3. Coinductive Predicates

Example of Coinductive Definition

Informal example 3 (the $subll$ predicate) revisited

Given a set A, let LazyList(A) be the set of "lazy lists" (finite or infinite lists) with elements in A – they have the form $[a_1, a_2, \ldots, a_n]$ or $[a_1, a_2, \ldots]$. We write $a\#as$ for the lazy list obtained by consing a to as.

We wish to define the sublist relation, $subll$, on lazy lists.

The relation $subl$ on (finite) lists is defined inductively by the rules:

$$\frac{\cdot}{subl\ []\ as}\ \text{(Nil)} \qquad\qquad \frac{subl\ as\ as'}{subl\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subl\ as\ as'}{subl\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

The inductive interpretation means: smallest relation closed under the rules (Nil), (ConsR) and (Cons).

Informal example 3 (the $subll$ predicate) revisited

Given a set A, let LazyList(A) be the set of "lazy lists" (finite or infinite lists) with elements in A – they have the form $[a_1, a_2, \ldots, a_n]$ or $[a_1, a_2, \ldots]$. We write $a\#as$ for the lazy list obtained by consing a to as.

We wish to define the sublist relation, $subll$, on lazy lists.

The relation $subl$ on (finite) lists is defined inductively by the rules:

$$\frac{\cdot}{subl\ []\ as}\ \text{(Nil)} \qquad\qquad \frac{subl\ as\ as'}{subl\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subl\ as\ as'}{subl\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

The inductive interpretation means: smallest relation closed under the rules (Nil), (ConsR) and (Cons). We've seen that this does not work for lazy lists: No infinite list would be a sublist of any list.

Informal example 3 (the $subll$ predicate) revisited

Given a set A, let LazyList(A) be the set of "lazy lists" (finite or infinite lists) with elements in A – they have the form $[a_1, a_2, \ldots, a_n]$ or $[a_1, a_2, \ldots]$. We write $a\#as$ for the lazy list obtained by consing a to as.

We wish to define the sublist relation, $subll$, on lazy lists.

The relation $subl$ on (finite) lists is defined inductively by the rules:

$$\frac{\cdot}{subl\ []\ as}\ \text{(Nil)} \qquad\qquad \frac{subl\ as\ as'}{subl\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subl\ as\ as'}{subl\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

The inductive interpretation means: smallest relation closed under the rules (Nil), (ConsR) and (Cons). We've seen that this does not work for lazy lists: No infinite list would be a sublist of any list.

Should we rather go for the greatest relation closed under these rules?

Informal example 3 (the $subll$ predicate) revisited

Given a set A, let LazyList(A) be the set of "lazy lists" (finite or infinite lists) with elements in A – they have the form $[a_1, a_2, \ldots, a_n]$ or $[a_1, a_2, \ldots]$. We write $a\#as$ for the lazy list obtained by consing a to as.

We wish to define the sublist relation, $subll$, on lazy lists.

The relation $subl$ on (finite) lists is defined inductively by the rules:

$$\frac{\cdot}{subl\ []\ as}\ \text{(Nil)} \qquad\qquad \frac{subl\ as\ as'}{subl\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subl\ as\ as'}{subl\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

The inductive interpretation means: smallest relation closed under the rules (Nil), (ConsR) and (Cons). We've seen that this does not work for lazy lists: No infinite list would be a sublist of any list.

Should we rather go for the greatest relation closed under these rules? No! This would give us the total relation $\lambda as, as'.\ \top$.

Informal example 3 (the $subll$ predicate) revisited

Given a set A, let LazyList(A) be the set of "lazy lists" (finite or infinite lists) with elements in A – they have the form $[a_1, a_2, \ldots, a_n]$ or $[a_1, a_2, \ldots]$. We write $a\#as$ for the lazy list obtained by consing a to as.

We wish to define the sublist relation, $subll$, on lazy lists.

The relation $subl$ on (finite) lists is defined inductively by the rules:

$$\frac{\cdot}{subl\ []\ as}\ \text{(Nil)} \qquad\qquad \frac{subl\ as\ as'}{subl\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subl\ as\ as'}{subl\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

The inductive interpretation means: smallest relation closed under the rules (Nil), (ConsR) and (Cons). We've seen that this does not work for lazy lists: No infinite list would be a sublist of any list.

Should we rather go for the greatest relation closed under these rules?
No! This would give us the total relation $\lambda as, as'. \top$.
Let's take it easy, starting with selecting the properties that we want...

Desired properties for the predicate subll

Say $A = \mathbb{N}$.

For finite lists, subll should behave just like subl, e.g.,

- $\mathit{subll}\ [1, 3, 4]\ [1, 2, 3, 4]$
- $\mathit{subll}\ [1, 2]\ [1, 2, 3, 4]$
- $\mathit{subll}\ [1, 3]\ [1, 2, 3, 4]$

Also, e.g.,

- subll zeros zeros, in fact $\mathit{subll}\ \mathit{as}\ \mathit{as}$ for any as
- $\mathit{subll}\ [0, 2, 4, 6, \ldots]\ [0, 1, 2, 3, \ldots]$

Desired properties for the predicate $subll$

Say $A = \mathbb{N}$.

For finite lists, $subll$ should behave just like $subl$, e.g.,

- $subll\ [1,3,4]\ [1,2,3,4]$
- $subll\ [1,2]\ [1,2,3,4]$
- $subll\ [1,3]\ [1,2,3,4]$

Also, e.g.,

- $subll$ zeros zeros, in fact $subll\ as\ as$ for any as
- $subll\ [0,2,4,6,\ldots]\ [0,1,2,3,\ldots]$

$subll\ as\ as'$ should hold if and only if:
assuming as' has the form $[a'_i]_{i<length\ as'}$ (with $length\ as' \in \mathbb{N} \cup \{\infty\}$)
there exists $[j_p]_{p<length\ as}$ such that $\forall p.\ p+1 < length\ as \longrightarrow j_p < j_{p+1}$
and $as = [a'_{j_p}]_{p<length\ as}$.

Desired properties for the predicate $subll$

$$\frac{\cdot}{subll\ []\ as}\ (\text{Nil}) \qquad \frac{subll\ as\ as'}{subll\ as\ (a\#as')}\ (\text{ConsR})$$

$$\frac{subll\ as\ as'}{subll\ (a\#as)\ (a\#as')}\ (\text{Cons})$$

Desired properties for the predicate $subll$

$$\frac{\cdot}{subll\ []\ as}\ \text{(Nil)} \qquad \frac{subll\ as\ as'}{subll\ as\ (a\#as')}\ \text{(ConsR)}$$

✓

$$\frac{subll\ as\ as'}{subll\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

Desired properties for the predicate $sublll$

$$\frac{\cdot}{sublll\ []\ as}\ \text{(Nil)} \qquad \frac{sublll\ as\ as'}{sublll\ as\ (a\#as')}\ \text{(ConsR)}$$

✓

$$\frac{sublll\ as\ as'}{sublll\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\frac{\begin{array}{c} sublll\ bs\ bs' \qquad\qquad \forall as.\ bs = []\wedge bs' = as \longrightarrow P \\ \forall as, as', a.\ bs = as \wedge bs' = a\#as' \wedge sublll\ as\ as' \longrightarrow P \\ \forall as, as', a.\ bs = a\#as \wedge bs' = a\#as' \wedge sublll\ as\ as' \longrightarrow P \end{array}}{P}\ \text{(Cases)}$$

Desired properties for the predicate $subll$

$$\frac{\cdot}{subll\ []\ as}\ \text{(Nil)} \qquad \frac{subll\ as\ as'}{subll\ as\ (a\#as')}\ \text{(ConsR)}$$

✓

$$\frac{subll\ as\ as'}{subll\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\frac{\begin{array}{c} subll\ bs\ bs' \qquad \forall as.\ bs = [] \wedge bs' = as \longrightarrow P \\ \forall as, as', a.\ bs = as \wedge bs' = a\#as' \wedge subll\ as\ as' \longrightarrow P \\ \forall as, as', a.\ bs = a\#as \wedge bs' = a\#as' \wedge subll\ as\ as' \longrightarrow P \end{array}}{P}\ \text{(Cases)}$$

Desired properties for the predicate $sublll$

$$\frac{\cdot}{sublll\ []\ as}\ \text{(Nil)} \qquad \frac{sublll\ as\ as'}{sublll\ as\ (a\#as')}\ \text{(ConsR)}$$

✓

$$\frac{sublll\ as\ as'}{sublll\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\frac{\begin{array}{c} sublll\ bs\ bs' \qquad \forall as.\ bs = [] \wedge bs' = as \longrightarrow P \\ \forall as, as', a.\ bs = as \wedge bs' = a\#as' \wedge sublll\ as\ as' \longrightarrow P \\ \forall as, as', a.\ bs = a\#as \wedge bs' = a\#as' \wedge sublll\ as\ as' \longrightarrow P \end{array}}{P}\ \text{(Cases)}$$

or, equivalently...

$$\begin{array}{rl} sublll\ bs\ bs' \longrightarrow & \exists as.\ bs = [] \wedge bs' = as \\ & \vee \\ & \exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge sublll\ as\ as' \\ & \vee \\ & \exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge sublll\ as\ as' \end{array}$$

Desired properties for the predicate subll

$$\frac{\cdot}{\mathit{subll}\ []\ as}\ \text{(Nil)} \qquad \frac{\mathit{subll}\ as\ as'}{\mathit{subll}\ as\ (a\#as')}\ \text{(ConsR)}$$

✓

$$\frac{\mathit{subll}\ as\ as'}{\mathit{subll}\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\frac{\begin{array}{c}\mathit{subll}\ bs\ bs' \qquad \forall as.\ bs = [] \wedge bs' = as \longrightarrow P \\ \forall as, as', a.\ bs = as \wedge bs' = a\#as' \wedge \mathit{subll}\ as\ as' \longrightarrow P \\ \forall as, as', a.\ bs = a\#as \wedge bs' = a\#as' \wedge \mathit{subll}\ as\ as' \longrightarrow P\end{array}}{P}\ \text{(Cases)}$$ ✓

or, equivalently...

$$\begin{array}{ll}\mathit{subll}\ bs\ bs' \longrightarrow & \exists as.\ bs = [] \wedge bs' = as \\ & \vee \\ & \exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge \mathit{subll}\ as\ as' \\ & \vee \\ & \exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge \mathit{subll}\ as\ as'\end{array}$$ ✓

Desired properties for the predicate $subll$

How about induction?

$$\frac{\begin{array}{c} subll\ bs\ bs' \qquad\qquad \forall as.\ P\ []\ as \\ \forall as, as', a.\ subll\ as\ as' \ \wedge\ P\ as\ as' \longrightarrow P\ as\ (a\#as') \\ \forall as, as', a.\ subll\ as\ as' \ \wedge\ P\ as\ as' \longrightarrow P\ (a\#as)\ (a\#as') \end{array}}{P\ bs\ bs'}\ \text{(Induct)}$$

Desired properties for the predicate $sublll$

How about induction?

$$\frac{\begin{array}{c} sublll\ bs\ bs' \qquad\qquad \forall as.\ P\ []\ as \\ \forall\, as, as', a.\ sublll\ as\ as' \ \wedge\ P\ as\ as' \longrightarrow P\ as\ (a\#as') \\ \forall\, as, as', a.\ sublll\ as\ as' \ \wedge\ P\ as\ as' \longrightarrow P\ (a\#as)\ (a\#as') \end{array}}{P\ bs\ bs'}\ \text{(Induct)}$$

It would allow us to prove, e.g., $sublll\ as\ as'$ implies as is finite.

Desired properties for the predicate $sublls$

How about induction?

$$\frac{\begin{array}{c} sublls\ bs\ bs' \qquad\qquad \forall as.\ P\ []\ as \\ \forall as, as', a.\ sublls\ as\ as' \ \wedge\ P\ as\ as' \longrightarrow P\ as\ (a\#as') \\ \forall as, as', a.\ sublls\ as\ as' \ \wedge\ P\ as\ as' \longrightarrow P\ (a\#as)\ (a\#as') \end{array}}{P\ bs\ bs'}\ \text{(Induct)}$$

It would allow us to prove, e.g., $sublls\ as\ as'$ implies as is finite.

This would imply, e.g.,

- $\neg\ sublls$ zeros zeros
- $\neg\ sublls\ [0, 2, 4, 6, \ldots]\ [0, 1, 2, 3, \ldots]$

Desired properties for the predicate $sublll$

How about induction?

$$\frac{\begin{array}{c} sublll\ bs\ bs' \qquad\qquad \forall as.\ P\ []\ as \\ \forall as, as', a.\ sublll\ as\ as' \wedge P\ as\ as' \longrightarrow P\ as\ (a\#as') \\ \forall as, as', a.\ sublll\ as\ as' \wedge P\ as\ as' \longrightarrow P\ (a\#as)\ (a\#as') \end{array}}{P\ bs\ bs'}\ \text{(Induct)} \qquad \textbf{x}$$

It would allow us to prove, e.g., $sublll\ as\ as'$ implies as is finite.

This would imply, e.g.,

- $\neg\ sublll$ zeros zeros
- $\neg\ sublll\ [0, 2, 4, 6, \ldots]\ [0, 1, 2, 3, \ldots]$

Desired properties for the predicate subll

$$\frac{\cdot}{\mathit{subll}\ []\ as}\ \text{(Nil)} \qquad \frac{\mathit{subll}\ as\ as'}{\mathit{subll}\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{\mathit{subll}\ as\ as'}{\mathit{subll}\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\begin{aligned}
\mathit{subll}\ bs\ bs' \longrightarrow\ & \exists as.\ bs = [] \wedge bs' = as \\
& \vee \\
& \exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge \mathit{subll}\ as\ as' \\
& \vee \\
& \exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge \mathit{subll}\ as\ as'
\end{aligned}$$

Desired properties for the predicate $subll$

$$\frac{\cdot}{subll\ []\ as}\ \text{(Nil)} \qquad \frac{subll\ as\ as'}{subll\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subll\ as\ as'}{subll\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\begin{aligned} subll\ bs\ bs' \longrightarrow\ & \exists as.\ bs = [] \wedge bs' = as \\ & \vee \\ & \exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge subll\ as\ as' \\ & \vee \\ & \exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge subll\ as\ as' \end{aligned}$$

How can we prove $subll\ [0, 2, 4, 6, \ldots]\ [0, 1, 2, 3, 4, 5, \ldots]$?

Desired properties for the predicate $subll$

$$\frac{\cdot}{subll\ []\ as}\ \text{(Nil)} \qquad \frac{subll\ as\ as'}{subll\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subll\ as\ as'}{subll\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\begin{aligned} subll\ bs\ bs' \longrightarrow\ & \exists as.\ bs = [] \wedge bs' = as \\ & \vee \\ & \exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge subll\ as\ as' \\ & \vee \\ & \exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge subll\ as\ as' \end{aligned}$$

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\vdots}{subll\ [4,6,\ldots]\ [4,5,\ldots]}\ \text{(Cons)}}{subll\ [4,6,\ldots]\ [3,4,5,\ldots]}\ \text{(ConsR)}}{subll\ [2,4,6,\ldots]\ [2,3,4,5,\ldots]}\ \text{(Cons)}}{subll\ [2,4,6,\ldots]\ [1,2,3,4,5,\ldots]}\ \text{(ConsR)}}{subll\ [0,2,4,6,\ldots]\ [0,1,2,3,4,5,\ldots]}\ \text{(Cons)}$$

Desired properties for the predicate subll

$$\frac{\cdot}{\mathit{subll}\ []\ as}\ (\text{Nil}) \qquad \frac{\mathit{subll}\ as\ as'}{\mathit{subll}\ as\ (a\#as')}\ (\text{ConsR})$$

$$\frac{\mathit{subll}\ as\ as'}{\mathit{subll}\ (a\#as)\ (a\#as')}\ (\text{Cons})$$

$$\begin{aligned}\mathit{subll}\ bs\ bs' \longrightarrow\ & \exists as.\ bs = [] \wedge bs' = as \\ & \vee \\ & \exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge \mathit{subll}\ as\ as' \\ & \vee \\ & \exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge \mathit{subll}\ as\ as'\end{aligned}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\mathit{subll}\ [4,6,\ldots]\ [4,5,\ldots]}\ (\text{Cons})}{\mathit{subll}\ [4,6,\ldots]\ [3,4,5,\ldots]}\ (\text{ConsR})}{\mathit{subll}\ [2,4,6,\ldots]\ [2,3,4,5,\ldots]}\ (\text{Cons})}{\mathit{subll}\ [2,4,6,\ldots]\ [1,2,3,4,5,\ldots]}\ (\text{ConsR})}{\mathit{subll}\ [0,2,4,6,\ldots]\ [0,1,2,3,4,5,\ldots]}\ (\text{Cons})$$

So accepting infinite proofs with our introduction rules would solve our problem...

Desired properties for the predicate $sublll$

$$\frac{\cdot}{sublll\ []\ as}\ \text{(Nil)} \qquad \frac{sublll\ as\ as'}{sublll\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{sublll\ as\ as'}{sublll\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\begin{aligned} sublll\ bs\ bs' \longrightarrow\ & \exists as.\ bs = [] \wedge bs' = as \\ & \vee \\ & \exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge sublll\ as\ as' \\ & \vee \\ & \exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge sublll\ as\ as' \end{aligned}$$

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\vdots}{sublll\ [4,6,\ldots]\ [4,5,\ldots]}\ \text{(Cons)}}{sublll\ [4,6,\ldots]\ [3,4,5,\ldots]}\ \text{(ConsR)}}{sublll\ [2,4,6,\ldots]\ [2,3,4,5,\ldots]}\ \text{(Cons)}}{sublll\ [2,4,6,\ldots]\ [1,2,3,4,5,\ldots]}\ \text{(ConsR)}}{sublll\ [0,2,4,6,\ldots]\ [0,1,2,3,4,5,\ldots]}\ \text{(Cons)}$$

So accepting infinite proofs with our introduction rules would solve our problem...
But how about something finitely expressible – a <u>blueprint</u> for an infinite proof?

Desired properties for the predicate $subll$

$$\frac{\cdot}{subll\ []\ as}\ (\text{Nil}) \qquad \frac{subll\ as\ as'}{subll\ as\ (a\#as')}\ (\text{ConsR})$$

$$\frac{subll\ as\ as'}{subll\ (a\#as)\ (a\#as')}\ (\text{Cons})$$

$$\begin{aligned} subll\ bs\ bs' \longrightarrow\ & \exists as.\ bs = [] \wedge bs' = as \\ & \vee \\ & \exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge subll\ as\ as' \\ & \vee \\ & \exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge subll\ as\ as' \end{aligned}$$

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\vdots}{subll\ [4,6,\ldots]\ [4,5,\ldots]}\ (\text{Cons})}{subll\ [4,6,\ldots]\ [3,4,5,\ldots]}\ (\text{ConsR})}{subll\ [2,4,6,\ldots]\ [2,3,4,5,\ldots]}\ (\text{Cons})}{subll\ [2,4,6,\ldots]\ [1,2,3,4,5,\ldots]}\ (\text{ConsR})}{subll\ [0,2,4,6,\ldots]\ [0,1,2,3,4,5,\ldots]}\ (\text{Cons})$$

So accepting infinite proofs with our introduction rules would solve our problem...
But how about something finitely expressible – a <u>blueprint</u> for an infinite proof?

$$\begin{aligned} P\ bs\ bs'\ =\ \exists k \in \mathbb{N}.\ & bs = [2k, 2k+2, 2k+4 \ldots] \ \wedge\ bs' = [2k, 2k+1, 2k+2, \ldots] \vee \\ & bs = [2k+2, 2k+4 \ldots] \ \wedge\ bs' = [2k+1, 2k+2, \ldots] \end{aligned}$$

Desired properties for the predicate $subll$

$$\frac{\cdot}{subll\ []\ as}\ \text{(Nil)} \qquad \frac{subll\ as\ as'}{subll\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subll\ as\ as'}{subll\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\begin{aligned} subll\ bs\ bs' \longrightarrow\ & \exists as.\ bs = []\wedge bs' = as \\ & \vee \\ & \exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge subll\ as\ as' \\ & \vee \\ & \exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge subll\ as\ as' \end{aligned}$$

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\vdots}{subll\ [4,6,\ldots]\ [4,5,\ldots]}\ \text{(Cons)}}{subll\ [4,6,\ldots]\ [3,4,5,\ldots]}\ \text{(ConsR)}}{subll\ [2,4,6,\ldots]\ [2,3,4,5,\ldots]}\ \text{(Cons)}}{subll\ [2,4,6,\ldots]\ [1,2,3,4,5,\ldots]}\ \text{(ConsR)}}{subll\ [0,2,4,6,\ldots]\ [0,1,2,3,4,5,\ldots]}\ \text{(Cons)}$$

So accepting infinite proofs with our introduction rules would solve our problem...
But how about something finitely expressible – a blueprint for an infinite proof?

$$\begin{aligned} P\ bs\ bs' \ =\ \exists k \in \mathbb{N}.\ & bs = [2k, 2k+2, 2k+4\ldots]\ \wedge\ bs' = [2k, 2k+1, 2k+2, \ldots] \vee \\ & bs = [2k+2, 2k+4\ldots]\ \wedge\ bs' = [2k+1, 2k+2, \ldots] \end{aligned}$$

Desired properties for the predicate $sublll$

$$\frac{\cdot}{sublll\ []\ as}\ \text{(Nil)} \qquad \frac{sublll\ as\ as'}{sublll\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{sublll\ as\ as'}{sublll\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\begin{aligned} P\ bs\ bs' \longrightarrow\ & \exists as.\ bs = [] \land bs' = as \\ & \lor \\ & \exists as, a, as'.\ bs = as \land bs' = a\#as' \land P\ as\ as' \\ & \lor \\ & \exists a, as, as'.\ bs = a\#as \land bs' = a\#as' \land P\ as\ as' \end{aligned}$$

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\vdots}{sublll\ [4,6,\ldots]\ [4,5,\ldots]}\ \text{(Cons)}}{sublll\ [4,6,\ldots]\ [3,4,5,\ldots]}\ \text{(ConsR)}}{sublll\ [2,4,6,\ldots]\ [2,3,4,5,\ldots]}\ \text{(Cons)}}{sublll\ [2,4,6,\ldots]\ [1,2,3,4,5,\ldots]}\ \text{(ConsR)}}{sublll\ [0,2,4,6,\ldots]\ [0,1,2,3,4,5,\ldots]}\ \text{(Cons)}$$

So accepting infinite proofs with our introduction rules would solve our problem...
But how about something finitely expressible – a blueprint for an infinite proof?

$$\begin{aligned} P\ bs\ bs' = \exists k \in \mathbb{N}.\ & bs = [2k, 2k+2, 2k+4 \ldots] \land bs' = [2k, 2k+1, 2k+2, \ldots] \lor \\ & bs = [2k+2, 2k+4 \ldots] \land bs' = [2k+1, 2k+2, \ldots] \end{aligned}$$

Desired properties for the predicate $subll$

This leads to the coinduction rule for $subll$:

$$\frac{P\ cs\ cs' \qquad \forall\, bs, bs'.\ P\ bs\ bs' \longrightarrow \begin{array}{l} (\exists as.\ bs = [\,] \wedge bs' = as) \vee \\ (\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge P\ as\ as') \vee \\ (\exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge P\ as\ as') \end{array}}{subll\ cs\ cs'}\ (\text{Coinduct})$$

Desired properties for the predicate $sublll$

This leads to the coinduction rule for $subll$:

$$\frac{P\ cs\ cs' \qquad \forall bs, bs'.\ P\ bs\ bs' \longrightarrow \begin{array}{l}(\exists as.\ bs = [] \wedge bs' = as) \vee \\ (\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge P\ as\ as') \vee \\ (\exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge P\ as\ as')\end{array}}{subll\ cs\ cs'}\ (\text{Coinduct})$$

Terminology: consistent with some rules = "closed backwards" under these rules

Desired properties for the predicate $subll$

This leads to the coinduction rule for $subll$:

$$\frac{P\ cs\ cs' \qquad \forall bs, bs'.\ P\ bs\ bs' \longrightarrow \begin{array}{l} (\exists as.\ bs = [] \wedge bs' = as) \vee \\ (\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge P\ as\ as') \vee \\ (\exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge P\ as\ as') \end{array}}{subll\ cs\ cs'}\ \text{(Coinduct)}$$

Terminology: consistent with some rules = "closed backwards" under these rules

Coinduction says: If a relation P is consistent with the introduction rules (Nil), (ConsR) and (Cons), then $P\ cs\ cs'$ implies $subll\ cs\ cs'$ (for every cs, cs'), i.e., $P \leq subll$.

Desired properties for the predicate $sublI$

This leads to the coinduction rule for $sublI$:

$$\frac{P\ cs\ cs' \qquad \forall bs, bs'.\ P\ bs\ bs' \longrightarrow \begin{array}{l}(\exists as.\ bs = [] \land bs' = as) \lor \\ (\exists as, a, as'.\ bs = as \land bs' = a \# as' \land P\ as\ as') \lor \\ (\exists a, as, as'.\ bs = a \# as \land bs' = a \# as' \land P\ as\ as')\end{array}}{sublI\ cs\ cs'}\ \text{(Coinduct)}$$

Terminology: <u>consistent</u> with some rules = “closed backwards” under these rules

Coinduction says: If a relation P is consistent with the introduction rules (Nil), (ConsR) and (Cons), then $P\ cs\ cs'$ implies $sublI\ cs\ cs'$ (for every cs, cs'), i.e., $P \leq sublI$.
In other words, $sublI$ is the greatest (largest) relation that is consistent with the introduction rules.

Desired properties for the predicate $subll$

This leads to the coinduction rule for $subll$:

$$\frac{P\ cs\ cs' \qquad \forall bs, bs'.\ P\ bs\ bs' \longrightarrow \begin{array}{l}(\exists as.\ bs = [] \wedge bs' = as) \vee \\ (\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge P\ as\ as') \vee \\ (\exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge P\ as\ as')\end{array}}{subll\ cs\ cs'}\ \text{(Coinduct)}$$

Desired properties for the predicate $sublI$

This leads to the coinduction rule for $sublI$:

$$
\frac{\begin{array}{l} P\ cs\ cs' \\[2ex] \forall bs, bs'.\ P\ bs\ bs' \longrightarrow \begin{array}{l} (\exists as.\ bs = [] \wedge bs' = as) \vee \\ (\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge P\ as\ as') \vee \\ (\exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge P\ as\ as') \end{array} \end{array}}{sublI\ cs\ cs'}\ (\text{Coinduct})
$$

Remember the operator F on relations extracted from the intro rules:

$$
F\ P \;=\; \lambda bs, bs'.\ \begin{array}{l} (\exists as.\ bs = [] \wedge bs' = as) \vee \\ (\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge P\ as\ as') \vee \\ (\exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge P\ as\ as') \end{array}
$$

Desired properties for the predicate $sublI$

This leads to the coinduction rule for $subll$:

$$\frac{P\ cs\ cs' \qquad \forall bs, bs'.\ P\ bs\ bs' \longrightarrow F\ P\ bs\ bs'}{subll\ cs\ cs'}\ \text{(Coinduct)}$$

Remember the operator F on relations extracted from the intro rules:

$$\begin{array}{lll} & & (\exists as.\ bs = [\,] \wedge bs' = as) \vee \\ F\ P & = & \lambda bs, bs'.\ \ (\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge P\ as\ as') \vee \\ & & (\exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge P\ as\ as') \end{array}$$

Desired properties for the predicate $subll$

This leads to the coinduction rule for $subll$:

$$\frac{P\ cs\ cs' \qquad \forall bs, bs'.\ P\ bs\ bs' \longrightarrow F\ P\ bs\ bs'}{subll\ cs\ cs'}\ \text{(Coinduct)}$$

Remember the operator F on relations extracted from the intro rules:

$$F\ P\ =\ \lambda bs, bs'.\ \begin{array}{l}(\exists as.\ bs = [\,] \wedge bs' = as) \vee \\ (\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge P\ as\ as') \vee \\ (\exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge P\ as\ as')\end{array}$$

Alternative formulation of the rule:

$$\frac{P \le F\ P}{P \le subll}\ \text{(Coinduct)}$$

Desired properties for the predicate $sublL$

This leads to the coinduction rule for $sublL$:

$$\frac{P\ cs\ cs' \qquad \forall bs, bs'.\ P\ bs\ bs' \longrightarrow F\ P\ bs\ bs'}{sublL\ cs\ cs'}\ \text{(Coinduct)}$$

Remember the operator F on relations extracted from the intro rules:

$$F\ P \ = \ \lambda bs, bs'.\ \begin{array}{l}(\exists as.\ bs = [] \wedge bs' = as) \vee \\ (\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge P\ as\ as') \vee \\ (\exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge P\ as\ as')\end{array}$$

Alternative formulation of the rule:

$$\frac{P \leq F\ P}{P \leq sublL}\ \text{(Coinduct)}$$

And since also $sublL \leq F\ sublL$, we have that $sublL$ is the largest post-fixpoint of F

Desired properties for the predicate $subll$

This leads to the coinduction rule for $subll$:

$$\frac{\begin{array}{l} P\ cs\ cs' \\ \forall bs, bs'.\ P\ bs\ bs' \longrightarrow F\ P\ bs\ bs' \end{array}}{subll\ cs\ cs'}\ \text{(Coinduct)}$$

Remember the operator F on relations extracted from the intro rules:

$$F\ P \ = \ \lambda bs, bs'.\ \begin{array}{l} (\exists as.\ bs = [] \wedge bs' = as) \vee \\ (\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge P\ as\ as') \vee \\ (\exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge P\ as\ as') \end{array}$$

Alternative formulation of the rule:

$$\frac{P \leq F\ P}{P \leq subll}\ \text{(Coinduct)}$$

And since also $subll \leq F\ subll$, we have that $subll$ is the largest post-fixpoint of F – Knaster-Tarski again!

Recipe for making sense of coinductive specifications

The relation $\mathit{subll} : \mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}$
specified coinductively by the rules:

$$\frac{\cdot}{\mathit{subll}\ []\ \mathit{as}}\ (\mathsf{Nil}) \qquad \frac{\mathit{subll}\ \mathit{as}\ \mathit{as}'}{\mathit{subll}\ \mathit{as}\ (a\#\mathit{as}')}\ (\mathsf{ConsR})$$

$$\frac{\mathit{subll}\ \mathit{as}\ \mathit{as}'}{\mathit{subll}\ (a\#\mathit{as})\ (a\#\mathit{as}')}\ (\mathsf{Cons})$$

Recipe for making sense of coinductive specifications

The relation $sublll : \mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}$
specified coinductively by the rules:

$$\frac{\cdot}{sublll\ []\ as}\ (\mathsf{Nil}) \qquad \frac{sublll\ as\ as'}{sublll\ as\ (a\#as')}\ (\mathsf{ConsR})$$

$$\frac{sublll\ as\ as'}{sublll\ (a\#as)\ (a\#as')}\ (\mathsf{Cons})$$

"Coinductively" means: greatest relation consistent with the given rules.

Recipe for making sense of coinductive specifications

The relation $sublll : \mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}$ specified coinductively by the rules:

$$\frac{\cdot}{sublll\ []\ as}\ (\mathsf{Nil}) \qquad \frac{sublll\ as\ as'}{sublll\ as\ (a\#as')}\ (\mathsf{ConsR})$$

$$\frac{sublll\ as\ as'}{sublll\ (a\#as)\ (a\#as')}\ (\mathsf{Cons})$$

"Coinductively" means: <u>greatest relation consistent with the given rules</u>.
More precisely: We define $sublll = \mathsf{J}_F$, where $F : (\mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}) \to (\mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool})$ is defined as follows, for all $R : \mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}$:

Recipe for making sense of coinductive specifications

The relation $sublI : \mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}$ specified coinductively by the rules:

$$\frac{\cdot}{sublI\ [\,]\ as}\ (\mathsf{Nil}) \qquad \frac{sublI\ as\ as'}{sublI\ as\ (a\#as')}\ (\mathsf{ConsR})$$

$$\frac{sublI\ as\ as'}{sublI\ (a\#as)\ (a\#as')}\ (\mathsf{Cons})$$

"Coinductively" means: <u>greatest relation consistent with the given rules</u>.
More precisely: We define $sublI = \mathsf{J}_F$, where $F : (\mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}) \to (\mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool})$ is defined as follows, for all $R : \mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}$:

$$\begin{aligned} F\ R = \lambda bs, bs'.\ & \exists as.\ bs = [\,] \wedge bs' = as \\ & \vee \\ & \exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge R\ as\ as' \\ & \vee \\ & \exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge R\ as\ as' \end{aligned}$$

Recipe for making sense of coinductive specifications

The relation $sublll : \mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}$
specified coinductively by the rules:

$$\frac{\cdot}{sublll\ []\ as}\ (\mathsf{Nil}) \qquad \frac{sublll\ as\ as'}{sublll\ as\ (a\#as')}\ (\mathsf{ConsR})$$

$$\frac{sublll\ as\ as'}{sublll\ (a\#as)\ (a\#as')}\ (\mathsf{Cons})$$

"Coinductively" means: <u>greatest relation consistent with the given rules</u>.
More precisely: We define $sublll = \mathsf{J}_F$, where $F : (\mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}) \to (\mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool})$ is defined as follows, for all $R : \mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}$:

$$\begin{aligned} F\ R\ =\ \lambda bs, bs'.\ & \exists as.\ bs = [] \wedge bs' = as \\ & \vee \\ & \exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge R\ as\ as' \\ & \vee \\ & \exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge R\ as\ as' \end{aligned}$$

$\boxed{F \text{ is monotonic, so } \mathsf{J}_F \text{ exists by Knaster-Tarski.}}$

Recipe for making sense of coinductive specifications

Thanks to	... we obtain
$sublll$ being a pre-fixpoint of F	the introduction rules (Nil), (ConsR), (Cons)
$sublll$ being a post-fixpoint of F	the case distinction rule (Cases)
$sublll$ being $\geq$ all post-fixpoints of F	the coinduction rule (Coinduct)

Recipe for making sense of coinductive specifications

Thanks to	... we obtain
$subll$ being a pre-fixpoint of F	the introduction rules (Nil), (ConsR), (Cons)
$subll$ being a post-fixpoint of F	the case distinction rule (Cases)
$subll$ being $\geq$ all post-fixpoints of F	the coinduction rule (Coinduct)

$$\frac{\cdot}{subll\ []\ as}\ \text{(Nil)} \qquad \frac{subll\ as\ as'}{subll\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subll\ as\ as'}{subll\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

Recipe for making sense of coinductive specifications

Thanks to	... we obtain
$sublI$ being a pre-fixpoint of F	the introduction rules (Nil), (ConsR), (Cons)
$sublI$ being a post-fixpoint of F	the case distinction rule (Cases)
$sublI$ being ≥ all post-fixpoints of F	the coinduction rule (Coinduct)

$$\frac{\cdot}{sublI\ []\ as}\ \text{(Nil)} \qquad \frac{sublI\ as\ as'}{sublI\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{sublI\ as\ as'}{sublI\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\frac{\begin{array}{c} sublI\ bs\ bs' \qquad\qquad \forall as.\ bs = [] \wedge bs' = as \longrightarrow P \\ \forall as, as', a.\ bs = as \wedge bs' = a\#as' \wedge sublI\ as\ as' \longrightarrow P \\ \forall as, as', a.\ bs = a\#as \wedge bs' = a\#as' \wedge sublI\ as\ as' \longrightarrow P \end{array}}{P}\ \text{(Cases)}$$

Recipe for making sense of coinductive specifications

Thanks to	... we obtain
$subll$ being a pre-fixpoint of F	the introduction rules (Nil), (ConsR), (Cons)
$subll$ being a post-fixpoint of F	the case distinction rule (Cases)
$subll$ being ≥ all post-fixpoints of F	the coinduction rule (Coinduct)

$$\frac{\cdot}{subll\ []\ as}\ \text{(Nil)} \qquad \frac{subll\ as\ as'}{subll\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subll\ as\ as'}{subll\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\frac{\begin{array}{l} P\ cs\ cs' \\ \forall\, bs, bs'.\ P\ bs\ bs' \longrightarrow \begin{array}{l} (\exists as.\ bs = [] \wedge bs' = as) \vee \\ (\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge P\ as\ as') \vee \\ (\exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge P\ as\ as') \end{array} \end{array}}{subll\ cs\ cs'}\ \text{(Coinduct)}$$

Recipe for making sense of coinductive specifications

Thanks to	... we obtain
$subll$ being a pre-fixpoint of F	the introduction rules (Nil), (ConsR), (Cons)
$subll$ being a post-fixpoint of F	the case distinction rule (Cases)
$subll$ being $\geq$ all post-fixpoints of F	the coinduction rule (Coinduct)

$$\frac{\cdot}{subll\ []\ as}\ \text{(Nil)} \qquad \frac{subll\ as\ as'}{subll\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subll\ as\ as'}{subll\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

$$\frac{\begin{array}{l} P\ cs\ cs' \\ \forall bs, bs'.\ P\ bs\ bs' \longrightarrow \begin{array}{l} (\exists as.\ bs = [] \wedge bs' = as) \vee \\ (\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge (subll\ as\ as' \vee P\ as\ as')) \vee \\ (\exists a, as, as'.\ bs = a\#as \wedge bs' = a\#as' \wedge (subll\ as\ as' \vee P\ as\ as')) \end{array} \end{array}}{subll\ cs\ cs'}$$

$subll$ is also the greatest (post-)fixpoint of
$G = \lambda P.\ F\ (subll \vee P) = \lambda P.\ F\ (\lambda as, as'.\ subll\ as\ as' \vee P\ as\ as')$.

Coinductive definitions are subtle!

$sublL$: LazyList(A) → LazyList(A) → Bool defined coinductively by the following rules:

$$\frac{\cdot}{sublL\ []\ as}\ (\mathsf{Nil}) \qquad \frac{sublL\ as\ as'}{sublL\ as\ (a\#as')}\ (\mathsf{ConsR})$$

$$\frac{sublL\ as\ as'}{sublL\ (a\#as)\ (a\#as')}\ (\mathsf{Cons})$$

Is this really the correct sublist relation on lazy lists?

Coinductive definitions are subtle!

$sublls$: LazyList(A) → LazyList(A) → Bool defined coinductively by the following rules:

$$\frac{\cdot}{sublls\ []\ as}\ (\text{Nil}) \qquad \frac{sublls\ as\ as'}{sublls\ as\ (a\#as')}\ (\text{ConsR})$$

$$\frac{sublls\ as\ as'}{sublls\ (a\#as)\ (a\#as')}\ (\text{Cons})$$

Is this really the correct sublist relation on lazy lists?

Infinite proof of the fact that $[0, 0, \ldots]$ is a sublist of $[1, 1, \ldots]$:

$$\cfrac{\cfrac{\vdots}{sublls\ [0, 0, \ldots]\ [1, 1, \ldots]}\ (\text{ConsR})}{sublls\ [0, 0, \ldots]\ [1, 1\ldots]}\ (\text{ConsR})$$

Coinductive definitions are subtle!

$sublll$: LazyList(A) → LazyList(A) → Bool defined coinductively by the following rules:

$$\frac{\cdot}{sublll\ []\ as}\ (\mathsf{Nil}) \qquad \frac{sublll\ as\ as'}{sublll\ as\ (a\#as')}\ (\mathsf{ConsR})$$

$$\frac{sublll\ as\ as'}{sublll\ (a\#as)\ (a\#as')}\ (\mathsf{Cons})$$

Is this really the correct sublist relation on lazy lists?
Infinite proof of the fact that $[0, 0, \ldots]$ is a sublist of $[1, 1, \ldots]$:

$$\cfrac{\cfrac{\vdots}{sublll\ [0, 0, \ldots]\ [1, 1, \ldots]}\ (\mathsf{ConsR})}{sublll\ [0, 0, \ldots]\ [1, 1 \ldots]}\ (\mathsf{ConsR})$$

Proof by coinduction: Take $P\ bs\ bs'$ be $bs = [0, 0, \ldots] \wedge bs' = [1, 1, \ldots]$.
Then P is consistent with the rules, because $P\ bs\ bs'$ implies $\exists as, a, as'.\ bs = as \wedge bs' = a\#as' \wedge P\ as\ as'$: just take $as = [0, 0, \ldots]$, $a = 1$ and $as' = [1, 1, \ldots]$.
Therefore $P \leq sublll$, i.e., $sublll\ [0, 0, \ldots]\ [1, 1, \ldots]$ holds.

Incorrect coinductive definition of "sublist" for lazy lists:

$$\frac{\cdot}{sublI\ [\,]\ as}\ (\text{Nil}) \qquad \frac{sublI\ as\ as'}{sublI\ as\ (a\#as')}\ (\text{ConsR})$$

$$\frac{sublI\ as\ as'}{sublI\ (a\#as)\ (a\#as')}\ (\text{Cons})$$

Exercise: What relation does this really define?

Incorrect coinductive definition of "sublist" for lazy lists:

$$\frac{\cdot}{\mathit{subll}\ []\ \mathit{as}}\ \textsf{(Nil)} \qquad \frac{\mathit{subll}\ \mathit{as}\ \mathit{as}'}{\mathit{subll}\ \mathit{as}\ (a\#\mathit{as}')}\ \textsf{(ConsR)}$$

$$\frac{\mathit{subll}\ \mathit{as}\ \mathit{as}'}{\mathit{subll}\ (a\#\mathit{as})\ (a\#\mathit{as}')}\ \textsf{(Cons)}$$

Exercise: What relation does this really define?

Incorrect coinductive definition of "sublist" for lazy lists:

$$\frac{\cdot}{sublI\ [\,]\ as}\ \text{(Nil)} \qquad \frac{sublI\ as\ as'}{sublI\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{sublI\ as\ as'}{sublI\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

Exercise: What relation does this really define?

One way to correct it (where $\_@\_ : \mathsf{List}(A) \to \mathsf{LazyList}(A) \to \mathsf{LazyList}(A)$ denotes the appending of a list to a lazy lazy-list):

$$\frac{\cdot}{sublI\ [\,]\ as}\ \text{(Nil)} \qquad \frac{sublI\ as\ as'}{sublI\ (a\#as)\ (bs\ @\ (a\#as'))}\ \text{(ConsAppend)}$$

Induction versus Coinduction

The semantic foundations for induction and coinduction are perfectly dual – via Knaster-Tarski:

- induction: smallest/least pre-fixpoint
- coinduction: largest/greatest post-fipoint

But they have quite different intuitions:

- induction – whatever can be proved using a finite number of rule applications
- coinduction – whatever can be proved using an infinite number of rule applications

Induction versus Coinduction

The semantic foundations for induction and coinduction are perfectly dual – via Knaster-Tarski:

- induction: smallest/least pre-fixpoint
- coinduction: largest/greatest post-fipoint

But they have quite different intuitions:

- induction – whatever can be proved using a finite number of rule applications
- coinduction – whatever can be proved using an infinite number of rule applications

"Smallest versus largest" is mathematically elegant, but the "finite versus infinite proofs" is what gives us guidance when defining or proving things.

Induction versus Coinduction

The semantic foundations for induction and coinduction are perfectly dual – via Knaster-Tarski:

- induction: smallest/least pre-fixpoint
- coinduction: largest/greatest post-fipoint

But they have quite different intuitions:

- induction – whatever can be proved using a finite number of rule applications
- coinduction – whatever can be proved using an infinite number of rule applications

"Smallest versus largest" is mathematically elegant, but the "finite versus infinite proofs" is what gives us guidance when defining or proving things.

$\boxed{\text{Can we make this intuition precise?}}$

Fix a set A. A rule over A is a pair $r = (H, a)$, $H \subseteq A$ is a finite set and $a \in A$.

Fix a set A. A rule over A is a pair $r = (H, a)$, $H \subseteq A$ is a finite set and $a \in A$. If H has the form $\{a_1, \ldots, a_n\}$, we can write the rule r as

$$\frac{a_1 \ \ldots \ a_n}{a}$$

Fix a set A. A rule over A is a pair $r = (H, a)$, $H \subseteq A$ is a finite set and $a \in A$. If H has the form $\{a_1, \ldots, a_n\}$, we can write the rule r as

$$\frac{a_1 \ldots a_n}{a}$$

If $H = \emptyset$, the rule r is called an axiom and can be written as follows:

$$\frac{\cdot}{a}$$

Fix a set A. A rule over A is a pair $r = (H, a)$, $H \subseteq A$ is a finite set and $a \in A$. If H has the form $\{a_1, \ldots, a_n\}$, we can write the rule r as

$$\frac{a_1 \ldots a_n}{a}$$

If $H = \varnothing$, the rule r is called an axiom and can be written as follows:

$$\frac{.}{a}$$

We fix $\mathcal{R}$, a set of rules over A.
We define/specify $\mathsf{I}_{\mathcal{R}}$ inductively by the rules in $\mathcal{R}$, namely:

$$\frac{(H, a) \in \mathcal{R} \qquad\qquad \forall b \in H.\ \mathsf{I}_{\mathcal{R}}\ b}{\mathsf{I}_{\mathcal{R}}\ a}$$

We define/specify $\mathsf{J}_{\mathcal{R}}$ coinductively by the same rules, namely:

$$\frac{(H, a) \in \mathcal{R} \qquad\qquad \forall b \in H.\ \mathsf{J}_{\mathcal{R}}\ b}{\mathsf{J}_{\mathcal{R}}\ a}$$

According to our semantic recipe, the above mean:

We define $F : (A \to \mathsf{Bool}) \to (A \to \mathsf{Bool})$, the operator associated to $\mathcal{R}$, by applying the rules to its input predicate (like we did before in our examples):

$$F\ P\ =\ \lambda a.\ \exists H.\ (H, a) \in \mathcal{R}\ \wedge (\forall a \in H.\ P\ a)$$

F is monotonic, so I_F and J_F exist by Knaster-Tarski.

We define

- $\mathsf{I}_{\mathcal{R}} = \mathsf{I}_F$
- $\mathsf{J}_{\mathcal{R}} = \mathsf{J}_F$

Rule-based definitions

An $\mathcal{R}$-proof tree π is a (possibly infinite) tree whose nodes are labeled with elements of A and such that successor nodes correspond to rules; more precisely, if a node N is labeled with a and its successor nodes $N_1, \ldots, N_k$ are labelled with $a_1, \ldots, a_k$, then $(\{a_1 \ldots a_k\}, a) \in \mathcal{R}$.

An $\mathcal{R}$-proof tree π is a (possibly infinite) tree whose nodes are labeled with elements of A and such that successor nodes correspond to rules; more precisely, if a node N is labeled with a and its successor nodes $N_1, \ldots, N_k$ are labelled with $a_1, \ldots, a_k$, then $(\{a_1 \ldots a_k\}, a) \in \mathcal{R}$.

If a labels the root of an $\mathcal{R}$-proof tree π, we say that π proves a.

An $\mathcal{R}$-proof tree π is a (possibly infinite) tree whose nodes are labeled with elements of A and such that successor nodes correspond to rules; more precisely, if a node N is labeled with a and its successor nodes $N_1, \ldots, N_k$ are labelled with $a_1, \ldots, a_k$, then $(\{a_1 \ldots a_k\}, a) \in \mathcal{R}$.

If a labels the root of an $\mathcal{R}$-proof tree π, we say that π proves a.

An $\mathcal{R}$-proof tree is said to be finite if its set of nodes is finite.

Rule-based definitions

An $\underline{\mathcal{R}\text{-proof tree}}$ π is a (possibly infinite) tree whose nodes are labeled with elements of A and such that successor nodes correspond to rules; more precisely, if a node N is labeled with a and its successor nodes $N_1, \ldots, N_k$ are labelled with $a_1, \ldots, a_k$, then $(\{a_1 \ldots a_k\}, a) \in \mathcal{R}$.

If a labels the root of an $\mathcal{R}$-proof tree π, we say that $\underline{\pi \text{ proves } a}$.

An $\mathcal{R}$-proof tree is said to be $\underline{\text{finite}}$ if its set of nodes is finite.

Assuming $(\{c, b\}, a)$, $(\{d, b\}, b)$, $(\varnothing, c)$, $(\varnothing, d)$, $(\varnothing, b) \in \mathcal{R}$, here's an example of a finite $\mathcal{R}$-proof tree:

$$\dfrac{\dfrac{.}{\mathsf{c}} \qquad \dfrac{\dfrac{.}{\mathsf{d}} \quad \dfrac{.}{\mathsf{b}}}{\mathsf{b}}}{\mathsf{a}}$$

Rule-based definitions

An $\mathcal{R}$-proof tree π is a (possibly infinite) tree whose nodes are labeled with elements of A and such that successor nodes correspond to rules; more precisely, if a node N is labeled with a and its successor nodes $N_1, \ldots, N_k$ are labelled with $a_1, \ldots, a_k$, then $(\{a_1 \ldots a_k\}, a) \in \mathcal{R}$.

If a labels the root of an $\mathcal{R}$-proof tree π, we say that π proves a.

An $\mathcal{R}$-proof tree is said to be finite if its set of nodes is finite.

Assuming $(\{c, b\}, a)$, $(\{d, b\}, b)$, $(\varnothing, c)$, $(\varnothing, d)$, $(\varnothing, b) \in \mathcal{R}$, and here's an example of an infinite $\mathcal{R}$-proof tree:

$$\dfrac{\dfrac{\cdot}{c} \qquad \dfrac{\dfrac{\cdot}{d} \qquad \dfrac{\dfrac{\cdot}{d} \qquad \dfrac{\dfrac{\cdot}{d} \qquad \dfrac{\dfrac{\cdot}{d} \qquad \vdots}{b}}{b}}{b}}{b}}{a}$$

IMO, crucial for the understanding of coinduction

Theorem. For all $a \in A$:
(1) $\mathsf{I}_{\mathcal{R}}\ a$ holds iff there exists a finite $\mathcal{R}$-proof tree that proves a.
(2) $\mathsf{J}_{\mathcal{R}}\ a$ holds iff there exists a (possibly infinite) $\mathcal{R}$-proof tree that proves a.

Note: It's not really about finite versus infinite – that only happens "by coincidence", since we used finitely branching rule systems.

Remove the restriction that rules (A, a) have the set of hypotheses A finite.
Say a tree is well-founded if it has no infinite paths.

More General Version. For all $a \in A$:
(1) $\mathsf{I}_{\mathcal{R}}\ a$ holds iff there exists a well-founded $\mathcal{R}$-proof tree that proves a.
(2) $\mathsf{J}_{\mathcal{R}}\ a$ holds iff there exists a (possibly non-well-founded) $\mathcal{R}$-proof tree that proves a.

An (obviously incomplete ☺) list of good sources of learning about induction and coinduction

Jacobs and Rutten 1997. A tutorial on coalgebra and coinduction

Paulson 2000. A fixedpoint approach to (co)inductive and (co)datatype definitions

Pierce 2002. Types and Programming Languages (Section 21.1. Induction and Coinduction)

Bertot 2008. CoInduction in Coq

Blanchette, Popescu & Traytel 2015. Witnessing (Co)datatypes

Kozen & Silva 2017. Practical coinduction

Chlipala 2019. Certified Programming with Dependent Types (Chapter 5. Infinite data and proofs)