Typed Operational Semantics

Healfdene Goguen

[1] INRIA Sophia-Antipolis, 2004, route des Lucioles, B. P. 93
06902 Sophia-Antipolis Cedex, France
[2] Department of Computer Science, University of Edinburgh
The King's Buildings, Edinburgh, EH9 3JZ, United Kingdom

Abstract. This paper introduces typed operational semantics, a class of formal systems which define a reduction to normal form for the well-typed terms of a particular type theory. These systems lead to a new approach to the metatheory for type theories, which we develop here for the simply typed lambda calculus. A similar approach can be used to study systems with dependent types.

1 Introduction

Untyped reduction is an intuitive operational semantics for type theory and a useful tool for implementations of type theories. However, it fails to give an adequate explanation of computation, because it lacks type information, an explicit reduction strategy and explicit mention of normal forms. We introduce a new style of operational semantics for type theory, which we call typed operational semantics, that define a reduction to normal form for well-typed terms. These systems give both an appealing account of reduction in type theory and an elegant development of the metatheory.

The central result about a typed operational semantics is the soundness of the semantics for the original type theory, for which proofs resemble traditional proofs of normalization. Completeness of the semantics is straightforward. We show how to exploit this equivalence between the declarative and operational presentations of type theory to obtain straightforward proofs of important metatheoretic properties, such as strengthening, subject reduction and strong normalization.

This paper introduces typed operational semantics in the context of the simply typed lambda calculus. This allows us to give a concise presentation of the important characteristics of the systems. However, the most important benefits of typed operational semantics are demonstrated in systems with dependent types, where our development of the metatheoretic properties is considerably simpler than the existing techniques. We discuss this briefly in Sect. 7, but a full treatment can be found elsewhere [7, 8].

This paper is organized as follows. In Sect. 2 we give a brief introduction to the simply typed lambda calculus and present the syntax of the semantically motivated presentation. We introduce the general idea of typed operational semantics by presenting a simple formal system in Sect. 3. In Sect. 4 we introduce the typed operational semantics studied in this paper and briefly discuss the rules of inference of this system. Section 5 contains the basic development of the metatheory. In Sect. 6 we give a proof of soundness of the typed operational semantics for the semantic presentation, closely related to the proof of strong normalization for the simply typed lambda calculus. Section 7 outlines the difficulties in the metatheory of systems with dependent types and gives an intuition for why typed operational semantics yield a considerably simpler treatment of these systems. We discuss related work in Sect. 8 and make concluding remarks in Sect. 9.

2 $\lambda^{\rightarrow}$: The Semantic Presentation

In this section we introduce the syntax of types and terms for the simply typed lambda calculus and present the judgements and rules of inference for the system $\lambda^{\rightarrow}$. Mitchell [14] gives a thorough introduction to this system.

2.1 Types and Terms

We first introduce the types of the simply typed lambda calculus. We assume that there is only one base type o. For any two types A and B, we have the type $A \rightarrow B$ of functions from A to B.

We assume the existence of an infinite set V of variables. We use the usual language of terms for the simply typed lambda calculus: *variables* x if $x \in V$, *abstraction* $\lambda x{:}A.M$, and *application* $M(N)$. We identify terms which are equivalent up to the renaming of bound variables and write $M \equiv N$ if M and N are equal in this way. We write $\mathrm{FV}(M)$ for the free variables in a term M, those variables not bound by abstractions. We denote the substitution of N for the free variable x in M by $[N/x]M$, where this substitution is defined to avoid the capture of free variables.

A *pre-context* is a sequence $\Gamma = x_1{:}A_1, \ldots, x_n{:}A_n$ of pairs of variables $x_i \in V$ and types A_i. We denote the empty pre-context by $()$. We write $dom(\Gamma)$ for the set $\{x_1, \ldots, x_n\}$. A *context* is a pre-context where the x_i are distinct.

2.2 Judgements and Derivations

We have two judgement forms in our presentation of the simply typed lambda calculus: $\Gamma \vdash M : A$, meaning that the term M has type A in context Γ; and $\Gamma \vdash M = N : A$, meaning that the terms M and N are equal and of type A in context Γ. We call the second judgement form *judgemental equality.* The rules of inference for the system are given in Figs. 1 and 2. We write $\Gamma \vdash J$ for an arbitrary judgement in $\lambda^{\rightarrow}$.

We have added rules of inference for thinning and substitution, following Martin-Löf's semantic explanation of type theory [12]. Although the usual presentations of the simply typed lambda calculus do not have these rules, the rules are always admissible: that is, if there are derivations of the premisses then

$$(\mathit{Var})\ \frac{\Gamma \text{ context} \quad x{:}A \in \Gamma}{\Gamma \vdash x \,:\, A} \qquad (\mathit{Thin})\ \frac{\Gamma_0, \Gamma_1 \vdash M \,:\, A \quad z \notin \mathit{dom}(\Gamma_0, \Gamma_1)}{\Gamma_0, z{:}C, \Gamma_1 \vdash M \,:\, A}$$

$$(\lambda)\ \frac{\Gamma, x{:}A \vdash M \,:\, B}{\Gamma \vdash \lambda x{:}A.M \,:\, A \to B} \qquad (\mathit{App})\ \frac{\Gamma \vdash M \,:\, A \to B \quad \Gamma \vdash N \,:\, A}{\Gamma \vdash M(N) \,:\, B}$$

$$(\mathit{Subst})\ \frac{\Gamma_0, z{:}C, \Gamma_1 \vdash M \,:\, A \quad \Gamma_0 \vdash N \,:\, C}{\Gamma_0, \Gamma_1 \vdash [N/z]M \,:\, A}$$

Fig. 1. Typing Rules

$$(\mathit{Refl})\ \frac{\Gamma \vdash M \,:\, A}{\Gamma \vdash M = M \,:\, A} \qquad (\mathit{Sym})\ \frac{\Gamma \vdash M = N \,:\, A}{\Gamma \vdash N = M \,:\, A}$$

$$(\mathit{Trans})\ \frac{\Gamma \vdash M = N \,:\, A \quad \Gamma \vdash N = P \,:\, A}{\Gamma \vdash M = P \,:\, A}$$

$$(\beta)\ \frac{\Gamma, x{:}A \vdash M \,:\, B \quad \Gamma \vdash N \,:\, A}{\Gamma \vdash (\lambda x{:}A.M)(N) = [N/x]M \,:\, B}$$

$$(\eta)\ \frac{\Gamma \vdash M \,:\, A \to B}{\Gamma \vdash \lambda x{:}A.M(x) = M \,:\, A \to B}$$

$$(\lambda\text{-}\mathit{Eq})\ \frac{\Gamma, x{:}A \vdash M = N \,:\, B}{\Gamma \vdash \lambda x{:}A.M = \lambda x{:}A.N \,:\, A \to B}$$

$$(\mathit{App}\text{-}\mathit{Eq})\ \frac{\Gamma \vdash M = P \,:\, A \to B \quad \Gamma \vdash N = Q \,:\, A}{\Gamma \vdash M(N) = P(Q) \,:\, B}$$

$$(\mathit{Subst\ Eq})\ \frac{\Gamma_0, z{:}C, \Gamma_1 \vdash M \,:\, A \quad \Gamma_0 \vdash P = Q \,:\, C}{\Gamma_0, \Gamma_1 \vdash [P/z]M = [Q/z]M \,:\, A}$$

$$(\mathit{Eq\ Subst})\ \frac{\Gamma_0, z{:}C, \Gamma_1 \vdash M = N \,:\, A \quad \Gamma_0 \vdash P \,:\, C}{\Gamma_0, \Gamma_1 \vdash [P/z]M = [P/z]N \,:\, A}$$

Fig. 2. Judgemental Equality

there is a derivation of the conclusion. Therefore, adding them is not an essential change to the system.

3 Typed Operational Semantics

A typed operational semantics presents type theory from the perspective of computation instead of that of logical inference: we still need the full type information to derive the well-typedness of any term, but we replace the logical rules for application and abstraction by rules which instead express the reduction behavior of these terms in the calculus. As an example of a typed operational semantics, we present the system λ^V in Fig. 3. This system has one judgement form, $\Gamma \vdash^V M \to^{\mathrm{nf}} P \,:\, A$, by which we mean informally that both M and P

have type A in context Γ, and that M reduces under the call-by-value strategy to normal form P.

$$(V\text{-}Base)\ \frac{\Gamma \vdash^{V} M_i \rightarrow^{\mathrm{nf}} P_i \ :\ A_i \quad (1 \le i \le n)}{\Gamma \vdash^{V} x(M_1, \ldots, M_n) \rightarrow^{\mathrm{nf}} x(P_1, \ldots, P_n) \ :\ B} \quad \textit{if } x{:}A_1 \rightarrow \ldots \rightarrow A_n \rightarrow B \in \Gamma$$

$$(V\text{-}\lambda)\quad \frac{\Gamma, x{:}A \vdash^{V} M \rightarrow^{\mathrm{nf}} P \ :\ B}{\Gamma \vdash^{V} \lambda x{:}A.M \rightarrow^{\mathrm{nf}} \lambda x{:}A.P \ :\ A \rightarrow B} \quad \textit{if } P \equiv Q(x) \textit{ implies } x \in \mathrm{FV}(Q)$$

$$(V\text{-}\eta)\quad \frac{\Gamma, x{:}A \vdash^{V} M \rightarrow^{\mathrm{nf}} P(x) \ :\ B}{\Gamma \vdash^{V} \lambda x{:}A.M \rightarrow^{\mathrm{nf}} P \ :\ A \rightarrow B} \quad \textit{if } x \notin \mathrm{FV}(P)$$

$$(V\text{-}\beta)\quad \frac{\begin{array}{c}\Gamma \vdash^{V} M \rightarrow^{\mathrm{nf}} \lambda x{:}A.P \ :\ B \rightarrow C \qquad \Gamma \vdash^{V} N \rightarrow^{\mathrm{nf}} Q \ :\ B \\ \Gamma \vdash^{V} [Q/x]P \rightarrow^{\mathrm{nf}} R \ :\ C\end{array}}{\Gamma \vdash^{V} M(N) \rightarrow^{\mathrm{nf}} R \ :\ C}$$

Fig. 3. Call-By-Value Reduction

λ^V is typical of typed operational semantics in being a deterministic reduction to normal form with type information. We can prove strengthening, normalization and the decidability of type-checking for $\lambda^{\rightarrow}$ by proving these results in λ^V and transferring them by soundness and completeness. However, λ^V encodes a particular reduction strategy to normal form, so we are unable to use this system to establish properties about arbitrary reductions, such as strong normalization or subject reduction. In the next section, we consider a typed operational semantics that is universal with respect to reductions and therefore allows us to show these results as well.

4 λ^S: A System for Standard Reduction

4.1 Standard Reduction

Standard reduction [2] is an important notion in the untyped lambda calculus: any reduction sequence can be expressed as a standard, or left to right, reduction sequence. In this section we introduce the system λ^S, a synthesis of this reduction and the semantic presentation $\lambda^{\rightarrow}$. We shall use this system to study the metatheory of the simply typed lambda calculus. Because the reduction used in λ^S is universal with respect to reductions, the important results about the relationship between typed terms and reduction follow easily by reasoning in this system.

4.2 Judgements and Derivations

We now define the calculus λ^S. The judgement forms for λ^S and their intuitive meanings are:

- $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$, meaning that M has normal form P of type A in context Γ.
- $\Gamma \vdash^S M \rightarrow^{\mathrm{wh}} N : A$, meaning that M weak head reduces to N of type A in context Γ.

We write $\Gamma \vdash^S J$ for an arbitrary judgement in λ^S. We also use the abbreviation $\Gamma \vdash^S M : A$ for $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ when the normal form P is not important.

The typed operational semantics λ^S is defined by the rules of inference in Figs. 4 and 5.

$$(S\text{-}Base)\ \frac{\Gamma \vdash^S M_i \rightarrow^{\mathrm{nf}} P_i : A_i \quad (1 \leq i \leq n)}{\Gamma \vdash^S x(M_1, \ldots, M_n) \rightarrow^{\mathrm{nf}} x(P_1, \ldots, P_n) : B}\ \ \textit{if } x{:}A_1 \rightarrow \ldots \rightarrow A_n \rightarrow B \in \Gamma$$

$$(S\text{-}\lambda)\ \frac{\Gamma, x{:}A \vdash^S M \rightarrow^{\mathrm{nf}} P : B}{\Gamma \vdash^S \lambda x{:}A.M \rightarrow^{\mathrm{nf}} \lambda x{:}A.P : A \rightarrow B}\ \ \textit{if } P \equiv Q(x) \textit{ implies } x \in \mathrm{FV}(Q)$$

$$(S\text{-}\eta)\ \frac{\Gamma, x{:}A \vdash^S M \rightarrow^{\mathrm{nf}} P(x) : B \qquad \Gamma \vdash^S P \rightarrow^{\mathrm{nf}} Q : A \rightarrow B}{\Gamma \vdash^S \lambda x{:}A.M \rightarrow^{\mathrm{nf}} Q : A \rightarrow B}$$

$$(S\text{-}WH)\ \frac{\Gamma \vdash^S M \rightarrow^{\mathrm{wh}} N : A \qquad \Gamma \vdash^S N \rightarrow^{\mathrm{nf}} P : A}{\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A}$$

Fig. 4. Canonical Forms

$$(W\text{-}\beta)\ \frac{\Gamma \vdash^S \lambda x{:}A.M : A \rightarrow B \qquad \Gamma \vdash^S N : A}{\Gamma \vdash^S (\lambda x{:}A.M)(N) \rightarrow^{\mathrm{wh}} [N/x]M : B}$$

$$(W\text{-}App)\ \frac{\Gamma \vdash^S M \rightarrow^{\mathrm{wh}} P : A \rightarrow B \qquad \Gamma \vdash^S N : A}{\Gamma \vdash^S M(N) \rightarrow^{\mathrm{wh}} P(N) : B}$$

Fig. 5. Weak Head Reduction

4.3 Discussion

We discuss the rules of inference of λ^S in order to illustrate the basic idea of the system and to explain some choices in the presentation.

First, a variable applied to a sequence of terms, which we call a *base term*, evaluates by the rule (*S-Base*) to the variable followed by the normal forms of the terms. We need to know that x occurs in the context to ensure well-typedness.

The rules (S-λ) and (S-η) for abstraction implement η-postponement and preserve normality of the right-hand side of the reduction. The rules first fully reduce the body M of $\lambda x{:}A.M$, and by analysis of the resulting normal form either the abstraction is already in normal form or a final η-reduction takes place. This yields a simple factorization of the reduction sequence into β-reductions followed by η-reductions.

The second premiss of the rule (S-η), requiring the normal form of the η-redex to be well-typed the context Γ, is unnecessary from the point of view of our reduction, although it serves as a side condition requiring x to be distinct from the free variables of P. By strengthening the premisses, we have weakened the rule and simplified proofs by induction on derivations of λ^S. Furthermore, using our metatheoretic development, we can prove the admissibility of the more natural η-reduction rule:

$$(S\text{-}\eta\text{'})\ \frac{\Gamma, x{:}A \vdash^S M \to^{\mathrm{nf}} P(x) : B}{\Gamma \vdash^S \lambda x{:}A.M \to^{\mathrm{nf}} P : A \to B}\ \textit{if } x \notin \mathrm{FV}(P)$$

Weak head reduction is defined by the rules of inference for the judgement form $\Gamma \vdash^S M \to^{\mathrm{wh}} N : A$. Intuitively weak head redices are outermost redices: either a β-redex or a weak head redex on the left of an application. By using this judgement form, we also require the reduct to be well-typed, because the only rule incorporating derivations of this judgement into the main judgement $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ is the rule (S-WH).

We do not use the normal forms of either $\lambda x{:}A.M$ or N in the rule (S-β). However, it is important from the point of view of reduction that the two terms are normalizing, and from the point of view of typing that the two terms are well-typed.

5 Metatheory

In this section we prove the essential metatheoretic properties for the simply typed lambda calculus. Our presentation is influenced by Luo's thesis [9] and developments of Pure Type Systems [3, 17], although the formal system is considerably different. Since we shall eventually show the equivalence of λ^S and $\lambda^{\to}$, all of the results for λ^S will transfer to $\lambda^{\to}$ as well.

5.1 Basic Definitions

We first introduce the basic notions of reduction and parallel substitution.

The untyped reduction rules β and η are the usual analogues of the corresponding rules for judgemental equality. If R is a relation on terms then we denote its compatible closure by $M \rhd_R N$, its transitive closure by R^+ and its reflexive and transitive closure by R^*. Untyped reduction, $M \rhd N$, is the compatible closure of $\beta\eta$.

A term M is *normal* if it has no reductions, that is if there is no N such that $M \rhd N$. A term is *strongly normalizing* if all reduction sequences starting from that term terminate. A term M satisfies the *diamond property* if $M \rhd^* N$ and $M \rhd^* P$ imply that there is a Q such that $N \rhd^* Q$ and $P \rhd^* Q$.

A *pre-substitution* for a finite set of variables S is a function from S to terms. If δ is a pre-substitution for $dom(\Gamma)$ then we write $\widehat{\delta}(M)$ for the result of simultaneously substituting the values for the variables in the domain of δ:

$$\widehat{\delta}(M) =_{\mathrm{df}} [\delta(x_1), \ldots, \delta(x_n)/x_1, \ldots, x_n]M$$

If ϕ is a pre-substitution for Φ then the *composition* of ϕ and δ, $\phi \circ \delta$, is

$$\phi \circ \delta(x) =_{\mathrm{df}} \widehat{\phi}(\delta(x))$$

We write $\delta[x{:=}M]$ for the extended pre-substitution for $dom(\Gamma, x{:}A)$. If Δ has all components of Γ then we define the substitutions $\mathrm{weak}_{\Gamma}^{\Delta}(x) =_{\mathrm{df}} x$ for $x \in dom(\Gamma)$, and $\mathrm{id}_{\Gamma} =_{\mathrm{df}} \mathrm{weak}_{\Gamma}^{\Gamma}$.

A *substitution* δ from Δ to Γ is a pre-substitution for $dom(\Gamma)$ such that $\Delta \vdash^S \delta(x) : A$ for each $x{:}A \in \Gamma$. A *renaming* δ from Δ to Γ is a substitution from Δ to Γ such that $\delta(x) = y$, where $y \in V$, for each $x{:}A \in \Gamma$.

An important property of renamings is that if δ is a renaming from Δ to Γ and ϕ is a renaming from Φ to Δ then $\phi \circ \delta$ is a renaming from Φ to Γ. This property is not straightforward for arbitrary substitutions, and we shall only know it as a consequence of our final soundness theorem.

5.2 Basic Metatheory

In this section we establish the basic metatheoretic properties we need for the two calculi presenting the simply typed lambda calculus. Most of the results will be for the system λ^S, because we shall transfer the results from this system to $\lambda^{\rightarrow}$ by soundness and completeness results.

The results in this section follow by straightforward induction on derivations unless otherwise mentioned.

Lemma 1. *If* $\Gamma \vdash M = N : A$ *then* $\Gamma \vdash M : A$ *and* $\Gamma \vdash N : A$.

Lemma 2 Free Variables. *If* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ *then* $\mathrm{FV}(M) \subseteq dom(\Gamma)$ *and* $\mathrm{FV}(P) \subseteq dom(\Gamma)$.

In contrast to the semantic presentation, we need to show Free Variables before Generation to treat the rule of inference for η-redices.

Lemma 3 Generation. *Any judgement which is derivable in* λ^S *has a uniquely determined last rule of inference.*

Lemma 4 Uniqueness of Normal Forms. *If* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ *and* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} Q : B$ *then* $P \equiv Q$ *and* $A = B$.

This lemma states that the operational semantics is deterministic. The system also has the stronger property of being *syntax directed*, meaning that for each context Γ and term M there are at most one term M and type A such that $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$, but we do not need this fact explicitly.

Lemma 5 Context. *If $\Gamma \vdash^S J$ then Γ is a context.*

Lemma 6 Completeness. *If $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ then $\Gamma \vdash M = P : A$.*

We now show that any judgement in a context Γ also holds in any context which has all components of Γ. We first prove a general lemma inspired by McKinna and Pollack's treatment [13], where we allow names of variables in the context to change.

Lemma 7 Renaming. *If δ is a renaming from Δ to Γ and $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ then $\Delta \vdash^S \widehat{\delta}(M) \rightarrow^{\mathrm{nf}} \widehat{\delta}(P) : A$.*

The reasoning for the rules $(S\text{-}\lambda)$ and $(S\text{-}\eta)$ is complicated. We have the usual Thinning lemma as a corollary.

Corollary 8 Thinning. *If $\Gamma \vdash^S J$ and Δ is a context with all components of Γ then $\Delta \vdash^S J$.*

Strengthening for the simply typed lambda calculus is a straightforward result. However, the property can be very difficult to prove for more complicated calculi with dependent types, and the fact that the typed operational semantics is deterministic leads to a simple and elegant proof of this property for such systems.

Lemma 9 Strengthening. *If $\Gamma_0, z{:}C, \Gamma_1 \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ and $z \notin \mathrm{FV}(M)$ then $\Gamma_0, \Gamma_1 \vdash^S M \rightarrow^{\mathrm{nf}} P : A$.*

Proving a substitution or cut lemma for λ^S directly seems to be difficult. Indeed, substitution on the judgements of λ^S is difficult to define, because we need to know the normal form of the substituted term. Substitution is therefore closely related to soundness of λ^S for $\lambda^{\rightarrow}$, which we have emphasized by including the rule (*Subst*) in the rules of inference for $\lambda^{\rightarrow}$.

5.3 $\boldsymbol{\lambda^S}$ and Reduction

The calculus λ^S is a synthesis of standard reduction and the typing rules from the simply typed lambda calculus, which makes it a powerful tool for showing results about the relationship between typing in λ^S and untyped reduction.

Lemma 10 Adequacy for Reduction.

- *If $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ then $M \rhd^* P$ and P is normal, and furthermore there is an N such that $M \rhd^*_\beta N \rhd^*_\eta P$.*

- *If* $\Gamma \vdash^S M \rightarrow^{\mathrm{wh}} N : A$ *then* $M \rhd_\beta N$.

Corollary 11. *If* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ *and* M *is normal then* $M \equiv P$.

We now prove a general lemma about the relationship between untyped reduction and the system λ^S. This lemma will imply the important properties of strong normalization, subject reduction and the diamond property.

Definition 12. The predicate $\mathcal{S}_\Gamma^{P,A}(M)$ is defined as the least relation such that $\mathcal{S}_\Gamma^{P,A}(M)$ holds if:

- $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$,
- for all N which are subterms of M there are B and Q such that $\Gamma, \Delta \vdash^S N \rightarrow^{\mathrm{nf}} Q : B$, where Δ is the sequence of binders which N occurs under in M, and
- $M \rhd N$ implies $\mathcal{S}_\Gamma^{P,A}(N)$ for all N.

$\mathcal{C}_\Gamma^{P,A}(M)$ will be the same as $\mathcal{S}_\Gamma^{P,A}(M)$ except that in the last clause instead of $M \rhd N$ we use $M \rhd^+ N$.

This definition is closely related to the definition of strong normalization in Altenkirch's thesis [1], although we have added type information. We have used this definition because of the similarity of the proofs of subject reduction, strong normalization and the diamond property in this setting. The predicate is well-defined because it is strictly positive. We have the obvious generation lemma for $\mathcal{S}_\Gamma^{P,A}(M)$, and we can also show that $\mathcal{S}_\Gamma^{P,A}(M)$ iff $\mathcal{C}_\Gamma^{P,A}(M)$, and that Strengthening holds for the predicates.

We need an auxiliary lemma about η-reduction, which follows using Generation, Adequacy for Reduction and Strengthening.

Lemma 13 Subject Reduction for (η). *If* $\Gamma \vdash^S \lambda x{:}A.M(x) \rightarrow^{\mathrm{nf}} P : A \rightarrow B$ *and* $x \notin \mathrm{FV}(M)$ *then* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A \rightarrow B$.

Lemma 14 Reduction.

- *If* $\Gamma \vdash^S M \rightarrow^{\mathrm{nf}} P : A$ *then* $\mathcal{S}_\Gamma^{P,A}(M)$.
- *If* $\Gamma \vdash^S M \rightarrow^{\mathrm{wh}} N : A$ *and* $\mathcal{S}_\Gamma^{P,A}(N)$ *then* $\mathcal{S}_\Gamma^{P,A}(M)$.

Proof. By induction on derivations. We outline two cases:

- (S-λ). We need to show that:
  - $\Gamma \vdash^S \lambda x{:}A.M \rightarrow^{\mathrm{nf}} \lambda x{:}A.P : A \rightarrow B$,
  - there are C and Q such that $\mathcal{S}_{\Gamma,\Delta}^{Q,C}(N)$ for all subterms N of M, and
  - $\lambda x{:}A.M \rhd N$ implies $\mathcal{S}_\Gamma^{\lambda x:A.P, A \rightarrow B}(N)$.

  The first follows directly and the second follows by inductive hypothesis. For the last, we know that $M \rhd N$ implies that $\mathcal{S}_{\Gamma,x:A}^{P,B}(N)$ for all N. The result follows by induction on $\lambda x{:}A.M \rhd N$, where we use Strengthening, Lemma 13 and Uniqueness of Normal Forms for (η).

– $(W\text{-}\beta)$. We know that $\mathcal{S}_\Gamma^{P,A\to B}(\lambda x{:}A.M)$ and $\mathcal{S}_\Gamma^{Q,A}(N)$ by inductive hypothesis, and $\mathcal{S}_\Gamma^{R,B}([N/x]M)$ by assumption, so $\mathcal{C}_\Gamma^{R,B}([N/x]M)$. The result follows by induction on the normalization premisses.

$\square$

All of the important results about reduction are corollaries to Adequacy for Reduction and Reduction.

Corollary 15 Strong Normalization for λ^S. *If $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ then M is strongly normalizing.*

Corollary 16 Subject Reduction for λ^S. *If $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ and $M \rhd^* N$ then $\Gamma \vdash^S N \to^{\mathrm{nf}} P : A$.*

Corollary 17 Diamond Property for λ^S. *If $\Gamma \vdash^S M \to^{\mathrm{nf}} Q : A$, $M \rhd^* N$ and $M \rhd^* P$ then $N \rhd^* Q$ and $P \rhd^* Q$.*

Finally, we show the admissibility of the more natural η rule using Adequacy for Reduction, Subject Reduction, Generation, Corollary 11 and Strengthening.

Proposition 18 Admissibility of (S-η'). *If $\Gamma, x{:}A \vdash^S M \to^{\mathrm{nf}} P(x) : B$ and $x \notin \mathrm{FV}(P)$ then $\Gamma \vdash^S \lambda x{:}A.M \to^{\mathrm{nf}} P : A \to B$.*

We now have a proof of the diamond property for $\beta\eta$-equality. It is well-known that this property fails on non-well-typed terms with type labels: the term $\lambda x{:}A.(\lambda y{:}B.M)(x)$ reduces to either $\lambda x{:}A.[x/y]M$ by (β) or $\lambda y{:}B.M$ by (η), and if A and B are not the same then the two terms have no common reduct. On the other hand, this counterexample does not hold for well-typed terms, because by the rules of inference the type labels A and B must be identical for any well-typed term. The relative simplicity of our approach serves to emphasize the usefulness of having type information in the computational presentation of type theory.

6 Soundness

In this section we prove the fundamental result of soundness, which relates the semantic presentation $\lambda^{\to}$ to the typed operational semantics λ^S. Specifically, we show that if $\Gamma \vdash M : A$ then there is a P such that $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$. The soundness result is crucial to our treatment of the metatheory of the simply typed lambda calculus, because we use it to transfer the metatheoretic development of λ^S to $\lambda^{\to}$.

An important first step in a proof of normalization is to elicit the reduction behavior of the system being studied. Our typed operational semantics serves this purpose: the rules of inference show that the well-typed terms are exactly those constructed with variables and abstraction and a series of weak head expansions. The technique we use to show soundness is closely related to existing proofs of normalization because these proofs do construct such reduction sequences. A proof of soundness of λ^V for $\lambda^{\rightarrow}$ would be easier than the proof of soundness for λ^S, in the same way that proofs of normalization are easier than proofs of strong normalization.

Our proof follows Coquand and Gallier's technique [5] of a Kripke-style interpretation and uses finite contexts and the general notion of substitutions. We show that if a term is well-typed in $\lambda^{\rightarrow}$ then that term is well-typed in λ^S, relative to any substitution.

6.1 The Interpretation

We use a slight variation of the usual model construction to show normalization.

Definition 19 Semantic Object. A *semantic object* for Δ and A is a term M such that $\Delta \vdash^S M : A$.

We emphasize that semantic objects are always normalizing. Therefore, the fact that an element of the interpretation is normalizing will require no proof, unlike many proofs of normalization [14, 16].

Definition 20 Interpretation. Let Δ be a context. The interpretation of a type A in Δ, $[\![A]\!]\Delta$, is given by induction on the structure of types:

- $[\![o]\!]\Delta =_{\mathrm{df}} \{M \mid M$ is a semantic object for Δ and $o\}$.
- $[\![A \rightarrow B]\!]\Delta$ is the set of semantic objects M for Δ and $A \rightarrow B$ such that for any renaming δ from Δ' to Δ and any $N \in [\![A]\!]\Delta'$ we know that $\widehat{\delta}(M)(N) \in [\![B]\!]\Delta'$.

Definition 21 Interpretation of Contexts. The interpretation of a context Γ in a context Δ, $[\![\Gamma]\!]\Delta$, is a set of substitutions from Δ to Γ defined by induction on the structure of Γ:

- $[\![()]\!]\Delta =_{\mathrm{df}} \{\epsilon\}$.
- $[\![\Gamma, x{:}A]\!]\Delta =_{\mathrm{df}} \{\rho[x{:=}M] \mid \rho \in [\![\Gamma]\!]\Delta$ and $M \in [\![A]\!]\Delta\}$.

6.2 Properties of the Interpretation

In this section we list the basic properties of the interpretation.

Definition 22 Saturated Set. A set S of semantic objects for Δ and A is a *saturated set* for Δ and A if:

(S1) If M is a base term then $M \in S$.
(S2) If $N \in S$ and $\Delta \vdash^S M \rightarrow^{\mathrm{wh}} N : A$ then $M \in S$.

The usual third condition on saturated sets, that if $M \in S$ then M is strongly normalizing, is enforced by the definition of semantic object.

The following properties are standard.

Lemma 23 Saturated Set. *$[\![A]\!]\Delta$ is a saturated set for any Δ and A.*

Lemma 24 Monotonicity. *If $M \in [\![A]\!]\Delta$ and δ is a renaming from Δ' to Δ then $\widehat{\delta}(M) \in [\![A]\!]\Delta'$.*

Lemma 25 Monotonicity for Contexts. *If $\rho \in [\![\Gamma]\!]\Delta$ and δ is a renaming from Δ' to Δ then $\delta \circ \rho \in [\![\Gamma]\!]\Delta'$.*

6.3 Soundness

Finally, we can show the soundness result.

Lemma 26 Soundness. *If $\Gamma \vdash M : A$ and $\rho \in [\![\Gamma]\!]\Delta$ then $\widehat{\rho}(M) \in [\![A]\!]\Delta$.*

Proof. By induction on derivations of $\Gamma \vdash M : A$.

We outline (λ), the one difficult case. We first show that $\lambda x{:}A.M$ is a semantic object for Δ and $A \to B$ by considering the substitution

$$(\mathrm{weak}_{\Delta}^{\Delta, y:A} \circ \rho)[x{:=}y] \in [\![\Gamma, x{:}A]\!]\Delta, y{:}A$$

for y fresh in Δ, using $(S\text{-}\lambda)$ or $(S\text{-}\eta)$ as appropriate. We then need to show that $\widehat{\delta}(\lambda x{:}A.M)(N) \in [\![B]\!]\Delta'$ for any renaming δ from Δ' to Δ and any $N \in [\![A]\!]\Delta'$, which we show using Renaming, $(W\text{-}\beta)$, *(S2)*, and the definition of $[\![A \to B]\!]\Delta$. □

It is straightforward to show that $\mathrm{id}_\Gamma \in [\![\Gamma]\!]\Gamma$, and the main result follows.

Corollary 27. *If $\Gamma \vdash M : A$ then there is a P such that $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$.*

Corollary 28 Soundness for $\Gamma \vdash M = N : A$. *If $\Gamma \vdash M = N : A$ then there is a P such that $\Gamma \vdash^S M \to^{\mathrm{nf}} P : A$ and $\Gamma \vdash^S N \to^{\mathrm{nf}} P : A$.*

Proof. By induction on derivations that $\Gamma \vdash M = N : A$, using for example Uniqueness of Normal Forms for (*Trans*), and Soundness and Lemma 13 for (η).

The rule (*App-Eq*) follows by Soundness, Adequacy for Reduction, Subject Reduction and Uniqueness of Normal Forms. □

6.4 Consequences of Soundness

We now transfer the metatheoretic study of λ^S in Sect. 5 to the simply typed lambda calculus by using Corollary 27 and Completeness.

Corollary 29 Strong Normalization. *If $\Gamma \vdash M : A$ then M is strongly normalizing.*

Corollary 30 Subject Reduction. *If $\Gamma \vdash M : A$ and $M \rhd N$ then $\Gamma \vdash N : A$.*

Corollary 31 Church–Rosser. *If $\Gamma \vdash M = N : A$ then there is a P such that $M \rhd^* P$ and $N \rhd^* P$.*

We also know the properties of Free Variables, Context, Renaming and Strengthening for $\lambda^{\to}$.

7 Dependent Types

We have demonstrated how typed operational semantics give a new approach to the metatheory for the simply typed lambda calculus, but there is one important aspect of typed operational semantics which is only brought out in the study of systems with dependent types and $\beta\eta$-equality. In these systems, with type equalities that allow a term to have syntactically different but judgementally equal types, there is a complex interaction between the properties of strengthening, subject reduction, Church–Rosser and strong normalization. The counterexample to the diamond property for non-well-typed terms mentioned in Sect. 5.3 now presents further difficulties: the labels on the two abstractions can be judgementally equal but require an arbitrary number of reductions to reach a common reduct.

Salvesen [15] gave the first proof of subject reduction and Church–Rosser for Pure Type Systems, and Geuvers [6] gives another proof. These proofs show strong normalization first, and use this to show strengthening, subject reduction and Church–Rosser, using complicated syntactic arguments on a different term structure. Coquand [4] shows normalization and the other results for Martin-Löf's Logical Framework with $\beta\eta$-equality by proving the completeness of an untyped algorithm for testing conversion. The algorithm comprises a reduction relation to weak head normal form and a binary relation which compares these weak head normal forms for $\beta\eta$-convertibility.

The development using typed operational semantics is very similar to the above treatment of the simply typed lambda calculus: the essential difference is in the definition of the system, which makes use of the normal forms of types in the rules for application, where the application is well-typed only if the normal forms of the types of the subterms agree. Normalization replaces the type equality rule in the semantic presentation, which coerces a term from one type to another, and leads to a system where there is at most one derivation for each judgement. This, together with the separate rule of inference for η, allows us to follow the outline of the development for the simply typed lambda calculus, proving strengthening directly and following by showing the other important metatheoretic properties.

Luo's type theory UTT [10] includes an impredicative universe of propositions, predicative universes of types and a general schema for inductive types, formulated in Martin-Löf's Logical Framework with η-equality on the framework function spaces and abstractions with labels. These features lead to the problems in the metatheory which we have mentioned above, and which are the same as those in the Calculus of Constructions with η-conversion. We have used typed operational semantics to develop the metatheory of this system [7, 8].

8 Related Work

Several ideas have been influential in the development of typed operational semantics. Definitions of typed reduction are rare in the literature, but Martin-Löf [11] proves normalization and Church–Rosser for a reduction defined by the rules of inference for judgemental equality excluding symmetry.

Mitchell [14] defines a type closed predicate to be a predicate $\mathcal{P}$ on terms which satisfies the following conditions:

- If $\mathcal{P}(M_1), \ldots, \mathcal{P}(M_n)$ then $\mathcal{P}(x(M_1, \ldots, M_n))$, for x of appropriate type.
- If for all N, if $\mathcal{P}(N)$ then $\mathcal{P}(M(N))$, then $\mathcal{P}(M)$.
- If $\mathcal{P}(N)$ and $\mathcal{P}(([N/x]M)(N_1, \ldots, N_n))$ then $\mathcal{P}((\lambda x{:}A.M)(N, N_1, \ldots, N_n))$.

Typed operational semantics have several advantages over these predicates. Most importantly, our technique is appealing and familiar because of its similarity to the dynamic semantics of programming languages. More technically, typed operational semantics include information about reduction, normal forms and contexts, and furthermore the rules of inference have a finite number of premisses, which is not the case for the second rule for type closed predicates. Each of these differences either makes typed operational semantics simpler to use than type closed predicates or leads to a larger class of results which can be proved using our technique.

However, the similarity of typed operational semantics and type closed predicates makes clear the view of λ^S as an alternative induction principle for well-typed terms in the simply typed lambda calculus: the rules of inference can be seen as conditions on predicates, and reasoning by induction on derivations corresponds to verifying these conditions. One way of thinking of λ^S is as a formal system expressing general properties of predicates which need to be established in order to know that the predicate holds for judgements in the simply typed lambda calculus. The soundness result says that such predicates are a model for the simply typed lambda calculus.

Coquand's algorithm for testing conversion [4], mentioned in Sect. 7, also provided part of the inspiration for typed operational semantics. His algorithm implements a particular strategy of reduction to normal form and also does not include type information. This means that there is no direct induction principle in this setting, which leads to proofs of strengthening and Church–Rosser that are indirect and more complicated than ours. Furthermore, it seems that strong normalization cannot be proved using this technique.

9 Conclusions

We have introduced a new class of formal systems, typed operational semantics, which present type theory from a computational perspective. In doing so, we have arrived at a strategy for studying reduction in type theory opposite to the frequently adopted approach of removing type information from the proof of normalization. The specific typed operational semantics that we study, based on standard reduction, itself gives a precise, coherent description of the relationship between typing and reduction. We have demonstrated that this system gives a new treatment of fundamental results, including strong normalization, Church–Rosser and subject reduction, and we have outlined how this approach simplifies the study of systems with dependent types and η-equality considerably.

Acknowledgments

Zhaohui Luo taught me much of what I know about the metatheory for type theories. The work in this paper was influenced by discussions with him and James McKinna. Rod Burstall suggested presenting typed operational semantics in the context of the simply typed lambda calculus. Adriana Compagnoni and Jan Smith read drafts of this paper and suggested improvements. Finally, the anonymous referees made useful suggestions, including adding the section which discusses the technique for dependent types.

References

1. T. Altenkirch. *Constructions, Inductive Types and Strong Normalization.* PhD thesis, University of Edinburgh, 1993.
2. H. Barendregt. *The Lambda Calculus: its Syntax and Semantics.* North Holland, revised edition, 1984.
3. H. Barendregt. Lambda calculi with types. In S. Abramsky, D. M. Gabbai, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2. Oxford University Press, 1991.
4. T. Coquand. An algorithm for testing conversion in type theory. In G. Huet and G. Plotkin, editors, *Logical Frameworks.* Cambridge University Press, 1991.
5. T. Coquand and J. Gallier. A proof of strong normalization for the theory of constructions using a Kripke-like interpretation. In *Workshop on Logical Frameworks–Preliminary Proceedings*, 1990.
6. H. Geuvers. *Logics and Type Systems.* PhD thesis, Katholieke Universiteit Nijmegen, Sept. 1993.
7. H. Goguen. The metatheory of *UTT*. Submitted to Proceedings of BRA Workshop on Types for Proofs and Programs, 1994.
8. H. Goguen. *A Typed Operational Semantics for Type Theory.* PhD thesis, University of Edinburgh, Aug. 1994.
9. Z. Luo. *An Extended Calculus of Constructions.* PhD thesis, University of Edinburgh, Nov. 1990.
10. Z. Luo. *Computation and Reasoning.* Oxford University Press, 1994.
11. P. Martin-Löf. An intuitionistic theory of types: Predicative part. In H. Rose and J. C. Shepherdson, editors, *Logic Colloqium 1973*, pages 73–118, 1975.
12. P. Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1984.
13. J. McKinna and R. Pollack. Pure type systems formalized. In M. Bezem and J. F. Groote, editors, *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, pages 289–305. Springer–Verlag, LNCS 664, Mar. 1993.
14. J. Mitchell. Type systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science.* North Holland, 1990.
15. A. Salvesen. The Church-Rosser property for pure type systems with $\beta\eta$-reduction, Nov. 1991. Unpublished manuscript.
16. W. W. Tait. Intensional interpretation of functionals of finite type I. *Journal of Symbolic Logic*, 32, 1967.
17. L. van Benthem Jutting, J. McKinna, and R. Pollack. Typechecking in pure type systems. In H. Barendregt and T. Nipkow, editors, *Types for Proofs and Programming.* Springer–Verlag, 1993.