Polymorfismus

Parametrický polymorfismus

Příklad:

$$\textbf{Definition}\ \text{twice}\ (A : \textbf{Type})\ (f : A \to A)\ (x : A) : A := f\ (f\ x).$$

Typ funkce twice je

$$\forall A : \textbf{Type}, (A \to A) \to A \to A$$

V syntaxi Rocqu je tento typ zapsán takto:

```
forall A : Type, (A -> A) -> A -> A
```

Funkci twice je možné zavolat například takto:

```
twice nat succ 5
```

(Funkce succ zde musí být typu $\text{nat} \to \text{nat}$ a hodnota 5 typu nat.)

Parametrický polymorfismus

Poznámky:

- V teorii typů se varianta typovaného lambda kalkulu, který je rozšířen o tento druh polymorfismu, označuje jako **systém F**.
- V Rocqu (na rozdíl od systému F) se nejedná o samostatnou syntaktickou konstrukci, ale (podobně jako v případě „obyčejných“ funkcí) jde o speciální případ **dependentního produktu**.

Implicitní argumenty

U „typových“ argumentů často dává smysl nastavit je jako **implicitní**:

- pomocí speciálních závorek ($\{\cdots\}$ nebo $[\cdots]$) nebo
- pomocí příkazu **Arguments**

Tyto implicitní argumeny se pak při volání funkce neuvádí a Rocq sám zkusí doplnit jejich hodnotu pomocí typové inference.

Příklad:

Definition twice $\{A : \mathbf{Type}\}$ $(f : A \to A)$ $(x : A) : A := f\ (f\ x)$.

Funkci twice je pak možné zavolat takto:

twice succ 5

Implicitní argumenty

- Implicitní argumenty nejsou v principu omezeny na „typové“ argumenty (tj. argumenty typu **Type**, **Set**, apod.).

  U argumentů „běžných“ typů ale většinou není možné automaticky dopočítat jejich hodnotu na základě hodnot a typů ostatních argumentů, které jsou explicitně uvedeny.

- Pokud chceme implicitní argumenty uvést explicitně (což je někdy potřeba, protože Rocq nemusí být vždy schopen pomocí typové inference doplnit jejich hodnotu), můžeme použít zápis jména funkce se symbolem “@”:

```
@twice nat succ 5
```

Datové typy

- **Typová universa**
  - hierarchie typových univers je dána v jádru systému „napevno“
  - uživatel je nemá možnost definovat ani měnit
  - jsou to jediné typy, které existují ještě předtím, než se začnou načítat standardní knihovny
- **Induktivní datové typy**
  - definují se pomocí klíčových slov **Inductive**, **CoInductive** a **Variant**
  - každá taková definice zavádí nový typ (případně i více typů)
  - mohou se odkazovat k již dříve definovaným datovým typům
- **Dependentní produkt**
  - jeden konkrétní způsob, jak ze dvou existujících typů vytvořit nový typ
  - typ funkcí $A \to B$ z A do B je speciálním případem dependentního produktu

Induktivní typy

Seznam tvořený přirozenými čísly:

```
Inductive natlist : Set :=
  | nil : natlist
  | cons : nat → natlist → natlist.
```

Induktivní typy

Seznam tvořený přirozenými čísly:

Inductive natlist : **Set** :=
| nil : natlist
| cons : nat → natlist → natlist.

Induktivní typy mohou být parametrizovány:

Inductive list $(A : \textbf{Type})$: **Type** :=
| nil : list A
| cons : $A \to$ list $A \to$ list A.

Typy:

list : **Type** → **Type**

nil : $\forall A : \textbf{Type}$, list A
cons : $\forall A : \textbf{Type}$, $A \to$ list $A \to$ list A

Induktivní typy

Návratový typ, který by mohl být použit k reprezentaci výsledku výpočtu, který může skončit chybou:

$$\begin{aligned}&\textbf{Inductive}\ \mathsf{Result}\ (A : \textbf{Type}) : \textbf{Type} := \\ &\quad |\ \mathsf{OK} : A \to \mathsf{Result}\ A \\ &\quad |\ \mathsf{Err} : \mathsf{ErrorInfo} \to \mathsf{Result}\ A.\end{aligned}$$

Induktivní typy

Reprezentace regulárních výrazů:

Inductive RegExp $(S : \mathbf{Set}) : \mathbf{Set} :=$
- $\mid$ Empty : RegExp S
- $\mid$ Eps : RegExp S
- $\mid$ Symb : $S \to$ RegExp S
- $\mid$ Dot : RegExp $S \to$ RegExp $S \to$ RegExp S
- $\mid$ Plus : RegExp $S \to$ RegExp $S \to$ RegExp S
- $\mid$ Star : RegExp $S \to$ RegExp S.

Odpovídá standardní induktivní definici regulárních výrazů nad abecedou Σ:

$$
\begin{aligned}
e ::= \;& \emptyset \\
\mid\; & \varepsilon \\
\mid\; & a \qquad (\text{kde } a \in \Sigma) \\
\mid\; & e_1 \cdot e_2 \\
\mid\; & e_1 + e_2 \\
\mid\; & e_1^*
\end{aligned}
$$

Induktivní typy

Některé standardní typy:

- Kartézský součin ($A \times B$):

  Inductive prod $(A\ B : \mathbf{Type}) : \mathbf{Type}\ :=$
  $\quad \mid$ pair $: A \to B \to$ prod $A\ B$.

- Disjunktní sjednocení ($A + B$):

  Inductive sum $(A\ B : \mathbf{Type}) : \mathbf{Type}\ :=$
  $\quad \mid$ inl $: A \to$ sum $A\ B$
  $\quad \mid$ inr $: B \to$ sum $A\ B$.

Induktivní typy

- Volitelná hodnota typu A (tj. taková, která nemusí existovat):

  Inductive option $(A : \mathbf{Type}) : \mathbf{Type} :=$
  $\quad | \ \mathrm{Some} : A \to \mathrm{option}\ A$
  $\quad | \ \mathrm{None} : \mathrm{option}\ A.$

- Jednoprvkový typ:

  Inductive unit : **Set** :=
  $\quad | \ \mathrm{tt} : \mathrm{unit}.$

- Prázdný typ:

  Inductive Empty_set : **Set** := .

Induktivní typy

Může být najednou definováno několik induktivních typů, které odkazují na sebe navzájem:

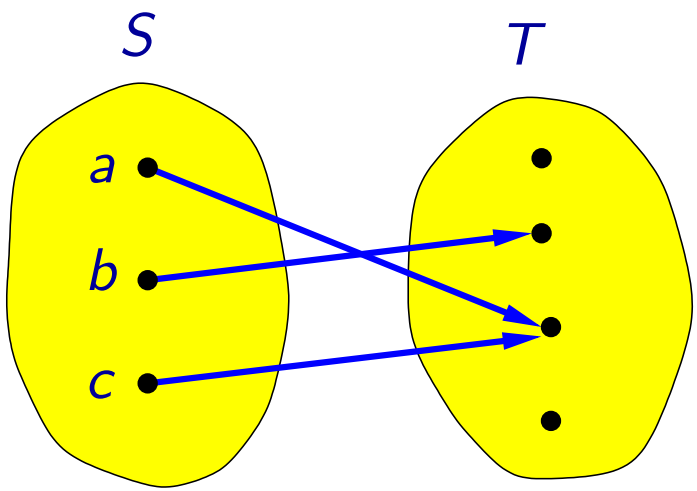
Inductive tree $(A\ B : \mathbf{Type}) : \mathbf{Type} :=$
$\quad | \ \text{leaf} : B \to \text{tree}\ A\ B$
$\quad | \ \text{node} : A \to \text{forest}\ A\ B \to \text{tree}\ A\ B$

with forest $(A\ B : \mathbf{Type}) : \mathbf{Type} :=$
$\quad | \ \text{one\_tree} : \text{tree}\ A\ B \to \text{forest}\ A\ B$
$\quad | \ \text{cons} : \text{tree}\ A\ B \to \text{forest}\ A\ B \to \text{forest}\ A\ B.$

Dependentní typy

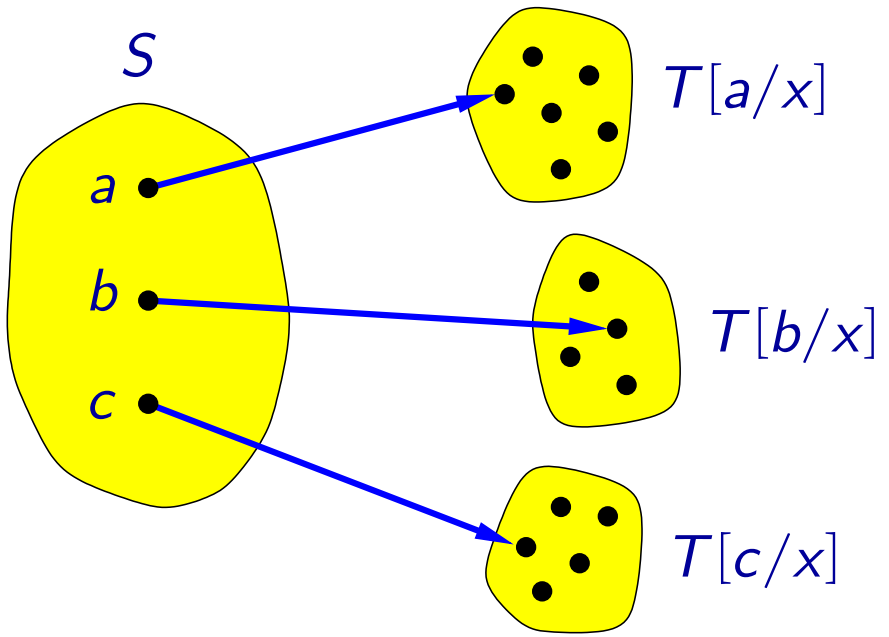
Standardní funkce:

$$f : S \to T$$



Dependentní funkce:

$$f : \forall x : S, T$$



výraz T může obsahovat volnou proměnnou x

Dependentní typy

Poznámka: V literatuře se většinou místo zápisu $f : \forall x : S, T$ používá zápis $f : \prod x : S, T$.

Poznámka: Název produkt pochází pravděpodobně z následující analogie:

- Na množinu všech funkcí typu $S \to T$, kde např. $S = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$, bychom se alternativně mohli dívat jako na kartézský součin:
$$T \times T \times T \times T \times T$$
- Na množinu všech funkcí typu $\mathbb{N} \to T$ bychom se alternativně mohli dívat jako na „nekonečný kartézský součin“:
$$T \times T \times T \times T \times T \times \cdots$$
- Na množinu všech funkcí typu $\prod x : S, T$, kde např. $S = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$, bychom se alternativně mohli dívat jako na kartézský součin:
$$T[\mathtt{a}/x] \times T[\mathtt{b}/x] \times T[\mathtt{c}/x] \times T[\mathtt{d}/x] \times T[\mathtt{e}/x]$$

Dependentní typy

„Obyčejné“ funkční typy jsou speciálním případem dependentních produktů:

Funkční typ $S \to T$ je to samé jako dependentní produkt

$$\forall x : S, T$$

kde se x nevyskytuje jako volná proměnná v T.

V Rocqu jsou tedy „obyčejné“ funkční typy reprezentovány jako pouhá notační zkratka pro příslušný dependentní produkt:

Notation "$A \to B$" := $(\forall \_ : A, B)$: type_scope

Dependentní typy

Speciálním případem dependentního produktu je **parametrický polymorfismus** — případ, kde typ S je typové universum:

$$\forall x : S, T$$

Například

$$\forall X : \mathbf{Type}, X$$
$$\forall X : \mathbf{Type}, X \to X \to X$$
$$\forall X : \mathbf{Type}, (\mathsf{nat} \to X \to X) \to \mathsf{nat} \to \mathsf{list}\ X$$

Dependentní typy

Dependentní typy je možné používat i jako typy **konstruktorů** v induktivních typech:

Inductive A : **Type** :=
| c : $\forall X$: **Type**, $(X \to X) \to A$.

Dependentní typy

Jako induktivní typ je možné vyjádřit například tzv. **dependent sum**, v literatuře většinou označovaný

$$\sum x : A, B$$

kde se x může vyskytovat jako volná proměnná v B:

Inductive dep_sum : **Type** :=
| dep_pair : $\forall x : A,\ B \rightarrow$ dep_sum.

Reprezentuje typ uspořádaných dvojic $\langle a, b \rangle$, kde a je typu A a b je typu $B[a/x]$.

Logika

Typové universum Prop

Jak už bylo uvedeno dříve, **výroky** jsou v Rocqu reprezentovány jako prvky speciálního typového universa **Prop**:

- z formálního hlediska jsou výroky speciálním případem typů
- prvky těchto typů jsou **důkazy**

  Tj. pokud platí

  $$E[\Gamma] \vdash A : \mathbf{Prop}$$

  znamená to, že term A reprezentuje výrok.

  Pokud pak navíc platí

  $$E[\Gamma] \vdash t : A$$

  znamená to, že t je **důkazem** tvrzení A (resp. že t je term, který reprezentuje tento důkaz).

- platí **Prop** : **Type**

Typové universum Prop

Předpokládejme, že M je nějaký typ (např. z universa **Set**):

- unární predikát P reprezentující nějakou vlastnost prvků z M bude typu

$$M \to \mathbf{Prop}$$

  Pokud bude $E[\Gamma] \vdash x : M$, bude $E[\Gamma] \vdash P\ x : \mathbf{Prop}$.

- binární predikát R reprezentující nějakou binární relaci na prvcích z M bude typu

$$M \to M \to \mathbf{Prop}$$

Typové universum Prop

- binární logické spojky jako konjunkce, disjunkce, implikace, atd. budou typu

$$\textbf{Prop} \to \textbf{Prop} \to \textbf{Prop}$$

- univerzální a existenční kvantifikátor nad prvky typu M budou typu

$$(M \to \textbf{Prop}) \to \textbf{Prop}$$

Poznámka: Věci jako logické spojky a kvantifikátory nejsou „natvrdo“ zabudovány v jazyce Rocqu ani v jeho formálním typovém systému.

Tyto věci jsou nadefinovány v jazyce Rocqu v rámci standardních knihoven.

Pokud uživatel chce, může si nadefinovat vlastní alternativy těchto definic.

Logické spojky a kvantifikátory

$\rightarrow$	->		implikace
$\wedge$	/\	and	konjunkce
$\vee$	\/	or	disjunkce
$\leftrightarrow$	<->	iff	ekvivalence
$\neg$	~	neg	negace
$\bot$		False	
$\top$		True	
$\forall$		forall	univerzální kvantifikátor
$\exists$		exists	existenční kvantifikátor

Taktiky

- **Implikace**:

  Taktika intro h.

$$\frac{\cdots, h : A \vdash B}{\cdots \vdash A \to B}$$

  Taktika apply h.

$$\frac{\cdots, h : A \to B \vdash A}{\cdots, h : A \to B \vdash B}$$

$$\frac{\cdots, h : A \to B \to C \vdash A \qquad \cdots, h : A \to B \to C \vdash B}{\cdots, h : A \to B \to C \vdash C}$$

Taktiky

- **Konjunkce**:

  Taktika split.

$$\frac{\cdots \vdash A \qquad \cdots \vdash B}{\cdots \vdash A \wedge B}$$

  Taktika destruct h as $[h_1\ h_2]$.

$$\frac{\cdots, h_1 : A,\ h_2 : B \vdash C}{\cdots, h : A \wedge B \vdash C}$$

Taktiky

- **Disjunkce**:

  Taktika left.

$$\frac{\cdots \vdash A}{\cdots \vdash A \vee B}$$

  Taktika right.

$$\frac{\cdots \vdash B}{\cdots \vdash A \vee B}$$

  Taktika destruct h as $[h_1 \mid h_2]$.

$$\frac{\cdots, h_1 : A \vdash C \qquad \cdots, h_2 : B \vdash C}{\cdots, h : A \vee B \vdash C}$$

Taktiky

- **Ekvivalence**:

  Taktika split.

$$\frac{\cdots \vdash A \to B \qquad \cdots \vdash B \to A}{\cdots \vdash A \leftrightarrow B}$$

  Taktika destruct h as $[h_1\ h_2]$.

$$\frac{\cdots, h_1 : A \to B,\ h_2 : B \to A \vdash C}{\cdots, h : A \leftrightarrow B \vdash C}$$

Taktiky

- **Negace**:

  Taktika intro h.

$$\frac{\cdots, h : A \vdash \bot}{\cdots \vdash \neg A}$$

  Taktika apply h.

$$\frac{\cdots, h : \neg A \vdash A}{\cdots, h : \neg A \vdash \bot}$$

$$\frac{\cdots, h : A \to \neg B \vdash A \qquad \cdots, h : A \to \neg B \vdash B}{\cdots, h : A \to \neg B \vdash \bot}$$

Taktiky

- **False**:

  Taktika exfalso *h*.

$$\frac{\ldots \vdash \bot}{\ldots \vdash A}$$

  Taktika contradition.
  Taktika contradict *h*.
  Taktika destruct *h*.

$$\frac{}{\ldots, h : \bot \vdash A}$$

- **True**:

  Taktika apply I.
  Taktika exact I.
  Taktika constructor.

$$\frac{}{\ldots \vdash \top}$$

Taktiky

- **Univerzální kvantifikátor**:

  Taktika intro x.

$$\frac{\cdots, x : A \vdash B}{\cdots \vdash \forall x : A, B}$$

  Taktika apply h.

$$\frac{}{\cdots, h : \forall x : A, B \vdash B[t/x]}$$

  Taktika generalize t.

$$\frac{\cdots \vdash \forall x : A, B}{\cdots \vdash B[t/x]}$$

Taktiky

- **Existenční kvantifikátor**:

  Taktika exists t.

$$\frac{\ldots \vdash B[t/x]}{\ldots \vdash \exists x : A, B}$$

  Taktika destruct h as $[x\ h_1]$.

$$\frac{\ldots, x : A, h_1 : B \vdash C}{\ldots, h : \exists x : A, B \vdash C}$$

Taktiky

- **Rovnost**:

  Taktika reflexivity.

$$\frac{}{\cdots \vdash t_1 = t_1}$$

  Taktika symmetry.

$$\frac{\cdots \vdash t_2 = t_1}{\cdots \vdash t_1 = t_2}$$

  Taktika transitivity t.

$$\frac{\cdots \vdash t_1 = t \qquad \cdots \vdash t = t_2}{\cdots \vdash t_1 = t_2}$$

Taktiky

- **Důkaz pomocného tvrzení**:

  Taktika assert $(h : A)$.

$$\frac{\cdots \vdash A \qquad \cdots, h : A \vdash B}{\cdots \vdash B}$$

  Taktika enough $(h : A)$.

$$\frac{\cdots, h : A \vdash B \qquad \cdots \vdash A}{\cdots \vdash B}$$

  Taktika cut A.

$$\frac{\cdots \vdash A \to B \qquad \cdots \vdash A}{\cdots \vdash B}$$