STRUCTURAL INDUCTION

Goal: Allow for induction on inductively-defined sets ("$\leftrightarrow$" types).

Example. For some base type a, Haskell defines lists kind of like

```
data List a = Nil | Cons a (List a)
```

Nil: empty ↪ `[]`

Cons a (List a): nonempty ↪ `x:xs`

So to prove a predicate $P : \{\text{lists of type a}\} \to \{\bot, \top\}$, we prove

1. P(Nil) holds
2. If P(xs) holds for some list xs, Then P(Cons x xs) holds

Ex: For the list concat operator (++) of Haskell,

```
(++) :: [a] -> [a] -> [a]
[]     ++ ys = ys          -- eq. 1
(x:xs) ++ ys = x : (xs ++ ys)   -- eq. 2
```

(a) Left identity = $\forall ys.\ [] ++ ys = ys$

Right identity = $\forall xs.\ xs ++ [] = xs$

Associativity = $\forall xs\ ys\ zs.\ (xs ++ ys) ++ zs = xs ++ (ys ++ zs)$

(b) Proof of left identity: by definition (eq. 1)

Proof of right identity:

① $xs = []$. Then $[] ++ xs \overset{\text{def}}{=} [] ++ [] \overset{\text{eq. 1}}{=} [] \overset{\text{def.}}{=} xs$ ☐

② $xs = x : xs'$. Assume $xs' ++ [] = xs'$. Then

$$xs ++ [] \overset{\text{def.}}{=} (x : xs') ++ [] \overset{\text{eq. 2}}{=} x : (xs' ++ []) \overset{\text{I.H.}}{=} x : xs \overset{\text{def}}{=} xs$$ ☐

Proof of associativity: induction on xs, holding ys and zs arbitrary

① $xs = []$. Then

$$\begin{aligned} (xs ++ ys) ++ zs &= ([] ++ ys) ++ zs && \text{def.} \\ &= ys ++ zs && \text{eq. 1} \\ &= [] ++ (ys ++ zs) && \text{right identity} \\ &= xs ++ (ys ++ zs) && \text{def.} \end{aligned}$$ ☐

② $xs = x : xs'$. Assume $(xs' ++ ys) ++ zs = xs' ++ (ys ++ zs)$. Then

$$\begin{aligned} (xs ++ ys) ++ zs &= ((x : xs') ++ ys) ++ zs && \text{def.} \\ &= (x : (xs' ++ ys)) ++ zs && \text{eq. 2} \\ &= x : ((xs' ++ ys) ++ zs) && \text{eq. 2} \\ &= x : (xs' ++ (ys ++ zs)) && \text{I.H.} \\ &= (x : xs') ++ (ys ++ zs) && \text{eq. 2 reversed} \\ &= xs ++ (ys ++ zs) \end{aligned}$$ ☐

NATURAL DEDUCTION & INFERENCE RULES

antecedents / premises

$$\frac{a_1 \quad a_2 \quad a_3 \quad \cdots \quad a_n}{C}$$

consequent / conclusion

$\cong$ "if $a_1, a_2, \ldots, a_n$ are all true, then C is true"

Very useful for neatly presenting inductively-defined sets $\Rightarrow$ assign. 0. We express the antecedent and consequents in terms of judgments.

Ex. 1: Defining red-black trees using inference rules.

```
data RBColour
  = Red
  | Black

data RBTree
  = RBLeaf
  | RBNode RBColour Item RBTree RBTree
```

(a) Set of all RBTrees:

$$\frac{}{\text{Red RBColour}} \qquad \frac{}{\text{Black RBColour}}$$

$$\frac{}{\text{RBLeaf RBTree}} \qquad \frac{c\ \text{RBColour} \quad x\ \text{Item} \quad t_1\ \text{RBTree} \quad t_2\ \text{RBTree}}{\text{RBNode}\ c\ x\ t_1\ t_2\ \text{RBTree}}$$

(b) Define only the set of valid red-black trees:

Trick: use a helper judgment that drops the black root requirement
Exercise: define this helper judgment! See official sol^n's for details.

Using the helper judgment OK^R we may define OK:

$$\frac{}{\text{RBLeaf OK}} \qquad \frac{x\ \text{Item} \quad t_1\ OK^R \quad t_2\ OK^R}{\text{RBNode Black}\ x\ t_1\ t_2\ OK}$$

Ex. 1: Bool = { bool exprs. using $\wedge, \vee, \neg, T, F, ()$ }, e.g. $\neg(T \vee T)$

(a) Non-simultaneous inductive def'n (aka. inference rules) of Bool:

$$\frac{x \in \{T, F\}}{x \text{ Bool}} \qquad \frac{\otimes \in \{\wedge, \vee\} \quad e_1 \text{ Bool} \quad e_2 \text{ Bool}}{e_1 \otimes e_2 \text{ Bool}} \qquad \frac{e \text{ Bool}}{\neg e \text{ Bool}} \qquad \frac{e \text{ Bool}}{(e) \text{ Bool}}$$

(b) Is this def'n ambiguous? Yes, bc. $T \wedge T \wedge T$ has 2 derivations:

$$\frac{\dfrac{}{T \text{ Bool}} \quad \dfrac{\dfrac{}{T \text{ Bool}} \quad \dfrac{}{T \text{ Bool}}}{T \wedge T \text{ Bool}}}{T \wedge T \wedge T \text{ Bool}} \quad \text{and} \quad \frac{\dfrac{\dfrac{}{T \text{ Bool}} \quad \dfrac{}{T \text{ Bool}}}{T \wedge T \text{ Bool}} \quad \dfrac{}{T \text{ Bool}}}{T \wedge T \wedge T \text{ Bool}}$$

(c) Unambiguous def'n of Bool, with $\neg > \wedge > \vee$ precedence, left. assoc.

Trick: split up the Bool judgment into 3 "subjudgments":

most restrictive ... least restrictive

Atom $\subset$ AExp $\subset$ OExp

- Atom: highest precedence: constants, paren'd exprs + negated exprs.
- AExp: middle precedence: all atoms, plus exprs. with $\wedge$ at top level
- OExp: lowest precedence: all and-exprs., plus exprs. with $\vee$ at top level

Now the rules are

$$\frac{e \text{ Atom}}{e \text{ AExp}} \qquad \frac{e \text{ AExp}}{e \text{ OExp}}$$

← from subset relations

$$\frac{x \in \{T, F\}}{x \text{ Atom}} \qquad \frac{e \text{ Atom}}{\neg e \text{ Atom}} \qquad \frac{e \text{ OExp}}{(e) \text{ Atom}}$$

← use OExp here bc. any expr. is an OExp, but not all exprs. are AExp or Atoms

$$\frac{e_1 \text{ AExp} \quad e_2 \text{ Atom}}{e_1 \wedge e_2 \text{ AExp}} \qquad \frac{e_1 \text{ OExp} \quad e_2 \text{ AExp}}{e_1 \vee e_2 \text{ OExp}}$$

- use same judgment on left to enforce left associativity
- use next most restrictive property on right to enforce precedence

SIMULTANEOUS INDUCTION

In other words: Rule induction with simultaneously-defined judgments.

Ex: Finishing the $M = L$ equivalence proof.

$$\frac{}{\varepsilon\ \mathbf{M}}M_E \qquad \frac{s\ \mathbf{M}}{(s)\ \mathbf{M}}M_N \qquad \frac{s_1\ \mathbf{M} \quad s_2\ \mathbf{M}}{s_1 s_2\ \mathbf{M}}M_J$$

$$\frac{}{\varepsilon\ \mathbf{L}}L_E \qquad \frac{s\ \mathbf{L}}{(s)\ \mathbf{N}}N_N \qquad \frac{s_1\ \mathbf{N} \quad s_2\ \mathbf{L}}{s_1 s_2\ \mathbf{L}}L_J$$

Claim. If $s\ L$, Then $s\ M$.

Proof. First attempt:

① Base case, L_E: $S = \varepsilon$. But $s\ M$ by rule M_E. □

② Inductive case, L_J: $S = S_1 S_2$. Assume that $S_2\ M$. Then

unprovable!

$$\frac{s_1\ M \qquad \dfrac{}{S_2\ M}\ \text{I.H.}}{s_1 s_2\ M}\ M_J$$

Need another inductive hypothesis for s_1

Problem: proof goal too specific, cannot leverage fact that s_1 is derived via. $N \Longrightarrow$ cannot get an inductive hypothesis for s_1.
So prove instead $s\ L$ OR $s\ N$, Then $s\ M$. Implies original goal!

Can now prove ②, but must prove an extra case:

③ Inductive case, N_N: $S = (s')$. Assume $s'M$. Then

$$\frac{\dfrac{}{s'\ M}\ \text{I.H.}}{(s')\ M}\ M_N$$

□