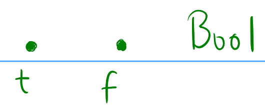
Reminder: Inductive types are freely generated by canonical terms.

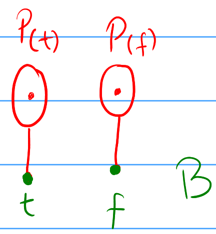
data Bool:
  true : Bool
  false : Bool



To define a (dependent) function it suffices to define it on its canonical elements, hence given

- $P : \mathrm{Bool} \to \mathcal{U}$
- $pt : P(\mathrm{true})$
- $pF : P(\mathrm{false})$

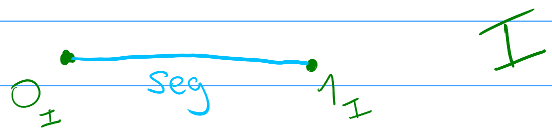
we obtain $\prod_{x:\mathrm{Bool}} P(x)$



and it satisfies the obvious computation rules.

Higher Inductive types are freely generated by canonical terms and paths between them.

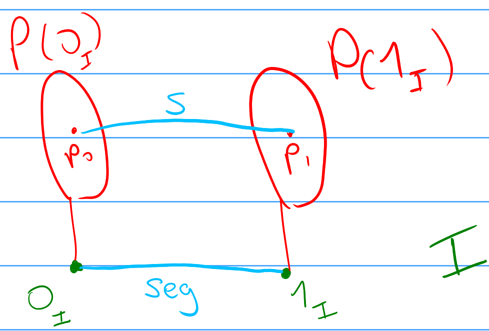
data I:
  $0_I : I$
  $1_I : I$
  $\mathrm{seg} : 0_I = 1_I$



What does a dependent function look like?, Given

- $P : I \to \mathcal{U}$
- $p_0 : P(0_I)$
- $p_1 : P(1_I)$
- $s : p_0 = p_1$

we obtain?



this doesn't typecheck!

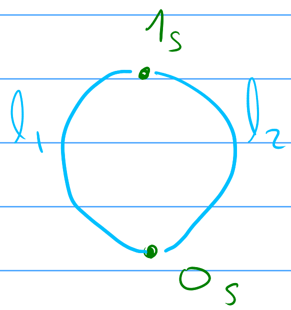
Where should s live then?
the natural choice is

$$\boxed{s : (0_I, p_0) =_{\Sigma P} (1_I, p_1)}$$

this works! But it is not right in general...

Consider the "circle":

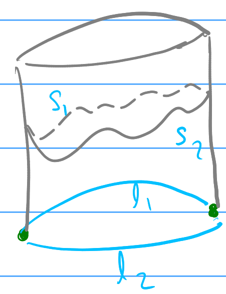
data S^1:
- $0_s : S^1$
- $1_s : S^1$
- $l_1 : 0_s = 1_s$
- $l_2 : 0_s = 1_s$



S^1

According to our previous example, given $P : S^1 \to \mathcal{U}$

- $p_0 : P(0_s)$
- $p_1 : P(1_s)$
- $s_1 : (0_s, p_0) = (1_s, p_1)$
- $s_2 : (0_s, p_0) = (1_s, p_1)$



$$\sum_{x : S^1} P(x)$$

Our type assumptions do not sufficiently axiomatize our guiding image, s_i should be over l_i.

the right type is then

$$\boxed{s_i : \sum_{q : (0_s, p_0) = (1_s, p_1)} pr_1(q) = l_i}$$

That type is a mouthful, luckily we can simplify:

$$\sum_{q : (0_s, p_0) = (1_s, p_1)} pr_1(q) = l_i$$

$$\simeq \sum_{q : \sum_{r : 0_s = 1_s} tr_{p,r}\, p_0 = p_1} pr_1(q) = l_i$$

$$\simeq \sum_{r : 0_s = 1_s} \sum_{t : tr_{p,r}\, p_0 = p_1} r = l_i$$

$$\simeq \sum_{r : 0_s = 1_s} (r = l_i) \times (tr_{p,r}\, p_0 = p_1)$$

$$\boxed{\simeq tr_{p, l_i}\, p_0 = p_1}$$

This is all we need!

The above type corresponds to a "dependent path" between $p_0 : P(0_s)$ and $p_1 : P(1_s)$ over l_i.

Hence, the notation

$$(p_0 =^P_l p_1) :\equiv tr_{p,l}\, p_0 = p_1$$

makes sense.

Summarizing, the induction principle of S^1 is

$$\prod_{P:S^1\to\mathcal{U}}\ \prod_{P_0:P(0_S)}\ \prod_{P_1:P(1_S)} (P_0 =^P_{\ell_1} P_1) \to (P_0 =^P_{\ell_2} P_1) \to \prod_{x:S^1} P(x)$$

Similarly for the interval we have:

$$\prod_{P:I\to\mathcal{U}}\ \prod_{P_0:P(0_I)}\ \prod_{P_1:P(1_I)} (P_0 =^P_{seg} P_1) \to \prod_{x:I} P(x)$$

What about the computation rules?

In the case of the interval for example, the function f obtained should satisfy

$$f(0_S) \equiv P_0 \qquad\qquad f(1_S) \equiv P_1$$

for the dependent path $t : p_0 =^P_{seg} p_1$, we would like it to be equal to some other canonical path $t_f : P_0 =^P_{seg} P_1$ obtained from f.

Is there one?

Lemma: Given a dep. function $f : \prod_{x:A} P(x)$ we have a map

$$\mathrm{apd}_f : \prod_{p:x=y} f(x) =^P_p f(y)$$

Proof:

We need to show

$$\mathrm{tr}_{P,p}\, f(x) = f(y)$$

inducting on p, both sides become $f(x)$. $\square$

With this lemma, the final computation rule would be

$$\mathrm{apd}_f(\mathrm{seg}) = \tau$$

(it is debatable whether these rules should be judgemental or propositional)

<u>Note:</u> For a constant family $P :\equiv \lambda x. B : X \to \mathcal{U}$ we have that for $p : x = y$

$$(u =^P_p v) = u = v$$

while

$$\mathrm{apd}_f(p) =_{(u=v)} \mathrm{ap}_f(p)$$

Lemma: The interval implies funext.

Proof. Given $H : \prod_{x:X} fx = gx$, define for each $x:X$

$$P_x : I \to B$$
$$P_x(0_I) :\equiv f(x)$$
$$P_x(1_I) :\equiv g(x)$$
$$P_x(\mathrm{seg}_I) :\equiv H(x)$$

Finally, define $q : I \to (X \to Y)$

by $q(i,x) = P_x(i)$

then,
$$q(\mathrm{seg}_I) : q(0_I) = q(1_I)$$
$$: (\lambda x. f(x)) = (\lambda x. g(x))$$
$$: f = g$$

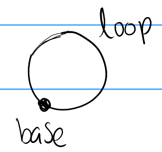
We show some other interesting HITS:

<u>the circle S^1 (Again):</u>

data S^1:

 base : S^1

 loop : base = base



S^1

Given $P : S^1 \to \mathcal{U}$, and

$$\left.\begin{array}{l} \bullet\ b : P(\text{base}) \\ \bullet\ \ell : b =^{P}_{\text{loop}} b. \end{array}\right\} \text{ we obtain } \prod_{x : S^1} P(x)$$

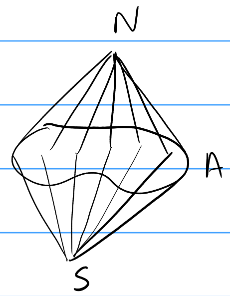
<u>The suspension ΣA</u>, for $A : \mathcal{U}$

data ΣA:

 $N : \Sigma A$

 $S : \Sigma A$

 merid : $A \to (N = S)$



ΣA

Our original circle is then ΣBool.

<u>Pushouts</u>: Given a span

$$\begin{array}{ccc} C & \xrightarrow{g} & B \\ f \downarrow & & \\ A & & \end{array}$$

we have $A \sqcup^C B$ defined as:

data $A \sqcup^C B$:

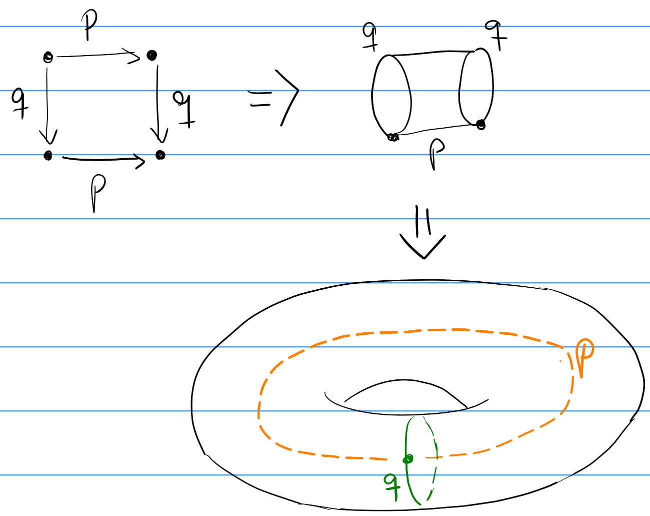
- $\text{inl} : A \to A \sqcup^C B$
- $\text{inr} : B \to A \sqcup^C B$
- $\text{glue} : \prod_{c : C} \text{inl}(f_c) = \text{inr}(g_c)$

It is also fine to give paths between paths.

The torus

data t^2:

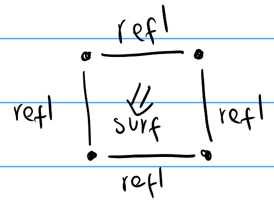
- $b : t^2$
- $p : b = b$
- $q : b = b$
- $r : p \cdot q = q \cdot p$

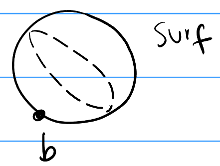


the 2-sphere:

data s^2:

- $b : s^2$
- $\mathrm{surf} : \mathrm{refl}_b = \mathrm{refl}_b$





the n-sphere:

data s^n

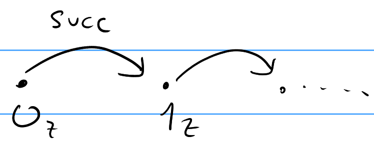
- $b : s^n$
- $\mathrm{fill} : \underbrace{\mathrm{refl}_{\mathrm{refl}_{\ddots_{\mathrm{refl}_b}}}}_{n-1 \text{ times}} = \mathrm{refl}_{\ddots_{\mathrm{refl}_b}}$

(not recommended)

Common types can also be defined more neatly.

The integers

data $\mathbb{Z}$:
  $0_{\mathbb{Z}} : \mathbb{Z}$
  $succ : \mathbb{Z} \simeq \mathbb{Z}$
  $isSet_{\mathbb{Z}} : isSet\ \mathbb{Z}$



For non-dependent functions its ind. principle is easy:

- $a : A$
- $s : A \simeq A$
- $isSet_A : isSet\ A$

$\Big\} \quad \mathbb{Z} \to A$

This definition is fine since it is just adding some paths.

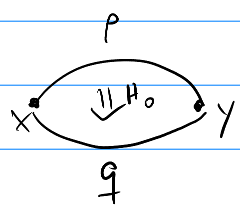
Exploiting this, we can get n-truncations:

$\|-\|_n$

data $\|A\|_n$:
  $|-|_n : A \to \|A\|_n$
  $H_n : is\_hlevel\ n\ A$

For a concrete example, unfolding the case $n=0$:

$$H_0 : \prod_{p,q : x=y} p = q$$



The material on π(S1) is from the HoTT Book, Section 8.1.