Equivalence of System F and $\lambda 2$: A Case Study of Context Morphisms

— Extended Abstract —

Jonas Kaiser, Tobias Tebbi, and Gert Smolka

Abstract. We give a machine-checked proof of the equivalence of the usual, two-sorted presentation of System F and its single-sorted pure type system variant $\lambda 2$. This is established by reducing the typing problem of F to $\lambda 2$ and vice versa. The difficulty lies in aligning different binding-structures and different contexts (dependent vs. non-dependent). The use of de Bruijn syntax, parallel substitutions, and context morphism lemmas leads to a surprisingly elegant proof. We use the Coq proof assistant and the substitution library AUTOSUBST.

1 Introduction

There are different presentations of "System F" in the literature, and we often found that properties that have been proven for presentation X are assumed for presentation Y, without verifying that the presentations are in fact equivalent. As it turns out, such an equivalence proof, if done in full formal detail, is surprisingly intricate. The arising proof obligations become particularly involved when the syntax of one presentation has to be translated to that of another. Our goal is to demonstrate that a pure de Bruijn setup with parallel substitutions and the systematic use of context morphism lemmas leads to an elegant treatment of the hidden complexities. Parallel substitutions (i.e. multi-variable substitutions) lead to clear and natural lemma statements [7], contradicting the usual criticism [2] of de Bruijn formalisations.

Here we consider the usual two-sorted presentation, as given by Harper in [6], and a pure type system (PTS) variant $\lambda 2$, closely related to the one found in Barendregt's λ-cube [3]. The respective type systems are given in Figures 1 and 2. Further details are given in Section 2.

In his doctoral thesis, Geuvers [4] presents a proof sketch that valid typing judgements can be translated between these two presentations. Apart from that we are not aware of other treatments of this correspondence. We improve upon Geuvers' result in two ways. First, we also translate invalid typing judgements to invalid typing judgements, thus obtaining full computational reductions for the decision problem of typability. Second, our proof is completely formalised with the Coq proof assistant.[1]

Our equivalence result defines syntactic translations $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ and asserts the following equivalences:

$$(1)\qquad \vdash_{\mathrm{F}} s : A \quad\text{iff}\quad \vdash_{\lambda 2} \lfloor s\rfloor : \lfloor A\rfloor \qquad\qquad (2)\qquad \vdash_{\lambda 2} a : b \quad\text{iff}\quad \vdash_{\mathrm{F}} \lceil a\rceil : \lceil b\rceil$$

Note that the systems differ in two main aspects. First, F syntactically distinguishes terms and types, while $\lambda 2$ only possesses a single sort of terms. In addition, their inference systems

[1] The accompanying formalisation can be found at `https://www.ps.uni-saarland.de/extras/hor16/`.

$$A, B, C \;:=\; x_{\mathsf{ty}} \mid A \to B \mid \forall . A \qquad s, t \;:=\; x_{\mathsf{ter}} \mid s\, t \mid \lambda A. s \mid s\, A \mid \Lambda. s \qquad x \in \mathbb{N}$$

$$\frac{x < N}{N \vdash^{\mathsf{ty}}_{\mathsf{F}} x_{\mathsf{ty}}} \qquad \frac{N \vdash^{\mathsf{ty}}_{\mathsf{F}} A \qquad N \vdash^{\mathsf{ty}}_{\mathsf{F}} B}{N \vdash^{\mathsf{ty}}_{\mathsf{F}} A \to B} \qquad \frac{(N+1) \vdash^{\mathsf{ty}}_{\mathsf{F}} A}{N \vdash^{\mathsf{ty}}_{\mathsf{F}} \forall . A}$$

$$\frac{A_x = A \qquad N \vdash^{\mathsf{ty}}_{\mathsf{F}} A}{N; A_n, \ldots, A_0 \vdash^{\mathsf{ter}}_{\mathsf{F}} x_{\mathsf{ter}} : A} \qquad \frac{N; \Gamma \vdash^{\mathsf{ter}}_{\mathsf{F}} s : A \to B \qquad N; \Gamma \vdash^{\mathsf{ter}}_{\mathsf{F}} t : A}{N; \Gamma \vdash^{\mathsf{ter}}_{\mathsf{F}} s\, t : B} \qquad \frac{N; \Gamma, A \vdash^{\mathsf{ter}}_{\mathsf{F}} s : B \qquad N \vdash^{\mathsf{ty}}_{\mathsf{F}} A}{N; \Gamma \vdash^{\mathsf{ter}}_{\mathsf{F}} \lambda A. s : A \to B}$$

$$\frac{N; \Gamma \vdash^{\mathsf{ter}}_{\mathsf{F}} s : \forall . A \qquad N \vdash^{\mathsf{ty}}_{\mathsf{F}} B}{N; \Gamma \vdash^{\mathsf{ter}}_{\mathsf{F}} s\, B : A[B \cdot \mathsf{id}]} \qquad \frac{(N+1); \Gamma[+1] \vdash^{\mathsf{ter}}_{\mathsf{F}} s : A}{N; \Gamma \vdash^{\mathsf{ter}}_{\mathsf{F}} \Lambda. s : \forall . A}$$

Figure 1: System F – two-sorted de Bruijn syntax, type formation and typing rules.

$$a, b, c, d \;:=\; u \mid x \mid a\, b \mid \lambda a. b \mid \Pi a. b \qquad u \in \{*, \Box\} \qquad x \in \mathbb{N}$$

$$\frac{}{0 : s[+1] \;\in\; \Gamma, s} \qquad \frac{x : s \;\in\; \Gamma}{(x+1) : s[+1] \;\in\; \Gamma, t} \qquad \frac{}{\Gamma \vdash_2 * : \Box} \qquad \frac{x : a \in \Gamma \qquad \Gamma \vdash_2 a : u}{\Gamma \vdash_2 x : a}$$

$$\frac{\Gamma \vdash_2 a : u \qquad \Gamma, a \vdash_2 b : *}{\Gamma \vdash_2 \Pi a. b : *} \qquad \frac{\Gamma \vdash_2 a : \Pi c. d \qquad \Gamma \vdash_2 b : c}{\Gamma \vdash_2 a\, b : d[b \cdot \mathsf{id}]} \qquad \frac{\Gamma, a \vdash_2 b : c \qquad \Gamma \vdash_2 a : u \qquad \Gamma, a \vdash_2 c : *}{\Gamma \vdash_2 \lambda a. b : \Pi a. c}$$

Figure 2: PTS – single-sorted de Bruijn syntax, dependent context lookup, and typing rules of $\lambda 2$.

exhibit a number of structural differences. We are going to treat these issues separately by decomposing the above equivalence result into two parts.

First we will introduce an auxiliary type system P (Figure 3) for the PTS syntax that matches the inference structure of F and is equivalent to $\lambda 2$. We then proceed to prove the equivalence of F and P under the given syntax translations. This latter part is involved as the translations $\lceil \cdot \rceil$ from the PTS syntax into the two sorts of F are necessarily partial.

We found that systematically phrasing all essential statements for both parts of the proof as context morphism lemmas [5, 1] leads to a clear and elegant formalisation.

In summary, our contributions are:

- A proof of the equivalence, via the computational reduction of the decision problem of typability, of two common presentations of F that is fully machine-checked with the proof assistant Coq.
- A case-study illustrating that working with pure de Bruijn syntax and parallel substitutions is not only possible but desirable as it leads to elegant theorem statements and proofs.
- The observation that context morphisms are a powerful tool for relating syntaxes and type systems with different binding structures.

2 Parallel Substitutions and Context Morphisms

We are treating syntax in a formal setting and hence opt for a pure de Bruijn presentation throughout this work. Instead of using names, **variables** are represented as indices, that is, natural numbers that designate their corresponding (nameless) binder, counting upwards in the

$$\frac{x : * \in \Gamma}{\Gamma \vdash^{\mathsf{ty}}_{\mathsf{P}} x} \qquad \frac{\Gamma \vdash^{\mathsf{ty}}_{\mathsf{P}} a \qquad \Gamma, a \vdash^{\mathsf{ty}}_{\mathsf{P}} b}{\Gamma \vdash^{\mathsf{ty}}_{\mathsf{P}} \Pi a.\, b} \qquad \frac{\Gamma, * \vdash^{\mathsf{ty}}_{\mathsf{P}} a}{\Gamma \vdash^{\mathsf{ty}}_{\mathsf{P}} \Pi *.\, a}$$

$$\frac{x : a \in \Gamma \qquad \Gamma \vdash^{\mathsf{ty}}_{\mathsf{P}} a}{\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}} x : a} \qquad \frac{\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}} a : \Pi c.\, d \qquad \Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}} b : c}{\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}} a\, b : d[b \cdot \mathsf{id}]} \qquad \frac{\Gamma, a \vdash^{\mathsf{ter}}_{\mathsf{P}} b : c \qquad \Gamma \vdash^{\mathsf{ty}}_{\mathsf{P}} a}{\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}} \lambda a.\, b : \Pi a.\, c}$$

$$\frac{\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}} a : \Pi *.\, b \qquad \Gamma \vdash^{\mathsf{ty}}_{\mathsf{P}} c}{\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}} a\, c : b[c \cdot \mathsf{id}]} \qquad \frac{\Gamma, * \vdash^{\mathsf{ter}}_{\mathsf{P}} a : b}{\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}} \lambda *.\, a : \Pi *.\, b}$$

Figure 3: PTS – auxiliary type system P.

syntax tree.

Parallel **substitutions** (written σ, τ) are functions from variables to terms. A variable x is **free** in a term s if $x + k$ occurs in s under k binders. We write $s[\sigma]$ for s with every free variable x being replaced by $\sigma(x)$. The **composition** $\sigma \circ \tau$ is defined as $(\sigma \circ \tau)(x) := \sigma(x)[\tau]$. We also frequently encounter the **identity substitution** id, defined as $\mathsf{id}(x) := x$, and the **shift substitution** $+1$, defined as $+1(x) := x+1$, both of which also constitute examples of **renamings**[2] (written ξ, ζ), that is, substitutions that map variables to variables. Note that a substitution σ can be viewed as the infinite stream $\sigma(0), \sigma(1), \ldots$ of terms. This motivates the definition of a **cons** operation $s \cdot \sigma = s, \sigma(0), \sigma(1), \ldots$, which allows us to succinctly handle beta reduction: the term $(\lambda.\, s)t$ reduces to $s[t \cdot \mathsf{id}]$. The renaming $-1 := 0 \cdot \mathsf{id}$ is a right inverse of $+1$. For a detailed treatment of the underlying theory and its implementation in the AUTOSUBST library, see [7].

Working with parallel substitutions can be seen as a "big-step" approach, a phrase coined by Adams in [1], since they act on all variables at once. The approach avoids an auxiliary shift operation on terms, lemmas generalised to add a type in the middle of a context and other ad-hoc techniques contributing to the bad reputation of de Bruijn indices.

A substitution σ is a **context morphism**[3] from a context Γ to a context Δ for a judgement $\vdash$ if it maps variable judgements under the context Γ into judgements under σ and under the context Δ:

$$\vdash \sigma : \Gamma \to \Delta \;:=\; \forall x, t.\;\; \Gamma \vdash x : t \;\to\; \Delta \vdash x[\sigma] : t[\sigma]$$

If we consider **weakening** – below left in its usual de Bruijn presentation – as a "small step" lemma, in that it only changes the context in a single position, we can formulate a **context morphism lemma** as its "big step" variant – below right – that has the potential to change the complete context. Weakening is then a corollary of the context morphism lemma, using the context morphism $\vdash +1 : \Gamma \to \Gamma, u$.

$$\frac{\Gamma \vdash s : t}{\Gamma, u \vdash s[+1] : t[+1]} \qquad \frac{\Gamma \vdash s : t \qquad \vdash \sigma : \Gamma \to \Delta}{\Delta \vdash s[\sigma] : t[\sigma]}$$

A context morphism is a typed substitution and the context morphism lemma is a substitutivity property of the typing judgement. Context morphism lemmas admit direct proofs by induction on the typing derivation.

[2] Note that we do not require renamings to be injective or surjective. For example, $+1$ is not surjective.

[3] Context morphisms are written as $\sigma : \Delta \longrightarrow \Gamma$ (note the inverted order of contexts) in [5] and correspond to Adams' [1] satisfaction relation $\Delta \vDash \sigma :: \Gamma$.

$$
\begin{array}{rlrll}
\lfloor x_{\mathsf{ty}} \rfloor &:= x & \lceil x \rceil^{\Gamma}_{\mathsf{ty}} &:= x_{\mathsf{ty}} & \text{if } x : * \in \Gamma \\
\lfloor A \to B \rfloor &:= \Pi \lfloor A \rfloor .\, \lfloor B \rfloor [\uparrow 1] & \lceil \Pi *.\, a \rceil^{\Gamma}_{\mathsf{ty}} &:= \forall .\, \lceil a \rceil^{\Gamma,*}_{\mathsf{ty}} & \\
\lfloor \forall .\, A \rfloor &:= \Pi *.\, \lfloor A \rfloor & \lceil \Pi a.\, b \rceil^{\Gamma}_{\mathsf{ty}} &:= \lceil a \rceil^{\Gamma}_{\mathsf{ty}} \to \lceil b \rceil^{\Gamma,a}_{\mathsf{ty}} [\uparrow 1] & \\
\lfloor x_{\mathsf{ter}} \rfloor &:= x & \lceil x \rceil^{\Gamma}_{\mathsf{ter}} &:= x_{\mathsf{ter}} & \text{for some } a \neq * \text{ with } x : a \in \Gamma \\
\lfloor s\, t \rfloor &:= \lfloor s \rfloor \, \lfloor t \rfloor & \lceil a\, b \rceil^{\Gamma}_{\mathsf{ter}} &:= \lceil a \rceil^{\Gamma}_{\mathsf{ter}} \, \lceil b \rceil^{\Gamma}_{\mathsf{ty}} & \\
\lfloor \lambda A.\, s \rfloor &:= \lambda \lfloor A \rfloor .\, \lfloor s[\uparrow 1, \mathsf{id}] \rfloor & \lceil a\, b \rceil^{\Gamma}_{\mathsf{ter}} &:= \lceil a \rceil^{\Gamma}_{\mathsf{ter}} \, \lceil b \rceil^{\Gamma}_{\mathsf{ter}} & \\
\lfloor s\, A \rfloor &:= \lfloor s \rfloor \, \lfloor A \rfloor & \lceil \lambda *.\, a \rceil^{\Gamma}_{\mathsf{ter}} &:= \Lambda .\, \lceil a \rceil^{\Gamma,*}_{\mathsf{ter}} [\mathsf{id}, \uparrow 1] & \\
\lfloor \Lambda .\, s \rfloor &:= \lambda *.\, \lfloor s[\mathsf{id}, \uparrow 1] \rfloor & \lceil \lambda a.\, b \rceil^{\Gamma}_{\mathsf{ter}} &:= \lambda \lceil a \rceil^{\Gamma}_{\mathsf{ty}} .\, \lceil b \rceil^{\Gamma,a}_{\mathsf{ter}} [\uparrow 1, \mathsf{id}] &
\end{array}
$$

Figure 4: Syntax translations.

3 The Equivalence Proof

The proof of the equivalence of λ2 and P is straightforward if propagation ($\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}} s : t \;\to\; \Gamma \vdash^{\mathsf{ty}}_{\mathsf{P}} t$) holds for P. Propagation follows from substitutivity of type formation, which is relatively easy to prove when phrased as a context morphism lemma. We omit further details here in favour of the more involved equivalence proof of F and P.

As the syntaxes of F and P are different, we have to introduce translations $\lfloor A \rfloor$ and $\lfloor s \rfloor$ from F-types and terms to PTS terms, as well as the necessarily partial translations $\lceil a \rceil^{\Gamma}_{\mathsf{ty}}$ and $\lceil a \rceil^{\Gamma}_{\mathsf{ter}}$ that try to construct, from a PTS term a, an F-type or, respectively, an F-term. The latter translations depend on a context Γ to disambiguate type and term variables. To avoid clutter, we omit side conditions asserting that a given translation result is defined. The full definition is given in Figure 4. Note that if b appears in a valid PTS judgement under Γ, then exactly one of $\lceil b \rceil^{\Gamma}_{\mathsf{ter}}$ and $\lceil b \rceil^{\Gamma}_{\mathsf{ty}}$ is defined. This explains the two defining equations for $\lceil a\, b \rceil^{\Gamma}_{\mathsf{ter}}$. We further observe that terms of F contain both term and type variables, so we write $s[\tau, \sigma]$ for the parallel application of a type substitution τ and a term substitution σ.

Theorem 1 *The typing problem under the empty context $\langle\rangle$ can be reduced from* P *to* F *and vice versa.*

$$(1) \qquad \vdash^{\mathsf{ter}}_{\mathsf{F}} s : A \quad \textit{iff} \quad \vdash^{\mathsf{ter}}_{\mathsf{P}} \lfloor s \rfloor : \lfloor A \rfloor \qquad\qquad (2) \qquad \vdash^{\mathsf{ter}}_{\mathsf{P}} a : b \quad \textit{iff} \quad \vdash^{\mathsf{ter}}_{\mathsf{F}} \lceil a \rceil^{\langle\rangle}_{\mathsf{ter}} : \lceil b \rceil^{\langle\rangle}_{\mathsf{ty}}$$

To prove this theorem, we need to generalise to non-empty contexts. This requires relating equivalent contexts. We do this using context morphisms generalised to work between different judgements. For example:

$$\xi, \zeta : (\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}}) \to (N; \Delta \vdash^{\mathsf{ter}}_{\mathsf{F}}) := \forall x a.\;\; \Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}} x : a \;\to\; N; \Delta \vdash^{\mathsf{ter}}_{\mathsf{F}} \zeta(x) : \lceil a \rceil^{\Gamma}_{\mathsf{ty}} [\xi]$$

The overall proof involves several such morphisms, which are all defined analogously. With context morphisms, stating an inductively provable version of the fact that the translation preserves typing becomes a matter of routine.

Lemma 2 (Preservation of Typing under Translations)

$$\frac{N; \Delta \vdash^{\mathsf{ter}}_{\mathsf{F}} s : A \qquad \xi : (N \vdash^{\mathsf{ty}}_{\mathsf{F}}) \to (\Gamma \vdash^{\mathsf{ty}}_{\mathsf{P}}) \qquad \xi, \zeta : (N; \Delta \vdash^{\mathsf{ter}}_{\mathsf{F}}) \to (\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}})}{\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}} \lfloor s[\xi, \zeta] \rfloor : \lfloor A \rfloor [\xi]}$$

$$\frac{\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}} a : b \qquad \xi : (\Gamma \vdash^{\mathsf{ty}}_{\mathsf{P}}) \to (N \vdash^{\mathsf{ty}}_{\mathsf{F}}) \qquad \xi, \zeta : (\Gamma \vdash^{\mathsf{ter}}_{\mathsf{P}}) \to (N; \Delta \vdash^{\mathsf{ter}}_{\mathsf{F}})}{N; \Delta \vdash^{\mathsf{ter}}_{\mathsf{F}} \lceil a \rceil^{\Gamma}_{\mathsf{ter}} [\xi, \zeta] : \lceil b \rceil^{\Gamma}_{\mathsf{ty}} [\xi]}$$

The proof relies on the substitutivity of the translation functions, which is also proven using context morphisms. Lemma 2 yields the forward direction of Theorem 1 as a corollary. For the backwards direction, we additionally need the fact that our translations are partial inverses of each other.

Lemma 3 (Cancellation of Term Translations)

$$\frac{\lceil a[\xi] \rceil^{\Gamma}_{\mathsf{ter}} = s}{\lfloor s \rfloor = a[\xi]} \qquad \frac{\Gamma \vDash^{\mathsf{ter}}_{\mathsf{P}} \lfloor s[\xi, \zeta] \rfloor : c \qquad \Gamma \vDash_{\mathsf{P}} \xi \parallel \zeta}{\lceil \lfloor s[\xi, \zeta] \rfloor \rceil^{\Gamma}_{\mathsf{ter}} = s[\xi, \zeta]}$$

The left rule, PFP-cancellation, is proven without assumptions by induction on a, generalising over all renamings to accommodate for the renamings in the definition of the translation. The right rule, FPF-cancellation, is more involved, because we need to make sure that no structure was lost when translating to PTS syntax. We assert a P-typing, but this is not enough. We also have to forbid reinterpretation of variables by enforcing that ξ maps only to type variables and ζ maps only to term variables according to Γ, written $\Gamma \vDash_{\mathsf{P}} \xi \parallel \zeta$.

4 Conclusions

It has been observed [1, 7] that context morphism lemmas provide elegant proofs for properties of a single type system. In a single type system, the changes to the context in each step of a type derivation are small and localised. Thus working without context morphisms is tolerable. In contrast to this, modelling the precise relationship between a two-sorted context and a single-sorted dependent context results in an explosion of complexity that quickly becomes unmanageable, as we had to painfully experience. Context morphism lemmas, applied in a systematic manner, have provided substantial relief. They serve as a template to state provable lemmas and avoid the construction of monolithic, fragile and design-dependent invariants.

References

[1] Robin Adams. Formalized metatheory with terms represented by an indexed family of types. In *Types for Proofs and Programs*, volume 3839 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2006.

[2] Brian E. Aydemir, Arthur Charguéraud, Benjamin C. Pierce, Randy Pollack, and Stephanie Weirich. Engineering formal metatheory. In *POPL*, pages 3–15. ACM, 2008.

[3] Henk Barendregt. Introduction to generalized type systems. *Journal of Functional Programming*, 1(2):125–154, 1991.

[4] Jan Herman Geuvers. *Logics and type systems*. PhD thesis, Katholieke Universiteit Nijmegen, 1993.

[5] Healfdene Goguen and James McKinna. Candidates for substitution. *LFCS report series - Laboratory for Foundations of Computer Science ECS LFCS*, 1997.

[6] Robert Harper. *Practical foundations for programming languages*. Cambridge University Press, 2013.

[7] Steven Schäfer, Tobias Tebbi, and Gert Smolka. Autosubst: Reasoning with de Bruijn terms and parallel substitutions. In *Interactive Theorem Proving, Proceedings*, volume 9236 of *Lecture Notes in Computer Science*, pages 359–374. Springer, 2015.