Introduction à

Partie 1 : Programmer en Rocq

Fonctions

Type	L’utiliser	L’obtenir
Π x : A, B x A -> B := Π _ : A, B		

Fonctions

Type	L’utiliser	L’obtenir
Π x : A, B x A -> B := Π _ : A, B	Contexte f : A -> B f … : B	

Fonctions

Type	L’utiliser	L’obtenir
Π x : A, B x A -> B := Π _ : A, B	Contexte f : A -> B f … : B	(fun x : A => …) : Π x : A, B x

Types inductifs

Type	L’utiliser	L’obtenir
Inductive prod (A B : Type) : Type := | pair : A -> B -> A * B		
Inductive sum (A B : Type) : Type := | inl : A -> A + B | inr : B -> A + B		

Types inductifs

Type	L'utiliser	L'obtenir
Inductive prod (A B : Type) : Type := | pair : A -> B -> A * B	Contexte p : A * B match p with | (a,b) => … end : C	
Inductive sum (A B : Type) : Type := | inl : A -> A + B | inr : B -> A + B	Contexte s : A + B match s with | inl a => … | inr b => … end : C	

Types inductifs

Type	L'utiliser	L'obtenir
Inductive prod (A B : Type) : Type := | pair : A -> B -> A * B	Contexte p : A * B match p with | (a,b) => … end : C	(… , …) : A * B
Inductive sum (A B : Type) : Type := | inl : A -> A + B | inr : B -> A + B	Contexte s : A + B match s with | inl a => … | inr b => … end : C	inl … : A + B inr … : A + B

Partie 1bis: Prouver en Rocq

Avec Curry-Howard-De Bruijn

Fonctions

Type	L’utiliser	L’obtenir
forall x : A, P x P -> Q := forall _ : P, Q		

Fonctions

Type	L'utiliser	L'obtenir
forall x : A, P x P -> Q := forall _ : P, Q	Contexte f : A -> P f GOAL : P [apply f]	

Fonctions

Type	L'utiliser	L'obtenir
forall x : A, P x P -> Q := forall _ : P, Q	Contexte f : A -> P f GOAL : P [apply f]	(fun x : A => GOAL) : forall x : A, P x [intro x]

Types inductifs

Type	L'utiliser	L'obtenir
Inductive and (P Q : Prop) : Prop := | conj : P -> Q -> P /\ Q		
Inductive or (P Q : Prop) : Prop := | or_introl : P -> P \/ Q | or_intror : Q -> P \/ Q		

Types inductifs

Type	L'utiliser	L'obtenir

Inductive and (P Q : Prop) :
  Prop :=
| conj : P -> Q -> P /\ Q
	Contexte H : P /\ Q
	
	match H with
	| conj HP HQ => GOAL
	end : C
	  [destruct H as [HP HQ]]	

Inductive or (P Q : Prop) : Prop := | or_introl : P -> P \/ Q | or_intror : Q -> P \/ Q		

Types inductifs

Type	L'utiliser	L'obtenir

Inductive and (P Q : Prop) :
  Prop :=
| conj : P -> Q -> P /\ Q
	Contexte H : P /\ Q
	
	match H with
	| conj HP HQ => GOAL
	end : C
	  [destruct H as [HP HQ]]	

Inductive or (P Q : Prop) :
  Prop :=
| or_introl : P -> P \/ Q
| or_intror : Q -> P \/ Q
	Contexte H : P \/ Q
	
	match H with
	| or_introl HP => GOAL1
	| or_intror HQ => GOAL2
	end : C
	  [destruct H as [HP | HQ]]	

Types inductifs

Type	L'utiliser	L'obtenir

Inductive and (P Q : Prop) :
  Prop :=
| conj : P -> Q -> P /\ Q
	Contexte H : P /\ Q
	
	match H with
	| conj HP HQ => GOAL
	end : C
	  [destruct H as [HP HQ]]
		conj GOAL1 GOAL2 : P /\ Q
		[split]

Inductive or (P Q : Prop) :
  Prop :=
| or_introl : P -> P \/ Q
| or_intror : Q -> P \/ Q
	Contexte H : P \/ Q
	
	match H with
	| or_introl HP => GOAL1
	| or_intror HQ => GOAL2
	end : C
	  [destruct H as [HP | HQ]]	

Types inductifs

Type	L'utiliser	L'obtenir

Inductive and (P Q : Prop) :
  Prop :=
| conj : P -> Q -> P /\ Q
	Contexte H : P /\ Q
	
	match H with
	| conj HP HQ => GOAL
	end : C
	  [destruct H as [HP HQ]]
		conj GOAL1 GOAL2 : P /\ Q
		[split]

Inductive or (P Q : Prop) :
  Prop :=
| or_introl : P -> P \/ Q
| or_intror : Q -> P \/ Q
	Contexte H : P \/ Q
	
	match H with
	| or_introl HP => GOAL1
	| or_intror HQ => GOAL2
	end : C
	  [destruct H as [HP | HQ]]
		or_introl GOAL : P \/ Q
		[left]
		or_intror GOAL : P \/ Q
		[right]