Introduction to the Literature On Programming Language Design

Gary T. Leavens

TR #93-01c
Jan. 1993, revised Jan. 1994, Feb. 1996, and July 1999

Keywords: programming languages, semantics, type systems, polymorphism, type theory, data abstraction, functional programming, object-oriented programming, logic programming, declarative programming, parallel and distributed programming languages.
1992 CR Categories: D.1.1 [*Programming Techniques*] Applicative (Functional) Programming; D.1.3 [*Programming Techniques*] Concurrent Programming; D.1.5 [*Programming Techniques*] Object-oriented Programming; D.1.6 [*Programming Techniques*] Logic Programming; D.2.2 [*Software Engineering*] Tools and Techniques — Modules and interfaces; D.2.4 [*Software Engineering*] Program Verification; D.2.4 [*Software Engineering*] Program Verification; D.3.1 [*Programming Languages*] Formal Definitions and Theory; D.3.2 [*Programming Languages*] Language Classifications; D.3.3 [*Programming Languages*] Language Constructs and Features; F.3.1 [*Logics and Meaning of Programs*] Specifying and verifying and reasoning about programs; F.3.2 [*Logics and Meaning of Programs*] Semantics of Programming Languages; F.3.3 [*Logics and Meaning of Programs*] Studies of Program Constructs; H.2.3 [*Database Management*] Languages—Database (persistent) programming languages; I.2.5 [*Artificial Intelligence*] Programming Languages and Software.

Department of Computer Science
226 Atanasoff Hall
Iowa State University
Ames, Iowa 50011-1040, USA

Introduction to the Literature On Programming Language Design

Gary T. Leavens*
Department of Computer Science, Iowa State University
Ames, IA, 50011-1040 USA
leavens@cs.iastate.edu

July 29, 1999

Abstract

This is an introduction to the literature on programming language design and related topics. It is intended to cite the most important work, and to provide a place for students to start a literature search.

This is a selective *introduction* to the literature on programming language design. The intended audience is graduate students beginning a study of programming languages. Instead of trying to be comprehensive, references are given that are to works of lasting value, up-to-date surveys, or that seem to be important or interesting at the moment.[1] Besides references that have intrinsic interest, a few are included because they are the original sources and are likely to be referenced by others doing related work (e.g. [Chu41]). To probe an area more deeply, start with the papers mentioned, follow their references, and also use the *Science Citation Index* to see what papers have referenced the ones mentioned.

As a general aid to finding papers, many older references are reprinted in a collection called *TUTORIAL Programming Language Design* edited by A. I. Wasserman and available from the IEEE Computer Society [Was80]. Readers with web browsers may also want to look at the following web page, titled "Programming Language and Compiler Bibliographies."

`http://www.cs.cmu.edu/~mleone/language/bibliographies.html`

However, this tends to have better coverage of recent technical reports, and its coverage of older journal papers and other published material is spotty.

1 Generalities

There are several excellent undergraduate texts on programming languages, including: [PZ96] [Seb96] [Set96] [FWH92] [Hen90] [Kam90] [Wat90] [GJ87] [Mac99] [Ten81]. Graduate texts include: [SK95] [Sta95] [Sch94] [Win93] [Wat91] [Mey90] [Gor88].

Volume B of the *Handbook of Theoretical Computer Science* contains many detailed surveys relevant to formal models and semantics [vL90].

Journals include a substantial coverage of programming languages include *ACM Transactions on Programming Languages and Systems* (abbreviated TOPLAS), *ACM Letters on Programming Languages and Systems* (abbreviated LOPLAS), *ACM SIGPLAN Notices*, *Computer Languages* (Pergamon Press), *Journal of Programming Languages* (Chapman and Hall), *Information and Computation* (formerly *Information and Control* Academic Press), and *Acta Informatica* (Springer-Verlag). Applied areas, such as specification and

*This work was supported in part by the National Science Foundation under Grants CCR-9108654, CCR-9593168, and CCR-9503168.

[1]Naturally these judgements are personal, but it is hoped that authors of that vast bulk of papers that are not cited here will not take offense.

verification have their own journals (e.g., *IEEE Transactions on Software Engineering*, *Science of Computer Programming*, *Formal Aspects of Computing*). Other journals can be found by scanning the bibliography.

Conferences devoted to topics related to programming languages include the Annual ACM Symposium on Principles of Programming Languages (POPL), the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), the ACM SIGPLAN International Conference on Functional Programming (formerly called the Symposium on LISP and Functional Programming, LFP), and the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPOPP). Conferences with a more theoretical bent include the IEEE Annual Symposium on Logic in Computer Science (LICS), Mathematical Foundations of Programming Semantics (MFPS), the European Symposium on Programming (ESOP), the International Colloquium on Automata, Languages, and Programming (ICALP), and the International Symposium on Mathematical Foundations of Computer Science. Conferences not sponsored by the ACM or IEEE are often available in the Springer-Verlag series of Lecture Notes in Computer Science (LNCS). The above list does not include conferences devoted to software engineering (e.g. TAPSOFT) or particular methods (e.g., category theory) where other important work is found.

Both Hoare and Wirth have written guidelines for programming language designers [Hoa80] [Wir74]. Floyd's Turing Award lecture discusses his view of programming languages as capturing paradigms of programming [Flo79]. Another viewpoint is that language design should proceed from semantic principles [Ten77] [Bac78a].

The truism that bad programs can be written in any programming language is described as "Flon's Axiom" in a paper written at the height of the structured programming controversy [Flo75]. Shaw and Wulf point out that language designers can provide reasonable defaults while still giving programmers the ability to change them [SW80].

The idea of the expressiveness of a language is just starting to receive satisfactory formal treatments [Fel90] [Mit91].

2 Semantics

Some texts that cover many different approaches to semantics are: [NN92] [Win93] [SK95].

2.1 Operational Semantics

Landin's SECD machine is defined in his paper "The Mechanical Evaluation of Expressions" [Lan64] (see also Henderson's book [Hen80]).

A more systematic style of operational semantics based on rewrite rules is found in Plotkin's structural operational semantics [Plo77] [Hen90] [Ast91].

Several meta-circular interpreters for variants of LISP are discussed in Steele and Sussman's paper *The Art of the Interpreter* [SS78a]. An excellent and more readily accessible discussion is found in Abelson and Sussman's book [ASS96], which uses Scheme. A more detailed treatment of interpreters is found in [Kam90] [FWH92]. See [KdRB91] for an approach using the CLOS meta-object protocol. More recently, structuring mechanisms for semantics based on monads [Mog90] [Wad92] [Wad97] have lead to more modular structuring of definitional interpreters [Ste94] [LHJ95].

2.2 λ-Calculus

A standard reference on Church's λ-calculus [Chu41] is Barendregt's book [Bar84]. Informal introductions include [Gor88]. For the typed λ-calculus, see [GLT89] or [Mit90b].

The use of λ-calculus for describing programming languages and as the inspiration for programming language design has been investigated by Landin [Lan65] [Lan66].

2.3 Denotational Semantics

A short summary of the denotational approach to programming language semantics [Sco81] can be found in Tennent's article "The Denotational Semantics of Programming Languages" [Ten76]. A survey is found in [Mos90].

Introductory texts include [Gor79], [All86], [Wat91] [Sch94]. An excellent text with mathematical depth is [Gun92]. Standard works on denotational semantics are the books by Stoy [Sto77] and Schmidt [Sch86], both of which offer a comprehensive and mathematical treatment. Schmidt's book [Sch86] can be consulted for references to denotational descriptions of real languages.

To read some of the most technical works, one will need some familiarity with category theory [Lan71] [Gol84] [LS91]; the following are good introductions that emphasize semantic applications [Hoa89] [BW90a] [AL91b] [Pie91] [Wal91] [Gun92].

One can use a typed functional programming language, such as Standard ML, to implement a denotational semantics. Two descriptions of this are [Wat86] [MA89].

Action semantics, an offshoot of denotational semantics, is described in [Wat91] and more fully in [Mos92].

2.4 Axiomatic Semantics

An early presentation of axiomatic semantics is Hoare's paper "An Axiomatic Basis for Computer Programming" [Hoa69].

An example of the use of axiomatic semantics to define a programming language is Hoare and Wirth's axiomatic definition of Pascal [HW73]. A book on axiomatic semantics slanted towards programming language theory (as opposed to verification) is Hesselink's monograph [Hes92]. Foundational material that may help in reading this monograph is found in [DS90], which goes over notational issues as well as the underlying mathematics.

A very introductory tutorial on verification from the software engineering perspective is [LG86, chapter 11]. The idea of developing a proof of a program at the same time the program is being developed has been eloquently advocated by Dijkstra and Gries [Dij76] [Gri81]. More recent treatments in this style advocate a calculational approach [Mor94] [Coh90] [GS94]. See [Al91a] for an introduction that discusses concurrent and distributed programs.

A survey of program verification for imperative programs is [Cou90].

A language designed to support program verification is Euclid [LGH$^+$78] [PHL$^+$77]. Another such language is Alphard [SWL77] [Sha81].

2.5 Algebraic Semantics

Another approach to the semantics of programming languages is the algebraic approach; specifying the behavior of programs instead of directly specifying the function computed. A book-length treatment is [BHK89], which uses tools described in [Kli93]. Examples include [MA86] [BWP87] [PGM90] [SK95, Chapter 12].

3 Type Systems

3.1 Background

The purpose of type checking is nicely summarized by Morris as a mechanism that allows program modules to protect objects from unwanted discovery, modification, and impersonation [Mor73]. Wegbreit's discussion of the extensible language EL1, is also good background [Weg74].

3.2 Polymorphism

An introductory survey of modern polymorphic type systems and research results is Cardelli and Wegner's paper "On Understanding Types, Data Abstraction and Polymorphism" [CW85]. See also [Har84] [Car91] [DT88]; the latter two have much material related to object-oriented programming. A still more recent survey is [Mit90b].

Standard references include Girard's system Fω [Gir71] (see also [Gir86] [GLT89]), and Reynolds independent work [Rey74], sometimes called the Girard-Reynolds second order lambda calculus (or SOL). Modern expositions are found in [MP85] [MH88] [Rey85] [Mit90b] [Car91] [Sch94].

Other kinds of type information may be incorporated into a type system and checked at the same time as types [GL86] [LG88] [OG89].

3.3 Type Inference

Type inference is also sometimes called type reconstruction The basic idea is described in Milner's paper "A Theory of Type Polymorphism in Programming" [Mil78]. A more modern exposition of Milner's ideas is [Car87]. A top-down variation of the standard algorithm is studied in [LY98].

This type inference system is used in the programming language Standard ML [GMW79] [MTH90] [MT91]. Textbooks about programming in Standard ML include [Pau91] [Sok91] [Sta92] [Ull94]. ML also has a very interesting module system [HL94] [Ler94] [Mac84].

An important research problem [HM95] has been extending this type system to handle subtyping [EST95] [Nor99] [OSW99] [PT98] [Pot99].

An extension of this type system to handle static overloading (*ad hoc* polymorphism) is described in[WB89] and further refined in [Jon95] [NP94]. One way to handle mutation in this type system is described in [Har94]. There are also object-oriented extensions to ML [RV98] [RR96] [FR99].

The language Poly [Mat85b] [Mat85a] uses type inference to infer type parameters. An algorithm for inferring operation parameters is described in [CW90]. Other extensions to the basic type inference algorithm are described in [OG89] [TJ94] [Boe89]. Along similar lines, one can use a mix of type annotations and inference to allow polymorphic functions to be passed as arguments [OL96].

The computational complexity of various type checking and type inference problems has also been an active area of research [Wel94] [Sch98] [Hen99]. See [Tiu90] for a survey.

3.4 Type Theory

Type theory, narrowly defined, uses the tools of constructive logic to study type systems such as the above. Logical inference systems can often be translated directly into type systems due to the "Formula as Types" notion or the "Curry-Howard isomorphism" [How80] [GLT89, Chapter 3] [Con89]. Thus much research in type theory lies on the border of mathematics and computer science. Another motivation is to use type information to capture behavioral specifications, thus allowing one to reason about programs in the programming language [NP83] [Dyb90].

After reading the references above, one will still need an introduction to some of the more technical aspects of type theory. Good book-length treatments are [Tho91] and [Sch94]. Those wishing a shorter introduction might try [Rey85] (which is not comprehensive, but is tutorial) [Bac89] (which especially focuses on Martin-Löf style type theory) and [Sce90] (which focuses more on the calculus of constructions). After reading one of these the student may want to read [PDM89] for some practical hints.

In the past, some groups working on type theory have included deBruijn and others working on AUTOMATH [dB80], Martin-Löf's and followers [ML75] [ML82] [Bac89] [BCMS89], Constable's group at Cornell [CZ84], and Coquand and Huet's group [CH88] has also been influential for modern language design.

Some of the latest work on type theory uses linear logic [Gir93] instead of a more standard logic. A linear type system allows a value to be used only once [Bak91] [Mac93] [SBvEP94] [Kob99].

There are also connections between type theory and abstract interpretation [Cou97] [PP98].

3.5 Data Abstraction and Types

A good introductory treatment of the idea of data abstraction is found in [LG86]. A more technically oriented introduction is [CL90]. Another excellent paper is [Coo91], which distinguishes between programming with abstract data types and object-oriented programming.

CLU is a language designed around data abstraction [LSAS77] [LAB$^+$81] [LG86]. It also has an interesting control abstraction and exception handling mechanisms [LS79].

Alphard, designed around the same time as CLU and with many of the same goals, has surprising differences [SWL77] [Sha81].

The language Russell was developed at Cornell to investigate how types can be treated as values. There are many papers that have appeared about Russell, but perhaps the best introduction to the language is the paper "Data Types are Values" [DD85], which can be consulted for other references.

Much recent work involves object-oriented languages. For example, Trellis/Owl [SCB+86] features strong type checking and a declared (i.e., by-name) subtype relation. By contrast many other languages feature structural subtyping, including Emerald [BHJL86] [BHJ+87] [BH90] [BH91] [Car91]. Two excellent books, one by Abadi and Cardelli [AC96] and another by Castagna [Cas97], are a good starting point in this area. Other theoretical work in this area includes the following [BCP96] [Car88b] [Car88a] [AC93] [CMMS94] [CCH+89] [CHC90] [Coo89] [BTGS90] [BCM+93] [BM92] [Bru93] [BCM+93] [PS94] [Bru94] [PT94] [Aba94] [AC94] [AC95] [EST95] [FM98]. (Cardelli has been one of the most active in this area, and most of the literature will cite one of his papers.) For work that directly bears on multimethods (as in CLOS), see [Rey80] [Ghe91a] [Ghe91b] [CGL92] [Cha92] [CGL95] [Cas93] [Cas95] [Cas97] [CL95] [LP99] [MC99]. For adding multimethods to conventional languages see [BC97] [LM98]. For a tutorial discussion of the problems of typing binary methods, see [BCC+95].

Programming languages with separate compilation do some of their type checking at link-time [LAB+81] [Ler94] [Str91]. This interaction of types, separate compilation, linking, and modules has recently been formalized [Car97] [GM99]. Some recent theoretical work been on combining modules and object-oriented programming features [FF99] [FR99] [MC99].

4 Alternative Programming Models

4.1 Functional Programming

A survey of functional programming is [Hud89], which also discusses the language Haskell [HF92] [HJW+92] [Sno92] [Dav92]. Another survey is [Bar90]. There are several good books on functional programming, including [Hen87] [BW88] [Hen80] [SF89] [Oka98]. For an introduction that also treats language implementation issues, see [Pey87]. The articles in [Tur90b] make an interesting introduction to some research topics.

John Backus, one of the designers of Fortran, proposed a new language for functional programming without any names called FP in his Turing Award Lecture [Bac78a]. A functional programming style using a more congenial notation based on Landin's ISWIM [Lan66] is developed in Henderson's book *Functional Programming: Application and Implementation* [Hen80].

One axis of variation in functional languages is between the lazy and strict (eager) languages [Wad96]. ML is eager, but Haskell and its predecessor Miranda [Tur90a] are lazy. Miranda also has an interesting notion of data abstraction.

Recently, various approaches to incorporating state information in a safe way into functional languages have centered around the use of monads [LS97] [Mog90] [Wad92] [Wad97]. There is syntactic support for monads in Haskell. Others are exploring alternatives to monads [CH97] [Kag97] [Ode99] [SBvEP94] [Wad99].

Erlang [AWWV95] is used by Ericsson telephone company in several commercial products.

4.2 Logic Programming

Kowalski's paper "Algorithm = Logic + Control" is a good introduction to logic programming in an idealized setting [Kow79]. A classic textbook on Prolog is the book by Clocksin and Mellish [CM81]; another good text is Sterling and Shapiro's [SS94]. A short description and evaluation of Prolog is found in the paper "The Prolog Phenomenon" [McD80]. A survey is [Apt90].

Several "AI languages" preceded the development of Prolog; for example, Planner [SWC71] and Conniver [MS74].

Much work has focused on concurrent logic programming languages [Sha89], which are perhaps more like CSP than logic programming. An evaluation of the Fifth-Generation project and some history of concurrent logic programming languages is found in [SW93]. A related language is Andorra Prolog [BHW89].

Work on type checking for logic programming languages is surveyed in the collection [Pfe92]. The language λProlog features type inference, type checking, and higher-order programming constructs [Mil90a] [Mil89a] [NM90] [MNPS91].

4.3 Other Declarative Programming Paradigms

Constraint-based languages will probably be important in the future. An early attempt was embodied in Steele and Sussman's work, as described in [SS80]. An overview is given in Leler's book [Lel88]. A survey is found in [vHS+96]. The language CLP($\mathcal{R}$) is a well-known constraint logic programming language [JMSY92].

Some higher-order equational logic languages based on variations of narrowing [Sla74] have started to appear in research languages [Mil91] [Pfe91] [Qia94].

Languages based on term rewriting without logic variables also allow for parallelism. A standard example is OBJ [FGJM85], which also has an interesting module system [Gog84].

4.4 Object-Oriented Programming

A good, but not very technical, introduction to object-oriented concepts is given by Cox [Cox86]; his book also discusses the language Objective-C. A more technical introduction is Budd's book [Bud91b]. Meyer's book on Eiffel also has more technical meat [Mey88], as well as a focus on software engineering concerns. Another fairly complete treatment is given in Goldberg and Robson's book on Smalltalk-80 [GR83]. A graduate-level introduction is [BGHS91].

Descriptions of object-oriented design methods are found in [Boo91] [WBWW90] [dCLF92] [dCF92] [dCLF93]. Both [Boo91] and [dCLF93] have many references. A treatment of object-oriented design that focuses more on C++ is found in [Mul89]. See also the survey by [WBJ90]. Much recent work in this area has focused on design patterns [Joh92] [GHJV95].

A collection of papers is found in [Pet87]. More collections of edited research papers are [SW87] [KL89].

The major annual conferences on object-oriented programming are the *European Conference on Object-Oriented Programming* (ECOOP) and *Object-Oriented Programming Systems, Languages and Applications* (OOPSLA). The ECOOP tends to be more academic, while OOPSLA is more practical. The OOPSLA proceedings have been published as special issues of ACM *SIGPLAN Notices* (November 1986, December 1987 with an addendum in May 1988, and November 1988, and October of the following years). ECOOP and OOPSLA had a joint conference in 1990.

There are now three journals devoted to object-oriented programming. The *Journal of Object-Oriented Programming* (JOOP) is the oldest. Two more academic journals are *Theory and Practice of Object Systems* (TAPOS), and *Object-Oriented Systems*.

The best documented and the cleanest object-oriented programming language is Smalltalk-80 [Ing81] [GR83]. Squeak is a recent dialect [IKM+97].

A more widely used language is C++ [ES90]. Good introductions to C++ include [Str97] and [Lip91].

Multiple inheritance, a feature not found in Smalltalk-80, seems to be quite useful. Snyder's analysis of the design issues involved is insightful [Sny86], although his viewpoint is different from that of most advocates of multiple inheritance.

Semantics of inheritance (as opposed to type theory or semantics of subtyping) appear in [Kam88] [BC90] [CP89]. See [Tai96] for a survey on the notion of inheritance.

My own views on the subject of object-oriented specification, verification, and subtyping can be found in [LW90] [Lea91b] [LW95]. For a contrast, see also [BW90b] [Ame87] [AvdL90] [Mey88] [MOM90] [Coo92] [UR92] [LW94].

The concept of delegation is explored in [Lie86], in Actra [LTP86] [LaL89], and in Ungar's language "Self" [US87].

The contrast between message passing and other kinds of polymorphism is one of binding time. Some relevant semantic models are discussed in [Mit90a] and [CHC90].

A dated survey of the literature on object-oriented programming is [Lea91a].

4.5 Blends of Various Paradigms

Various authors have tried to blend various paradigms. Blends of functional and logic programming are found in [GM86] [JG89]. Blends of imperative and logic programming ideas are found in [Bud91a] [AS98]. Goguen and Meseguer even try to unify everything [GM87].

Another approach is a multiparadigm language. Leda is one example [Bud95].

5 Language Case Studies

The following entries are intended to be selective rather than comprehensive. Instead they are biased towards the most interesting languages and references for the programming language designer. Thus, although COBOL [ANS74] was (is?) the most widely used language on the planet, its influence on programming language design has been small. While the languages discussed below are often obscure, they demonstrate interesting issues in language design.

Those interested in history for its own sake, or in delving further into early languages, should look at the proceedings of the two History of Programming Languages Conferences [Wex78] [Wex93]. The first conference covers the earliest languages, including COBOL, BASIC, and many others not discussed below. Many of the more established languages have their definitions standardized. These are often published by the American National Standards Institute (ANSI), the International Standards Organization (ISO), or the IEEE.

Other resources for case studies include Kamin's book [Kam90], which has several case studies put in a common framework. The "Grand Tour" book by Horowitz has articles about specific languages as well [Hor87]. See also the languages mentioned under the various paradigms above.

5.1 FORTRAN

The first widely used programming language was FORTRAN. See [Bac78b] for a discussion of the history of FORTRAN, what early versions of FORTRAN were like, and early references. The development of FORTRAN IV is discussed in [BH64].

John Backus, who headed the team that developed FORTRAN, later became dissatisfied with the influence that FORTRAN had on programming languages [Bac78a].

5.2 Algol 60

The Algol 60 report is a true classic [NBB+63]. Among other innovations, it introduced the syntax formalism now known as BNF. Despite the precise use of English in the report, Knuth and others were able to find problems with the language definition [Knu67].

5.3 Algol 68

Algol 68 is a direct descendent of Algol. It is a more powerful and more complete language than Algol 60; for example, it has user-defined types, overloading of operators, and mechanisms for parallel processing. The language design is fascinating and bristles with examples of orthogonality (one of many terms coined in the Algol 68 design). The revised report is a forbidding document, which has an innovative formal mechanism for defining the language's semantics [vWMP+77]. Because the revised report is difficult to follow, Tanenbaum's tutorial is probably a better place to start [Tan76]. Those seriously interested in Algol 68 will want to consult [LvdM77].

5.4 C

A popular descendent of Algol 68 is the lower-level language C [KR78]. C represents the best of several languages that support low-level programming while maintaining the portability of the resulting program. The language has recently been standardized, and incorporates several changes from its variant C++ [Str91].

5.5 Algol W, Pascal, Modula-2, and Oberon

Wirth and Hoare's language known as Algol W can be thought of as an improved version of Algol 60 [WH66]. Wirth's language Pascal [Wir71] [JW74] has been enormously popular, attracting detractors [Hab73] and defenders [LD75]. Pascal is, in part, a response to the complexity of Algol 68. All this attention has provoked Wirth to reassessing Pascal [Wir75] and to the design of Modula-2 [Wir85], and Oberon [Wir88]. Oberon has object-oriented features, as does the (non-Wirth) language Modula-3 [CDJ+89] [Nel91].

5.6 Euclid

Euclid is an attempt to improve on Pascal in a different direction [PHL$^+$77] [LGH$^+$78]. Specifically, it attempts to support program verification.

5.7 Ada

Ada was designed by first setting out requirements for the language [Hig78] and then designing and revising a language to meet those requirements [IBH$^+$79] [Ada83] [IBFW91]. Recently, the language was revised to add some object-oriented features [BB$^+$95] [Ada95].

5.8 Java

Java [GJS96] [AG98] is an object-oriented language with a syntax similar to C++. However underneath, it is like Lisp, as it features garbage collection and implicit pointers. It also has reflective features, including dynamic class loaders [LB98]. Although it has a strong, static, and safe [DEK99] [NvO98] type system, because it has a universal type (`Object`) that is a supertype of (almost) all the other types and an operation to dynamically check types (`instanceof`, with checked type casts), it can be programmed as if it were a dynamically-typed language. Several authors have proposed extensions to add parametric polymorphism to the language [BOSW98] [CS98] [MBL97] [OW97] [Tho97b].

5.9 Lisp-like Languages

The original LISP is described in the *LISP 1.5 Programmer's Manual* [MAE$^+$65]. The language has since evolved in many directions. MacLisp is a main-stream dialect that provided many system building tools [Pit83]. The successor to MacLisp is Common Lisp [Ste84] [Ste90]. Unlike most earlier dialects of Lisp, Common Lisp has static scoping.

Scheme was the first dialect of Lisp to emphasize static scoping [SS78b] [ASS96]. See also the references in the *Revised^5 report* [KCE98].

ZetaLisp (as found on Symbolics Lisp Machines) includes the influential *Flavors* mechanism for object-oriented programming [WM80] [SMW84] [Sym84]. The Flavors mechanism evolved into the Common Lisp Object System [Kee89] [Ste90] [Pae93]. The meta-object system of Common Lisp is described in [KdRB91]; it provides a very flexible way to extend the language. The ideas in the meta-object system have led to Aspect-Oriented Programming [KLM$^+$97] that is designed to allow the isolation of concerns that would otherwise be spread throughout a program.

5.10 Snobol, SL5, and Icon

SNOBOL4 is a language for string processing that is unlike any other [GPP71]. Because it is so unstructured, its designers have tried to place its powerful features in a more structured framework. The language SL5, one attempt in this direction, is notable for its flexible procedure mechanism [HG78]. Another descendent of SNOBOL4, Icon, is a more conventional programming language with innovative control structures [GG83].

5.11 APL

Another unconventional language is APL [Ive62] [Ive91]. APL has array processing features without equal and generic operators that can be combined in interesting ways [Ive79]. More references on APL can be found by consulting the proceedings of the yearly *International Conference on APL* (sponsored by the ACM). (In the past this was often just called *APL 83* or some such name.)

The most recent version of APL, APL2, is described in [Bro88] [BC91].

6 Parallel and Distributed Programming Languages

Parallel programming languages are a hot topic of current research, and one with considerable overlap with operating systems, networking, and database systems. Many of the above areas and articles have implications for parallel programming. This area also has several journals and conferences of its own. Some journals that publish programming language related materials but which were not mentioned above include *Distributed Computing*, *IEEE Transactions on Parallel and Distributed Systems*, and *International Journal of Parallel Programming*. See also *ACM SIGOPS Operating Systems Review*. Important conferences not mentioned above include the annual *ACM Symposium on Operating Systems Principles*, *ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming*, and the *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*.

A recent survey is [ST98]. An older, but still good, survey on parallel programming language issues is [AS83]. A survey that focuses on the Linda model appears in [CG89]. (See [BZ91] [BZ92] [GC92] for more work on Linda.) An older survey that focuses on object-oriented aspects is [WKH92], and a more recent survey [BGL98] also discusses distributed object-oriented programs. A survey that focuses on distributed programs is [And91b]. Andrew's textbook on concurrent programming languages also contains material on program verification as well as some survey material [And91a]. Data flow languages are surveyed in [Ack82]. A useful collection of papers is [ST95]. An introduction to concurrent programming techniques is [Sno92].

Actor languages are discussed in [Agh91]. A concurrent version of ML is described in [Rep93]. Some aspects of data parallel programming languages are described in [HS86] [Gri93]. The language Jade is described in [RL92] [RSL93].

Data-parallel languages allow one to split data among processors, and to execute the same code for each. A prominent example is High Performance Fortran [Lov93]. The language Orca combines both data and task parallelism [HBJ98].

Distributed programming is a subarea of parallel programming with its own set of problems [Mul93]. A survey appears in [BST89], which can be consulted for other references. A survey, with a focus on object-orientation is [CC91]. Other sources of references are is the collection of reprints [AS91], and the book [GR93]. Of particular interest are the languages Argus [LS83] [Wei90] and SR [AOC$^+$88] [AO93], The Amoeba distributed operating system also has interesting implications for language designers [TvRvS$^+$90] [TKB92].

Mobile code is a recent research interest. It was featured in Emerald [BHJL86] [BHJ$^+$87] [BH90] [BH91], and of course is part of Java. Some other mobile languages of interest include Obliq [Car95], Mobile UNITY [Rom98], and Distributed Oz [vRHB$^+$97]. A recent survey of the field is [Tho97a].

An older but still heavily used approach to the semantics of concurrent processes is Petri nets [Pet77] [PC92]. A classic reference for the operational semantics of concurrent processes is Milner's book on CCS [Mil89b]. Another widely used approach is Hoare's CSP [Hoa78] [BHR84] [LS84] [Hoa85]. See [Hen88] [BM90] [Mil90b] [MPW92] for more work in the semantics of concurrency, and [LL90] and [Bro91] for work in the semantics of distributed systems.

7 The Future

A (by now a bit dated) summary of research directions for language design is given in [LLM89].

Consumers of programming languages (programmers and language standardization committees) seem to be fairly conservative, and interested more in performance than elegance or expressive power. See [Gab93] for a pessimistic view of what this means for language design.

Acknowledgements

Thanks to my teachers. Thanks especially to Lawrence Flon, who introduced me to the literature on programming languages. Thanks also to Barbara Liskov, David Gifford, and William Weihl, who helped shape my attitudes towards it. Thanks to students of the class Com S 541 at Iowa State University who have given me feedback on the literature recommendations contained herein.

References

[Aba94] Martín Abadi. Baby Modula-3 and a Theory of Objects. *Journal of Functional Programming*, 4(2):249–283, April 1994.

[AC93] Roberto M. Amadio and Luca Cardelli. Subtyping Recursive Types. *ACM Transactions on Programming Languages and Systems*, 15(4):575–631, September 1993.

[AC94] Martín Abadi and Luca Cardelli. A Theory of Primitive Objects — Untyped and First-Order Systems. In Masami Hagiya and John C. Mitchell, editors, *Theoretical Aspects of Computer Software*, volume 789 of *Lecture Notes in Computer Science*, pages 296–320. Springer-Verlag, April 1994.

[AC95] Martín Abadi and Luca Cardelli. On Subtyping and Matching. In Walter Olthoff, editor, *ECOOP '95: Object-Oriented Programming 9th European Conference, Aarhus, Denmark*, number 952 in Lecture Notes in Computer Science, pages 145–167. Springer-Verlag, New York, N.Y., 1995.

[AC96] Martín Abadi and Luca Cardelli. *A Theory of Objects*. Monographs in Computer Science. Springer-Verlag, New York, N.Y., 1996.

[Ack82] W. B. Ackerman. Data Flow Languages. *Computer*, 15(2):15–25, February 1982.

[Ada83] American National Standards Institute. *Reference Manual for the Ada Programming Language*, February 1983. ANSI/MIL-STD 1815A. Also published by Springer-Verlag as LNCS 155.

[Ada95] International Organization for Standardization. *Ada 95 Reference Manual. The Language. The Standard Libraries*, January 1995. ANSI/ISO/IEC-8652:1995.

[AG98] Ken Arnold and James Gosling. *The Java Programming Language*. The Java Series. Addison-Wesley, Reading, MA, second edition, 1998.

[Agh91] Gul Agha. The Structure and Semantics of Actor Languages. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Foundations of Object-Oriented Languages, REX School/Workshop, Noordwijkerhout, The Netherlands, May/June 1990*, volume 489 of *Lecture Notes in Computer Science*, pages 1–59. Springer-Verlag, New York, N.Y., 1991.

[Al91a] Krzysztof R. Apt and Ernst-Rudiger 0lderog. Introduction to Program Verification. In E. J. Neuhold and M. Paul, editors, *Formal Description of Programming Concepts*, IFIP State-of-the-Art Reports, pages 363–429. Springer-Verlag, New York, N.Y., 1991.

[AL91b] Andrea Asperti and Guiseppe Longo. *Categories, Types and Structures*. The MIT Press, Cambridge, Mass, 1991.

[All86] Lloyd Allison. *A Practical Introduction to Denotational Semantics*. Cambridge University Press, New York, N.Y., 1986.

[Ame87] Pierre America. Inheritance and Subtyping in a Parallel Object-Oriented Language. In Jean Bezivin et al., editors, *ECOOP '87, European Conference on Object-Oriented Programming, Paris, France*, pages 234–242, New York, N.Y., June 1987. Springer-Verlag. Lecture Notes in Computer Science, Volume 276.

[And91a] Gregory R. Andrews. *Concurrent Programming: Principles and Practice*. The Benjamin/Cummings Publishing Company, 1991.

[And91b] Gregory R. Andrews. Paradigms for Process Interaction in Distributed Programs. *ACM Computing Surveys*, 23(1):49–90, March 1991.

[ANS74] American National Standards Institute, New York, N.Y. *American National Standard Programming Language COBOL*, 1974. ANS X3.23-1974.

[AO93] Gregory R. Andrews and Ronald A. Olsson. *The SR Programming Language: Concurrency in Practice.* The Benjamin/Cummings Publishing Company, Redwood City, CA, 1993.

[AOC+88] Gregory R. Andrews, Ronald A. Olsson, Michael Coffin, Irving Elshoff, Kelvin Nilsen, Titus Purdin, and Gregg Townsend. An Overview of the SR Language and Implementation. *ACM Transactions on Programming Languages and Systems*, 10(1):51–86, January 1988.

[Apt90] Krzysztof R. Apt. Logic Programming. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 10, pages 493–574. The MIT Press, New York, N.Y., 1990.

[AS83] Gregory R. Andrews and Fred B. Schneider. Concepts and Notations for Concurrent Programming. *ACM Computing Surveys*, 15(1):3–43, March 1983.

[AS91] Akkihebba L. Ananda and Balasubramaniam Srinivasan. *Distributed Computing Systems: Concepts and Structures.* IEEE Computer Society Press Reprint Collection. IEEE Computer Society Press, Los Alamitos, California, 1991.

[AS98] Krzysztof Apt and Andrea Schaerf. Alma-O: An Imperative Language that Supports Declarative Programming. *ACM Transactions on Programming Languages and Systems*, 20(5):1014–1066, September 1998.

[ASS96] Harold Abelson, Gerald Jay Sussman, and Julie Sussman. *Structure and Interpretation of Computer Programs.* McGraw Hill, Cambridge, Mass., second edition, 1996.

[Ast91] Edigio Astesiano. Inductive and Operational Semantics. In E. J. Neuhold and M. Paul, editors, *Formal Description of Programming Concepts*, IFIP State-of-the-Art Reports, pages 51–136. Springer-Verlag, New York, N.Y., 1991.

[AvdL90] Pierre America and Frank van der Linden. A Parallel Object-Oriented Language with Inheritance and Subtyping. *ACM SIGPLAN Notices*, 25(10):161–168, October 1990. *OOPSLA ECOOP '90 Proceedings*, N. Meyrowitz (editor).

[AWWV95] J. L. Armstrong, M. C. Williams, C. Wikström, and S. R. Virding. *Concurrent Programming in Erlang.* Prentice Hall, 2nd edition edition, 1995.

[Bac78a] John Backus. Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs. *Communications of the ACM*, 21(8):613–641, August 1978.

[Bac78b] John Backus. The History of FORTRAN I, II, and III. *ACM SIGPLAN Notices*, 13(8):165–180, August 1978.

[Bac89] R. C. Backhouse. Constructive Type Theory – An Introduction. In Manfred Broy, editor, *Constructive Methods in Computing Science*, volume F55 of *NATO ASI Series*, pages 9–60. Springer-Verlag, New York, N.Y., 1989.

[Bak91] Henry G. Baker. Lively Linear Lisp — 'Look Ma, No Garbage!'. *ACM SIGPLAN Notices*, 27(8):89–98, August 1991.

[Bar84] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North-Holland Publishing Co., New York, N.Y., 1984. Revised Edition.

[Bar90] H. P. Barendregt. Functional Programming and Lambda Calculus. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 7, pages 321–363. The MIT Press, New York, N.Y., 1990.

[BB+95] John Barnes, Ben Brosgol, et al. Ada 95 Rationale. The Language. The Standard Libraries. Technical report, Intermetrics Inc., 733 Concord Av, Cambridge, MA 02138, January 1995.

[BC90] Gilad Bracha and William Cook. Mixin-Based Inheritance. *ACM SIGPLAN Notices*, 25(10):303–311, October 1990. *OOPSLA ECOOP '90 Proceedings*, N. Meyrowitz (editor).

[BC91] J. A. Brown and H. P. Crowder. APL2: Getting Started. *IBM Systems Journal*, 30(4):433–445, 1991.

[BC97] John Boyland and Giuseppe Castagna. Parasitic Methods: Implementation of Multi-Methods for Java. In *Conference Proceedings of OOPSLA '97, Atlanta*, volume 32(10) of *ACM SIGPLAN Notices*, pages 66–76. ACM, October 1997.

[BCC$^+$95] Kim Bruce, Luca Cardelli, Giuseppe Castagna, The Hopkins Object Group, Gary T. Leavens, and Benjamin Pierce. On Binary Methods. *Theory and Practice of Object Systems*, 1(3):221–242, 1995.

[BCM$^+$93] Kim B. Bruce, Jon Crabtree, Thomas P. Murtagh, Robert van Gent, Allyn Dimock, and Robert Muller. Safe and decidable type checking in an object-oriented language. *ACM SIGPLAN Notices*, 28(10):29–46, October 1993. *OOPSLA '93 Proceedings*, Andreas Paepcke (editor).

[BCMS89] Roland Backhouse, Paul Chisholm, Grant Malcolm, and Erik Saaman. Do-it-Yourself Type Theory. *Formal Aspects of Computing*, 1(1):19–84, January – March 1989.

[BCP96] Kim B. Bruce, Luca Cardelli, and Benjamin C. Pierce. Comparing Object Encodings. In *Invited lecture at Third Workshop on Foundations of Object Oriented Languages (FOOL 3)*, July 1996. Available electronically through `http://www.cs.williams.edu/~kim/FOOL/Abstracts.html`.

[BGHS91] Gordon Blair, John Gallagher, David Hutchison, and Doug Shepherd, editors. *Object-Oriented Languages, Systems and Applications*. Pitman Publishing, London, 1991. ISBN 0-273-03132-5.

[BGL98] Jean-Pierre Briot, Rachid Guerraoui, and Klaus-Peter Lohr. Concurrency and Distribution in Object-Oriented Programming. *ACM Computing Surveys*, 30(3):291–329, September 1998.

[BH64] J. W. Backus and W. P. Heising. FORTRAN. *IEEE Transactions on Electronic Computers*, EC-13(4):382–385, 1964.

[BH90] Andrew P. Black and Norman C. Hutchinson. Typechecking Polymorphism in Emerald. Technical Report TR 90-34, Department of Computer Science; The University of Arizona, Tucson, AZ 85721, December 1990.

[BH91] Andrew P. Black and Norman Hutchinson. Typechecking Polymorphism in Emerald. Technical Report CRL 91/1 (Revised), Digital Equipment Corporation, Cambridge Research Lab, Cambridge, Mass., July 1991.

[BHJ$^+$87] Andrew Black, Norman Hutchinson, Eric Jul, Henry Levy, and Larry Carter. Distribution and Abstract Types in Emerald. *IEEE Transactions on Software Engineering*, SE-13(1):65–76, January 1987.

[BHJL86] Andrew Black, Norman Hutchinson, Eric Jul, and Henry Levy. Object Structure in the Emerald System. *ACM SIGPLAN Notices*, 21(11):78–86, November 1986. OOPSLA '86 Conference Proceedings, Norman Meyrowitz (editor), September 1986, Portland, Oregon.

[BHK89] J. A. Bergstra, J. Heering, and P. Klint. *Algebraic Specification*. ACM Press and Addison-Wesley, 1989.

[BHR84] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A Theory of Communicating Sequential Processes. *Journal of the ACM*, 31(3):560–599, July 1984.

[BHW89] P. Brand, S. Haridi, and D.H.D. Warren. Andorra Prolog. *New Generation Computing*, 7(2-3):109–129, 1989.

[BM90] Bard Bloom and Albert R. Meyer. Experimenting with Process Equivalence. In M. Z. Kwiatkowska, M. W. Shields, and R. M. Thomas, editors, *Semantics for Concurrency, Leicester*, Workshops in Computing, pages 81–95. Springer-Verlag, New York, N.Y., 1990.

[BM92] Kim Bruce and John C. Mitchell. PER models of subtyping, recursive types and higher-order polymorphism. In *Conference Record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 316–327. ACM, January 1992.

[Boe89] Hans-J. Boehm. Type Inference in the Presence of Type Abstraction. *ACM SIGPLAN Notices*, 24(7):192–206, July 1989. Proceedings of the SIGPLAN '89 Conference on Programming Language Design and Implementation, Portland, Oregon, June.

[Boo91] Grady Booch. *Object-Oriented Design: With Applications.* Benjamin Cummings, New York, N.Y., 1991.

[BOSW98] Gilad Bracha, Martin Odersky, David Stoutamire, and Philip Wadler. Making the Future Safe for the Past: Adding Genericity to the Java Programming Language. In *OOPSLA '98 Conference Proceedings*, volume 33(10) of *ACM SIGPLAN Notices*, pages 183–200, October 1998.

[Bro88] James A. Brown. *APL2 at a glance.* Prentice Hall, Englewood Cliffs, N.J., 1988.

[Bro91] Manfred Broy. Formalization of Distributed, Concurrent, Reactive Systems. In E. J. Neuhold and M. Paul, editors, *Formal Description of Programming Concepts*, IFIP State-of-the-Art Reports, pages 319–361. Springer-Verlag, New York, N.Y., 1991.

[Bru93] K. Bruce. Safe Type Checking in a Statically Typed Object-Oriented Programming Language. In *Proc. ACM Symp. on Principles of Programming Languages*, pages 285–298, 1993.

[Bru94] K. B. Bruce. A Paradigmatic Object-Oriented Programming Language: Design, Static Typing and Semantics. *Journal of Functional Programming*, 4(2):127–206, April 1994.

[BST89] Henri E. Bal, Jennifer G. Steiner, and Andrew S. Tanenbaum. Programming Languages for Distributed Computing Systems. *ACM Computing Surveys*, 21(3):261–322, September 1989.

[BTGS90] V. Breazu-Tannen, C. A. Gunter, and A. Scedrov. Computing with Coercions. In *Proceedings of the 1990 ACM Conference on LISP and Functional Programming, Nice, France*, pages 44–60. ACM, June 1990.

[Bud91a] Timothy Budd. Blending Imperative and Relational Programming. *IEEE Software*, 8(1):58–65, January 1991.

[Bud91b] Timothy Budd. *Object-Oriented Programming.* Addison-Wesley, New York, N.Y., 1991.

[Bud95] Timothy A. Budd. *Multiparadigm Programming in LEDA.* Addison-Wesley, New York, N.Y., 1995.

[BW88] Richard J. Bird and Philip Wadler. *Introduction to Functional Programming.* International Series in Computer Science. Prentice-Hall, New York, N.Y., 1988.

[BW90a] Michael Barr and Charles Wells. *Category Theory for Computing Science.* International Series in Computer Science. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1990. ISBN 0-13-120486-6.

[BW90b] Kim B. Bruce and Peter Wegner. An Algebraic Model of Subtype and Inheritance. In Francois Bançilhon and Peter Buneman, editors, *Advances in Database Programming Languages*, pages 75–96. Addison-Wesley, Reading, Mass., August 1990.

[BWP87] Manfred Broy, Martin Wirsing, and Petter Pepper. On the Algebraic Definition of Programming Languages. *ACM Transactions on Programming Languages and Systems*, 9(1):54–99, January 1987.

[BZ91] Paul Butcher and Hussein Zedan. Lucinda–An Overview. *ACM SIGPLAN Notices*, 26(8):90–100, August 1991.

[BZ92] Paul Butcher and Hussein Zedan. Lucinda - A Polymorphic Linda. In J. P. Banatre and D. Le Metayer, editors, *Research Directions in High-Level Parallel Programming Languages, Mont Saint-Michel, France, June 1991, Proceedings*, volume 574 of *Lecture Notes in Computer Science*, pages 126–146. Springer-Verlag, New York, N.Y., 1992.

[Car87] Luca Cardelli. Basic Polymorphic Typechecking. *Science of Computer Programming*, 8(2):147–172, April 1987.

[Car88a] Luca Cardelli. A Semantics of Multiple Inheritance. *Information and Computation*, 76(2/3):138–164, February/March 1988. A revised version of the paper that appeared in the 1984 Semantics of Data Types Symposium, LNCS 173, pages 51–66.

[Car88b] Luca Cardelli. Structural Subtyping and the Notion of Power Type. In *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages, San Diego, Calif.*, pages 70–79. ACM, January 1988.

[Car91] Luca Cardelli. Typeful Programming. In E. J. Neuhold and M. Paul, editors, *Formal Description of Programming Concepts*, IFIP State-of-the-Art Reports, pages 431–507. Springer-Verlag, New York, N.Y., 1991.

[Car95] Luca Cardelli. A language with distributed scope. *Computing Systems*, 8(1):27–59, January 1995. A preliminary version appeared in POPL '95.

[Car97] Luca Cardelli. Program Fragments, Linking, and Modularization. In *Conference Record of POPL 97: The 24TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Paris, France*, pages 266–277–, New York, N.Y., January 1997. ACM.

[Cas93] G. Castagna. A Meta-Language for Typed Object-Oriented Languages. In R. K. Shyamasundar, editor, *Foundations of Software Technology and Theoretical Computer Science*, volume 761 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, October 1993.

[Cas95] Giuseppe Castagna. Covariance and contravariance: conflict without a cause. *ACM Transactions on Programming Languages and Systems*, 17(3):431–447, 1995.

[Cas97] Giuseppe Castagna. *Object-Oriented Programming: A Unified Foundation.* Progress in Theoretical Computer Science. Birkhauser, Boston, 1997.

[CC91] Roger S. Chin and Samuel T. Chanson. Distributed Object-Based Programming Systems. *ACM Computing Surveys*, 23(1):91–124, March 1991.

[CCH$^+$89] Peter Canning, William Cook, Walter Hill, John Mitchell, and Walter Olthoff. F-Bounded Polymorphism for Object-Oriented Programming. In *Fourth International Conference on Functional Programming and Computer Architecture*. ACM, September 1989. Also technical report STL-89-5, from Software Technology Laboratory, Hewlett-Packard Laboratories.

[CDJ$^+$89] Luca Cardelli, Jim Donahue, Mick Jordan, Bill Kalsow, and Greg Nelson. The Modula-3 Type System. In *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas*, pages 202–212. ACM, January 1989.

[CG89] Nicholas Carriero and David Gelernter. How to Write Parallel Programs: A Guide for the Perplexed. *ACM Computing Surveys*, 21(3):323–357, September 1989.

[CGL92] Giuseppe Castagna, Giorgio Ghelli, and Giuseppe Longo. A Calculus for Overloaded Functions with Subtyping. In *ACM Conference on LISP and Functional Programming*, pages 182–192. ACM, June 1992. To appear in *Information and Computation*.

[CGL95] Giuseppe Castagna, Giorgio Ghelli, and Giuseppe Longo. A Calculus for Overloaded Functions with Subtyping. *Information and Computation*, 117(1):115–135, February 1995. A preliminary version appeared in *ACM Conference on LISP and Functional Programming*, June 1992 (pp. 182–192).

[CH88] Thierry Coquand and Gérard Huet. The Calculus of Constructions. *Information and Computation*, 76(2/3):95–120, February/March 1988.

[CH97] Chih-Ping Chen and Paul Hudak. Rolling Your Own Mutable ADT — A Connection Between Linear Types and Monads —. In *Conference Record of POPL 97: The 24TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Paris, France*, pages 54–66, New York, N.Y., January 1997. ACM.

[Cha92] Craig Chambers. Object-Oriented Multi-Methods in Cecil. In Ole Lehrmann Madsen, editor, *ECOOP '92, European Conference on Object-Oriented Programming, Utrecht, The Netherlands*, volume 615 of *Lecture Notes in Computer Science*, pages 33–56. Springer-Verlag, New York, N.Y., 1992.

[CHC90] William R. Cook, Walter L. Hill, and Peter S. Canning. Inheritance is Not Subtyping. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages, San Francisco, California*, pages 125–135, January 1990. Also STL-89-17, Software Technology Laboratory, Hewlett-Packard Laboratories, Palo Alto, Calif., July 1989.

[Chu41] A. Church. *The Calculi of Lambda Conversion*, volume 6 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, N.J., 1941. Reprinted by Klaus Reprint Corp., New York in 1965.

[CL90] Luca Cardelli and Xavier Leroy. Abstract Types and the Dot Notation. Technical Report 56, Digital Equipment Corporation, Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301, March 1990. Order from src-report@src.dec.com.

[CL95] Craig Chambers and Gary T. Leavens. Typechecking and Modules for Multi-Methods. *TOPLAS*, 17(6):805–843, November 1995.

[CM81] W. F. Clocksin and C. S. Mellish. *Programming in Prolog*. Springer-Verlag, New York, N.Y., 1981.

[CMMS94] Luca Cardelli, Simone Martini, John C. Mitchell, and Andre Scedrov. An Extension of System F with Subtyping. *Information and Computation*, 109(1/2):4–56, Feb 1994.

[Coh90] Edward Cohen. *Programming in the 1990s: An Introduction to the Calculation of Programs*. Springer-Verlag, New York, N.Y., 1990.

[Con89] Robert L. Constable. Assigning Meaing to Proofs: a semantic basis for problem solving environments. In Manfred Broy, editor, *Constructive Methods in Computing Science*, volume F55 of *NATO ASI Series*, pages 63–91. Springer-Verlag, New York, N.Y., 1989.

[Coo89] W. R. Cook. A Proposal for Making Eiffel Type-safe. *The Computer Journal*, 32(4):305–311, August 1989.

[Coo91] William R. Cook. Object-Oriented Programming Versus Abstract Data Types. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Foundations of Object-Oriented Languages, REX School/Workshop, Noordwijkerhout, The Netherlands, May/June 1990*, volume 489 of *Lecture Notes in Computer Science*, pages 151–178. Springer-Verlag, New York, N.Y., 1991.

[Coo92] W. R. Cook. Interfaces and Specifications for the Smalltalk-80 Collection Classes. *ACM SIGPLAN Notices*, 27(10):1–15, October 1992. *OOPSLA '92 Proceedings*, Andreas Paepcke (editor).

[Cou90] Patrick Cousot. Methods and Logics for Proving Programs. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 15, pages 841–993. The MIT Press, New York, N.Y., 1990.

[Cou97] Patrick Cousot. Types as Abstract Interpretation. In *Conference Record of POPL 97: The 24TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Paris, France*, pages 316–331, New York, N.Y., January 1997. ACM.

[Cox86] Brad J. Cox. *Object Oriented Programming: an Evolutionary Approach.* Addison-Wesley Publishing Co., Reading, Mass., 1986.

[CP89] William Cook and Jens Palsberg. A Denotational Semantics of Inheritance and its Correctness. *ACM SIGPLAN Notices*, 24(10):433–443, October 1989. OOPSLA '89 Conference Proceedings, Norman Meyerowitz (editor), October 1989, New Orleans, Louisiana.

[CS98] Robert Cartwright and Guy L. Steele Jr. Compatible Genericity with Run-time Types for the Java Programming Language. In *OOPSLA '98 Conference Proceedings*, volume 33(10) of *ACM SIGPLAN Notices*, pages 201–215, October 1998.

[CW85] Luca Cardelli and Peter Wegner. On Understanding Types, Data Abstraction and Polymorphism. *ACM Computing Surveys*, 17(4):471–522, December 1985.

[CW90] G. V. Cormack and A. K. Wright. Type-dependent Parameter Inference. *ACM SIGPLAN Notices*, 25(6):127–136, June 1990. Proceedings of the ACM SIGPLAN '90 Conference on Programming Language Design and Implementation, White Plains, NY.

[CZ84] Robert L. Constable and Daniel R. Zlatin. The Type Theory of PL/CV3. *ACM Transactions on Programming Languages and Systems*, 6(1):94–117, January 1984.

[Dav92] Anthony J. T. Davie. *An Introduction to Functional Programming Systems Using Haskell.* Cambridge Computer Science Texts. Cambridge University Press, New York, N.Y., 1992.

[dB80] N. G. de Bruijn. A Survey of the Project AUTOMATH. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 579–606. Academic Press, Inc., New York, N.Y., 1980.

[dCF92] Dennis de Champeaux and Penelope Faure. A Comparative Study of object-oriented analysis methods. *Journal of Object-Oriented Programming*, 5(1):21–33, March 1992.

[dCLF92] Dennis de Champeaux, Doug Lea, and Penelope Faure. The Process of Object-Oriented Design. *ACM SIGPLAN Notices*, 27(10):45–62, October 1992. *OOPSLA '92 Proceedings*, Andreas Papecke (editor).

[dCLF93] Dennis de Champeaux, Doug Lea, and Penelope Faure. *Object Oriented System Development.* Addison-Wesley Publishing Co., Mass,, 1993.

[DD85] James Donahue and Alan Demers. Data Types are Values. *ACM Transactions on Programming Languages and Systems*, 7(3):426–445, July 1985.

[DEK99] Sophia Drossopoulou, Susan Eisenbach, and Sarfraz Khurshid. Is the Java Type System Sound? *Theory and Practice of Object Systems*, 5(1):3–24, 1999.

[Dij76] Edsger W. Dijkstra. *A Discipline of Programming.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1976.

[DS90] Edsger W. Dijkstra and Carel S. Scholten. *Predicate Calculus and program semantics.* Springer-Verlag, NY, 1990.

[DT88] Scott Danforth and Chris Tomlinson. Type Theories and Object-Oriented Programming. *ACM Computing Surveys*, 20(1):29–72, March 1988.

[Dyb90] Peter Dybjer. Comparing Integrated and External Logics of Functional Programs. *Science of Computer Programming*, 14(1):59–79, June 1990.

[ES90] Margaret A. Ellis and Bjarne Stroustrup. *The Annotated C++ Reference Manual.* Addison-Wesley Publishing Co., Reading, Mass., 1990.

[EST95] J. Eifrig, S. Smith, and V. Trifonov. Sound polymorphic type inference for objects. In *OOPSLA '95 Conference Proceedings*, volume 30(10) of *ACM SIGPLAN Notices*, pages 169–184, 1995.

[Fel90] Matthias Felleisen. On the Expressive Power of Programming Languages. In N. Jones, editor, *ESOP '90 3rd European Symposium on Programming, Copenhagen, Denmark*, volume 432 of *Lecture Notes in Computer Science*, pages 134–151. Springer-Verlag, New York, N.Y., May 1990.

[FF99] Robert Bruce Findler and Matthew Flatt. Modular Object-Oriented Programming with Units and Mixins. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming (ICFP '98)*, volume 34(1) of *ACM SIGPLAN Notices*, pages 94–104. ACM, June 1999.

[FGJM85] Kokichi Futatsugi, Joseph A. Goguen, Jean-Pierre Jouannaud, and Jose Meseguer. Principles of OBJ2. In *Conference Record of the Twelfth Annual ACM Symposium on Principles of Programming Languages*, pages 52–66. ACM, January 1985.

[Flo75] Lawrence Flon. On Research in Structured Programming. *ACM SIGPLAN Notices*, 10(10):16–17, October 1975.

[Flo79] Robert W. Floyd. The Paradigms of Programming. *Communications of the ACM*, 22(8):455–460, August 1979.

[FM98] Kathleen Fischer and John C. Mitchell. On the Relationship Between Classes, Objects and Data Abstraction. *Theory and Practice of Object Systems*, 4(1):3–25, 1998.

[FR99] Kathleen Fischer and John Reppy. The design of a class mechanism for Moby. *ACM SIGPLAN Notices*, 34(5):37–49, May 1999. Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI).

[FWH92] Daniel P. Friedman, Mitchell Wand, and Christopher T. Haynes. *Essentials of Programming Languages.* McGraw-Hill Book Co., New York, N.Y., 1992.

[Gab93] Richard P. Gabriel. The end of history and the last programming language. *Journal of Object-Oriented Programming*, 6(4):90–94, July 1993.

[GC92] David Gelernter and Nicholas Carriero. Coordination Languages and their Significance. *Communications of the ACM*, 35(2):97–107, February 1992.

[GG83] Ralph E. Griswold and Madge T. Griswold. *The Icon Programming Language.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1983.

[Ghe91a] Giorgio Ghelli. Modelling Features of Object-Oriented Languages in Second Order Functional Languages with Subtypes. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Foundations of Object-Oriented Languages, REX School/Workshop, Noordwijkerhout, The Netherlands, May/June 1990*, volume 489 of *Lecture Notes in Computer Science*, pages 311–340. Springer-Verlag, New York, N.Y., 1991.

[Ghe91b] Giorgio Ghelli. A Static Type System for Message Passing. *ACM SIGPLAN Notices*, 26(11):129–145, November 1991. OOPSLA '91 Conference Proceedings, Andreas Paepcke (editor), October 1991, Phoenix, Arizona.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, Mass., 1995.

[Gir71] Jean-Yves Girard. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In *Proceedings 2nd Scandinavian Logic Symposium*, pages 63–92, Amsterdam, 1971. North-Holland.

[Gir86] J. Y. Girard. The System **F** of variable types, fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

[Gir93] Jean-Yves Girard. Linear Logic: A Survey. In F. L. Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*, volume 94 of *NATO ASI Series. Series F : Computer and System Sciences*, pages 63–112. Springer-Verlag, New York, N.Y., 1993.

[GJ87] Carlo Ghezzi and Mehdi Jazayeri. *Programming Language Concepts 2/E.* John Wiley and Sons, New York, N.Y., 1987.

[GJS96] James Gosling, Bill Joy, and Guy Steele. *The Java Language Specification.* The Java Series. Addison-Wesley, Reading, MA, 1996.

[GL86] David K. Gifford and John M. Lucassen. Integrating Functional and Imperative Programming. In *ACM Conference on LISP and Functional Programming*, pages 28–38. ACM, August 1986.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types.* Cambridge University Press, New York, N.Y., 1989.

[GM86] Joseph A. Goguen and José Meseguer. Eqlog: Equality, Types, and Generic Modules for Logic Programming. In Douglas DeGroot and Gary Lindstrom, editors, *Functional and Logic Programming*, pages 295–363. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1986. An earlier version appeared in the *Journal of Logic Programming*, 1984, Volume 1, Number 2, Pages 179-209.

[GM87] Joseph A. Goguen and José Meseguer. Unifying Functional, Object-Oriented and Relational Programming with Logical Semantics. In Bruce Shriver and Peter Wegner, editors, *Research Directions in Object-Oriented Programming*, pages 417–477. The MIT Press, Cambridge, Mass., 1987.

[GM99] Neal Glew and Greg Morrisett. Type-Safe Linking and Modular Assembly Language. In *Conference Record of POPL 99: The 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, Texas*, pages 250–261, New York, N.Y., January 1999. ACM.

[GMW79] Michael J. Gordon, Robin Milner, and Christopher P. Wadsworth. *Edinburgh LCF*, volume 78 of *Lecture Notes in Computer Science.* Springer-Verlag, New York, N.Y., 1979. The second author is listed on the cover as Arthur J. Milner, which is clearly a mistake.

[Gog84] Joseph A. Goguen. Parameterized Programming. *IEEE Transactions on Software Engineering*, SE-10(5):528–543, September 1984.

[Gol84] R. Goldblatt. *Topoi: The Categorical Analysis of Logic (Revised Edition)*, volume 98 of *Studies in Logic and the Foundations of Mathematics.* North-Holland, New York, N.Y., 1984.

[Gor79] Michael J. C. Gordon. *The Denotational Description of Programming Languages.* Springer-Verlag, New York, N.Y., 1979.

[Gor88] Michael J. C. Gordon. *Programming Language Theory and its Implementation.* Prentice Hall International Series in Computer Science. Prentice-Hall, Inc., New York, N.Y., 1988.

[GPP71] R. E. Griswold, J. F. Poage, and I. P. Polonsky. *The SNOBOL4 Programming Language (second edition).* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1971.

[GR83] Adele Goldberg and David Robson. *Smalltalk-80, The Language and its Implementation.* Addison-Wesley Publishing Co., Reading, Mass., 1983.

[GR93] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufman, 1993.

[Gri81] David Gries. *The Science of Programming.* Springer-Verlag, New York, N.Y., 1981.

[Gri93] Andrew S. Grimshaw. Easy-to-Use Object-Oriented Parallel Processing with Mentat. *IEEE Computer*, 26(5):39–51, May 1993.

[GS94] David Gries and Fred B. Schneider. *A Logical Approach to Discrete Math.* Texts and Monographs in Computer Science. Springer-Verlag, New York, N.Y., 1994.

[Gun92] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques.* Foundations of Computing. The MIT Press, Cambridge, Mass., 1992.

[Hab73] A. N. Habermann. Critical Comments on the Programming Language Pascal. *Acta Informatica*, 3(1):47–57, 1973.

[Har84] D. M. Harland. *Polymorphic Programming Languages: Design and Implementation.* John Wiley and Sons, New York, N.Y., 1984.

[Har94] Robert Harper. A Simplified Account of Polymorphic References. *Information Processing Letters*, 51:201–206, 1994.

[HBJ98] Saniya Ben Hassan, Henri E. Bal, and Ceriel J. H. Jacobs. A Task- and Data-Parallel Programming Language Based on Shared Objects. *ACM Transactions on Programming Languages and Systems*, 20(6):1131–1170, November 1998.

[Hen80] Peter Henderson. *Functional Programming: Application and Implementation.* International Series in Computer Science. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1980.

[Hen87] Martin C. Henson. *Elements of Functional Languages.* Blackwell Scientific Publications, Oxford, England, 1987.

[Hen88] Matthew Hennessy. *Algebraic Theory of Processes.* The MIT Press, Cambridge, Mass., 1988.

[Hen90] Matthew Hennessy. *The Semantics of Programming Languages: an Elementary Introduction using Structural Operational Semantics.* John Wiley and Sons, New York, N.Y., 1990.

[Hen99] Fritz Henglein. Breaking Through the n^3 Barrier: Faster Object Type Inference. *Theory and Practice of Object Systems*, 5(1):57–72, 1999.

[Hes92] Wim H. Hesselink. *Programs, Recursion, and Unbounded Choice*, volume 27 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, New York, N.Y., 1992.

[HF92] Paul Hudak and Joseph H. Fasel. A Gentle Introduction to Haskell. *ACM SIGPLAN Notices*, 27(5), May 1992.

[HG78] David R. Hanson and Ralph E. Griswold. The SL5 Procedure Mechanism. *Communications of the ACM*, 21(5):392–400, May 1978.

[Hig78] High Order Language Working Group, Department of Defense. Department of Defense Requirements for High Order Computer Programming Languages: Steelman. Technical report, U. S. Department of Defense, June 1978.

[HJW$^+$92] Paul Hudak, Simon Peyton Jones, Philip Wadler, et al. Report on the Programming Language Haskell: A Non-strict, Purely Functional Language, version 1.2. *ACM SIGPLAN Notices*, 27(5), May 1992.

[HL94] Robert Harper and Mark Lillibridge. A Type-Theoretic Approach to Higher-Order Modules with Sharing. In *Conference Record of POPL '94: 21ST ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, Oregon*, pages 123–137, New York, N.Y., January 1994. ACM.

[HM95] My Hoang and John C. Mitchell. Lower bounds on type inference with subtypes. In *Conference Record of POPL '95: 22nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Francisco, Calif.*, pages 176–185, New York, N.Y., January 1995. ACM.

[Hoa69] C. A. R. Hoare. An Axiomatic Basis for Computer Programming. *Communications of the ACM*, 12(10):576–583, October 1969.

[Hoa78] C. A. R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, 21(8):666–677, August 1978.

[Hoa80] C. A. R. Hoare. Hints on Programming Language Design. In Anthony I. Wasserman, editor, *Tutorial Programming Language Design*, pages 43–52. IEEE, October 1980.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1985.

[Hoa89] C. A. R. Hoare. Notes on an Approach to Category Theory for Computer Scientists. In Manfred Broy, editor, *Constructive Methods in Computing Science*, volume F55 of *NATO ASI Series*, pages 245–305. Springer-Verlag, New York, N.Y., 1989.

[Hor87] Ellis Horowitz. *Programming Languages: A Grand Tour (Third Edition).* Computer Science Press, Rockville, Maryland, 1987.

[How80] W. A. Howard. The Formulae-as-Types notion of Construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, Inc., New York, N.Y., 1980.

[HS86] W. Daniel Hillis and Guy L. Steele Jr. Data Parallel Algorithms. *Communications of the ACM*, 29(12):1170–1183, December 1986.

[Hud89] Paul Hudak. Conception, Evolution, and Application of Functional Programming Languages. *ACM Computing Surveys*, 21(3):359–411, September 1989.

[HW73] C. A. R. Hoare and N. Wirth. An Axiomatic Definition of the Programming Language Pascal. *Acta Informatica*, 2(4):335–355, 1973.

[IBFW91] J. Ichbiah, J. Barnes, R. Firth, and M. Woodger. *Rationale for the Design of the Ada Programming Language.* Cambridge University Press, New York, N.Y., 1991. ISBN 0-521-39267-5.

[IBH$^+$79] J. D. Ichbiah, J. G. P. Barnes, J. C. Heliard, B. Krieg-Brueckner, O. Roubine, and B. A. Wichmann. Reference Manual and Rationale for the Ada Programming Language. *ACM SIGPLAN Notices*, 14(6), June 1979. This version of the language is now obsolete, but the rationale (part B) is still valuable.

[IKM$^+$97] Dan Ingalls, Ted Kaehler, John Maloney, Scott Wallace, and Alan Kay. Back to the Future: The Story of Squeak, A Practical Smalltalk Written in Itself. In *Conference Proceedings of OOPSLA '97, Atlanta*, volume 32(10) of *ACM SIGPLAN Notices*, pages 318–326. ACM, October 1997.

[Ing81] D. H. H. Ingalls. Design Principles Behind Smalltalk. *BYTE*, 6(8):286–298, August 1981.

[Ive62] K. Iverson. *A Programming Language.* John Wiley and Sons, New York, N.Y., 1962.

[Ive79] Kenneth E. Iverson. Operators. *ACM Transactions on Programming Languages and Systems*, 1(2):161–176, October 1979.

[Ive91] Kenneth E. Iverson. A Personal view of APL. *IBM Systems Journal*, 30(4):582–593, 1991.

[JG89] Bharat Jayaraman and Gopal Gupta. EqL: The Language and Its Implementation. *IEEE Transactions on Software Engineering*, 15(6):771–779, June 1989.

[JMSY92] Joxan Jaffar, Spiro Michaylov, Peter J. Stuckey, and Roland H. C. Yap. The CLP($\mathcal{R}$) Language and System. *ACM Transactions on Programming Languages and Systems*, 14(3):339–395, July 1992.

[Joh92] Ralph E. Johnson. Documenting Frameworks using Patterns. *ACM SIGPLAN Notices*, 27(10):63–76, October 1992. *OOPSLA '92 Proceedings*, Andreas Paepcke (editor).

[Jon95] Mark P. Jones. A system of constructor classes: overloading and implicit higher-order polymorphism. *Journal of Functional Programming*, 5(1):1–35, Jan 1995. An earlier version appeared in FPCA '93.

[JW74] Kathleen Jensen and Niklaus Wirth. *PASCAL User Manual and Report (second edition).* Springer-Verlag, New York, N.Y., 1974.

[Kag97] Koji Kagawa. Compositional References for Stateful Functional Programming. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming (ICFP '97)*, volume 32(8) of *ACM SIGPLAN Notices*, pages 217–226. ACM, August 1997.

[Kam88] Samuel Kamin. Inheritance in Smalltalk-80: A Denotational Definition. In *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages, San Diego, Calif.*, pages 80–87. ACM, January 1988.

[Kam90] Samuel N. Kamin. *Programming Languages: An Interpreter-Based Approach.* Addison-Wesley Publishing Co., Reading, Mass., 1990.

[KCE98] Richard Kelsey, William Clinger, and Jonathan Rees (Editors). Revised^5 Report on the Algorithmic Language Scheme. *ACM SIGPLAN Notices*, 33(9):26–76, September 1998.

[KdRB91] Gregor Kiczales, Jim des Rivieres, and Daniel G. Bobrow. *The Art of the Metaobject Protocol.* The MIT Press, Cambridge, Mass., 1991.

[Kee89] Sonya E. Keene. *Object-Oriented Programming in Common Lisp.* Addison Wesley, Reading, Mass., 1989.

[KL89] Won Kim and Frederick H. Lochovsky, editors. *Object-Oriented Concepts, Databases, and Applications.* Addison-Wesley Publishing Co., Reading, Mass., 1989.

[Kli93] Paul Klint. A Meta-Environment for Generating Programming Environments. *ACM Transactions on Software Engineering and Methodology*, 2(2):176–201, April 1993.

[KLM$^+$97] Gregor Kiczales, John Lamping, Anurag Menhdhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-Oriented Programming. In Mehmet Akşit and Satoshi Matsuoka, editors, *ECOOP '97 — Object-Oriented Programming 11th European Conference, Jyväskylä, Finland*, volume 1241 of *Lecture Notes in Computer Science*, pages 220–242. Springer-Verlag, New York, N.Y., June 1997.

[Knu67] Donald E. Knuth. The Remaining Trouble Spots in Algol 60. *Communications of the ACM*, 10(1):611–617, October 1967.

[Kob99] Naoki Kobayashi. Quasi-Linear Types. In *Conference Record of POPL 99: The 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, Texas*, pages 29–42, New York, N.Y., January 1999. ACM.

[Kow79] Robert Kowalski. Algorithm = Logic + Control. *Communications of the ACM*, 22(7):424–435, July 1979.

[KR78] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1978.

[LAB$^+$81] Barbara Liskov, Russell Atkinson, Toby Bloom, Eliot Moss, J. Craig Schaffert, Robert Scheifler, and Alan Snyder. *CLU Reference Manual*, volume 114 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, N.Y., 1981.

[LaL89] Wilf R. LaLonde. Designing Families of Data Types Using Exemplars. *ACM Transactions on Programming Languages and Systems*, 11(2):212–248, April 1989.

[Lan64] P. J. Landin. The Mechanical Evaluation of Expressions. *Computer Journal*, 6:308–320, 1964. See also Landin's paper "A Lambda-Calculus Approach" in *Advances in Programming and Non-Numerical Computation*, L. Fox (ed.), Pergamon Press, Oxford, 1966.

[Lan65] P. J. Landin. A Correspondence Algol 60 and Church's Lambda Notation. *Communications of the ACM*, 8:89–101, 158–165, 1965.

[Lan66] P. J. Landin. The Next 700 Programming Languages. *Communications of the ACM*, 9(3):157–166, March 1966.

[Lan71] Saunders Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, N.Y., 1971.

[LB98] Sheng Liang and Gilad Bracha. Ownership Types for Flexible Alias Protection. In *OOPSLA '98 Conference Proceedings*, volume 33(10) of *ACM SIGPLAN Notices*, pages 36–47. ACM, October 1998.

[LD75] O. Lecarme and P. Desjardins. Reply to a paper by A. N. Habermann on the Programming Language Pascal. *Acta Informatica*, 4(3):231–243, 1975. An earlier version appeared in ACM SIGPLAN Notices, October, 1974.

[Lea91a] Gary T. Leavens. Introduction to the Literature on Object-Oriented Design, Programming, and Languages. *OOPS Messenger*, 2(4), October 1991.

[Lea91b] Gary T. Leavens. Modular Specification and Verification of Object-Oriented Programs. *IEEE Software*, 8(4):72–80, July 1991.

[Lel88] Wm Leler. *Constraint Programming Languages: Their Specification and Generation*. Addison-Wesley Publishing Co., Reading, Mass., 1988.

[Ler94] Xavier Leroy. Manifest types, modules, and separate compilation. In *Conference Record of POPL '94: 21ST ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, Oregon*, pages 109–122, New York, N.Y., January 1994. ACM.

[LG86] Barbara Liskov and John Guttag. *Abstraction and Specification in Program Development*. The MIT Press, Cambridge, Mass., 1986.

[LG88] John M. Lucassen and David K. Gifford. Polymorphic Effect Systems. In *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages, San Diego, Calif.*, pages 47–57. ACM, January 1988.

[LGH$^+$78] R. L. London, J. V. Guttag, J. J. Horning, B. W. Lampson, J. G. Mitchell, and G. J. Popek. Proof Rules for the Programming Language Euclid. *Acta Informatica*, 10(1):1–26, 1978.

[LHJ95] Sheng Liang, Paul Hudak, and Mark Jones. Monad Transformers and Modular Interpreters. In *Conference Record of POPL '94: 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Francisco, California*, pages 333–343. ACM, January 1995.

[Lie86] Henry Lieberman. Using Prototypical Objects to Implement Shared Behavior in Object Oriented Systems. *ACM SIGPLAN Notices*, 21(11):214–223, November 1986. OOPSLA '86 Conference Proceedings, Norman Meyrowitz (editor), September 1986, Portland, Oregon.

[Lip91] Stanley B. Lippman. *C++ Primer: 2nd Edition.* Addison-Wesley Publishing Co., Reading, Mass., 1991.

[LL90] Leslie Lamport and Nancy Lynch. Distributed Computing: Models and Methods. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 19, pages 1157–1199. The MIT Press, New York, N.Y., 1990.

[LLM89] Gary Lindstrom, Barbara Liskov, and David MacQueen. Critical Research Directions in Programming Languages. *ACM SIGPLAN Notices*, 24(11):10–25, November 1989.

[LM98] Gary T. Leavens and Todd D. Millstein. Multiple Dispatch as Dispatch on Tuples. In *OOPSLA '98 Conference Proceedings*, volume 33(10) of *ACM SIGPLAN Notices*, pages 374–387, October 1998.

[Lov93] D. B. Loveman. High Performance Fortran. *IEEE Parallel and Distributed Technology: Systems and Applications*, 1(1):25–42, 1993.

[LP99] Gary T. Leavens and Don Pigozzi. Class-Based and Algebraic Models of Objects. In Rance Cleaveland, Michael Mislove, and Philip Mulry, editors, *US—Brazil Joint Workshops on the Formal Foundations of Software Systems*, volume 14 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1999. http://www.elsevier.nl/locate/entcs/volume14.html.

[LS79] Barbara H. Liskov and Alan Snyder. Exception Handling in CLU. *IEEE Transactions on Software Engineering*, SE-5(6):546–558, November 1979.

[LS83] Barbara Liskov and Robert Scheifler. Guardians and Actions: Linguistic Support for Robust, Distributed Programs. *ACM Transactions on Programming Languages and Systems*, 5(3):381–404, July 1983.

[LS84] Leslie Lamport and Fred B. Schneider. The "Hoare Logic" of CSP and All That. *ACM Transactions on Programming Languages and Systems*, 6(2):281–296, April 1984.

[LS91] F. W. Lawvere and Stephen H. Schanuel. *Conceptual Mathematics: a first introduction to categories.* Buffalo Workshop Press, Buffalo, NY, 1991.

[LS97] John Launchbury and Amr Sabry. Monadic State: Axiomatization and Type Safety. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming (ICFP '97)*, volume 32(8) of *ACM SIGPLAN Notices*, pages 227–238. ACM, August 1997.

[LSAS77] Barbara Liskov, Alan Snyder, Russell Atkinson, and Craig Schaffert. Abstraction Mechanisms in CLU. *Communications of the ACM*, 20(8):564–576, August 1977.

[LTP86] Wilf R. LaLonde, Dave A. Thomas, and John R. Pugh. An Exemplar Based Smalltalk. *ACM SIGPLAN Notices*, 21(11):322–330, November 1986. OOPSLA '86 Conference Proceedings, Norman Meyrowitz (editor), September 1986, Portland, Oregon.

[LvdM77] C. H. Lindsey and S. G. van der Meulen. *Informal Introduction to ALGOL 68 (revised edition).* North-Holland Publishing Co., New York, N.Y., 1977.

[LW90] Gary T. Leavens and William E. Weihl. Reasoning about Object-oriented Programs that use Subtypes (extended abstract). In N. Meyrowitz, editor, *OOPSLA ECOOP '90 Proceedings*, volume 25(10) of *ACM SIGPLAN Notices*, pages 212–223. ACM, October 1990.

[LW94] Barbara Liskov and Jeannette Wing. A Behavioral Notion of Subtyping. *ACM Transactions on Programming Languages and Systems*, 16(6):1811–1841, November 1994.

[LW95] Gary T. Leavens and William E. Weihl. Specification and Verification of Object-Oriented Programs Using Supertype Abstraction. *Acta Informatica*, 32(8):705–778, November 1995.

[LY98] Oukseh Lee and Kwangkeun Yi. Proofs about a Folklore Let-Polymorphic Type Inference Algorithm. *ACM Transactions on Programming Languages and Systems*, 20(4):707–723, July 1998.

[MA86] Ernest G. Manes and Michael A. Arbib. *Algebraic Approaches to Program Semantics.* Springer-Verlag, New York, N.Y., 1986.

[MA89] C. McDonald and L. Allison. Denotational Semantics of a Command Interpreter and their Implementation in Standard ML. *The Computer Journal*, 32(5):422–431, October 1989.

[Mac84] David MacQueen. Modules for Standard ML. In *Proceedings of the Symposium on LISP and Functional Programming, Austin, Texas*, pages 198–207. ACM, August 1984.

[Mac93] Ian Mackie. Lilac — A Functional Programming Language Based on Linear Logic. *Journal of Functional Programming*, 4(4):395–433, 1993.

[Mac99] Bruce J. MacLennan. *Principles of Programming Languages.* Oxford University Press, New York, N.Y., third edition, 1999.

[MAE$^+$65] John McCarthy, Paul W. Abrahams, Daniel J. Edwards, Timothy P. Hart, and Michael I. Levin. *LISP 1.5 Programmer's Manual.* The MIT Press, Cambridge, Mass., 1965.

[Mat85a] David C. J. Matthews. An overview of the Poly Programming Language. In *Persistence and Data Types: Papers for the Appin Workshop*, pages 265–274. Universities of Glasgow and St. Andrews, Departments of Computer Science, August 1985. Persistent Programming Research Report 16.

[Mat85b] David C. J. Matthews. Poly Manual. *ACM SIGPLAN Notices*, 20(9):52–76, September 1985.

[MBL97] Andrew C. Myers, Joseph A. Bank, and Barbara Liskov. Parameterized Types for Java. In *Conference Record of POPL '97: The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 132–145, New York, N.Y., January 1997. ACM.

[MC99] Todd Millstein and Craig Chambers. Modular Statically Typed Multimethods. In Rachid Guerraoui, editor, *ECOOP '99 — Object-Oriented Programming 13th European Conference, Lisbon Portugal*, volume 1628 of *Lecture Notes in Computer Science*, pages 279–303. Springer-Verlag, New York, N.Y., June 1999.

[McD80] Drew McDermott. The Prolog Phenomenon. *ACM SIGART Newsletter*, pages 16–20, July 1980. Number 72.

[Mey88] Bertrand Meyer. *Object-oriented Software Construction.* Prentice Hall, New York, N.Y., 1988.

[Mey90] Bertrand Meyer. *Introduction to the Theory of Programming Languages.* International Series in Computer Science. Prentice Hall, New York, N.Y., 1990.

[MH88] John C. Mitchell and Robert Harper. The Essence of ML. In *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages, San Diego, Calif.*, pages 28–46. ACM, January 1988.

[Mil78] Robin Milner. A Theory of Type Polymorphism in Programming. *Journal of Computer and System Sciences*, 17(3):348–375, December 1978.

[Mil89a] Dale Miller. Abstractions in Logic Programs. In P. Odifreddi, editor, *Logic and Computer Science*, pages 329–359. Academic Press, 1989.

[Mil89b] Robin Milner. *Communication and Concurrency.* International Series in Computer Science. Prentice Hall, New York, N.Y., 1989.

[Mil90a] Dale Miller. A Logic Programming Language with Lambda-Abstraction Function Variables, and Simple Unification. In Peter Schroeder-Heister, editor, *Extensions of Logic Programming, International Workshop, Tubingen, FRG, December, 1989*, volume 475 of *Lecture Notes in Computer Science*, pages 253–282. Springer-Verlag, New York, N.Y., 1990.

[Mil90b] Robin Milner. Operational and Algebraic Semantics of Concurrent Processes. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 19, pages 1201–1242. The MIT Press, New York, N.Y., 1990.

[Mil91] Dale Miller. A Logic Programming Language with Lambda-Abstraction, Function Variables, and Simple Unification. *Journal of Logic and Computation*, 1(4):497–536, 1991.

[Mit90a] John C. Mitchell. Toward a typed foundation for method specialization and inheritance. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages, San Francisco, Calif.*, pages 109–124. ACM, January 1990.

[Mit90b] John C. Mitchell. Type systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 8, pages 365–458. North-Holland, New York, N.Y., 1990.

[Mit91] John C. Mitchell. On abstraction and the expressive power of programming languages. In *Conference on Theoretical Aspects of Computer Software, Sendi Japan*, September 1991.

[ML75] P. Martin-Löf. An Intuitionistic Theory of Types: Predictive Part. In H. E. Rose and J. C. Sheperdson, editors, *Logic Colloquium '73*, volume 80 of *Studies in Logic*, pages 73–118. North-Holland Publishing Co., New York, N.Y., 1975.

[ML82] Per Martin-Löf. Constructive Mathematics and Computer Programming. In L. J. Cohen et al., editors, *Logic, Methodology, and Philosophy of Science VI (Proceedings of the Sixth International Congress; Hannover, 1979)*, volume 104 of *Studies in Logic and the Foundations of Mathematics*, pages 153–175. North Holland, Amsterdam, 1982.

[MNPS91] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform Proofs as a Foundation for Logic Programming. *Annals of Pure and Applied Logic*, 51(1-2):125–158, March 1991.

[Mog90] Eugenio Moggi. An abstract view of programming languages. Technical Report ECS-LFCS-90-113, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, Edinburgh, EH9 3JZ, 1990.

[MOM90] Narciso Marti-Oliet and Jose Meseguer. Inclusions and Subtypes. Technical Report SRI-CSL-90-16, Computer Science Laboratory, SRI International, 333 Ravenswood Ave., Menlo Park, Calif., December 1990.

[Mor73] James H. Morris, Jr. Protection in Programming Languages. *Communications of the ACM*, 16(1):15–21, January 1973.

[Mor94] Carroll Morgan. *Programming from Specifications: Second Edition*. Prentice Hall International, Hempstead, UK, 1994.

[Mos90] Peter D. Mosses. Denotational Semantics. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 11, pages 577–631. The MIT Press, New York, N.Y., 1990.

[Mos92] Peter D. Mosses. *Action Semantics*, volume 26 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, New York, N.Y., 1992.

[MP85] John C. Mitchell and Gordon D. Plotkin. Abstract Types have Existential Type. In *Conference Record of the 12th Annual ACM Symposium on Principles of Programming Languages, New Orleans, Louisana*, pages 37–51. ACM, January 1985.

[MPW92] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, (Parts I and II). *Information and Computation*, 100:1–77, 1992.

[MS74] Drew V. McDermott and Gerald Jay Sussman. The CONNIVER Reference Manual. AI Memo 295a, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, January 1974.

[MT91] Robin Milner and Mads Tofte. *Commentary on Standard ML.* The MIT Press, Cambridge, Mass., 1991.

[MTH90] Robin Milner, Mads Tofte, and Robert Harper. *The Definition of Standard ML.* The MIT Press, Cambridge, Mass., 1990.

[Mul89] Mark Mullin. *Object Oriented Program Design With Examples in C++.* Addison-Wesley Publishing Co., Reading, Mass., 1989.

[Mul93] Sape Mullender, editor. *Distributed Systems.* Addison-Wesley, New York, N.Y., second edition, 1993.

[NBB$^+$63] Peter Naur, J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, and M. Woodger. Revised Report on the Algorithmic Language ALGOL 60. *Communications of the ACM*, 1(17), January 1963.

[Nel91] Greg Nelson. *Systems Programming with Modula-3.* Prentice-Hall, 1991.

[NM90] G. Nadathur and D. Miller. Higher-Order Horn Clauses. *Journal of the ACM*, 37(4):777–814, October 1990.

[NN92] F. Nielson and H.R. Neilson. *Semantics with Applications - A Formal Introduction.* John Wiley and Sons, New York, N.Y., 1992.

[Nor99] Johan Nordlander. Pragmatic Subtyping in Polymorphic Languages. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming (ICFP '98)*, volume 34(1) of *ACM SIGPLAN Notices*, pages 216–227. ACM, June 1999.

[NP83] Bengt Nordström and Kent Peterson. Types and Specifications. In R. E. A. Mason, editor, *Information Processing 83*, pages 915–920. Elsevier Science Publishers B.V. (North-Holland), September 1983. Proceedings of the IFIP 9th World Computer Congress, Paris, France.

[NP94] Tobias Nipkov and Christian Prehofer. Type Reconstruction for Type Classes. *Journal of Functional Programming*, 5(2):201–224, April 1994.

[NvO98] Tobias Nipkow and David von Oheimb. Java$_{light}$ is Type-Safe — Definitely. In *Conference Record of POPL 98: The 25TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Diego, California*, pages 161–170, New York, N.Y., January 1998. ACM.

[Ode99] Martin Odersky. Programming with Variable Functions. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming (ICFP '98)*, volume 34(1) of *ACM SIGPLAN Notices*, pages 105–116. ACM, June 1999.

[OG89] James William O'Toole and David K. Gifford. Type Reconstruction with First-Class Polymorphic Values. *ACM SIGPLAN Notices*, 24(7):207–217, July 1989. Proceedings of the SIGPLAN '89 Conference on Programming Language Design and Implementation, Portland, Oregon, June.

[Oka98] Chris Okasaki. *Purely Functional Data Structures.* Cambridge University Press, Cambridge, UK, 1998.

[OL96] Martin Odersky and Konstantin Läufer. Putting Type Annotations to Work. In *Conference Record of POPL '96: The 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, St. Petersberg Beach, Florida*, pages 54–67, New York, N.Y., January 1996. ACM.

[OSW99] Martin Odersky, Martin Sulzmann, and Martin Wehr. Type Inference with Constrained Types. *Theory and Practice of Object Systems*, 5(1):35–55, 1999.

[OW97] Martin Odersky and Philip Wadler. Pizza into Java: Translating Theory into Practice. In *Conference Record of POPL '97: The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 146–159, New York, N.Y., January 1997. ACM.

[Pae93] Andreas Paepcke. *Object-Oriented Programming: The CLOS Perspective.* The MIT Press, 1993.

[Pau91] Laurence C. Paulson. *ML for the Working Programmer.* Cambridge University Press, New York, N.Y., 1991.

[PC92] Yiannis E. Papelis and Thomas L. Casavant. Specification and Analysis of Parallel/Distributed Software and Systems by using Petri Nets with Transition Enabling Functions. *IEEE Transactions on Software Engineering*, 18(3):252–261, March 1992.

[PDM89] Benjamin Pierce, Scott Dietzen, and Spiro Michaylov. Programming in Higher-Order Typed Lambda-Calculi. Technical Report CMU-CS-89-111, School of Computer Science, Carnegie Mellon University, March 1989.

[Pet77] J. L. Peterson. Petri Nets. *ACM Computing Surveys*, 9(3):221–252, September 1977.

[Pet87] Gerald E. Peterson, editor. *Tutorial: Object-Oriented Computing.* IEEE Computer Society Press, Los Angeles, Calif., 1987. Volume 1: concepts; volume 2: implementations.

[Pey87] S. L. Peyton Jones. *The Implementation of Functional Programming Languages.* Prentice-Hall, Englewood Cliffs, N.J., 1987.

[Pfe91] Frank Pfenning. Logic Programming in the LF Logical Framework. In Gérard Huet and Gordon Plotkin, editors, *Logical Frameworks*, pages 149–181. Cambridge University Press, 1991.

[Pfe92] Frank Pfenning, editor. *Types in Logic Programming.* Logic Programming Series. The MIT Press, Cambridge, Mass., 1992.

[PGM90] S. Prasad, A. Giacalone, and P. Mishra. Operational and algebraic semantics of Facile: A symmetric integration of concurrent and functional programming. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *LNCS*, pages 765–780. Springer-Verlag, July 1990.

[PHL$^+$77] G. J. Popek, J. J. Horning, B. W. Lampson, J. G. Mitchell, and R. L. London. Notes on the Design of Euclid. *ACM SIGPLAN Notices*, 12(3):11–18, March 1977. Proceedings of an ACM Conference on Language Design for Reliable Software, Raliegh, North Carolina, March, 1977.

[Pie91] Benjamin C. Pierce. *Basic Category Theory for Computer Scientists.* The MIT Press, Cambridge, Mass, 1991.

[Pit83] Kent M. Pitman. The Revised MacLisp Manual. Technical Report TR-295, Massachusetts Institute of Technology, Laboratory for Computer Science, May 1983.

[Plo77] G. D. Plotkin. LCF Considered as a Programming Language. *Theoretical Computer Science*, 5:223–255, 1977.

[Pot99] François Pottier. A Framework for Type Inference with Subtyping. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming (ICFP '98)*, volume 34(1) of *ACM SIGPLAN Notices*, pages 228–238. ACM, June 1999.

[PP98] Jens Palsberg and Christina Pavlopoulou. From Polyvariant Flow Information to Intersection and Union Types. In *Conference Record of POPL 98: The 25TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Diego, California*, pages 197–208, New York, N.Y., January 1998. ACM.

[PS94] Jens Palsberg and Michael I. Schwartzbach. *Object-Oriented Type Systems*. John Wiley and sons, 1994.

[PT94] Benjamin C. Pierce and David N. Turner. Simple Type-Theoretic Foundations for Object-Oriented Programming. *Journal of Functional Programming*, 4(2):207–248, April 1994. A preliminary version appeared in POPL 1993.

[PT98] Benjamin C. Pierce and David N. Turner. Local Type Inference. In *Conference Record of POPL 98: The 25TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Diego, California*, pages 252–265, New York, N.Y., January 1998. ACM.

[PZ96] Terrence W. Pratt and Marvin V. Zelkowitz. *Programming Languages: Design and Implementation*. Prentice-Hall, Englewood Cliffs, NJ, third edition edition, 1996.

[Qia94] Zhenyu Qian. Higher-Order Equational Logic Programming. In *Conference Record of POPL '94: 21ST ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, Oregon*, pages 254–267, New York, N.Y., January 1994. ACM.

[Rep93] John H. Reppey. Concurrent Programming with Events - The Concurrent ML Manual. Technical report, AT&T Bell Labs, February 1993. Available by anonymous ftp from research.att.com.

[Rey74] J. C. Reynolds. Towards a Theory of Type Structure. In *Programming Symposium, Proceedings, Colloque sur la Programmation, Paris, April 1974*, volume 19 of *Lecture Notes in Computer Science*, pages 408–425. Springer-Verlag, New York, N.Y., 1974.

[Rey80] John C. Reynolds. Using Category Theory to Design Implicit Conversions and Generic Operators. In Neil D. Jones, editor, *Semantics-Directed Compiler Generation, Proceedings of a Workshop, Aarhus, Denmark*, volume 94 of *Lecture Notes in Computer Science*, pages 211–258. Springer-Verlag, January 1980.

[Rey85] John C. Reynolds. Three Approaches to Type Structure. In Hartmut Ehrig, Christiane Floyd, Maurice Nivat, and James Thatcher, editors, *Mathematical Foundations of Software Development, Proceedings of the International Joint Conference on Theory and Practice of Software Development (TAPSOFT), Berlin. Volume 1: Colloquium on Trees in Algebra and Programming (CAAP '85)*, volume 185 of *Lecture Notes in Computer Science*, pages 97–138. Springer-Verlag, New York, N.Y., March 1985.

[RL92] Martin C. Rinard and Monica S. Lam. Semantic Foundations of Jade. In *Conference Record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 105–118. ACM, January 1992.

[Rom98] Peter J. McCann Gruia-Catalin Roman. Compositional Programming Abstractions for Mobile Computing. *IEEE Transactions on Software Engineering*, 24(2):97–110, February 1998.

[RR96] John Reppy and Jon Riecke. Simple Objects for Standard ML. *ACM SIGPLAN Notices*, 31(5):171–180, May 1996. Proceedings of the 1996 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI).

[RSL93] Martin C. Rinard, Daniel J. Scales, and Monica S. Lam. Jade: A High Level Machine Independent Language for Parallel Programming. *COMPUTER*, 26(6):28–38, June 1993.

[RV98] Didier Rémy and Jérôme Vouillon. Objective ML: An Effective Object-Oriented Extension of ML. *Theory and Practice of Object Systems*, 4(1):27–52, 1998.

[SBvEP94] S. Smetsers, E. Barendsen, M. v. Eekelen, and R. Plasmeijer. Guaranteeing Safe Destructive Updates Through a Type System with Uniqueness Information for Graphs. *Lecture Notes in Computer Science*, 776:358–379, 1994.

[SCB$^+$86] Craig Schaffert, Topher Cooper, Bruce Bullis, Mike Kilian, and Carrie Wilpolt. An Introduction to Trellis/Owl. *ACM SIGPLAN Notices*, 21(11):9–16, November 1986. OOPSLA '86 Conference Proceedings, Norman Meyrowitz (editor), September 1986, Portland, Oregon.

[Sce90] Andre Scedrov. A Guide to Polymorphic Types. In P. Odifreddi, editor, *Logic and Computer Science*, volume 31 of *APIC Series*, pages 387–420. Academic Press, New York, N.Y., 1990.

[Sch86] David A. Schmidt. *Denotational Semantics: A Methodology for Language Development.* Allyn and Bacon, Inc., Boston, Mass., 1986.

[Sch94] David A. Schmidt. *The Structure of Typed Programming Languages.* Foundations of Computing Series. MIT Press, Cambridge, Mass., 1994.

[Sch98] Aleksy Schubert. Second-order unification and type inference for Church-style polymorphism. In *Conference Record of POPL 98: The 25TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Diego, California*, pages 279–288, New York, N.Y., January 1998. ACM.

[Sco81] Dana Scott. Lectures on a Mathematical Theory of Computation. Technical Monograph PRG-19, Oxford University Computing Laboratory, Programming Research Group, 1981. Appears in Theoretical foundations of programming methodology : lecture notes of an international summer school, directed by F.L. Bauer, E.W. Dijkstra, and C.A.R. Hoare (Ridel, 1982).

[Seb96] Robert W. Sebesta. *Concepts of Programming Languages.* Benjamin/Cummings, Redwood City, Calif., third edition, 1996.

[Set96] Ravi Sethi. *Programming Languages: Concepts and Constructs.* Addison-Wesley, Reading, Mass., second edition, 1996.

[SF89] George Springer and Daniel P. Friedman. *Scheme and the Art of Programming.* McGraw-Hill, New York, N.Y., 1989.

[Sha81] Mary Shaw. *ALPHARD: Form and Content.* Springer-Verlag, New York, N.Y., 1981.

[Sha89] Ehud Shapiro. The Family of Concurrent Logic Programming Languages. *ACM Computing Surveys*, 21(3):413–510, September 1989.

[SK95] Kenneth Slonneger and Barry L. Kurtz. *Formal Syntax and Semantics of Programming Languages.* Addison-Wesley, New York, N.Y., 1995.

[Sla74] James R. Slagle. Automated Theorem-Proving for Theories with Simplifiers, Commutativity, and Associativity. *Journal of the ACM*, 21(4):622–642, October 1974.

[SMW84] Richard Stallman, David Moon, and Daniel Weinreb. *Lisp Machine Manual (sixth edition).* Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, Mass., June 1984.

[Sno92] C. R. Snow. *Concurrent Programming*, volume 26 of *Cambridge Computer Science Texts.* Cambridge University Press, New York, N.Y., 1992.

[Sny86] Alan Snyder. Encapsulation and Inheritance in Object-Oriented Programming Languages. *ACM SIGPLAN Notices*, 21(11):38–45, November 1986. OOPSLA '86 Conference Proceedings, Norman Meyrowitz (editor), September 1986, Portland, Oregon.

[Sok91] S. Sokolowski. *Applicative High Order Programming.* Chapman and Hall Computing Series. Chapman & Hall Computing, New York, N.Y., 1991.

[SS78a] Guy Lewis Steele Jr. and Gerald Jay Sussman. The Art of the Interpreter or, The Modularity Complex (Parts Zero, One, and Two). AI Memo 453, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, May 1978.

[SS78b] Guy Lewis Steele Jr. and Gerald Jay Sussman. Revised Report on SCHEME A Dialect of LISP. AI Memo 452, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, January 1978.

[SS80] G. Sussman and G. Steele. Constraints: a Language for Expressing Almost-Hierarchical Descriptions. *Artificial Intelligence*, 14:1–39, 1980.

[SS94] Leon Sterling and Ehud Shapiro. *The Art of Prolog.* The MIT Press, Cambridge, Mass., second edition, 1994.

[ST95] David B. Skillicorn and Domenico Talia. *Programming Languages for Parallel Processing.* IEEE Computer Society Press, 1995.

[ST98] David B. Skillicorn and Domenico Talia. Models and Languages for Parallel Computation. *ACM Computing Surveys*, 30(2):123–169, June 1998.

[Sta92] Ryan Stanisfer. *ML Primer.* Prentice Hall, Englewood Cliffs, NJ, 1992.

[Sta95] Ryan Stanisfer. *The Study of Programming Languages.* Prentice Hall, Englewood Cliffs, NJ, 1995.

[Ste84] Guy L. Steele Jr. *Common LISP: The Language.* Digital Press, Burlington, Mass., 1984.

[Ste90] Guy L. Steele Jr. *Common LISP: The Language.* Digital Press, Bedford, Mass., second edition, 1990.

[Ste94] Guy L. Steele, Jr. Building Interpreters by Composing Monads. In *Conference Record of POPL '94: 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, Oregon*, pages 472–492. ACM, January 1994.

[Sto77] J. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.* The MIT Press, Cambridge, Mass., 1977.

[Str91] Bjarne Stroustrup. *The C++ Programming Language: Second Edition.* Addison-Wesley Publishing Co., Reading, Mass., 1991.

[Str97] Bjarne Stroustrup. *The C++ Programming Language: Third Edition.* Addison-Wesley Publishing Co., Reading, Mass., 1997.

[SW80] Mary Shaw and William A. Wulf. Toward Relaxing Assumptions in Languages and Their Implementations. *ACM SIGPLAN Notices*, 15(3):45–61, March 1980.

[SW87] Bruce Shriver and Peter Wegner, editors. *Research Directions in Object-Oriented Programming.* The MIT Press, Cambridge, Mass., 1987.

[SW93] Ehud Shapiro and David H.D. Warren. The Fifth Generation Project: Personal Perspectives. *Communications of the ACM*, 36(3):46–48, March 1993.

[SWC71] Gerald Jay Sussman, Terry Winograd, and Eugene Charniak. Micro-PLANNER Reference Manual. AI Memo 203A, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, December 1971.

[SWL77] Mary Shaw, William A. Wulf, and R. L. London. Abstraction and Verification in Alphard: Defining and Specifying Iteration and Generators. *Communications of the ACM*, 20(8):553–564, August 1977.

[Sym84] Symbolics, Inc. *Lisp Machine Manual.* Cambridge, Mass., March 1984. Eight volumes.

[Tai96] Antero Taivalsaari. On the Notion of Inheritance. *ACM Computing Surveys*, 28(3):438–479, September 1996.

[Tan76] Andrew S. Tanenbaum. A Tutorial on Algol 68. *ACM Computing Surveys*, 8(2), June 1976.

[Ten76] R. D. Tennent. The Denotational Semantics of Programming Languages. *Communications of the ACM*, 19:437–453, August 1976.

[Ten77] R. D. Tennent. Language Design Methods Based on Semantic Principles. *Acta Informatica*, 8:97–112, 1977.

[Ten81] R. D. Tennent. *Principles of Programming Languages.* Prentice-Hall International, Englewood Cliffs, N.J., 1981.

[Tho91] Simon Thompson. *Type Theory and Functional Programming.* International Computer Science Series. Addison-Wesley Publishing Co., 1991.

[Tho97a] Tommy Thorn. Programming Languages for Mobile Code. *ACM Computing Surveys*, 29(3):213–239, September 1997.

[Tho97b] Kresten Krab Thorup. Genericity in Java with Virtual Types. In Mehmet Akşit and Satoshi Matsuoka, editors, *ECOOP '97 — Object-Oriented Programming 11th European Conference, Jyväskylä, Finland*, volume 1241 of *Lecture Notes in Computer Science*, pages 444–471. Springer-Verlag, New York, N.Y., June 1997.

[Tiu90] Jerzy Tiuryn. Type Inference Probelms: A Survey. In B. Rovan, editor, *Mathematical Foundations of Computer Science 1990, Banskà Bystrica, Czechoslovakia*, volume 452 of *Lecture Notes in Computer Science*, pages 105–120. Springer-Verlag, New York, N.Y., 1990.

[TJ94] Jean-Pierre Talpin and Pierre Jouvelot. The Type and Effect Discipline. *Information and Computation*, 111(2):245–296, June 1994.

[TKB92] Andrew S. Tanenbaum, M. Frans Kaashoek, and Henri E. Bal. Parallel Programming Using Shared Objects and Broadcasting. *Computer*, 25(8):10–19, August 1992.

[Tur90a] David A. Turner. An Overview of Miranda. In David A. Turner, editor, *Research Topics in Functional Programming*, University of Texas at Austin Year of Programming Series, pages 1–16. Addison-Wesley Publishing Co., New York, N.Y., 1990.

[Tur90b] David A. Turner, editor. *Research Topics in Functional Programming.* University of Texas at Austin Year of Programming Series. Addison-Wesley Publishing Co., New York, N.Y., 1990.

[TvRvS$^+$90] Andrew S. Tanenbaum, Robert van Renesse, Hans van Staveren, Gregory J. Sharp, Sape J. Mullender, Jack Jansen, and Guido van Rossum. Experience with the Amoeba Distributed Operating System. *Communications of the ACM*, 33(12):46–63, December 1990.

[Ull94] Jeffry D. Ullman. *Elements of ML Programming.* Prentice Hall, Englewood Cliffs, NJ, 1994.

[UR92] Mark Utting and Ken Robinson. Modular Reasoning in an Object-Oriented Refinement Calculus. In R. S. Bird, C. C. Morgan, and J. C. P. Woodcock, editors, *Mathematics of Program Construction, Second International Conference, Oxford, U.K., June/July*, volume 669 of *Lecture Notes in Computer Science*, pages 344–367. Springer-Verlag, New York, N.Y., 1992.

[US87] David Ungar and Randall B. Smith. Self: The Power of Simplicity. *ACM SIGPLAN Notices*, 22(12):227–241, December 1987. OOPSLA '87 Conference Proceedings, Norman Meyrowitz (editor), October 1987, Orlando, Florida.

[vHS$^+$96] Pascal van Hentenryck, Vijay Saraswat, et al. Strategic Directions in Object-Oriented Programming. *ACM Computing Surveys*, 28(4):701–726, December 1996.

[vL90] Jan van Leeuwen. *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics. The MIT Press, New York, N.Y., 1990.

[vRHB+97] Peter van Roy, Seif Haridi, Per Brand, Gert Smolka, Michael Mehl, and Ralf Scheidhauer. Mobile Objects in Distributed Oz. *ACM Transactions on Programming Languages and Systems*, 19(5):804–851, September 1997.

[vWMP+77] A. van Wijngaarden, B. J. Mailloux, J. E. L. Peck, C. H. Koster, M. Sintzoff, C. H. Lindsey, L. G. L. T. Meertens, and R. G. Fisker. Revised Report on the Algorithmic Language ALGOL 68. *ACM SIGPLAN Notices*, 12(5):1–70, 1977. This has also been published by Springer-Verlag, New York, N. Y., and in Acta Informatica, volume 5, pages 1-236 (1975).

[Wad92] Philip Wadler. The Essence of Functional Programming. In *Conference Record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 1–14. ACM, January 1992.

[Wad96] Philip Wadler. Lazy Versus Strict. *ACM Computing Surveys*, 28(2):318–320, June 1996.

[Wad97] Philip Wadler. How to declare an Imperative. *ACM Computing Surveys*, 29(3):240–263, September 1997.

[Wad99] Philip Wadler. The marriage of effects and monads. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming (ICFP '98)*, volume 34(1) of *ACM SIGPLAN Notices*, pages 63–74. ACM, June 1999.

[Wal91] R. F. C. Walters. *Categories and Computer Science*, volume 28 of *Cambridge Computer Science Texts*. Cambridge University Press, New York, N.Y., 1991.

[Was80] Anthony I. Wasserman. *TUTORIAL Programming Language Design*. IEEE Computer Society Press, Los Alamitos, Calif., 1980. Initally presented at Compsac80, The IEEE Computer Society's Fourth International Computer Software & Applications Conference, October 27-31, 1980. The IEEE catalog number is EHO 164-4.

[Wat86] D. A. Watt. Executable Denotational Semantics. *Software: Practice and Experience*, 16(1):13–43, 1986.

[Wat90] David A. Watt. *Programming Language Concepts and Paradigms*. Prentice Hall International Series in Computer Science. Prentice-Hall, New York, N.Y., 1990.

[Wat91] David A. Watt. *Programming Language Syntax and Semantics*. Prentice Hall International Series in Computer Science. Prentice-Hall, New York, N.Y., 1991.

[WB89] Philip Wadler and Stephen Blott. How to make ad-hoc Polymorphism less ad hoc. In *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas*, pages 60–76. ACM, January 1989.

[WBJ90] Rebecca J. Wirfs-Brock and Ralph E. Johnson. Surveying Current Research in Object-Oriented Design. *Communications of the ACM*, 33(9):105–124, September 1990.

[WBWW90] Rebecca Wirfs-Brock, Brian Wilkerson, and Lauren Wiener. *Designing Object-Oriented Software*. Prentice-Hall, Englewood Cliffs, NJ 07632, 1990.

[Weg74] Ben Wegbreit. The Treatment of Data Types in EL1. *Communications of the ACM*, 17(5):251–264, May 1974.

[Wei90] William E. Weihl. Linguistic Support for Atomic Data Types. *ACM Transactions on Programming Languages and Systems*, 12(2):178–202, April 1990.

[Wel94] J. B. Wells. Typability and Type Checking in the Second-Order λ-Calculus Are Equivalent and Undecidable. In *Proceedings Ninth Annual IEEE Symposium on Logic in Computer Science, Paris*, pages 176–185, 1994.

[Wex78] Richard L. Wexelblat, editor. *Preprints — History of Programming Languages Conference, Los Angeles, California*. ACM, June 1978. *ACM SIGPLAN Notices*, Volume 13, Number 8, August 1978.

[Wex93] Richard L. Wexelblat, editor. *ACM SIGPLAN History of Programming Languages Conference (HOPL II), Preprints, Cambridge, MA, USA*. ACM, March 1993. *ACM SIGPLAN Notices*, Volume 28, Number 3.

[WH66] N. Wirth and C. A. R. Hoare. A Contribution to the development of ALGOL. *Communications of the ACM*, 9(6):413–432, June 1966.

[Win93] Glynn Winskel. *The Formal Semantics of Programming Languages*. Foundations of Computer Science Series. The MIT Press, Cambridge, Mass., 1993.

[Wir71] N. Wirth. The Programming Language Pascal. *Acta Informatica*, 1(1):35–63, 1971.

[Wir74] Niklaus Wirth. On the Design of Programming Languages. In *Information Processing 74*, pages 386–393, New York, N.Y., 1974. North-Holland Publishing Co.

[Wir75] Niklaus Wirth. An Assessment of the Programming Language Pascal. *IEEE Transactions on Software Engineering*, pages 192–198, June 1975.

[Wir85] Niklaus Wirth. *Programming in Modula-2 (3rd corrected edition)*. Springer-Verlag, New York, N.Y., 1985.

[Wir88] N. Wirth. Type Extensions. *ACM Transactions on Programming Languages and Systems*, 10(2):204–214, April 1988.

[WKH92] Barbara B. Wyatt, Krishna Kavi, and Steve Hufnagel. Parallelism in Object-Oriented Languages: A Survey. *IEEE Software*, 9(6):56–66, November 1992.

[WM80] Daniel Weinreb and David Moon. Flavors: Message Passing in the Lisp Machine. AI Memo 602, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, November 1980.