Strong Induction Principles in the Locally Nameless Representation of Binders (Preliminary Notes)

Christian Urban[1] and Robert Pollack[2]

[1] TU Munich, Germany
[2] Edinburgh University, UK

Abstract. When using the *locally nameless* representation technique for binders, proofs by induction over a definition traditionally involve a weak and strong version of this definition, and a proof that shows both versions derive the same judgements. In these notes we demonstrate with two examples that it is often sufficient to define just the weak version and then use the infrastructure provided by the nominal Isabelle package to derive automatically and in a uniform way a strong induction principle for this weak version. The strong induction principle offers a similar convenience in induction proofs as the traditional approach using weak and strong versions of the definition. From our experience with the nominal Isabelle package, we conjecture that the presented technique can be used in many rule and structural induction proofs.

1 Introduction

The idea that bound variables and free (global) variables should be represented by distinct syntactic classes of names goes back at least to Gentzen and Prawitz. Following a suggestion by Coquand [3], McKinna and Pollack formalized a significant amount of lambda calculus and type theory using such a representation [7, 8]. This work introduced a new technique for handling the requirement of choosing fresh global variables that often occurs in reasoning about binding. (Weakening lemmas are a prototypical example of the problem.) With this technique, reasoning about lambda calculus and type theory is straightforward, if heavy. McKinna and Pollack required very little argument about alpha conversion.[3] Nonetheless, the use of names for bound variables is not a perfect fit to the intuitive notion of binding, so Pollack [10] suggested that the McKinna–Pollack approach to reasoning with two species of variables also works well with a representation that uses names for global variables, and de Bruijn indices for bound variables. This *locally nameless* representation, in which alpha equivalence "classes" have exactly one element, had previously been used in Huet's

[3] E.g. they proved that the representation of Pure Type Systems (PTS) is closed under alpha conversion, but this fact is not needed in reasoning about PTS, including a strengthening lemma and correctness of typechecking algorithms.

Constructive Engine [5] (this required de Bruijn lifting, which the McKinna–Pollack approach obviates) and by Andy Gordon [4] (this requires an extensional logic, which the McKinna–Pollack approach does not). The locally nameless representation with McKinna–Pollack style reasoning has recently been used by several researchers (with several proof tools) for solutions to the POPLmark Challenge [1, 2, 6, 11].

Nonetheless, McKinna–Pollack style reasoning is very heavy. In this paper we show how to considerably lighten the load using the nominal Isabelle package [12]. The most interesting contribution described in these notes is an observation that the technique of strengthening induction principles implemented in this package is also applicable to the locally nameless representation of alpha equivalence classes. To set the stage for our contribution, we present this representation in some detail.

2 Locally Nameless Representation

Consider the following datatype of *locally nameless* pre-terms:

$$t ::= \mathit{Var}\, x \mid \mathit{Bnd}\, i \mid \mathit{App}\, t_1\, t_2 \mid \mathit{Lam}\, t$$

where i is a natural number index and x is a name. As we shall see, a predicate is needed to restrict the pre-terms to those that correspond to well-formed lambda-terms: all indices must actually be bound. A frequently needed operation for pre-terms is the substitution of a term for a de-Bruijn index, defined as follows:

$$\begin{array}{lcl}
\mathit{vsub}\ (\mathit{Var}\, x)\ n\ s & \stackrel{\text{def}}{=} & \mathit{Var}\, x \\
\mathit{vsub}\ (\mathit{Bnd}\, i)\ n\ s & \stackrel{\text{def}}{=} & \begin{cases} \mathit{Bnd}\, i & i < n \\ s & i = n \\ \mathit{Bnd}\ (i-1) & i > n \end{cases} \\
\mathit{vsub}\ (\mathit{App}\, t_1\, t_2)\ n\ s & \stackrel{\text{def}}{=} & \mathit{App}\ (\mathit{vsub}\, t_1\, n\, s)\ (\mathit{vsub}\, t_2\, n\, s) \\
\mathit{vsub}\ (\mathit{Lam}\, t)\ n\ s & \stackrel{\text{def}}{=} & \mathit{Lam}\ (\mathit{vsub}\, t\, (n+1)\, s)
\end{array}$$

Since the "s" argument to vsub will always be a correct (i.e. de Bruijn closed) pre-term, no de Bruijn lifting of s is needed in the Lam case of this definition. It is useful to introduce the following short-hand for the cases where a lambda-abstraction is "opened up" and the zero-index needs to be replaced by a variable.

$$\mathit{freshen}\, t\, x \stackrel{\text{def}}{=} \mathit{vsub}\, t\, 0\ (\mathit{Var}\, x)\ .$$

We can now define the typing relation of simply typed lambda calculus, written $\vdash_w$ (Table 1). In the Lam_w rule, $x \# t$ stands for x not occurring syntactically in t. Types are defined as usual; typing-contexts are lists of (variable, type)-pairs. Note that $\vdash_w$ establishes context validity at the leaves of derivations Var_w and preserves it through the other rules. The side condition $x \# t$ in the rule Lam_w is necessary to prevent too many judgements from being derivable: otherwise x could occur in the rule's conclusion.

$$\frac{\mathit{valid}\,\Gamma \quad (x{:}T) \in \Gamma}{\Gamma \vdash_w \mathit{Var}\,x : T}\ \mathit{Var}_w \qquad \frac{\Gamma \vdash_w t_1 : T_1 \to T_2 \quad \Gamma \vdash_w t_2 : T_2}{\Gamma \vdash_w \mathit{App}\,t_1\,t_2 : T_2}\ \mathit{App}_w$$

$$\frac{x \,\#\, t \quad (x{:}T_1){::}\Gamma \vdash_w \mathit{freshen}\,t\,x : T_2}{\Gamma \vdash_w \mathit{Lam}\,t : T_1 \to T_2}\ \mathit{Lam}_w$$

$$\frac{}{\mathit{valid}\,[]} \qquad \frac{x \,\#\, \Gamma \quad \mathit{valid}\,\Gamma}{\mathit{valid}\,(x{:}T){::}\Gamma}$$

Table 1. Typing Rules in the locally nameless representation.

A Problem with Locally Nameless Representation: Consider proving the weakening property

$$\Gamma_1 \vdash_w t : T \Rightarrow (\forall \Gamma_2.\ \mathit{valid}\,\Gamma_2 \Rightarrow \Gamma_1 \subseteq \Gamma_2 \Rightarrow \Gamma_2 \vdash_w t : T) \tag{1}$$

by induction on the first assumption. This results in a proof obligation for the lambda case

$$\Gamma_2 \vdash_w \mathit{Lam}\,t : T_1 \to T_2 \tag{2}$$

with the assumptions

$$x_0 \,\#\, t, \qquad \Gamma_1 \subseteq \Gamma_2 \qquad \text{and} \qquad \mathit{valid}\,\Gamma_2,$$

for an arbitrary name x_0 (notionally coming from the hypothetical derivation of $\Gamma_1 \vdash_w t : T$ being eliminated). The induction hypothesis is

$$\forall \Gamma_2.\ \mathit{valid}\,\Gamma_2 \Rightarrow (x_0{:}T_1){::}\Gamma_1 \subseteq \Gamma_2 \Rightarrow \Gamma_2 \vdash_w \mathit{freshen}\,t\,x_0 : T_2\ .$$

Applying rule Lam_w to (2) shows we need some name z with

$$z \,\#\, t \qquad \text{and} \qquad (z{:}T_1){::}\Gamma_2 \vdash_w \mathit{freshen}\,t\,z : T_2\ . \tag{3}$$

Taking $z = x_0$ and applying the induction hypothesis gives the goals

$$(x_0{:}T_1){::}\Gamma_1 \subseteq (x_0{:}T_1){::}\Gamma_2 \qquad \text{and} \qquad \mathit{valid}\,(x_0{:}T_1){::}\Gamma_2\ .$$

While the first of these can be discharged using the assumption $\Gamma_1 \subseteq \Gamma_2$, there is no obvious way to prove the second goal, as we cannot show $x_0 \,\#\, \Gamma_2$. The problem is that the particular x_0 appearing in the induction hypothesis is not fresh enough. A direct proof of weakening can be still obtained but requires some non-trivial renamings (see for example [9]).

McKinna–Pollack Style: In order to overcome the problem of needing renamings, [7] defines an alternative *strong* typing relation, written $\vdash_s$ (Table 2).[4]

[4] $\vdash_w$ and $\vdash_s$ are called "weak" and "strong" because $\vdash_s \Rightarrow \vdash_w$ is trivial.

$$\frac{valid\,\Gamma \quad (x{:}T) \in \Gamma}{\Gamma \vdash_s Var\,x : T}\ Var_s \qquad \frac{\Gamma \vdash_s t_1 : T_1 \to t_2 \quad \Gamma \vdash_s t_2 : T_2}{\Gamma \vdash_s App\,t_1\,T_2 : T_2}\ Var_s$$

$$\frac{\forall x.\, x \# \Gamma \Rightarrow (x{:}T_1){::}\Gamma \vdash_s freshen\,t\,x : T_2}{\Gamma \vdash_s Lam\,t : T_1 \to T_2}\ Lam_s$$

Table 2. "Strong" Typing Rules.

The condition $x \# \Gamma$ in rule Lam_s is necessary for enough judgements to be derivable, as the premise is not derivable for any x occurring in Γ. The essential point however is that $\vdash_w$ and $\vdash_s$ are provably equivalent, that is

$$\Gamma \vdash_w t : T \ \Leftrightarrow\ \Gamma \vdash_s t : T. \tag{4}$$

Using this equivalence, one should in proof by induction over the typing rules eliminate $\vdash_s$ (as the induction hypothesis generated by the single premise of rule Lam_s accepts any sufficiently fresh name $x \# \Gamma$) and introduce $\vdash_w$ (as introduction rule Lam_w only requires one name, $x \# t$). The trick in the McKinne–Pollack approach is therefore *not* to prove (1), but instead to prove

$$\Gamma_1 \vdash_s t : T \Rightarrow (\forall \Gamma_2.\, valid\,\Gamma_2 \Rightarrow \Gamma_1 \subseteq \Gamma_2 \Rightarrow \Gamma_2 \vdash_w t : T)\ . \tag{5}$$

Eliminating $\vdash_s$, the induction hypothesis becomes

$$\forall x\,\Gamma_2.\, x \# \Gamma_1 \Rightarrow\ valid\,\Gamma_2 \Rightarrow (x{:}T_1){::}\Gamma_1 \subseteq \Gamma_2 \Rightarrow \Gamma_2 \vdash_w freshen\,t\,x : T_2$$

allowing to use any $x \# \Gamma_1$. Pick $y \# (t, \Gamma_2)$ to instantiate x, and the proof goes through smoothly.

The main problem with this approach is that the equivalence between $\vdash_w$ and $\vdash_s$ is not trivial to prove, and we don't know how to do it automatically. Even the statements of the "weak" and "strong" definitions are not obvious; e.g. the freshness conditions in $\vdash_w$ and $\vdash_s$. Another problem is that we must still explicitly choose a sufficiently fresh name in each induction proof, as we chose $y \# (t, \Gamma_2)$ in the proof of weakening.

3 A Strengthened Induction Principle

The main point of this paper is that, following [12], we may be able to directly derive an induction principle for an inductively defined relation $\mathcal{R}$ (e.g. $\vdash_w$) that is strengthened in the sense that a specified name (e.g. x in rule Lam_w) is chosen fresh for any given finitely supported object.[5] This technique eliminates the need to define an auxiliary relation $\vdash_s$, and packages up the actual choosing of a sufficiently fresh name.

For this to work, we must show that $\mathcal{R}$ is *equivariant*[6] and satisfies the requirements to be *variable condition compatible* (vc-compatible) set out in [12].

[5] That means it cannot mention all names as free.

[6] For a definition of equivariance see for example [12].

For vc-compatibility, we must show that the name x is not in the support of the conclusion of Lam_w (i.e. not in the support of Γ, t and $T_1 \to T_2$), given the side conditions and premises of Lam_w. In this rule x cannot be free in Γ, as otherwise $(x{:}T_1)::\Gamma$ is not a valid context and cannot be part of the typing-judgement in the premise; x cannot be free in t because of the side-condition $x \# t$ imposed in Lam_w; and x cannot be free in $T_1 \to T_2$ because types do not contain any variables. In addition we must show that *valid* and $\vdash_w$ are equivariant (this part is done automatically by nominal Isabelle package provided we supply the fact that *freshen* is equivariant).

Having checked these conditions, we can apply the results from [12] and obtain the following strong induction principle for $\vdash_w$:

$$\frac{\begin{array}{l}\forall \Gamma\, x\, t\, c.\\ \quad valid\, \Gamma \wedge (x{:}T) \in \Gamma \Rightarrow P\, c\, \Gamma\, (Var\, x)\, T\\ \forall \Gamma\, t_1\, t_2\, T_1\, T_2\, c.\\ \quad (\forall d.\, P\, d\, \Gamma\, t_1\, (T_1 \to T_2)) \wedge (\forall d.\, P\, d\, \Gamma\, t_2\, T_1) \Rightarrow P\, c\, \Gamma\, (App\, t_1\, t_2)\, T_2\\ \forall x\, \Gamma\, t\, T_1\, T_2\, c.\\ \quad x \# (t, c) \wedge (\forall d.\, P\, d\, ((x{:}T_1)::\Gamma)\, (freshen\, t\, x)\, T_2) \Rightarrow P\, c\, \Gamma\, (Lam\, t)\, (T_1 \to T_2)\end{array}}{\Gamma \vdash_w t : T \Rightarrow P\, c\, \Gamma\, t\, T}$$

In this induction principle we have to establish the lambda-case under the assumption that x is fresh for the induction context c (which is required to be finitely supported). When applying this induction principle, we can instantiate c appropriately, mimicking in some sense the usual variable convention about binders.

Let us illustrate the usage of the strong induction principle in case of the weakening lemma. We show by induction that

$$\Gamma_1 \vdash_w t : T \Rightarrow\ valid\, \Gamma_2 \Rightarrow \Gamma_1 \subseteq \Gamma_2 \Rightarrow \Gamma_2 \vdash_w t : T \tag{6}$$

holds, for which we instantiate the induction context c with Γ_2. In (6), Γ_1 and Γ_2 are implicitly universally quantified, but note that in contrast to (5), we do not generalize the induction predicate over Γ_2, as this will be implicitly done by the choice to instantiate c with Γ_2 (see quantifiers $(\forall d \ldots)$ in the premises of the strong induction principle). In the lambda-case this means that we have to show:

$$\Gamma_2 \vdash_w Lam\, t : T_1 \to T_2$$

using the induction hypothesis

$$\forall \Gamma_2.\, valid\, \Gamma_2 \Rightarrow (x{:}T_1)::\Gamma_1 \subseteq \Gamma_2 \Rightarrow \Gamma_2 \vdash_w t : T_2 \tag{7}$$

and the assumptions

$$x \# \Gamma_2 \qquad valid\, \Gamma_2 \qquad \Gamma_1 \subseteq \Gamma_2 \qquad x \# t\,.$$

From the first two assumptions we can infer that $(x{:}T_1)::\Gamma_2$ is valid, and from the third that $(x{:}T_1)::\Gamma_1 \subseteq (x{:}T_1)::\Gamma_2$ holds. Consequently, we can use the induction

$$\frac{}{vc_w\ (Var\ x)} \qquad \frac{vc_w\ t_1 \quad vc_w\ t_2}{vc_w\ (App\ t_1\ t_2)} \qquad \frac{vc_w\ (freshen\ t\ x)}{vc_w\ (Lam\ t)}$$

$$\frac{}{vc_s\ (Var\ x)} \qquad \frac{vc_s\ t_1 \quad vc_s\ t_2}{vc_s\ (App\ t_1\ t_2)} \qquad \frac{\forall x.\ vc_s\ (freshen\ t\ x)}{vc_s\ (Lam\ t)}$$

Table 3. "Weak" and "Strong" Rules for Term Well-Formedness.

hypothesis shown in (7) to obtain $(x{:}T_1){::}\Gamma_2 \vdash_w t : T_2$. We can use Lam_w with the fourth assumption to infer that $\Gamma_2 \vdash_w Lam\ t : T_1 \to T_2$ holds, which concludes the proof. The proof is very easy, because by instantiating the induction context with Γ_2, we obtain in the induction step the additional freshness condition $x \# \Gamma_2$, which was not available from the rule induction principle that comes for "free" with $\vdash_w$.

4 Well-Formedness of Terms

An aspect of locally nameless representation that we have only alluded to so far is well-formedness of terms. The pre-term $Lam\ 2$ is not considered a well-formed term because it contains an unbound index: well-formed terms must be de Bruijn closed.[7] (This has been implicitly used in the definition of $vsub$.) We can formalise the well-formedness property by inductive definition, and there are weak and strong forms of the definition, vc_w and vc_s (Table 3). It is possible to prove that vc_w and vc_s are equivalent, but not automatically.

As for typing, we want to derive a strengthened induction principle for vc_w, and completely avoid the use of vc_s. However, a problem arises when we try to do this. Unlike the case for $\vdash_w$, vc_w is *not* vc-compatible. We must consider another definition of the relation, vc:

$$\frac{}{vc\ (Var\ x)} \qquad \frac{vc\ t_1 \quad vc\ t_2}{vc\ (App\ t_1\ t_2)} \qquad \frac{x \# t \quad vc\ (freshen\ t\ x)}{vc\ (Lam\ t)}$$

where we require in the third rule that x must be fresh for t in order to show that x is fresh for the conclusion of that rule. With this we obtain automatically the following strong induction principle for vc:

$$\frac{\begin{array}{l}\forall x\ c.\ P\ c\ (Var\ x)\\ \forall t_1\ t_2\ c.\ (\forall d.\ P\ d\ t_1) \wedge (\forall d.\ P\ d\ t_2) \Rightarrow P\ c\ (App\ t_1\ t_2)\\ \forall x\ t\ c.\ x \#\ (t, c) \wedge (\forall d.\ P\ d\ (freshen\ t\ x)) \Rightarrow P\ c\ (Lam\ t)\end{array}}{vc\ t \Rightarrow P\ c\ t} \tag{8}$$

[7] The reason we have not needed to mention well-formedness so far is that if $\Gamma \vdash_w t : T$, then t is well-formed.

To illustrate the usage of this strong induction principle, we will prove the lemma

$$vc\,s \quad \Rightarrow \quad \forall n.\ s = vsub\ s\ n\ (Var\ x) \tag{9}$$

which states that *vsub* does "nothing" to well-formed terms. In the proof we need the following auxiliary properties:

$$y \# (s,t) \ \wedge \ vsub\ s\ n\ (Var\ y) = vsub\ t\ n\ (Var\ y) \quad \Rightarrow \quad s = t\ , \tag{10}$$

$$y \# (s,t) \quad \Rightarrow \quad y \# vsub\ s\ n\ t\ , \tag{11}$$

$$\begin{aligned} n < m \quad \Rightarrow \quad & vsub\ (vsub\ s\ n\ (Var\ x))\ (m-1)\ (Var\ y) \\ & = \ vsub\ (vsub\ s\ m\ (Var\ y))\ n\ (Var\ x)\ . \end{aligned} \tag{12}$$

Our proof of (9) proceeds by strong induction on $vc\,s$ with the induction context set to x. Only the *Lam* case gives a non-trivial goal

$$Lam\ t = vsub\ (Lam\ t)\ n\ (Var\ x)$$

with induction hypothesis

$$\forall n\ x.\ freshen\ t\ y = vsub\ (freshen\ t\ y)\ n\ (Var\ x)$$

and assumptions $y \# t$ (coming from the *vc* rule for *Lam*) and $y \# x$ (coming from the strengthened induction context). By the definition of *vsub*, and injectivity of constructors, the goal becomes

$$t = vsub\ t\ (n+1)\ (Var\ x)$$

which is solved by (10) if we can show

$$\begin{gathered} y \# (t, vsub\ t\ (n+1)\ (Var\ x))\ , \\ freshen\ t\ y = freshen\ (vsub\ t\ (n+1)\ (Var\ x))\ y\ . \end{gathered}$$

The first of these is straightforward by (11) using the assumptions $y \# (t,x)$. For the second, we have

$$\begin{aligned} freshen\ t\ y &= vsub\ (freshen\ t\ y)\ n\ (Var\ x) && \text{by ih} \\ &= freshen\ (vsub\ t\ (n+1)\ (Var\ x))\ y && \text{by (12)} \end{aligned}$$

and we are finished. Crucial in this proof is that the freshness condition $y \# x$ coming from the strong induction holds, as otherwise we could not have appealed directly to (10) and (11).

5 Discussion and Related Work

Judging from the experience of the strong induction principles with nominal datatypes, we conjecture that the presented approach can be used in many rule and structural induction proofs in the locally nameless approach. By relying on the infrastructure afforded by the nominal Isabelle package, we obtain the strong

induction principles automatically and they are derived in a uniform way. We also get for "free" the conditions of inductive definitions involving locally nameless (pre-)terms for when they are compatible with the strengthening (see [12]). In comparison, in the existing work on locally nameless representation there is no uniform treatment for obtaining the equivalence between the weak and the strong versions of an inductive definition. There seems to be also no treatment of generalisations to cases where a rule contains more than one "binder". Even so, the approach we propose in this paper will only be completely satisfactory, if some problems can be worked out.

Potential Shortcomings: We have already noted some potential problems. We observed that the strengthened induction principle proved for $\vdash_w$ is probably not able to directly prove the property that $\vdash_w$ and $\vdash_s$ are equivalent. Along the same lines, but worse, the strengthened induction principle proved for vc is probably not able to directly prove that vc is equivalent to vc_w, let alone to vc_s. In practice, these may not be serious incompletenesses, and if necessary, the full McKinna–Pollack technology can be used to prove these statements without changing representation. The user isn't committing to a limited representation.

Another shortcoming of this approach may be more serious in practice.[8] In McKinna-Pollack style the strong relations give strong *inversion* principles, as well as strong induction principles. For example, the inversion principles for Lam_w and Lam_s are respectively:

$$\begin{array}{l}\Gamma \vdash_w Lam\, t : T \;\Rightarrow\\ \qquad \exists\, x\, T_1\, T_2.\ (x \,\#\, t \;\wedge\; (x{:}T_1){::}\Gamma \vdash_w freshen\, t\, x : T_2 \;\wedge\; T = T_1{\to}T_2),\end{array}$$

$$\begin{array}{l}\Gamma \vdash_s Lam\, t : T \;\Rightarrow\\ \qquad \exists\, T_1\, T_2.\ \forall\, x.\ (x \,\#\, \Gamma \;\Rightarrow\; (x{:}T_1){::}\Gamma \vdash_s freshen\, t\, x : T_2 \;\wedge\; T = T_1{\to}T_2)\end{array}$$

Given that $\vdash_w$ and $\vdash_s$ are provably equivalent by McKinna-Pollack technology, and Γ is finitely supported, the latter of these is clearly stronger. We do not yet see how to get strong inversion principles from the strengthened induction outlined in this note.

Related Work Aydemir et. al. argue in [1] that it is more convenient to define the strong typing rule for lambda (in our notation) as:

$$\frac{\forall x \notin L.\ (x{:}T_1){::}\Gamma \vdash freshen\, t_1\, x : T_2}{\Gamma \vdash_c Lam\, t : T_1 \to T_2}$$

where L stands for a finite set of variables. (The "c" in $\vdash_c$ stands for "cofinite quantification".) We trivially have $\vdash_s \Rightarrow \vdash_c \Rightarrow \vdash_w$. $\vdash_c$ results in a strong induction principle (similar to, but weaker than $\vdash_s$), and proof of the weakening lemma for this relation goes through smoothly. More interesting, *introduction*

[8] Thanks to Julien Narboux for pointing this out.

of $\vdash_c$ is *stronger* than introduction of $\vdash_s$, so that equivalence of $\vdash_c$ and $\vdash_w$ can be proved without resorting to name permutation arguments. Alternatively, a strong introduction lemma

$$\frac{x \# t \quad (x{:}T_1){::}\Gamma \vdash_c \mathit{freshen}\ t\ x : T_2}{\Gamma \vdash_c \mathit{Lam}\ t : T_1 \to T_2}$$

can be proved without name permutation arguments. With this lemma, everything that can be proved using the equivalence of $\vdash_c$ and $\vdash_w$ can be proved without defining $\vdash_w$.

The main difference between their work and ours is that we derive an adequately strong induction principle from the weak version of the typing rules, which is preferable to an infinitely branching system such as $\vdash_c$. Further, our strengthened induction principle not only allows specifying a co-finite set to avoid (as does the principle from $\vdash_c$), but chooses a fresh name, which has to be done manually in the style of [1].

References

1. B. Aydemir, A. Charguéraud, B. C. Pierce, R. Pollack, and S. Weirich. Engineering Formal Metatheory, 2007. Submitted for publication.
2. A. Chlipala. POPLmark challenge part 1a solution. www.cs.berkeley.edu/~adamc/poplmark.
3. T. Coquand. An algorithm for testing conversion in Type Theory. In G. Huet and G. Plotkin, editors, *Logical Frameworks*. Camb. Univ. Press, 1991.
4. A. Gordon. A mechanism of name-carrying syntax up to alpha-conversion. In *Higher Order Logic Theorem Proving and its Applications. Proceedings, 1993*, LNCS 780. Springer-Verlag, 1993.
5. G. Huet. The constructive engine. In R. Narasimhan, editor, *A Perspective in Theoretical Computer Science*. World Scientific Publishing, 1989. Commemorative Volume for Gift Siromoney.
6. X. Leroy. A Locally Nameless Solution to the POPLmark Challenge. Research report 6098, INRIA, Jan. 2007.
7. J. McKinna and R. Pollack. Pure Type Systems Formalized. In *Proc. of the International Conference on Typed Lambda Calculi and Applications (TLCA)*, number 664 in LNCS, pages 289–305. Springer-Verlag, 1993.
8. J. McKinna and R. Pollack. Some Type Theory and Lambda Calculus Formalised. *Journal of Automated Reasoning*, 23(1-4), 1999.
9. A. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, (186):165–193, 2003.
10. R. Pollack. Closure under alpha-conversion. In H. Barendregt and T. Nipkow, editors, *TYPES'93: Workshop on Types for Proofs and Programs, Nijmegen, May 1993, Selected Papers*, volume 806 of *LNCS*, pages 313–332. Springer-Verlag, 1994.
11. W. Ricciotti. POPLmark challenge part 1a solution. ricciott.web.cs.unibo.it.
12. C. Urban, S. Berghofer, and M. Norrish. Barendregt's Variable Convention in Rule Inductions. In *Proc. of the 21th International Conference on Automated Deduction (CADE)*, volume 4603 of *LNAI*, pages 35–50, 2007.