Jacek Chrząszcz, Daria Walukiewicz-Chrząszcz

Consistency and completness of rewriting in the calculus of constructions

Institute of Informatics
Warsaw University

Rewriting in the calculus of constructions

$$\textbf{(conv)} \quad \frac{\Gamma \vdash a : b \qquad \Gamma \vdash b' : \star/\Box}{\Gamma \vdash a : b'} \quad (b =_\beta b')$$

Rewriting in the calculus of constructions

$$\textbf{(conv)} \quad \frac{\Gamma \vdash a : b \qquad\qquad \Gamma \vdash b' : \star/\Box}{\Gamma \vdash a : b'} \quad (b =_{\beta\cup R} b')$$

Example:

$$R = \left\{ \begin{array}{l} + : \mathtt{nat} \to \mathtt{nat} \to \mathtt{nat} \\ \quad x + 0 \longrightarrow x \\ \quad x + (S\ y) \longrightarrow S\ (x + y) \\ \quad x + (y + z) \longrightarrow (x + y) + z \end{array} \right\}$$

Rewriting in the calculus of constructions

$$\textbf{(conv)}\quad \frac{\Gamma \vdash a : b \qquad \Gamma \vdash b' : \star/\Box}{\Gamma \vdash a : b'} \quad (b =_{\beta\cup R} b')$$

Example:

$$R = \left\{ \begin{array}{l} + : \mathtt{nat} \to \mathtt{nat} \to \mathtt{nat} \\ \quad x + 0 \longrightarrow x \\ \quad x + (S\ y) \longrightarrow S\ (x + y) \\ \quad x + (y + z) \longrightarrow (x + y) + z \end{array} \right\}$$

$$R, \quad A : P(2) \to Q, \quad B : P(2 + 0) \ \vdash \ A\ B : Q$$

Rewriting in the calculus of constructions

$$\textbf{(conv)} \quad \frac{\Gamma \vdash a : b \qquad \Gamma \vdash b' : \star/\Box}{\Gamma \vdash a : b'} \quad (b =_{\beta \cup R} b')$$

Example:

$$R \;=\; \left\{ \begin{array}{l} + : \texttt{nat} \rightarrow \texttt{nat} \rightarrow \texttt{nat} \\ \quad x + 0 \longrightarrow x \\ \quad x + (S\ y) \longrightarrow S\ (x + y) \\ \quad x + (y + z) \longrightarrow (x + y) + z \end{array} \right\}$$

$$R, \quad A : P(2) \rightarrow Q, \;\; B : P(2 + 0) \;\; \vdash \;\; A\ B : Q$$

Motivation — Coq with rewriting

- easy and comfortable way for defining functions
- larger conversion results in simpler proofs

Necessary metatheoretical properties of CC+Rew

- termination
- confluence
- subject reduction

Necessary metatheoretical properties of CC+Rew

- termination
- confluence
- subject reduction
- completeness
- logical consistency

Necessary metatheoretical properties of CC+Rew

- termination
- confluence
- subject reduction
- completeness
- logical consistency

Our result

Completness checking algorithm and proof of consistency.

Logical consistency

- CC:
  no proof of $\forall x : Prop, x$ in the empty environment

Logical consistency

- CC:
  no proof of $\forall x : Prop, x$ in the empty environment
- CIC:
  no proof of $\forall x : Prop, x$ in any closed environment (only inductive definitions)

Logical consistency

- CC:
  no proof of $\forall x : Prop, x$ in the empty environment
- CIC:
  no proof of $\forall x : Prop, x$ in any closed environment (only inductive definitions)
- CC+Rew:
  no proof of $\forall x : Prop, x$ in any closed environment (only inductive and complete rewrite definitions)

Logical consistency

- CC:
  no proof of $\forall x : Prop, x$ in the empty environment
- CIC:
  no proof of $\forall x : Prop, x$ in any closed environment (only inductive definitions)
- CC+Rew:
  no proof of $\forall x : Prop, x$ in any closed environment (only inductive and complete rewrite definitions)

Canonicity lemma

If $E \vdash t : T$ in a closed environment E then t reduces to a canonical form

Completeness checking algorithm

- starts with $\Delta \vdash f\ x_1 \ldots x_i \ldots x_n,$

Completness checking algorithm

- starts with $\Delta \vdash f\ x_1 \dots x_i \dots x_n,$
- performs successive splittings

$$
\begin{aligned}
\{\,\Delta_1 &\vdash f\ \ \dots(c_1\ \vec{y})\dots, \\
\Delta_2 &\vdash f\ \ \dots(c_2\ \vec{y})\dots, \\
&\quad \dots\dots\dots \\
\Delta_n &\vdash f\ \ \dots(c_n\ \vec{y})\dots\ \}
\end{aligned}
$$

Completness checking algorithm

- starts with $\Delta \vdash f\ x_1 \dots x_i \dots x_n,$
- performs successive splittings
$$\begin{aligned}\{\Delta_1 &\vdash f\ \ \dots (c_1\ \vec{y}) \dots,\\ \Delta_2 &\vdash f\ \ \dots (c_2\ \vec{y}) \dots,\\ &\ \ \dots\dots\dots\\ \Delta_n &\vdash f\ \ \dots (c_n\ \vec{y}) \dots\ \}\end{aligned}$$
- until all goals are reducible by rewrite rules

Completeness checking algorithm

- starts with $\Delta \vdash f\ x_1 \ldots x_i \ldots x_n,$
- performs successive splittings

$$
\begin{aligned}
\{\Delta_1 &\vdash f\ \ \ldots (c_1\ \vec{y}) \ldots, \\
\Delta_2 &\vdash f\ \ \ldots (c_2\ \vec{y}) \ldots, \\
&\ldots\ldots\ldots \\
\Delta_n &\vdash f\ \ \ldots (c_n\ \vec{y}) \ldots\ \}
\end{aligned}
$$

- until all goals are reducible by rewrite rules

Guttag, Hornig 1978, Thiel 1984, Kounalis 1985,

Completeness checking algorithm

- starts with $\Delta \vdash f\ x_1 \ldots x_i \ldots x_n,$
- performs successive splittings

$$\begin{aligned} \{\Delta_1 &\vdash f\ \ \ldots (c_1\ \vec{y}) \ldots, \\ \Delta_2 &\vdash f\ \ \ldots (c_2\ \vec{y}) \ldots, \\ &\ldots\ldots\ldots \\ \Delta_n &\vdash f\ \ \ldots (c_n\ \vec{y}) \ldots\ \} \end{aligned}$$

- until all goals are reducible by rewrite rules

Guttag, Hornig 1978, Thiel 1984, Kounalis 1985,

Coquand 1992, McBride 1999, Schürmann, Pfenning 2003

Example

Example accepted by the algorithm:

$\texttt{list : nat} \longrightarrow \texttt{Set}$

Example

Example accepted by the algorithm:

```
list : nat → Set
nil  : list 0
```

Example

Example accepted by the algorithm:

```
list : nat → Set
nil : list 0
cons : ∀ a:A, ∀ n:nat, list n → list (S n)
```

Example

Example accepted by the algorithm:

```
list : nat → Set
nil : list 0
cons : ∀ a:A, ∀ n:nat, list n → list (S n)

head : ∀ n:nat, list (S n) → A
```

Example

Example accepted by the algorithm:

```
list : nat → Set
nil : list 0
cons : ∀ a:A, ∀ n:nat, list n → list (S n)

head : ∀ n:nat, list (S n) → A
```

$R = \{$ `head n (cons a n l) ⟶ a`

Example

Example accepted by the algorithm:

```
list : nat → Set
nil : list 0
cons : ∀ a:A, ∀ n:nat, list n → list (S n)

head : ∀ n:nat, list (S n) → A
```

$R = \{$ `head n (cons a n l)` $\longrightarrow$ `a`

`head ? nil`

Example

Example accepted by the algorithm:

```
list : nat → Set
nil : list 0
cons : ∀ a:A, ∀ n:nat, list n → list (S n)

head : ∀ n:nat, list (S n) → A
```

$R = \{$ `head n (cons a n l)` $\longrightarrow$ `a`

~~`head ? nil`~~

More examples

```
K : ∀ A:Set, ∀ a b:A, ∀ p q:eq A a b, eq (eq A a b) p q
```

More examples

```
K : ∀ A:Set, ∀ a b:A, ∀ p q:eq A a b, eq (eq A a b) p q
K A a a (refl A a) (refl A a) ⟶ refl (eq A a a) (refl A a)
```

More examples

`K : ∀ A:Set, ∀ a b:A, ∀ p q:eq A a b, eq (eq A a b) p q`

`K A a a (refl A a) (refl A a)` $\longrightarrow$ `refl (eq A a a) (refl A a)`

$+$: `nat` $\rightarrow$ `nat` $\rightarrow$ `nat`

$$x + 0 \longrightarrow x$$
$$x + (S\ y) \longrightarrow S\ (x + y)$$
$$x + (y + z) \longrightarrow (x + y) + z$$

More examples

`K : ∀ A:Set, ∀ a b:A, ∀ p q:eq A a b, eq (eq A a b) p q`

`K A a a (refl A a) (refl A a)` $\longrightarrow$ `refl (eq A a a) (refl A a)`

$+$: `nat` $\rightarrow$ `nat` $\rightarrow$ `nat`

$$x + 0 \longrightarrow x$$
$$x + (S\ y) \longrightarrow S\ (x + y)$$
$$x + (y + z) \longrightarrow (x + y) + z$$

`id : ord` $\rightarrow$ `ord`

More examples

K : ∀ A:Set, ∀ a b:A, ∀ p q:eq A a b, eq (eq A a b) p q

K A a a (refl A a) (refl A a) $\longrightarrow$ refl (eq A a a) (refl A a)

$+$: nat $\rightarrow$ nat $\rightarrow$ nat

$$x + 0 \longrightarrow x$$
$$x + (S\ y) \longrightarrow S\ (x + y)$$
$$x + (y + z) \longrightarrow (x + y) + z$$

id : ord $\rightarrow$ ord

id o $\longrightarrow$ o

id (s x) $\longrightarrow$ s (id x)

id (lim f) $\longrightarrow$ lim (λn:nat. id (f n))

id (id x) $\longrightarrow$ id x

Consistency and completeness of rewriting in CC

Rewriting

- not limited to pattern-matching
- first-order matching

Consistency and completeness of rewriting in CC

Rewriting

- not limited to pattern-matching
- first-order matching

Interesting question

What are the rules not used during completeness check ?
(inductive consequences?)

Consistency and completeness of rewriting in CC

Rewriting

- not limited to pattern-matching
- first-order matching

Interesting question

What are the rules not used during completeness check ?
(inductive consequences?)

Paper

available at `http://www.mimuw.edu.pl/~chrzaszc/papers/`