Coq and Nuprl
Wojciech Moczydłowski

- History
- World, type system
- Inductive types in Coq
- Extraction in Coq
- Other features of Coq

History

Coq	**Nuprl**
Lambda calculus with types.	Type theory.
Church, Curry, Howard, Girard, Huet, Coquand, Paulin-Mohring.	Russell, Church, Curry, Howard, Martin-Löf, Nuprl group.

Ideology

Coq	**Nuprl**
A **Calculus** of Inductive Constructions.	A foundational system, intented to represent constructive mathematics.
Not assuming stance on FOM.	

Propositions-as-types principle

Type system: Judgements

Coq	**Nuprl**
t : T	**s=t : T**
t is of type T	**s is equal to t in type T**
	T=S **types T and S are equal**

Consistency

Coq	Nuprl
1. Syntactic - proof-theoretic methods, strong normalization. 2. Semantic - models in (domain,set, category) theory.	1. Syntactic - no strong normalization. 2. Semantic - Allen, Howe, Moran. Domain models for Martin-Löf's type theory

Rules

Coq	Nuprl
Only **typing** rules. Core λC has <10 rules (PTS presentation). Inductive definitions - probably about <20 more. Extraction - ??? Reduction - ???	Judgements + extraction terms + tactics. More than 100 rules.

Marketing

World

Coq (8.0)	Nuprl
Set, $Type_i$ predicative Prop impredicative Note: In Coq 7.2 Set impredicative as well	Completely predicative

Type system

Coq	Nuprl
Closed. No really new types can be added.	Open-ended. New types can and are being added.

Logic

Coq	**Nuprl**
Only universal quantifier built-in.	All the logic built-in.
Rest defined using **inductive types** (including equality).	
Can also use Girard's ideas.	

Coq - inductive definitions

- Very generic mechanism.
- Used to define logic, natural numbers, lists, inductive predicates and others...
- To each inductive definition correspond generated principles of induction and recursion.

Coq - inductive types Logic

Inductive False := .
Inductive True := I : True.
Inductive and (A B : Prop) : Prop
:= conj A → B → A ∧ B.
Inductive or (A B : Prop) : Prop :=
or_introl : A → A ∨ B
or_intror : B → A ∨ B.

Natural numbers

Coq	**Nuprl**
Another inductive type.	Built-in construct.

Coq - inductive types
Natural numbers

Inductive nat := 0 : nat
S : nat → nat.

Recursion:
∀ P : nat → Set,
P 0 →
(∀ n : nat. P n → P (S n)) →
∀ n : nat. P n

Coq - inductive types
Natural numbers

Inductive nat := 0 : nat
S : nat → nat.

Induction:
∀ P : nat → Prop,
P 0 →
(∀ n : nat. P n → P (S n)) →
∀ n : nat. P n

Coq - inductive types
List

Inductive List (A:Set) : Set :=
Nil : List A
Cons : A → List A → List A

Recursion:
∀A ∀ P : List A → Set.
P (Nil A) →
(∀ a : A, l : List A. P l → P (a::l))
→ ∀ l : List A.

Coq - inductive types
List

Inductive List (A:Set) : Set :=
Nil : List A
Cons : A → List A → List A

Induction:
∀A ∀ P : List A → Prop.
P (Nil A) →
(∀ a : A, l : List A. P l → P (a::l))
→ ∀ l : List A.

Coq - inductive types <=

Inductive le (n:nat) : nat → Prop
:= le_n : le n n
le_S : ∀m:nat,
le n m → le n (S m).

Coq - extraction

- External mechanism.
- **Proof irrelevance** - Prop doesn't contain computational content and isn't supposed to. Set and Type hierarchy, however, do.

Coq - extraction

Ind. ex (A : x) (P:A → y) : z

ex_intro : ∀x : A, P x → ex A P.

- (x, y, z) = (Type, Prop, Prop)

No computational content

Notation: *exists x : A, P x*

Coq - extraction

Ind. ex (A : x) (P:A → y) : z

ex_intro : ∀x : A, P x → ex A P.

- (x, y, z) = (Set, Prop, Set)

Witness is extracted, proof not.

Notation: *{ x : A P x }*

Coq - extraction

Ind. ex (A : x) (P:A → y) : z

ex_intro : ∀x : A, P x → ex A P.

- (x, y, z) = (Set, Set, Set)

  Everything is extracted.

  Isomorphic to Σ type.

  Notation: *{ x : A & P x }*

Programming language

Coq	**Nuprl**
Not very strong, due to strong normalization.	Full power of Y combinator.
Restrictions on possible programs - structural recursion.	

Environment

Coq	**Nuprl**
Text-mode interface for user interaction.	Sophisticated programming environment, integrated editor, library management etc.
External graphic environment is being developed.	

Installation and system requirements

Coq	**Nuprl**
Installation: easy. Sys.req: Modest.	Installation: hard. Sys.req: High.
Systems: Windows, Unix (Linux/ MacOS/ Solaris...).	Systems: Unix

Coq - other informations

- User base: over 250 people subscribed to the mailing list.
- New book: Coq'Art (2004).
- Website: coq.inria.fr
- Documentation tools.
- Why - a tool for proving correctness of imperative programs. Can use Coq as a backend prover.