# Realizability in Cyclic Proof

## Extracting Ordering Information for Infinite Descent

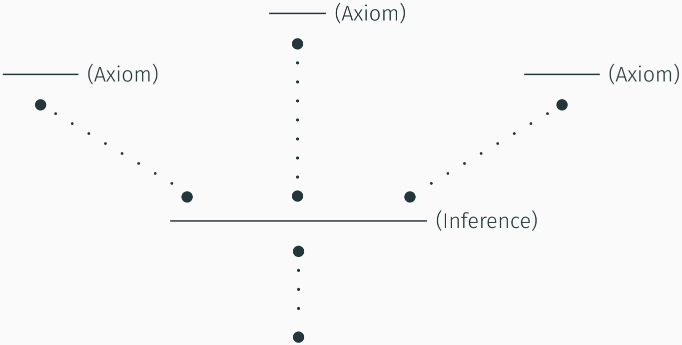
Reuben N. S. Rowe [1] James Brotherston [2]

Kent PLAS Seminar, Monday 23rd October 2017

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

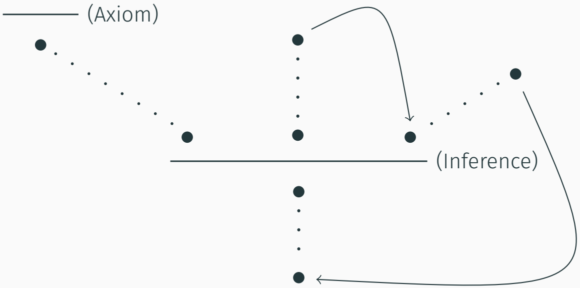
[2]Department of Computer Science, UCL, London, UK

# What is Cyclic Proof?



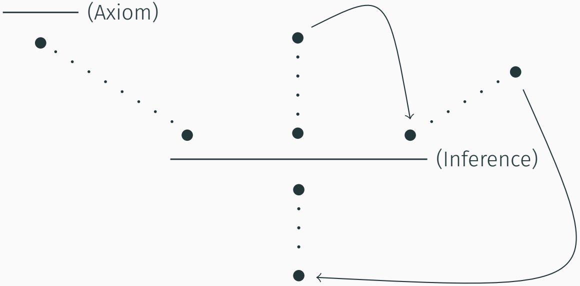

- We are all familiar with proofs as finite trees

# 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

## Example: First Order Logic

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

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

# Example: First Order Logic

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{$(\mathrm{ER}_1)$}
\UnaryInfC{$\vdash \mathrm{E}\,0, \mathrm{O}\,0$}
\RightLabel{$(=\mathrm{L})$}
\UnaryInfC{$x = 0 \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\AxiomC{$\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\RightLabel{$(\mathrm{Subst})$}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{E}\,y, \mathrm{O}\,y$}
\RightLabel{$(\mathrm{PR})$}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{O}\,y, \mathrm{E}\,y$}
\RightLabel{$(\mathrm{OR}_1)$}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{O}\,y, \mathrm{O}\,\mathrm{s}y$}
\RightLabel{$(\mathrm{ER}_2)$}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{E}\,\mathrm{s}y, \mathrm{O}\,\mathrm{s}y$}
\RightLabel{$(=\mathrm{L})$}
\UnaryInfC{$x = \mathrm{s}y, \mathrm{N}\,y \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\RightLabel{$(\mathrm{Case\ N})$}
\BinaryInfC{$\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\end{prooftree}
$$

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

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

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

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

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

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

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

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

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

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

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

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

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

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{$(\mathrm{ER}_1)$}
\UnaryInfC{$\vdash \mathrm{E}\,0, \mathrm{O}\,0$}
\RightLabel{$(=\mathrm{L})$}
\UnaryInfC{$x = 0 \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\AxiomC{$\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\RightLabel{(Subst)}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{E}\,y, \mathrm{O}\,y$}
\RightLabel{(PR)}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{O}\,y, \mathrm{E}\,y$}
\RightLabel{$(\mathrm{OR}_1)$}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{O}\,y, \mathrm{O}\,\mathrm{s}y$}
\RightLabel{$(\mathrm{ER}_2)$}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{E}\,\mathrm{s}y, \mathrm{O}\,\mathrm{s}y$}
\RightLabel{$(=\mathrm{L})$}
\UnaryInfC{$x = \mathrm{s}y, \mathrm{N}\,y \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\RightLabel{(Case N)}
\BinaryInfC{$\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\end{prooftree}
$$

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

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

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

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

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

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

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

$$
\dfrac{
\dfrac{\dfrac{}{\vdash \mathsf{E}\,0, \mathsf{O}\,0}\,(\mathrm{ER}_1)}{x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathrm{L})
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x}{\color{orange}{\mathsf{N}\,y} \vdash \mathsf{E}\,y, \mathsf{O}\,y}\,(\mathrm{Subst})}{\color{blue}{\mathsf{N}\,y} \vdash \mathsf{O}\,y, \mathsf{E}\,y}\,(\mathrm{PR})}{\color{blue}{\mathsf{N}\,y} \vdash \mathsf{O}\,y, \mathsf{O}\,\mathsf{s}y}\,(\mathrm{OR}_1)}{\color{blue}{\mathsf{N}\,y} \vdash \mathsf{E}\,\mathsf{s}y, \mathsf{O}\,\mathsf{s}y}\,(\mathrm{ER}_2)}{x = \mathsf{s}y, \color{blue}{\mathsf{N}\,y} \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(=\mathrm{L})
}{\color{blue}{\mathsf{N}\,x} \vdash \mathsf{E}\,x, \mathsf{O}\,x}\,(\text{Case N})
$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

# The Semantics of Inductive Predicate 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 $\Phi$ induce *characteristic operators* $\varphi_\Phi$ on predicate interpretations $X$ (functions from predicate formulas to sets of models):

$$\begin{aligned}\varphi_\Phi(X)(\mathrm{P}\,\vec{t}\theta) = \{m \mid \mathrm{P}_1\,\vec{t_1}, \ldots, \mathrm{P}_j\,\vec{t_j} \Rightarrow \mathrm{P}\,\vec{t} \in \Phi \,\wedge\, \forall x \in \mathrm{dom}(\theta) : m(x) = [\![\theta(x)]\!]_m \\ \wedge\, \forall 1 \le i \le j : m \in X(\mathrm{P}_i\,\vec{t_i}\theta)\}\end{aligned}$$

# The Semantics of Inductive Predicate 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 $\Phi$ induce *characteristic operators* $\varphi_\Phi$ on predicate interpretations $X$ (functions from predicate formulas to sets of models):

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

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

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

# The Semantics of Inductive Predicate 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 $\mathbf{\Phi}$ induce *characteristic operators* $\varphi_{\mathbf{\Phi}}$ on predicate interpretations $X$ (functions from predicate formulas to sets of models):

$$\begin{aligned}\varphi_{\mathbf{\Phi}}(X)(\mathsf{P}\,\vec{t}\theta) = \{m \mid {} & \mathsf{P}_1\,\vec{t_1}, \ldots, \mathsf{P}_j\,\vec{t_j} \Rightarrow \mathsf{P}\,\vec{t} \in \mathbf{\Phi} \wedge \forall x \in \mathrm{dom}(\theta) : m(x) = [\![\theta(x)]\!]_m \\ & \wedge \forall 1 \leq i \leq j : m \in X(\mathsf{P}_i\,\vec{t_i}\theta)\}\end{aligned}$$

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

$$\begin{aligned}\varphi_{\mathbf{\Phi}}(\varphi_{\mathbf{\Phi}}(X_{\perp}))(\mathsf{N}\,x) &= \{[x \mapsto 0], [x \mapsto \mathsf{s}0]\} \\ \varphi_{\mathbf{\Phi}}(\varphi_{\mathbf{\Phi}}(X_{\perp}))(\mathsf{E}\,x) &= \{[x \mapsto 0]\} \\ \varphi_{\mathbf{\Phi}}(\varphi_{\mathbf{\Phi}}(X_{\perp}))(\mathsf{O}\,x) &= \{[x \mapsto \mathsf{s}0]\}\end{aligned}$$

# The Semantics of Inductive Predicate 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 $\Phi$ induce *characteristic operators* $\varphi_\Phi$ on predicate interpretations $X$ (functions from predicate formulas to sets of models):

$$\begin{aligned}\varphi_\Phi(X)(\mathsf{P}\,\vec{t}\theta) = \{m \mid \mathsf{P}_1\,\vec{t_1}, \ldots, \mathsf{P}_j\,\vec{t_j} \Rightarrow \mathsf{P}\,\vec{t} \in \Phi \ \wedge\ \forall x \in \mathrm{dom}(\theta) : m(x) = [\![\theta(x)]\!]_m \\ \wedge\ \forall 1 \leq i \leq j : m \in X(\mathsf{P}_i\,\vec{t_i}\theta)\}\end{aligned}$$

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

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

$$\varphi_\Phi(\varphi_\Phi(\varphi_\Phi(X_\perp)))(\mathsf{E}\,x) = \{[x \mapsto 0], [x \mapsto \mathsf{ss}0]\}$$

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

# The Semantics of Inductive Predicate 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 $\Phi$ induce *characteristic operators* $\varphi_\Phi$ on predicate interpretations $X$ (functions from predicate formulas to sets of models):

$$\begin{aligned}\varphi_\Phi(X)(\mathsf{P}\,\vec{t}\theta) = \{m \mid \mathsf{P}_1\,\vec{t_1},\ldots,\mathsf{P}_j\,\vec{t_j} \Rightarrow \mathsf{P}\,\vec{t} \in \Phi \,\wedge\, \forall x \in \mathrm{dom}(\theta) : m(x) = [\![\theta(x)]\!]_m \\ \wedge\, \forall 1 \leq i \leq j : m \in X(\mathsf{P}_i\,\vec{t_i}\theta)\}\end{aligned}$$

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

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

# The Semantics of Inductive Predicate 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 $\mathbf{\Phi}$ induce *characteristic operators* $\varphi_{\mathbf{\Phi}}$ on predicate interpretations $X$ (functions from predicate formulas to sets of models):

$$\varphi_{\mathbf{\Phi}}(X)(\mathsf{P}\,\vec{t}\theta) = \{m \mid \mathsf{P}_1\,\vec{t_1},\ldots,\mathsf{P}_j\,\vec{t_j} \Rightarrow \mathsf{P}\,\vec{t} \in \mathbf{\Phi} \,\wedge\, \forall x \in \mathrm{dom}(\theta) : m(x) = [\![\theta(x)]\!]_m \\ \wedge\, \forall 1 \leq i \leq j : m \in X(\mathsf{P}_i\,\vec{t_i}\theta)\}$$

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

$$[\![\cdot]\!]^{\mathbf{\Phi}}_0 \sqsubseteq [\![\cdot]\!]^{\mathbf{\Phi}}_1 \sqsubseteq [\![\cdot]\!]^{\mathbf{\Phi}}_2 \sqsubseteq \ldots \sqsubseteq [\![\cdot]\!]^{\mathbf{\Phi}}_\alpha \sqsubseteq \ldots [\![\cdot]\!]^{\mathbf{\Phi}}$$

# Models as Realizers

- We say that a model $m \in [\![\mathsf{P}\,\vec{t}]\!]^{\Phi}$ realizes $\mathsf{P}\,\vec{t}$ (wrt. $\Phi$)
- We define a realization function $\Theta$:

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

# Models as Realizers

- We say that a model $m \in [\![\mathsf{P}\,\vec{t}]\!]^{\Phi}$ realizes $\mathsf{P}\,\vec{t}$ (wrt. $\Phi$)
- We define a realization function $\Theta$:

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

- The logical inference rules have the property that

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

for a counter-model $m$ of $\Gamma, \mathsf{P}\,\vec{t} \vdash \Delta$, there exists a counter-model $m'$ of some $\Sigma_i \vdash \Pi_i$ (local soundness) and if $\mathsf{P}\,\vec{t} \in \Sigma_i$ then $\Theta(\mathsf{P}\,\vec{t}, m') \leq \Theta(\mathsf{P}\,\vec{t}, m)$

# Models as Realizers

- We say that a model $m \in [\![\mathsf{P}\,\vec{t}]\!]^{\Phi}$ realizes $\mathsf{P}\,\vec{t}$ (wrt. $\Phi$)
- We define a realization function $\Theta$:

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

- The logical inference rules have the property that

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

  for a counter-model $m$ of $\Gamma, \mathsf{N}\,t \vdash \Delta$, there exists a counter-model $m'$ of either $\Gamma, t = 0 \vdash \Delta$ or $\Gamma, t = sx, \mathsf{N}\,x \vdash \Delta$ and if the latter then $\Theta(\mathsf{N}\,x, m') < \Theta(\mathsf{N}\,\vec{t}, m)$

# Soundness of Cyclic Proof

- 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

- 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

- 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

- 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 implies an infinite sequence of (counter) models

# Soundness of Cyclic Proof

- 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 implies an infinite sequence of (counter) models
    - These can be mapped to a non-increasing chain of ordinals using the realization function

# Soundness of Cyclic Proof

- 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 implies an infinite sequence of (counter) models
    - These can be mapped to a non-increasing chain of ordinals using the realization function
  - Global trace condition then implies this chain is infinitely descending

# Soundness of Cyclic Proof

- 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 implies an infinite sequence of (counter) models
    - These can be mapped to a non-increasing chain of ordinals using the realization function
  - Global trace condition then implies this chain is infinitely descending
    - But the ordinals are well-founded . . . contradiction

## Cyclic Proof vs Explicit Induction

- Explicit induction requires induction hypothesis $F$ up-front

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

# 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

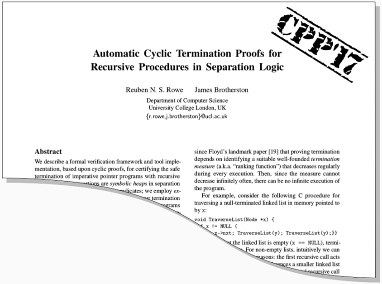
- Explicit induction requires induction hypothesis $F$ up-front

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

- Cyclic proof enables '*discovery*' of induction hypotheses
- Complex induction schemes naturally represented by nested and overlapping cycles
- Every sequent provable using the explict induction rule is also derivable using cyclic proof

```
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);

   }
}
```

```
struct ll { int data; ll *next; }
list(x) ⇔ (x = NULL ∧ emp) ∨ (x ↦ (d, l) ∗ 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) }
```


CPP'17

**Automatic Cyclic Termination Proofs for Recursive Procedures in Separation Logic**

Reuben N. S. Rowe  James Brotherston

Department of Computer Science
University College London, UK
{r.rowe,j.brotherston}@ucl.ac.uk

**Abstract**

We describe a formal verification framework and tool implementation, based upon cyclic proofs, for certifying the safe termination of imperative pointer programs with recursive … are *symbolic heaps* in separation … predicates; we employ *ex-* … termination … programs …

since Floyd's landmark paper [19] that proving termination depends on identifying a suitable well-founded *termination measure* (a.k.a. "ranking function") that decreases regularly during every execution. Then, since the measure cannot decrease infinitely often, there can be no infinite execution of the program.

For example, consider the following C procedure for traversing a null-terminated linked list in memory pointed to by x:

```
void TraverseList(Node *x) {
 … x != NULL {
 … x->nxt; TraverseList(y); TraverseList(y);}}
```

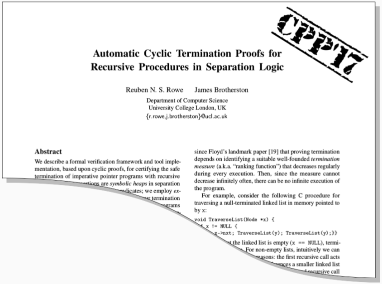
… the linked list is empty (x == NULL), termi- … For non-empty lists, intuitively we can … reasons: the first recursive call acts … a smaller linked list … recursive call

# Cyclic Proofs of Program Termination

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

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

```c
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) }
```


CPP'17

**Automatic Cyclic Termination Proofs for Recursive Procedures in Separation Logic**

Reuben N. S. Rowe James Brotherston

Department of Computer Science
University College London, UK
{r.rowe,j.brotherston}@ucl.ac.uk

**Abstract**

We describe a formal verification framework and tool implementation, based upon cyclic proofs, for certifying the safe termination of imperative pointer programs with recursive

since Floyd's landmark paper [19] that proving termination depends on identifying a suitable well-founded *termination measure* (a.k.a. "ranking function") that decreases regularly during every execution. Then, since the measure cannot decrease infinitely often, there can be no infinite execution of the program.

For example, consider the following C procedure for traversing a null-terminated linked list in memory pointed to by x:

# Cyclic Proofs of Program Termination

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

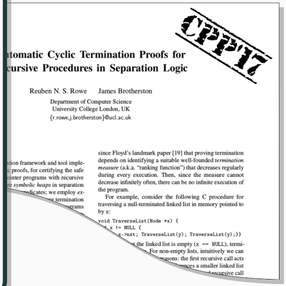
$\mathsf{list}(x) \Leftrightarrow (x = \mathtt{NULL} \wedge \mathsf{emp}) \vee (x \mapsto (d, l) * \mathsf{list}(l))$

`void rev(ll *x)` $\{\,\mathsf{list}_\alpha(\mathtt{x})\,\}$ `{ ... }` $\{\,\mathsf{list}_\alpha(\mathtt{x})\,\}$

`void shuffle(ll *x)` $\{\,\mathsf{list}_\alpha(\mathtt{x})\,\}$ `{`

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

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



tomatic Cyclic Termination Proofs for
cursive Procedures in Separation Logic

CPP'17

Reuben N. S. Rowe James Brotherston

Department of Computer Science
University College London, UK
{r.rowe,j.brotherston}@ucl.ac.uk

tion framework and tool imple-
c proofs, for certifying the safe
ointer programs with recursive
re *symbolic heaps* in separation

since Floyd's landmark paper [19] that proving termination depends on identifying a suitable well-founded *termination measure* (a.k.a. "ranking function") that decreases regularly during every execution. Then, since the measure cannot decrease infinitely often, there can be no infinite execution of the program.

For example, consider the following C procedure for traversing a null-terminated linked list in memory pointed to by x:

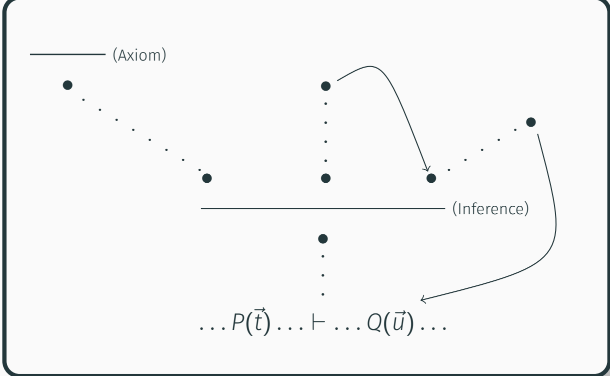
```
void TraverseList(Node *x) {
```

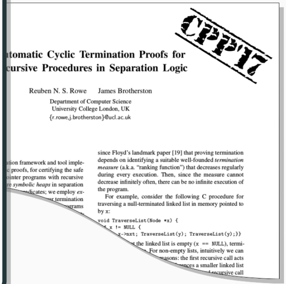
```
struct ll { int data; ll *next; }
```

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

`void rev(ll *x)` $\{\, \mathsf{list}_\alpha(\mathtt{x}) \,\}$ `{ ... }` $\{\, \mathsf{list}_\alpha(\mathtt{x}) \,\}$

`void shuffle(ll *x)` $\{\, \mathsf{list}_\alpha(\mathtt{x}) \,\}$ `{`





Automatic Cyclic Termination Proofs for Recursive Procedures in Separation Logic

Reuben N. S. Rowe    James Brotherston

Department of Computer Science
University College London, UK
{r.rowe,j.brotherston}@ucl.ac.uk

since Floyd's landmark paper [19] that proving termination depends on identifying a suitable well-founded *termination measure* (a.k.a. "ranking function") that decreases regularly during every execution. Then, since the measure cannot decrease infinitely often, there can be no infinite execution of the program.

For example, consider the following C procedure for traversing a null-terminated linked list in memory pointed to by x:

```
void TraverseList(Node *x) {
```

# Cyclic Proofs of Program Termination

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

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

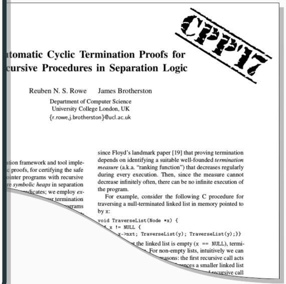
`void rev(ll *x)` $\{\, \mathsf{list}_\alpha(\mathtt{x}) \,\}$ `{ ... }` $\{\, \mathsf{list}_\alpha(\mathtt{x}) \,\}$

`void shuffle(ll *x)` $\{\, \mathsf{list}_\alpha(\mathtt{x}) \,\}$ `{`

(Axiom)

$$\forall \alpha : [\![ P\,\vec{t} \,]\!]_\alpha \subseteq [\![ Q\,\vec{u} \,]\!]_\alpha$$

$$\ldots P(\vec{t}) \ldots \vdash \ldots Q(\vec{u}) \ldots$$



...tomatic Cyclic Termination Proofs for ...cursive Procedures in Separation Logic

Reuben N. S. Rowe James Brotherston

Department of Computer Science
University College London, UK
{r.rowe,j.brotherston}@ucl.ac.uk

since Floyd's landmark paper [19] that proving termination depends on identifying a suitable well-founded *termination measure* (a.k.a. "ranking function") that decreases regularly during every execution. Then, since the measure cannot decrease infinitely often, there can be no infinite execution of the program.

For example, consider the following C procedure for traversing a null-terminated linked list in memory pointed to by x:

```
void TraverseList(Node *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)

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

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

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

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

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

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(E R$_1$)}
\UnaryInfC{$\vdash \mathrm{E}\,0,\ \mathrm{O}\,0$}
\RightLabel{(=L)}
\UnaryInfC{$x = 0 \vdash \mathrm{E}\,x,\ \mathrm{O}\,x$}
\AxiomC{$\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\RightLabel{(Subst)}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{E}\,y, \mathrm{O}\,y$}
\RightLabel{(PR)}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{O}\,y,\ \mathrm{E}\,y$}
\RightLabel{(O R)}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{O}\,y,\ \mathrm{O}\,\mathrm{s}y$}
\RightLabel{(E R$_2$)}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{E}\,\mathrm{s}y,\ \mathrm{O}\,\mathrm{s}y$}
\RightLabel{(=L)}
\UnaryInfC{$x = \mathrm{s}y, \mathrm{N}\,y \vdash \mathrm{E}\,x,\ \mathrm{O}\,x$}
\RightLabel{(Case N)}
\BinaryInfC{$\mathrm{N}\,x \vdash \mathrm{E}\,x,\ \mathrm{O}\,x$}
\end{prooftree}
$$

# Extracting Semantic Orderings: Example (1)

$$
\begin{aligned}
&\Rightarrow \mathrm{N}\,0 \\
\mathrm{N}\,x &\Rightarrow \mathrm{N}\,sx \\
&\Rightarrow \mathrm{E}\,0 \\
\mathrm{O}\,x &\Rightarrow \mathrm{E}\,sx \\
\mathrm{E}\,x &\Rightarrow \mathrm{O}\,sx
\end{aligned}
$$

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{$(\mathrm{E}\,\mathrm{R}_1)$}
\UnaryInfC{$\vdash \mathrm{E}\,0,\ \mathrm{O}\,0$}
\RightLabel{$(=\mathrm{L})$}
\UnaryInfC{$x = 0 \vdash \mathrm{E}\,x,\ \mathrm{O}\,x$}
\AxiomC{$\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\RightLabel{(Subst)}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{E}\,y, \mathrm{O}\,y$}
\RightLabel{(PR)}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{O}\,y,\ \mathrm{E}\,y$}
\RightLabel{$(\mathrm{O}\,\mathrm{R})$}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{O}\,y,\ \mathrm{O}\,sy$}
\RightLabel{$(\mathrm{E}\,\mathrm{R}_2)$}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{E}\,sy,\ \mathrm{O}\,sy$}
\RightLabel{$(=\mathrm{L})$}
\UnaryInfC{$x = sy, \mathrm{N}\,y \vdash \mathrm{E}\,x,\ \mathrm{O}\,x$}
\RightLabel{(Case N)}
\BinaryInfC{$\mathrm{N}\,x \vdash \mathrm{E}\,x,\ \mathrm{O}\,x$}
\end{prooftree}
$$

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

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

# Extracting Semantic Orderings: Example (1)

$$
\begin{aligned}
&\Rightarrow \mathsf{N}\,0 \\
\mathsf{N}\,x &\Rightarrow \mathsf{N}\,sx \\
&\Rightarrow \mathsf{E}\,0 \\
\mathsf{O}\,x &\Rightarrow \mathsf{E}\,sx \\
\mathsf{E}\,x &\Rightarrow \mathsf{O}\,sx
\end{aligned}
$$

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

## Extracting Semantic Orderings: Example (1)

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

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

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. $\forall m : m \in [\![\mathrm{N}\,0]\!]_1$)

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

# Extracting Semantic Orderings: Example (1)

$$\begin{aligned} &\Rightarrow \mathsf{N}\,0 \\ \mathsf{N}\,x &\Rightarrow \mathsf{N}\,sx \\ &\Rightarrow \mathsf{E}\,0 \\ \mathsf{O}\,x &\Rightarrow \mathsf{E}\,sx \\ \mathsf{E}\,x &\Rightarrow \mathsf{O}\,sx \end{aligned}$$

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

Not grounded!

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{$(\mathsf{E}\,\mathsf{R}_1)$}
\UnaryInfC{$\vdash \mathsf{E}\,0, \neg\,\mathsf{E}\,0$}
\AxiomC{}
\RightLabel{$(\mathsf{E}\,\mathsf{R}_1)$}
\UnaryInfC{$\vdash \mathsf{E}\,0$}
\RightLabel{$(\neg \mathsf{L})$}
\UnaryInfC{$\neg\,\mathsf{E}\,0 \vdash$}
\RightLabel{$(\mathsf{WR})$}
\UnaryInfC{$\neg\,\mathsf{E}\,0 \vdash \mathsf{O}\,0$}
\RightLabel{$(\mathsf{Cut})$}
\BinaryInfC{$\vdash \mathsf{E}\,0, \mathsf{O}\,0$}
\RightLabel{$(=\mathsf{L})$}
\UnaryInfC{$x = 0 \vdash \mathsf{E}\,x, \mathsf{O}\,x$}
\AxiomC{$\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$}
\RightLabel{$(\mathsf{Subst})$}
\UnaryInfC{$\mathsf{N}\,y \vdash \mathsf{E}\,y, \mathsf{O}\,y$}
\RightLabel{$(\mathsf{PR})$}
\UnaryInfC{$\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{E}\,y$}
\RightLabel{$(\mathsf{O}\,\mathsf{R})$}
\UnaryInfC{$\mathsf{N}\,y \vdash \mathsf{O}\,y, \mathsf{O}\,\mathsf{s}y$}
\RightLabel{$(\mathsf{E}\,\mathsf{R}_2)$}
\UnaryInfC{$\mathsf{N}\,y \vdash \mathsf{E}\,\mathsf{s}y, \mathsf{O}\,\mathsf{s}y$}
\RightLabel{$(=\mathsf{L})$}
\UnaryInfC{$x = \mathsf{s}y, \mathsf{N}\,y \vdash \mathsf{E}\,x, \mathsf{O}\,x$}
\RightLabel{$(\mathsf{Case}\ \mathsf{N})$}
\BinaryInfC{$\mathsf{N}\,x \vdash \mathsf{E}\,x, \mathsf{O}\,x$}
\end{prooftree}
$$

# Extracting Semantic Orderings: Example (1)

A negative trace

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash E\,0, \neg E\,0}(E\,R_1) \qquad \dfrac{\dfrac{\dfrac{}{\vdash E\,0}(E\,R_1)}{\neg E\,0 \vdash}(\neg L)}{\neg E\,0 \vdash O\,0}(WR)}{\vdash E\,0, O\,0}(Cut)}{x = 0 \vdash E\,x, O\,x}(=L) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{N\,x \vdash E\,x, O\,x}{N\,y \vdash E\,y, O\,y}(Subst)}{N\,y \vdash O\,y, E\,y}(PR)}{N\,y \vdash O\,y, O\,sy}(O\,R)}{N\,y \vdash E\,sy, O\,sy}(E\,R_2)}{x = sy, N\,y \vdash E\,x, O\,x}(=L)}{N\,x \vdash E\,x, O\,x}(Case\ N)$$

# Extracting Semantic Orderings: Example (1)

A positive trace

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(E $R_1$)}
\UnaryInfC{$\vdash \mathrm{E}\,0, \neg\, \mathrm{E}\,0$}
\AxiomC{}
\RightLabel{(E $R_1$)}
\UnaryInfC{$\vdash \mathrm{E}\,0$}
\RightLabel{($\neg$L)}
\UnaryInfC{$\neg\, \mathrm{E}\,0 \vdash$}
\RightLabel{(WR)}
\UnaryInfC{$\neg\, \mathrm{E}\,0 \vdash \mathrm{O}\,0$}
\RightLabel{(Cut)}
\BinaryInfC{$\vdash \mathrm{E}\,0, \mathrm{O}\,0$}
\RightLabel{($=$L)}
\UnaryInfC{$x = 0 \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\AxiomC{$\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\RightLabel{(Subst)}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{E}\,y, \mathrm{O}\,y$}
\RightLabel{(PR)}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{O}\,y, \mathrm{E}\,y$}
\RightLabel{(O R)}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{O}\,y, \mathrm{O}\,sy$}
\RightLabel{(E $R_2$)}
\UnaryInfC{$\mathrm{N}\,y \vdash \mathrm{E}\,sy, \mathrm{O}\,sy$}
\RightLabel{($=$L)}
\UnaryInfC{$x = sy, \mathrm{N}\,y \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\RightLabel{(Case N)}
\BinaryInfC{$\mathrm{N}\,x \vdash \mathrm{E}\,x, \mathrm{O}\,x$}
\end{prooftree}
$$

# Extracting Semantic Orderings: Example (2)

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

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

# Extracting Semantic Orderings: Example (2)

$$
\begin{aligned}
&\Rightarrow \mathsf{N}\,0\\
\mathsf{N}\,x &\Rightarrow \mathsf{N}\,\mathsf{s}x\\
&\Rightarrow \mathsf{E}\,0\\
\mathsf{O}\,x &\Rightarrow \mathsf{E}\,\mathsf{s}x\\
\mathsf{E}\,x &\Rightarrow \mathsf{O}\,\mathsf{s}x\\
\mathsf{N}\,\mathsf{ss}0 &\Rightarrow \mathsf{O}\,\mathsf{sss}0
\end{aligned}
$$

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

# Extracting Semantic Orderings: Example (2)

$$
\begin{aligned}
&\Rightarrow \mathsf{N}\,0 \\
\mathsf{N}\,x &\Rightarrow \mathsf{N}\,\mathsf{s}x \\
&\Rightarrow \mathsf{E}\,0 \\
\mathsf{O}\,x &\Rightarrow \mathsf{E}\,\mathsf{s}x \\
\mathsf{E}\,x &\Rightarrow \mathsf{O}\,\mathsf{s}x \\
\mathsf{N}\,\mathsf{ss}0 &\Rightarrow \mathsf{O}\,\mathsf{sss}0
\end{aligned}
$$

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{$(\mathsf{N}\,R_1)$}
\UnaryInfC{$\vdash \mathsf{N}\,0$}
\RightLabel{$(=L)$}
\UnaryInfC{$x = 0 \vdash \mathsf{N}\,x$}
\AxiomC{}
\RightLabel{(Ax)}
\UnaryInfC{$\mathsf{N}\,\mathsf{ss}0 \vdash \mathsf{N}\,\mathsf{ss}0$}
\RightLabel{$(\mathsf{N}\,R_2)$}
\UnaryInfC{$\mathsf{N}\,\mathsf{ss}0 \vdash \mathsf{N}\,\mathsf{sss}0$}
\RightLabel{$(=L)$}
\UnaryInfC{$y = \mathsf{sss}0, \mathsf{N}\,\mathsf{ss}0 \vdash \mathsf{N}\,y$}
\AxiomC{$\mathsf{E}\,x \vdash \mathsf{N}\,x$}
\RightLabel{(Subst)}
\UnaryInfC{$\mathsf{E}\,z \vdash \mathsf{N}\,z$}
\RightLabel{$(\mathsf{N}\,R_2)$}
\UnaryInfC{$\mathsf{E}\,z \vdash \mathsf{N}\,\mathsf{s}z$}
\RightLabel{$(=L)$}
\UnaryInfC{$y = \mathsf{s}z, \mathsf{E}\,z \vdash \mathsf{N}\,y$}
\RightLabel{(Case O)}
\BinaryInfC{$\mathsf{O}\,y \vdash \mathsf{N}\,y$}
\RightLabel{$(\mathsf{N}\,R_2)$}
\UnaryInfC{$\mathsf{O}\,y \vdash \mathsf{N}\,\mathsf{s}y$}
\RightLabel{$(=L)$}
\UnaryInfC{$x = \mathsf{s}y, \mathsf{O}\,y \vdash \mathsf{N}\,x$}
\RightLabel{(Case E)}
\BinaryInfC{$\mathsf{E}\,x \vdash \mathsf{N}\,x$}
\end{prooftree}
$$

$$\Rightarrow \mathsf{N}\,0$$
$$\mathsf{N}\,x \Rightarrow \mathsf{N}\,sx$$
$$\Rightarrow \mathsf{E}\,0$$
$$\mathsf{O}\,x \Rightarrow \mathsf{E}\,sx$$
$$\mathsf{E}\,x \Rightarrow \mathsf{O}\,sx$$
$$\mathsf{N}\,ss0 \Rightarrow \mathsf{O}\,sss0$$

$$\dfrac{\dfrac{}{x = 0 \vdash \mathsf{N}\,x}\,(=\!L) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{N}\,ss0 \vdash \mathsf{N}\,ss0}\,(Ax)}{\mathsf{N}\,ss0 \vdash \mathsf{N}\,sss0}\,(\mathsf{N}\,R_2)}{y = sss0, \mathsf{N}\,ss0 \vdash \mathsf{N}\,y}\,(=\!L) \qquad \dfrac{\dfrac{\dfrac{\mathsf{E}\,x \vdash \mathsf{N}\,x}{\mathsf{E}\,z \vdash \mathsf{N}\,z}\,(Subst)}{\mathsf{E}\,z \vdash \mathsf{N}\,sz}\,(\mathsf{N}\,R_2)}{y = sz, \mathsf{E}\,z \vdash \mathsf{N}\,y}\,(=\!L)}{\mathsf{O}\,y \vdash \mathsf{N}\,y}\,(\text{Case O})}{\mathsf{O}\,y \vdash \mathsf{N}\,sy}\,(\mathsf{N}\,R_2)}{x = sy, \mathsf{O}\,y \vdash \mathsf{N}\,x}\,(=\!L)}{\mathsf{E}\,x \vdash \mathsf{N}\,x}\,(\text{Case E})$$

This trace is partially maximal: the final predicate is the active formula of an axiom

# Extracting Semantic Orderings: Example (2)

$$
\begin{aligned}
&\Rightarrow \mathsf{N}\,0\\
\mathsf{N}\,x &\Rightarrow \mathsf{N}\,sx\\
&\Rightarrow \mathsf{E}\,0\\
\mathsf{O}\,x &\Rightarrow \mathsf{E}\,sx\\
\mathsf{E}\,x &\Rightarrow \mathsf{O}\,sx\\
\mathsf{N}\,ss0 &\Rightarrow \mathsf{O}\,sss0
\end{aligned}
$$

$$
\dfrac{
  \dfrac{\dfrac{}{\vdash \mathsf{N}\,0}\,(\mathsf{N}\,R_1)}{x = 0 \vdash \mathsf{N}\,x}\,(=\mathrm{L})
  \qquad
  \dfrac{\dfrac{\dfrac{
    \dfrac{\dfrac{\dfrac{}{\mathsf{N}\,ss0 \vdash \mathsf{N}\,ss0}\,(\mathrm{Ax})}{\mathsf{N}\,ss0 \vdash \mathsf{N}\,sss0}\,(\mathsf{N}\,R_2)}{y = sss0, \mathsf{N}\,ss0 \vdash \mathsf{N}\,y}\,(=\mathrm{L})
    \qquad
    \dfrac{\dfrac{\dfrac{\mathsf{E}\,x \vdash \mathsf{N}\,x}{\mathsf{E}\,z \vdash \mathsf{N}\,z}\,(\mathrm{Subst})}{\mathsf{E}\,z \vdash \mathsf{N}\,sz}\,(\mathsf{N}\,R_2)}{y = sz, \mathsf{E}\,z \vdash \mathsf{N}\,y}\,(=\mathrm{L})
  }{\mathsf{O}\,y \vdash \mathsf{N}\,y}\,(\mathrm{Case}\ \mathsf{O})}{\mathsf{O}\,y \vdash \mathsf{N}\,sy}\,(\mathsf{N}\,R_2)}{x = sy, \mathsf{O}\,y \vdash \mathsf{N}\,x}\,(=\mathrm{L})
}{\mathsf{E}\,x \vdash \mathsf{N}\,x}\,(\mathrm{Case}\ \mathsf{E})
$$

# 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 with the left-hand trace matching in the length and final predicate
- 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 $\alpha$*

### 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 $\alpha$*

## 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 $m \in [\![\mathrm{Q}\,\vec{y}]\!]_\alpha$

# 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 $\Sigma$ be an alphabet, and $(V, \oplus, \otimes)$ a semiring of weights. A weighted automaton $\mathscr{A}$ is a tuple $(Q, q_I, F, \gamma)$ 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 $\Sigma$ be an alphabet, and $(V, \oplus, \otimes)$ a semiring of weights. A weighted automaton $\mathscr{A}$ is a tuple $(Q, q_I, F, \gamma)$ 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 $\Sigma$ be an alphabet, and $(V, \oplus, \otimes)$ a semiring of weights. A weighted automaton $\mathscr{A}$ is a tuple $(Q, q_I, F, \gamma)$ 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 $\Sigma$ be an alphabet, and $(V, \oplus, \otimes)$ a semiring of weights. A weighted automaton $\mathscr{A}$ is a tuple $(Q, q_I, F, \gamma)$ 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: Existing Results

## 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 $\mathcal{L}_1(w) \leq \mathcal{L}_2(w)$

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

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

# Weighted Automata: Existing Results

### 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 $\mathcal{L}_1(w) \leq \mathcal{L}_2(w)$

### 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)

*Given two finite-valued weighted automata $\mathscr{A}$ and $\mathscr{B}$, it is decidable whether $\mathcal{L}_{\mathscr{A}} \leq \mathcal{L}_{\mathscr{B}}$*

# Weighted Automata from Cyclic Entailment Proofs

Given a cyclic entailment proof $\mathcal{P}$, we can construct two kinds of finite-valued sum automata, $\mathscr{A}_{\mathcal{P}}[n]$ $(n \in \mathbb{N})$ and $\mathscr{C}_{\mathcal{P}}$:

- The words accepted are paths in the proof from the root sequent
- Transitions corresponding to a case split have unit weight
    - The value of a path is the maximum number of unfoldings in the traces along the path

# Weighted Automata from Cyclic Entailment Proofs

Given a cyclic entailment proof $\mathcal{P}$, we can construct two kinds of finite-valued sum automata, $\mathscr{A}_{\mathcal{P}}[n]$ ($n \in \mathbb{N}$) and $\mathscr{C}_{\mathcal{P}}$:

- The words accepted are paths in the proof from the root sequent
- Transitions corresponding to a case split have unit weight
  - The value of a path is the maximum number of unfoldings in the traces along the path
- Each $\mathscr{A}_{\mathcal{P}}[n]$ considers only a subset of the paths in the proof

# Weighted Automata from Cyclic Entailment Proofs

Given a cyclic entailment proof $\mathcal{P}$, we can construct two kinds of finite-valued sum automata, $\mathscr{A}_{\mathcal{P}}[n]$ ($n \in \mathbb{N}$) and $\mathscr{C}_{\mathcal{P}}$:

- The words accepted are paths in the proof from the root sequent
- Transitions corresponding to a case split have unit weight
    - The value of a path is the maximum number of unfoldings in the traces along the path
- Each $\mathscr{A}_{\mathcal{P}}[n]$ considers only a subset of the paths in the proof
    - The complete automaton is not, in general, finite-valued

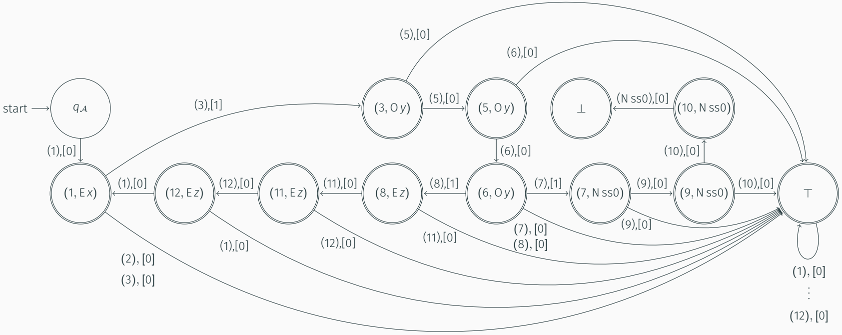
# Weighted Automata from Cyclic Entailment Proofs

Given a cyclic entailment proof $\mathcal{P}$, we can construct two kinds of finite-valued sum automata, $\mathscr{A}_{\mathcal{P}}[n]$ ($n \in \mathbb{N}$) and $\mathscr{C}_{\mathcal{P}}$:

- The words accepted are paths in the proof from the root sequent
- Transitions corresponding to a case split have unit weight
    - The value of a path is the maximum number of unfoldings in the traces along the path
- Each $\mathscr{A}_{\mathcal{P}}[n]$ considers only a subset of the paths in the proof
    - The complete automaton is not, in general, finite-valued
- $\mathscr{C}_{\mathcal{P}}$ is grounded when all final states correspond to ground predicate instances

# Weighted Automata from Cyclic Entailment Proofs

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



# An Equivalence between Realizability and Weighted Inclusion

The construction of the weighted automata admits the following result:

## 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 $\mathcal{P}$)*

# An Equivalence between Realizability and Weighted Inclusion

The construction of the weighted automata admits the following result:

**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 $[\![\mathsf{P}\,\vec{t}]\!]_\alpha \subseteq [\![\mathsf{Q}\,\vec{u}]\!]_\alpha$ from a proof of $\Gamma, \mathsf{P}\,\vec{t} \vdash \Sigma, \mathsf{Q}\,\vec{u}$

Then we can safely trace across an active cut formula

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

## Corollary: Bootstrapping Cyclic Entailment Systems

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

Then we can safely trace across an active cut formula

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

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

# Conclusions

- We have shown that information about inclusions between the semantics of inductive predicates can be extracted from cyclic proofs of entailments
- This information can be used to construct 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?