An Introduction to Cyclic Proofs

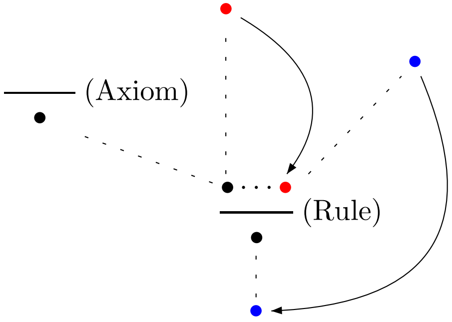
James Brotherston

University College London

PARIS workshop, FLoC, Oxford, 7th July 2018

Cyclic pre-proofs

A cyclic pre-proof is a derivation tree with a backlink from each open leaf ("bud") to an identical "companion":



Cyclic proof = pre-proof $\mathcal{P}$ + soundness condition $S(\mathcal{P})$.

An invalid pre-proof

$$\cfrac{\cfrac{\vdash \bot}{\vdash \bot, \bot}\;(\mathrm{Weak})}{\vdash \bot}\;(\mathrm{Contr})$$

An invalid pre-proof

$$\cfrac{\cfrac{\vdash \bot}{\vdash \bot, \bot}\ (\text{Weak})}{\vdash \bot}\ (\text{Contr})$$

- This is certainly a pre-proof, but obviously it cannot be accepted as valid!

An invalid pre-proof

$$\dfrac{\dfrac{\vdash \bot}{\vdash \bot, \bot}\ (\text{Weak})}{\vdash \bot}\ (\text{Contr})$$

- This is certainly a pre-proof, but obviously it cannot be accepted as valid!
- Here, we formed a cycle but failed to make any appreciable "progress".

The need for a soundness condition

- In any reasonable proof system the rules must be locally sound: if all premises of the rule are valid then so is its conclusion.

The need for a soundness condition

- In any reasonable proof system the rules must be locally sound: if all premises of the rule are valid then so is its conclusion.

- When proofs are finite trees, this guarantees that any provable judgement is valid: supposing not, then some axiom in the tree must be invalid, contradiction.

The need for a soundness condition

- In any reasonable proof system the rules must be locally sound: if all premises of the rule are valid then so is its conclusion.
- When proofs are finite trees, this guarantees that any provable judgement is valid: supposing not, then some axiom in the tree must be invalid, contradiction.
- However, when proofs are cyclic graphs, local soundness just says that if the root judgement is invalid then there is an infinite path of invalid judgements in the tree.

The need for a soundness condition

- In any reasonable proof system the rules must be locally sound: if all premises of the rule are valid then so is its conclusion.
- When proofs are finite trees, this guarantees that any provable judgement is valid: supposing not, then some axiom in the tree must be invalid, contradiction.
- However, when proofs are cyclic graphs, local soundness just says that if the root judgement is invalid then there is an infinite path of invalid judgements in the tree.
- A soundness condition for cyclic proofs must therefore rule out the existence of such paths.

Infinite descent

Because the ordinary methods now in the books were insufficient for demonstrating such difficult propositions, I finally found a totally unique route for arriving at them . . . which I called infinite descent . . .
If there were any integral right triangle that had an area equal to a square, there would be another triangle less than that one which would have the same property. . .
Now it is the case that, given a number, there are not infinitely many numbers less than that one in descending order . . . Whence one concludes that it is therefore impossible that there be any right triangle of which the area is a square. . .

Pierre de Fermat, *Relation des nouvelles decouvertes en la science des nombres*, letter to Pierre de Carcavi, 1659

Infinite descent example

Theorem
$\sqrt{2}$ *is not rational.*

Proof.

Infinite descent example

Theorem

$\sqrt{2}$ *is not rational.*

Proof.

Suppose for contradiction that $\sqrt{2} = x/y$ for $x, y \in \mathbb{N}$.

Infinite descent example

Theorem

$\sqrt{2}$ *is not rational.*

Proof.

Suppose for contradiction that $\sqrt{2} = x/y$ for $x, y \in \mathbb{N}$. Then $x^2 = 2y^2$. Consequently $x(x - y) = y(2y - x)$, so that:

$$\frac{2y - x}{x - y} = \frac{x}{y} = \sqrt{2}.$$

Infinite descent example

Theorem
$\sqrt{2}$ is not rational.

Proof.
Suppose for contradiction that $\sqrt{2} = x/y$ for $x, y \in \mathbb{N}$. Then $x^2 = 2y^2$. Consequently $x(x-y) = y(2y-x)$, so that:

$$\frac{2y-x}{x-y} = \frac{x}{y} = \sqrt{2}.$$

Define $x' = 2y - x$ and $y' = x - y$. Then $x'/y' = \sqrt{2}$. Now observe that $1 < x^2/y^2 < 4$, so $y < x < 2y$, and so $0 < y' < y$.

Infinite descent example

Theorem
$\sqrt{2}$ *is not rational.*

Proof.
Suppose for contradiction that $\sqrt{2} = x/y$ for $x, y \in \mathbb{N}$. Then $x^2 = 2y^2$. Consequently $x(x - y) = y(2y - x)$, so that:

$$\frac{2y - x}{x - y} = \frac{x}{y} = \sqrt{2}.$$

Define $x' = 2y - x$ and $y' = x - y$. Then $x'/y' = \sqrt{2}$. Now observe that $1 < x^2/y^2 < 4$, so $y < x < 2y$, and so $0 < y' < y$. But then we have $x', y' \in \mathbb{N}$ such that $\sqrt{2} = x'/y'$, and $y' < y$. This gives an infinite descent from y. $\square$

Example: μ-calculus properties of processes

"Clock" process Cl repeatedly ticks:

$$Cl =_{\mathrm{def}} tick.Cl$$

Example: μ-calculus properties of processes

"Clock" process Cl repeatedly ticks:

$$Cl =_{\mathrm{def}} tick.Cl$$

The μ-calculus formula $\nu X. \langle tick \rangle X$ means "the action 'tick' can be performed infinitely often".

Example: μ-calculus properties of processes

"Clock" process Cl repeatedly ticks:

$$Cl =_{\mathrm{def}} tick.Cl$$

The μ-calculus formula $\nu X.\,\langle tick\rangle X$ means "the action 'tick' can be performed infinitely often".

$$\dfrac{\dfrac{\dfrac{Cl \vdash \nu X.\,\langle tick\rangle X}{tick.Cl \vdash \langle tick\rangle \nu X.\,\langle tick\rangle X}\,(\langle tick\rangle)}{Cl \vdash \langle tick\rangle \nu X.\,\langle tick\rangle X}\,(\mathrm{Cl})}{Cl \vdash \nu X.\,\langle tick\rangle X}\,(\nu)$$

Example: μ-calculus properties of processes

"Clock" process Cl repeatedly ticks:

$$Cl =_{\text{def}} tick.Cl$$

The μ-calculus formula $\nu X.\, \langle tick \rangle X$ means "the action 'tick' can be performed infinitely often".

$$\dfrac{\dfrac{\dfrac{Cl \vdash \nu X.\, \langle tick \rangle X}{tick.Cl \vdash \langle tick \rangle \nu X.\, \langle tick \rangle X}\,(\langle tick \rangle)}{Cl \vdash \langle tick \rangle \nu X.\, \langle tick \rangle X}\,(\mathrm{Cl})}{Cl \vdash \nu X.\, \langle tick \rangle X}\,(\nu)$$

Soundness: two explanations

Suppose that $Cl \not\models \nu X. \langle tick \rangle X$. Then *every* judgement along the single infinite path in the proof is invalid.

Soundness: two explanations

Suppose that $Cl \not\models \nu X.\, \langle tick \rangle X$. Then *every* judgement along the single infinite path in the proof is invalid.

1. By supposition there are no infinite *tick* sequences from Cl. However, the infinite path *does* create such an infinite sequence, since $(\langle tick \rangle)$ is applied infinitely often.

Soundness: two explanations

Suppose that $Cl \not\models \nu X.\langle tick\rangle X$. Then *every* judgement along the single infinite path in the proof is invalid.

1. By supposition there are no infinite *tick* sequences from Cl. However, the infinite path *does* create such an infinite sequence, since $(\langle tick\rangle)$ is applied infinitely often.

2. There must be some ordinal-indexed overapproximation of the fixed point $\nu^\alpha X.\langle tick\rangle X$ of which Cl is not a member. Unfolding νX infinitely often (by (ν)) creates an infinite descending chain of such ordinals, from α — but these are well-founded.

Hoare logic

Imperative program verification is classically based on Hoare triples $\{P\}\,C\,\{Q\}$ where C is a program and P, Q are formulas.

Hoare logic

Imperative program verification is classically based on Hoare triples $\{P\}\, C\, \{Q\}$ where C is a program and P, Q are formulas.

We assume a programming language with an operational semantics given by $\langle C, \sigma \rangle \to \langle C', \sigma' \rangle$, where σ, σ' range over program states.

Hoare logic

Imperative program verification is classically based on Hoare triples $\{P\}\, C\, \{Q\}$ where C is a program and P, Q are formulas.

We assume a programming language with an operational semantics given by $\langle C, \sigma\rangle \to \langle C', \sigma'\rangle$, where σ, σ' range over program states. We also have a relation $\sigma \models P$ between states and formulas.

Hoare logic

Imperative program verification is classically based on Hoare triples $\{P\}\,C\,\{Q\}$ where C is a program and P, Q are formulas.

We assume a programming language with an operational semantics given by $\langle C, \sigma\rangle \rightarrow \langle C', \sigma'\rangle$, where σ, σ' range over program states. We also have a relation $\sigma \models P$ between states and formulas.

Then $\{P\}\,C\,\{Q\}$ is valid when:

$$\text{if } \sigma \models P \text{ and } \langle C, \sigma\rangle \rightarrow^* \langle \sigma'\rangle \text{ then } \sigma' \models Q \ .$$

Example: Hoare logic

Let C be the program

```
while i>0 {if * then i--;};
```

where `*` is a nondeterministic condition.

Example: Hoare logic

Let C be the program

```
while i>0 {if * then i--;};
```

where `*` is a nondeterministic condition. Let's show $\{i \geq 0\}\, C\, \{i = 0\}$.

Example: Hoare logic

Let C be the program

$$\texttt{while i>0 } \{\texttt{if * then i--;}\}\texttt{;}$$

where `*` is a nondeterministic condition. Let's show $\{i \geq 0\}\, C\, \{i = 0\}$.

$$\dfrac{\dfrac{\dfrac{}{i \geq 0, i \not> 0 \vdash i = 0}\,(\vdash)}{\{i \geq 0, i \not> 0\}\, \epsilon\, \{i = 0\}}\,(\epsilon) \qquad \dfrac{\dfrac{\{i \geq 0\}\, C\, \{i = 0\}}{\{i > 0\}\, \texttt{i--;}C\, \{i = 0\}}\,(\texttt{--}) \quad \dfrac{\{i \geq 0\}\, C\, \{i = 0\}}{\{i > 0\}\, C\, \{i = 0\}}\,(\vdash)}{\{i > 0\}\, \texttt{if * then i--;}C\, \{i = 0\}}\,(\texttt{if})}{\{i \geq 0\}\, C\, \{i = 0\}}\,(\texttt{while})$$

Example: Hoare logic

Let C be the program

```
while i>0 {if * then i--;};
```

where * is a nondeterministic condition. Let's show $\{i \geq 0\}\,C\,\{i = 0\}$.

$$
\dfrac{
\dfrac{
\dfrac{}{i \geq 0, i \not> 0 \vdash i = 0}\,(\vdash)
}{\{i \geq 0, i \not> 0\}\,\epsilon\,\{i = 0\}}\,(\epsilon)
\qquad
\dfrac{
\dfrac{\{i \geq 0\}\,C\,\{i = 0\}}{\{i > 0\}\,\texttt{i--;}C\,\{i = 0\}}\,(\texttt{--})
\quad
\dfrac{\{i \geq 0\}\,C\,\{i = 0\}}{\{i > 0\}\,C\,\{i = 0\}}\,(\vdash)
}{\{i > 0\}\,\texttt{if * then i--;}C\,\{i = 0\}}\,(\texttt{if})
}{\{i \geq 0\}\,C\,\{i = 0\}}\,(\texttt{while})
$$

Soundness explanation

Suppose that $\{i \geq 0\}\, C\, \{i = 0\}$ is invalid.

Soundness explanation

Suppose that $\{i \geq 0\}\, C\, \{i = 0\}$ is invalid.

I.e., there are states σ, σ' with $\sigma \models i \geq 0$ and $\langle C, \sigma \rangle \to^* \langle \sigma' \rangle$ but $\sigma' \not\models i = 0$.

Soundness explanation

Suppose that $\{i \geq 0\}\, C\, \{i = 0\}$ is invalid.

I.e., there are states σ, σ' with $\sigma \models i \geq 0$ and $\langle C, \sigma \rangle \rightarrow^* \langle \sigma' \rangle$ but $\sigma' \not\models i = 0$.

As usual, we get an infinite path of invalid triples through the proof, which must traverse one or both cycles infinitely often.

Soundness explanation

Suppose that $\{i \geq 0\}\, C\, \{i = 0\}$ is invalid.

I.e., there are states σ, σ' with $\sigma \models i \geq 0$ and $\langle C, \sigma \rangle \to^* \langle \sigma' \rangle$ but $\sigma' \not\models i = 0$.

As usual, we get an infinite path of invalid triples through the proof, which must traverse one or both cycles infinitely often.

But program commands are symbolically executed infinitely often along this path. Thus the assumed execution from $\langle C, \sigma \rangle$ is in fact infinite: contradiction.

A quick aside

One can draw a rough analogy between cyclic Hoare proofs and abstract interpretation, also used to verify imperative code:

A quick aside

One can draw a rough analogy between cyclic Hoare proofs and abstract interpretation, also used to verify imperative code:

Abstract interpretation		Cyclic Hoare proofs
abstract domain	$\sim$	formula language
symbolic execution	$\sim$	symbolic execution
widening	$\sim$	generalisation
narrowing	$\sim$	instantiation
invariance	$\sim$	proof cycle

Inductive definitions in first-order logic

Consider these inductive definitions of predicates N, E, O:

$$\begin{array}{rcl} & \Rightarrow & N0 \\ Nx & \Rightarrow & Nsx \end{array} \qquad \begin{array}{rcl} & \Rightarrow & E0 \\ Ex & \Rightarrow & Osx \\ Ox & \Rightarrow & Esx \end{array}$$

Inductive definitions in first-order logic

Consider these inductive definitions of predicates N, E, O:

$$\begin{array}{rcl} & \Rightarrow & N0 \\ Nx & \Rightarrow & Nsx \end{array} \qquad \begin{array}{rcl} & \Rightarrow & E0 \\ Ex & \Rightarrow & Osx \\ Ox & \Rightarrow & Esx \end{array}$$

These definitions generate case-split rules, e.g., for N:

$$\frac{\Gamma, t = 0 \vdash \Delta \qquad \Gamma, t = sx, Nx \vdash \Delta}{\Gamma, Nt \vdash \Delta} \text{ (Case } N\text{)}$$

(where x is fresh).
Note that Nx in the right-hand premise is obtained by *unfolding* Nt in the conclusion.

Example, inductive definitions

We'll prove that every natural number is either even or odd, i.e. $Nx \vdash Ex \vee Ox$.

Example, inductive definitions

We'll prove that every natural number is either even or odd, i.e. $Nx \vdash Ex \vee Ox$.

$$
\dfrac{
  \dfrac{
    \dfrac{}{\vdash E0, O0}\,(E)
  }{x = 0 \vdash Ex, Ox}\,(=)
  \qquad
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{Nx \vdash Ox, Ex}{Ny \vdash Oy, Ey}\,(\text{Subst})
        }{Ny \vdash Oy, Osy}\,(O)
      }{Ny \vdash Esy, Osy}\,(E)
    }{x = sy, Ny \vdash Ex, Ox}\,(=)
  }{}
}{
  \dfrac{Nx \vdash Ex, Ox}{Nx \vdash Ex \vee Ox}\,(\vee)
}\,(\text{Case } N)
$$

Example, inductive definitions

We'll prove that every natural number is either even or odd, i.e. $Nx \vdash Ex \vee Ox$.

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(E)}
\UnaryInfC{$\vdash E0, O0$}
\RightLabel{$(=)$}
\UnaryInfC{$x = 0 \vdash Ex, Ox$}
\AxiomC{$Nx \vdash Ox, Ex$}
\RightLabel{(Subst)}
\UnaryInfC{$Ny \vdash Oy, Ey$}
\RightLabel{(O)}
\UnaryInfC{$Ny \vdash Oy, Osy$}
\RightLabel{(E)}
\UnaryInfC{$Ny \vdash Esy, Osy$}
\RightLabel{$(=)$}
\UnaryInfC{$x = sy, Ny \vdash Ex, Ox$}
\RightLabel{(Case N)}
\BinaryInfC{$Nx \vdash Ex, Ox$}
\RightLabel{$(\vee)$}
\UnaryInfC{$Nx \vdash Ex \vee Ox$}
\end{prooftree}
$$

Note that here we examine formulas on the left of the turnstile!

Explanation of soundness

Suppose that $Nx \vdash Ex \vee Ox$ is invalid, meaning that $M \models_\rho Nx$ (for some structure M and valuation ρ) but $M \not\models_\rho Ex \vee Ox$.

As usual, we have that every sequent on the infinite path is invalid.

Explanation of soundness

Suppose that $Nx \vdash Ex \vee Ox$ is invalid, meaning that $M \models_\rho Nx$ (for some structure M and valuation ρ) but $M \not\models_\rho Ex \vee Ox$.

As usual, we have that every sequent on the infinite path is invalid. We can either notice:

1. that $[\![N]\!]_M$ is a well-founded set and we have an infinite descent in these "numerals", from $\rho(x)$, because of the infinite unfolding of Nx; or

Explanation of soundness

Suppose that $Nx \vdash Ex \vee Ox$ is invalid, meaning that $M \models_\rho Nx$ (for some structure M and valuation ρ) but $M \not\models_\rho Ex \vee Ox$.

As usual, we have that every sequent on the infinite path is invalid. We can either notice:

1. that $[\![N]\!]_M$ is a well-founded set and we have an infinite descent in these "numerals", from $\rho(x)$, because of the infinite unfolding of Nx; or
2. that if $\rho(x) \in [\![N]\!]_M$ that it is a member of some underapproximation $[\![N]\!]^\alpha_M$, and we have an infinite descent in these approximant ordinals, again because of the infinite unfolding of N.

Example (2), inductive definitions

Here's a proof of the converse statement, $Ex \vee Ox \vdash Nx$.

Example (2), inductive definitions

Here's a proof of the converse statement, $Ex \vee Ox \vdash Nx$.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{\vdash N0}\,(N)
    }{x = 0 \vdash Nx}\,(=)
    \qquad
    \dfrac{
      \dfrac{
        \dfrac{Ox \vdash Nx}{Oy \vdash Ny}\,(\text{Subst})
      }{Oy \vdash Nsy}\,(N)
    }{x = sy, Oy \vdash Nx}\,(=)
  }{Ex \vdash Nx}\,(\text{Case } E)
  \qquad
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{Ex \vdash Nx}{Ey \vdash Ny}\,(\text{Subst})
      }{Ey \vdash Nsy}\,(N)
    }{x = sy, Ey \vdash Nx}\,(=)
  }{Ox \vdash Nx}\,(\text{Case } O)
}{Ex \vee Ox \vdash Nx}\,(\vee)
$$

Example (2), inductive definitions

Here's a proof of the converse statement, $Ex \vee Ox \vdash Nx$.

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(N)}
\UnaryInfC{$\vdash N0$}
\RightLabel{$(=)$}
\UnaryInfC{$x = 0 \vdash Nx$}
\AxiomC{$Ox \vdash Nx$}
\RightLabel{(Subst)}
\UnaryInfC{$Oy \vdash Ny$}
\RightLabel{(N)}
\UnaryInfC{$Oy \vdash Nsy$}
\RightLabel{$(=)$}
\UnaryInfC{$x = sy, Oy \vdash Nx$}
\RightLabel{(Case E)}
\BinaryInfC{$Ex \vdash Nx$}
\AxiomC{$Ex \vdash Nx$}
\RightLabel{(Subst)}
\UnaryInfC{$Ey \vdash Ny$}
\RightLabel{(N)}
\UnaryInfC{$Ey \vdash Nsy$}
\RightLabel{$(=)$}
\UnaryInfC{$x = sy, Ey \vdash Nx$}
\RightLabel{(Case O)}
\UnaryInfC{$Ox \vdash Nx$}
\RightLabel{$(\vee)$}
\BinaryInfC{$Ex \vee Ox \vdash Nx$}
\end{prooftree}
$$

Soundness justification is as before.

Remark on soundness

Our soundness justifications often rely on reasoning of the form "*this* formula instance in the proof is a fixed point unfolding of *that* one".

Some proof rules can complicate this reasoning.

Remark on soundness

Our soundness justifications often rely on reasoning of the form "*this* formula instance in the proof is a fixed point unfolding of *that* one".

Some proof rules can complicate this reasoning. Some instances:

$$\frac{A \vdash B}{A, Px \vdash B}\,(\text{Weak}) \qquad \frac{Py \vdash B}{Px, x = y \vdash B}\,(=) \qquad \frac{Px \vdash F \quad F \vdash B}{Px \vdash B}\,(\text{Cut})$$

$$\frac{Px \vdash Fx}{Pz \vdash Fz}\,(\text{Subst}) \qquad \frac{x = sy, Ey \vdash B}{Ox \vdash B}\,(\text{Case } O)$$

Remark on soundness

Our soundness justifications often rely on reasoning of the form "*this* formula instance in the proof is a fixed point unfolding of *that* one".

Some proof rules can complicate this reasoning. Some instances:

$$\frac{A \vdash B}{A, Px \vdash B}\,(\text{Weak}) \qquad \frac{Py \vdash B}{Px, x = y \vdash B}\,(=) \qquad \frac{Px \vdash F \quad F \vdash B}{Px \vdash B}\,(\text{Cut})$$

$$\frac{Px \vdash Fx}{Pz \vdash Fz}\,(\text{Subst}) \qquad \frac{x = sy, Ey \vdash B}{Ox \vdash B}\,(\text{Case } O)$$

Dealing with this is essentially a matter of book-keeping. And it might not be necessary if there are no tricky rules.

Traces

- In a rule instance, a pair of “related” formula occurrences (or other proof annotations) (A, B) in the conclusion and some premise respectively is called a trace pair.

Traces

- In a rule instance, a pair of "related" formula occurrences (or other proof annotations) (A, B) in the conclusion and some premise respectively is called a trace pair.
- A trace pair is called progressing if B is actually obtained by unfolding A (and not just "the same" formula).

Traces

- In a rule instance, a pair of “related” formula occurrences (or other proof annotations) (A, B) in the conclusion and some premise respectively is called a trace pair.
- A trace pair is called progressing if B is actually obtained by unfolding A (and not just “the same” formula).
- A trace along a path in a pre-proof is obtained by chaining trace pairs together in the obvious way.

Traces

- In a rule instance, a pair of "related" formula occurrences (or other proof annotations) (A, B) in the conclusion and some premise respectively is called a trace pair.
- A trace pair is called progressing if B is actually obtained by unfolding A (and not just "the same" formula).
- A trace along a path in a pre-proof is obtained by chaining trace pairs together in the obvious way.
- A trace is infinitely progressing if it contains infinitely many progressing trace pairs.

A general soundness condition

Given some appropriate[1] notion of "trace pairs" for a cyclic proof system, one can then state a general soundness condition:

[1]This is a formalisable concept.

A general soundness condition

Given some appropriate[1] notion of "trace pairs" for a cyclic proof system, one can then state a general soundness condition:

> *A pre-proof is a cyclic proof if, for every infinite path in the proof, there is an infinitely progressing trace along some tail of the path.*

[1]This is a formalisable concept.

A general soundness condition

Given some appropriate[1] notion of "trace pairs" for a cyclic proof system, one can then state a general soundness condition:

> *A pre-proof is a cyclic proof if, for every infinite path in the proof, there is an infinitely progressing trace along some tail of the path.*

Virtually all the cyclic systems I know use a condition of this form, or which can be rewritten as such.

[1]This is a formalisable concept.

Two relevant facts

Given the soundness condition of the previous form,

1. Cyclic proofs then become sound.

Two relevant facts

Given the soundness condition of the previous form,

1. Cyclic proofs then become sound. If not, then there is an infinite path of invalid judgements in the proof. There is an infinitely progressing trace following this path. This can be used to realise an infinite descending chain of values in a well-founded set: contradiction.

Two relevant facts

Given the soundness condition of the previous form,

1. Cyclic proofs then become sound. If not, then there is an infinite path of invalid judgements in the proof. There is an infinitely progressing trace following this path. This can be used to realise an infinite descending chain of values in a well-founded set: contradiction.

2. It is decidable whether a pre-proof $\mathcal{P}$ is a cyclic proof or not.

Two relevant facts

Given the soundness condition of the previous form,

1. Cyclic proofs then become sound. If not, then there is an infinite path of invalid judgements in the proof. There is an infinitely progressing trace following this path. This can be used to realise an infinite descending chain of values in a well-founded set: contradiction.

2. It is decidable whether a pre-proof $\mathcal{P}$ is a cyclic proof or not. Build two Büchi automata: B_1 accepting all infinite paths in $\mathcal{P}$; and B_2 accepting all paths with an infinitely progressing trace on some tail. The soundness condition amounts to checking $\mathcal{L}(B_1) \subseteq \mathcal{L}(B_2)$.

Some logics with cyclic proof systems

- μ-calculus (modal, first-order, process verification)
- temporal logic (CTL, LTL,...)
- first-order logic with ind. defns
- separation logic with ind. defns
- Hoare logic and variants (e.g. termination)
- linear logic with fixed points
- modal logic (of certain kinds)
- Kleene algebra
- combinations of the above

This is by no means a complete list!