Barendregt's Theory of the λ-Calculus, Refreshed and Formalized

Adrienne Lancelot
Inria & LIX, Ecole Polytechnique, UMR 7161, Paris, France
Université Paris Cité, CNRS, IRIF, F-75013, Paris, France

Beniamino Accattoli
Inria & LIX, Ecole Polytechnique, UMR 7161, Paris, France

Maxime Vemclefs
Independent, Paris, France

Abstract

Barendregt's book on the untyped λ-calculus refines the inconsistent view of β-divergence as representation of the undefined via the key concept of head reduction.

In this paper, we put together recent revisitations of some key theorems laid out in Barendregt's book, and we formalize them in the Abella proof assistant. Our work provides a compact and refreshed presentation of the core of the book.

The formalization faithfully mimics pen-and-paper proofs. Two interesting aspects are the manipulation of contexts for the study of contextual equivalence and a formal alternative to the informal trick at work in Takahashi's proof of the *genericity lemma*. As a by-product, we obtain an alternative definition of contextual equivalence that does not mention contexts.

2012 ACM Subject Classification Theory of computation → Lambda calculus

Keywords and phrases lambda-calculus, head reduction, equational theory

Digital Object Identifier 10.4230/LIPIcs.ITP.2025.13

Supplementary Material *Software (Abella formalization code)*:
`https://github.com/adrilancelot/Abella-lambda-Barendregt-theory` [40]
archived at `swh:1:dir:b20ffd2d8d946adac1eb2fffa72112d23a2deeed`

1 Introduction

Barendregt's 1984 classic book [15] still is, today, the main reference for the untyped calculus. Its theme is the construction of the semantics of the λ-calculus induced by partial recursive functions. The crucial concept is the one of head (β-)reduction, the importance of which emerged in the 1970s in studies by Barendregt [14, 16] and Wadsworth [64, 65]. In this paper, we collect simplifications of key results of that theory, to give a new compact, refreshed presentation, backed by formal proofs.

Barendregt's book is a heavy monography, often too technical for beginners. Additionally, along the years typed λ-calculi have stolen the spotlight, with the result that nowadays the untyped λ-calculus is often seen as a topic of the past. Therefore, our work is also an attempt to preserve the beautiful but somewhat endangered theory of the untyped λ-calculus, making it accessible to a wider and younger audience, hopefully as a blueprint for new studies.

Partial Recursive Functions. Partial recursive functions (shortened to PRFs) are based on the undefined value $\bot$, that induces the extensional preorder $f \leq_{\text{PRF}} g$ holding when $\forall n \in \mathbb{N},\ f(n) = \bot$ or $f(n) =_{\mathbb{N}} g(n)$, having as minimum the everywhere undefined function $f_\bot(n) \coloneqq \bot$.

PRFs and the λ-calculus can represent each other. Moving to the λ-calculus provides dynamic computations, which yield different syntactic implementations of functions that – extensionally – do the same thing. Barendregt's theory of the λ-calculus is based on the idea of studying the semantics – that is, the equational theory – induced on λ-terms by the representation of the extensional order on PRFs.

Meaningless Terms and Equational Theories. One of the first choices to make is how to represent *undefinedness* [64, 14, 16]. Intuitively, it corresponds to divergence. In the untyped λ-calculus, the paradigmatic never-ending computations is $\Omega \coloneqq (\lambda x.xx)(\lambda x.xx)$, that rewrites to itself in one β-step. It is clear that Ω is a representation of the everywhere undefined function $f_\perp$. What is less clear is the criterion that establishes when a given term is *meaningless*, that is, when it represents the undefined.

Identifying meaningless terms goes hand in hand with the study of program equivalences and equational theories: which λ-terms should be considered equivalent? A good criterion to validate a notion of meaningless term is *collapsibility*, that is, whether the equational theories that identify all those terms are *consistent*, that is, do not equate all terms.

A Natural and Inconsistent Equational Theory. It is natural to define as meaningless all the β-divergent terms. It turns out, however, that they are not collapsible: equational theories identifying β-divergent terms are inconsistent; this is the starting point of Barendregt's theory. We provide a formalization of the simple standard proof of this fact, in Sect. 3. An intuition as to why this identification cannot work is that fixpoint combinators, that allow the encoding of recursion (and the minimization of PRFs) in the λ-calculus, are all β-divergent.

Head Reduction to the Rescue. Barendregt and Wadsworth hence introduced a notion of meaninglessness based on divergence for a sub-reduction of β, namely *head reduction*. Restricting the rewriting rule restricts the set of meaningless term. Going back to our example, fixpoint combinators terminate for head reduction, while Ω does not.

A number of results in Barendregt's book focus on showing that the choice of head divergent as meaningless is a good one. In particular, he gives two proofs of their collapsibility, one considering $\mathcal{H}$, the minimal equational theory identifying head divergent terms, and one considering (head) contextual equivalence $\simeq_{\mathcal{C}}$ (also known as $\mathcal{H}^*$). Both collapsibility proofs are based on the crucial *genericity lemma*. For $\simeq_{\mathcal{C}}$, he also proves *maximality*: $\simeq_{\mathcal{C}}$ is the maximal consistent equational theory identifying head divergent terms.

Refreshing Barendregt's Theory. The goal of the paper is to prove the collapsibility theorem, moderninizing and simplyfing the theory leading to it. For that, we choose the proof of collapsibility based on contextual equivalence, given the prominence that this concept has – along the decades – achieved in the study of programming languages. Additionally, we also prove the maximality theorem for $\simeq_{\mathcal{C}}$. Along the way, we formalize many theorems of Barendregt's theory (confluence, standardization, head factorization, head normalization, solvability, genericity, and $\simeq_{\mathcal{C}}$ is an equational theory), but do so mostly without using their original proofs in the book [15]. We use several revisitations by Accattoli et al. [8], Takahashi [57], and Accattoli and Lancelot [9] that allow for easier, self-contained proofs.

We also adopt a different, strictly more general definition of head reduction – following Accattoli and Dal Lago [7] – that however validates the same properties with the same proofs. Additionally, we deal with the contextual *preorder* $\precsim_{\mathcal{C}}$ rather than with contextual *equivalence*, and more generally deal with *in*equational theories. Lastly, our formalization work contributes some revisitations of its own, mentioned below.

The Formalization. For the formalization, we rely on the Abella proof assistant [28, 12]. Our formal proofs faithfully follow the pen-and-paper ones in the cited papers. This is partly due to Abella's primitive support for binders and for Miller and Tiu's ∇ *(nabla) quantifier* [46, 29], that allows a smooth treatment of free variables, but it is also due to the careful details provided in the neat pen-and-paper proofs we try to mimic.

The only point where there is a gap between the formalization and pen-and-paper reasoning is the treatment of contexts, in particular for contextual equivalence. Plugging in contexts is a variant of meta-level substitution that can capture variables, and that it is *not* primitively supported by Abella. Therefore, we represent contexts indirectly, via sub-term predicates: we represent $C\langle t\rangle$ saying that t is a sub-term of $u = C\langle t\rangle$, see sections 6 and 9.

Formalizing Genericity. For the theory, the main contribution of our formalization concerns the genericity lemma. In a nutshell, genericity states that the contextual closure is monotone (with respect to the contextual preorder $\precsim_{\mathcal{C}}$) on meaningless terms, *i.e.* if $\bot \precsim_{\mathcal{C}} t$ then $C\langle\bot\rangle \precsim_{\mathcal{C}} C\langle t\rangle$. While the statement is natural, its proofs are usually involved; it rather deserves to be referred to as *theorem*. Barendregt originally used a topological argument [15, Prop. 14.3.24] but other arguments exist in the literature: via intersection types by Ghilezan [31], rewriting-based ones by Takahashi [57], Kuper [39], Kennaway et al. [37], and Endrullis and de Vrijer [25], and via Taylor expansion by Barbarossa and Manzonetto [13].

We follow Takahashi's rewriting-based technique – and, more precisely, its revisited presentation by Accattoli and Lancelot [9] – because it is self-contained and relies on other theorems we already prove (the head normalization one). To formalize Takahashi's argument, we isolate two new concepts: a *disentangling* lemma (Lemma 14) turning context plugging into meta-level substitutions up to β-reduction, and a corollary of the head normalization theorem (Cor. 12) that encapsulates the rewriting argument in the proof of genericity.

As an high-level by-product, we show that the disentangling lemma in fact leads to a new notion of *substitutions equivalence* that is equivalent to contextual equivalence and does not mention contexts – it is the distilled essence of Takahashi's argument. We believe it to be an interesting contribution, perhaps the main take-away from our formalization effort.

Related Work. Formalizations of the λ-calculus abound. There are countless formalizations of confluence (e.g. [49, 54, 56, 52, 35, 63, 34, 48, 62, 66]), and some go further, dealing with standardization (e.g. McKinna and Pollack [43], Norrish [50], Crary [22], Guidi [33], Gheri and Popescu [30]) but we are not aware of any previous work dealing with the more advanced theorems we treat here (such as solvability, genericity, collapsibility, maximality). Larchey-Wendling formalizes in Coq good portions of Krivine's book on the λ-calculus [38]; the Coq sources are available [41], but there is no associated paper. Norrish formalizes in HOL4 the relationship between PRFs and the λ-calculus [51].

Independently and in parallel to us, Norrish and Tian [59] formalized in HOL4 both the operational characterization of solvability (that we also formalize) and a restricted form of Böhm's separation theorem. Böhm's theorem is a very challenging result to mechanize; Norrish and Tian's development is the first formalized proof of a separation result. Their work appears in the very same conference of this paper.

There are also many formalizations of techniques for program equivalence: e.g., for coinductive operational program equivalences in Coq by Biernacki et al. [18] and in Abella by Momigliano [47], equational theories for call-by-(push-)value in Coq by Rizkallah et al. [55], and for call-by-push-value in Coq (and partially in Abella) by Forster et al. [26, 27].

Defined in 01-lambda_terms_and_beta.thm.

```
Kind  tm   type.

Type  abs  (tm -> tm) -> tm.
Type  app  tm -> tm -> tm.
```

```
Define tm : tm -> prop by
  nabla x, tm x;
  tm (abs T) := nabla x, tm (T x);
  tm (app T U) := tm T /\ tm U.
```

Figure 1 λ-terms and the predicate for inducting on them in Abella.

Our formal revisited results for the λ-calculus are similar in spirit to those in Accattoli [4] (cube property) and Accattoli et al. [5] (translation of λ into the π-calculus).

Abella Sources. They can be found on GitHub [40].

2 Formalizing the λ-Calculus in Abella

This section overviews the adopted approach to formalize the λ-calculus in Abella.

Key Features of Abella. Abella [28, 12] belongs to the family of proof assistants based on higher-order abstract formalisms – namely, the one adopted by Abella is Miller and Nadathur's λ-*tree syntax* [45] – which provides primitive support for binders and meta-level substitution. We assume basic familiarity with these formalisms. A survey by Kaiser et al. compares Coq, Abella, and Beluga by studying System F in the three frameworks [36].

Abella has two layers, the *specification level* and the *reasoning level*. They are based on different logics, the reasoning level being more powerful and provided with special tactics to reason about the specification level. In particular, it is only at the reasoning level that one can use Miller and Tiu's ∇ *(nabla) quantifier* [46, 29], that complements the primitive support for binders with a smooth treatment of free variables.

Reasoning Level. In many formalizations in Abella, definitions are given at the specification level while statements and proofs are given at the reasoning level. In particular, this is often done when formalizing *typed* calculi, since Abella provides some tactics for the involved typing contexts at the specification level. We follow another approach, giving the definitions at the reasoning level, which is sometimes preferred when formalizing untyped calculi. One of the reasons is that in this way we can exploit ∇ to formalize terms with free variables, obtaining definitions, statements, and reasoning that are closer to those with pen-and-paper. The same approach is used also (at least) by section 7.3 of the Abella tutorial by Baelde et al. [12], Tiu and Miller [60], Accattoli [4], Chaudhuri et al. [21], and Accattoli et al. [5].

λ-Terms and Induction on Types in Abella. On paper, λ-terms are defined inductively, starting from a set of variables and via the abstraction and the application constructors:

$$\textsc{Terms}\quad t, u \quad ::= \quad x \mid \lambda x.t \mid tu$$

On paper, we shall follow the standard conventions that application is left associative and has precedence over abstractions, so that $\lambda x.tus$ denotes $\lambda x.((tu)s)$.

In Abella, it is standard to define λ-terms via a type `tm` and two constructors `app` and `abs` for applications and abstractions, without specifying variables, as in Fig. 1 (left).

Note that `abs` takes an argument of type `tm` → `tm` which is how Abella encodes binders. More precisely, an object-level binding constructor such as $\lambda x.t$ is encoded via a pair: an ordinary constructor `abs` applied to a *meta-level abstraction* of type `tm` → `tm`. For example,

the term $\lambda x.xx$ that binds x in the scope xx is encoded as `abs x\ app x x` (that is parsed as `abs (x\ app x x)`) where `x\ app x x` is a meta-level abstraction of type `tm` $\to$ `tm` in the syntax of Abella. In the rest of the paper, with an abuse of terminology, we call *binders* such terms of type `tm` $\to$ `tm`.

Reasoning by induction on the structure of `tm` terms is not possible in Abella because of the *open world assumption*, stating that new constructors can always be added later to any type. Thus, one rather defines an additional `tm` predicate, as in Fig. 1 (right), which is added to the hypotheses of statements whenever one needs to reason by induction on terms. The first clause of the `tm` predicate uses nabla to say that the free variable `x` is a term. Variables with capitalized names in the last two clauses are implicitly quantified by $\forall$ at the clause level. The second clause states that an abstraction `abs T` is a term if its body is a term. The body is obtained applying the binder `T` (of type `tm` $\to$ `tm`) to a fresh variable `x` to obtain a term of type `tm`. Such application corresponds in a pen-and-paper proof to the (usually implicit) logical step that replaces the bound variable with a fresh one.

β-Reduction. The primitive support for substitution of Abella shows up in the following definition of β-reduction, where $t\{x\leftarrow u\}$ is the capture-avoiding meta-level substitution of u for all the occurrences of x in t, represented in Abella simply with `T U` where `T` is a binder.

$$\frac{}{(\lambda x.t)u \mapsto_\beta t\{x\leftarrow u\}} \qquad \frac{t \to_\beta t'}{tu \to_\beta t'u} \qquad \frac{t \to_\beta t'}{\lambda x.t \to_\beta \lambda x.t'} \qquad \frac{u \to_\beta u'}{tu \to_\beta tu'}$$

Defined in `01-lambda_terms_and_beta.thm`.

```
Define beta : tm -> tm -> prop by
  beta (app (abs T) U) (T U);
  beta (app T U) (app T' U) := beta T T';
  beta (abs T) (abs T') := nabla x, beta (T x) (T' x);
  beta (app T U) (app T U') := beta U U'.
```

Confluence and Parallel β. We shall repeatedly use the confluence property of β-reduction, that is, the following statement.

▶ **Theorem 1** (Confluence). *Let t be a term. If $t \to_\beta^* u_1$ and $t \to_\beta^* u_2$ then there exists s such that $u_1 \to_\beta^* s$ and $u_2 \to_\beta^* s$.*

Stated and Proved in `02-2-confluence.thm`.

This is possibly the theorem with the highest number of formalized proofs. We provide one following the standard Tait–Martin-Löf technique based on parallel β-reduction $\Rightarrow_\beta$, defined as follows, and to which we shall come back to in Sect. 5:

$$\text{Parallel } \beta\text{-reduction } \Rightarrow_\beta$$

$$\frac{}{x \Rightarrow_\beta x} \qquad \frac{t \Rightarrow_\beta t'}{\lambda x.t \Rightarrow_\beta \lambda x.t'} \qquad \frac{t \Rightarrow_\beta t' \qquad u \Rightarrow_\beta u'}{tu \Rightarrow_\beta t'u'} \qquad \frac{t \Rightarrow_\beta t' \qquad u \Rightarrow_\beta u'}{(\lambda x.t)u \Rightarrow_\beta t'\{x\leftarrow u'\}} \tag{1}$$

3 The Natural Theory is Inconsistent

Barendregt's theory stems from the study of the representation of partial recursive functions in the λ-calculus, and in particular from the observation that the natural equational theory that considers all β-diverging terms as representations of the undefined is inconsistent.

Inequational Theories. An equational theory for the λ-calculus is an equivalence relation that contains β and is *compatible* (a property also referred to as *context-closure* or *compositionality*). We shall rather work with *in*equational theories, simply obtained by starting with a preorder rather than an equivalence, as they carry enough information and avoid duplications of symmetric proofs.

Defined in `03-1-preorders_ineq_theory.thm`.

▶ **Definition 2** (Inequational theory). *An inequational theory $\mathcal{R}$ for the λ-calculus (also infixely noted $\leq_{\mathcal{R}}$) is a relation on terms such that:*

- Preorder*: $\mathcal{R}$ is reflexive and transitive;*
- Compatibility*: if $t \leq_{\mathcal{R}} u$ then $\lambda x.t \leq_{\mathcal{R}} \lambda x.u$, $ts \leq_{\mathcal{R}} us$, and $st \leq_{\mathcal{R}} su$;*
- Contains β*: if $t \to_\beta u$ then $t \leq_{\mathcal{R}} u$ and $u \leq_{\mathcal{R}} t$.*

The smallest (in)equational theory is β-conversion, that is, the transitive, reflexive and symmetric closure of β-reduction.

The Natural Theory is Inconsistent. In rewriting theory, normal forms are the results of the rewriting computational process. When using λ-terms to represent partial recursive functions (shortened to PRFs), it is tempting to consider β-normal forms as results and β-diverging terms as representing the special *undefined* value $\bot$ of PRFs. From an equational point of view, this means considering the (in)equational theory that identifies all β-diverging terms. It turns out that such a natural approach does not work because any such (in)equational theory is inconsistent, as the closure properties of theories end up identifying all terms.

Defined in `03-2-natural_theory_is_inconsistent.thm`.

▶ **Definition 3** (β-divergence). *A term t is coinductively defined as β-divergent if there exists u such that $t \to_\beta u$ and u is β-divergent.*

Stated and Proved in `03-2-natural_theory_is_inconsistent.thm`.

▶ **Proposition 4** (The natural inequational theory is inconsistent). *Let $\mathcal{R}$ be an inequational theory such that $t \leq_{\mathcal{R}} u$ for any two β-divergent terms. Then $t \leq_{\mathcal{R}} u$ for any two λ-terms.*

Proof. Consider the term $s_r := (\lambda y.yr\Omega)(\lambda z.\lambda w.z)$ parametric in r, where $\Omega := (\lambda x.xx)(\lambda x.xx)$ is the paradigmatic β-divergent term. Note that $s_r \to_\beta^* r$. Then note that $t \leq_{\mathcal{R}} s_t \leq_{\mathcal{R}} s_u \leq_{\mathcal{R}} u$ for any two terms t and u, because $\mathcal{R}$ contains β and s_r is β-divergent for any r. ◀

4 Head Reduction

The key concept of Barendregt's theory of the λ-calculus is head reduction. Intuitively, it is a restriction of β that is still being able to simulate partial recursive functions. By being a restriction, it terminates on more terms. Consequently, the set of head diverging terms is *smaller*. At the end of the paper, we shall prove one of the main theorems of Barendregt's theory: head diverging terms are *collapsible*, that is, they can be consistently equated.

Traditional Definition of Head Reduction. The traditional notion of head reduction $\to_{\mathtt{h}}$ is defined as follows:

$$\lambda x_1 \dots \lambda x_n.(\lambda y.t)us_1 \cdots s_k \quad \to_{\mathtt{h}} \quad \lambda x_1 \dots \lambda x_n.t\{y{\leftarrow}u\}s_1 \cdots s_k \qquad \text{with } n, k \geq 0.$$

For formalizing it, one needs an inductive definition, which is easily obtained, as follows:

$$\frac{}{(\lambda x.t)u \to_{\mathtt{h}} t\{x{\leftarrow}u\}} \qquad \frac{tu \to_{\mathtt{h}} s}{tur \to_{\mathtt{h}} sr} \qquad \frac{t \to_{\mathtt{h}} u}{\lambda x.t \to_{\mathtt{h}} \lambda x.u} \tag{2}$$

Traditional head reduction is a deterministic rewriting rule. When it exists, the head redex is the leftmost-outermost β-redex.

Refreshing Head Reduction. The second clause in (2) is somewhat ugly. One might wonder what happens if one adopts the following more elegant definition of head reduction – to be referred to as *refreshed* here – as done by Accattoli and Dal Lago [7]:

$$\frac{}{(\lambda x.t)u \to_{\mathtt{h}} t\{x\leftarrow u\}} \qquad \frac{t \to_{\mathtt{h}} u}{ts \to_{\mathtt{h}} us} \qquad \frac{t \to_{\mathtt{h}} u}{\lambda x.t \to_{\mathtt{h}} \lambda x.u} \tag{3}$$

Defined in `04-1-head.thm`.

Strictly speaking, the refreshed definition is not equivalent because it is more liberal: it defines a *non-deterministic* rewriting rule. For instance, one has the refreshed head step $r := (\lambda x.(\lambda y.t)u)s \to_{\mathtt{h}} (\lambda x.t\{y\leftarrow u\})s$ besides the traditional one $r \to_{\mathtt{h}} ((\lambda y.t)u)\{x\leftarrow s\}$.

It turns out, however, that – beyond determinism – the refreshed definition preserves all the properties of traditional head reduction. In particular, it induces the same sets of head terminating and diverging terms, and the same notion of head normal form. And it actually verifies a relaxed form of determinism, as discussed below.

In this work, we adopt the refreshed definition to show that it can indeed replace the traditional one in Barendregt's theory. We actually developed the whole formalization with respect to both definitions. There is only one point where the difference has a minimum impact, we shall point it out.

Head Normal Forms. The formalization forces to clarify that there are *two* predicates for head normal forms, often confused in pen-and-paper proofs. The *dynamic*, or rewriting one simply says that $\to_{\mathtt{h}}$ does not apply. The *static* one is an inductive description of head normal forms, via the auxiliary notion of rigid term, and is modeled on the following grammar.

$$\text{Rigid terms} \quad r ::= x \mid rt \qquad\qquad \text{Head normal form} \quad h ::= r \mid \lambda x.h$$

▶ **Proposition 5** (Characterization of head normal forms).

1. Dynamic to static*: if t is a term and it does not $\to_{\mathtt{h}}$-reduce then t is a head normal form;*
2. Static to dynamic*: if h is a head normal form then h does not $\to_{\mathtt{h}}$-reduce.*

Stated and Proved in `04-2-head_nfs.thm`.

Our formalization uses repeatedly the following easy *stability* of head normal forms.

▶ **Lemma 6.** *If h be a head normal form and $h \to_\beta t$ then t is a head normal form.*

Stated and Proved in `04-4-head_stabilizing.thm`.

Diamond Property. In fact, not only refreshed head reduction is confluent, it actually has the stronger *diamond property*: according to Martini and Dal Lago [23], a relation $\to_{\mathsf{r}}$ is *diamond* if $u_1 \,{}_{\mathsf{r}}\!\leftarrow t \to_{\mathsf{r}} u_2$ imply $u_1 = u_2$ or $u_1 \to_{\mathsf{r}} s \,{}_{\mathsf{r}}\!\leftarrow u_2$ for some s[1]. If $\to_{\mathsf{r}}$ is diamond then $\to_{\mathsf{r}}$ is confluent and moreover:

1. *Length invariance*: all evaluations to normal form with the same start term have the same length (*i.e.* if $d\colon t \to_{\mathsf{r}}^* u$ and $d'\colon t \to_{\mathsf{r}}^* u$ with u $\to_{\mathsf{r}}$-normal then $|d| = |d'|$);
2. *Uniform normalization*: t is weakly $\to_{\mathsf{r}}$-normalizing – or simply *terminating* – if and only if it is strongly $\to_{\mathsf{r}}$-normalizing.

Essentially, the diamond property captures a more liberal form of determinism.

Stated and Proved in `04-1-head.thm`.

For (refreshed) head reduction, we formalize the diamond property and the length invariance and uniform normalization corollaries. In fact, these properties are never used after this section, apart from the important fact that we implicitly rest on uniform normalization for using weak normalization as the termination property of reference for head reduction.

Stated and Proved in `04-5-nat-and-head_uniform_termination.thm`.

[1] Note that this formulation of the diamond property is weaker than the one for parallel β-reduction $\Rightarrow_\beta$ at work in the Tait–Martin-Löf proof of confluence (which does not have "$u_1 = u_2$ or"). This is because $\Rightarrow_\beta$ is reflexive (that is, $t \Rightarrow_\beta t$ for any term t), thus it does not need that refinement, which is instead mandatory for non-reflexive reductions such as head reduction.

The proof of uniform termination is the only place where adopting refreshed head reduction requires slightly more work: for traditional head reduction, it is an immediate consequence of determinism, while in the refreshed case it has to be proved by induction on the number of head steps using the diamond property.

5 Normalization, Factorization, and Standardization

Barendregt's study of equational theories is of a semantical nature, but it is based on a key operational result, the following (untyped) head normalization theorem.

Stated and Proved in 05-7-head_normalization.thm.

▶ **Theorem 7** (Head Normalization). *Let t be a term. If $t \to_\beta^* u$ and u does not $\to_{\mathtt{h}}$-reduce then $\to_{\mathtt{h}}$ terminates on t.*

In this section, we overview two proofs of such a theorem, both formalized in Abella. It is a contribution of this paper to show that head normalization is – repeatedly – the crucial operational result for head reduction in Barendregt's theory.

Understanding the Statement. The statement of the head normalization theorem has three subtleties. Firstly, because of the non-determinism of our refreshed head reduction, the conclusion of the general rewriting formulation of normalization would ask for strong normalization of $\to_{\mathtt{h}}$ on t. Because of uniform normalization, we can simplify it as termination of $\to_{\mathtt{h}}$. Secondly, t might $\to_\beta$-reduce to a term in $\to_{\mathtt{h}}$-normal form in many different ways, possibly without using $\to_{\mathtt{h}}$, so that the hypotheses do not immediately imply that $\to_{\mathtt{h}}$ terminates. Thirdly, the conclusion is "$\to_{\mathtt{h}}$ *terminates on* t" and not $t \to_{\mathtt{h}}^* u$, because in general maximal $\to_{\mathtt{h}}$-sequences from t all end on the same term s but it might be that $s \neq u$. For instance, let $\mathtt{I} \coloneqq \lambda y.y$: then $\mathtt{I}(x(\mathtt{II})) \to_\beta \mathtt{I}(x\mathtt{I}) \to_\beta x\mathtt{I}$ is a $\to_\beta$-sequence to head normal form, and yet the unique maximal $\to_{\mathtt{h}}$-sequence $\mathtt{I}(x(\mathtt{II})) \to_{\mathtt{h}} x(\mathtt{II})$ ends in a different term.

Stated and Proved in 05-7-head_normalization.thm.

Normalization from Factorization. Following Accattoli et al. [8], there is a very simple abstract proof of head normalization based on two properties. To state them, we need (refreshed) non-head reduction $\to_{\neg\mathtt{h}}$, given by (traditional $\to_{\neg\mathtt{h}}$ – sometimes called *internal* – has an additional clause, see [8]):

Defined in 05-2-non-head_reduction.thm.

$$\textsc{Non-Head reduction } \to_{\neg\mathtt{h}}$$

$$\frac{t \to_\beta t'}{ut \to_{\neg\mathtt{h}} ut'} \qquad \frac{t \to_{\neg\mathtt{h}} t'}{\lambda x.t \to_{\neg\mathtt{h}} \lambda x.t'} \qquad \frac{t \to_{\neg\mathtt{h}} t'}{tu \to_{\neg\mathtt{h}} t'u}.$$

The two properties then are (where $\cdot$ denotes the concatenation of rewriting relations):

1. *Persistence*: if $t \to_{\mathtt{h}} u$ and $t \to_{\neg\mathtt{h}} s$ then $s \to_{\mathtt{h}} r$ for some r;
2. *Factorization*: if $t \to_\beta^* u$ then $t \to_{\mathtt{h}}^* \cdot \to_{\neg\mathtt{h}}^* u$.

Stated and Proved in 05-7-head_normalization.thm.

Intuitively, persistence states that non-head steps cannot erase head steps. Its proof is straightforward.

Factorization expresses the fact that head reduction is more important than its complement; it is a non-trivial theorem. To appreciate the result, note that the opposite factorization does *not* hold, that is, one cannot have $t \to_{\neg\mathtt{h}}^* \cdot \to_{\mathtt{h}}^* u$. For instance, the sequence:

$$t \;:=\; (\lambda x.xy(xy))(\lambda z.z) \;\to_{\mathtt{h}}\; (\lambda z.z)y((\lambda z.z)y) \;\to_{\neg\mathtt{h}}\; (\lambda z.z)yy \;=:\; u$$

cannot be re-organized as to have the $\to_{\neg\mathtt{h}}$ step before the $\to_{\mathtt{h}}$ step.

We provide two proofs of factorization (inducing two proofs of head normalization), one as a corollary of the standardization theorem and one based on parallel reduction.

Factorization via Standardization. Standardization is another classic operational theorem. It is a complex, hard to grasp rewriting topic, the full appreciation of which requires technical concepts such as residuals. For the λ-calculus, there exist simple treatments of standardization based on Plotkin's approach [53] (originally desgined for the call-by-value λ-calculus) that succeed in hiding all the technicalities. Our formalization of standardization is the adaptation to the reasoning level of Abel's Abella development [1], who builds on Loader [42], who builds on Xi [67], who, in turn, builds on Plotkin.

The theorem states that every reduction sequence can be re-organized into a special *standard* form. Plotkin's technique stems from the existence in the λ-calculus of a very simple inductive definition of standard sequence, given here in refreshed form.

Refreshed standard reduction sequence $\rightsquigarrow_{\mathtt{st}}$

$$\frac{}{x \rightsquigarrow_{\mathtt{st}} x} \qquad \frac{t \rightsquigarrow_{\mathtt{st}} t'}{\lambda x.t \rightsquigarrow_{\mathtt{st}} \lambda x.t'} \qquad \frac{t \rightsquigarrow_{\mathtt{st}} t' \qquad u \rightsquigarrow_{\mathtt{st}} u'}{tu \rightsquigarrow_{\mathtt{st}} t'u'} \qquad \frac{t \to_{\mathtt{h}} u \qquad u \rightsquigarrow_{\mathtt{st}} u'}{t \rightsquigarrow_{\mathtt{st}} u'}.$$

Defined in `05-1-standardization.thm`.

▶ **Theorem 8** (Standardization). *Let t be a term. If $t \to_\beta^* u$ then $t \rightsquigarrow_{\mathtt{st}} u$.*

Stated and Proved in `05-1-standardization.thm`.

Our definition of standard sequence is slightly different from Abel's (and most treatments of standardization) as we use refreshed head reduction in the fourth clause, while he uses weak (i.e. not under abstraction) head reduction, which is deterministic. Our approach changes the notion of standard sequence: for instance, $(\lambda x.(\lambda y.t)u)s \to_{\mathtt{h}} (\lambda x.t\{y{\leftarrow}u\}) \to_{\mathtt{h}} t\{x{\leftarrow}u\}\{x{\leftarrow}s\}$ is a standard sequence for us, while traditionally is not. Technically, standard sequences in our case are not unique, or, equivalently, we are standardizing with respect to a *partial* order on redexes (rather than the total leftmost-outermost order). The theory of standardization for partial orders is complex [32, 44, 6]. Our proof of standardization, however, is remarkably simple: it follows exactly the structure of Abel's, and it is actually even simpler, since moving definitions to the reasoning level removes the need of a few lemmas. Additionally, our theorem implies factorization exactly as the traditional standardization one.

The implication is proved in `05-3-head_factorization.thm`

Factorization via Parallel Reduction. Takahashi [58] gives an original proof of head factorization based on parallel β-reduction $\Rightarrow_\beta$, similarly to the famous Tait–Martin-Löf proof for confluence. Accattoli et al. [8] revisit Takahashi's work, simplifying her proof technique. In particular, they identify two abstract properties of parallel β with respect to head reduction – the merge and split properties below – from which factorization easily follows, similarly to how confluence follows from the diamond property for $\Rightarrow_\beta$.

We have formalized their proof, and refer to their paper for extensive discussions, here we only provide a quick overview. The formalization follows their pen-and-paper proofs faithfully. The slight difference, again, is that we adopt the refreshed definition of head reduction, while in [8] they work with the traditional one. Beyond the already discussed refreshed non-head reduction $\to_{\neg\mathtt{h}}$, we also need its parallel variant $\Rightarrow_{\neg\mathtt{h}}$ defined as follows (traditional $\Rightarrow_{\neg\mathtt{h}}$ has an additional clause):

The proof is spread over three files (05-4 to 05-6)

Parallel non-head reduction $\Rightarrow_{\neg\mathtt{h}}$

$$\frac{}{x \Rightarrow_{\neg\mathtt{h}} x} \qquad \frac{t \Rightarrow_{\neg\mathtt{h}} t'}{\lambda x.t \Rightarrow_{\neg\mathtt{h}} \lambda x.t'} \qquad \frac{t \Rightarrow_{\neg\mathtt{h}} t' \qquad u \Rightarrow_\beta u'}{tu \Rightarrow_{\neg\mathtt{h}} t'u'}$$

Defined in `05-4-parallel_head_factorization_up_to_merge.thm`.

The proof of factorization is based on the following three properties, the first of which is simply the basic requirement for parallel reductions:

- *Parallel*: $\to_{\neg\mathtt{h}} \subseteq \Rightarrow_{\neg\mathtt{h}} \subseteq \to_{\neg\mathtt{h}}^*$;
- *Merge*: if $t \Rightarrow_{\neg\mathtt{h}} \cdot \to_{\mathtt{h}} u$ then $t \Rightarrow_\beta u$;
- *Split*: if $t \Rightarrow_\beta u$ then $t \to_{\mathtt{h}}^* \cdot \Rightarrow_{\neg\mathtt{h}} u$.

Parallel and merge are proved in 05-4-parallel_head_factorization_up_to_merge.thm

The parallel and merge properties are easily proved. Split instead is tricky. Accattoli et al. show that it can be proved easily by considering the following refinement $\overset{n}{\Rightarrow}_\beta$ of parallel β-reduction $\Rightarrow_\beta$, where the index $n \in \mathbb{N}$ intuitively denotes the number of ordinary β-steps done in parallel by the underlying $\Rightarrow_\beta$ step, and $|t|_x$ (used in the fourth clause) is the number of occurrences of x in t:

Indexed parallel β reduction

Defined in 05-5-Indexed_parallel_beta_and_arithmetic.thm.

$$\frac{}{x \overset{0}{\Rightarrow}_\beta x} \qquad \frac{t \overset{n}{\Rightarrow}_\beta t'}{\lambda x.t \overset{n}{\Rightarrow}_\beta \lambda x.t'} \qquad \frac{t \overset{n}{\Rightarrow}_\beta t' \qquad u \overset{m}{\Rightarrow}_\beta u'}{tu \overset{n+m}{\Rightarrow}_\beta t'u'} \qquad \frac{t \overset{n}{\Rightarrow}_\beta t' \qquad u \overset{m}{\Rightarrow}_\beta u'}{(\lambda x.t)u \overset{n+|t'|_x \cdot m+1}{\Rightarrow}_\beta t'\{x \leftarrow u'\}}$$

The key property of $\overset{n}{\Rightarrow}_\beta$ is its substitutivity, which is proved by decorating with indices the standard proof of the underlying substitutivity property for $\Rightarrow_\beta$ (which is used in Tait–Martin-Löf's proof of confluence).

Stated and Proved in 05-5-Indexed_parallel_beta_and_arithmetic.thm.

▶ **Lemma 9** (Substitutivity of $\overset{n}{\Rightarrow}_\beta$). *If $t \overset{n}{\Rightarrow}_\beta t'$ and $u \overset{m}{\Rightarrow}_\beta u'$, then $t\{x \leftarrow u\} \overset{k}{\Rightarrow}_\beta t'\{x \leftarrow u'\}$ where $k = n + |t'|_x \cdot m$.*

The formalization of this proof is faithful to Accattoli et al.'s pen-and-paper proof. The slightly annoying aspect is that it requires basic arithmetical properties for manipulating the indices, and Abella has no arithmetical library. Thus, we had to prove them all by hand.

6 Solvability and Head Contexts

The concept of solvability, due to Wadsworth [65], provides an interactive point of view on why head divergence is a better representation of the undefined. It shall be used to prove the maximality theorem at the end of the paper.

Solvability, On Paper. To define solvability, we need to concept of (refreshed) head contexts H, defined inductively by $H \coloneqq \langle\cdot\rangle \mid Hu \mid \lambda x.H$, together with the standard plugging operation $H\langle t\rangle$ of a term t in a context (here H), which might involve capture of free variables; for instance, $(\lambda x.\langle\cdot\rangle u)\langle xy\rangle = \lambda x.xyu$.

Defined in 06-solvability.thm.

▶ **Definition 10** ((Un)Solvable terms). *A term t is* solvable *if there is a head context H such that $H\langle t\rangle \to_\beta^* \mathtt{I} \coloneqq \lambda x.x$, and* unsolvable *otherwise.*

Stated and Proved in 06-solvability.thm.

▶ **Theorem 11** (Operational characterization of solvability, Wadsworth [65]). *t is solvable (resp. unsolvable) if and only if t is head normalizing (resp. head divergent).*

Such a characterization is of a rewriting nature, but not of t in isolation, rather of t with respect to a head context. The idea is that the head context can decompose t into pieces and leave nothing, that is, it can *(dis)solve it.* The requirement of a head context H (rather than a general context C) forces that contexts cannot simply erase t and replace it with the identity (a context taking such shortcut is $C \coloneqq (\lambda x.\mathtt{I})\langle\cdot\rangle$, but it is not a head context), they rather have to dissolve t by *interacting with its internal structure.*

Solvability shows that there are two kinds of strong (that is, all paths) divergence:

- *Unremovable strong divergence*, that is, strong divergence that cannot be removed by a head context. The looping term Ω is of this kind, because no head context can erase it.
- *Removable strong divergence*, that is, sub-terms that are strongly divergent but that they appear in argument position and thus can be removed by a head context. For instance, Ω in $x\Omega$ is removable by the head context $(\lambda x.\langle\cdot\rangle)(\lambda y.\mathtt{I})$; note that it happens in Prop. 4.

Unremovable strong divergences have to be considered as loops, or undefined terms. On the other hand, terms with removable strong divergences should not be considered in the same way, as there are some scenarios/interactions in which their divergence gets removed.

(Head) Contexts in Abella. To formalize the definition of solvability, we shall find some way of formalizing contexts in Abella. Unfortunately, there is no way of having a direct, first-class representation because plugging can capture free variables: there is no primitive support for it in Abella, nor it can be defined by the user, as it goes against the treatment of free variables in Abella. The workaround is building a predicate `head_ctx` of type `tm → tm → prop` such that `head_ctx t u` holds if there exists a head context H such that $H\langle t\rangle = u$; thus one is actually stating that t is a head sub-term of u. The predicate is easily defined inductively, breaking down the structure of the context construct by construct.

Defined in `08-solvability.thm`.

```
Define head_ctx : tm -> tm -> prop by
  head_ctx T T;
  head_ctx T (app HT U) := head_ctx T HT /\ tm U ;
  nabla x, head_ctx (T x) (abs HT) := nabla x, head_ctx (T x) (HT x).

Define solvable : tm -> prop by
  solvable T := exists HT, tm HT /\ head_ctx T HT /\ beta* HT (abs x\ x).
```

Proving Solvability. The characterization of solvability (Thm. 11) has two directions. Proving that head normalizing terms are solvable requires to refine the static predicates for head normal forms with information about the head variable (whether it is free, and, in case, which one is it). The solving context is then built by induction on the refined predicates by substituting on the head variable x a term erasing all the arguments of x and leaving the identity, plus providing dummy arguments for extra head abstractions, if any.

Proving that solvable terms are head terminating requires a big tool. The fact that $\to_\beta$ appears in the definition of solvable means we need to use head normalization (Thm. 7). Then, what remains to show is the easy fact that, if $H\langle t\rangle$ is head normalizing, then so is t.

7 Technical Interlude: Formalizing Contexts for Two Terms

For solvability, we represented head contexts indirectly via a sub-term predicate `head_ctx` : `tm → tm → prop` because the head context was always used together with a plugged term. This context predicate can easily be extended to general contexts (where the hole is not necessarily in head position), as follows:

Unused in the formalization.

```
Define ctx : tm -> tm -> prop by
  ctx T T;
  ctx T (app P Q) := ctx T P \/ ctx T Q;
  nabla x, ctx (T x) (abs CT) := nabla x, ctx (T x) (CT x).
```

For both the study of genericity and the definition of the contextual preorder in the next sections, a context C shall be plugged with *two* different terms t and u. The just defined `ctx` predicate unfortunately cannot be used for that. But it can be generalized to the following 4-ary `ctxs` predicate on terms, that represents two terms t and u (as the first and third components) together with the two terms $C\langle t\rangle$ and $C\langle u\rangle$ (as second and fourth components):

Defined in 07-1-contexts-and-contextual-preorders.thm.

```
Define ctxs : tm -> tm -> tm -> tm -> prop by
  ctxs T T U U;
  ctxs T (app A B) U (app C D) := (ctxs T A U C /\ B = D /\ tm D)
                                \/ (ctxs T B U D /\ A = C /\ tm C);
  nabla x, ctxs (T x) (abs CT) (U x) (abs CU):=
    nabla y, ctxs (T y) (CT y) (U y) (CU y).
```

8 Formalizing Genericity

In this section, we discuss and prove the result known as Barendregt's genericity lemma, which shall be used to prove both the collapsibility and maximality theorems of the next sections. In fact, genericity has an involved proof, so we rather refer to it as a theorem.

Light and Heavy Genericity. Genericity has recently been revisited by Accattoli and Lancelot [9] who identify a simpler alternative statement, dubbed *light genericity*, while they refer to Barendregt's statement as *heavy*. They show that the light version is enough for the main aim of genericity, that is, proving the collapsibility of head diverging terms. The two statements follow:

- *Light genericity*: let u be $\to_{\mathtt{h}}$-divergent and C be a context such that $C\langle u\rangle$ is $\to_{\mathtt{h}}$-normalizing. Then, $C\langle t\rangle$ is $\to_{\mathtt{h}}$-normalizing for all t.
- *Heavy genericity*: let u be $\to_{\mathtt{h}}$-divergent and C be a context such that $C\langle u\rangle \to_\beta^* n$ with n β-normal. Then, $C\langle t\rangle \to_\beta^* n$ for all t.

Preliminary. The proofs of the genericity properties that we shall develop are based on the head normalization theorem. Actually, we rather use the following strengthened corollary (its proof uses confluence as well), the isolation of which is a contribution of this paper. It shall provide very compact and elegant proofs of genericity.

Stated and Proved in 08-1-extended_normalization.thm.

▶ **Corollary 12** (Extended head normalization). *Let t be a term and $t \to_\beta^* u$. Then t is head terminating if and only if u is head terminating.*

Proving Genericity. Accattoli and Lancelot give a proof of light genericity obtained by adapting and polishing Takahashi's one for heavy genericity, from her two-page long paper titled *a simple proof of the genericity lemma* [57]. The technique is based on:

1. *Substitutions*: factoring the statement via an auxiliary one based on substitutions instead of contexts, as substitutions interact nicely with reduction;
2. *Trick*: reducing contexts to substitutions via a closure of the plugged term, and transferring termination between the two via a reasoning that is here encapsulated in the extended head normalization property above.

We formalize the proofs of both light and heavy genericity. The difficulty is formalizing Takahashi's trick, recalled next by adapting Accattoli and Lancelot's proof using Cor. 12.

Stated and Proved in 08-2-genericity.thm.

▶ **Theorem 13** (Light genericity). *Let u be head divergent and s be any term.*

1. Light genericity as substitution: *if t is a term and $t\{x\leftarrow u\}$ is head terminating then $t\{x\leftarrow s\}$ is head terminating.*
2. Light genericity as context: *if C is a context and $C\langle u\rangle$ is head terminating then $C\langle s\rangle$ is head terminating.*

Proof.

1. See [9]. In fact, [9] uses head normalization, our Abella proof does not.
2. **[Takahashi's trick]** Let $\mathtt{fv}(u) \cup \mathtt{fv}(s) = \{x_1, \dots, x_k\}$, and y be a variable fresh with respect to $\mathtt{fv}(u) \cup \mathtt{fv}(s) \cup \mathtt{fv}(C)$ and not captured by C. Note that $\bar{u} \coloneqq \lambda x_1. \dots \lambda x_k.u$ is a closed term. Consider $t \coloneqq C\langle y x_1 \dots x_k\rangle$, and note that:

$$t\{y{\leftarrow}\bar{u}\} = C\langle \bar{u} x_1 \dots x_k\rangle = C\langle (\lambda x_1. \dots \lambda x_k.u) x_1 \dots x_k\rangle \to_\beta^k C\langle u\rangle. \tag{4}$$

The fact that u is head divergent implies that $\bar{u}$ is head divergent. Similarly, take $\bar{s} \coloneqq \lambda x_1. \dots \lambda x_k.s$ and note that $t\{y{\leftarrow}\bar{s}\} \to_\beta^* C\langle s\rangle$. By extended normalization (Cor. 12), $C\langle u\rangle$ head terminating implies that so is $t\{y{\leftarrow}\bar{u}\}$. By *genericity as substitution*, $t\{y{\leftarrow}\bar{s}\}$ is head terminating. By extended normalization (Cor. 12), $C\langle s\rangle$ is head terminating. ◀

Programming Takahashi's Trick. During the formalization process, we went back and forth with multiple phrasings of Takahashi's trick. The difficulty is formalizing the argument for (4), which relies on knowing the set of free variables of a term – unproblematic on paper but tricky in Abella – and then uses many abstractions and applications at once. Our attempts led us to the following *disentangling* lemma that neatly isolates the idea with respect to a single term, avoiding the set of free variables of the term and the k-ary notations.

Stated and Proved in `08-2-genericity.thm`.

▶ **Lemma 14** (Disentangling). *Let C be a context. Then there exist t_C and a variable $x \notin \mathtt{fv}(C)$ such that for all terms u there exists u_C such that $t_C\{x{\leftarrow}u_C\} \to_\beta^* C\langle u\rangle$. Moreover, if u is head divergent then u' is head divergent.*

In fact, the proof of the lemma collects – one at at time – all the variables captured by the context, rather than all the free variables of the plugged term.

Proof. By induction on C. Cases:

- *Empty*, *i.e.* $C = \langle\cdot\rangle$. Pick any x, the statement holds with respect to $t_C \coloneqq x$ and $u_C \coloneqq u$.
- *Abstraction*, *i.e.* $C = \lambda y.C'$. By *i.h.*, there exists $t_{C'}$ and $x' \notin \mathtt{fv}(C')$ such that for all u, there exists $u_{C'}$ such that $t_{C'}\{x'{\leftarrow}u_{C'}\} \to_\beta^* C'\langle u\rangle$ and if u is head divergent then so is $u_{C'}$. Then let x be a fresh variable and set $t_C \coloneqq \lambda y.t_{C'}\{x'{\leftarrow}xy\}$ and $u_C \coloneqq \lambda y.u_{C'}$. Note that u_C is head divergent if $u_{C'}$ is, and that:

$$\begin{array}{rcccl} t_C\{x{\leftarrow}u_C\} & = & \lambda y.t_{C'}\{x'{\leftarrow}xy\}\{x{\leftarrow}\lambda y.u_{C'}\} & = & \lambda y.t_{C'}\{x'{\leftarrow}(\lambda y.u_{C'})y\} \\ & \to_\beta^* & \lambda y.t_{C'}\{x'{\leftarrow}u_{C'}\} & \to_\beta^* & \lambda y.C'\langle u\rangle \quad = \quad C\langle u\rangle. \end{array}$$

  Where the first $\to_\beta^*$ is obtained via a standard substitution lemma.
- *Application left*, *i.e.* $C = C'r$. By *i.h.*, there exists $t_{C'}$ and $x' \notin \mathtt{fv}(C')$ such that for all u, there exists $u_{C'}$ such that $t_{C'}\{x'{\leftarrow}u_{C'}\} \to_\beta^* C'\langle u\rangle$ and if u is head divergent then so is $u_{C'}$. Pick x fresh for C and note that $t_{C'}\{x'{\leftarrow}x\}\{x{\leftarrow}u_{C'}\} = t_{C'}\{x'{\leftarrow}u_{C'}\}$. The statement holds with respect to $t_C \coloneqq t_{C'}\{x'{\leftarrow}x\}r$ and $u_C \coloneqq u_{C'}$.
- *Application right*, *i.e.* $C = rC'$. Analogous to the previous case. ◀

The lemma captures the difficult-to-formalize part of the trick, and the proof is basic enough to be easily formalized. Unfortunately, the statement concerns a first-class context, while our representation of contexts in Abella is indirect, via the sub-term predicate, that is, always mentioning a plugged-in term – first-class contexts are not available in Abella.

Disentangling, Sub-Term Style. Therefore, we have to rephrase the disentangling lemma in sub-term style. For that, we introduce a `disentangling` predicate, that holds as `disentangling` $u\ C\langle u\rangle\ t_C\ u_C$ (using the notation of the lemma) and where the type of the third argument is `tm` $\to$ `tm` because it is given as a binder to capture the substitution on x.

Defined in `08-2-genericity.thm`.

```
Define disentangling : tm -> tm -> (tm -> tm) -> tm -> prop by
  disentangling T T (x\x) T;
  disentangling T (app A B) C1 T' :=
    exists C,
    (disentangling T A C T' /\ C1 =  (x\ app (C x) B) /\ tm B)
          \/
    (disentangling T B C T' /\ C1 = (x\ app A (C x)) /\ tm A);
  nabla x, disentangling (T x) (abs A)
  (x\ abs (y\ (C y (app (x) (y))))  (abs (y\ T' y)) :=
     nabla y, disentangling (T y) (A y) (x\ C y x) (T' y).
```

The disentangling lemma (Lemma 14) is represented in sub-term style by two statements. The first one expresses the fact that the disentangling predicate does capture the property of the lemma for a given plugged term u:

Stated and Proved in `08-2-genericity.thm`.

```
Theorem disentangling_trick : forall U CU U' T_C,
  tm CU -> disentangling U CU T_C U' ->
    beta* (T_C U') CU /\ ( head_div U -> head_div U' ) /\
    ( nabla x, tm (T_C x) ) /\  tm U'.
```

The second statement expresses the fact that t_C does not depend on u. This is essential: the proof of *light genericity as context* (Thm. 13.2) requires to apply the lemma in sub-term style twice (for u and s in the notations of Thm. 13), obtaining two terms t_C and t'_C that might be different. We have to ensure instead that they coincide, which is expressed as follows:

Stated and Proved in `08-2-genericity.thm`.

```
Theorem ctxs_disentangling : forall U CU S CS,
  ctxs U CU S CS ->
    exists T_C U' S', disentangling U CU T_C U' /\ disentangling S CS T_C S'.
```

Formal Proofs of Light and Heavy Genericity. Using this formalization of Takahashi's trick, the formalized proofs of light and heavy genericity smoothly follow the pen-and-paper proofs of Accattoli and Lancelot [9] and Takahashi [57]. To formalize heavy genericity, we also have to make explicit a mutually inductive grammar of β-normal forms, which Takahashi hid on paper by writing down full normal forms.

Grammar of β-normal forms defined in `08-3-beta-nfs.thm`.

Stated and Proved in `08-4-heavy_genericity.thm`.

▶ **Theorem 15** (Heavy genericity). *Both heavy genericity as substitution and as context hold.*

▶ Remark 16. Takahashi presents light genericity as a corollary of heavy genericity. We deliberately do not do it. Firstly, because light genericity is an easier, indipendently provable concept, as pointed out by Accattoli and Lancelot. Secondly, because we actually use parts of the proof of light genericity in the proof of heavy genericity.

9 The Contextual Preorder

This section starts the study of the (head) contextual preorder $\precsim_{\mathcal{C}}$, that is, the asymmetric variant of head contextual equivalence. For this paper, the contextual preorder is an essential concept: in Sect. 10, we shall show that $\precsim_{\mathcal{C}}$ is an inequational theory that consistently collapses head divergent terms, and in Sect. 11 that it is the maximal such theory.

Head Contextual Preorder and Equivalence. A natural notion of program equivalence is *contextual equivalence*, that equates terms which behave the same way in any context. We now define the head contextual preorder, refining the equivalence to an asymmetric relation and where the "behavior" of a term is whether or not it is head terminating.

▶ **Definition 17** (Head Contextual Preorder and Equivalence). *The* head contextual preorder $\precsim_{\mathcal{C}}$ *and* head contextual equivalence $\simeq_{\mathcal{C}}$ *are defined as follows:*

- $t \precsim_{\mathcal{C}} u$ *if, for all contexts* C, $C\langle t\rangle$ *is* $\to_{\mathtt{h}}$*-terminating implies that* $C\langle u\rangle$ *is* $\to_{\mathtt{h}}$*-terminating;*
- $t \simeq_{\mathcal{C}} u$ *is the equivalence relation induced by* $\precsim_{\mathcal{C}}$*, that is,* $t \simeq_{\mathcal{C}} t'$ *if* $t \precsim_{\mathcal{C}} u$ *and* $u \precsim_{\mathcal{C}} t$*.*

Defined in `09-1-head_contextual_preorder.thm`.

Head contextual equivalence is sometimes referred to as (the equational theory) $\mathcal{H}^*$, especially in Barendregt's study of the λ-calculus [15].

Contextual relations are sometimes defined slightly differently: they may be restricted to contexts C such that $C\langle t\rangle$ and $C\langle t'\rangle$ are *closed* terms. The contextual relations without the closed requirement are in general stronger, yet in some cases they are equivalent. When the observation is head termination – as it is the case here – they are equivalent, see [9].

Contextual Preorder, in Abella. We may now transpose the definition of the head contextual preorder to Abella:

Defined in `09-1-head_contextual_preorder.thm`.

```
Define ctx_preord : tm -> tm -> prop by
  ctx_preord P Q := forall CP CQ, tm P -> tm Q ->
   ctxs P CP Q CQ -> head_terminating CP -> head_terminating CQ.
```

Disentangling and the Substitution Preorder. The disentangling lemma of the previous section is a technical lemma but it actually has an interesting high-level consequence: it allows one to reformulate contextual preorder/equivalence with no reference to contexts.

First of all, note that the term u_C in the statement of the lemma can be described as $\lambda y_1 \dots \lambda y_n . u$ for some variables $y_1, \dots, y_n$ with $n \geq 0$. Then, the following new substitution preorder is shown equivalent to the contextual preorder thanks to the disentangling lemma. This is an original contribution of our work, and the distilled essence of Takahashi's trick.

▶ **Definition 18** (Substitution preorder). *The* substitution preorder $t \precsim_{\mathtt{sub}} u$ *holds when* $s\{x \leftarrow \lambda y_1 \dots \lambda y_n . t\} \to_{\mathtt{h}}$*-terminating implies that* $s\{x \leftarrow \lambda y_1 \dots \lambda y_n . u\}$ *is* $\to_{\mathtt{h}}$*-terminating for all terms* s*, variables* x*, and lists of variables* $y_1, \dots, y_n$ *with* $n \geq 0$*.*

Defined in `09-2-subst_equivalence.thm`.

▶ **Proposition 19** (Substitution and contextual preorders coincide). $t \precsim_{\mathtt{sub}} u$ *if and only if* $t \precsim_{\mathcal{C}} u$*.*

Stated and Proved in `09-2-subst_equivalence.thm`.

Proof. Direction $\Rightarrow$. Suppose that $C\langle t\rangle$ is $\to_{\mathtt{h}}$-terminating. By disentangling, there exists s_C, x and a lists of variables $y_1, \dots, y_n$ with $n \geq 0$ such that $t' \coloneqq s_C\{x \leftarrow \lambda y_1 \dots \lambda y_n . t\} \to_\beta^* C\langle t\rangle$. By the head normalization theorem, t' is $\to_{\mathtt{h}}$-terminating. By substitution preorder, $u' \coloneqq s_C\{x \leftarrow \lambda y_1 \dots \lambda y_n . u\}$ is $\to_{\mathtt{h}}$-terminating. By the disentangling lemma, $u' \to_\beta^* C\langle u\rangle$. By extended head normalization (Cor. 12), $C\langle u\rangle$ is $\to_{\mathtt{h}}$-normalizing.

Direction $\Leftarrow$. Suppose that $t' \coloneqq s\{x \leftarrow \lambda y_1 \dots \lambda y_n . t\}$ is $\to_{\mathtt{h}}$-normalizing. Consider the context $C \coloneqq (\lambda x . s)(\lambda y_1 \dots \lambda y_n . \langle\cdot\rangle)$ and note that $C\langle t\rangle \to_{\mathtt{h}} t'$, so that $C\langle t\rangle$ is $\to_{\mathtt{h}}$-normalizing. By contextual preorder, $C\langle u\rangle$ is $\to_{\mathtt{h}}$-normalising. Note that $C\langle u\rangle \to_{\mathtt{h}} u' \coloneqq s\{x \leftarrow \lambda y_1 \dots \lambda y_n . u\}$, so that u' is $\to_{\mathtt{h}}$-normalizing. ◀

10 Collapsibility

Here, we formalize the consistent collapsibility of head divergent terms using light genericity.

Terminology: Ground. For the study of collapsibility, the following terminology borrowed from Accattoli and Lancelot [9] shall simplify the discussion.

Defined in `04-3-head_terminating_predicates.thm`.

▶ **Definition 20** (Ground theory). *A term t is minimum for a preorder $\leq$ if $t \leq u$ for all u. An inequational theory $\leq_{\mathcal{R}}$ is* ground *if head divergent terms are minimum elements for $\leq_{\mathcal{R}}$.*

Note that if a theory $\leq_{\mathcal{R}}$ is ground then one has $t \leq_{\mathcal{R}} u$ and $u \leq_{\mathcal{R}} t$ for any two minimum elements t and u, which implies that $\leq_{\mathcal{R}}$ equates all head divergent terms.

Collapsibility of Head Diverging Terms. We are now ready for proving that all head diverging terms can be consistently equated (unlike β-diverging terms) by showing that the contextual preorder is an equational theory that is ground, and thus equates them.

Stated and Proved in `10-Collapsibility.thm`.

▶ **Theorem 21** (Collapsibility).

1. Inequational*: the contextual preorder $\precsim_{\mathcal{C}}$ is an inequational theory;*
2. Consistency*: the contextual preorder $\precsim_{\mathcal{C}}$ is consistent;*
3. Collapse*: the contextual preorder $\precsim_{\mathcal{C}}$ is ground.*

The difficult part of the theorem is the first point; precisely, that $\precsim_{\mathcal{C}}$ contains β. We formalize the proof by Accattoli and Lancelot [9] that uses non-trivial rewriting theorems, such as head normalisation (Thm. 7) and confluence of $\to_\beta$ (Thm. 1). Up to minor details, the formal proof follows the pen-and-paper one.

Proof.

1. It follows the proof in [9].
2. $\mathtt{I} \not\precsim_{\mathcal{C}} \Omega$ holds by considering the empty context $C := \langle\cdot\rangle$.
3. Note that light genericity (Thm. 13.2) is exactly the fact that $\precsim_{\mathcal{C}}$ is ground. ◀

Barendregt's book studies in-depth $\mathcal{H}$, the *smallest* equational theory equating all head divergent terms. We consider its associated preorder, and show it consistent.

The preorder is defined in `10-2-definition-H.thm` and proved consistent in `10-3-consistency-H.thm`.

▶ **Proposition 22.** *Let $\leq_{\mathcal{H}}$ be the smallest inequational ground theory. Then $\leq_{\mathcal{H}}$ is consistent.*

Proof. It follows from the fact that $\leq_{\mathcal{H}}$ is included in $\precsim_{\mathcal{C}}$, which is consistent (Thm. 21). ◀

11 Maximality

In this section, we prove that the contextual preorder is the maximal consistent equational theory equating head divergent terms. The proof uses solvability and light genericity.

Terminology: Adequate, and Constructive Variant. We qualify some inequational theories as adequate, following Accattoli and Lancelot and various work on program equivalence.

▶ **Definition 23** (Adequate theory). *An inequational theory $\leq_{\mathcal{R}}$ is* adequate *if head termination is stable by $\leq_{\mathcal{R}}$, i.e. for all t, u, $t \to_{\mathtt{h}}$-terminating and $t \leq_{\mathcal{R}} u$ implies $u \to_{\mathtt{h}}$-terminating.*

An inadequate theory is a theory that does not satisfy the adequacy property and a *constructively inadequate* theory is a theory where there (constructively) exists t and u such that t is head normalizing but u is head divergent.

Maximality of the Head Contextual Preorder. Another cornerstone result in Barendregt's study of equational theories is the fact that head contextual equivalence $\simeq_{\mathcal{C}}$ is the unique maximal consistent ground equational theory, that is, it captures all the identifications that one can do on top of identifying head divergent terms without risking identifying everything.

Barendregt shows that $\simeq_{\mathcal{C}}$ is a *maximal consistent* theory [15, Thm 16.2.6] relying on Böhm's separation theorem. Barendregt and Manzonetto refine the result for $\precsim_{\mathcal{C}}$ [17], using the same technique. We formalize the simpler proof by Accattoli and Lancelot [9] based on light genericity and solvability, and not needing Böhm's theorem. Our technique originates from the proof of maximality for CbV by Egidi et al. [24], also used by Arrial et al. [10].

▶ **Theorem 24** ([9]).

1. *Let $\mathcal{T}$ be an inequational theory that is ground but constructively inadequate*[2]*. Then $\mathcal{T}$ is inconsistent.*
2. Maximality of $\precsim_{\mathcal{C}}$*: $\precsim_{\mathcal{C}}$ is a maximal consistent inequational theory.*

Stated and Proved in `11-maximality.thm`.

Issues with Maximality in Abella. There are two difficulties when dealing with Thm. 24 in Abella. Firstly, Point 1 of Thm. 24 and the very concept of maximality require quantification over equational theories. This cannot be done in Abella because there is no quantification over binary term relations: quantifying over the type `prop` is disabled [3] (because of soundness issues). Fortunately, we may still phrase maximality by showing it for a constant `r` of type `tm → tm → prop` for which we know nothing.

```
Type r tm -> tm -> prop.
```

Stated and Proved in `11-maximality.thm`.

```
Theorem inadequate_ground_ineq_theories_are_inconsistent :
  ineq_theory r -> ground r ->
  (exists T U, tm T /\ head_terminating T /\ r T U /\ head_div U /\ tm U) ->
    inconsistent r.
```

The proof of this statement mimics the pen-and-paper proof of Point 1 of Thm. 24, as written by Accattoli and Lancelot [9].

Secondly, there is a delicate point in formalizing the proof of Point 2 of the theorem. The paper proof starts by picking a theory $\mathcal{T}$ that is strictly larger than $\precsim_{\mathcal{C}}$ and two terms t and u that are related in $\mathcal{T}$ but not by $\precsim_{\mathcal{C}}$. As t and u are not contextually related, there exists a context C for which, say, $C\langle t\rangle$ is head terminating while $C\langle u\rangle$ is not. *This step cannot be formalized in an intuitionistic setting* because it directly changes a $\neg\forall\phi$ statement into a $\exists\neg\phi$ statement and also uses the fact that $\neg(\phi \Rightarrow \psi) \Rightarrow \phi \wedge \neg\psi$; in all generality, these statements are equivalent to the law of excluded middle [19, 61]. We circumvent the issue by adding as an axiom the exact statement that is needed for our proof to go through.

Stated and Proved in `11-maximality.thm`.

```
Axiom[a] non_ctx_related : forall P Q,
  tm P -> tm Q -> (ctx_preord P Q -> false) ->
    exists CP CQ, ctxs P CP Q CQ /\ head_terminating CP /\ head_div CQ.

Theorem head_ctx_preord_is_maximal :
  ineq_theory r -> (forall P Q, ctx_preord P Q -> r P Q) ->
  ( exists P Q, tm P /\ tm Q /\ r P Q /\ (ctx_preord P Q -> false) ) ->
    inconsistent r.
```

[a] Axioms in Abella appear as theorems with skipped proofs.

[2] Point 1 of this theorem usually states $\mathcal{T}$ inadequate, we refine the statement to *constructively inadequate*, so that the proof of Point 1 is constructive.

12 Conclusions

We give an original and formalized presentation of the core of the theory of the untyped λ-calculus laid out in Barendregt's book, building over recent revisited definitions and proof techniques. Beyond the big picture and the formalization itself, the main contributions are the representation of contexts in higher-order syntaxes, the first formal proof of the crucial result of genericity, and the isolation of the substitution preorder.

This paper can also be seen as an extended exercise in formalizing a language with binders, in the spirit of the POPLmark challenge [11] and its variants [2, 20], with emphasis on formalizing contexts (as in contextual equivalence). It would be interesting to adapt our development to other proof assistants.

Future Work. We plan to explore the intuistionic/classical aspects of the maximality theorem (constructive proofs? constructive definition of contextual preorder?), for which we currently resort to a classical axiom, and to formalize Böhm's separation theorem building on Norrish and Tian's work [59].

References

1 Andreas Abel. Abella proof of the standardization theorem for the lambda calculus, 2009. URL: `https://abella-prover.org/examples/lambda-calculus/sred.html`.

2 Andreas Abel, Guillaume Allais, Aliya Hameer, Brigitte Pientka, Alberto Momigliano, Steven Schäfer, and Kathrin Stark. POPLMark reloaded: Mechanizing proofs by logical relations. *Journal of Functional Programming*, 29:e19, 2019. `doi:10.1017/S0956796819000170`.

3 Abella. Reference guide. `https://abella-prover.org/reference-guide.html`.

4 Beniamino Accattoli. Proof pearl: Abella formalization of λ-calculus cube property. In Chris Hawblitzel and Dale Miller, editors, *Certified Programs and Proofs - Second International Conference, CPP 2012, Kyoto, Japan, December 13-15, 2012. Proceedings*, volume 7679 of *Lecture Notes in Computer Science*, pages 173–187. Springer, 2012. `doi:10.1007/978-3-642-35308-6_15`.

5 Beniamino Accattoli, Horace Blanc, and Claudio Sacerdoti Coen. Formalizing functions as processes. In Adam Naumowicz and René Thiemann, editors, *14th International Conference on Interactive Theorem Proving, ITP 2023, July 31 to August 4, 2023, Bialystok, Poland*, volume 268 of *LIPIcs*, pages 5:1–5:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. `doi:10.4230/LIPICS.ITP.2023.5`.

6 Beniamino Accattoli, Eduardo Bonelli, Delia Kesner, and Carlos Lombardi. A nonstandard standardization theorem. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 659–670. ACM, 2014. `doi:10.1145/2535838.2535886`.

7 Beniamino Accattoli and Ugo Dal Lago. On the invariance of the unitary cost model for head reduction. In Ashish Tiwari, editor, *23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan*, volume 15 of *LIPIcs*, pages 22–37. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. `doi:10.4230/LIPICS.RTA.2012.22`.

8 Beniamino Accattoli, Claudia Faggian, and Giulio Guerrieri. Factorization and normalization, essentially. In *Programming Languages and Systems - 17th Asian Symposium, APLAS 2019, Nusa Dua, Bali, Indonesia, December 1-4, 2019, Proceedings*, pages 159–180, 2019. `doi:10.1007/978-3-030-34175-6_9`.

9 Beniamino Accattoli and Adrienne Lancelot. Light genericity. In Naoki Kobayashi and James Worrell, editors, *Foundations of Software Science and Computation Structures - 27th International Conference, FoSSaCS 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part II*, volume 14575 of *Lecture Notes in Computer Science*, pages 24–46. Springer, 2024. doi:10.1007/978-3-031-57231-9_2.

10 Victor Arrial, Giulio Guerrieri, and Delia Kesner. Genericity through stratification. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '24, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3661814.3662113.

11 Brian E. Aydemir, Aaron Bohannon, Matthew Fairbairn, J. Nathan Foster, Benjamin C. Pierce, Peter Sewell, Dimitrios Vytiniotis, Geoffrey Washburn, Stephanie Weirich, and Steve Zdancewic. Mechanized metatheory for the masses: The POPLMark challenge. In Joe Hurd and Thomas F. Melham, editors, *Theorem Proving in Higher Order Logics, 18th International Conference, TPHOLs 2005, Oxford, UK, August 22-25, 2005, Proceedings*, volume 3603 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2005. doi:10.1007/11541868_4.

12 David Baelde, Kaustuv Chaudhuri, Andrew Gacek, Dale Miller, Gopalan Nadathur, Alwen Tiu, and Yuting Wang. Abella: A system for reasoning about relational specifications. *J. Formaliz. Reason.*, 7(2):1–89, 2014. doi:10.6092/ISSN.1972-5787/4650.

13 Davide Barbarossa and Giulio Manzonetto. Taylor subsumes Scott, Berry, Kahn and Plotkin. *Proc. ACM Program. Lang.*, 4(POPL):1:1–1:23, 2020. doi:10.1145/3371069.

14 Hendrik Pieter Barendregt. *Some extensional term models for combinatory logics and λ-calculi*. PhD thesis, Univ. Utrecht, 1971.

15 Hendrik Pieter Barendregt. *The Lambda Calculus – Its Syntax and Semantics*, volume 103 of *Studies in logic and the foundations of mathematics*. North-Holland, 1984.

16 Henk Barendregt. Representing 'undefined' in lambda calculus. *J. Funct. Program.*, 2(3):367–374, 1992. doi:10.1017/S0956796800000447.

17 Henk Barendregt and Giulio Manzonetto. *A Lambda Calculus Satellite*. College Publications, 2022. URL: https://www.collegepublications.co.uk/logic/mlf/?00035.

18 Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. Proving soundness of extensional normal-form bisimilarities. *Electronic Notes in Theoretical Computer Science*, 336:41–56, 2018. The Thirty-third Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXIII). doi:10.1016/j.entcs.2018.03.015.

19 Douglas Bridges and Fred Richman. *Varieties of Constructive Mathematics*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1987. doi:10.1017/CBO9780511565663.

20 Marco Carbone, David Castro-Perez, Francisco Ferreira, Lorenzo Gheri, Frederik Krogsdal Jacobsen, Alberto Momigliano, Luca Padovani, Alceste Scalas, Dawit Legesse Tirore, Martin Vassor, Nobuko Yoshida, and Daniel Zackon. The concurrent calculi formalisation benchmark. In Ilaria Castellani and Francesco Tiezzi, editors, *Coordination Models and Languages - 26th IFIP WG 6.1 International Conference, COORDINATION 2024, Held as Part of the 19th International Federated Conference on Distributed Computing Techniques, DisCoTec 2024, Groningen, The Netherlands, June 17-21, 2024, Proceedings*, volume 14676 of *Lecture Notes in Computer Science*, pages 149–158. Springer, 2024. doi:10.1007/978-3-031-62697-5_9.

21 Kaustuv Chaudhuri, Matteo Cimini, and Dale Miller. A lightweight formalization of the metatheory of bisimulation-up-to. In Xavier Leroy and Alwen Tiu, editors, *Proceedings of the 2015 Conference on Certified Programs and Proofs, CPP 2015, Mumbai, India, January 15-17, 2015*, pages 157–166. ACM, 2015. doi:10.1145/2676724.2693170.

22 Karl Crary. A simple proof of call-by-value standardization. Technical Report CMU-CS-09-137, Carnegie Mellon University, 2009.

23 Ugo Dal Lago and Simone Martini. The weak lambda calculus as a reasonable machine. *Theor. Comput. Sci.*, 398(1-3):32–50, 2008. doi:10.1016/J.TCS.2008.01.044.

24 Lavinia Egidi, Furio Honsell, and Simona Ronchi Della Rocca. Operational, denotational and logical descriptions: a case study. *Fundam. Informaticae*, 16(1):149–169, 1992. doi:10.3233/FI-1992-16205.

25 Jörg Endrullis and Roel C. de Vrijer. Reduction under substitution. In Andrei Voronkov, editor, *Rewriting Techniques and Applications, 19th International Conference, RTA 2008, Hagenberg, Austria, July 15-17, 2008, Proceedings*, volume 5117 of *Lecture Notes in Computer Science*, pages 425–440. Springer, 2008. doi:10.1007/978-3-540-70590-1_29.

26 Yannick Forster, Ohad Kammar, Sam Lindley, and Matija Pretnar. On the expressive power of user-defined effects: effect handlers, monadic reflection, delimited control. *Proc. ACM Program. Lang.*, 1(ICFP), August 2017. doi:10.1145/3110257.

27 Yannick Forster, Steven Schäfer, Simon Spies, and Kathrin Stark. Call-by-push-value in Coq: operational, equational, and denotational theory. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2019, pages 118–131, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3293880.3294097.

28 Andrew Gacek. The Abella interactive theorem prover (system description). In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *Lecture Notes in Computer Science*, pages 154–161. Springer, 2008. doi:10.1007/978-3-540-71070-7_13.

29 Andrew Gacek, Dale Miller, and Gopalan Nadathur. Nominal abstraction. *Inf. Comput.*, 209(1):48–73, 2011. doi:10.1016/J.IC.2010.09.004.

30 Lorenzo Gheri and Andrei Popescu. Case Studies in Formal Reasoning About lambda-Calculus: Semantics, Church-Rosser, Standardization and HOAS. *CoRR*, abs/2107.11674, 2021. arXiv:2107.11674.

31 Silvia Ghilezan. Full intersection types and topologies in lambda calculus. *J. Comput. Syst. Sci.*, 62(1):1–14, 2001. doi:10.1006/jcss.2000.1703.

32 Georges Gonthier, Jean-Jacques Lévy, and Paul-André Melliès. An abstract standardisation theorem. In *Proceedings of the Seventh Annual Symposium on Logic in Computer Science (LICS '92), Santa Cruz, California, USA, June 22-25, 1992*, pages 72–81. IEEE Computer Society, 1992. doi:10.1109/LICS.1992.185521.

33 Ferruccio Guidi. Standardization and confluence in pure lambda-calculus formalized for the Matita theorem prover. *J. Formaliz. Reason.*, 5(1):1–25, 2012. doi:10.6092/ISSN.1972-5787/3392.

34 Peter V. Homeier. A proof of the Church-Rosser theorem for the λ-calculus in higher order logic. In *TPHOLs'01: Supplemental Proceedings*, pages 207–222, 2001.

35 Gérard P. Huet. Residual theory in lambda-calculus: A formal development. *J. Funct. Program.*, 4(3):371–394, 1994. doi:10.1017/S0956796800001106.

36 Jonas Kaiser, Brigitte Pientka, and Gert Smolka. Relating System F and Lambda2: A Case Study in Coq, Abella and Beluga. In Dale Miller, editor, *2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017)*, volume 84 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:19, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSCD.2017.21.

37 Richard Kennaway, Vincent van Oostrom, and Fer-Jan de Vries. Meaningless terms in rewriting. *J. Funct. Log. Program.*, 1999(1), 1999. URL: http://danae.uni-muenster.de/lehre/kuchen/JFLP/articles/1999/A99-01/A99-01.html.

38 Jean-Louis Krivine. *Lambda-calculus, types and models*. Ellis Horwood series in computers and their applications. Masson, 1993.

39 Jan Kuper. Proving the genericity lemma by leftmost reduction is simple. In Jieh Hsiang, editor, *Rewriting Techniques and Applications, 6th International Conference, RTA-95, Kaiserslautern, Germany, April 5-7, 1995, Proceedings*, volume 914 of *Lecture Notes in Computer Science*, pages 271–278. Springer, 1995. doi:10.1007/3-540-59200-8_63.

40 Adrienne Lancelot, Beniamino Accattoli, , and Maxime Vemclefs. adrilancelot/Abella-lambda-Barendregt-theory. Software, swhId: swh:1:dir:b20ffd2d8d946adac1eb2fffa72112d23a2deeed (visited on 2025-07-17). URL: https://github.com/adrilancelot/Abella-lambda-Barendregt-theory, doi:10.4230/artifacts.23906.

41 Dominique Larchey-Wendling. A constructive mechanization of lambda calculus in Coq. Online comments and Coq formalization, 2017. URL: https://homepages.loria.fr/DLarchey/Lambda_Calculus/.

42 Ralph Loader. Notes on simply typed lambda calculus. Technical report ECS-LFCS-98-381, University of Edinburgh, 1998. URL: https://www.lfcs.inf.ed.ac.uk/reports/98/ECS-LFCS-98-381/.

43 James McKinna and Robert Pollack. Some lambda calculus and type theory formalized. *J. Autom. Reasoning*, 23(3-4):373–409, 1999. doi:10.1023/A:1006294005493.

44 Paul-André Melliès. Axiomatic rewriting theory I: A diagrammatic standardization theorem. In Aart Middeldorp, Vincent van Oostrom, Femke van Raamsdonk, and Roel C. de Vrijer, editors, *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion of His 60th Birthday*, volume 3838 of *Lecture Notes in Computer Science*, pages 554–638. Springer, 2005. doi:10.1007/11601548_23.

45 Dale Miller and Gopalan Nadathur. *Programming with Higher-Order Logic*. Cambridge University Press, 2012. URL: http://www.cambridge.org/de/academic/subjects/computer-science/programming-languages-and-applied-logic/programming-higher-order-logic?format=HB.

46 Dale Miller and Alwen Tiu. A proof theory for generic judgments. *ACM Trans. Comput. Log.*, 6(4):749–783, 2005. doi:10.1145/1094622.1094628.

47 Alberto Momigliano. A supposedly fun thing I may have to do again: a HOAS encoding of Howe's method. In *Proceedings of the Seventh International Workshop on Logical Frameworks and Meta-Languages, Theory and Practice*, LFMTP '12, pages 33–42, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2364406.2364411.

48 Julian Nagele, Vincent van Oostrom, and Christian Sternagel. A short mechanized proof of the Church-Rosser theorem by the Z-property for the $\lambda\beta$-calculus in Nominal Isabelle. *CoRR*, abs/1609.03139, 2016. arXiv:1609.03139.

49 Tobias Nipkow. More Church-Rosser Proofs. *J. Autom. Reason.*, 26(1):51–66, 2001. doi:10.1023/A:1006496715975.

50 Michael Norrish. Mechanising lambda-calculus using a classical first order theory of terms with permutations. *High. Order Symb. Comput.*, 19(2-3):169–195, 2006. doi:10.1007/S10990-006-8745-7.

51 Michael Norrish. Mechanised computability theory. In Marko C. J. D. van Eekelen, Herman Geuvers, Julien Schmaltz, and Freek Wiedijk, editors, *Interactive Theorem Proving - Second International Conference, ITP 2011, Berg en Dal, The Netherlands, August 22-25, 2011. Proceedings*, volume 6898 of *Lecture Notes in Computer Science*, pages 297–311. Springer, 2011. doi:10.1007/978-3-642-22863-6_22.

52 Frank Pfenning. A Proof of the Church-Rosser Theorem and its Representation in a Logical Framework. Technical Report CMU-CS-92-186, Carnegie Mellon University, 1992. URL: https://apps.dtic.mil/sti/citations/ADA256574.

53 Gordon D. Plotkin. Call-by-name, call-by-value and the lambda-calculus. *Theor. Comput. Sci.*, 1(2):125–159, 1975. doi:10.1016/0304-3975(75)90017-1.

54 Ole Rasmussen. The Church-Rosser Theorem in Isabelle: A Proof Porting Experiment. Technical Report 164, University of Cambridge, 1995.

55 Christine Rizkallah, Dmitri Garbuzov, and Steve Zdancewic. A formal equational theory for call-by-push-value. In Jeremy Avigad and Assia Mahboubi, editors, *Interactive Theorem Proving - 9th International Conference, ITP 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10895 of *Lecture Notes in Computer Science*, pages 523–541. Springer, 2018. doi:10.1007/978-3-319-94821-8_31.

56 Natarajan Shankar. A mechanical proof of the Church-Rosser theorem. *J. ACM*, 35(3):475–522, 1988. doi:10.1145/44483.44484.

57 Masako Takahashi. A simple proof of the genericity lemma. In Neil D. Jones, Masami Hagiya, and Masahiko Sato, editors, *Logic, Language and Computation: Festschrift in Honor of Satoru Takasu*, pages 117–118. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994. doi:10.1007/BFb0032397.

58 Masako Takahashi. Parallel reductions in lambda-calculus. *Inf. Comput.*, 118(1):120–127, 1995. doi:10.1006/INCO.1995.1057.

59 Chun Tian and Michael Norrish. Mechanising Böhm trees and $\lambda\eta$-completeness. In Yannick Forster and Chantal Keller, editors, *16th International Conference on Interactive Theorem Proving (ITP 2025)*, volume 352 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:18, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITP.2025.28.

60 Alwen Tiu and Dale Miller. Proof search specifications of bisimulation and modal logics for the pi-calculus. *ACM Trans. Comput. Log.*, 11(2):13:1–13:35, 2010. doi:10.1145/1656242.1656248.

61 A. S. Troelstra and Dirk Van Dalen. *Constructivism in Mathematics: An Introduction*. North Holland, Amsterdam, 1988.

62 Christian Urban. Nominal techniques in Isabelle/HOL. *J. Autom. Reason.*, 40(4):327–356, 2008. doi:10.1007/S10817-008-9097-2.

63 René Vestergaard and James Brotherston. A formalised first-order confluence proof for the lambda-calculus using one-sorted variable names. *Inf. Comput.*, 183(2):212–244, 2003. doi:10.1016/S0890-5401(03)00023-3.

64 Christopher P. Wadsworth. *Semantics and pragmatics of the lambda-calculus*. PhD Thesis, Oxford, 1971.

65 Christopher P. Wadsworth. The relation between computational and denotational properties for scott's d_{infty}-models of the lambda-calculus. *SIAM J. Comput.*, 5(3):488–521, 1976. doi:10.1137/0205036.

66 Xinyi Wan and Qinxiang Cao. Formalization of lambda calculus with explicit names as a nominal reasoning framework. In Holger Hermanns, Jun Sun, and Lei Bu, editors, *Dependable Software Engineering. Theories, Tools, and Applications - 9th International Symposium, SETTA 2023, Nanjing, China, November 27-29, 2023, Proceedings*, volume 14464 of *Lecture Notes in Computer Science*, pages 262–278. Springer, 2023. doi:10.1007/978-981-99-8664-4_15.

67 Hongwei Xi. Upper bounds for standardizations and an application. *J. Symb. Log.*, 64(1):291–303, 1999. doi:10.2307/2586765.