Notes on Normalization
International School on Rewriting
Loria, 2007

Daniel J. Dougherty
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609 USA
E-mail: `dd@cs.wpi.edu`

(revised) July 10, 2007

Abstract

We present the basic techniques for analyzing the notion of strong normalization in the λ-calculus, focusing on intersection type systems.

1 Background

In these informal notes we have not attempted to give complete background details nor to ascribe results to the proper authors. For a comprehensive introduction to the untyped λ-calculus see Barendregt's book [Bar84]. The book by Krivine [Kri93] has an elegant introduction of intersection types, going well beyond what we present here.

It is worth mentioning that often (usually) intersection types are developed along with some structure — typically a partial order —- on the set of types. This is an important aspect of one of the main applications of intersection types, building models for the untyped λ-calculus. Since our modest focus is on strong normalization only we present the minimal mechanism to analyze reduction.

2 Terms and reduction

Syntax

We first define the set of untyped λ-terms. A term is either

- a variable $x, y, z, \ldots,$
- an application (MN), or
- an abstraction $(\lambda x.M)$

The *free variables* in a term are:

$$\begin{array}{lcl} FVx & = & \{x\} \\ FV\lambda x.M & = & FVM \setminus \{x\} \\ FV(MN) & = & FV(M) \cup FV(N) \end{array}$$

A variable occurrence which is not free is called a *bound* occurrence. We assume Barendregt's [Bar84] convention, namely that *a variable does not occur free and bound in the same term.* For instance, we assume that x does not occur free in N in the term $(\lambda x.M)N$.

We avoid writing parentheses when possible, under convention that application to the associate left. We may also write repeated abstractions using a single λ. For example, under these conventions, the term $(((\lambda x.(\lambda y.M))N)P)$ can be written as $(\lambda xy.M)NP$

Substitution The substitution of term N for variable x in term M is denoted $M[x := N]$. This substitution must be defined carefully to avoid the unintended "capture" of a free variable in N by a binder of M. We do not give a careful definition here: the Barendregt convention described above suffices to ensure that a naive replacement will not go awry.

Reduction

The key equation of the λ-calculus relates application and abstraction.

$$(\beta) \qquad (\lambda x.M)N \;=\; M[x := N]$$

When oriented left-to-right this is β-*reduction*.

Another important equation with an associated notion of reduction is

$$(\eta) \qquad \lambda x.Fx \;=\; F \qquad \text{if } x \notin FV(F)$$

But for simplicity we will not work with η in these notes.

A term to which no reductions apply is called a *normal form*. This is ambiguous of course since there are several notions of reduction one might consider; one should speak of, for example, β-normal forms or $\beta\eta$-normal forms and so on. In these notes we will always be considering β-reduction, and so "normal form" will always mean "β-normal form."

Confluence Observe that there is a critical pair in β-reduction: in $(\lambda x.M)N$ both M and N might contain redexes. It is not hard to see that β-reduction is locally confluent, but since, as we will see, not all terms have terminating reduction behavior it is not obvious that the reduction is confluent. However, confluence does in fact hold: consult see [Bar84] for several proofs!

3 Normalization

It is easy to see that not all reductions out of all terms will terminate. Consider for example the term $(\lambda x.xx)(\lambda x.xx)$, which reduces to itself.

It is reasonable to hope that we might be able identify some nice classes of well-behaved terms; of course that is what these notes are all about.

A term M is said to be *normalizing* if there exists some reduction out of M which results in a normal (ie irreducible) term. Sometimes such an M is called *weakly normalizing*.

A term M is said to be *strongly normalizing* if every reduction out of M is finite. By virtue of the fact that reduction is confluent, this means that all reductions terminate in the same normal form.

It turns out that the best way to study normalization properties of λ-terms is through type systems. The following results hold.

- Each term which is typable under the most naive type system (the system of *simple types*) is strongly normalizing. (But there are strongly normalizing terms which cannot be typed with simple types.)
- If we extend the system of simple types to the *intersection types* system we achieve a *characterization* of strong normalization: a term is strongly normalizing if and only of it can be typed with an intersection type.
- If we refine the intersection types system just referred to, we can achieve a characterization of terms with certain intermediate normalization behavior, such as those terms which are weakly normalizing, or those which normalize under the *head reduction* relation. For lack of time and space these notes will not elaborate on this point.

Let $\mathcal{SN}$ denote the set of all strongly normalizing terms.

4 An instructive failure

Let us try to prove that all terms are $\mathcal{SN}$, by induction on terms. This is false, as we know, but it will be instructive to see what goes wrong.

- Variables are certainly $\mathcal{SN}$.
- If M is $\lambda x.B$ and by induction, B is $\mathcal{SN}$, then we may conclude M is $\mathcal{SN}$.
- If M is an application let us write is as FA. By induction F and A are $\mathcal{SN}$. But what if F looks like $\lambda x.B$ and we perform a β-reduction at the top, obtaining $B[x := A]$? Our induction hypothesis doesn't tell us anything about this term, and we are stuck.

This suggests that we need a stronger induction hypothesis, one that says that the strongly normalizing terms are closed under substitution. Unfortunately this is simply false. The term $\lambda z.zz$ is in $\mathcal{SN}$, and the term xx is in $\mathcal{SN}$, but substituting $\lambda z.zz$ for x in xx leads to a non-$\mathcal{SN}$ term.

This exercise might seem silly since we knew at the start that we cannot prove all terms to be $\mathcal{SN}$. But if we are interested in showing certain *classes* of terms be $\mathcal{SN}$, then the above discussion suggests that we should focus on classes which are well-behaved with respect to substitution. All of this motivates our next definition, of saturated set.

5 Saturated Sets

Definition 1. A set $\mathcal{S}$ of terms is *saturated* if, for all $n \geq 0$,

1. $\mathcal{S} \subseteq \mathcal{SN}$,
2. $xA_1 \ldots A_n \in \mathcal{S}$ provided each A_i is in $\mathcal{SN}$, and
3. $(\lambda x.A_0)A_1 \ldots A_n \in \mathcal{S}$ provided each A_i is in $\mathcal{SN}$ and $A_0[x := A_1] \ldots A_n \in \mathcal{S}$.

Note that by taking $n = 0$ in the first clause above we conclude that every saturated set contains all variables.

Definition 2 (Function space)**.** If $\mathcal{A}$ and $\mathcal{B}$ are sets of terms then $\mathcal{A} \multimap \mathcal{B}$ is $\{M \mid \forall A \in \mathcal{A}, (MA) \in \mathcal{B}\}$.

The following are easy consequences of the definition.

Lemma 3. *Let $\mathcal{A}$ and $\mathcal{B}$ be sets of terms.*

1. *If $\mathcal{A}$ and $\mathcal{B}$ are saturated then so is $\mathcal{A} \multimap \mathcal{B}$.*
2. *If $\mathcal{A}$ and $\mathcal{B}$ are saturated then so is $\mathcal{A} \cap \mathcal{B}$.*
3. *$\mathcal{SN}$ is saturated.*

Proof. (Hints.)

1. To see that $(\mathcal{A} \multimap \mathcal{B}) \subseteq \mathcal{SN}$, take $M \in (\mathcal{A} \multimap \mathcal{B})$, then note that since each saturated set contains all variables, we can take a variable $x \in \mathcal{A}$ and be assured that $Mx \in \mathcal{B}$. Since $\mathcal{B}$ is saturated Mx is $\mathcal{SN}$, which implies M is $\mathcal{SN}$. The other clauses in the definition of saturated can be verified easily.
2. Easy.
3. The proof uses the observation that if there is an infinite reduction out of a term $(\lambda x.A_0)A_1 \ldots A_n$ then there is an infinite reduction out of the term obtained by doing the head reduction first.

□

6 Simple types

Definition 4 (The system of type assignment $\mathcal{T}^{\rightarrow}$)**.** Given an infinite set of type-variables the set of types is formed by closing the type-variables under the operations $\sigma \rightarrow \tau$ and $\sigma \cap \tau$.

A *statement* is an expression of the form $M\colon \tau$; where M is a term, the *subject* of the statement, and τ is a type. A basis is a set of statements with distinct variables as subjects. A judgment is a triple (Γ, M, τ) where Γ is a basis, M is a term, and τ is a type; the notion of a judgment's being derivable in system $\mathcal{T}^{\rightarrow}$, denoted $\Gamma \vdash M\colon \tau$ is given by the rules of inference in Table 1.

We say that a term M is *simply typable* if there exists a Γ and a τ such that $\Gamma \vdash M\colon \tau$ in the above system.

$$\text{start} \quad \frac{(x\colon \sigma) \in \Gamma}{\Gamma \vdash x\colon \sigma}$$

$$\rightarrow\text{I} \quad \frac{\Gamma, x\colon \sigma \vdash M\colon \tau}{\Gamma \vdash \lambda x.M\colon \sigma \rightarrow \tau} \qquad \rightarrow\text{E} \quad \frac{\Gamma \vdash M\colon \sigma \rightarrow \tau \qquad \Gamma \vdash N\colon \sigma}{\Gamma \vdash (MN)\colon \tau}$$

Table 1: Typing rules for $\mathcal{T}^{\rightarrow}$

Basic properties of typing

Lemma 5 (Basic typing properties)**.**

1. *If* $\Gamma \vdash M\colon \tau$ *and* $\Gamma \subseteq \Gamma'$ *then* $\Gamma' \vdash M\colon \tau$.
2. *If* $\Gamma \vdash M\colon \tau$ *and* Γ' *agrees with* Γ *on the free variables of* M *then* $\Gamma' \vdash M\colon \tau$.
3. *(Inversion)* $\Gamma \vdash \lambda x.B\colon \sigma \rightarrow \tau$ *if and only if* $x\colon \sigma, \Gamma \vdash B\colon \tau$.

Proof. Easy. □

Type inference An important question about any types system is whether there is an algorithm for determining what types (if any) a term has in the system. We note in passing that For the system $\mathcal{T}^{\rightarrow}$ of simple types typability is decidable. An algorithm witnessing this fact fact can be found in many textbooks (this algorithm is the basis for type-checking in languages like Haskell and ML).

Subject Reduction

The following lemma is the heart of Subject Reduction.

Lemma 6 (Substitution lemma)**.** *Suppose* $\Gamma, x : \sigma \vdash M : \tau$ *and* $\Gamma \vdash N : \sigma$. *Then*

$$\Gamma \vdash M[x := N] : \tau$$

Proof. Induct over the derivation of $\Gamma, x : \sigma \vdash M : \tau$. □

Theorem 7 (Subject Reduction)**.** *Suppose* $\Gamma \vdash M\colon \tau$ *and* $M \rightarrow M_1$. *Then* $\Gamma \vdash M_1 : \tau$.

Proof. By induction over derivation of $\Gamma \vdash M\colon \tau$. We organize the rest of the argument by cases according to the shape of the term M.

If M is an abstraction $\lambda x.B$ then M_1 is of the form $\lambda x.B_1$ with $B \rightarrow B_1$. We show that $\Gamma \vdash M\colon \tau$ implies $\Gamma \vdash M_1 : \tau$; we need only consider the situation which the typing of M ends with $\rightarrow$-introduction. Then $\tau = (\tau_1 \rightarrow \tau_2)$ and $\Gamma, y\colon \tau_1 \vdash B\colon \tau_2$. The induction hypothesis yields the desired typing for B_1 and then we may type M_1 as desired.

In case M is FA and was typed by

$$\rightarrow\text{E}\quad \frac{\Gamma \vdash F : \sigma \rightarrow \tau \qquad \Gamma \vdash A : \sigma}{\Gamma \vdash (FA) : \tau}$$

there are three cases for the reduction to M_1. If M_1 is $F_1 A$ with $F \rightarrow F_1$ or M_1 is FA_1 with $A \rightarrow A_1$ then we may apply the induction hypothesis to the typing of F or A as appropriate. On the other hand, if M is $(\lambda x.B)A$ and M_1 is $B[x := A]$ then we may apply Lemma 5 to conclude that $\Gamma, x : \sigma \vdash B : \tau$ and then use Lemma 6.

□

Here is a subtlety.

Corollary 8. *Reduction on typable terms is confluent.*

Proof. It may seem that there is nothing to prove here since we have confluence on all λ-terms. But it is not true in general that a sub-system of a confluent rewriting system will be confluent. (Easy exercise: make a counterexample.) But the Subject Reduction theorem implies that if we start from a typable term and chase through the usual diagram embodying the confluence property —- when the terms are considered as untyped terms — the terms we build will all be typable and so the reductions are indeed reductions over the set of typable term. □

7 Simply-typable terms are strongly normalizing

Definition 9. An *interpretation* I is a function from types to sets of terms obeying the following

- $I_{\alpha \rightarrow \beta} = I_\alpha \twoheadrightarrow I_\beta$

Obviously an interpretation is completely determined by its value on the type variables. Suppose I is an interpretation and $\mathcal{X}$ is a set of terms such that I_t is saturated for each type-variable t. Then I_τ is saturated for each type τ, by Lemma 3.

Theorem 10 (Soundness)**.** *Let I be an interpretation such that I_t is saturated for each type-variable t. If M is typable with type τ then $M \in I_\tau$.*

Proof. Let us say that a substitution θ is saturated if $\theta(x_i) \in I_{\alpha_i}$ for $1 \leq i \leq n$. It is convenient to prove the following more general claim.

> *Let I be as in the statement of the theorem, suppose $\Gamma \vdash M : \tau$, and let θ be a saturated substitution defined on the variables in the domain of Γ. Then $\theta M \in I_\tau$.*

Every saturated set contains all variables (cf Definition 1 part 2), so the identity substitution is a saturated substitution. Thus the statement above indeed implies the theorem.

The proof of the claim is by induction on the derivation of $\Gamma \vdash M : \tau$; we organize the argument according to the last inference in the derivation.

The start rule. Here $M \equiv x$ with $(x : \alpha) \in \Gamma$. We want to show that show that $\theta x \in I_\alpha$, which holds by assumption.

The $\rightarrow$-E rule. We have

$$\frac{\Gamma \vdash U : \alpha \rightarrow \beta \qquad \Gamma \vdash V : \alpha}{\Gamma \vdash UV : \beta}$$

To show that $\theta(UV) \in I_\beta$; note that $\theta(UV) = \theta(U)\theta(V)$ and use the induction hypothesis and the definition of $I_{\alpha \rightarrow \beta}$.

The $\to$-I rule.

$$\frac{\Gamma,(x\colon \alpha)\;\vdash\; B\colon \beta}{\Gamma\;\vdash\;\lambda x.B\colon \alpha\to\beta}$$

To show that $\theta(\lambda x.B) \in I_{\alpha\to\beta}$, write this as $(\lambda x.\theta B)$ (renaming x if necessary). It suffices to choose an arbitrary $A \in I_\alpha$, and argue that $(\lambda x.\theta B)A \in I_\beta$. By saturation (cf Definition 1 part 1) it suffices to see that $\theta'(B) \in I_\beta$ where θ' is the substitution obtained from θ by mapping x to A. But θ' is a saturated substitution so the induction hypothesis applies.

□

Proposition 11. *If M is typable in $\mathcal{T}^{\to}$ then M is strongly normalizing.*

Proof. Suppose M is typable with type τ. Let I be the interpretation I mapping each variable to the set $\mathcal{SN}$ of strongly normalizing terms. Since $\mathcal{SN}$ is a saturated set the Soundness Theorem applies, and we conclude that $M \in I_\tau$ and since I_τ is saturated we conclude that M is $\mathcal{SN}$. □

8 Intersection types

It is natural to ask: are strongly normalizing terms simply-typable? It is easy to see that the answer is no. The standard example is the term $\lambda x.xx$. We now define a type system which will in fact charaterize the strongly normalizing terms. The amazing thing is that we have to make ony the smallest change in the types and rules.

Definition 12 (The system of type assignment $\mathcal{T}^{\cap}$)**.** The set of intersection types is formed by closing the type-variables under the operations $\sigma\to\tau$ and $\sigma\cap\tau$.

The notion of a judgment's being derivable in system $\mathcal{T}^{\cap}$, is given by the rules of inference in Table 2.

$$\text{start}\quad \frac{(x\colon \sigma)\in\Gamma}{\Gamma\vdash x\colon \sigma}$$

$$\to\text{-I}\quad \frac{\Gamma,x\colon \sigma\vdash M\colon \tau}{\Gamma\vdash \lambda x.M\colon \sigma\to\tau} \qquad \to\text{-E}\quad \frac{\Gamma\vdash M\colon \sigma\to\tau \qquad \Gamma\vdash N\colon \sigma}{\Gamma\vdash (MN)\colon \tau}$$

$$\cap\text{-I}\quad \frac{\Gamma\vdash M\colon \sigma_1 \qquad \Gamma\vdash M\colon \sigma_2}{\Gamma\vdash M\colon \sigma_1\cap\sigma_2} \qquad \cap\text{-E}\quad \frac{\Gamma\vdash M\colon \sigma_1\cap\sigma_2}{\Gamma\vdash M\colon \sigma_i}\quad i\in\{1,2\}$$

Table 2: Typing rules for $\mathcal{T}^{\cap}$

Basic properties of intersection typing

Lemma 13 (Basic typing properties)**.**

1. *If $\Gamma\vdash M\colon \tau$ and $\Gamma\subseteq\Gamma'$ then $\Gamma'\vdash M\colon \tau$.*
2. *If $\Gamma\vdash M\colon \tau$ and Γ' agrees with Γ on the free variables of M then $\Gamma'\vdash M\colon \tau$.*
3. *(Inversion)* $\Gamma\vdash \lambda x.B\colon \sigma\to\tau$ *if and only if* $x\colon \sigma,\Gamma\vdash B\colon \tau$.

Proof. The first three assertions are routine inductions. The Inversion property requires a little trickery in the presence of the intersection rules, which change type of a judgment without a change in the term. The key is to prove a more general claim.

Claim: if $\Gamma\vdash \lambda x.B\colon \pi$ where $p_i = p_{i_1}\cap p_{i_2}\ldots\cap p_{i_n}$ for $(n\geq 1)$ and no p_{i_j} is an intersection, and for some j, p_{i_j} is $\sigma\to\tau$ then $\Gamma,x\colon \sigma\vdash B\colon \tau$.

Note that the result we want is the case $n = 1$ of the claim. The proof of the claim is an easy induction on the size of the typing tree.

□

Lemma 14 (Substitution lemma)**.** *Suppose* $\Gamma, x : \sigma \vdash M : \tau$ *and* $\Gamma \vdash N : \sigma$. *Then*

$$\Gamma \vdash M[x := N] : \tau$$

Proof. Induct over the derivation of $\Gamma, x : \sigma \vdash M : \tau$. □

Type inference It is undecidable whether a term M is typable in system $\mathcal{T}^{\cap}$. The follows from the fact that — as we will prove — a term is typable if and only if it is strongly normalizing (the latter fact is well-known to be undecidable, by Scott's Theorem).

Subject Reduction

Theorem 15 (Subject Reduction)**.** *Suppose* $\Gamma \vdash M\colon \tau$ *and* $M \to M_1$. *Then* $\Gamma \vdash M_1 : \tau$.

Proof. Exactly as for simple types. □

Corollary 16. *Reduction on typable terms is confluent.*

9 Intersection-typable terms are strongly normalizing

By adding intersections we are now able to type many more terms. But still all the terms we can type are strongly normalizing. Furthermore the proof that intersection typable terms are strongly normalizing needs only a tiny modification to the proof simple types.

1. add to the definition of interpretation the clause

$$I_{\alpha \cap \beta} = I_\alpha \cap I_\beta$$

2. note that since saturated sets are closed under intersection it is still true that each I_τ is saturated.
3. In the Type Soundness theorem, check that the two new cases, for the rules $\cap$-I and $\cap$-E, submit to an immediate application of the induction hypothesis and the definition of $I_{\alpha\cap\beta}$.

Then we have:

Proposition 17. *If M is typable in $\mathcal{T}^{\cap}$ then M is strongly normalizing.*

Proof. Just as for simlpe types. □

10 Strongly normalizing terms are intersection-typable

Finally we show that with intersection types we have captured precisely the strongly normalizing terms. First we collect some observations about terms and possible typings.

Definition 18. Let Γ_1 and Γ_2 be bases. The basis $\Gamma_1 \sqcap \Gamma_2$ contains $x\colon \sigma$ if either:

- $(x\colon \sigma)$ is in Γ_1 and $x \notin \mathcal{D}om(\Gamma_2)$, or
- $(x\colon \sigma)$ is in Γ_2 and $x \notin \mathcal{D}om(\Gamma_1)$, or
- $\sigma = \sigma_1 \cap \sigma_2$ with $(x\colon \sigma_1) \in \Gamma_1$ and $(x\colon \sigma_2) \in \Gamma_2$.

Lemma 19. *if* $\Gamma \vdash M\colon \tau$ *then for all* Γ', $\Gamma \sqcap \Gamma' \vdash M\colon \tau$.

A convenient consequence of this lemma is the following.

Corollary 20. *If M and N are each typable, then there is a single basis Γ such that M and N are each typable under Γ.*

Proof. Apply the $\sqcap$ operator to the respective given bases. □

Lemma 21.

1. *If B is typable then $\lambda x.B$ is typable.*
2. *If $A_1, \ldots, A_n$ are each typable $(n \geq 0)$ then $xA_1 \cdots A_n$ is typable.*
3. *Suppose $\Gamma \vdash B[x := A] : \sigma$ and A is typable under Γ. Then there is a basis Γ' such that $\Gamma' \vdash (\lambda x.B)A : \sigma$*

Proof.

1. Easy induction on terms.
2. Another induction on terms. By induction we have for each i that $\Gamma_i \vdash A_i : \tau_i$ for some Γ_i and τ_i. We can build a typing for $xA_1 \cdots A_n$ using Lemma 5 (exercise).
3. It is convenient to establish the following preliminary claim (which is a sort of weak converse to the Substitution Lemma).

   Suppose
   $$\Gamma \vdash B[x := A] : \tau \text{ and } \Gamma \vdash A : \sigma$$
   Then there exists σ' such that for any fresh variable y we have
   $$\Gamma, y : \sigma' \vdash B[x := y] : \tau \text{ and } \Gamma \vdash A : \sigma'$$

   The proof of the claim is an induction over the structure of B, with a sub-induction over the size of the typing of B.

   The desired assertion follows, since from $\Gamma, y : \sigma' \vdash B[x := y] : \tau$ we conclude $\Gamma \vdash \lambda x.B : \sigma' \to \tau$ by applying typing rule $(\to\text{I})$ and renaming the bound variable; then $\Gamma' \vdash (\lambda x.B)A : \sigma$ follows.

□

Proposition 22. *If N is a normal form then there is a basis Γ and a type σ such that $\Gamma \vdash N : \sigma$*

Proof. An easy induction using the first two parts of Lemma 21. □

Proposition 23. *If a term is strongly normalizing then it is typable in system $\mathcal{T}^{\cap}$.*

Proof. For an $\mathcal{SN}$ term M let $bd(M)$ denote the length of the longest reduction out of M (this number exists by König's Lemma).

Let M be $\mathcal{SN}$; we prove that M is typable by induction over $bd(M)$, with a secondary induction over the structure of M.

The base case is when M is a normal form; here we apply Proposition 22. For the induction step we examine cases for the form of M.

- when M is $\lambda x.P$: since P must be $\mathcal{SN}$, an easy application of the induction hypothesis applies (using Lemma 21.1).
- when M is $xA_1 \cdots A_n$: then again we apply induction (with the help of Lemma 21.2).
- when M is $(\lambda x.B)AA_1 \cdots A_n$: consider the term $N \equiv (B[x := A])A_1 \cdots A_n$. This term is typable by induction. Also note that $bd(A)$ is less than $bd(M)$, so by induction A is typable. So Lemma 21.3 applies to any typing which has $(B[x := A])$ as its conclusion. So in the typing derivation for $N \equiv (B[x := A])A_1 \cdots A_n$ we may replace all subtrees with conclusion $(B[x := A])$ by typings of $(\lambda x.B)A$ with the same type. These new subtrees might have a different basis at the bottom but we can invoke Corollary 20 to ensure that we build a valid typing for M eventually.

□

Suppose $\Gamma \vdash M : \tau$; we show that for some Γ and some τ; we have $\Gamma' \vdash M : \tau'$. (Note that we cannot, in general, arrange that the typing basis and the result type be preserved.)

Theorem 24. *A term is $\mathcal{SN}$ if and only if it is typable in system $\mathcal{T}^{\cap}$.*

Proof. By Propositions 17 and 23. □

References

[Bar84] H. P. Barendregt. *The Lambda-Calculus, its syntax and semantics*. Studies in Logic and the Foundation of Mathematics. Elsevier Science Publishers B. V. (North-Holland), Amsterdam, 1984. Second edition.

[Kri93] J.-L. Krivine. *Lambda calculus, types and models*. Ellis Horwood, 1993.