Inductive Consequences in the Calculus of Constructions

Daria Walukiewicz-Chrząszcz and Jacek Chrząszcz

Institute of Informatics, Warsaw University
ul. Banacha 2, 02-097 Warsaw, Poland

{daria,chrzaszcz}@mimuw.edu.pl

Abstract. In our previous paper we showed a procedure to check completeness of user-defined rewrite systems in the Calculus of Constructions. In many cases this procedure uses only a subset of the rules given by the user, showing that this subset is already complete. In this paper we show that in practical cases the additional rules are inductive consequences of the rules used to prove completeness.

1 Introduction

For a few recent years there is ongoing research aiming at adding rewriting into theorem provers based on type theory and Curry-Howard Isomorphism such as Coq.

Defining functions by rewriting is both easier than using simple pattern matching and moreover, the definition by rewriting can contain rules which add more equations to conversion compared to regular definition by pattern matching. Consider for example addition on unary natural numbers. Using pattern matching and recursion, it can be defined either by induction on the first argument or by induction on the second argument. Using rewriting one can define addition in a symmetric way, therefore making the conversion richer and hence the proofs shorter and more automatic.

In order not to undermine good metatheoretical properties strong normalization and confluence of rewriting must be proved.

In our previous paper [3] we formulated the notion of completeness and showed that completeness implies logical constistency. We also showed a procedure to check completeness in an automatic way. Following the techniques known for first-order rewriting and adapted to higher order by [2], our procedure examines squeletons of all possible instances of arguments to a given function symbols and checks whether they are reducible by rewriting rules defining that function.

In many cases this procedure uses only a subset of the rules given by the user, showing that this subset is already complete.

In this paper we show that in practical cases the additional rules are inductive consequences of the rules used to prove completeness. For example, since two rewriting rules defining addition by induction on the first argument form already a complete system, the additional rules defining the addition by induction on the second argument are inductive consequences of the first two.

For first-order rewriting, this result is a simple consequence of classic theorems about ground reductions. However, for higher-order rewriting, the very notion of inductive consequences had to be adapted. The proof we give is quite simple for the cases where the confluence of critical pairs of the user-defined system cas be proved without rewriting under a binder.

The situation is much more difficult, when this condition is not satisfied, which is usually the case for functions defined over functional inductive types. First of all we must assume the extensionality of conversion, otherwise the result simply does not hold. It is interesting to note that in the work of Nicalas Oury [1], the same assumption is also present even though in the latter work the opposite operation is performed: all proved equations, also by induction, participate in convertibility.

2 Inductive consequences

In this section we try to prove that all rewrite rules which where not used in the proof of completeness are "inductive consequences" of other rules.

We do not know exactly what "inductive consequences" are. We try to formalize it in several ways. Unfortunately, the following "simplest" version of the theorem is not true.

Lemma 1 (Version which in general is not true). *Let E be a closed environment and let $E \vdash Comp(\Gamma; R')$. Let R be the set of rewrite rules used to prove completeness. Suppose that R is confluent (as $R \subseteq R'$ it is obviously stronly normalizing) and $E \vdash Acc(\Gamma, R)$. Then for every rule $G \vdash l \longrightarrow r \in R' - R$ and for every substitution $\sigma : G \rightarrow E; \mathsf{Rew}(\Gamma; R)$ we have $E; \mathsf{Rew}(\Gamma; R) \vdash l\sigma \approx r\sigma$.*

This is not true, because the function symbols from Γ might get under a binder and might be applied to a bound variable instead of a canonical term on which they are reducible. Here is the example:

```
Inductive ord : Set:=
  o : ord
| s : ord -> ord
| lim : (nat -> ord) -> ord.

Rewriting n2o : nat -> ord
Rules
  n2o O --> o
  n2o (S x) --> S (n2o x)

Rewriting id : ord -> ord
Rules
  id o --> o
  id (s x) --> s (id x)
  id (lim f) --> lim (fun n => id (f n))

  id (id x) --> id x
```

The last rewriting system is confluent (unlike the one in which the last rule is replaced with `id x -> x` because the critical pair for `x=(lim f)` needs eta to be joinable).

Now, for $\sigma = \{$`x` $\mapsto$ `lim (fun n => n2o n)`$\}$ one has:

```
lσ=id (id (lim (fun n => n2o n)))
  -->  id (lim (fun n' => id ((fun n => n2o n) n')))
  -->  id (lim (fun n' => id (n2o n')))
  -->  lim (fun n'' => id ((fun n' => id (n2o n')) n''))
  -->  lim (fun n'' => id (id (n2o n'')))

rσ=id (lim (fun n => n2o n))
  -->  lim (fun n' => id ((fun n => n2o n) n'))
  -->  lim (fun n' => id (n2o n'))
```

and they are not equal.

2.1 Inductive consequences when critical pairs are joinable without rewriting under binders

In this subsection we will present a proof stating that in closed environmnets rewriting with $R' - R$ is not needed and can already be done using R. This is done for the case when the critical pairs can be joined with no rewriting under a binder.

Definition 1. *Let us define $E; \mathsf{Rew}(\Gamma; R) \vdash s \xrightarrow{\not\lambda}_S t$ as the smallest relation generated by the rewrite system S ($S = R' - R$ or $S = R$) containing S and stable by contexts that do not involve binders.*

Lemma 2. *Let E be a closed environment and let $E \vdash Acc(\Gamma, R')$ and $E \vdash Comp(\Gamma; R')$. Let R be the set of rewrite rules used to prove completeness. Suppose that critical pairs for R' are joinable in $\xrightarrow{\not\lambda}_{R'}$ and $E \vdash Acc(\Gamma, R)$. For every rewrite step $E; \mathsf{Rew}(\Gamma; R) \vdash s \xrightarrow{\not\lambda}_{R'-R} t$ one has $E; \mathsf{Rew}(\Gamma; R) \vdash s \approx t$.*

Proof. By induction on the well-founded rewrite relation generated by R'.

Since $E; \mathsf{Rew}(\Gamma; R) \vdash s \xrightarrow{\not\lambda}_{R'-R} t$ there exists a rule $G \vdash l \longrightarrow r \in R' - R$ a substitution $\sigma : G \to E; \mathsf{Rew}(\Gamma; R)$ and a position p such that $s = s[l\sigma]_p$, $t = s[r\sigma]_p$ and there are no binders on the path from the root to p. Hence $E; \mathsf{Rew}(\Gamma; R) \vdash l\sigma \xrightarrow{\not\lambda}_{R'-R} r\sigma$.

By completness of R the term $l\sigma$ can be head reduced by $D \vdash g -- > d \in R$. Let us consider the critical pair of $l \longrightarrow r$ and $g \longrightarrow d$. There exist θ the smallest substitution unifying l and g and we have $l\theta = g\theta$ and $l\sigma = l\theta\rho$ for some ρ. Then $E; \mathsf{Rew}(\Gamma; R) \vdash l\sigma \xrightarrow{\not\lambda}_{R'-R} r\sigma$ and $E; \mathsf{Rew}(\Gamma; R) \vdash l\sigma = g\theta\rho \xrightarrow{\not\lambda}_{R} d\theta\rho$. Since critical pairs are joinable in $\xrightarrow{\not\lambda}_{R'}$ there exist term e such that $E; \mathsf{Rew}(\Gamma; R) \vdash r\sigma \xrightarrow{\not\lambda\,*}_{R'} e$ and $E; \mathsf{Rew}(\Gamma; R) \vdash d\theta\rho \xrightarrow{\not\lambda\,*}_{R'} e$. Since $\xrightarrow{\not\lambda}$ is stable by context without binders and since there are no binders on the path p we have also $E; \mathsf{Rew}(\Gamma; R) \vdash s \xrightarrow{\not\lambda}_{R} s[d\theta\rho]_p$, $E; \mathsf{Rew}(\Gamma; R) \vdash t \xrightarrow{\not\lambda\,*}_{R'} s[e]_p$ and $E; \mathsf{Rew}(\Gamma; R) \vdash s[d\theta\rho]_p \xrightarrow{\not\lambda\,*}_{R'} s[e]_p$.

By induction hypothesis all rewrite $\xrightarrow{\not\lambda}_{R'-R}$ steps in $E; \mathsf{Rew}(\Gamma; R) \vdash t \xrightarrow{\not\lambda\,*}_{R'} s[e]_p$ and $E; \mathsf{Rew}(\Gamma; R) \vdash s[d\theta\rho]_p \xrightarrow{\not\lambda\,*}_{R'} s[e]_p$ can be replaced by convertibility in $E; \mathsf{Rew}(\Gamma; R)$. Of course all $\xrightarrow{\not\lambda}_{R}$ steps can also be replaced by convertibility in in $E; \mathsf{Rew}(\Gamma; R)$. Since conversion is symmetric and transitive we get $E; \mathsf{Rew}(\Gamma; R) \vdash t \approx s[d\theta\rho]_p$. Since conversion in $E; \mathsf{Rew}(\Gamma; R)$ contains $\xrightarrow{\not\lambda}_{R}$ we have also $E; \mathsf{Rew}(\Gamma; R) \vdash s \approx s[d\theta\rho]_p$. We conclude $E; \mathsf{Rew}(\Gamma; R) \vdash s \approx t$ by transitivity of $\approx$.

2.2 Rewriting system used by the cover checking algorithm does not have to be confluent

Rewrite systems that occur in rewrite definitions in the environment are by definition terminating and confluent. It follows immediately that the subsytems used by cover checking algorithm are terminating. Unfortunately they do not have to be confluent. More precisely there is a terminating and confluent system and a run of the cover checking alogithm for that system such that the rules used in that run (for immediate coverage) are not confluent.

```
Inductive I :  Set:=
  C : I
| D : I

Inductive Cmp : I -> I -> Set :=
  CC : Cmp C C
| CD : Cmp C D
| DD : Cmp D D

Rewriting F: nat -> (a,b:I), (Cmp a b) -> Set :=
Rules
  F   O    _ _ _ --> O
  F (S x) C _ _ --> S (F x C C CC)
  F (S x) _ D _ --> S x
  F   x   C _ _ --> x
```

It can be checked that the definition of F is terminating and confluent (all critical pairs are joinable). Completness of definition can be shown by the cover checking algorithm started with $F\ x\ y\ z\ u$ performing subsequent splittings on x, y, z and u and using only first three rules. But the system consisting only of these rules is not confluent: the critical pair between the second and the third rules is:

```
  S x <-- F (S x) C D _ --> S (F x C C CC)
```

and it cannot be joined.

References

1. Nicolas Oury. *Egalite et filtrage avec types dependants dans le Calcul des Constructions Inductives.* PhD thesis, University of Paris-Sud, Sept 2006.
2. Carsten Schürmann and Frank Pfenning. A coverage checking algorithm for LF. In D. Basin and B. Wolff, editors, *Proceedings of the Theorem Proving in Higher Order Logics 16th International Conference*, volume 2758 of *Lecture Notes in Computer Science*, pages 120–135, Rome, Italy, September 2003. Springer.
3. Daria Walukiewicz-Chrząszcz and Jacek Chrząszcz. Consistency and completeness of rewriting in the calculus of constructions. In *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 619–631. Springer, 2006.