\,(\mathrm{Id}) \quad \dfrac{\mathsf{ls}(x,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}{\mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(v,z)}\,(\mathrm{Subst})}{x \mapsto v * \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash x \mapsto v * \mathsf{ls}(v,z)}\,(*)}{x \mapsto v * \mathsf{ls}(v,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(\mathsf{ls}\mathrm{R}_2)}{\mathsf{ls}(x,y) * \mathsf{ls}(y,z) \vdash \mathsf{ls}(x,z)}\,(\text{Case } \mathsf{ls})$$

A Cyclic Proof of List Segment Concatenation



A Cyclic Proof of List Segment Concatenation



A Simple Imperative Language

(Terms)	$t ::= \texttt{nil} \mid x$	
(Boolean Expressions)	$B ::= t\texttt{=}t \mid t\texttt{!=}t$	
(Programs)	$C ::= \varepsilon$	(stop)
	$\mid \quad x\texttt{:=}t\texttt{;}C$	(assignment)
	$\mid \quad x\texttt{:=[}y\texttt{];}C \mid \texttt{[}x\texttt{]:=}y\texttt{;}C$	(load/store)
	$\mid \quad \texttt{free(}x\texttt{);}C \mid x\texttt{:=new;}C$	(de/allocate)
	$\mid \quad \texttt{if(}B\texttt{)then\{}C\texttt{\};}C$	(conditional)
	$\mid \quad \texttt{while(}B\texttt{)do\{}C\texttt{\};}C$	(loop)

A Simple Imperative Language

$$
\begin{array}{rll}
\text{(Terms)} & t ::= \texttt{nil} \mid x & \\
\text{(Boolean Expressions)} & B ::= t\texttt{=}t \mid t\texttt{!=}t & \\
\text{(Programs)} & C ::= \varepsilon & \text{(stop)} \\
& \mid \quad x\texttt{:=}t\texttt{;}C & \text{(assignment)} \\
& \mid \quad x\texttt{:=[}y\texttt{];}C \mid \texttt{[}x\texttt{]:=}y\texttt{;}C & \text{(load/store)} \\
& \mid \quad \texttt{free(}x\texttt{);}C \mid x\texttt{:=new;}C & \text{(de/allocate)} \\
& \mid \quad \texttt{if(}B\texttt{)then\{}C\texttt{\};}C & \text{(conditional)} \\
& \mid \quad \texttt{while(}B\texttt{)do\{}C\texttt{\};}C & \text{(loop)}
\end{array}
$$

- The following program deallocates a linked list

```
while(x!=nil)do{y:=[x];free(x);x=y}
```

Program Verification by Symbolic Execution

- We use Hoare logic for proving triples $\{P\}\, C\, \{Q\}$ using Separation Logic as an assertion language

Program Verification by Symbolic Execution

- We use Hoare logic for proving triples $\{P\}\, C\, \{Q\}$ using Separation Logic as an assertion language
- Program commands are executed symbolically by the proof rules, e.g.

$$\text{(load):}\quad \frac{\{x = v[x'/x] \land (P * y \mapsto v)[x'/x]\}\, C\, \{Q\}}{\{P * y \mapsto v\}\, x\texttt{:=}[y]\texttt{;}\, C\, \{Q\}}\ (x' \text{ fresh})$$

Program Verification by Symbolic Execution

- We use Hoare logic for proving triples $\{P\}\, C\, \{Q\}$ using Separation Logic as an assertion language
- Program commands are executed symbolically by the proof rules, e.g.

$$\text{(load):}\quad \frac{\{x = v[x'/x] \land (P * y \mapsto v)[x'/x]\}\, C\, \{Q\}}{\{P * y \mapsto v\}\, x\texttt{:=[}y\texttt{];}\, C\, \{Q\}}\ (x' \text{ fresh})$$

$$\text{(free):}\quad \frac{\{P\}\, C\, \{Q\}}{\{P * x \mapsto v\}\, \texttt{free(}x\texttt{);}\, C\, \{Q\}}$$

Handling Loops in Cyclic Proofs

- The standard Hoare rule for handling `while` loops:

$$\frac{\{B \wedge P\}\, C_1\, \{P\} \quad \{\neg B \wedge P\}\, C_2\, \{Q\}}{\{P\}\, \texttt{while}\,(B)\,\texttt{do}\,\{\, C_1\,\};C_2\, \{Q\}}$$

Handling Loops in Cyclic Proofs

- The standard Hoare rule for handling `while` loops:

$$\frac{\{t = z \wedge B \wedge P\}\, C_1\, \{t < z \wedge P\} \quad \{\neg B \wedge P\}\, C_2\, \{Q\}}{\{P\}\, \texttt{while}\,(B)\,\texttt{do}\,\{\, C_1 \,\};C_2\, \{Q\}}$$

  t is the loop variant

Handling Loops in Cyclic Proofs

- The standard Hoare rule for handling `while` loops:

$$\frac{\{t = z \wedge B \wedge P\}\, C_1\, \{t < z \wedge P\} \quad \{\neg B \wedge P\}\, C_2\, \{Q\}}{\{P\}\, \texttt{while}\,(B)\,\texttt{do}\,\{\, C_1 \,\};C_2\, \{Q\}}$$

  t is the loop variant

- With cyclic proof, it is enough just to unfold loops

$$\frac{\{B \wedge P\}\, C_1;\texttt{while}\,(B)\,\texttt{do}\,\{\, C_1 \,\};C_2\, \{Q\} \quad \{\neg B \wedge P\}\, C_2\, \{Q\}}{\{P\}\, \texttt{while}\,(B)\,\texttt{do}\,\{\, C_1 \,\};C_2\, \{Q\}}$$

Example: Deallocating the Linked List

```
while(x!=nil)do{y:=[x];free(x);x=y}
```

Example: Deallocating the Linked List

$\{\mathsf{ls}(\mathtt{x}, \mathtt{nil})\}$ `while(x!=nil)do{y:=[x];free(x);x=y}`

Example: Deallocating the Linked List

$$\{\mathsf{ls}(x, \texttt{nil})\} \quad \texttt{while}\,(x!=\texttt{nil})\,\texttt{do}\,\{\,y:=[x];\texttt{free}(x);x=y\,\} \quad \{\texttt{emp}\}$$

Example: Deallocating the Linked List

$$\{\mathsf{ls}(x, \mathtt{nil})\} \quad \mathtt{while}\,(x!=\mathtt{nil})\,\mathtt{do}\,\{\,y:=[x];\mathtt{free}(x);x=y\,\} \quad \{\mathtt{emp}\}$$

$$\dfrac{\left\{\begin{array}{c} x \neq \mathtt{nil} \\ \wedge\ \mathsf{ls}(x, \mathtt{nil}) \end{array}\right\}\ y:=[x];\mathtt{free}(x);x=y;\mathtt{while}\,(x!=\mathtt{nil})\,\mathtt{do}\,\{\,y:=[x];\mathtt{free}(x);x=y\,\}\ \{\mathtt{emp}\} \qquad \left\{\begin{array}{c} x = \mathtt{nil} \\ \wedge\ \mathsf{ls}(x, \mathtt{nil}) \end{array}\right\}\ \varepsilon\ \{\mathtt{emp}\}}{\{\mathsf{ls}(x, \mathtt{nil})\}\ \mathtt{while}\ \ldots\ \{\mathtt{emp}\}}\ (\mathtt{while})$$

Example: Deallocating the Linked List

$$\{\mathsf{ls}(\mathtt{x},\mathtt{nil})\}\quad \mathtt{while(x!=nil)do\{\,y:=[x];free(x);x=y\,\}}\quad \{\mathtt{emp}\}$$

$$\dfrac{\left\{\begin{array}{c}\mathtt{x}\neq\mathtt{nil}\\ \wedge\ \mathsf{ls}(\mathtt{x},\mathtt{nil})\end{array}\right\}\ \mathtt{y:=[x];}\ldots\ \{\mathtt{emp}\}\qquad \left\{\begin{array}{c}\mathtt{x}=\mathtt{nil}\\ \wedge\ \mathsf{ls}(\mathtt{x},\mathtt{nil})\end{array}\right\}\ \varepsilon\ \{\mathtt{emp}\}}{\{\mathsf{ls}(\mathtt{x},\mathtt{nil})\}\ \mathtt{while}\ \ldots\ \{\mathtt{emp}\}}\ (\text{while})$$

Example: Deallocating the Linked List

$$\{\mathsf{ls}(\mathtt{x},\mathtt{nil})\}\ \ \mathtt{while(x!=nil)do\{\,y:=[x];free(x);x=y\,\}}\ \ \{\mathtt{emp}\}$$

$$\dfrac{\left\{\begin{array}{c}\mathtt{x}\neq\mathtt{nil}\\ \wedge\,\mathsf{ls}(\mathtt{x},\mathtt{nil})\end{array}\right\}\ \mathtt{y:=[x];}\ldots\ \{\mathtt{emp}\}\qquad \dfrac{}{\left\{\begin{array}{c}\mathtt{x}=\mathtt{nil}\\ \wedge\,\mathsf{ls}(\mathtt{x},\mathtt{nil})\end{array}\right\}\ \varepsilon\ \{\mathtt{emp}\}}\,(\models)}{\{\mathsf{ls}(\mathtt{x},\mathtt{nil})\}\ \mathtt{while}\ \ldots\ \{\mathtt{emp}\}}\,(\text{while})$$

Example: Deallocating the Linked List

$$\{\mathsf{ls}(\mathtt{x},\mathtt{nil})\}\ \ \mathtt{while(x!=nil)do\{y:=[x];free(x);x=y\}}\ \ \{\mathtt{emp}\}$$

$$\dfrac{\dfrac{\left\{\begin{array}{c}\mathtt{x}\neq\mathtt{nil}\\ \wedge\,\mathtt{x}=\mathtt{nil}\\ \wedge\,\mathtt{emp}\end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\} \qquad \left\{\begin{array}{c}\mathtt{x}\mapsto v\\ *\,\mathsf{ls}(v,\mathtt{nil})\end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}{\left\{\mathtt{x}\neq\mathtt{nil}\ \wedge\ \mathsf{ls}(\mathtt{x},\mathtt{nil})\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}\ (\text{unfold ls}) \qquad \dfrac{}{\left\{\mathtt{x}=\mathtt{nil}\ \wedge\ \mathsf{ls}(\mathtt{x},\mathtt{nil})\right\}\ \varepsilon\ \{\mathtt{emp}\}}\ (\models)}{\{\mathsf{ls}(\mathtt{x},\mathtt{nil})\}\ \mathtt{while}\ \ldots\ \{\mathtt{emp}\}}\ (\text{while})$$

Example: Deallocating the Linked List

$$\{\mathsf{ls}(x, \mathtt{nil})\}\ \ \mathtt{while(x!=nil)do\{\,y:=[x];free(x);x=y\,\}}\ \ \{\mathtt{emp}\}$$

$$\dfrac{\dfrac{}{\left\{\begin{array}{c} x \neq \mathtt{nil} \\ \wedge\, x = \mathtt{nil} \\ \wedge\, \mathtt{emp} \end{array}\right\}\ \mathtt{y:=[x];\ \ldots}\ \{\mathtt{emp}\}}(\bot) \qquad \left\{\begin{array}{c} x \mapsto v \\ * \ \mathsf{ls}(v, \mathtt{nil}) \end{array}\right\}\ \mathtt{y:=[x];\ \ldots}\ \{\mathtt{emp}\}}{\left\{\begin{array}{c} x \neq \mathtt{nil} \\ \wedge\, \mathsf{ls}(x, \mathtt{nil}) \end{array}\right\}\ \mathtt{y:=[x];\ \ldots}\ \{\mathtt{emp}\}}(\text{unfold } \mathsf{ls}) \qquad \dfrac{}{\left\{\begin{array}{c} x = \mathtt{nil} \\ \wedge\, \mathsf{ls}(x, \mathtt{nil}) \end{array}\right\}\ \varepsilon\ \{\mathtt{emp}\}}(\models)$$

$$\vdots$$

$$\{\mathsf{ls}(x, \mathtt{nil})\}\ \mathtt{while\ \ldots}\ \{\mathtt{emp}\}\ (\text{while})$$

Example: Deallocating the Linked List

$$\{\mathsf{ls}(\mathtt{x},\mathtt{nil})\}\ \ \mathtt{while(x!=nil)do\{y:=[x];free(x);x=y\}}\ \ \{\mathtt{emp}\}$$

$$\dfrac{\dfrac{}{\left\{\begin{array}{c}\mathtt{x}\neq\mathtt{nil}\\ \wedge\,\mathtt{x}=\mathtt{nil}\\ \wedge\,\mathtt{emp}\end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\bot) \qquad \dfrac{\left\{\begin{array}{c}\mathtt{x}\mapsto y\\ *\,\mathsf{ls}(y,\mathtt{nil})\end{array}\right\}\ \mathtt{free(x);}\ \ldots\ \{\mathtt{emp}\}}{\left\{\begin{array}{c}\mathtt{x}\mapsto v\\ *\,\mathsf{ls}(v,\mathtt{nil})\end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\mathsf{load})}{\left\{\begin{array}{c}\mathtt{x}\neq\mathtt{nil}\\ \wedge\,\mathtt{x}=\mathtt{nil}\\ \wedge\,\mathtt{emp}\end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\mathsf{unfold}\ \mathsf{ls}) \qquad \dfrac{}{\left\{\begin{array}{c}\mathtt{x}=\mathtt{nil}\\ \wedge\,\mathsf{ls}(\mathtt{x},\mathtt{nil})\end{array}\right\}\ \varepsilon\ \{\mathtt{emp}\}}(\models)}{\left\{\begin{array}{c}\mathtt{x}\neq\mathtt{nil}\\ \wedge\,\mathsf{ls}(\mathtt{x},\mathtt{nil})\end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\mathsf{while})$$

$$\{\mathsf{ls}(\mathtt{x},\mathtt{nil})\}\ \mathtt{while}\ \ldots\ \{\mathtt{emp}\}$$

Example: Deallocating the Linked List

$$\{\mathsf{ls}(x,\mathtt{nil})\}\ \ \mathtt{while(x!=nil)do\{\,y:=[x];free(x);x=y\,\}}\ \ \{\mathtt{emp}\}$$

$$\dfrac{\dfrac{}{\left\{\begin{array}{c} x \neq \mathtt{nil} \\ \wedge\ x = \mathtt{nil} \\ \wedge\ \mathtt{emp} \end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\bot) \qquad \dfrac{\dfrac{\dfrac{\{\mathsf{ls}(y,\mathtt{nil})\}\ \mathtt{x=y;}\ \ldots\ \{\mathtt{emp}\}}{\left\{\begin{array}{c} x \mapsto y \\ * \mathsf{ls}(y,\mathtt{nil}) \end{array}\right\}\ \mathtt{free(x);}\ \ldots\ \{\mathtt{emp}\}}(\mathsf{free})}{\left\{\begin{array}{c} x \mapsto v \\ * \mathsf{ls}(v,\mathtt{nil}) \end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\mathsf{load})}{\left\{\begin{array}{c} x \neq \mathtt{nil} \\ \wedge\ \mathsf{ls}(x,\mathtt{nil}) \end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\mathsf{unfold}\ \mathsf{ls}) \qquad \dfrac{}{\left\{\begin{array}{c} x = \mathtt{nil} \\ \wedge\ \mathsf{ls}(x,\mathtt{nil}) \end{array}\right\}\ \varepsilon\ \{\mathtt{emp}\}}(\models)$$

$$\vdots$$

$$\{\mathsf{ls}(x,\mathtt{nil})\}\ \mathtt{while}\ \ldots\ \{\mathtt{emp}\}\ (\mathsf{while})$$

Example: Deallocating the Linked List

$$\{\mathsf{ls}(x,\mathtt{nil})\}\ \ \mathtt{while(x!=nil)do\{\,y:=[x];free(x);x=y\,\}}\ \ \{\mathtt{emp}\}$$

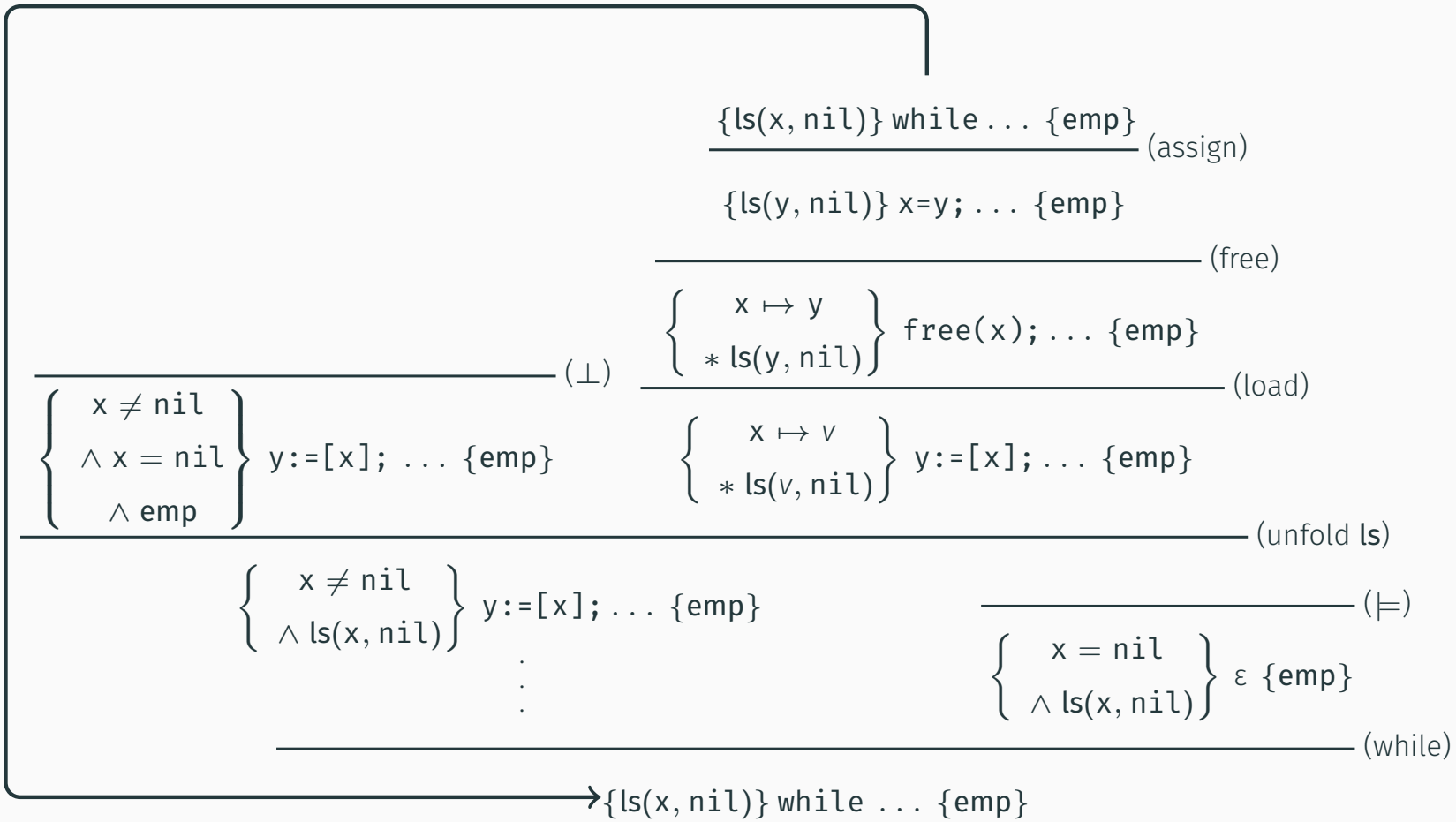
$$\dfrac{\dfrac{}{\left\{\begin{array}{c} x \neq \mathtt{nil} \\ \wedge\ x = \mathtt{nil} \\ \wedge\ \mathtt{emp} \end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\bot) \qquad \dfrac{\dfrac{\dfrac{\{\mathsf{ls}(x,\mathtt{nil})\}\ \mathtt{while}\ \ldots\ \{\mathtt{emp}\}}{\{\mathsf{ls}(y,\mathtt{nil})\}\ \mathtt{x=y;}\ \ldots\ \{\mathtt{emp}\}}(\text{assign})}{\left\{\begin{array}{c} x \mapsto y \\ * \mathsf{ls}(y,\mathtt{nil}) \end{array}\right\}\ \mathtt{free(x);}\ \ldots\ \{\mathtt{emp}\}}(\text{free})}{\left\{\begin{array}{c} x \mapsto v \\ * \mathsf{ls}(v,\mathtt{nil}) \end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\text{load})}{\left\{\begin{array}{c} x \neq \mathtt{nil} \\ \wedge\ \mathsf{ls}(x,\mathtt{nil}) \end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\text{unfold ls}) \qquad \dfrac{}{\left\{\begin{array}{c} x = \mathtt{nil} \\ \wedge\ \mathsf{ls}(x,\mathtt{nil}) \end{array}\right\}\ \varepsilon\ \{\mathtt{emp}\}}(\models)$$

$$\vdots$$

$$\dfrac{\left\{\begin{array}{c} x \neq \mathtt{nil} \\ \wedge\ \mathsf{ls}(x,\mathtt{nil}) \end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\} \qquad \left\{\begin{array}{c} x = \mathtt{nil} \\ \wedge\ \mathsf{ls}(x,\mathtt{nil}) \end{array}\right\}\ \varepsilon\ \{\mathtt{emp}\}}{\{\mathsf{ls}(x,\mathtt{nil})\}\ \mathtt{while}\ \ldots\ \{\mathtt{emp}\}}(\text{while})$$

Example: Deallocating the Linked List

$$\{\mathsf{ls}(x,\mathtt{nil})\}\ \ \mathtt{while(x!=nil)do\{\,y:=[x];free(x);x=y\,\}}\ \ \{\mathtt{emp}\}$$



Example: Deallocating the Linked List

$$\{\mathsf{ls}(x,\mathtt{nil})\}\ \ \mathtt{while(x!=nil)do\{\,y:=[x];free(x);x=y\,\}}\ \ \{\mathtt{emp}\}$$

$$\dfrac{\dfrac{}{\left\{\begin{array}{c} x \neq \mathtt{nil} \\ \wedge\ x = \mathtt{nil} \\ \wedge\ \mathtt{emp} \end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\bot) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\{\mathsf{ls}(x,\mathtt{nil})\}\ \mathtt{while}\ \ldots\ \{\mathtt{emp}\}}}{\{\mathsf{ls}(y,\mathtt{nil})\}\ \mathtt{x=y;}\ \ldots\ \{\mathtt{emp}\}}(\mathtt{assign})}{\left\{\begin{array}{c} x \mapsto y \\ *\ \mathsf{ls}(y,\mathtt{nil}) \end{array}\right\}\ \mathtt{free(x);}\ \ldots\ \{\mathtt{emp}\}}(\mathtt{free})}{\left\{\begin{array}{c} x \mapsto v \\ *\ \mathsf{ls}(v,\mathtt{nil}) \end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\mathtt{load})}{\left\{\begin{array}{c} x \neq \mathtt{nil} \\ \wedge\ \mathsf{ls}(x,\mathtt{nil}) \end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\}}(\mathtt{unfold\ ls}) \qquad \dfrac{}{\left\{\begin{array}{c} x = \mathtt{nil} \\ \wedge\ \mathsf{ls}(x,\mathtt{nil}) \end{array}\right\}\ \varepsilon\ \{\mathtt{emp}\}}(\models)$$

$$\vdots$$

$$\dfrac{\left\{\begin{array}{c} x \neq \mathtt{nil} \\ \wedge\ \mathsf{ls}(x,\mathtt{nil}) \end{array}\right\}\ \mathtt{y:=[x];}\ \ldots\ \{\mathtt{emp}\} \qquad \left\{\begin{array}{c} x = \mathtt{nil} \\ \wedge\ \mathsf{ls}(x,\mathtt{nil}) \end{array}\right\}\ \varepsilon\ \{\mathtt{emp}\}}{\{\mathsf{ls}(x,\mathtt{nil})\}\ \mathtt{while}\ \ldots\ \{\mathtt{emp}\}}(\mathtt{while})$$

Soundness of Cyclic Proof: Elements

- Fix a set of models to interpret proof system judgements

Soundness of Cyclic Proof: Elements

- Fix a set of models to interpret proof system judgements
- Fix some values that we can trace along paths in the proof
  - In our examples: inductive predicate instances

Soundness of Cyclic Proof: Elements

- Fix a set of models to interpret proof system judgements
- Fix some values that we can trace along paths in the proof
  - In our examples: inductive predicate instances
- Identify progression points of the proof system for Θ, e.g.

$$\frac{x = y \land \mathbf{emp} \vdash F \quad x \mapsto v * \mathsf{ls}(v, y) \vdash F}{\mathsf{ls}(x, y) \vdash F} \text{ (Case } \mathbf{ls}\text{)}$$

Soundness of Cyclic Proof: Elements

- Fix a set of models to interpret proof system judgements
- Fix some values that we can trace along paths in the proof
  - In our examples: inductive predicate instances
- Identify progression points of the proof system for Θ, e.g.

$$\frac{x = y \wedge \mathbf{emp} \vdash F \quad x \mapsto v * \mathsf{ls}(v, y) \vdash F}{\mathsf{ls}(x, y) \vdash F} \text{ (Case } \mathbf{ls}\text{)}$$

- Define a realization function Θ that maps pairs of models and trace values into a well-founded set

Soundness of Cyclic Proof: Descending Counter-models

The realization function must satisfy the following for each rule

$$\frac{\mathcal{J}_1[\tau_1] \quad \dots \quad \mathcal{J}_n[\tau_n]}{\mathcal{J}[\tau]}$$

if model $m \not\models \mathcal{J}$ then there is a model $m' \not\models \mathcal{J}_i$ for some i, and:

- $\Theta(m', \tau_i) \leq \Theta(m, \tau)$
- $\Theta(m', \tau_i) < \Theta(m, \tau)$ if there is a progression from τ to τ_i

Soundness of Cyclic Proof: General Principle

- Impose global trace condition on proof graphs:
  - Every infinite path must have an infinitely progressing trace
  - This condition is decidable using Büchi automata

Soundness of Cyclic Proof: General Principle

- Impose global trace condition on proof graphs:
  - Every infinite path must have an infinitely progressing trace
  - This condition is decidable using Büchi automata
- We obtain an infinite descent proof-by-contradiction:

Soundness of Cyclic Proof: General Principle

- Impose global trace condition on proof graphs:
  - Every infinite path must have an infinitely progressing trace
  - This condition is decidable using Büchi automata
- We obtain an infinite descent proof-by-contradiction:
  - Assume the conclusion of the proof is invalid

Soundness of Cyclic Proof: General Principle

- Impose global trace condition on proof graphs:
  - Every infinite path must have an infinitely progressing trace
  - This condition is decidable using Büchi automata
- We obtain an infinite descent proof-by-contradiction:
  - Assume the conclusion of the proof is invalid
  - Local soundness $\Rightarrow$ infinite sequence of (counter) models

Soundness of Cyclic Proof: General Principle

- Impose global trace condition on proof graphs:
  - Every infinite path must have an infinitely progressing trace
  - This condition is decidable using Büchi automata
- We obtain an infinite descent proof-by-contradiction:
  - Assume the conclusion of the proof is invalid
  - Local soundness $\Rightarrow$ infinite sequence of (counter) models
  - Global trace condition $\Rightarrow$ corresponding sequence of trace values

Soundness of Cyclic Proof: General Principle

- Impose global trace condition on proof graphs:
  - Every infinite path must have an infinitely progressing trace
  - This condition is decidable using Büchi automata
- We obtain an infinite descent proof-by-contradiction:
  - Assume the conclusion of the proof is invalid
  - Local soundness $\Rightarrow$ infinite sequence of (counter) models
  - Global trace condition $\Rightarrow$ corresponding sequence of trace values
    - Θ maps these to a (non-increasing) chain of ordinals

Soundness of Cyclic Proof: General Principle

- Impose global trace condition on proof graphs:
  - Every infinite path must have an infinitely progressing trace
  - This condition is decidable using Büchi automata
- We obtain an infinite descent proof-by-contradiction:
  - Assume the conclusion of the proof is invalid
  - Local soundness $\Rightarrow$ infinite sequence of (counter) models
  - Global trace condition $\Rightarrow$ corresponding sequence of trace values
    - Θ maps these to a (non-increasing) chain of ordinals
    - The trace is infinitely progressing $\Rightarrow$ chain is infinitely descending

Soundness of Cyclic Proof: General Principle

- Impose global trace condition on proof graphs:
  - Every infinite path must have an infinitely progressing trace
  - This condition is decidable using Büchi automata
- We obtain an infinite descent proof-by-contradiction:
  - Assume the conclusion of the proof is invalid
  - Local soundness $\Rightarrow$ infinite sequence of (counter) models
  - Global trace condition $\Rightarrow$ corresponding sequence of trace values
    - Θ maps these to a (non-increasing) chain of ordinals
    - The trace is infinitely progressing $\Rightarrow$ chain is infinitely descending
    - But the ordinals are well-founded . . . contradiction

$\mathrm{FOL}_{\mathrm{ID}}$: Semantics of Inductive Definitions

Definition (Inductive Definition Set)

An *inductive definition set* contains productions $\mathrm{P}_1\,\vec{t_1}, \ldots, \mathrm{P}_j\,\vec{t_j} \Rightarrow \mathrm{P}_0\,\vec{t_0}$

Definition (Characteristic Operators)

Inductive definition sets Φ induce *characteristic operators* φ_Φ on predicate interpretations X (functions from predicate formulas to sets of models):

$$\varphi_\Phi(X)(\mathrm{P}\,\vec{u}) = \{m \mid \mathrm{P}_1\,\vec{t_1}, \ldots, \mathrm{P}_j\,\vec{t_j} \Rightarrow \mathrm{P}\,\vec{t} \in \Phi \wedge m(\vec{u}) = m(\vec{t}\theta) \wedge \forall 1 \leq i \leq j : m \in X(\mathrm{P}_i\,\vec{t_i}\theta)\}$$

$\mathrm{FOL}_{\mathrm{ID}}$: Semantics of Inductive Definitions

Definition (Inductive Definition Set)

An *inductive definition set* contains productions $\mathrm{P}_1\,\vec{t_1}, \ldots, \mathrm{P}_j\,\vec{t_j} \Rightarrow \mathrm{P}_0\,\vec{t_0}$

Definition (Characteristic Operators)

Inductive definition sets Φ induce *characteristic operators* φ_Φ on predicate interpretations X (functions from predicate formulas to sets of models):

$$\varphi_\Phi(X)(\mathrm{P}\,\vec{u}) = \{m \mid \mathrm{P}_1\,\vec{t_1}, \ldots, \mathrm{P}_j\,\vec{t_j} \Rightarrow \mathrm{P}\,\vec{t} \in \Phi \wedge m(\vec{u}) = m(\vec{t}\theta) \wedge \forall 1 \leq i \leq j : m \in X(\mathrm{P}_i\,\vec{t_i}\theta)\}$$

$$\Phi = \left\{ \frac{}{\mathrm{N}\,0} \quad \frac{\mathrm{N}\,x}{\mathrm{N}\,\mathrm{s}x} \qquad \frac{}{\mathrm{E}\,0} \quad \frac{\mathrm{E}\,x}{\mathrm{O}\,\mathrm{s}x} \quad \frac{\mathrm{O}\,x}{\mathrm{E}\,\mathrm{s}x} \right\} \qquad X_\bot(\mathrm{P}\,\vec{t}) = \emptyset \quad \text{for all } \mathrm{P}\,\vec{t}$$

$$\varphi_\Phi(X_\bot)(\mathrm{N}\,x) = \{[x \mapsto 0]\}$$
$$\varphi_\Phi(X_\bot)(\mathrm{E}\,x) = \{[x \mapsto 0]\}$$
$$\varphi_\Phi(X_\bot)(\mathrm{O}\,x) = \{\}$$

$\mathrm{FOL}_{\mathrm{ID}}$: Semantics of Inductive Definitions

Definition (Inductive Definition Set)

An *inductive definition set* contains productions $\mathsf{P}_1\,\vec{t_1}, \ldots, \mathsf{P}_j\,\vec{t_j} \Rightarrow \mathsf{P}_0\,\vec{t_0}$

Definition (Characteristic Operators)

Inductive definition sets Φ induce *characteristic operators* φ_Φ on predicate interpretations X (functions from predicate formulas to sets of models):

$$\varphi_\Phi(X)(\mathsf{P}\,\vec{u}) = \{m \mid \mathsf{P}_1\,\vec{t_1}, \ldots, \mathsf{P}_j\,\vec{t_j} \Rightarrow \mathsf{P}\,\vec{t} \in \Phi \wedge m(\vec{u}) = m(\vec{t}\theta) \wedge \forall 1 \leq i \leq j : m \in X(\mathsf{P}_i\,\vec{t_i}\theta)\}$$

$$\Phi = \left\{ \frac{}{\mathsf{N}\,0} \quad \frac{\mathsf{N}\,x}{\mathsf{N}\,\mathsf{s}x} \qquad \frac{}{\mathsf{E}\,0} \quad \frac{\mathsf{E}\,x}{\mathsf{O}\,\mathsf{s}x} \quad \frac{\mathsf{O}\,x}{\mathsf{E}\,\mathsf{s}x} \right\} \qquad X_\bot(\mathsf{P}\,\vec{t}) = \emptyset \quad \text{for all } \mathsf{P}\,\vec{t}$$

$$\varphi_\Phi(\varphi_\Phi(X_\bot))(\mathsf{N}\,x) = \{[x \mapsto 0], [x \mapsto \mathsf{s}0]\}$$
$$\varphi_\Phi(\varphi_\Phi(X_\bot))(\mathsf{E}\,x) = \{[x \mapsto 0]\}$$
$$\varphi_\Phi(\varphi_\Phi(X_\bot))(\mathsf{O}\,x) = \{[x \mapsto \mathsf{s}0]\}$$

$\mathrm{FOL}_{\mathrm{ID}}$: Semantics of Inductive Definitions

Definition (Inductive Definition Set)

An *inductive definition set* contains productions $\mathsf{P}_1\,\vec{t_1}, \ldots, \mathsf{P}_j\,\vec{t_j} \Rightarrow \mathsf{P}_0\,\vec{t_0}$

Definition (Characteristic Operators)

Inductive definition sets Φ induce *characteristic operators* φ_Φ on predicate interpretations X (functions from predicate formulas to sets of models):

$$\varphi_\Phi(X)(\mathsf{P}\,\vec{u}) = \{m \mid \mathsf{P}_1\,\vec{t_1}, \ldots, \mathsf{P}_j\,\vec{t_j} \Rightarrow \mathsf{P}\,\vec{t} \in \Phi \wedge m(\vec{u}) = m(\vec{t}\theta) \wedge \forall 1 \leq i \leq j : m \in X(\mathsf{P}_i\,\vec{t_i}\theta)\}$$

$$\Phi = \left\{ \frac{}{\mathsf{N}\,0} \quad \frac{\mathsf{N}\,x}{\mathsf{N}\,\mathsf{s}x} \qquad \frac{}{\mathsf{E}\,0} \quad \frac{\mathsf{E}\,x}{\mathsf{O}\,\mathsf{s}x} \quad \frac{\mathsf{O}\,x}{\mathsf{E}\,\mathsf{s}x} \right\} \qquad X_\bot(\mathsf{P}\,\vec{t}) = \emptyset \quad \text{for all } \mathsf{P}\,\vec{t}$$

$$\varphi_\Phi(\varphi_\Phi(\varphi_\Phi(X_\bot)))(\mathsf{N}\,x) = \{[x \mapsto 0], [x \mapsto \mathsf{s}0], [x \mapsto \mathsf{ss}0]\}$$
$$\varphi_\Phi(\varphi_\Phi(\varphi_\Phi(X_\bot)))(\mathsf{E}\,x) = \{[x \mapsto 0], [x \mapsto \mathsf{ss}0]\}$$
$$\varphi_\Phi(\varphi_\Phi(\varphi_\Phi(X_\bot)))(\mathsf{O}\,x) = \{[x \mapsto \mathsf{s}0]\}$$

$\mathrm{FOL}_{\mathrm{ID}}$: Semantics of Inductive Definitions

Definition (Inductive Definition Set)

An *inductive definition set* contains productions $\mathrm{P}_1\,\vec{t_1}, \ldots, \mathrm{P}_j\,\vec{t_j} \Rightarrow \mathrm{P}_0\,\vec{t_0}$

Definition (Characteristic Operators)

Inductive definition sets Φ induce *characteristic operators* φ_Φ on predicate interpretations X (functions from predicate formulas to sets of models):

$$\varphi_\Phi(X)(\mathrm{P}\,\vec{u}) = \{m \mid \mathrm{P}_1\,\vec{t_1}, \ldots, \mathrm{P}_j\,\vec{t_j} \Rightarrow \mathrm{P}\,\vec{t} \in \Phi \wedge m(\vec{u}) = m(\vec{t}\theta) \wedge \forall 1 \leq i \leq j : m \in X(\mathrm{P}_i\,\vec{t_i}\theta)\}$$

$$\Phi = \left\{ \frac{}{\mathrm{N}\,0} \quad \frac{\mathrm{N}\,x}{\mathrm{N}\,\mathrm{s}x} \qquad \frac{}{\mathrm{E}\,0} \quad \frac{\mathrm{E}\,x}{\mathrm{O}\,\mathrm{s}x} \quad \frac{\mathrm{O}\,x}{\mathrm{E}\,\mathrm{s}x} \right\} \qquad X_\bot(\mathrm{P}\,\vec{t}) = \emptyset \quad \text{for all } \mathrm{P}\,\vec{t}$$

$$X_\bot \sqsubseteq \varphi_\Phi(X_\bot) \sqsubseteq \varphi_\Phi(\varphi_\Phi(X_\bot)) \sqsubseteq \ldots \sqsubseteq \varphi_\Phi^\alpha(X_\bot) \sqsubseteq \ldots \sqsubseteq \mu X.\varphi_\Phi(X)$$

$\mathrm{FOL}_{\mathrm{ID}}$: Semantics of Inductive Definitions

Definition (Inductive Definition Set)

An *inductive definition set* contains productions $\mathsf{P}_1\,\vec{t_1}, \ldots, \mathsf{P}_j\,\vec{t_j} \Rightarrow \mathsf{P}_0\,\vec{t_0}$

Definition (Characteristic Operators)

Inductive definition sets Φ induce *characteristic operators* φ_Φ on predicate interpretations X (functions from predicate formulas to sets of models):

$$\varphi_\Phi(X)(\mathsf{P}\,\vec{u}) = \{m \mid \mathsf{P}_1\,\vec{t_1}, \ldots, \mathsf{P}_j\,\vec{t_j} \Rightarrow \mathsf{P}\,\vec{t} \in \Phi \wedge m(\vec{u}) = m(\vec{t}\theta) \wedge \forall 1 \leq i \leq j : m \in X(\mathsf{P}_i\,\vec{t_i}\theta)\}$$

$$\Phi = \left\{ \frac{}{\mathsf{N}\,0} \quad \frac{\mathsf{N}\,x}{\mathsf{N}\,\mathsf{s}x} \qquad \frac{}{\mathsf{E}\,0} \quad \frac{\mathsf{E}\,x}{\mathsf{O}\,\mathsf{s}x} \quad \frac{\mathsf{O}\,x}{\mathsf{E}\,\mathsf{s}x} \right\} \qquad X_\perp(\mathsf{P}\,\vec{t}) = \emptyset \quad \text{for all } \mathsf{P}\,\vec{t}$$

$$[\![\cdot]\!]^\Phi_0 \sqsubseteq [\![\cdot]\!]^\Phi_1 \sqsubseteq [\![\cdot]\!]^\Phi_2 \sqsubseteq \ldots \sqsubseteq [\![\cdot]\!]^\Phi_\alpha \sqsubseteq \ldots [\![\cdot]\!]^\Phi$$

$\mathsf{FOL}_{\mathsf{ID}}$: Models as Realizers

- We define the realization function Θ by:

$$\Theta(\mathsf{P}\,\vec{t}, m) \stackrel{\text{def}}{=} \min(\{\alpha \mid m \in [\![\mathsf{P}\,\vec{t}]\!]^{\Phi}_{\alpha}\})$$

$\mathrm{FOL}_{\mathrm{ID}}$: Models as Realizers

- We define the realization function Θ by:

$$\Theta(\mathrm{P}\,\vec{t}, m) \stackrel{\mathrm{def}}{=} \min(\{\alpha \mid m \in [\![\mathrm{P}\,\vec{t}]\!]^{\Phi}_{\alpha}\})$$

- Then the logical inference rules have the property that

$$\frac{\Sigma_1 \vdash \Pi_1 \quad \ldots \quad \Sigma_n \vdash \Pi_n}{\Gamma, \mathrm{P}\,\vec{t} \vdash \Delta}$$

  for model $m \not\models \langle \Gamma, \mathrm{P}\,\vec{t} \vdash \Delta \rangle$, there is model $m' \not\models \langle \Sigma_i \vdash \Pi_i \rangle$ (for some i) and if $\mathrm{P}\,\vec{t} \in \Sigma_i$ then $\Theta(\mathrm{P}\,\vec{t}, m') \leq \Theta(\mathrm{P}\,\vec{t}, m)$

$\mathrm{FOL}_{\mathrm{ID}}$: Models as Realizers

- We define the realization function Θ by:

$$\Theta(\mathrm{P}\,\vec{t}, m) \stackrel{\mathrm{def}}{=} \min(\{\alpha \mid m \in [\![\mathrm{P}\,\vec{t}]\!]^{\Phi}_{\alpha}\})$$

- Then the logical inference rules have the property that

$$\frac{\Gamma, t = 0 \vdash \Delta \quad \Gamma, t = \mathrm{s}x, \mathrm{N}\,x \vdash \Delta}{\Gamma, \mathrm{N}\,t \vdash \Delta}\ (\text{Case } N)$$

  for model $m \not\models \langle \Gamma, \mathrm{N}\,\vec{t} \vdash \Delta \rangle$, there is model m' with either $m' \not\models \langle \Gamma, t = 0 \vdash \Delta$ or $m' \not\models \langle \Gamma, t = \mathrm{s}x, \mathrm{N}\,x \vdash \Delta \rangle$, and if the latter then $\Theta(\mathrm{N}\,x, m') < \Theta(\mathrm{N}\,t, m)$

Cyclic Proofs vs Infinite Proofs

- Cyclic proofs are the (strict) regular subset of the set of non-well-founded proof trees

Cyclic Proofs vs Infinite Proofs

- Cyclic proofs are the (strict) regular subset of the set of non-well-founded proof trees

- For $\mathrm{FOL}_{\mathrm{ID}}$, the full infinite system is (cut-free) complete with respect to standard models (Brotherston & Simpson)

Cyclic Proofs vs Infinite Proofs

- Cyclic proofs are the (strict) regular subset of the set of non-well-founded proof trees
- For $\mathrm{FOL}_{\mathrm{ID}}$, the full infinite system is (cut-free) complete with respect to standard models (Brotherston & Simpson)
- Cut is likely not eliminable in the cyclic sub-system

Cyclic Proof vs Explicit Induction

- Explicit induction requires induction hypothesis F up-front

$$\frac{}{N\ 0} \qquad \frac{N\ x}{N\ sx} \qquad \frac{\Gamma \vdash F[0] \quad \Gamma, F[x] \vdash F[sx], \Delta \quad \Gamma, F[t] \vdash \Delta}{\Gamma, N\ t \vdash \Delta}\ (\text{Ind}\ N)$$

Cyclic Proof vs Explicit Induction

- Explicit induction requires induction hypothesis F up-front

$$\frac{}{N\ 0} \qquad \frac{N\ x}{N\ sx} \qquad \frac{\Gamma \vdash F[0] \quad \Gamma, F[x] \vdash F[sx], \Delta \quad \Gamma, F[t] \vdash \Delta}{\Gamma, N\ t \vdash \Delta}\ (\text{Ind } N)$$

- Cyclic proof enables '*discovery*' of induction hypotheses

Cyclic Proof vs Explicit Induction

- Explicit induction requires induction hypothesis F up-front

$$\frac{}{N\ 0} \qquad \frac{N\ x}{N\ sx} \qquad \frac{\Gamma \vdash F[0] \quad \Gamma, F[x] \vdash F[sx], \Delta \quad \Gamma, F[t] \vdash \Delta}{\Gamma, N\ t \vdash \Delta}\ (\text{Ind}\ N)$$

- Cyclic proof enables '*discovery*' of induction hypotheses

- Complex induction schemes naturally represented by nested and overlapping cycles

Cyclic Proof vs Explicit Induction: Results for $\mathrm{FOL}_{\mathrm{ID}}$

- Every sequent provable using the explict induction rule is also derivable using cyclic proof (Brotherston & Simpson)

Cyclic Proof vs Explicit Induction: Results for $\mathrm{FOL}_{\mathrm{ID}}$

- Every sequent provable using the explict induction rule is also derivable using cyclic proof (Brotherston & Simpson)

- For some sets of inductive definitions, cyclic proof is strictly more powerful (Berardi & Tatsuta, FoSSaCS'17)

Cyclic Proof vs Explicit Induction: Results for $\mathrm{FOL}_{\mathrm{ID}}$

- Every sequent provable using the explict induction rule is also derivable using cyclic proof (Brotherston & Simpson)
- For some sets of inductive definitions, cyclic proof is strictly more powerful (Berardi & Tatsuta, FoSSaCS'17)
- When arithmetic is included, cyclic proof and explicit induction are equivalent (Berardi & Tatsuta, LICS'17)

Other Cyclic Proof Systems

- μ-calculus (Sprenger and Dam, FoSSaCS'03)
- Temporal properties of heap-manipulating code
  (Tellez Espinosa and Brotherston, CADE'17)
- Kleene algebra (Das, TABLEAUX'17)

Part II

Realizability Results

Using Cyclic Proofs to Compute Semantic Information

Motivation: Cyclic Termination Proofs of Procedural Programs

```
struct ll { int data; ll *next; }

void rev(ll *x)  { /* reverses list */ }
void shuffle(ll *x)          {
   if ( x != NULL ) {

     ll *y = x->next;

     rev(y);

     shuffle(y);

   }
}
```

Motivation: Cyclic Termination Proofs of Procedural Programs

```
struct ll { int data; ll *next; }
```

$\mathsf{list}(x) \Leftrightarrow (x = \mathtt{NULL} \wedge \mathsf{emp}) \vee (x \mapsto (d, l) * \mathsf{list}(l))$

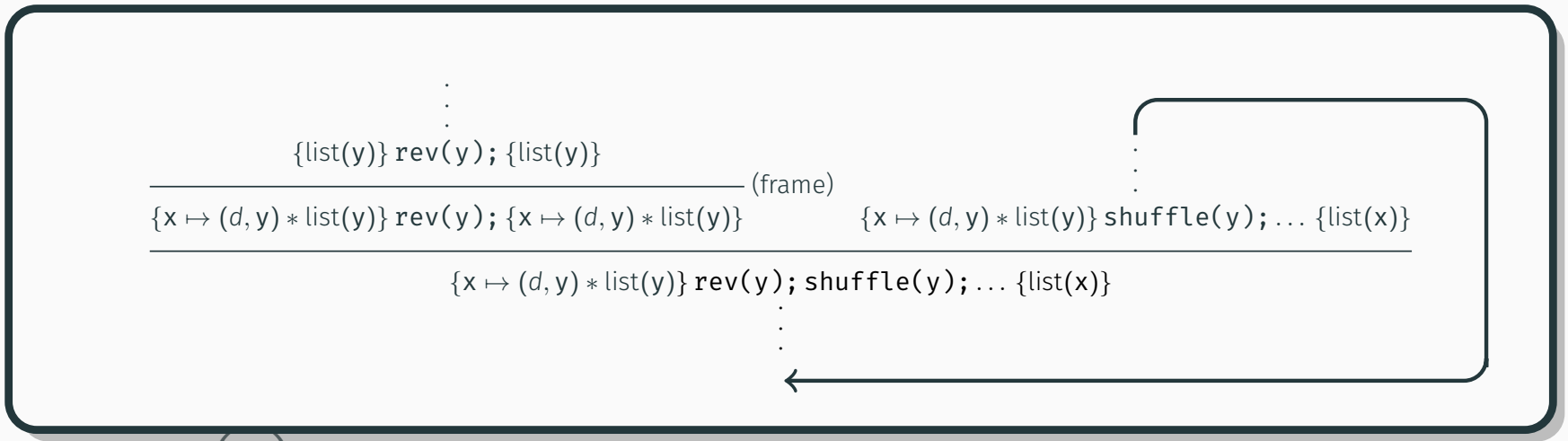
```
void rev(ll *x) { list(x) } { /* reverses list */ } { list(x) }

void shuffle(ll *x) { list(x) } {
   if ( x != NULL ) {
     { x ↦ (d, l) * list(l) }
     ll *y = x->next;
     { x ↦ (d, y) * list(y) }
     rev(y);
     { x ↦ (d, y) * list(y) }
     shuffle(y);
     { x ↦ (d, y) * list(y) }
   }
} { list(x) }
```

Motivation: Cyclic Termination Proofs of Procedural Programs

```
struct ll { int data; ll *next; }
```

$\mathsf{list}(x) \Leftrightarrow (x = \mathtt{NULL} \wedge \mathsf{emp}) \vee (x \mapsto (d, l) * \mathsf{list}(l))$

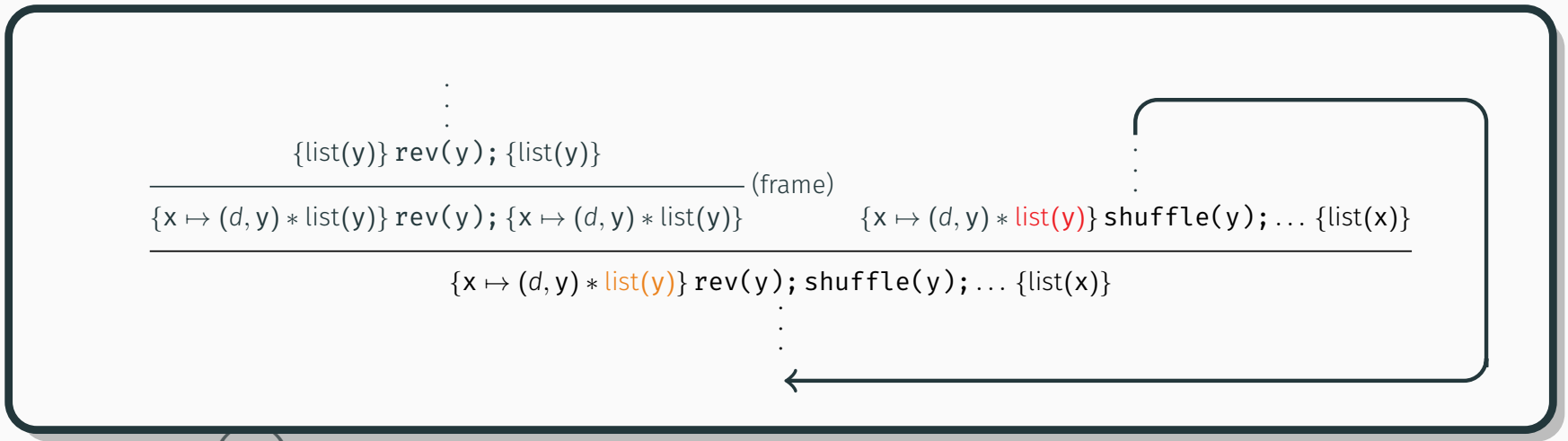



```
    rev(y);
    { x ↦ (d, y) * list(y) }
    shuffle(y);
    { x ↦ (d, y) * list(y) }
  }
} { list(x) }
```

Motivation: Cyclic Termination Proofs of Procedural Programs

```
struct ll { int data; ll *next; }
```

$\mathsf{list}(x) \Leftrightarrow (x = \mathtt{NULL} \wedge \mathsf{emp}) \vee (x \mapsto (d, l) * \mathsf{list}(l))$

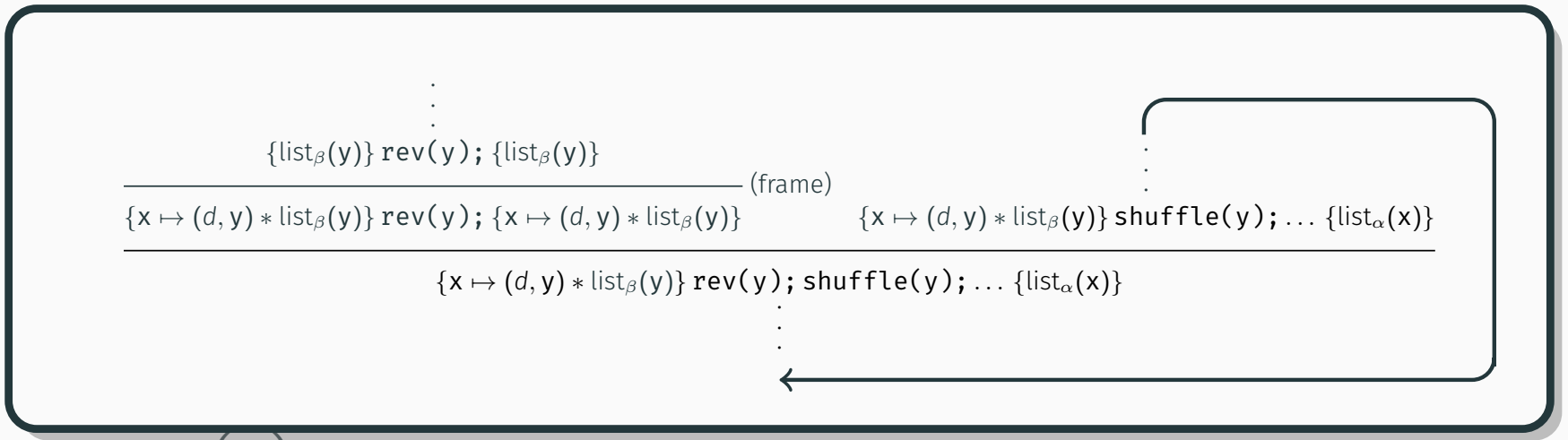



```
    rev(y);
    { x ↦ (d, y) * list(y) }
    shuffle(y);
    { x ↦ (d, y) * list(y) }
  }
} { list(x) }
```

Motivation: Cyclic Termination Proofs of Procedural Programs

```
struct ll { int data; ll *next; }
```

$\mathsf{list}(x) \Leftrightarrow (x = \mathtt{NULL} \wedge \mathsf{emp}) \vee (x \mapsto (d, l) * \mathsf{list}(l))$

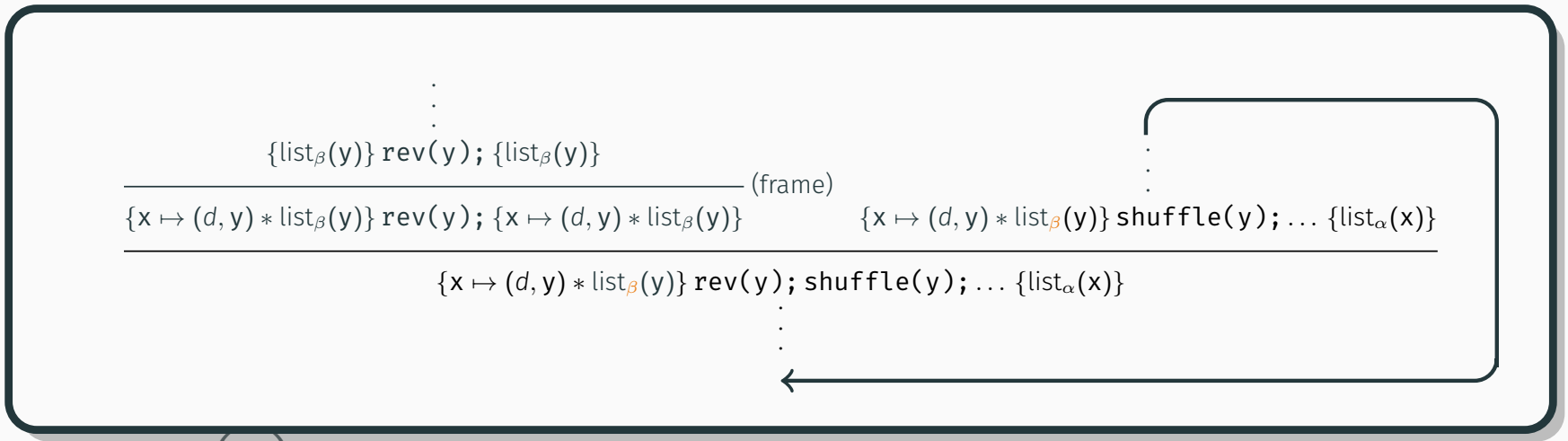



```
    rev(y);
    { x ↦ (d, y) * list(y) }
    shuffle(y);
    { x ↦ (d, y) * list(y) }
  }
} { list(x) }
```

Motivation: Cyclic Termination Proofs of Procedural Programs

```
struct ll { int data; ll *next; }
```

$\text{list}(x) \Leftrightarrow (x = \texttt{NULL} \wedge \text{emp}) \vee (x \mapsto (d, l) * \text{list}(l))$

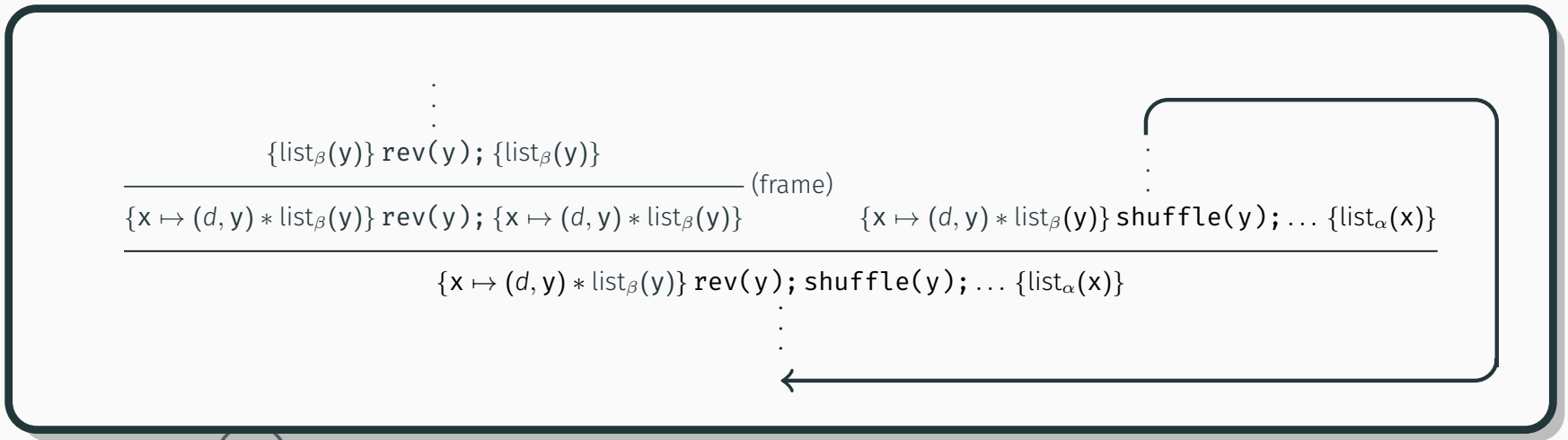



```
    rev(y);
    { x ↦ (d, y) * list(y) }
    shuffle(y);
    { x ↦ (d, y) * list(y) }
  }
} { list(x) }
```

Motivation: Cyclic Termination Proofs of Procedural Programs

```
struct ll { int data; ll *next; }
```

$\mathsf{list}(x) \Leftrightarrow (x = \mathtt{NULL} \wedge \mathsf{emp}) \vee (x \mapsto (d, l) * \mathsf{list}(l))$




```
    rev(y);
    { x ↦ (d, y) * list(y) }
    shuffle(
    { x ↦ (d, y
  }
} { list(x) }
```

$$\frac{\{x = \mathtt{NULL} \wedge \mathsf{emp}\}\ C\ \{\psi\} \quad \{x \mapsto (d, l) * \mathsf{list}_\beta(l) \wedge \beta < \alpha\}\ C\ \{\psi\}}{\{\mathsf{list}_\alpha(x)\}\ C\ \{\psi\}}\ \text{(Case list)}$$

Motivation: Cyclic Termination Proofs of Procedural Programs

```
struct ll { int data; ll *next; }
```

$\text{list}(x) \Leftrightarrow (x = \texttt{NULL} \wedge \text{emp}) \vee (x \mapsto (d, l) * \text{list}(l))$

```
void rev(ll *x) { list_α(x) } { /* reverses list */ } { list_α(x) }

void shuffle(ll *x) { list_α(x) } {
   if ( x != NULL ) {
     { x ↦ (d, l) * list_β(l) ∧ β < α }
     ll *y = x->next;
     { x ↦ (d, y) * list_β(y) ∧ β < α }
     rev(y);
     { x ↦ (d, y) * list_β(y) ∧ β < α }
     shuffle(y);
     { x ↦ (d, y) * list_β(y) ∧ β < α }
   }
} { list_α(x) }
```

Motivation: Cyclic Termination Proofs of Procedural Programs

```
struct ll { int data; ll *next; }
```

$\text{list}(x) \Leftrightarrow (x = \texttt{NULL} \wedge \text{emp}) \vee (x \mapsto (d, l) * \text{list}(l))$

```
void rev(ll *x) { list_α(x) } { ... } { list_α(x) }
void shuffle(ll *x) { list_α(x) } {
```

Intra-procedural analysis produces verification conditions, in the form of *entailments*, e.g.

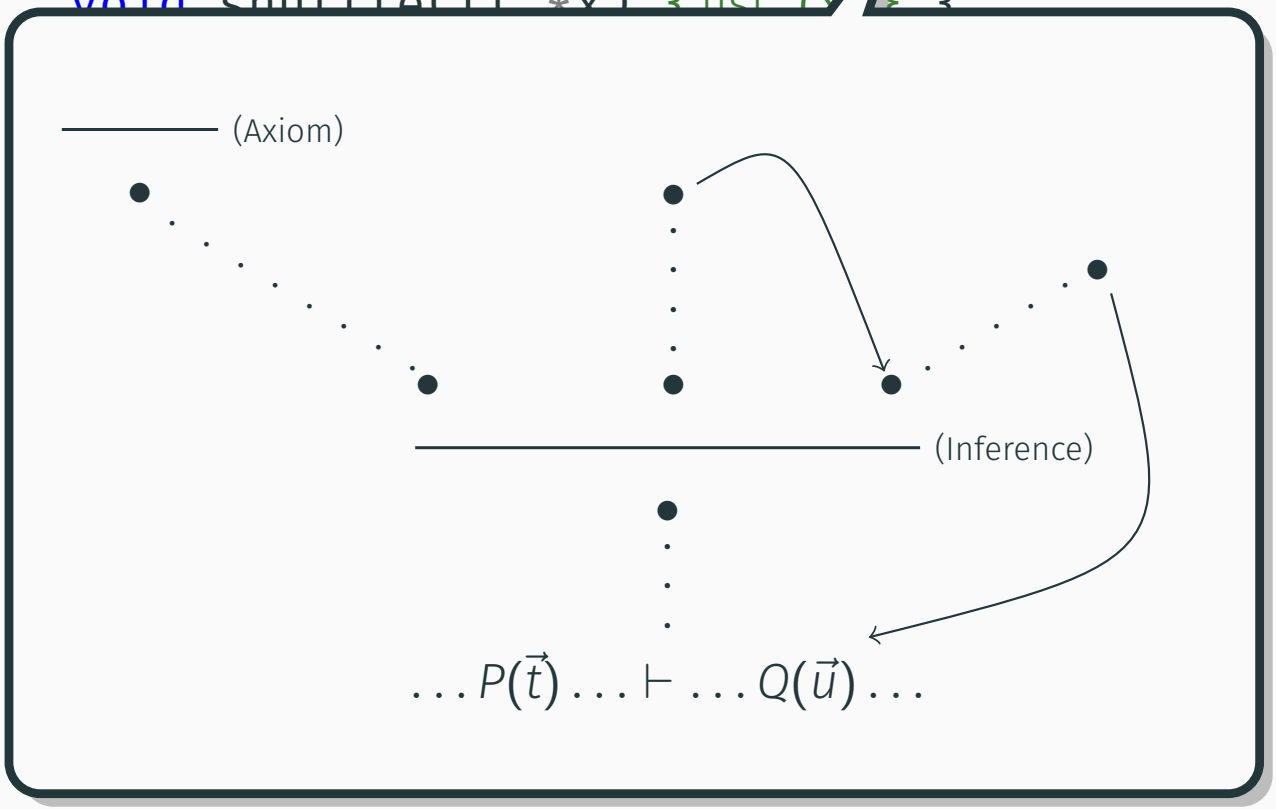
$$x \neq \texttt{NULL} \wedge x \mapsto (d, y) * \text{list}(y) \vdash \text{list}(x)$$

Motivation: Cyclic Termination Proofs of Procedural Programs

```
struct ll { int data; ll *next; }
```

$\text{list}(x) \Leftrightarrow (x = \texttt{NULL} \wedge \text{emp}) \vee (x \mapsto (d, l) * \text{list}(l))$

```
void rev(ll *x) { list_α(x) } { ... } { list_α(x) }
void shuffle(ll *x) { list_α(x) } {
```

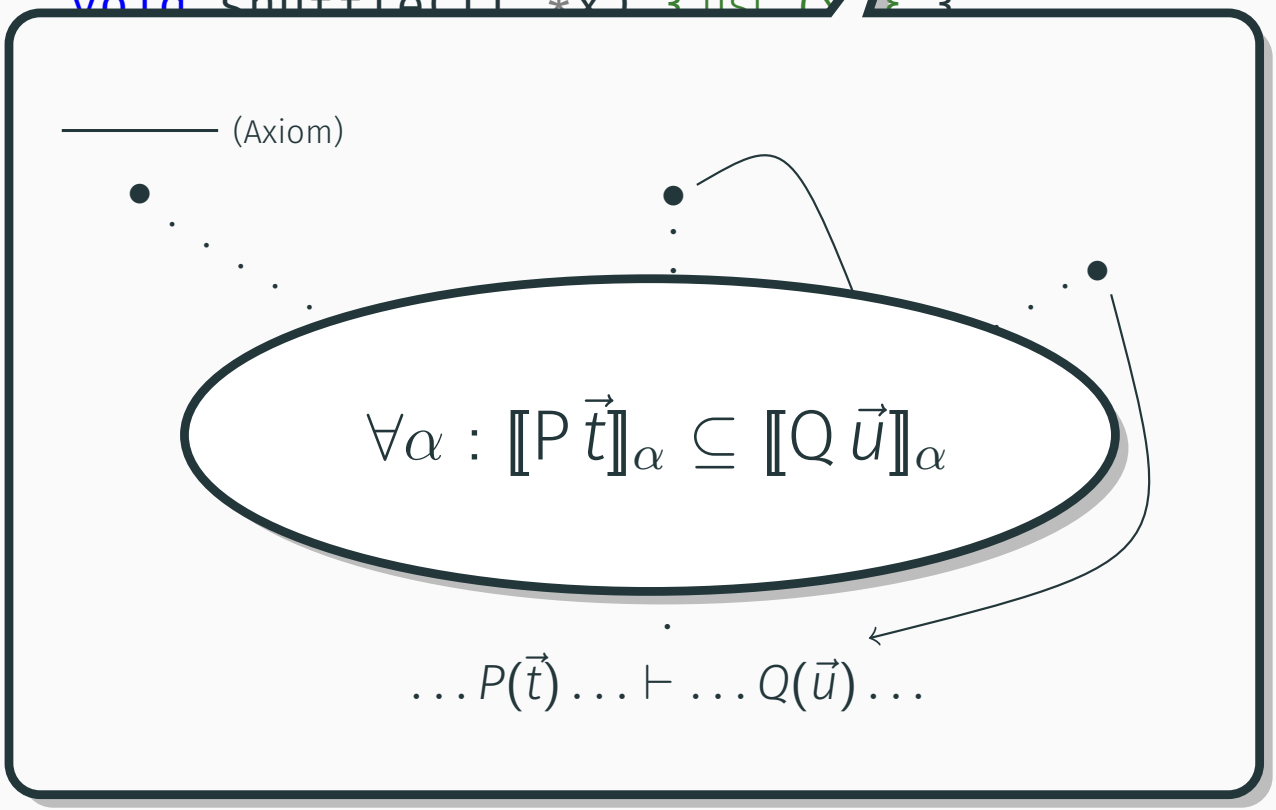


Motivation: Cyclic Termination Proofs of Procedural Programs

```
struct ll { int data; ll *next; }
```

$\text{list}(x) \Leftrightarrow (x = \texttt{NULL} \wedge \text{emp}) \vee (x \mapsto (d, l) * \text{list}(l))$

```
void rev(ll *x) { list_α(x) } { ... } { list_α(x) }
void shuffle(ll *x) { list_α(x) } {
```



Overview of Results

- Information about semantic inclusions between inductive predicates can be extracted from cyclic proofs of entailments

Overview of Results

- Information about semantic inclusions between inductive predicates can be extracted from cyclic proofs of entailments
    - These inclusions hold when the proof graph satisfies a structural (realizability) condition that we define

Overview of Results

- Information about semantic inclusions between inductive predicates can be extracted from cyclic proofs of entailments
    - These inclusions hold when the proof graph satisfies a structural (realizability) condition that we define
- The realizability condition is equivalent to a containment between two weighted automata that can be constructed from the proof graph

Overview of Results

- Information about semantic inclusions between inductive predicates can be extracted from cyclic proofs of entailments
    - These inclusions hold when the proof graph satisfies a structural (realizability) condition that we define
- The realizability condition is equivalent to a containment between two weighted automata that can be constructed from the proof graph
    - Under certain extra structural conditions, this containment falls within existing decidability results

Extracting Semantic Orderings: Basic Ideas

To extract these semantic relationships from cyclic proofs:

- We have to consider traces along the right-hand side of sequents, which are
  - maximally finite
  - matched by some left-hand trace along the same path
- We then count the number of times each trace progresses
  - the left-hand one must progress at least as often as the right-hand one

Extracting Semantic Orderings: Example (1)

$$\Rightarrow \mathsf{N}\,0$$
$$\mathsf{N}\,x \Rightarrow \mathsf{N}\,\mathsf{s}x$$
$$\Rightarrow \mathsf{E}\,0$$
$$\mathsf{O}\,x \Rightarrow \mathsf{E}\,\mathsf{s}x$$
$$\mathsf{E}\,x \Rightarrow \mathsf{O}\,\mathsf{s}x$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \mathsf{E}\,0,\ \mathsf{O}\,0}(\mathsf{E}\,\mathsf{R}_1)}{x = 0 \vdash \mathsf{E}\,x,\ \mathsf{O}\,x}(=\mathsf{L}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\mathsf{N}\,x \vdash \mathsf{E}\,x,\ \mathsf{O}\,x}{\mathsf{N}\,y \vdash \mathsf{E}\,y,\ \mathsf{O}\,y}(\mathsf{Subst})}{\mathsf{N}\,y \vdash\ \mathsf{O}\,y,\ \mathsf{E}\,y}(\mathsf{PR})}{\mathsf{N}\,y \vdash\ \mathsf{O}\,y,\ \mathsf{O}\,\mathsf{s}y}(\mathsf{O}\,\mathsf{R})}{\mathsf{N}\,y \vdash\ \mathsf{E}\,\mathsf{s}y,\ \mathsf{O}\,\mathsf{s}y}(\mathsf{E}\,\mathsf{R}_2)}{x = \mathsf{s}y, \mathsf{N}\,y \vdash\ \mathsf{E}\,x,\ \mathsf{O}\,x}(=\mathsf{L})}{\mathsf{N}\,x \vdash\ \mathsf{E}\,x,\ \mathsf{O}\,x}(\mathsf{Case}\ \mathsf{N})$$

Extracting Semantic Orderings: Example (1)

$$\Rightarrow \mathsf{N}\,0$$
$$\mathsf{N}\,x \Rightarrow \mathsf{N}\,\mathsf{s}x$$
$$\Rightarrow \mathsf{E}\,0$$
$$\mathsf{O}\,x \Rightarrow \mathsf{E}\,\mathsf{s}x$$
$$\mathsf{E}\,x \Rightarrow \mathsf{O}\,\mathsf{s}x$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \mathsf{E}\,0,\ \mathsf{O}\,0}(\mathsf{E}\,\mathsf{R}_1)}{x = 0 \vdash \mathsf{E}\,x,\ \mathsf{O}\,x}(=\mathsf{L}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\mathsf{N}\,x \vdash \mathsf{E}\,x,\ \mathsf{O}\,x}{\mathsf{N}\,y \vdash \mathsf{E}\,y,\ \mathsf{O}\,y}(\mathsf{Subst})}{\mathsf{N}\,y \vdash\ \mathsf{O}\,y,\ \mathsf{E}\,y}(\mathsf{PR})}{\mathsf{N}\,y \vdash\ \mathsf{O}\,y,\ \mathsf{O}\,\mathsf{s}y}(\mathsf{O}\,\mathsf{R})}{\mathsf{N}\,y \vdash\ \mathsf{E}\,\mathsf{s}y,\ \mathsf{O}\,\mathsf{s}y}(\mathsf{E}\,\mathsf{R}_2)}{x = \mathsf{s}y, \mathsf{N}\,y \vdash\ \mathsf{E}\,x,\ \mathsf{O}\,x}(=\mathsf{L})}{\mathsf{N}\,x \vdash\ \mathsf{E}\,x,\ \mathsf{O}\,x}(\mathsf{Case}\ \mathsf{N})$$

Extracting Semantic Orderings: Example (1)

$$\Rightarrow \mathrm{N}\,0$$
$$\mathrm{N}\,x \Rightarrow \mathrm{N}\,\mathrm{s}x$$
$$\Rightarrow \mathrm{E}\,0$$
$$\mathrm{O}\,x \Rightarrow \mathrm{E}\,\mathrm{s}x$$
$$\mathrm{E}\,x \Rightarrow \mathrm{O}\,\mathrm{s}x$$

$$\dfrac{\dfrac{}{\vdash \mathrm{E}\,0,\ \mathrm{O}\,0}\,(\mathrm{E}\,\mathrm{R}_1)}{x = 0 \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}\,(=\mathrm{L}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\mathrm{N}\,x \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}{\mathrm{N}\,y \vdash \mathrm{E}\,y,\ \mathrm{O}\,y}\,(\mathrm{Subst})}{\mathrm{N}\,y \vdash \mathrm{O}\,y,\ \mathrm{E}\,y}\,(\mathrm{PR})}{\mathrm{N}\,y \vdash \mathrm{O}\,y,\ \mathrm{O}\,\mathrm{s}y}\,(\mathrm{O}\,\mathrm{R})}{\mathrm{N}\,y \vdash \mathrm{E}\,\mathrm{s}y,\ \mathrm{O}\,\mathrm{s}y}\,(\mathrm{E}\,\mathrm{R}_2)}{x = \mathrm{s}y, \mathrm{N}\,y \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}\,(=\mathrm{L})$$

$$\dfrac{x = 0 \vdash \mathrm{E}\,x,\ \mathrm{O}\,x \qquad x = \mathrm{s}y, \mathrm{N}\,y \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}{\mathrm{N}\,x \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}\,(\mathrm{Case}\ \mathrm{N})$$

Extracting Semantic Orderings: Example (1)

$$\Rightarrow \mathrm{N}\,0$$

$$\mathrm{N}\,x \Rightarrow \mathrm{N}\,\mathrm{s}x$$

$$\Rightarrow \mathrm{E}\,0$$

$$\mathrm{O}\,x \Rightarrow \mathrm{E}\,\mathrm{s}x$$

$$\mathrm{E}\,x \Rightarrow \mathrm{O}\,\mathrm{s}x$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \mathrm{E}\,0,\ \mathrm{O}\,0}(\mathrm{E}\,\mathrm{R}_1)}{x = 0 \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}(=\mathrm{L}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\mathrm{N}\,x \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}{\mathrm{N}\,y \vdash \mathrm{E}\,y,\ \mathrm{O}\,y}(\mathrm{Subst})}{\mathrm{N}\,y \vdash \mathrm{O}\,y,\ \mathrm{E}\,y}(\mathrm{PR})}{\mathrm{N}\,y \vdash \mathrm{O}\,y,\ \mathrm{O}\,\mathrm{s}y}(\mathrm{O}\,\mathrm{R})}{\mathrm{N}\,y \vdash \mathrm{E}\,\mathrm{s}y,\ \mathrm{O}\,\mathrm{s}y}(\mathrm{E}\,\mathrm{R}_2)}{x = \mathrm{s}y, \mathrm{N}\,y \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}(=\mathrm{L})}{\mathrm{N}\,x \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}(\mathrm{Case\ N})$$

Extracting Semantic Orderings: Example (1)

$$\Rightarrow \mathrm{N}\,0$$

$$\mathrm{N}\,x \Rightarrow \mathrm{N}\,\mathrm{s}x$$

$$\Rightarrow \mathrm{E}\,0$$

$$\mathrm{O}\,x \Rightarrow \mathrm{E}\,\mathrm{s}x$$

$$\mathrm{E}\,x \Rightarrow \mathrm{O}\,\mathrm{s}x$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \mathrm{E}\,0,\ \mathrm{O}\,0}(\mathrm{E}\,\mathrm{R}_1)}{x = 0 \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}(=\mathrm{L}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\mathrm{N}\,x \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}{\mathrm{N}\,y \vdash \mathrm{E}\,y,\ \mathrm{O}\,y}(\mathrm{Subst})}{\mathrm{N}\,y \vdash \mathrm{O}\,y,\ \mathrm{E}\,y}(\mathrm{PR})}{\mathrm{N}\,y \vdash \mathrm{O}\,y,\ \mathrm{O}\,\mathrm{s}y}(\mathrm{O}\,\mathrm{R})}{\mathrm{N}\,y \vdash \mathrm{E}\,\mathrm{s}y,\ \mathrm{O}\,\mathrm{s}y}(\mathrm{E}\,\mathrm{R}_2)}{x = \mathrm{s}y,\ \mathrm{N}\,y \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}(=\mathrm{L})}{\mathrm{N}\,x \vdash \mathrm{E}\,x,\ \mathrm{O}\,x}(\mathrm{Case\ N})$$

Extracting Semantic Orderings: Example (1)

This trace is

- fully maximal: the final predicate is introduced by its rule
- grounded: the final predicate is derived from a zero premise production (N.B. $\forall m : m \in [\![\mathrm{N}\,0]\!]_1$)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x}{\mathrm{N}\,y \vdash \mathrm{E}\,y, \mathrm{O}\,y}(\text{Subst})}{\mathrm{N}\,y \vdash \mathrm{O}\,y, \mathrm{E}\,y}(\text{PR})}{\mathrm{N}\,y \vdash \mathrm{O}\,y, \mathrm{O}\,sy}(\mathrm{O}\,\mathrm{R})}{\mathrm{N}\,y \vdash \mathrm{E}\,sy, \mathrm{O}\,sy}(\mathrm{E}\,\mathrm{R}_2)}{x = sy, \mathrm{N}\,y \vdash \mathrm{E}\,x, \mathrm{O}\,x}(=\mathrm{L}) \qquad \dfrac{\dfrac{}{\vdash \mathrm{E}\,0, \mathrm{O}\,0}(\mathrm{E}\,\mathrm{R}_1)}{x = 0 \vdash \mathrm{E}\,x, \mathrm{O}\,x}(=\mathrm{L})}{\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x}(\text{Case N})$$

Extracting Semantic Orderings: Example (1)

$$\Rightarrow \mathsf{N}\,0$$

$$\mathsf{N}\,x \Rightarrow \mathsf{N}\,\mathsf{s}x$$

$$\Rightarrow \mathsf{E}\,0$$

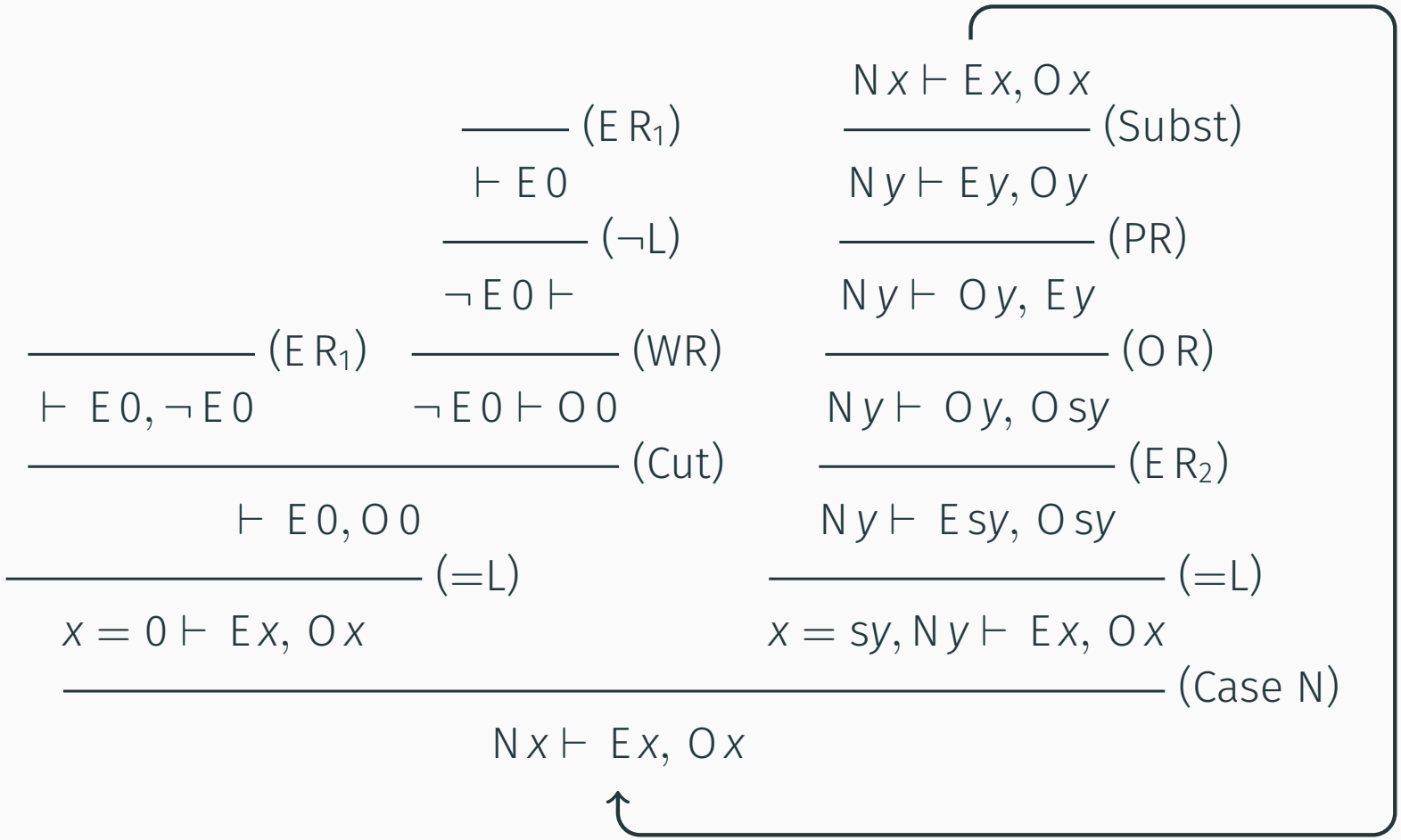
$$\mathsf{O}\,x \Rightarrow \mathsf{E}\,\mathsf{s}x$$

$$\mathsf{E}\,x \Rightarrow \mathsf{O}\,\mathsf{s}x$$

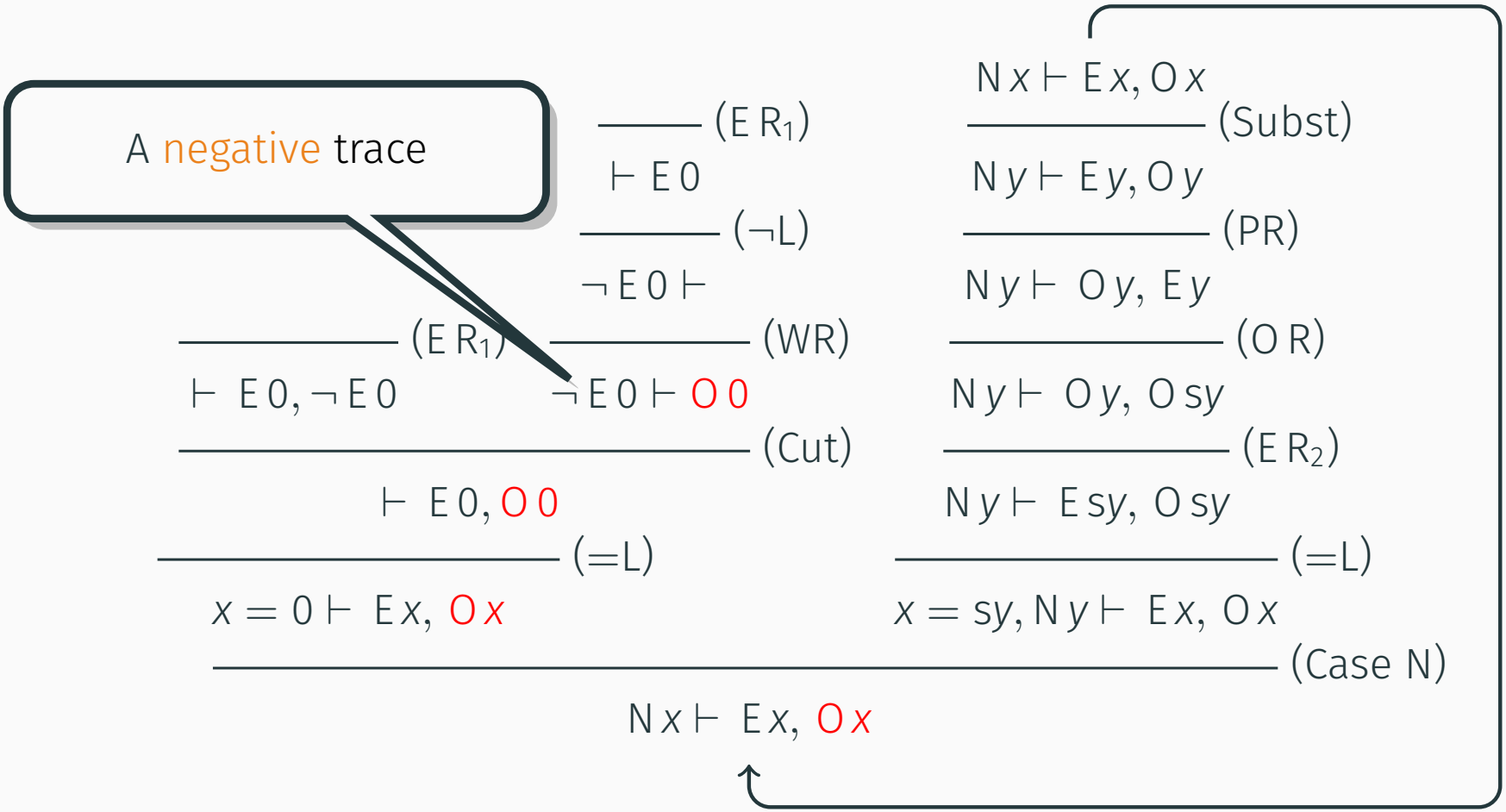
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \mathsf{E}\,0, \mathsf{O}\,0}(\mathsf{E}\,\mathsf{R}_1)}{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x}(=\mathsf{L}) \quad \dfrac{\dfrac{\dfrac{\dfrac{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}{\mathsf{N}\,y \vdash \mathsf{E}\,y, \mathsf{O}\,y}(\mathsf{Subst})}{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{E}\,y}(\mathsf{PR})}{\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{O}\,\mathsf{s}y}(\mathsf{O}\,\mathsf{R})}{\mathsf{N}\,y \vdash \mathsf{E}\,\mathsf{s}y, \mathsf{O}\,\mathsf{s}y}(\mathsf{E}\,\mathsf{R}_2)}{x = \mathsf{s}y, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x}(=\mathsf{L})}{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}(\mathsf{Case}\ \mathsf{N})$$

Not grounded!

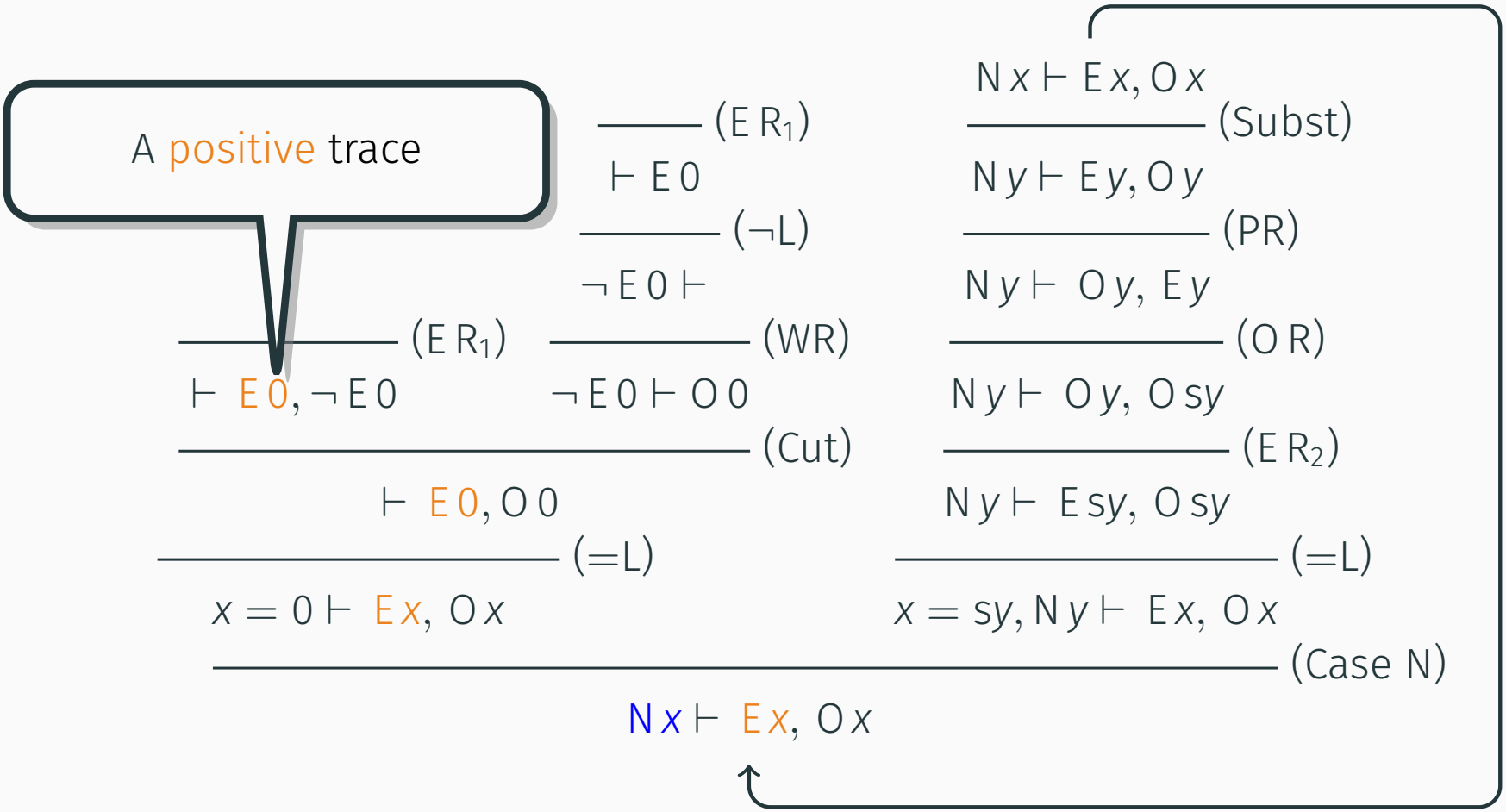
Extracting Semantic Orderings: Example (1)



Extracting Semantic Orderings: Example (1)



Extracting Semantic Orderings: Example (1)



Extracting Semantic Orderings: Example (2)

$$\Rightarrow \mathrm{N}\,0$$

$$\mathrm{N}\,x \Rightarrow \mathrm{N}\,\mathrm{s}x$$

$$\Rightarrow \mathrm{E}\,0$$

$$\mathrm{O}\,x \Rightarrow \mathrm{E}\,\mathrm{s}x$$

$$\mathrm{E}\,x \Rightarrow \mathrm{O}\,\mathrm{s}x$$

$$\mathrm{N}\,\mathrm{ss}0 \Rightarrow \mathrm{O}\,\mathrm{sss}0$$

$$\dfrac{\dfrac{\dfrac{}{\mathrm{N}\,\mathrm{ss}0 \vdash \mathrm{N}\,\mathrm{ss}0}\,(\mathrm{Ax})}{\mathrm{N}\,\mathrm{ss}0 \vdash \mathrm{N}\,\mathrm{sss}0}\,(\mathrm{N}\,R_2)}{y = \mathrm{sss}0, \mathrm{N}\,\mathrm{ss}0 \vdash \mathrm{N}\,y}\,(=\mathrm{L}) \qquad \dfrac{\dfrac{\mathrm{E}\,x \vdash \mathrm{N}\,x}{\mathrm{E}\,z \vdash \mathrm{N}\,z}\,(\mathrm{Subst})}{\mathrm{E}\,z \vdash \mathrm{N}\,\mathrm{s}z}\,(\mathrm{N}\,R_2)}{y = \mathrm{s}z, \mathrm{E}\,z \vdash \mathrm{N}\,y}\,(=\mathrm{L})$$

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathrm{N}\,0}\,(\mathrm{N}\,R_1)}{x = 0 \vdash \mathrm{N}\,x}\,(=\mathrm{L}) \qquad \dfrac{\dfrac{\mathrm{O}\,y \vdash \mathrm{N}\,y}{\mathrm{O}\,y \vdash \mathrm{N}\,\mathrm{s}y}\,(\mathrm{N}\,R_2)}{x = \mathrm{s}y, \mathrm{O}\,y \vdash \mathrm{N}\,x}\,(=\mathrm{L})}{\mathrm{E}\,x \vdash \mathrm{N}\,x}\,(\mathrm{Case\ E})$$

$$\mathrm{E}\,x \vdash \mathrm{N}\,x$$

Extracting Semantic Orderings: Example (2)

$$\Rightarrow \mathrm{N}\,0$$

$$\mathrm{N}\,x \Rightarrow \mathrm{N}\,\mathrm{s}x$$

$$\Rightarrow \mathrm{E}\,0$$

$$\mathrm{O}\,x \Rightarrow \mathrm{E}\,\mathrm{s}x$$

$$\mathrm{E}\,x \Rightarrow \mathrm{O}\,\mathrm{s}x$$

$$\mathrm{N}\,\mathrm{ss}0 \Rightarrow \mathrm{O}\,\mathrm{sss}0$$

$$\dfrac{\dfrac{\dfrac{}{\mathrm{N}\,\mathrm{ss}0 \vdash \mathrm{N}\,\mathrm{ss}0}\,(\mathrm{Ax})}{\mathrm{N}\,\mathrm{ss}0 \vdash \mathrm{N}\,\mathrm{sss}0}\,(\mathrm{N}\,R_2)}{y = \mathrm{sss}0, \mathrm{N}\,\mathrm{ss}0 \vdash \mathrm{N}\,y}\,(=\mathrm{L}) \qquad \dfrac{\dfrac{\mathrm{E}\,z \vdash \mathrm{N}\,z}{\mathrm{E}\,z \vdash \mathrm{N}\,\mathrm{s}z}\,(\mathrm{N}\,R_2)}{y = \mathrm{s}z, \mathrm{E}\,z \vdash \mathrm{N}\,y}\,(=\mathrm{L})$$

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathrm{N}\,0}\,(\mathrm{N}\,R_1)}{x = 0 \vdash \mathrm{N}\,x}\,(=\mathrm{L}) \qquad \dfrac{\dfrac{\mathrm{O}\,y \vdash \mathrm{N}\,y}{\mathrm{O}\,y \vdash \mathrm{N}\,\mathrm{s}y}\,(\mathrm{N}\,R_2)}{x = \mathrm{s}y, \mathrm{O}\,y \vdash \mathrm{N}\,x}\,(=\mathrm{L})}{\mathrm{E}\,x \vdash \mathrm{N}\,x}\,(\mathrm{Case\ E})$$

$$\dfrac{\mathrm{E}\,x \vdash \mathrm{N}\,x}{\mathrm{E}\,z \vdash \mathrm{N}\,z}\,(\mathrm{Subst})$$

$$\dfrac{\dots}{\mathrm{O}\,y \vdash \mathrm{N}\,y}\,(\mathrm{Case\ O})$$

Extracting Semantic Orderings: Example (2)

$$\Rightarrow \mathrm{N}\,0$$

$$\mathrm{N}\,x \Rightarrow \mathrm{N}\,\mathrm{s}x$$

$$\Rightarrow \mathrm{E}\,0$$

$$\mathrm{O}\,x \Rightarrow \mathrm{E}\,\mathrm{s}x$$

$$\mathrm{E}\,x \Rightarrow \mathrm{O}\,\mathrm{s}x$$

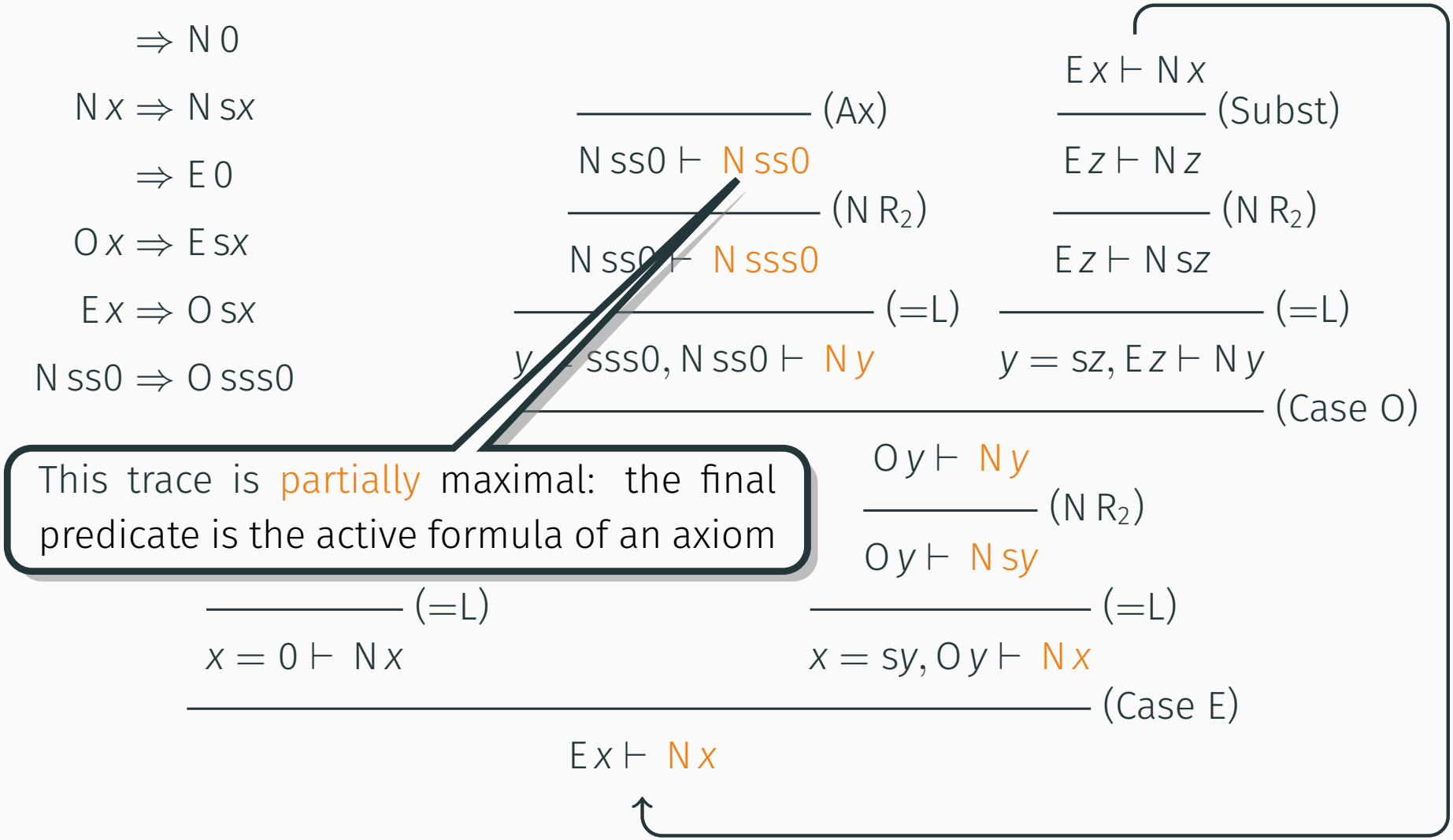
$$\mathrm{N}\,\mathrm{ss}0 \Rightarrow \mathrm{O}\,\mathrm{sss}0$$

$$\dfrac{\dfrac{}{\mathrm{N}\,\mathrm{ss}0 \vdash \mathrm{N}\,\mathrm{ss}0}\,(\mathrm{Ax})}{\mathrm{N}\,\mathrm{ss}0 \vdash \mathrm{N}\,\mathrm{sss}0}\,(\mathrm{N}\,R_2) \qquad \dfrac{\dfrac{\mathrm{E}\,x \vdash \mathrm{N}\,x}{\mathrm{E}\,z \vdash \mathrm{N}\,z}\,(\mathrm{Subst})}{\mathrm{E}\,z \vdash \mathrm{N}\,\mathrm{s}z}\,(\mathrm{N}\,R_2)$$

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathrm{N}\,0}\,(\mathrm{N}\,R_1)}{x = 0 \vdash \mathrm{N}\,x}\,(=\mathrm{L}) \qquad \dfrac{\dfrac{\mathrm{O}\,y \vdash \mathrm{N}\,y}{\mathrm{O}\,y \vdash \mathrm{N}\,\mathrm{s}y}\,(\mathrm{N}\,R_2)}{x = \mathrm{s}y, \mathrm{O}\,y \vdash \mathrm{N}\,x}\,(=\mathrm{L})}{\mathrm{E}\,x \vdash \mathrm{N}\,x}\,(\mathrm{Case\ E})$$

$$\dfrac{\dfrac{\mathrm{N}\,\mathrm{ss}0 \vdash \mathrm{N}\,\mathrm{sss}0}{y = \mathrm{sss}0, \mathrm{N}\,\mathrm{ss}0 \vdash \mathrm{N}\,y}\,(=\mathrm{L}) \qquad \dfrac{\mathrm{E}\,z \vdash \mathrm{N}\,\mathrm{s}z}{y = \mathrm{s}z, \mathrm{E}\,z \vdash \mathrm{N}\,y}\,(=\mathrm{L})}{\mathrm{E}\,x \vdash \mathrm{N}\,x}\,(\mathrm{Case\ O})$$

Extracting Semantic Orderings: Example (2)



Extracting Semantic Orderings: Example (2)

$$\Rightarrow \mathrm{N}\,0$$

$$\mathrm{N}\,x \Rightarrow \mathrm{N}\,\mathrm{s}x$$

$$\Rightarrow \mathrm{E}\,0$$

$$\mathrm{O}\,x \Rightarrow \mathrm{E}\,\mathrm{s}x$$

$$\mathrm{E}\,x \Rightarrow \mathrm{O}\,\mathrm{s}x$$

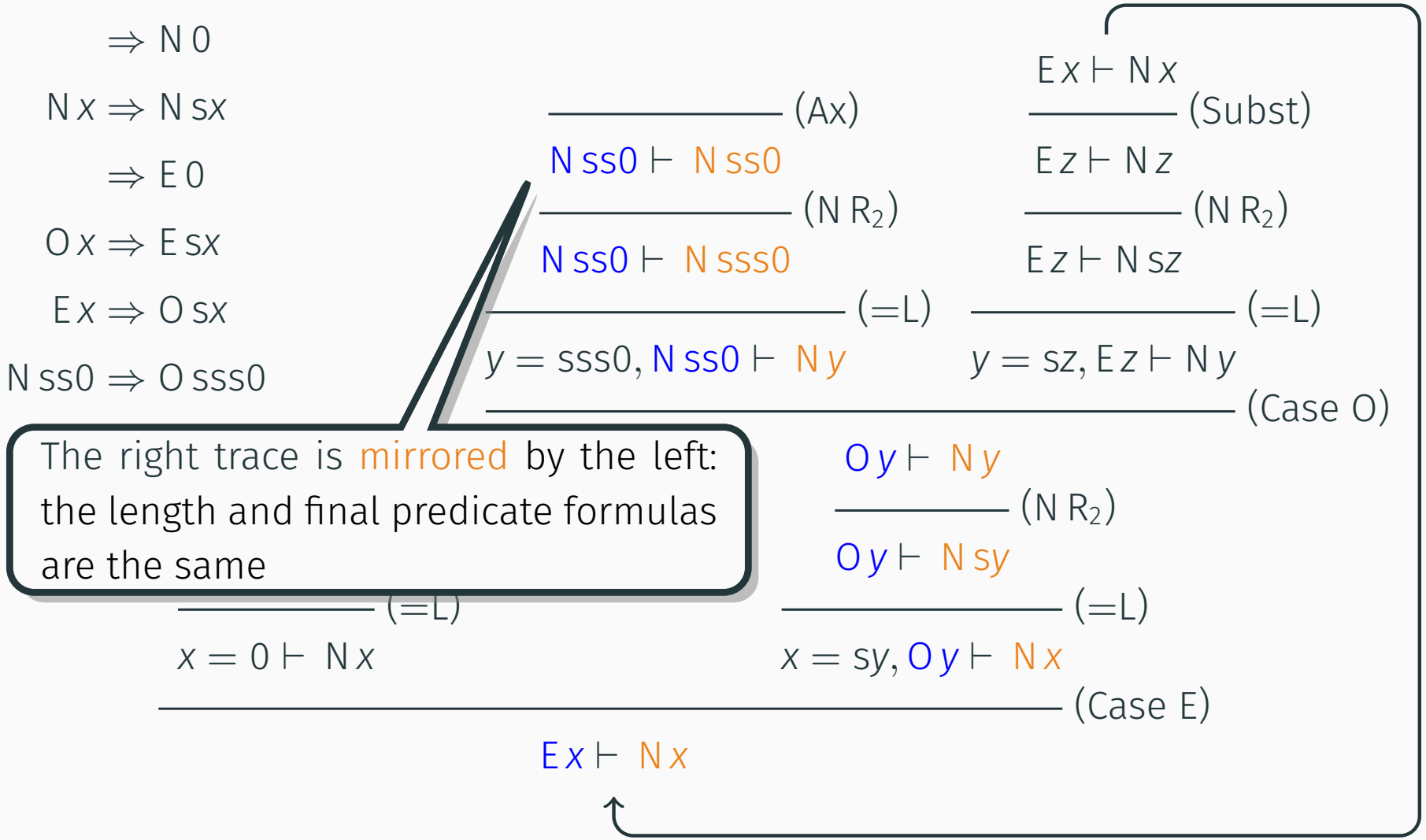
$$\mathrm{N}\,\mathrm{ss0} \Rightarrow \mathrm{O}\,\mathrm{sss0}$$

$$\dfrac{\dfrac{\dfrac{}{\mathrm{N}\,\mathrm{ss0} \vdash \mathrm{N}\,\mathrm{ss0}}\,(\mathrm{Ax})}{\mathrm{N}\,\mathrm{ss0} \vdash \mathrm{N}\,\mathrm{sss0}}\,(\mathrm{N}\,R_2)}{y = \mathrm{sss0}, \mathrm{N}\,\mathrm{ss0} \vdash \mathrm{N}\,y}\,(=\mathrm{L}) \qquad \dfrac{\dfrac{\dfrac{\mathrm{E}\,x \vdash \mathrm{N}\,x}{\mathrm{E}\,z \vdash \mathrm{N}\,z}\,(\mathrm{Subst})}{\mathrm{E}\,z \vdash \mathrm{N}\,\mathrm{sz}}\,(\mathrm{N}\,R_2)}{y = \mathrm{sz}, \mathrm{E}\,z \vdash \mathrm{N}\,y}\,(=\mathrm{L})$$

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathrm{N}\,0}\,(\mathrm{N}\,R_1)}{x = 0 \vdash \mathrm{N}\,x}\,(=\mathrm{L}) \qquad \dfrac{\dfrac{\dfrac{\mathrm{O}\,y \vdash \mathrm{N}\,y}{\mathrm{O}\,y \vdash \mathrm{N}\,\mathrm{s}y}\,(\mathrm{N}\,R_2)}{x = \mathrm{s}y, \mathrm{O}\,y \vdash \mathrm{N}\,x}\,(=\mathrm{L})}{\mathrm{E}\,x \vdash \mathrm{N}\,x}\,(\mathrm{Case\ E})$$

$$\mathrm{E}\,x \vdash \mathrm{N}\,x$$

Extracting Semantic Orderings: Example (2)



Extracting Semantic Orderings: A Realizability Condition

Definition (Realizability Condition)

For every positive maximal right-hand trace, there must exist a left-hand trace following the same path such that:

- either the right-hand trace is grounded, or it is partially maximal and mirrored by the left-hand trace
- right unfoldings $\leq$ left unfoldings

Soundness of the Realizability Condition

Theorem

Suppose $\mathcal{P}$ is a cyclic proof of $\mathsf{P}\,\vec{x} \vdash \mathsf{Q}\,\vec{y}$ satisfying the realizability condition, then $[\![\mathsf{P}\,\vec{x}]\!]_\alpha \subseteq [\![\mathsf{Q}\,\vec{y}]\!]_\alpha$ for all α

Proof.

Soundness of the Realizability Condition

Theorem

Suppose $\mathcal{P}$ is a cyclic proof of $\mathrm{P}\,\vec{x} \vdash \mathrm{Q}\,\vec{y}$ satisfying the realizability condition, then $[\![\mathrm{P}\,\vec{x}]\!]_\alpha \subseteq [\![\mathrm{Q}\,\vec{y}]\!]_\alpha$ for all α

Proof.

Pick a model $m \in [\![\mathrm{P}\,\vec{x}]\!]_\alpha$ (i.e. $\Theta(\mathrm{P}\,\vec{x}, m) \leq \alpha$)

- m corresponds to a positive maximal right-hand trace in $\mathcal{P}$
- The number of unfoldings in this right-hand trace is an upper bound on $\Theta(\mathrm{Q}\,\vec{y}, m)$
- The number of unfoldings in any left-hand trace following the same path is a lower bound on $\Theta(\mathrm{P}\,\vec{x}, m)$
- From the realizability condition, we have that $\Theta(\mathrm{Q}\,\vec{y}, m) \leq \Theta(\mathrm{P}\,\vec{x}, m)$
- Because approximations grow monotonically, also $m \in [\![\mathrm{Q}\,\vec{y}]\!]_\alpha$

The Descending Model Property

The inference rules satify the property that for all valid rule instances

$$\frac{\Delta_1 \vdash \Pi_1 \quad \ldots \quad \Delta_n \vdash \Pi_n}{\Gamma[\mathrm{P}\,\vec{t}] \vdash \Sigma[\mathrm{Q}\,\vec{u}]}$$

If there is a model $m \models \Gamma$, then there is a model $m' \models \Delta_i$, and either $\mathrm{Q}\,\vec{u}$ is terminal or there is a trace to some $\mathrm{R}\,\vec{v} \in \Pi_i$; moreover if there is a trace:

- from $\mathrm{P}\,\vec{t}$ to $\mathrm{P'}\,\vec{s} \in \Delta_i$ then $\Theta(\mathrm{P'}\,\vec{s}, m') \leq \Theta(\mathrm{P}\,\vec{t}, m)$
  ($<$ if the trace progresses)

- from $\mathrm{Q}\,\vec{u}$ to $\mathrm{Q'}\,\vec{r} \in \Pi_i$ and $\Theta(\mathrm{Q}\,\vec{u})$ defined, then $\Theta(\mathrm{Q'}\,\vec{r}, m') \geq \Theta(\mathrm{Q}\,\vec{u}, m)$
  ($>$ if the trace progresses)

The Descending Model Property

Almost all the inference rules satisfy the descending model property!

Consider the following two rules (for some unsatisfiable F):

$$(\neg\mathrm{R}):\ \frac{\Gamma, F \vdash \Delta}{\Gamma \vdash \neg F, \Delta} \qquad (\rightarrow\mathrm{R}):\ \frac{\Gamma, F \vdash G, \Delta}{\Gamma \vdash F \rightarrow G, \Delta}$$

The conclusions may have models, but the premises may not

Deciding the Realizability Condition

- We use weighted automata to decide whether the realizability condition holds
- We construct weighted automata that count the progression points in left and right-hand traces
- The realizability condition corresponds to an inclusion of the right-hand trace automaton within the left-hand one

Weighted Automata

Definition (Weighted Automata)

Let Σ be an alphabet, and $(V, \oplus, \otimes)$ a semiring of weights. A weighted automaton $\mathscr{A}$ is a tuple (Q, q_I, F, γ) consisting of a set Q of states containing an initial state $q_I \in Q$, a set $F \subseteq Q$ of final states, and a weighted transition function $\gamma : (Q \times \Sigma \times Q) \to V$.

Weighted Automata

Definition (Weighted Automata)

Let Σ be an alphabet, and $(V, \oplus, \otimes)$ a semiring of weights. A weighted automaton $\mathscr{A}$ is a tuple (Q, q_I, F, γ) consisting of a set Q of states containing an initial state $q_I \in Q$, a set $F \subseteq Q$ of final states, and a weighted transition function $\gamma : (Q \times \Sigma \times Q) \to V$.

1. The value of a run of $\mathscr{A}$ is the semiring product of all its transitions
2. The value of a word is the semiring sum of all runs accepting that word
3. The quantitative language $\mathcal{L}_{\mathscr{A}}$ is the function $\Sigma^* \rightharpoonup V$ computed by $\mathscr{A}$

Weighted Automata

Definition (Weighted Automata)

Let Σ be an alphabet, and $(V, \oplus, \otimes)$ a semiring of weights. A weighted automaton $\mathscr{A}$ is a tuple (Q, q_I, F, γ) consisting of a set Q of states containing an initial state $q_I \in Q$, a set $F \subseteq Q$ of final states, and a weighted transition function $\gamma : (Q \times \Sigma \times Q) \to V$.

1. The value of a run of $\mathscr{A}$ is the semiring product of all its transitions
2. The value of a word is the semiring sum of all runs accepting that word
3. The quantitative language $\mathcal{L}_{\mathscr{A}}$ is the function $\Sigma^* \rightharpoonup V$ computed by $\mathscr{A}$

Definition (Weighted Inclusion)

$\mathcal{L}_1 \leq \mathcal{L}_2$ if and only if for every word w such that $\mathcal{L}_1(w)$ is defined, $\mathcal{L}_2(w)$ is also defined and $\mathcal{L}_1(w) \leq \mathcal{L}_2(w)$

Weighted Automata

Definition (Weighted Automata)

Let Σ be an alphabet, and $(V, \oplus, \otimes)$ a semiring of weights. A weighted automaton $\mathscr{A}$ is a tuple (Q, q_I, F, γ) consisting of a set Q of states containing an initial state $q_I \in Q$, a set $F \subseteq Q$ of final states, and a weighted transition function $\gamma : (Q \times \Sigma \times Q) \to V$.

1. The value of a run of $\mathscr{A}$ is the semiring product of all its transitions
2. The value of a word is the semiring sum of all runs accepting that word
3. The quantitative language $\mathcal{L}_{\mathscr{A}}$ is the function $\Sigma^* \rightharpoonup V$ computed by $\mathscr{A}$

Definition (Weighted Inclusion)

$\mathcal{L}_1 \leq \mathcal{L}_2$ if and only if for every word w such that $\mathcal{L}_1(w)$ is defined, $\mathcal{L}_2(w)$ is also defined and $\mathcal{L}_1(w) \leq \mathcal{L}_2(w)$

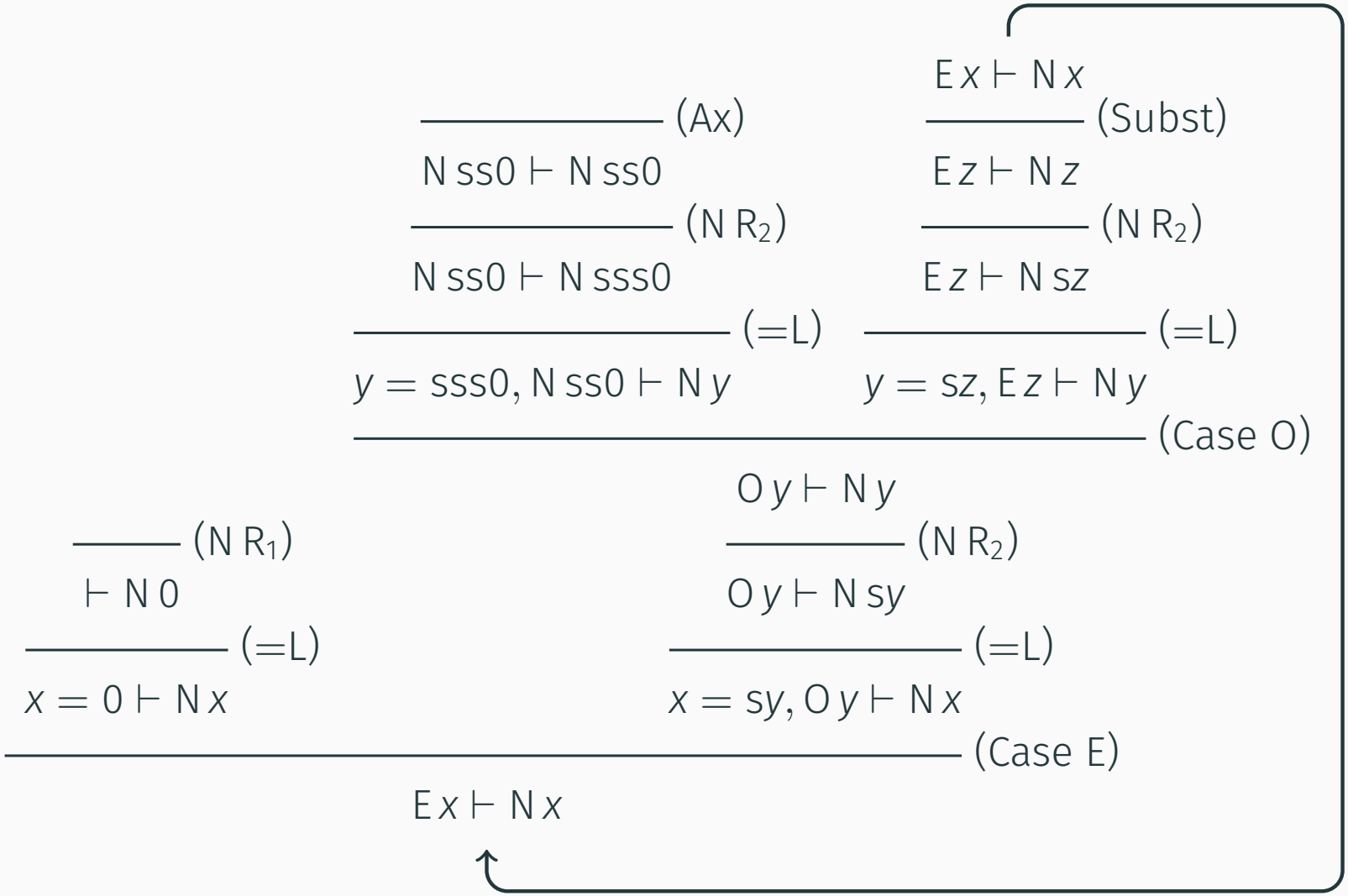
Sum automata are weighted automata over $(\mathbb{N}, \max, +)$

Weighted Automata from Cyclic Entailment Proofs

Given a cyclic entailment proof $\mathcal{P}$, we can construct two sum automata, $\mathscr{A}_\mathcal{P}$ and $\mathscr{C}_\mathcal{P}$:

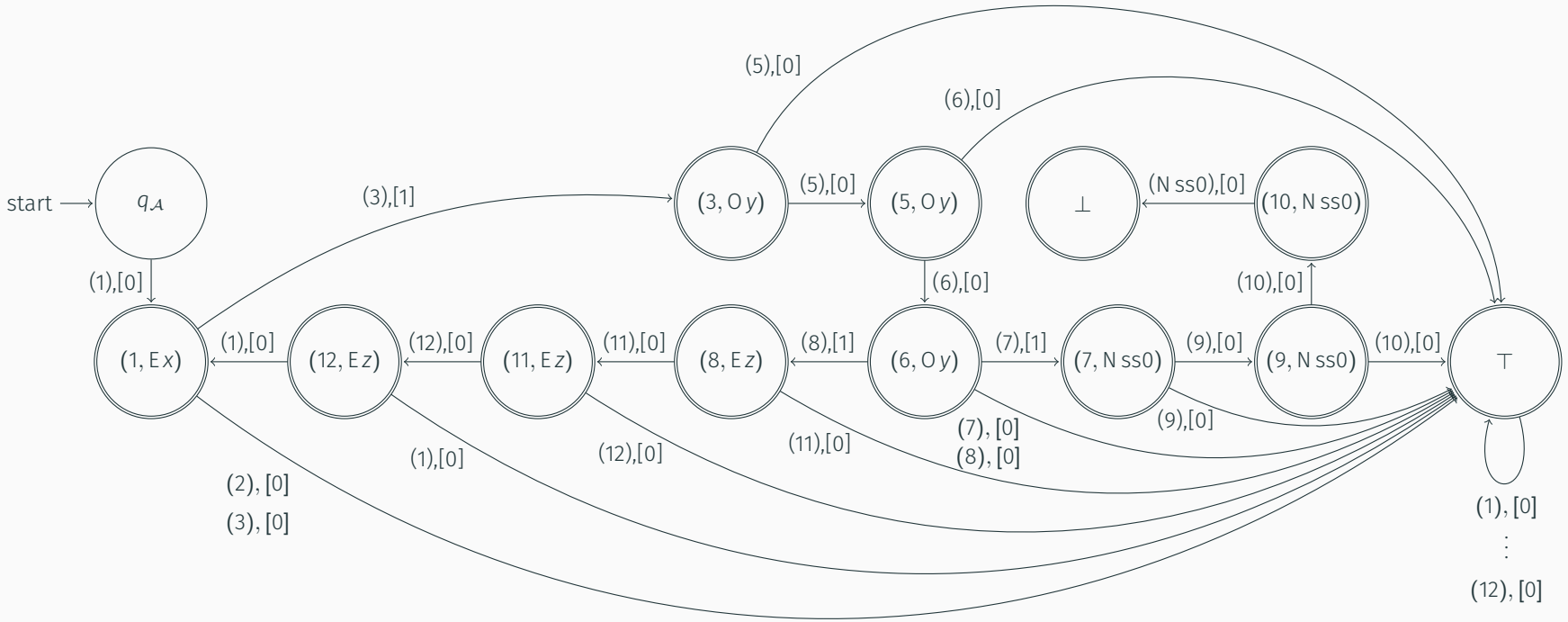
- States represent a particular trace value in a particular node
- The words accepted are paths in the proof from the root sequent
- Transitions corresponding to a case split have non-zero (unit) weight
    - The value of a path is the maximum number of unfoldings in the traces along the path
- $\mathscr{A}_\mathcal{P}$ can stop tracking traces at any point
- $\mathscr{C}_\mathcal{P}$ always tracks traces (i.e. considers only maximal ones)
- $\mathscr{C}_\mathcal{P}$ is grounded when all final states correspond to ground predicate instances

Weighted Automata from Cyclic Entailment Proofs



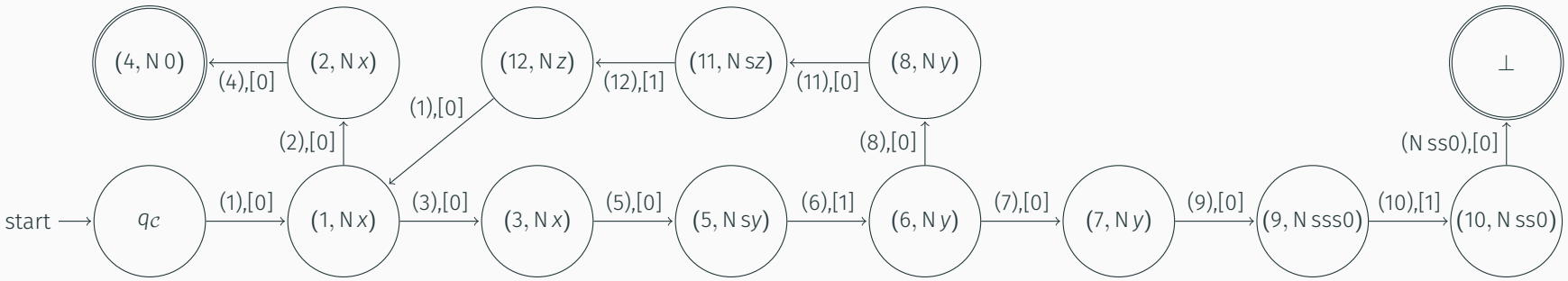
Weighted Automata from Cyclic Entailment Proofs

The left-hand automaton for the example proof of $\mathsf{E}\,x \vdash \mathsf{N}\,x$



Weighted Automata from Cyclic Entailment Proofs

The right-hand automaton for the example proof of $\mathsf{E}\,x \vdash \mathsf{N}\,x$



An Equivalence between Realizability and Weighted Inclusion

Theorem

$\mathcal{P}$ satisfies the realizability condition $\Leftrightarrow \mathscr{C}_{\mathcal{P}} \leq \mathscr{A}_{\mathcal{P}}$ and $\mathscr{C}_{\mathcal{P}}$ is grounded

An Equivalence between Realizability and Weighted Inclusion

Theorem

$\mathcal{P}$ satisfies the realizability condition $\Leftrightarrow \mathscr{C}_{\mathcal{P}} \leq \mathscr{A}_{\mathcal{P}}$ and $\mathscr{C}_{\mathcal{P}}$ is grounded

Theorem (Krob '94, Almagor Et Al. '11)

Given two quantitative languages (weighted automata) $\mathcal{L}_1$ and $\mathcal{L}_2$, it is undecidable whether $\mathcal{L}_1 \leq \mathcal{L}_2$

An Equivalence between Realizability and Weighted Inclusion

Theorem

$\mathcal{P}$ satisfies the realizability condition $\Leftrightarrow \mathscr{C}_{\mathcal{P}} \leq \mathscr{A}_{\mathcal{P}}$ and $\mathscr{C}_{\mathcal{P}}$ is grounded

Theorem (Krob '94, Almagor Et Al. '11)

Given two quantitative languages (weighted automata) $\mathcal{L}_1$ and $\mathcal{L}_2$, it is undecidable whether $\mathcal{L}_1 \leq \mathcal{L}_2$

Definition

A weighted automaton is called finite-valued if there exists a bound on the number of distinct values of accepting runs on any given word

Theorem (Filiot, Gentilini & Raskin '14)

$\mathcal{L}_{\mathscr{A}} \leq \mathcal{L}_{\mathscr{B}}$ is decidable for finite-valued weighted automata $\mathscr{A}$ and $\mathscr{B}$

Left-hand Trace Automata are not Finite-valued

The following configuration results in non-finite-valuedness:

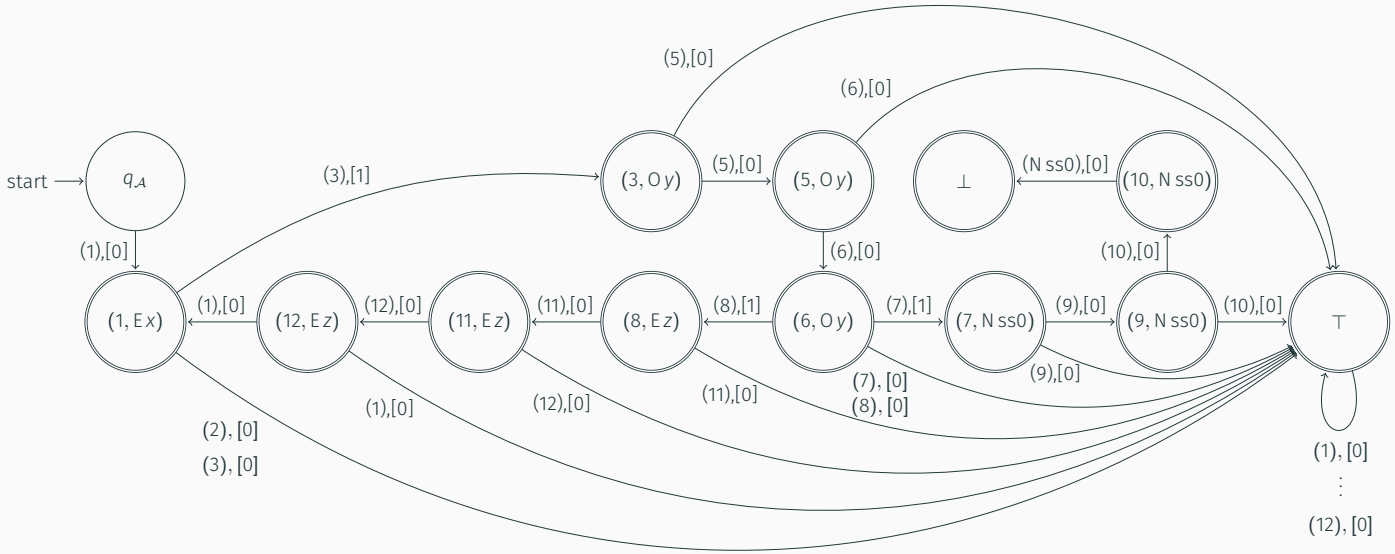
(Weber & Seidel, TCS 1991)

Left-hand Trace Automata are not Finite-valued

The following configuration results in non-finite-valuedness:

(Weber & Seidel, TCS 1991)

Consider our left-hand trace automaton:



Approximate Left-hand Trace Automata

We instead construct a sequence of left-hand trace automata $\mathscr{A}_{\mathcal{P}}[n]$ ($n \in \mathbb{N}$):

- we refine the state $\top$ into n states $\top_\nu^1, \ldots \top_\nu^n$ for each node ν of $\mathcal{P}$
- we restrict the transitions between them:
  - $\top_\nu^i$ transitions to $\top_\nu^{i+1}$ accepting only node ν
  - $\top_\nu^i$ transitions to itself accepting any node except ν

Approximate Left-hand Trace Automata

We instead construct a sequence of left-hand trace automata $\mathscr{A}_{\mathcal{P}}[n]$ ($n \in \mathbb{N}$):

- we refine the state $\top$ into n states $\top_\nu^1, \ldots \top_\nu^n$ for each node ν of $\mathcal{P}$
- we restrict the transitions between them:
  - $\top_\nu^i$ transitions to $\top_\nu^{i+1}$ accepting only node ν
  - $\top_\nu^i$ transitions to itself accepting any node except ν

Theorem

Each approximate left-hand trace automaton is finite-valued

Approximate Left-hand Trace Automata

We instead construct a sequence of left-hand trace automata $\mathscr{A}_{\mathcal{P}}[n]$ ($n \in \mathbb{N}$):

- we refine the state $\top$ into n states $\top_\nu^1, \ldots \top_\nu^n$ for each node ν of $\mathcal{P}$
- we restrict the transitions between them:
  - $\top_\nu^i$ transitions to $\top_\nu^{i+1}$ accepting only node ν
  - $\top_\nu^i$ transitions to itself accepting any node except ν

Theorem

Each approximate left-hand trace automaton is finite-valued[1]

[1] when traces are injective, i.e. do not merge

Approximate Left-hand Trace Automata

We instead construct a sequence of left-hand trace automata $\mathscr{A}_{\mathcal{P}}[n]$ ($n \in \mathbb{N}$):

- we refine the state $\top$ into n states $\top_\nu^1, \ldots \top_\nu^n$ for each node ν of $\mathcal{P}$
- we restrict the transitions between them:
  - $\top_\nu^i$ transitions to $\top_\nu^{i+1}$ accepting only node ν
  - $\top_\nu^i$ transitions to itself accepting any node except ν

Theorem

Each approximate left-hand trace automaton is finite-valued[1]

The price is that $\mathscr{A}_{\mathcal{P}}[n]$ only accepts a subset of all the paths in $\mathcal{P}$

[1] when traces are injective, i.e. do not merge

A Decidable Restriction for Realizability

Theorem

Let $\mathcal{P}$ be a cyclic entailment proof which is dynamic and balanced; then $\mathcal{P}$ satisfies the realizability condition if and only if $\mathscr{C}_{\mathcal{P}} \leq \mathscr{A}_{\mathcal{P}}[N]$ and $\mathscr{C}_{\mathcal{P}}$ is grounded (where N is a function of $\mathcal{P}$)

A Decidable Restriction for Realizability

Theorem

Let $\mathcal{P}$ be a cyclic entailment proof which is dynamic and balanced; then $\mathcal{P}$ satisfies the realizability condition if and only if $\mathscr{C}_{\mathcal{P}} \leq \mathscr{A}_{\mathcal{P}}[N]$ and $\mathscr{C}_{\mathcal{P}}$ is grounded (where N is a function of $\mathcal{P}$)

The cyclic proof is:

- dynamic when every (reachable) basic trace cycle has a non-zero number of progression points
- balanced when every (reachable) basic binary trace cycle has equal numbers of left and right-hand progression points
  - a binary cycle is a pair of left and right-hand trace cycles following the same path

The bound N is a function of other graph-theoretic quantities of $\mathcal{P}$

Corollary: Bootstrapping Cyclic Entailment Systems

Suppose we deduce $[\![\mathrm{P}\,\vec{t}]\!]_\alpha \subseteq [\![\mathrm{Q}\,\vec{u}]\!]_\alpha$ from a proof of $\Gamma, \mathrm{P}\,\vec{t} \vdash \Sigma, \mathrm{Q}\,\vec{u}$

Then we can safely trace across an active cut formula

$$\frac{\Gamma, \mathrm{P}\,\vec{t} \vdash \Sigma, \mathrm{Q}\,\vec{u} \qquad \mathrm{Q}\,\vec{u}, \Pi \vdash \Delta}{\Gamma, \mathrm{P}\,\vec{t}, \Pi \vdash \Sigma, \Delta}\ (\mathrm{Cut})$$

Corollary: Bootstrapping Cyclic Entailment Systems

Suppose we deduce $[\![\mathrm{P}\,\vec{t}]\!]_\alpha \subseteq [\![\mathrm{Q}\,\vec{u}]\!]_\alpha$ from a proof of $\Gamma, \mathrm{P}\,\vec{t} \vdash \Sigma, \mathrm{Q}\,\vec{u}$

Then we can safely trace across an active cut formula

$$\frac{\Gamma, \mathrm{P}\,\vec{t} \vdash \Sigma, \mathrm{Q}\,\vec{u} \qquad \mathrm{Q}\,\vec{u}, \Pi \vdash \Delta}{\Gamma, \mathrm{P}\,\vec{t}, \Pi \vdash \Sigma, \Delta}\ (\mathrm{Cut})$$

This is explicitly forbidden in existing cyclic proof systems, precisely because there is no way, in general, to ensure an inclusion between $[\![\mathrm{P}\,\vec{t}]\!]_\alpha$ and $[\![\mathrm{Q}\,\vec{u}]\!]_\alpha$

Conclusions

- We have shown that information about inclusions between the semantics of inductive predicates can be extracted from cyclic proofs of entailments
- This information should be useful for constructing ranking functions for programs
- Our results are formulated abstractly, and so hold for any cyclic proof system whose rules satisfy certain properties
- We use the term realizability because we extract semantic information from the proofs

Future Work

- Implement the decision procedure within the cyclic proof-based verification framework Cyclist
- Evaluate to what extent entailments found ‘in the wild’ satisfy the realizability condition
- Investigate further theoretical questions:
  - are there weaker structural properties of proofs that still admit completeness with the approximate automata
  - If the semantic inclusion $[\![\mathsf{P}\,\vec{x}]\!]_\alpha \subseteq [\![\mathsf{Q}\,\vec{y}]\!]_\alpha$ holds, is there a cyclic proof of $\mathsf{P}\,\vec{x} \vdash \mathsf{Q}\,\vec{y}$ satisfying the realizability condition?