Journal of Automated Reasoning manuscript No.
(will be inserted by the editor)

Appendix of
The Next 700 Challenge Problems for Reasoning with Higher-Order Abstract Syntax Representations

Part 2—A Survey

Amy P. Felty · Alberto Momigliano · Brigitte Pientka

March 19, 2014

A ORBI Specifications of Challenge Problems

We give here the `Syntax`, `Judgments`, `Rules`, `Schemas`, and `Definitions` sections of the ORBI specifications for all the benchmarks presented in **?** and formalized in the main paper and in Appendix B. The full ORBI files can be found at `https://github.com/pientka/ORBI`, and are called `EqualUntyped.orbi`, `EqualPoly.orbi`, `TypingSimple.orbi`, and `ParRed.orbi`, respectively.

A.1 Algorithmic and Declarative Equality for the Untyped Lambda-Calculus

```
%% Syntax
tm: type.
app: tm -> tm -> tm.
lam: (tm -> tm) -> tm.

%% Judgments
aeq: tm -> tm -> type.
deq: tm -> tm -> type.

%% Rules
ae_a: aeq M1 N1 -> aeq M2 N2 -> aeq (app M1 M2) (app N1 N2).
ae_l: ({x:tm} aeq x x -> aeq (M x) (N x))
        -> aeq (lam (\x. M x)) (lam (\x. N x)).

de_a: deq M1 N1 -> deq M2 N2 -> deq (app M1 M2) (app N1 N2).
de_l: ({x:tm} deq x x -> deq (M x) (N x))
```

A. P. Felty
School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada, E-mail: afelty@eecs.uottawa.ca

A. Momigliano
Dipartimento di Informatica, Università degli Studi di Milano, Italy, E-mail: momigliano@di.unimi.it

B. Pientka
School of Computer Science, McGill University, Montreal, Canada, E-mail: bpientka@cs.mcgill.ca

```
       -> deq (lam (\x. M x)) (lam (\x. N x)).
de_r: deq M M.
de_s: deq M1 M2 -> deq M2 M1.
de_t: deq M1 M2 -> deq M2 M3 -> deq M1 M3.

%% Schemas
schema xG: block (x:tm).
schema xaG: block (x:tm; u:aeq x x).
schema xdG: block (x:tm; u:deq x x).
schema daG: block (x:tm; u:deq x x; v:aeq x x).

%% Definitions
inductive xaR: {G:xG} {H:xaG} prop =
| xa_nil: xaR nil nil
| xa_cons: xaR G H -> xaR (G, block x:tm) (H, block x:tm; u:aeq x x).

inductive daR: {G:xaG} {H:xdG} prop =
| da_nil: daR nil nil
| da_cons: daR G H -> daR (G, block x:tm; v:aeq x x)
                          (H, block x:tm; u:deq x x).
```

A.2 Algorithmic Equality for the Polymorphic Lambda Calculus

```
%% Syntax
tp: type.
arr: tp -> tp -> tp.
all: (tp -> tp) -> tp.

tm: type.
app: tm -> tm -> tm.
lam: (tm -> tm) -> tm.
tapp: tm -> tp -> tm.
tlam: (tp -> tm) -> tm.

%% Judgments
atp: tp -> tp -> type.
aeq: tm -> tm -> type.

%% Rules
at_al: ({a:tp} atp a a -> atp (T a) (S a))
       -> atp (all (\a. T x) (all (\a. S a)).
at_a: atp T1 T2 -> atp S1 S2 -> atp (arr T1 S1) (arr T2 S2).
ae_l: ({x:tm} aeq x x -> aeq (M x) (N x))
       -> aeq (lam (\x. M x)) (lam (\x. N x)).
ae_a: aeq M1 N1 -> aeq M2 N2 -> aeq (app M1 M2) (app N1 N2).
ae_tl: ({a:tp} atp a a -> aeq (M a) (N a))
       -> aeq (tlam (\a. M a)) (tlam (\a. N a)).
ae_ta: aeq M N -> atp T S -> aeq (tapp M T) (tapp N S).

%% Schemas
schema aG: block (a:tp).
schema axG: block (a:tp) + block (x:tm).
schema atpG: block (a:tp; u:atp a a).
schema aeqG: block (a:tp; u:atp a a) + block (x:tm; v:aeq x x).

%% Definitions
inductive atpR: {G:aG} {H:atpG} prop =
| atp_nil: atpR nil nil
| atp_cons : atpR G H -> atpR (G, block a:tp) (H, block a:tp; u:atp a a).
```

```
inductive aeqR: {G:axG} {H:aeqG} prop =
| aeq_nil: aeqR nil nil
| aeq_cons1 : aeqR G H -> aeqR (G, block a:tp) (H, block a:tp; u:atp a a)
| aeq_cons2 : aeqR G H -> aeqR (G, block x:tm) (H, block s:tm; u:aeq x x).
```

A.3 Static Semantics of the Simply-Typed Lambda-Calculus

```
%% Syntax
tp: type.
i: tp.
arr: tp -> tp -> tp.

tm: type.
app: tm -> tm -> tm.
lam: tp -> (tm -> tm) -> tm.

%% Judgments
oft: tm -> tp -> type.

%% Rules
oft_l: ({x:tm} oft x A -> oft (M x) B) ->
       oft (lam A (\x. M x)) (arr A B).
oft_a: oft M (arr A B) -> oft N A -> oft (app M N) B.

%% Schemas
schema xtG: block (x:tp; u:oft x A).
```

A.4 Parallel Reduction for the Simply-Typed Lambda-Calculus

```
%% Syntax
tp: type.
i: tp.
arr: tp -> tp -> tp.

tm : type.
app : tm -> tm -> tm.
lam : tp -> (tm -> tm) -> tm.

%% Judgments
oft : tm -> tp -> type.
pr : tm -> tm -> type.

%% Rules
oft_l: ({x:tm} oft x T -> oft (M x) S)
       -> oft (lam T (\x. M x)) (arr T S).
oft_a: oft M1 (arr T2 T) -> oft M2 T2 -> oft (app M1 M2) T.

pr_l: ({x:tm} pr x x -> pr (M1 x) (M2 x))
       -> pr (lam T (\x. M1 x)) (lam T (\x. M2 x)).
pr_b: ({x:tm} pr x x -> pr (M1 x) (M2 x)) ->
         {T:tp} pr N1 N2 -> pr (app (lam T (\x. M1 x)) N1) (M2 N2).
pr_a: pr M1 M2 -> pr N1 N2 -> pr (app M1 N1) (app M2 N2).

%% Schemas
schema xtG: block (x:tm; v:oft x T).
schema xrG: block (x:tm; u:pr x x).
```

```
schema xrtG: block (x:tm; u:pr x x; v:oft x T).

%% Definitions
inductive xrtR: {G:xrG} {H:xtG} prop =
| xrt_nil : xrtR nil nil
| xrt_cons: xrtR G H ->
            xrtR (G, block x:tm; u:pr x x) (H, block x:tm; v:oft x A).
```

B Mechanization in Hybrid: Additional Proofs and Discussion

This appendix provides additional information that extends Section 6 of the main paper. Section B.1 includes the G version of the proof in Section 6.3 while Section B.2 provides details omitted from Section 6.6.

B.1 G Version of Completeness of Equality

The definition of $(\texttt{daG}\ \Phi_{da})$ is implemented as usual by the corresponding schema declaration in Appendix A.1, which here includes blocks of the form $(\texttt{is\_tm}\ x :: \texttt{deq}\ x\ x :: \texttt{aeq}\ x\ x)$. The contexts xaG and aG defined in Section 6.1.2 are also used here.

Note that H-Theorem 5 is stated using context Φ_{xa} and that H-Theorem 7 is stated using context Φ_a. Since we will need both theorems here, we need to promote them to Φ_{da}. As in Section 6.2.1, we need strengthening functions and a series of lemmas analogous to H-Lemmas 12-15. The strengthening functions must strengthen Φ_{da} to Φ_{xa} and Φ_a. The main clauses of these function definitions are:

$$\begin{aligned}
&\texttt{rm_da2xa}\,(\texttt{is_tm}\ z :: \texttt{deq}\ _\ _ :: \texttt{aeq}\ x\ y :: \Phi_{da}) = (\texttt{is_tm}\ z :: \texttt{aeq}\ x\ y :: \texttt{rm_da2xa}\ \Phi_{da})\\
&\texttt{rm_da2a}\,(\texttt{is_tm}\ _ :: \texttt{deq}\ _\ _ :: \texttt{aeq}\ x\ y :: \Phi_{da}) \quad = (\texttt{aeq}\ x\ y :: \texttt{rm_da2a}\ \Phi_{da})
\end{aligned}$$

Unlike in Section 6.2 where all the strengthening functions removed an alternative from a context schema, all those in this subsection involve schemas with just one alternative and modify every block by removing one or more atoms. The lemmas required to prove promotion are as follows. (The strengthening and weakening lemmas are again stated as a corollary without the lemmas they depend on.)

H-Lemma 31
1. $\texttt{daG}\ \Phi_{da} \to \texttt{xaG}\ (\texttt{rm\_da2xa}\ \Phi_{da})$.
2. $\texttt{daG}\ \Phi_{da} \to \texttt{aG}\ (\texttt{rm\_da2a}\ \Phi_{da})$.

H-Corollary 32 (C-Strengthening/Weakening)
1. $\texttt{daG}\ \Phi_{da} \to \{\Phi_{da} \vdash_n \langle \texttt{is\_tm}\ T\rangle\} \to \{(\texttt{rm\_da2xa}\ \Phi_{da}) \vdash_n \langle \texttt{is\_tm}\ T\rangle\}$.
2. $\texttt{daG}\ \Phi_{da} \to \{(\texttt{rm\_da2xa}\ \Phi_{da}) \vdash_n \langle \texttt{aeq}\ T\ T'\rangle\} \to \{\Phi_{da} \vdash_n \langle \texttt{aeq}\ T\ T'\rangle\}$.
3. $\texttt{daG}\ \Phi_{da} \to \{(\texttt{rm\_da2a}\ \Phi_{da}) \vdash_n \langle \texttt{aeq}\ T\ T'\rangle\} \longleftrightarrow \{\Phi_{da} \vdash_n \langle \texttt{aeq}\ T\ T'\rangle\}$.

With the above lemmas, we can now promote H-Theorems 5 and 7.

H-Lemma 33 (Promotion)
1. $\texttt{daG}\ \Phi_{da} \to \{\Phi_{da} \vdash_n \langle \texttt{is\_tm}\ M\rangle\} \to \{\Phi_{da} \vdash_n \langle \texttt{aeq}\ M\ M\rangle\}$.
2. $\texttt{daG}\ \Phi_{da} \to \{\Phi_{da} \vdash_n \langle \texttt{aeq}\ M\ N\rangle\} \to \{\Phi_{da} \vdash_n \langle \texttt{aeq}\ N\ M\rangle\}$.
3. $\texttt{daG}\ \Phi_{da} \to \{\Phi_{da} \vdash_n \langle \texttt{aeq}\ M\ L\rangle\} \to \{\Phi_{da} \vdash_n \langle \texttt{aeq}\ L\ N\rangle\} \to \{\Phi_{da} \vdash_n \langle \texttt{aeq}\ M\ N\rangle\}$.

H-Theorem 34 (Completeness, G Version)
$\texttt{daG}\,\Phi_{da} \to \{\Phi_{da} \vdash_n \langle \texttt{deq}\; M\;\; N\rangle\} \to \{\Phi_{da} \vdash_n \langle \texttt{aeq}\; M\;\; N\rangle\}$.

Proof The steps of the `de_r` and `de_t` cases are the same as in the R version, using promotion, height weakening, and in the latter case also the induction hypothesis. The `de_l` case also uses height weakening, and in addition requires both *d-wk* and *d-str*.[1]

B.2 Type Preservation for Parallel Reduction

We illustrate the problem mentioned in Section 6.6 of the proof of type preservation for parallel reduction in Hybrid by showing one case where the proof gets stuck. As usual, a Coq inductive definition implements context relation ($\texttt{xrtR}\,\Phi_r\;\Phi_t$) from the `Definitions` section of Appendix A.4, omitting the well-formedness annotations for terms. Since proof heights are not important in illustrating the problem, we elide them except for in the judgment that we induct over.

H-Attempt 35 (Type Preservation for Parallel Reduction)
$\texttt{xrtR}\,\Phi_r\;\Phi_t \to \{\Phi_r \vdash_m \langle \texttt{pr1}\; M\;\; N\rangle\} \to \{\Phi_t \vdash \langle \texttt{oft}\; M\;\; A\rangle\} \to \{\Phi_t \vdash \langle \texttt{oft}\; N\;\; A\rangle\}$.

Proof We attempt to follow the informal proof of Theorem 26 in **?**, here by using a complete induction on m, where we assume $i < m$ in the proof sketch below.
Case `pr_l`: We must show:

$$\frac{\texttt{h}_1 : \mathit{IH} \quad \texttt{h}_2 : \texttt{xrtR}\,\Phi_r\;\Phi_t \quad \texttt{h}_3 : \{\Phi_r \vdash_i \langle \texttt{pr1}\;(\texttt{lam}\;M')\;(\texttt{lam}\;N')\rangle\} \quad \texttt{h}_4 : \{\Phi_t \vdash \langle \texttt{oft}\;(\texttt{lam}\;M')\;A\rangle\}}{\{\Phi_t \vdash \langle \texttt{oft}\;(\texttt{lam}\;N')\;A\rangle\}}$$

Inversion on $\texttt{h}_4$ results in two subcases corresponding to the `s_bc` and `s_init` rules in Figure 11 on page 27 of the main paper, as usual.

Subcase `s_bc`: This case is similar to the `lam` case of other proofs (such as the `tm_l` case of H-Theorem 2). We show some detail to help illustrate the problem. We first apply a few more inversion steps to $\texttt{h}_3$ and $\texttt{h}_4$. We also apply SL rules `s_bc` and `s_all` in a backward direction to the conclusion, obtaining the new subgoal:

$$\frac{\mathit{IH} \quad \texttt{xrtR}\,\Phi_r\;\Phi_t \quad \texttt{h}_5 : \forall x.\texttt{proper}\,x \to \{(\texttt{pr1}\;x\;x :: \Phi_r) \vdash_{i-3} \langle \texttt{pr1}\;(M'\;x)\;(N'\;x)\rangle\} \quad \texttt{h}_6 : \forall x.\texttt{proper}\,x \to \{(\texttt{oft}\;x\;A' :: \Phi_t) \vdash \langle \texttt{oft}\;(M'\;x))\;B'\rangle\}}{\forall x.\texttt{proper}\,x \to \{\Phi_t \vdash ((\texttt{oft}\;x\;A')\;\texttt{imp}\;\langle \texttt{oft}\;(N'\;x)\;B'\rangle)\}}$$

Applying Coq's $\forall$-introduction at this point introduces a new x, which we use to instantiate $\texttt{h}_5$ and $\texttt{h}_6$ and complete this case.

Subcase `s_init`: We have the following subgoal:

$$\frac{\mathit{IH} \quad \texttt{xrtR}\,\Phi_r\;\Phi_t \quad \texttt{h}_5 : \{\Phi_r \vdash_i \langle \texttt{pr1}\;(\texttt{lam}\;M')\;(\texttt{lam}\;N')\rangle\} \quad \texttt{h}_6 : (\texttt{oft}\;(\texttt{lam}\;M')\;A) \in \Phi_t}{\{\Phi_t \vdash \langle \texttt{oft}\;(\texttt{lam}\;N')\;A\rangle\}}$$

[1] Also in this proof and a few others (e.g., H-Theorem 26 and H-Attempt 35), the inversion step uses specialized inversion lemmas, whose proofs follow from Coq's standard inversion.

which is not provable. In order to prove it, we would need a lemma similar to H-Lemma 27, which requires restricting variables in contexts to be of the form (`VAR` i). Here, from such a lemma and $\texttt{h}_6$, we could derive a contradiction as desired. This kind of variable restriction would have to be built into the definition of `xrtR`. Then this subcase becomes provable, but the previous one can no longer be proved. This is because the variable x introduced by ∀-introduction in that proof is an arbitrary term of type `expr`. Since it doesn't necessarily have the form (`VAR` i), the context relation would not hold, and thus it would not be possible to apply the induction hypothesis.

Fixing the above problem is discussed in Sections 6.6 and 8.2 in the main paper.