Coq и Agda

Coq написан на Ocaml, функциональный язык, французский кузен ML.
Работа в Coq: два варианта: 1-ый вариант - среда CoqIde, 2-ой вариант - ProofGeneral с использованием Emacs.
Agda устанаваливается (как правило) через пакетный менеджера Haskell'а с помощью команды в терминале (`cabal install Agda`). Работа с Agda также происходит в Emacs.

Базовые определения HoTT в Coq.

1. Путь:
Inductive paths $\{A\} : A \to A \to Type := idpath : \forall x, paths\ x\ x.$
Notation $"x \sim\sim> y" := (paths\ x\ y)\ (at\ level\ 70).$

2. Тотальное пространство расслоения:
Inductive total $\{B : Type\}(A : B \to Type) : Type := pair\ (x : B)(y : Ax).$

3. Прямое произведение типов:
Inductive total $\{B : Type\}(A : B \to Type) : Type := total\ (funx : A \Rightarrow B).$

4. Конкатенация путей:
Definition concat $\{A\}\{x\ y\ z : A\} : (x \sim\sim> y) \to (y \sim\sim> z) \to (x \sim\sim> z).$
Proof.
intros p q; *induction p*; *induction q*; *apply idpath*.
Defined.

Базовые определения HoTT в Coq.

5. Обратный путь:
Definition opposite $\{A\}\{x\ y : A\} : (x \sim\sim> y) \to (y \sim\sim> x)$.
intros p; *induction p*; *apply idpath*.
Defined.

6. Индукция путей, заданная через паттерн-матчинг:

```
Ltac path_induction :=
 intros; repeat progress (
  match goal with
   |[p : _ ~~> _ |- _] => induction p
   |_ => idtac
  end
 ); auto.
```

Базовые определения HoTT в Coq.

7. Отображение из простанства в простанство сохраняет пути (аналогично сохраняется при композиции отображений и при обратном отображении):
Lemma map $\{AB\}\{xy : A\}(f : A \to B)(p : x \sim\sim> y) : f\ x \sim\sim> f\ y$.
Proof.
path_induction; *apply idpath*.
Defined.

8. Конкатенация путей в пространстве тоже сохраняется при отображении из пространства в пространство:
Lemma concat_map A B $(x\ y\ z : A)(f : A \to B)(p : x \sim\sim> y)(q : y \sim\sim> z)$:
$map\ f\ (p \mathbin{@} q) \sim\sim> (map\ f\ p) \mathbin{@} (map\ f\ q)$.
Proof.
path_induction.
apply idpath.
Defined.

Базовые определения HoTT в Coq.

9. Хинты и тактики:

```
Hint Resolve
concat_map
opposite_map map_cancel
opposite_concat opposite_opposite
homotopy_concat : path_hints.

Ltac path_tricks :=
first
[apply homotopy_concat
| apply opposite_map
| apply opposite_opposite
| apply opposite_concat
| apply map_cancel
| idtac] ; auto with path_hints.
Ltac path_via x := apply @ concat with(y := x); path_tricks.
```

Базовые определения HoTT в Coq.

10. Утверждение с более интересным доказательством:

```
Lemma map_naturality A (f : A → A)(p : ∀x, fx ∼∼> x)(x y : A)(q : x ∼∼> y) :
map f q @ p y ∼∼>  p x @ q.
Proof.
induction q.
path_via (p x).
apply idpath_left_unit.
apply opposite; apply idpath_right_unit.
Defined.
```

Базовые определения HoTT в Coq.

11. Еще одно утверждение с более интересным доказательством:

```
Lemma homotopy_naturality A B (f g : A → B) (p : ∀x, f x ~~> g x)(x y : A)(q : x ~~>
y) : map f q @ p y ~~> p x @ map g q.
Proof.
inductionq.
path_via (p x).
path_via (idpath (f x) @ p x).
apply opposite; auto.
apply idpath.
apply idpath.
path_via (p x @ idpath (g x)).
apply opposite; auto.
apply idpath.
apply idpath.
Defined.
```

Базовые определения HoTT в Coq.

$$homotopy_naturality = fun(A\ B : Type)(f\ g : A \to B)(p : \forall x : A, fx \sim\sim> gx)(x\ y : A)(q : x \sim\sim> y) \Rightarrow$$
$$paths_rect\ A\ (fun\ (x0\ y0 : A)(q0 : x0 \sim\sim> y0) \Rightarrow\ map\ f\ q0\ @\ p\ y0 \sim\sim> p\ x0\ @\ map\ g\ q0)$$
$$(fun\ x0 : A \Rightarrow (homotopy_concat\ B\ (f\ x0)\ (f\ x0)\ (g\ x0)\ (map\ f$$
$$(idpath\ x0))(idpath\ (f\ x0))\ (p\ x0)\ (p\ x0)(!\ idpath\ (map\ f\ (idpath\ x0)))(idpath\ (p\ x0))\ @$$
$$idpath_left_unit\ \ B\ (f\ x0)\ (g\ x0)\ (p\ x0))\ @\ (!\ idpath_right_unit\ B\ (f\ x0)\ (g\ x0)\ (p\ x0)\ @$$
$$homotopy_concat\ \ B\ (f\ x0)\ (g\ x0)\ (g\ x0)\ (p\ x0)\ (p\ x0)\ (idpath\ (g\ x0))\ (map\ \ g\ (idpath\ \ x0))$$
$$(idpath\ (p\ x0))\ (idpath\ (map\ g\ (idpath\ x0))))))\ x\ y\ q :$$
$$\forall (A\ B : Type)(f\ g : A \to B)(p : \forall x : A, fx \sim\sim> gx)(x\ y : A)(q : x \sim\sim>$$
$$y), map\ f\ q\ @\ p\ y \sim\sim> p\ x\ @\ map\ g\ q$$

Базовые определения HoTT в Coq.

12. Перенос:
Theorem transport $\{A\}\{P : A \to Type\}\{x\ y : A\}(p : x \sim\sim> y) : P\ x \to P\ y$.
Proof.
path_induction.
Defined.

13. Тотальные пути:
Lemma total_paths $(A : Type)(P : A \to Type)(x\ y : sigT\ P)(p : projT1\ x \sim\sim> projT1 y) :$
$(transport\ p\ (projT2\ x) \sim\sim> projT2\ y) \to (x \sim\sim> y)$.
Proof.
intros q.
destruct x as $[x\ H]$; *destruct y as* $[y\ G]$.
simpl in $* \vdash *$.
induction p; *simpl in q*.
path_induction; *apply idpath*.
Defined.

Базовые определения HoTT в Coq.

14. Определение h-слоя:
Definition hfiber $\{AB\}(f : A \to B)(y : B) := \{x : A \wedge fx \sim\sim> y\}$.

15. Перенос h-слоя:
Lemma transport_hfiber $A\ B\ (f : A \to B)(x\ y : A)(z : B)(p : x \sim\sim> y)(q : fx \sim\sim> z)$:
transport $(P := funx \Rightarrow f\ x \sim\sim> z)p\ q \sim\sim> !(map\ f\ p)$ @ q.
Proof.
induction p.
path_via q.
path_via $(!(idpath\ (f\ x))$ @ $q)$.
path_via $(idpath\ (f\ x)$ @ $q)$.
apply opposite; *auto*.
apply idpath; *apply idpath*.
path_induction; *apply opposite*.
apply idmap_map.
Defined.

Базовые определения HoTT в Coq.

$$
\begin{aligned}
&transport_hfiber = fun(A\ B : Type)(f : A \to B)(x\ y : A)(z : B)(p : x \sim\sim> y)(q : fx \sim\sim> \\
&z) \Rightarrow \\
&paths_rect \Rightarrow (fun\ (x0\ y0 : A)(p0 : x0 \sim\sim> y0) \Rightarrow \forall q0 : fx0 \sim\sim> z, transport\ p0\ q0 \sim\sim> \\
&!\ map\ fp0\ @\ q0) \\
&(fun(x0 : A)(q0 : fx0 \sim\sim> z) \Rightarrow \\
&((!\ idpath_left_unit\ B\ (f\ x0)\ z\ (transport\ (idpath\ x0)\ q0)\ @\ homotopy_concat\ B\ (f\ x0)\ (f\ x0)\ z \\
&(idpath\ (f\ x0))\ (!\ idpath\ (f\ x0))\ q0\ q0\ (idpath\ (!\ idpath\ (f\ x0)))(idpath\ q0))\ @\ (let\ b := \\
&fx0\ in\ paths_rect\ B \\
&(fun\ (b0\ z0 : B)(q1 : b0 \sim\sim> z0) \Rightarrow !\ idpath\ b0\ @\ q1 \sim\sim> q1)(fun\ x1 : B \Rightarrow \\
&opposite_left_inverse\ B\ x1\ x1\ (idpath\ x1))\ b\ z\ q0))\ @ \\
&!\ idmap_map\ B\ (f\ x0)\ z\ q0)\ x\ y\ p\ q : \\
&\forall (A\ B : Type)(f : A \to B)(x\ y : A)(z : B)(p : x \sim\sim> y)(q : fx \sim\sim> z), transport\ p\ q \quad > \\
&!\ map\ f\ p\ @\ q
\end{aligned}
$$

Базовые определения HoTT в Coq.

16. Стягиваемый тип (тип под номером -2 в гомотопической иерархии):
Definition contractible A := $x : A \wedge \forall y : A,\ y \sim\sim> x$.

17. Тактика: «стягивание» h-слоя:

```
Ltac contract_hfiber y p :=
match goal with
| [⊢ contractible (@hfiber ___ ?f ?x )] ⇒
 eexists (existT (fun z ⇒ f z ∼∼> x) y p);
   letz := fresh "z" in
   letq := fresh "q" in
    intros [z q]
end.
```

Базовые определения HoTT в Coq.

18. Слабая эквивалентность между типами:
Definition isweq $\{A\ B\}(f : A \to B) := \forall y : B$, *contractible* (*hfiber f y*).
Definition weq A B $:= \{w : A \to B \wedge$ *isweq w*$\}$.

19. Слабая эквивалентность типа с самим собой:
Definition idweq A : *weq A A*.
Proof.
exists (*idmap A*).
intros x.
contract_hfiber x (*idpath x*).
apply total_paths with (*p* := *q*); *simpl*.
compute in q.
path_induction; *apply idpath*.
Qed.

Базовые определения HoTT в Coq.

20. Если между пространствами есть путь, то они слабо эквивалентны:
Definition path_to_weq $\{U\ V\} : U \sim\sim> V \to$ *weq* $U\ V$.
Proof.
intro p.
induction p as [S].
exact (*idweq* S).
Qed.

21. Аксиома унивалентности (слабо эквивалентные эквивалентны):
Axiom univalence : $\forall U\ V$, *isweq* (@*path_to_weq* $U\ V$).

22. Обратное (20) доказывается напрямую с помощью унивалентности:
Definition weq_to_path$\{U\ V\}$: *weq* $U\ V \to U \sim\sim> V$.
Proof.
apply univalence.
Qed.

Тип интервала.

$$
\begin{aligned}
&\mathit{Inductive}\ \mathit{interval} : \mathit{Type}_1 := \\
&\ |\mathit{zero} : \mathit{interval} \\
&\ |\mathit{one} : \mathit{interval}
\end{aligned}
$$

$$\mathit{Axiom}\ \mathit{seq} : \mathit{zero} \sim\sim> \mathit{one}.$$

$$
\begin{aligned}
&\mathit{Definition}\ \mathit{interval_ind}(P : \mathit{interval} \to \mathit{Type})(a : P\ \mathit{zero})(b : P\ \mathit{one})(p : \mathit{seg}\sharp a \sim\sim> b) : \forall x : \\
&\mathit{interval}, P\ x := \\
&\ \mathit{fun}\ x \Rightarrow (\mathit{match}\ x\ \mathit{return}_ \to P\ x\ \mathit{with} \\
&\qquad |\mathit{zero} \Rightarrow \mathit{fun}\ _ \Rightarrow a \\
&\qquad |\mathit{one} \Rightarrow \mathit{fun}\ _ \Rightarrow b \\
&\mathit{end})\ p.
\end{aligned}
$$

Тип интервала.

```
Axiom interval_ind_beta_seq :
∀(P : interval → Type)
 (a : P zero)(b : P one)(p : seg♯a ∼∼> b),
 apD(interval_ind P a b p) seg = p.

Definition interval_rec (P : Type)(a b : P)(p : a ∼∼> b) := interval → P :=
interval_ind (fun _ ⇒ P) a b (transport_const _ _@p).

Definition interval_rec_beta_
 (P : Type)(a b : P)(p : a ∼∼> b) :
 ap (interval_rec P a b p) seg = p.
```

Базовые определения HoTT в Agda.

1. Пути:

$$data\ Id\{A : Type\}(M : A) : A \to Type\ where$$
$$id : Id\ M\ M$$

$$Path : \{A : Type\} \to A \to A \to Type$$
$$Path = Id$$

$$_ == _ : \{A : Type\} \to A \to A \to Type$$
$$_ == _ = Path$$

$$infix\ 9\ _ == _$$

Базовые определения HoTT в Agda.

2. Перенос:
$transport : \{B : Type\}(E : B \to Type)\{b1\ b2 : B\} \to Path\ b1\ b2 \to (E\ b1 \to E\ b2)$
$transportC\ id = \lambda x \to x$

3. Отображение сохраняет:
$ap : \{A\ B : Type\}\{M\ N : A\}(f : A \to B) \to Path\{A\}M\ N \to Path\{B\}(f\ M)(f\ N)$
$ap\ f\ id = id$

$apd : \{B : Type\}\{E : B \to Type\}\{b_1\ b_2 : B\}(f : (x : B) \to E\ x)(\beta : Path\ b_1\ b_2) \to$
$Path\ (transport\ E\ \beta\ (f\ b_1))\ (f\ b_2)$
$apd\ f\ id = id$

4. Coe: *Id*-элиминатор:
$coe : \{A\ B : Type\} \to Path\ A\ B \to A \to B$
$coe = transport(\lambda x \to x)$

Базовые определения HoTT в Agda.

5. Объявление сигма-типа:

```
record Σe (A : Type)(B : A → Type) : Type where
 constructor _,_
 field
  fst : A
  snd : B fst

open Σe public

Σ : {A : Type} → (B : A → Type) → Type

Σ{A}B = Σe A B

infixr 0 _,_

syntax Σe A (λx → B) = Σ[x : A] B
```

Базовые определения HoTT в Agda.

6. Функции:

$$ap ==: \forall\{A\}\{B : A \to Type\}\{f\ g : (x : A) \to B\ x\}$$
$$\to Path\ f\ g \to \{x : A\} \to Path\ (f\ x)(g\ x)$$
$$ap == \alpha\{x\} = ap\ (\lambda f \to f\ x)\alpha$$

postulate :

$$\lambda ==: \forall\ \{A\}\{B : A \to Set\}\{f\ g : (x : A) \to B\ x\} \to ((x : A) \to Path\ (f\ x)(g\ x)) \to Path\ f\ g$$
$$\Pi == \eta : \forall\ \{A\}\{B : A \to Set\}\{f\ g : (x : A) \to B\ x\}$$
$$\to (\alpha : Path\ f\ g) \to \alpha == \lambda == (\lambda x \to ap == \alpha\{x\})$$
$$\Pi == \beta : \forall\ \{A\}\{B : A \to Set\}\{f\ g : (x : A) \to B\ x\}$$
$$\to (\alpha : (x : A) \to Path\ (f\ x)\ (g\ x))\{N : A\} \to ap == (\lambda == \alpha)\{N\} == (\alpha\ N)$$

Базовые определения HoTT в Agda.

7. Эквивалентности:

```
record IsEquiv{A B : Type}(f : A → B) : Type where
 constructor isequiv
 field
  g : B → A
  α : (x : A) → Path (g (f x)) x
  β : (y : B) → Path (f (g y)) y
  γ : (x : A) → Path(β (f x))(ap f (α x))
```

Базовые определения HoTT в Agda.

$Equiv : Type \to Type \to Type$
$Equiv\ A\ B = \Sigma(IsEquiv\{A\}\{B\})$

$equiv : \{A\ B : Type\}(f : A \to B)(g : B \to A)(\alpha : (x : A) \to Path(g\ (f\ x))\ x)(\beta : (y : B) \to Path\ (f\ (g\ y))\ y)(\gamma : (x : A) \to\ Path\ (\beta\ (f\ x))(ap\ f\ (\alpha x))) \to Equiv\ A\ B$
$equiv\ f\ g\ \alpha\ \beta\ \gamma = f,\ isequiv\ g\ \alpha\ \beta\ \gamma$

Базовые определения HoTT в Agda.

$$
\begin{aligned}
&\mathit{record}\ \mathit{IsHEquiv}\ \{A\ B : \mathit{Type}\}(f : A \to B) : \mathit{Type}\ \mathit{where}\\
&\quad \mathit{constructor}\ \mathit{ishequiv}\\
&\quad \mathit{field}\\
&\qquad g : B \to A\\
&\qquad \alpha : (x : A) \to \mathit{Path}\ (g\ (f\ x))\ x\\
&\qquad \beta : (y : B) \to \mathit{Path}\ (f\ (g\ y))\ y
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{HEquiv} : \mathit{Type} \to \mathit{Type} \to \mathit{Type}\\
&\mathit{HEquiv}\ A\ B = \Sigma(\mathit{IsHEquiv}\{A\}\{B\})
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{hequiv} : \{A\ B : \mathit{Type}\}(f : A \to B)(g : B \to A)(\alpha : (x : A) \to \mathit{Path}\ (g\ (f\ x))\ x)(\beta : (y : B) \to\\
&\mathit{Path}\ (f\ (g\ y))y) \to \mathit{HEquiv}\ A\ B\\
&\mathit{hequivf}\ g\ \alpha = f\ ,\ \mathit{ishequiv}\ g\ \alpha\ \beta
\end{aligned}
$$

Базовые определения HoTT в Agda.

Гомотопическая иерархия и стягиваемость:

$data\,TLevel : Type\ where$
$-2 : TLevel$
$S : TLevel \to TLevel$

$-1 : TLevel$
$-1 = S\ -2$

$Contractible : Type \to Type$
$Contractible\ A = \Sigma\lambda(c : A) \to (y : A) \to Path\ c\ y$

$HProp : Type \to Type$
$HProp\ A = NType - 1A$

$HSet : Type \to Type \qquad HGpd : Type \to Type$
$HSet\ A = NType(S(S-2))A \qquad HGpd\ A = NType(S(S(S-2)))\ A$

Унивалентность.

$$coe - is - equiv : \forall\{A\ B\}(p : Path\ A\ B) \to IsEquiv\ (coe\ p)$$

$$coe - equiv : \forall\{A\ B\}(p : Path\ A\ B) \to Equiv\ A\ B$$
$$coe - equivp = (coe\ p\ ,\ coe - is - equiv\ p)$$

$$postulate$$
$$univalence : \forall\{A\ B\} \to IsEquiv\{Path\ A\ B\}\{Equiv\ A\ B\}coe - equiv$$

$$ua : \forall\{A\ B\} \to Equiv\ A\ B \to Path\ A\ B$$
$$ua = IsEquiv.g\ univalence$$

Вычисление гомотопической группы окружности в Agda.

$$
\begin{aligned}
&S^1 : Set \\
&\ base : S^1 \\
&\ loop : base == base
\end{aligned}
$$

$$
\begin{aligned}
&S^1 - induction : (X : S^1 \to Type) \\
&base' : X base \\
&loop' : Path\ (transport\ X\ loop\ base')\ base' \to (y : S^1) \to X\ y
\end{aligned}
$$

$$
\begin{aligned}
&S^1 Cover : S^1 \to Type_0 \\
&S^1 Cover = S^1 - recursion\ Int\ (univalence\ succEquiv)\ x
\end{aligned}
$$

$$
\begin{aligned}
&encode : \{x : S^1\}(p : base == x) \to S^1\ x \\
&encodep = transportS^1\ p\ 0
\end{aligned}
$$

Вычисление гомотопической группы окружности в Agda.

Накручивание петли:
$loop^{\wedge} : (n : \mathbb{Z}) \to base == base$
$loop\ (pos\ 0) = idp$
$loop\ (S\ n) = loop \circ loop^{\wedge}(pos\ n)$
$loop\ (neg\ 0) = !\ loop$
$loop\ (neg\ (S\ n)) =! loop \circ loop^{\wedge}(neg\ n)$

Вычисление гомотопической группы окружности в Agda.

Продолжаем накручивать:

$loop^{\wedge}succ : (n : \mathbb{Z}) \to loop^{\wedge}n \circ loop = loop^{\wedge}(succ\ n)$

$loop^{\wedge}succ\ 0 = loop$

$loop^{\wedge}succ\ (pos\ n) = loop \circ loop^{\wedge}n$

$loop^{\wedge}succ\ (neg\ 0) = ! - inv - l\ loop$

$loop^{\wedge}succ\ (neg\ (S\ n)) =$

$(loop^{\wedge}(neg\ n) \circ !\ loop) \circ loop =$

$loop^{\wedge}(neg\ n) \circ (!\ loop \circ loop) =$

$loop^{\wedge}(neg\ n) \circ idp$

$loop^{\wedge}(neg\ n)$

Вычисление гомотопической группы окружности в Agda.

Кодирование накрученной петли:

$encode - loop^{\wedge} : (n : \mathbb{Z}) \to encode(loop^{\wedge} n) = n$

$encode - loop^{\wedge}\ 0 = idp$

$encode - loop^{\wedge}(pos\ 0) = S^1.coe - loop - \beta\ 0$

$encode - loop^{\wedge}(pos\ (S\ n)) =$

$coe\ (ap\ S^1\ Cover\ (loop^{\wedge}(pos\ n) \circ loop))\ 0$

$coe\ (ap\ S^1\ Cover\ (loop^{\wedge}(pos\ n)) \circ ap\ S^1\ Cover\ loop)\ 0$

$coe\ (ap\ S^1\ Cover\ loop)(coe\ (ap\ S^1\ Cover\ (loop^{\wedge}(pos\ n)))\ 0)$

$coe\ (ap\ S^1\ Cover\ loop)(pos\ n)$

$pos\ (S\ n)$

Вычисление гомотопической группы окружности в Agda.

Декодирование:

$$
\begin{aligned}
&decode : \{x : S^1\} \to Cover\ x \to Path\ base\ x \\
&decode\ \{x\} = S^1 - induction\ (\lambda x' \to Cover\ x' \to Path\ base\ x') \\
&\ loop^{\wedge} \\
&\ (transport\ (\lambda x' \to Cover\ x' \to Path\ base\ x')\ loop\ loop^{\wedge} \\
&\simeq transport\ (\lambda x' \to Path\ base\ x')\ loop \circ loop^{\wedge} \circ transport\ Cover\ (!\ loop) \\
&\simeq (\lambda p \to loop \circ p) \circ loop^{\wedge} transport\ S^1 Cover\ (!\ loop) \\
&\simeq (\lambda p \to loop \circ p) \circ loop^{\wedge} \circ pred \\
&\simeq (\lambda n \to loop \circ (loop^{\wedge}(pred\ n))) \\
&\simeq (\lambda n \to loop^{\wedge} n)
\end{aligned}
$$

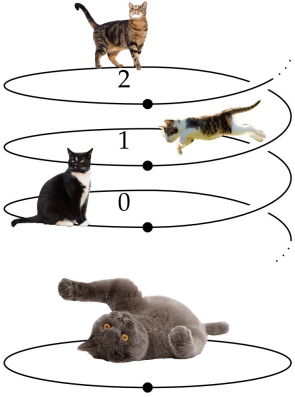
Вычисление гомотопической группы окружности в Agda.

$$encode - decode : \{x : S^1\} \to (c : Cover\ x) \to (encode\ (decode\ \{x\}\ c))c$$
$$encode - decode\{x\} = S^1 - induction$$
$$(\lambda\ (x : S^1) \to (c : Cover\ x) \to Path\ (encode\{x\}(decode\{x\}\ c))\ c)$$
$$encode - loop^{\wedge}\ proof\ x$$

$$decode - encode : \{x : S^1\}(\alpha : Path\ base\ x) \to Path\ (decode\ (encode\ \alpha))\alpha$$
$$decode - encode\{x\}\ \alpha = path - induction\ (\lambda\ (x^{'} : S^1)(\alpha^{'} : Path\ base\ x^{'}) \to$$
$$Path\ (decode\ (encode\ \alpha^{'}))\ \alpha^{'})\ id\ \alpha$$

$$\Omega_1[S^1] - is - Int : HEquiv\ (Path\ base\ base)\ \mathbb{Z}$$
$$\Omega_1[S^1] - is - Int = hequiv\ encode\ decode\ decode - encode\ encode - loop^{\wedge}$$

Merci!



$$\begin{array}{c} \mathbb{R} \\ \downarrow w \\ S^1 \end{array}$$