AN ANNOTATED BIBLIOGRAPHY ON THE CONSTRUCTION OF COMPILERS

BY

BARY W. POLLACK

STAN-(X-249-71
DECEMBER 1971

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY
An Annotated Bibliography on the Construction of Compilers*

1971       Bary W. Pollack
            Computer Science Department
            Stanford University

This bibliography is divided into 9 sections:
1. General Information on Compiling Techniques
2. Syntax- and Base-Directed Parsing
3. Parsing in General
4. Resource Allocation
5. Errors - Detection and Correction
6. Compiler Implementation in General
7. Details of Compiler Construction
8. Additional Topics
9. Miscellaneous Related References

Within each section the entries are alphabetical by author. Keywords describing the entry will be found for each entry set off by pound signs (#). Some amount of cross-referencing has been done; e.g., entries which fall into Section 3 as well as Section 7 will generally be found in both sections. However, entries will be found listed only under the principle or first author's name.

Computing Review citations are given following the annotation when available.

*This research was supported by the Atomic Energy Commission, Project SU-326F23.

1. GENERAL INFORMATION ON COMPILERS TECHNIQUES

1. 1 Abrahams, P. W.
Symbol manipulation languages.
Academic Press, N. Y.
# languages #

1. 2 Anonymous.
Philosophies for efficient processor construction.
ICC Dull, I, 2 (July 1962), 85-89.
# processors #
CR 4536.

1. 3 Parton, R. S.
A critical review of the state of the programming art.
# compilers #
"This is an overview of programming which includes several pages outlining compiler-writing techniques and the problems involved. It is a good, short introduction to the field of compiler writing."
CR 6842.

1. 4 Burkhardt, W. H.
Universal programming languages and processors: a brief survey and new concepts.
# language, compilers #
"This paper surveys the general concepts behind the compiling systems which are being developed or are currently in use."
CR 12747.

1. 5 Cheatham, T. F.
The architecture of compilers.
# compiler #

1. 6 Cheatham, T. F., and Sattley, K.
Syntax-directed compiling.
# syntax directed, compiler #
"This paper is a discussion of a top-down recognizer, for a syntax-directed compiler. Extensive examples are given."
CR 6304.
CO?¶PILER CCNSTRUCTTON TECHNIQUES

ANNOTATED BIBLIOGRAPHY

1. 7  Cocke, 3. . and Schwartz, J. T.
Programming languages and their compilers: preliminary notes.
2d rev. version.
New York, Courant Institute of Mathematical Sciences,
"This lengthy work describes in detail the workings of
several compilers. It is one of the most comprehensive
works of its type currently available. The work includes
two comprehensive bibliographies as well."

1. 8  Davis, R. M.
Programming language processors.
Academic Press, N. Y.
"This is one of the best overall summaries of the subject of
language processors. It is lengthy, well-written and covers
the topic both in depth and breadth."

1. 9  Elgot, C. C., and Robinson, A.
Random access stored-program machines, an approach to
programming languages.
"A class of machine models is introduced as a basis for
discussion. Address modification is discussed and the
relationship between problem-oriented languages and machine
languages is considered."
CR 8657.

1. 10  Feldman, J., and Gries, D.
Translator writing systems.
Comm ACM 11, 2 (Feb 1968), 77-113.
"This paper surveys critically the research efforts put into
automating compiler writing. The paper includes the formal.
study of syntax and its application to translator writing,
various approaches to automating semantic aspects of
translator writing and other related topics such as the
formal study of semantics, etc."
CR 14729.

1. 11  Floyd, R. W.
The syntax of programming languages--a survey,
"This article is a survey of the use of syntax in
programming languages. The paper discusses major problems
in finding efficient analyzers and fully satisfactory formal
grammars for programming languages."
1. 12 Foster, J. M.
Automatic syntactic analysis.

>This short monograph presents an excellent overview of the subjects of grammars, parsing, and syntactic analysis. The author covers top-down and bottom-up parsing, universal parsing methods, transition matrices, precedence grammars as well as several other important topics.

1. 13 Garwick, J. V.
The definition of programming languages by their compilers,
In Formal Language Description Languages for Computer Programming, T. B. Steel, Jr., (Ed.), North Holland Publishing Co., Amsterdam, (1966), 139-147.

>language, compiler

1. 14 Garwick, J. V.
The definition of programming languages by the compiler.

>languages, compilers

1. 15 Genuys, F., (Ed).
Programming languages, a NATO advanced study institute summer school.

>languages, compilers

1. 16 Glass, R. L.
An elementary discussion of compiler/interpreter writing.
Computing Surveys 1, 1 (Mar 1969), 06-77.

>compiler, interpreter

>An excellent overview of the problems involved in the implementation of compilers is presented and interpreters is presented.

1. 17 Good, I. J.
Number of possible strategies when writing compilers.
Comm ACM 11, 7 (July 1968), 474-474.

>compiling

>The author gives a mathematical formula for the number of strategies given K programming languages and J compilers, (J < K).
1. 18
Gorn, S.
Specification languages for mechanical languages and their processors, a baker's dozen.
# language, syntax #
"The author presents 13 languages, including the natural languages, Backus Normal Form, trees, incidence matrices and Turing machines. These languages provide different points of view of the same problem and aid the the clarification of problems in different ways."
CR 11417.

1. 19
, Gorn, S.
Mechanical pragmatics: a time-motion study of a miniature mechanical linguistic system;
# syntax, language #
"This article goes with the author's earlier '... a Baker's Dozen.' paper. A miniature object language and its syntax are created and then modified to demonstrate their relationship."

1. 20
Gorn, S.
Some basic terminology connected with mechanical languages and their processors.
# language #
"This article gives some terminology considered important by the author. A table summarizing the terms is given."

1. 21
Halpern, M.
Foundations of the case for natural language programming,
IEEE Spectrum (Mar 1967), 140-149.
Proc AFIPS 1966 PJCC, Vol 29, 639-649,
# languages #
"This paper is an attempt to clear away many misconceptions regarding the debate over-whether or not natural language is suitable for programming. The author is admittedly in favor of natural language programming."
CR 11511, 11935.

1. 22
. Harrison, M. C.
Data-structures and programming.
Courant Institute of Math. Sciences, New York Univ., N.Y.,
(Apr 1970).
# languages, compilers #
"This lengthy work discusses many of the data structures commonly found in the implementation of systems programs, including compilers and interpreters,"
1. 23 Hays, D. G.  
Introduction to computational linguistics.  
# parsing, storage allocation, automatic translation #  
"This volume is intended as an introduction to the field of  
computational linguistics. It contains good coverage on  
such topics as algorithms, storage structures,  
representation of data in storage, look-up techniques,  
parsing strategies, and formal grammar theory."

1. 24 Hext, J. B.  
Programming languages and compiling techniques.  
# compiling, language #

1. 25 Higman, B.  
A comparative study of programming languages.  
# syntax, semantics, formal languages, compiler #  
"This book covers a wide variety of topics including formal,  
languages, macrogenerators, different programming languages,  
list processing, etc."**  
CR 14510.

1. 26 Hopgood, P. R. A.  
Compiling techniques.  
pp.  
# compilers #  
"This book deals with modern techniques used in the design  
and implementation of compilers. It covers data structures,  
trees, graphs, arrays, tables, the description of languages,  
lexical and syntactic analysis, code generation, storage  
allocation and compiler-compilers. It is an excellent  
introduction to the field."

1. 27 International Computation Centre, (Eds).  
Symbolic language in data processing, proceedings of the  
# compiling #

1. 28 Irons, E. T.  
Towards more versatile mechanical translators.  
# translation #  
CR 5678.
1. 29 Iverson, K. E.  
A programming language.  
# language #  
"The author presents a programming language in detail and then applies the language to such topics as sorting and logical calculus. The book is in textbook format, with exercises at the end of each chapter."

1. 30 Katzan, H., Jr.  
Batch, conversational, and incremental compilers.  
# compilers #

1. 31 Klerer, M., and Reinfelds, J.  
Interactive systems for experimental applied mathematics.  
# compiling, processors #  
"This volume presents a series of papers on interactive on-line systems. It presents the users' point of view, components of interactive systems, automation of applied mathematics, and information on the implementation of interactive systems. It includes some information on the writing of interpreters."

1. 32 Knowlton, K. C.  
A programmer's description of L SIX.  
Comm ACM 9, 8 (Aug 1966), 616-625.  
# language #

1. 33 Knuth, D. E.  
The art of computer programming, Vol 1, Vol 2.  
# compilers #  
"An excellent work discussing many of the techniques used in the implementation of compilers."

1. 34 Knuth, D. E.  
History of writing compilers.  
# compilers #

1. 35 Knuth, D. E.  
A history of writing compilers.  
# compilers #  
"This paper describes the various components of compilers and how different compilers have handled formula breakdown and object code generation."  
CR 3133.
1. 36 Lampson, R. W.
Interactive machine programming.
# macros #

1. 37 Landen, W. H., and Battenburg, W. H.
On the efficient construction of automatic programming systems,
# compiling #

1. 38 Landin, P. J.
The next 700 programming languages.
Comm ACM 9, 3 (Mar 1966), 157-166.
# language #
"A family of unimplemented computer languages is described that is intended to span differences of application area by a unified framework. The design of a language is characterized by its physical representation and the choice of abstract entities, data types, lists, etc."

1. 39 Ledgard, H. F.
Ten mini-languages in need of formal definitions.
SIGPLAN 5, 4 & 5 (Apr 1970), 14-37.
# language, compilers #

1. 40 Lee, J. A. N.
The anatomy of a compiler.
# compiler, language, syntax #
"This book discusses formal definition of syntax, syntactic analysis, various compiler generators and similar subject areas."
CR 14728.

1. 41 Lomet, D. B.
The construction of efficient deterministic language processors.
# translators #
CR 19078.

1. 42 Luehbert, W. P., and Collom, P. W.
Signal Corps research and development on automatic programming of digital computers.
Comm ACM 2, 2 (Feb 1959), 22-27.
# translation, compiler, compiler-compiler #
"The authors trace the process of translation of a problem oriented language into a machine language. They then propose the creation of a universal language and of special-purpose compilers."
1. 43  Madnick, S. E.
String processing techniques.
Comm ACM 10, 7 (July 1967), 420-424.
  # storage allocation #

1. 44  Maurer, W. D.
Programming.
  # programming #

1. 45  McKeeman, W. M.
An approach to computer language design.
  # compiler, language #
CR 13436,

1. 46  McKeeman, W. M., Horning, J. J., and Wortman, D. B.
A compiler generator.
  # compiler, compiler-compiler #
"This book presents both an overview of the syntax-directed
precedence language approach to compiler writing and the
specific example of the XPL compiler which was developed at
Stanford University using this method."

1. 47  McKeeman, W. M., Nelson, E. C., and Wortman, D. B.
The XPL compiler generator system.
Proc AFIPS 1968 FJCC, Vol 33, 617-635.
  # compiler-compiler, compiler generator #
"This paper describes the XPL language and a set of programs
which constitute a translator writing system. XPL is
described by comparison with PL/1. The XPL language is
deliberately restricted to simple features which are useful
in writing translators."

1. 48  Napper, R. B. E.
The third-order compiler: a context for free man-machine
communication.
  # compiler-compiler #
"The author introduces the concept of third-order compilers
which would provide to the compiler-writer facilities
similar to those provided by the second-order compiler to
the ordinary programmer."
CR 12.360.
1. 49  Narasimhan, R.
Programming languages and computers: a unified meta-theory.
Academic Press, N. Y.
# language, theory #

1. 50  Naur, P.
Program translation viewed as a general data processing problem,
Comm ACM 9, 3 (Mar 1966), 176-179.
# translation#
"The paper attempts to obtain a broader viewpoint toward
compiler writing rather than considering it as a narrow
field of computer science. The author deals with structure,
reliability and techniques."

1. 51  Opler, A.
Requirements for real-time languages.
Comm ACM 9, 3 (Mar 1966), 196-199.
# languages, compiling#
"The unique requirements of real-time programming are
discussed with some attention being paid to special
compilation and execution peculiarities."

1. 52  Opler, A., Caracciolo, A., and Gorn, S.
Symposium on languages for processor construction,
# processor#
CR 7257.

1. 53  Orchard, and Hays, W.
The general problem of computing languages.
# languages#

1. 54  Paul, M.
Kolloquium fur sprachen und algorithmen.
Zeit. Math. Logik 8 (1962), 299-308. (German),
# language#

1. 55  Perlis, A. J.
The synthesis of algorithmic systems.
# compiling#
1. **Pollack, B.W.**  
The control program and associated subroutines.  
Stanford University, Paper AP-28, (June 1968).  
"This paper describes the detailed workings of a compiler/interpreter for a control program within a transformational grammar testing system."

1. **Pollack, B.W.**  
Compiler techniques.  
"This book presents a summary of the basic techniques necessary for the implementation of compilers. A wide variety of subjects is covered including syntax, parsing, resource allocation, detection and correction of errors, and details of compiler construction."

1. **Randell, B., and Russel, L. J.**  
ALGOL 60 implementation.  
"compiler"

1. **Presser, L.**  
The structure, specification, and evaluation of translators and translator writing systems.  
Rept. 68-51, Univ. of Calif., Los Angeles, Calif. (Oct 1968).  
"translators"

1. **Raphael, B.**  
The structure of programming languages.  
Coma ACM 9, 2 (Feb 1966), 67-71.  
"languages"

1. **Rosen, S.**  
Programming systems and languages.  
"languages"

1. **Rosen, S., (Ed).**  
Programming systems and languages.  
"languages and systems"
1. 63 Rosin, R. P.
Translation of artificial languages by compiler programs, research report and design for future languages.
 Proc ACM 14th Nat'l Conf. (1959), 75.
 # compiler, translation #

1. 64 Samelson, K.
Programming languages and their processing.
 Proc IFIP Congress, Munich, (1962), 487-492,
 # syntax, translator, generator #
"Samelson's article gives an introduction to language structure, pushdown stacks and different forms of processors."
 CR 3262,

1. 65 Samelson, K., and Rauer, P. L.
Sequential formula translation.
 Comm ACM 3, 2 (Feb 1960), 76-83.
 # translator #
"A brief history of sequential formula translation is given and the specific elements of translation, including the evaluation of arithmetic expressions, are discussed. The last-in-first-out principle is presented."
 CR 0219,

1. 66 Sammet, J. E.
Programming languages: history and fundamentals.
 # language #

1. 67 Scazighino, R. L.
Computer evolution to aid compilers.
 # compilers #.
 CR 4545.

1. 68 Schwartz, J. T., and Cocke, J.
Programming languages and their compilers, preliminary notes.
 # languages, compilers #
"A lengthy, extremely good summary of the work done in the field."
1. 69  
Steel, T. B., Jr., (Ed).  
Formal language description languages for computer programming.  
     # meta-languages, formal languages #

1. 70  
Wegner, P.  
An introduction to symbolic programming.  
     # languages #
     "This book is an introductory text covering the following topics: 1) elementary machine language, 2) programming in symbolic machine language, 3) extended assembly language, 4) FORTRAN, 5) the FORTRAN Monitor System."
     CR 4532.

1. 71  
Wegner, P.  
Programming languages, information structures and machine organization.  
     # languages, compilers #
     "This book discusses machine language, machine organization, assembly techniques, macro systems, lambda calculus, the structure of procedure-oriented languages and the run-time representation of dynamic systems."

1. 72  
Wegner, P., (Ed),  
Introduction to system programming.  
     # compilers #
     "This collection of articles includes two discussions of FORTRAN compilers, four of ALGOL compilers, and three of various commercial compilers. The topics of these articles include translation, optimization and stack techniques."
     CR 0640.

1. 73  
Yngve, V. H.  
Toward better programming languages.  
     # language #

1. 74  
Zemanek, H.  
Semiotics and programming languages.  
Comm ACM 9, 3 (Mar 1966), 139-143.  
     # languages #
     "This article concerns the application of 'semiotics' to programming languages. *Semiotics* consists of three branches: syntactics, semantics and pragmatics."
2. 0  SYNTAX- AND TABLE-DIRECTED PARSING

2. 1  Abramson, H. D.
The applicability matrix of a syntax directed parsing procedure.
BIT  8, 4 (1968), 253-261.
# syntax-directed, parsing #

2. 2  Abramson, H. D.
A note on left-recursive rules and the partitioning of a recognition matrix for syntax-directed translation.
BIT 10, 1 (1970), 1-5.
# parsing, formal grammar, syntax #

2. 3  Ackerman, A. F.
Generating PL/I phrase-structure productions at compile-time.
# compiling, phrase-structure #

2. 4  Aho, A. V., Hopcroft, J. E., and Ullman, J. D.
A general theory of translation.
# translation, compiling #
"The authors describe general translation theory which is fundamental to the theory of compiling. Translation is defined in terms of transducers and recognizers."
CR 7943.

3. 5  Aho, A. V., and Ullman, J. D.
# syntax-directed translation #

2. 6  Aho, A. V., and Ullman, J. D.
# formal theory of translation #
CR 18721,

2. 7  Anderson, R. H.
# syntax #

2. 8  Arden, B. W.
# compiler #
2. 9  Backus, J. W.
The syntax and semantics of the proposed international
algebraic language of the Zurich ACM-GAMM conference.
(1960).
# syntax, semantics, language #
"The syntax and semantics of ALGOL as it stood at that point
in its construction are given. Some elements included in
this paper were dropped before the 1960 report was issued."
CR 3158.

2. 10  Bandat, R. S., and Wilkins, R. L.
An experimental general purpose compiler.
# compiler, language, processor #
"The authors describe an approach to provide language
processors for the development of new programming languages
with a minimum investment in programmer time and effort.
The aim is to facilitate defining the syntax of new
programming languages to parse them so that there need be
only one output routine for each operator in the new
programming language. First a parsing program is
implemented and then a generic method for determining
hierarchy and syntactic legality of input characters is
designed."
CR 0017,

2. 11  Eanerji, R.
Some studies in syntax-directed parsing.
In Computation in Linguistics, P. Garvin, (Ed.),
Indiana Univ. Press, Indiana, (1966), 76.
# syntax-directed parsing #

2. 12  Barnett, M. P., and Futrelle, R. P.
Syntactic analysis by digital computer.
Coma ACM 5, 10 (Oct 1962), 515-526.
# syntactic analysis #
"A language (Shadow) is used to describe syntax; a Shadow
subroutine given a string and a syntax description, produces
the syntactic analysis as a table. The Shadow language is
discussed and some examples are given?

2. 13  Bastian, A. L.
A phrase-structure language translator.
(1962).
# phrase-structure languages, translator #
2. 14 Bell, J. R.
A new method for determining linear precedence functions for
precedence grammars.
Comm ACM 12, 10 (Oct 1969), 567-569.
# precedence, grammar #

2. 15 Berman, R., Sharp, J., and Stusges, L.
Syntactical charts of COBOL 61.
# syntax #
The authors constructed a syntax chart for COBOL 61. The
article itself gives a very brief description of the charts
which have been used in the design of the Burroughs B5000
COBOL-61 compiles."

2. 16 Blum, E. K.
Towards a theory of semantics and compilers for programming
languages.
Journal of Computer and System Sciences, 3, 3 (Aug 1969),
248-275.
# semantics, language, compilers #

2. 17 Boyle, J. M., and Grau, A. A.
An algorithmic semantics for ALGOL 60 identifier denotation.
# language, semantics #

2. 18 Erooker, R. A., and Morris, D.
A general translation program for phrase-structure languages.
Comm ACM 9, 1 (Jan 1962), 1-10.
# translation, phrase-structure, extendible #
"A compiler is described which works in two steps: the
syntax definition of a language is input, and then a source
program in that language is translated. Most of the
discussion is of phrase-structure and the translation
process. The authors build up the definitions and language
used in their paper 'Trees and Routines' which is published
in Computer Journal. The program, 1) in the primary phase,
accepts the definition of a phrase-structure language and 2)
in the secondary phases, translates a source program written
in that language. This program is extendable, with
allowances for new formats either in terms of the old format
or in terms of the basic assembly instructions,"

2. 19 Erooker, R. A., and Morris, D.
An assembly program for a phrase-structure language.
# phrase-structure language #
2. 20
Brooker, R. A., and Morris, D.
A description of Mercury-Autocode in terms of a phrase-structure language.

This article defines Mercury autocode in terms of a phrase-structure language. To facilitate complete understanding, the authors have included other information about Mercury autocode: source language, target language, metasyntactical language of the assembly program.

2. 21
Brooker, R. A., et. al.
Trees and routines.
Comp J 5 (1962), 33-47.

The authors go within phrases for a deeper look at structure and describe portions of a compiler organized around their definition of phrases, formats and routines.

2. 22
Burstell, R. M.
Some aspects of CPL semantics, No. 3.

2. 23
Caracciola Di Porino, A,
Some remarks on the syntax of symbolic programming languages.
Comm ACM 6, 8 (Aug 1963), 456-460.

This is an in-depth discussion of the syntax of formal languages, with illustrations drawn from the BNP of ALGOL. The basic point made is that symbolic programing languages are characterized by the fact that they are formal languages over two types of symbols: specific symbols and general symbols. The author suggests the formation of a new class of formal languages for defining formal text.

2. 24
Carr, J. W. III, and Weiland, J.
A non-recursive method of syntax specification.

The paper describes a non-recursive method for syntax specification. A non-recursive definition of ALGOL is given. The paper suggests that this is a more easily understood definition,
2. 25  Chapin, N.
Parsing of decision tables.
$parsing$
"The author describes techniques based on parsing of
decision tables which regard to horizontal and vertical data
structures, context-relation, etc. to reduce the size of
decision tables."
CR 13316.

2. 26  Charters, B. A., and Florentin, J. J.
A universal syntax-directed top-down analyzer,
J ACM 15, 3 (July 1968), 447-464.
$syntax-directed, compiler, formal$
"The authors give an algorithm that will analyze strings of
unbounded length using the rewriting rules of any
context-free grammar."
CR 15766.

2. 27  Cheatham, T. E., and Sattley, K.
Syntax-directed compiling.
$syntax directed, compiler$
"This paper is a discussion of a top-down recognizer, for a
syntax-directed compiler. Extensive examples are given,"
CR 6304.

2. 28  Clapp, L.
A syntax directed approach to automated aids for symbolic
mathematics.
Summary in Comm ACM 9, 8 (Aug 1966), 549.
$syntax-directed$
"This paper seems to have little direct relation to
compilers except that it describes a new use of the
syntax-directed techniques."

2. 29  Clapp, L. C.
A syntax-directed approach to automated aids for symbolic
mathematics.
ACM Symposium on Symbolic and Algebraic Maniipulations,
$syntax-directed, processor, syntax$
"This paper discusses the use of syntactic analysis of
mathematical expressions as the framework of a system to aid
the scientist in performing symbolic operations on
mathematical expressions. The advantage of the system is
that the basic approach may be developed without many a
priori restrictions on the nature of the mathematical
etities to be processed. The user can modify or extend the
syntax definitions once the basic structure has been
developed."
2.30 Cocke, J., and Schwartz, J. T.
Programming languages and their compilers: preliminary notes.
2d rev. version.
New York, Courant Institute of Mathematical Sciences,
"This lengthy work describes in detail the workings of several compilers. It is one of the most comprehensive works of its type currently available. The work includes two comprehensive bibliographies as well."

2.31 Cohen, D. J., and Gotlieb, C. C.
A list structure form of grammars for syntactic analysis.
"This paper presents a practical approach to the ordering of grammar rules for maximum efficiency whereby reordering of rules is adjusted to optimize the analysis of input string samples."

2.32 Cohen, J., and Nguyen-Dinh, X.
Note on grammar rules in syntax analyzers.
"This paper presents a practical approach to the ordering of grammar rules for maximum efficiency whereby reordering of rules is adjusted to optimize the analysis of input string samples."

2.37 Coles, S.
Syntax-directed interpretation of natural language.
"This is one of the best overall summaries of the subject of language processors. It is lengthy, well-written and covers the topic both in depth and breadth."

2.35 Dean, A. L.
Some results in the area of syntax directed compilers.
"This paper presents a practical approach to the ordering of grammar rules for maximum efficiency whereby reordering of rules is adjusted to optimize the analysis of input string samples."

2.36 DeRemer, F. L.
Practical translators for LR(k) languages.
2. 37 Coaolki, R.
A universal compiler system based on production rules.
BIT 8, No. 4, (1968), 262-275.
# syntax-directed, compiler #
"The author discusses a compiler system using production rules for translation. Source language syntax is defined in terms of phrase-structure grammar."

2. 38 Donovan, J. J., and Ledgard, H. F.
A formal system for the specification of the syntax and translation of computer languages.
# syntax, translation, language #
CR 0049.

2. 39 Duncan, F. A.
Our ultimate meta-language.
# meta-language #

2. 40 Parley, J. C.
Generating a recognizer for a BNF grammar,
# recognizer, generator #

2. 41 Parley, J. C., and Sturgis, H.
A formalism for translator interactions.
Comm ACM 13, 10 (Oct 1970), 607-617.
# translators #

2. 42 Eickel, J., Paul, M., Rauer, F. L., and Samelson, K.
A syntax controlled generator of formal language processors.
Comm ACM 6, 8 (Aug 1963), 451-406.
# syntax-directed, formal languages, processors #
"This paper describes the execution of an algorithm, the input for which is a language in Backus Normal Form and the output of which is a set of transition rules for a processor. This processor is then able to translate the original language into a sequential language of macro instructions2
CR 5998.

2. 43 Evans, A.
Syntax analysis by a production language,
# syntax analysis #
CR 13510.
2.44 Feldman, J. A.  
A formal semantics for computer languages and its application in a compiler-compiler.  
"A meta-language for specifying syntax and semantics is described. The meta-language is used as the basis for an efficient, functioning compiler-compiler."  
CR 10080.

2.45 Feldman, J. A.  
A formal semantics for computer oriented languages.  
CR 13841.

2.46 Feldman, J., and Gries, D.  
Translator writing systems.  
Coma ACM 11, 2 (Feb 1968), 77-113.  
"This paper surveys critically the research efforts put into automating compiler writing. The paper includes the formal study of syntax and its application to translator writing, various approaches to automating semantic aspects of translator writing and other related topics such as the formal study of semantics, etc."  
CR 14729.

2.47 Ferentzy, E. N., and Gabura, J. R.  
A syntax directed processor writing system.  
"The authors describe a processor writing system--MPL/I. The processor produced by MPL/I is a PL/I program plus syntax tables. The translator includes a driving mechanism making use of a parsing method developed by B. Domolki."

2.48 Floyd, R. W.  
A descriptive language for symbol manipulation.  
3 ACM 8, 4 (Oct 1961), 579-584.  
"The author presents notation to be used in the description of compilers and other complicated symbol manipulation algorithms. He is actually using his notation in the programming of an ALGOL translator for the UNIVAC 1105."

CR 2140.
2. 49  Floyd, R. W.
Syntactic analysis and operator precedence.
   JACM 10, 3 (July 1963), 316-333.
   # syntactic analysis #
   "The author defines the precedence grammars and languages,
   and describes an analyzer which can be designed from 'a
   matrix representation of a precedence relation between
   character pairs.' An appendix gives a summary of the theory
   of phrase-structure, operator, and precedence grammars,"

2. 50  Floyd, R. W.
The syntax of programming languages--a survey.
   # syntax #
   "This article is a survey of the use of syntax in
   programming languages, 'She paper discusses major problems
   in finding efficient analyzers and fully satisfactory formal
   grammars for programming languages.'"

2. 51  Floyd, R. W.
Rounded context syntactic analysis.
   # syntactic analysis #
   "The theory of bounded context grammar is presented and
   techniques for parsing phrases of such a grammar are given." CR 6074.

2. 52  Foster, J. M.
A syntax improving program.
   Comp J 11, 1 (1968), 31-34.
   # compiler, syntax, parsing #
   "The author describes a program which accepts a grammatic
   definition of a language as data and transforms it into an
   equivalent grammar that can be parsed by a simple parsing
   algorithm,"

2. 53  Foster, J. M.
Automatic syntactic analysis,
   # compiling, syntactic analysis, parsing #
   "This short monograph presents an excellent overview of the
   subjects of grammars, parsing, and syntactic analysis. The
   author covers top-down and bottom-up parsing, universal
   parsing methods, transition matrices, precedence grammars as
   well as several other important topics."
2.54 Fox, A. J., and Edwards, P. W.
Implementation of a syntax-driven interpreter for data retrieval.
"This paper describes the CLIC language and features @lambda-interpretation@." 

2.55 Foxley, E., and King, P.
The implementation of syntax analysis using ALGOL, and some mathematical implications.
Comp J 10 (Feb 1968), 325-335.
"syntactic analysis" 

2.56 Foxley, E., and King, P.
A meta-semantic language for use with a top-down syntax analyzer.
Proc IFIP (1968), Booklet B, 11-17.
"language, syntax analyzer" 

2.57 Gallie, T. El., Jr.
The Duke ALGOL compiler and syntactic routine method for syntax recognition,
"compiler, syntax, parsing" 

2.58 Garwick, J. V.
The definition of programming languages by their compilers.
In Formal Language Description Languages for Computer Programming, T. B. Steel, Jr., (Ed.), North Holland Publishing Co., Amsterdam, (1966), 139-147.
"language, compiler" 

2.59 Garwick, J. V.
The definition of programming languages by the compiler,
"languages, compilers"
2. 60 Gilbert, P.
On the syntax of algorithmic languages.
"The author presents a formal grammar that is
analysis-oriented. The model is called 'Analytical
grammar', and languages defined by its use are called
'analytic languages'. Any analytic grammar incorporates a
set of syntactic productions and a 'scan' which choses
productions for application to a string. Two primary
interests of the paper are in the subclasses of analytical
grammars that use simpler and more natural scans. Various
sub-models are discussed and equivalences are noted."
CR 9801.

2. 61 Gilbert, P., and McLe llan, W. A.
Compiler generation using formal specification of
procedure-oriented machine languages.
"The authors describe a compiler generation system which is
rigorously based and which allows formal specification of
both source language and machine language."
CR 0016.

2. 62 Glennie, A. E.
On the syntax machine and the construction of a universal
compiler.
Tech. Rept. No. 2, Computation Center,
CR 0016.

2. 63 Gorn, S.
mechanical pragmatics: a time-motion study of a miniature
mechanical linguistic system.
"This article goes with the author's earlier '... a Baker%'
Dozen.' paper. A miniature object language and its syntax
are created and then modified to demonstrate their
relationship,"

2. 64 Graham, R. M.
Bounded context translation.
"This paper presents a discussion of the use of bounded
context grammars in compiling. The approach of operator
precedence is used. Some attention is given to efficiency
and to algorithms used in syntax-directed compilers,"
2. 65  Grau, A. A.
A translator-oriented symbolic *programming* language.
_JACM_ 9, 4 (Oct 1962), 480-487.
# translation #
"The author presents a target language which may be used as an intermediate language in translation. **Features** of the language include a small number of instruction types and minimum parenthesis structure. The author discusses the operations and he ends with an application of this language to the translation of ALGOL."
CR 3868.

2. 66  Hamilton, J. A.
Investigation of a table-driven compiler system.
# table-driven compiler #

2. 67  Haynes, R. R., and Schutte, L. J.
Compilation of optimized syntactic *recognizers* from Floyd-Evans productions.
_SIGPLAN_ 5, 7 (July 1970), 38-51.
# syntax analysis, optimization, compiler #

2. 68  Hays, D. G.
Introduction to computational linguistics.
# parsing, storage allocation, automatic translation #
"This volume is intended as an introduction to the field of computational linguistics. **It** contains good coverage on such topics as algorithms, storage structures, representation of data in storage, look-up techniques, parsing strategies, and formal grammar theory."

2. 69  Hext, J. B.
Programming languages and *compiling* techniques.
# compiling, language #

2. 70  Hext, J. B., and Roberts, P. S.
Syntax analysis by Domolki's algorithm.
# syntax analysis #

2. 71  Higman, B.
A comparative study of programming languages.
# syntax, semantics, formal languages, compiler #
"This book covers a wide variety of topics including formal languages, macrogenerators, different programming languages, list processing, etc."
CR 14510.
2. 72 Halt., A. W.
A mathematical and applied investigation of free structures for computer syntactic analysis.
# syntactical analysis #

2. 73 Holt, A. W.
Automatic code translation system.
Final Report, Doc No. DA 36-039-sc-75047.
# translation #

2. 74 Hopgood, P. R. A.
Compiling techniques.
# compilers #
"This book deals with modern techniques used in the design and implementation of compilers. It covers data structures, trees, graphs, arrays, tables, the description of languages, lexical and syntactic analysis, code generation, storage allocation and compiler-compilers. It is an excellent introduction to the field."

2. 75 Huskey, H. D., Love, R., and Wirth, N.
A syntactic description of BC MELIAC.
Comm ACW 6, 7 (July 1963), 367-375.
# syntax, semantics #
"MELIAC compilers are one-pass and written in NBLIAC. The language's syntax (in ALGOL meta-language) and semantics are given, along with a syntactical flowchart."
CR 5041.

2. 76 Ingerman, P. Z.
A syntax oriented translator.
# syntax, translation #
"This short monograph describes a single syntax-directed translator. It covers its definition, syntax, parsing and extensions and relationships to other translators."
CR 11509.

2. 77 Ingerman, P. Z., Cotton, R. M., and Freedman, H. A.
A translation technique for languages whose syntax is expressible in extended Backus Normal Form.
# languages, translation #
2. 78 Irons, E. T.
A syntax directed compiler for ALGOL 60.
Comm ACM 4, 1 (Jan 1961), 51-06.

"Compilers not only translate one language into another but define the source language in terms of a second one, making it difficult to modify a compiler to reflect a language change. Irons has developed a compiler which keeps the two functions distinct, making modification simpler. The paper describes a compiling system consisting of a meta-language and a translator. Because of the separation of the two, extensions and modifications of the object language can be made more easily."

2. 79 Irons, E. T.
The structure and use of the syntax-directed compilers,

"This paper describes the structure and use of a compiling system in which the translator is independent of the translation rules and hence is independent of both the object and source language. The author first presents the meta-language, then examples of translation performed by the meta-language, and ends with a description of the recognition procedure."

2. 80 Tvesver, K. E.
A method of syntax specification.
Comm ACM 7, 10 (Oct 1964), 588-589.

"An addition of four simple conventions to BNF is described which simply make the notation more compact. The syntax of ALGOL 60, Revised is given as an example."

2. 81 Kanayama, Y.
A basic theory of syntax analysis in context-free phrase-structure languages.

"The author describes a computer program for syntax analysis in a context-free language. The method adopted is based on division of phrases into sub-phrases. This syntax analysis method can be applied to any grammar."

2. 82 Rasami, T., and Torii, K.
A syntax-analysis procedure for unambiguous context-free grammars.
3 ACM 16, 3 (July 1969), 423-431.
2. 83  Kirkley, C., and Rulifson, J.
LOTs: a syntax-directed compiler.
Internal Rept., Stanford Research Inst., Menlo Park, Calif.,
(Kay 1966).
# syntax-directed, compiler #

2. 84  Klerer, M., and Reinfelds, J.
Interactive systems for experimental applied mathematics,
# compiling, processors #
"This volume presents a series of papers on interactive
on-line systems, It presents the users' point of view,
components of interactive systems, automation of applied
mathematics, and information on the implementation of
interactive systems. It includes some information on the
writing of interpreters."

2. 85  Knuth, D. E.
On the translation of languages from left to right.
Info and Control 8 (Oct 1965), 607-639.
# translation #
"This paper describes a type of grammar which can be simply
translated from left to right with the proper algorithm.
Methods for generating recognizers for these grammars are
given."

2. 86  Knuth, D. E.
Backus normal form vs. Backus Naur form.
Comm ACM 7, 12 (Dec 1964), 735-736.
# syntax #

2. 87  Korenjak, A. J.
A practical method for constructing LR(k) grammars.
# context-free grammars #
-CR 18722,

2. 88  Kratky, G., and Kopetz, H.
The semantics of a mathematically oriented computer
language.
# semantics #
2. 89
Knino, S., and Oettinger, A. G.
Multiple-path syntactic analyzer.
Information Processing 62 (IFIP Congress),
Popplevell, (Ed.),

"A practical form of multiple-path analysis has been
discovered by the authors. The implementation and examples
are from the English language, but the techniques can be
applied to programming languages."
CR 3505.

2. 90
LaFrance, J. A.
Optimization of error-recovery in syntax-directed parsing
algorithms.
SIGPLAN 5, 7 (July 1970), 128.
(Abstract).
# optimization, parsing #

2. 91
LaFrance, J. A.
Optimization of error recovery in syntax-directed parsing
algorithms.
SIGPLAN 5, 12 (Dec 1970), 2-17.
# optimization, parsing, syntax-directed translation #

2. 92
Langmaack, H., and Eichel, J.
Prazisierung der begriffe phrasenstructur und structurelle
mehrdeutigkeit. In Chomsky-sprechen.
Rept. No. 6414, Rechencentrum der Technisch. Hochschule,
Munich, (1964).
# phrase-structure #
CR 7267.

2. 93
Lauer, P.
Formal definition of ALGOL 60.
Tech. Rept. No. TR 25.088, IBM Labs,, Vienna, Austria (Dec
1968).
# syntax, semantics #

2. 94
Learner, A., and Lin, A.L.
A note on transforming context-free grammars to Wirth-Weber
precedence form.
Comp J 13, 2 (May 1970), 142-144.
# context-free grammar #
"A technique is presented which will convert every CP
grammar into an equivalent Wirth-Weber simple precedence
grammar."
2. 95 Leavenworth, B. M.
Syntax macros and extended translation,
Comm ACM 9, 11 (Nov 1966), 790-793.
# syntax, translation #
"A translation approach is described which allows one to
extend the syntax and semantics of a given high-level base
language through the use of a new formalism called a
'syntax-macro'. Two types are discussed and examples are
given."

2. 96 Ledgard, H. P.
Production system: a formalism for specifying the syntax
and translation of computer languages.
Oxford Univ. Computing Lab., Programing Research Group,
(45 Banbury Road, Oxford, England), Rept. No. PRG-1 (Mar
# syntax-directed translation #

2. 97 Ledley, R. S., and Wilson, J. B.
Automatic-programming-language translation through
syntactical analysis.
# translation, syntactical analysis #
"This article presents methods and techniques of
syntax-directed automatic programing language translation
with examples taken from ALGOL. A single subroutine is
designed to translate any such syntactical and semantic
description into the machine language instructions. The
authors include several detailed figures to aid them in this
presentation."
CR 2603.

2. 98 Lee, J. A. N.
The anatomy of a compiler.
# compiler, language, syntax #
"This book discusses formal definition of syntax, syntactic
analysis, various compiler generators and similar subject
areas."
CR 14728.

2. 99 Lewis, P. M., II, and Stearns, R. E.
Syntax-directed transduction,
Comm ACM 15, 3 (July 1968), 465-488.
# compilers, syntax-directed, translation #
"The authors investigate some special conditions under which
syntax-directed translation can be performed on
deterministic pushdown machines."
2.100 \textbf{Lietzke, M. P.}\n\textit{A method of syntax checking ALGOL 60.}
\textit{Coma ACM 7, 8 (Aug 1964), 475-478.}
\# syntax \#
\textit{A syntax checker designed around ALGOL 60 is discussed. The checker is a set of mutually recursive processors tied together by bookkeeping subroutines. A method for error recovery is described.}
CR 6662.

2.101 \textbf{Liu, C. D., Chang, G. D., and Marks, R. E.}\n\textit{The design and implementation of a table driven compiler system.}
\# compiler \#
\textit{The authors present a generalized table driven compiler system which allow users to define their own special language. Table driven compiling is presented as an extension of syntax directed compiling.}

2.102 \textbf{Lomet, D. B.}\n\textit{The construction of efficient deterministic language processors.}
\# translators \#
CR 10078.

2.103 \textbf{Lucas, P.}\n\textit{Die strukturanalyse van formelubersetzern.}
\textit{Mailuefterl, Wien, (1961).} (German).
\# structural analysis, formal translation \#
CR 2136.

2.104 \textbf{Marimont, R. R.}\n\textit{Checking the consistency of precedence matrices.}
\textit{JACM 6, 2 (Apr 1959).}
\# precedence \#

2.105 \textbf{Martin, D. P.}\n\textit{Boolean matrix methods for the detection of simple precedence grammars.}
\textit{Comm ACM 11, 10 (Oct 1968), 685-687.}
\# grammars \#
\textit{The author describes a technique for computing the precedence relations of a context-free language using Boolean matrices. It translates the definitions of precedence into the representation of relations by Boolean matrices.}
CR 0159.
2.106 Rattison, R. L., and Mitchell, R. T.
A table driven compiler for use with automatic test equipment.
compiler
"When generating compilers for use with automatic test equipment, flexibility is needed in both source and object languages. The authors describe UTEC, a table driven system developed to facilitate compiler implementation and growth."

2.107 Mayoh, B. H.
Letter to the editor correcting E. T. Irons' A syntax-directed compiler for ALGOL 60., Coma ACM 4, 1 (Jan 1961), 51-06.
compiler
"Mahoh writes the editor of some possible corrections that can be made to Irons' article in a previous issue."

2.108 McClure, R. M.
TMG--a syntax directed compiler.
Proc ACM 20th Nat'l Conf. (1965), 262-274.
compiler
"This paper describes TMG, a syntax-directed compiler writing system. The system is directed towards straightforward and efficient translation of the input, thus there are virtually no facilities for optimization."

2.109 Nagao, M.
Syntactic analysis of a phrase-structure language.
syntactic analysis, phrase-structure language

2.110 Oettinges, A. G.
Automatic syntactic analysis and the pushdown store.
syntactic analysis

2.111 Parikh, R. J.
Language generating devices.
Quarterly Progress Rept. No. 60,
generator

2.112 Parikh, R. J.
On context-free languages.
context-free languages
2.113 Paul, M.
A general processor for certain formal languages. 

2.114 Paul, M.
ALGOL 60 processors and a processor generator. 

2.115 Paul, M. C.
A translation description system for computer languages. 

2.116 Pollack, B. W.
Compiler techniques. 

2.117 Pratt, T. W.
Syntax-directed translation for experimental programming languages. 

2.118 Pratt, T. W., and Lindsay, R. K.
A processor-building system for experimental programming language. 
2.119  Presser, L.
The structure, specification, and evaluation of translators and translator writing systems,
*Rept. 68-51, Univ. of Calif., Los Angeles, Calif.* (Oct 1968).
# translators #

2.120  Red'ko, V. N.
The syntactic analysis of Context-free languages,
In Cybernetics (May-June 1966).
Translation of *Kibernetika* 2, 3 (May-June 1966), 52–63.
(Russian).
# syntactic analysis, context-free, languages #
CR 0246.

2.121  Redziejovski, R. R.
On arithmetic expressions and trees.
*Comm ACM* 12, 2 (Feb 1969), 81–84.
# compiling #

2.122  Reeves, C. N.
Description of a syntax-directed translator.
*Comp J* 10 (1967), 244–206.
# syntax-directed, translator #
"The author describes an extension of ALGOL notation which permits the syntax and semantics of general languages to be specified compactly."
CR 15659.

2.123  Resnick, M., and Sable, J.
**INSCAN**: a syntax-directed language processor.
# syntax-directed, language, processor #
"INscan is a convenient tool for expressing the syntax of linear languages and for specifying the actions necessary to translate or otherwise process languages. It has been implemented at Auerbach. The INscan approach to language processor design separates the language scanning and translation function from the details of the post-translation processing and facilitates experimentation with the design of languages."
CR 15767.

2.124  Roberts, A. E.
The construction of recognisers,
# recognizers #
"This paper is a theoretical treatment of one method of constructing a recognizer from an arbitrary context-free grammar."
CR 11099.
2.125 Roberts, L. G.
A graphical service system with variable syntax.


2.126 Rochester, N.
A formalization of two-dimensional syntax description.

2.127 Sable, J. D.
Use of semantic structure in information systems.
Comm ACM 5, 1 (Jan 1962), 40-42.

2.128 Sherry, M.
Syntactic analysis in automatic translation.

2.129 Simpson, H. R.
A compact form of one-track syntax analyser,

2.130 Sklansky, J., Pinkelstein, R., and Russell, E. C.
A formalism for program translation.
J ACM 15, 2 (Apr 1968), 165-175.

2.131 Squire, J. S.
Translation algorithm for a multiple processor computer,
Cosm ACM 6, 7 (July 1963), 364.
2.132 Steel, T. B., Jr.
A formalization of semantics for programming language description.
# semantics, formal languages #

2.133 Strachey, C.
Towards a formal semantics.
# formal semantics #

2.134 Tarski, A.
Logic, Semantics, Metamathematics.
# semantics, meta-languages #

This is a collection of articles which are useful to the compiler writer if he is interested in the theory of semantics.

2.135 Unger, S. R.
On syntax directed translators.
# syntax-directed, translators #

2.136 Vandermey, J. E., Varney, R. C., and Patchen, R. E.
Symple—a general syntax-directed macro preprocessor.
# syntax-directed, macros, pre-processor #

2.137 van Wijngaarden, A.
Recursive definition of syntax and semantics.
# syntax, semantics #

2.138 van Wijngaarden, A., (Ed).
Draft report on ALGOL 68.
MR 93, fflatheatich Centrna, Amsterdam, Holland (1968).
# syntax, semantics #

This is the complete and formal description of the proposed ALGOL 68 language.
2.139 Warshall, S.
A syntax directed generator.
# syntax directed, generator #
"Warshall proposes a method of making code generation more
efficient by examining large pieces of the input before
generating code rather than coding every small piece of
input as soon as it is completely recognized. The generator
makes use of trees."
CR 2906,

2.140 Warshall, S., and Shapiro, R. M.
A general purpose table-driven compiler,
# compiler #
"A compiler is described which relies heavily on tables for
recognition, generation and code selection decisions. The
techniques used are discussed and examples are given."
CR 6664.

2.141 Whitney, G.
An extended BNF for specifying the syntax of declarations.
# syntax #

2.142 Wilkes, M. V.
The outer and inner syntaxes of a programming language.
Comp J 11, 4 (1968), 260-263.
# syntax, semantics #
"A discussion of syntax and semantics is presented?

2.143 Wilkes, M. V.
Constraint-type statements in programming languages.
Comm ACM 7, 10 (Oct 1964), 587.
# languages #
"A scheme of compilation is proposed which allows the left
part of an assignment statement to be an expression thus
implying relations among its variables, The system is
conceived as a way of making computers more readily
accessible to the general user."
CR 6939.

2.144 Wirth, N.
A basic course on compiler principles.
# syntax-directed, compiler #
"An introduction to phrase-structure languages is presented
as a basis for devising syntax-directed compilers. Both
theory and applications are presented."
2.145 Uirth, N. 
PI 360, a programming language for the 360 computers. 
ACM 15, 1 (Jan 1968), 37-74. 
"This article presents a syntax-directed meta-assembly language which is particularly suited to the IBM 360 computers."

2.146 Wirth, N. 
A programming language for the 360 computers. 
"A new ALGOL-like language is proposed which incorporates many improvements; a discussion and justification is presented.*"

2.147 Uirth, N. 
a generalization of ALGOL. 
ACM 6, 9 (Sept 1963), 547-064. 
"The proposed generalization can be summarized as the elimination of type declarations and the replacement of the procedure declaration by an assignment of a so-called quotation,' The language described features flexibility not present in ALGOL. It also eliminates the specification of array bounds, using dynamic storage instead."

2.148 Wirth, N., and Hoare, C. A. R. 
A contribution to the development of ALGOL. 
ACM 9, 6 (June 1966), 413-432. 
"A new ALGOL-like language is proposed which incorporates many improvements; a discussion and justification is presented.*"

2.149 Uirth, N., and Weber, H. 
EULER--a generalization of ALGOL and its formal definition: Part I, II. 
"A method for defining programming languages (simple precedence grammars) is developed which introduces a rigorous relationship between structure and meaning. A generalization of ALGOL is described in detail to show that block-structure, procedures, etc. can be adequately handled. Part II contains a formal description of the language EULER. An attempt is made to generalize ALGOL to create a simpler and more flexible language."
2.150 Wolman, B. L.
Operators for manipulating language structures.
ACM Symposium on Symbolic and Algebraic Manipulations,
"The algorithmic theory of languages provides a language
structure capable of representing the syntactic and semantic
structure of statements in algebraic, procedural or
graphical languages. Utilizing the semantic sequencing
information in the structure, operators defined for atomic
forms may be applied to arbitrarily complex structures to
provide a powerful manipulation capability. The author
describes a system constructed on these bases."

2.151 Zemanek, H.
Semiotics and programming languages.
Comm ACM 9, 3 (Mar 1966), 139-143.
"This article concerns the application of 'semiotics' to
programming languages. 'Semiotics' consists of three
branches: syntactics, semantics and pragmatics."
3. 0 PARSING IN GENERAL

3. 1 Abbot, R., and Kuno, S.
The predictive analyzer and context free grammars.
In Mathematical linguistics and automatic translation.
(1965).
# context-free, grammar #

3. 2 Abramson, H. D.
The applicability matrix of a syntax directed parsing procedure.
BIT 8, 4 (1968), 253-261.
# syntax-directed, parsing #

3. 3 Abramson, H. D.
A note on left-recursive rules and the partitioning of a recognition matrix for syntax-directed translation.
BIT 10, 1 (1970), 1-5.
# parsing, formal grammar, syntax #

3. 4 Aho, A. V., and Ullman, J. D.
Properties of syntax-directed translations.
Journal of Computer and System Sciences 3, 3 (Aug 1969), 319-334,
# formal theory of translation #

3. 5 Aho, A. V., and Ullman, J. D.
Translations on a context-free grammar,
# translation, context-free grammar #

3. 6 Panerji, R.
Some studies in syntax-directed parsing.
In Computation in Linguistics, P. Garvin, (Rd.),
Indiana Univ. Press, Indiana, (1966), 76.
# syntax-directed parsing #

3. 7 Ear-Hillel, Y., and Shamir, E.
Finite-state languages: formal representations and adequacy problems.
Bull. Res. Council of Israel 8F, 3 (Feb 60).
# formal languages #

3. 8 Ear-Hillel, Y., Perles, M., and Shamir, E.
On formal properties of simple phrase-structure grammars.
# formal language, phrase-structure grammar #
3. 9 Bar-Hillel, Y., Perles, M., and Shamir, E.
On formal properties of simple phrase-structure grammars.
Reprinted in Y. Bar-Hillel, (Ed.), Languages and Information, selected Essays.
formal grammar, phrase-structure

3. 10 Braffort, P., and Hirschberg, D., (Eds).
Computer programming and formal systems.
formal

3. 11 Brooker, R. A.
Top-to-bottom parsing rehabilitated.
parsing
"The author discusses the efficiency of the top-to-bottom parsing technique."
CR 15816.

3. 12 Brooker, R. A., and Morris, D.
A general translation program for phrase-structure languages.
Comm ACM 9, 1 (Jan 1962), 1-10.
translation, phrase-structure, extendible
"A compiler is described which works in two steps: the syntax definition of a language is input, and then a source program in that language is translated. Most of the discussion is of phrase-structure and the translation process. The authors build up the definitions and language used in their paper 'Trees and Routines' which is published in Computer Journal. The program, 1) in the primary phase, accepts the definition of a phrase-structure language and 2) in the secondary phases, translates a source program written in that language. This program is extendable, with allowances for new formats either in terms of the old format or in terms of the basic assembly instructions."

Trees and routines.
Comp 3 5 (1962), 33-47.
phrase-structure, translation, compilation
"The authors go within phrases for a deeper look at structure and describe Portions of a compiler organized around their definition of phrases, formats and routines."
3. 14 Brown, P. J.
Note on the proof of the non-existence of a phrase-structure grammar for ALGOL 60.
Comm ACM 6, 3 (Mar 1963), 105.
"Brown shows that some aspects of Floyd's non-existence proof are incomplete and not sufficiently generalized. This note extends the proof of the non-existence of a phrase-structure grammar to include the programs BEGIN; END and BEGIN END and just a dummy statement, all three of which are programs."

3. 15 Burks, A. W., and Wright, J.B.
Sequence generators, graphs and formal languages.
Info and Control 5, 3 (1962), 204-212.
"Burks and Wright show that some aspects of Floyd's non-existence proof are incomplete and not sufficiently generalized. This note extends the proof of the non-existence of a phrase-structure grammar to include the programs BEGIN; END and BEGIN END and just a dummy statement, all three of which are programs."

3. 16 Burstall, R. M.
Proving properties of programs by structural induction.
Corp J 12, 1 (Feb 1969), 41-48.
"Burstall demonstrates that ALGOL 60 is ambiguous and then discusses the question of whether an algorithm exists which can determine whether any given Backus system is ambiguous. He proves that there is no algorithm which can prove or disprove the ambiguity."

3. 17 Cantor, D. G.
On the ambiguity problem of Backus Systems.
"Cantor demonstrates that ALGOL 60 is ambiguous and then discusses the question of whether an algorithm exists which can determine whether any given Backus system is ambiguous. He proves that there is no algorithm which can prove or disprove the ambiguity."

3. 18 Chomsky, N.
A note on phrase-structure grammars.
Info and Control 2, 4 (1959), 393-395.
"Chomsky demonstrates that ALGOL 60 is ambiguous and then discusses the question of whether an algorithm exists which can determine whether any given Backus system is ambiguous. He proves that there is no algorithm which can prove or disprove the ambiguity."

3. 19 Chomsky, N.
Finite state languages.
Info and Control 1 (1958), 91-112.
"Chomsky demonstrates that ALGOL 60 is ambiguous and then discusses the question of whether an algorithm exists which can determine whether any given Backus system is ambiguous. He proves that there is no algorithm which can prove or disprove the ambiguity."

3. 20 Choaskey, N.
Formal properties of grammars.
Handbook of Mathematical Psychology, Pol 2.
Luce, Bush and Galanter, (Eds).
"Choaskey demonstrates that ALGOL 60 is ambiguous and then discusses the question of whether an algorithm exists which can determine whether any given Backus system is ambiguous. He proves that there is no algorithm which can prove or disprove the ambiguity."
3. 21 Chomsky, N.
Aspects of the theory of syntax.
# syntax, formal grammar
CR 10735.

3. 22 Chomsky, N.
Three models for the description of language.
TRR Trans Info. Theory, IT-2, 3 (Sept 1956), 113-124.
# language

3. 23 Chomsky, N.
On certain formal properties of grammars.
Info and Control 2 (June 1959), 137-167.
# formal grammars

3. 34 Chomsky, N.
A note on phrase-structure grammars.
Info and Control 2 (1959), 393-395.
# phrase-structure grammars

3. 25 Chomsky, N.
Syntactic structures.
# syntactic structures, syntax

3. 26 Chomsky, N., and Schutzenberger, M. P.
The algebraic theory of context-free languages.
In Computer Prog. and Formal Systems, Braffort & Hirschberg, (Eds.)
North Holland Puhltshing Co., Amsterdam, (1963), 118-161.
# language, context-free, formal

3. 27 Church, A.
The calculi of lambda-conversion.
# formal, language

3. 28 Cocke, J., and Schwartz, J. T.
Programming languages and their compilers: preliminary notes.
2d rev. version.
New York, Courant Institute of Mathematical Sciences,
# languages, compilers
"This lengthy work describes in detail the workings of several compilers. It is one of the most comprehensive works of its type currently available. The work includes two comprehensive bibliographies as well."
3. 39  Colmerauer, A.
Total precedence relations.
J ACM 17, 1 (Jan 1970), 14-30.
# precedence, grammar #

3. 30  Culik, K.
Well translatable grammars and ALGOL-like languages.
USAF Foreign Technology Div. (Wright-Patterson AFB, Ohio),
Rept. No. FTD-HT-23-613-38; CFSTI Rept. Wo. AD-683 105,
# translator #

3. 31  De Bakker, J.
Formal definition of algorithmic languages,
MR 74, Mathematisch Centrum, Amsterdam, Holland (May 1965).
Mathematisch Centrum Tract No. 16, Amsterdam, Holland
(1967).
# meta-language, formal #

3. 32  DeRemer, P. L.
Generating parsers for BNF grammars.
# parser #

3. 33  DeRemer, P. L.
Practical translators for LR(k) languages.
CFSTI, AD 699 501.
# translator #
# CR 7910. #

3. 34  Donovan, J. J., and Ledgard, H. F.
Canonic systems and their application to programming
languages.
# languages #

3. 35  Donovan, J. J., and Ledgard, H. F.
A formal system for the specification of the syntax and
translation of computer languages.
# syntax, translation, language #
CR 0049,

3. 36  Farley, J. C.
An efficient context-free parsing algorithm.
# parsing, context-free languages #
3. 37 Earley, J. C.
An efficient context-free parsing algorithm.
Carnegie-Mellon Univ., Dept. of Computer Science,
Pittsburgh, Pa., Air Force Office of Scientific Research
Rept. No. APOS-68-2185,
# parsing, context-free languages #

3. 38 Earley, J. C.
An LR(K) parsing algorithm.
# parsing #

3. 39 Earley, J. C.
Generating a recognizer for a BNF grammar.
Comp. Center Rept., Carnegie Inst. of Tech., Pittsburgh,
Pa., (1965).
# recognizer, generator #

3. 40 Earley, J. C., and Sturgis, H.
A formalism for translator interactions.
Comm ACM 13, 10 (Oct 1970), 607-617.
# translators #

3. 41 Eickel, J.
Generation of parsing algorithms for Chomsky 2-type
languages.
6401, Mathematisches Institut der Technisch, Hochschule,
Munich, (1964).
# generator, parser #

3. 42 Eickel, J., and Paul, M.
The parsing ambiguity problem in Chomsky-languages,
In Formal Language Description Languages for Computer
Programming, T. B. Steel, Jr., (Ed.), North Holland
# parsing, formal language #
CR 10946.

3. 43 Fischer, M. J.
Some properties of precedence languages.
In Proc ACM Symposium on Theory of Computing, (Bay 1969),
181-190.
# precedence #
3. 44  Floyd, R. W.
On the nonexistence of a phrase-structure grammar for ALGOL 60.
Comm ACM 5, 9 (Sept 1962), 483-484.

"The author, by means of examples, shows that all of the formation rules of ALGOL 60 cannot be stated as a phrase-structure language. The author suggests that other languages sharing with ALGOL 60 its requirement for declaration of variables, arrays, etc. could also not be represented by a phrase-structure grammar."

CR 3884.

3. 45  Floyd, R. W.
On ambiguity in phrase-structure languages.
Comm ACM 5, 10 (Oct 1962), 526, 534.

"The author asserts that unambiguous languages exist, but that there is no formal proof for that fact. With an example, he shows that there cannot be an algorithm sufficient to determine ambiguity or unambiguity in each case."

3. 46  Floyd, R. W.
Syntactic analysis and operator precedence.
3 ACM 10, 3 (July 1963), 316-333.

"The author defines the precedence grammars and languages, and describes an analyzer which can be designed from a matrix representation of a precedence relation between character pairs. An appendix gives a summary of the theory of phrase-structure, operator, and precedence grammars."

3. 47  Floyd, R. W.
Bounded context syntactic analysis.

"The theory of bounded context grammar is presented and techniques for parsing phrases of such a grammar are given."

CR 6074.

3. 48  Floyd, R. W.
A note on mathematical induction on phrase-structure grammars.

"The author first gives some basic definitions; then using these as a basis, he goes on to prove two theorems about phrase-structure grammars."

CR 2475.
3. 49 Floyd, R. W.
A machine-oriented recognition algorithm for context-free languages.
# context-free languages, parser #

3. 50 Foster, J. M.
Automatic syntactic analysis,
# compiling, syntactic analysis, parsing #
"This short monograph presents an excellent overview of the subjects of grammars, parsing, and syntactic analysis. The author covers top-down and bottom-up parsing, universal parsing methods, transition matrices, precedence grammars as well as several other important topics."

3. 51 Foxley, E., and King, P.
The implementation of syntax analysis using ALGOL, and some mathematical implications.
Comp 3 10 (Feb 1968), 325-335.
# syntactic analysis #

3. 52 Fujita, T.
A note on recursive languages,
# language, grammar #
"The author presents a general procedure for formalizing informal syntactical rules which are supplementary to structure grammars in describing a programming language."

3. 53 Gallie, T. M., Jr.
The Duke ALGOL compiler and syntactic routine method for syntax recognition.
# compiler, syntax, parsing #

3. 54 Gilbert, P.
On the syntax of algorithmic languages.
3 ACM 13, 1 (Jan 1966), 90-107.
# syntax, language #
"The author presents a formal grammar that is analysis-oriented. The model is called 'Analytical grammar', and languages defined by its use are called 'analytic languages'. Any analytic grammar incorporates a set of syntactic productions and a 'scan' which chooses productions for application to a string. Two primary interests of the paper are in the subclasses of analytic grammars that use simpler and more natural scans. Various sub-models are discussed and equivalences are noted." CR 9801.
3. 55  Ginsburg, S.
The mathematical theory of context-free languages.

3. 56  Ginsburg, S.
A survey of ALGOL-like description languages for context-free language theory.

3. 57  Ginsburg, S., and Greibach, S. A.
Deterministic context-free languages.

3. 58  Ginsburg, S., and Rice, H. G.
Two families of languages related to ALGOL.
J. ACM 9, 3 (July 1962), 350-370.

3. 59  Ginsburg, S., and Ullian, J.
Ambiguity in context-free languages.
J. ACM 13, 1 (Jan 1966), 62-89.

3. 60  Ginsburg, S., Greibach, S. A., and Harrison, M. A.
Stack automata and compiling.
J. ACM 14, 1 (Jan 1967), 172-201.
3. 61  Gorn, S.
Detection of generative ambiguities in context-free mechanical languages.


"This article presents a generalized prefix language and proceeds to construct a derivation generator, and a limited ambiguity detector. The author realizes that the ambiguity problem is unsolvable. Here he presents a processor capable of detecting generative ambiguities, a subset of the general problem. This processor: 1) recognizes generative admissability, 2) 'constructs the complete graph of the system, 3) the graph is broken open into an indefinite periodic tree,' and 4) this 'yields a four-tape generator of all derivations and words of the language.'"

CR 5106.

3. 62  Gorn, S.
The treatment of ambiguity and paradox in mechanical languages.

"This paper discusses mechanical languages, prefix extensions, syntactic and pragmatic ambiguities, and paradoxes. It is shown that there is a relationship between language extension and an increase in control ambiguity."

3. 63  Greibach, S. A.
A new normal form theorem for context-free phrase-structure grammars.

Comm ACM 12, 1 (Jan 1965), 42-52.

"A standard form is described for grammars where all productions are of the form \( Z \rightarrow a \ Y_1 \ldots \ Y_m \), where only \( a \) is a terminal symbol. This form is proved strongly equivalent to other forms. This form is particularly convenient for program translation."

CR 7830.

3. 64  Creibach, S. A.
Formal parsing systems.

Comm ACM 7, 8 (Aug 1964), 499-504.

CR 6878.

3. 65  Griffiths, T. V.
Top-down versus bottom-up analysis.

Proc IFIP (1968), Booklet B, 80-85.
3. 66  Griffiths, T. V., and Petrick, S. R.
On the relative efficiencies of context-free grammar recognizers.
Comm ACR 8, 5 (Ray 1965), 289-299.
"Various recognition procedures for CP grammars are
described and compared. The two major methods considered
are selective top-to-bottom and selective bottom-to-top."

3. 67  Gruska, J.
Unaabiguity and ambiguity of context-free grammars and
languages.
Proc IFIP Congress (1968), Mathematics, Booklet A, 135-139.

3. 68  Hays, D. G.
Introduction to computational linguistics.

3. 69  Holt, A. W.
A mathematical and applied investigation of free structures
for computer syntactic analysis.
PhD Dissertation, University of Pennsylvania, Philadelphia,

3. 70  Horning, J. J., and Lalonde, W. R.
Empirical comparison of LR(k) and precedence parsers.

3. 71  Ichbiah, J. D., and Morse, S. P.
A technique for generating almost optimal Floyd-Evans
productions for precedence grammars.
Irons, R. T.
An error-correcting parse algorithm.
Comm ACM 6, 11 (Nov 1963), 669-673.
"This article presents an algorithm which corrects syntax in a program. The program is parsed until an incorrect statement is found. The program then makes a tentative correction, and continues making tentative corrections until one is found that will parse consistently. This algorithm may have some importance in the future in the area of pattern recognition."
CR 5670.

Irons, R. T.
'Structural' connections in formal languages.
"Languages are discussed relative to their difficulty in parsing?"
CR 6212.

Irwin, L.
Implementing phrase-structure productions in PL/1.
"A simple technique is described for implementing productions of a context-free phrase-structure grammar in PL/1."

Iverson, K. E.
Formalism in programming languages.
Comm ACM 7, 2 (Feb 1964), 80-88.
"This paper describes a notation applicable to identities among statements within one language. It focuses on the practical implication of these identities in a compiler."

Kanayama, Y.
A basic theory of syntax analysis in context-free phrase-structure languages.
"The author describes a computer program for syntax analysis in a context-free language. The method adopted is based on division of phrases into sub-phrases. This syntax analysis method can be applied to any grammar."
3. 77 Kanayama, Y.
Analyzeability of sub-phrases and general theory of syntax analysis in context-free phrase-structure languages.

A sub-phrase unit is called analyzable if it is relevant to the syntax analysis. The author gives an algorithm to ascertain the analyzeability of a sub-phrase unit. Deletion of non-analyzable units makes the analysis efficient.

3. 78 Kasami, T., and Torii, K.
A syntax-analysis procedure for unambiguous context-free grammars.
*JACM 16, 3 (July 1969), 423-431.*

3. 79 Knuth, D. E.
On the translation of languages from left to right.
*Info and Control 8 (Oct 1965), 607-639.*

This paper describes a type of grammar which can be simply translated from left to right with the proper algorithm.
Methods for generating recognizers for these grammars are given.

3. 80 Knuth, D. E.
*Backus normal form vs. Backus Naur form,*
*Comm ACM 7, 12 (Dec 1964), 735-736.*

3. 81 Korenjak, A. J.
A practical method for constructing LR(k) grammars.
*Comm ACM 12, 11 (Nov 1969), 613-623.*

3. 82 Korenjak, A. J.
Deterministic language processing.
3. 03 Landin, P. J.
A correspondence between ALGOL 60 and Church's lambda-annotation.
Comm ACM 8, 2 & 3 (Feb, Mar 1965), 89-101, 158-167.
 Part I describes a variant of Church's lambda-notation which can be applied as a formal representation of the semantics of a language such as ALGOL 60. Part II is a description of a formal mapping of ALGOL 60 semantics onto the altered form of Church's lambda-notation set forth in Part I.

3. 04 Landweber, P. S.
Decision problems of phrase-Structure grammars.
 This paper discusses the various decision problems in the four basic Chomsky type languages. Decidability proofs are presented for several different cases.

3. 05 Learner, A., and Lim, A. L.
A note on transforming context-free grammars to Wirth-Weber precedence form.
Camp J 13, 2 (May 1970), 142-144.
 A technique is presented which will convert every CP grammar into an equivalent Wirth-Weber simple precedence grammar.

3. 06 Lee, J. A. N.
The anatomy of a compiler.
 This book discusses formal definition of syntax, syntactic analysis, various compiler generators and similar subject areas.
 CR 14728.

3. 07 Loeckx, J.
An algorithm for the construction of bounded-context parsers.

3. 88 London, R. L.
A computer program for discovering and proving recognition
rules for Backns Normal Form grammars.
"This paper describes a running computer program which
discovers and proves the validity of recognition rules for
simple BNF grammars. The program is meant to aid in the
construction of recognizers and to serve as a theorem
proving program. The program will discover rules for the
recognition of grammatical strings when given a simple BNF
grammar; it is a device for both constructing recognizers
and proving them valid."
CR 7267.

3. 89 Markov, A. A.
Theory of algorithms.
U. S. Bureau of Standards
OTS 60-51085, Clearinghouse, Springfield, Va.
# theory #

3. 90 Martin, D., and Estrin, G.
Models of computations and systems.
# theory #

3. 91 Martin, W. A.
A left-to-right then right-to-left parsing algorithm.
Proc. of 2'dnd Hawaii Int'l Conference on System Sciences,
Honolulu, (Jan 1969), 941-942.
# parsing #

3. 92 Maurer, W. D.
A theory of computer instructions,
MEM MAC-M-262, Project MAC, MIT, Cambridge, Mass., (Sept
1965).
# translators, program&g #

3. 93 McCarthy, J.
Recursive functions of symbolic expressions and their
computation by machine. Part I.
# theory, compiling #
"This article gives an introduction to the LISP language and
system in addition to presenting techniques for dealing with
recursive computations on symbolic expressions."
3.94  McCarthy, J.
A formal description of a subset of ALGOL.
In Formal Language Description Languages for Computer
Programming, T. B. Steel, Jr., (Ed.), North Holland
Publishing Co, Amsterdam, (1963), 1-12.
  # formal #

3.95  McCarthy, J., and Painter, J.
Correctness of a compiler for arithmetic expressions.
Stanford \textit{\&}, AD-662880, (\textit{\textbf{Apr}} 1966), 15.
  # compiler #

3.96  Narasiah, R.
Programming languages and computers: a unified \textit{meta-theory}.
Academic Press, N. Y.
  # language, theory #

3.97  Orgass, R. J.
A mathematical theory of computing machine structure and
programming.
RC 1463, TBM, Yorktown Heights, N. Y., (1967).
  # computing structure #

3.98  Painter, J. A.
Semantic correctness of a compiler for an ALGOL-'like
language.
  # theory #

3.99  Parikh, R. J.
On context-free languages.
\textit{3 ACM} 13, 4 (\textit{Oct} 1966), 570-581.
  # context-free languages, grammar #
"The author investigates certain properties of context-free
grammars. Questions regarding structure, possible ambiguity
and relationship to finite automata are considered."
  CR 11431.

3.100  Parikh, R. J.
Language generating devices.
Quarterly Progress \textit{Rept.} No. 60,
Research Lab of Electronics, \textit{MIT}, Cambridge, \textit{Mass.}, (Jan
1961), 199-212.
  # generator #

3.101  Parikh, R. J.
On context-free languages.
  # context-free languages #
3.102 Paul, M.  
Zur struktur formaler sprachen.  
Dissertation, Mainz, (1962).  (German).  
# formal language #

3.103 Paul, M.  
A general processor for certain formal languages.  
# formal languages, processors #

3.104 Paul, M. C., and Unger, S. H.  
Structural equivalence of context-free grammars.  
# context-free grammars, parsing #

3.105 Pollack, R. W.  
Compiler techniques,  
# compilers, translators, interpreters, processors #  
"This book presents a summary of the basic techniques necessary for the implementation of compilers. A wide variety of subjects is covered including syntax, parsing, resource allocation, detection and correction of errors, and details of compiler construction."

3.106 Red’ko, V. N.  
The syntactic analysis of context-free languages.  
In Cybernetics (Nay-June 1966).  
Translation of Kiberaetika 2, 3 (May-June 1966), 52-63.  
(Russian).  
# syntactic analysis, context-free, languages #  
CR 8246.

3.103 Riguett, J.  
Programmation et theories des categories.  
# theory #

3.108 Roberts, A. E.  
The construction of recognizers.  
# recognizers #  
"This paper is a theoretical treatment of one method of constructing a recognizer from an arbitrary context-free grammar?  
CR 11099."
3.109 Rosenkrantz, D.J.
Programmed grammars and classes of formal languages.
grammars, formal language
"Two new classes of grammars which lie between context-free
and context-sensitive grammars are defined."

3.110 Ross, D. T.
An algorithmic theory of language,
formal language

3.111 Ross, D. T.
On context and ambiguity in phrasing.
ACM 7, 2 (Feb 1964), 131-133.
$parsing$, syntactic
"This paper presents a discussion of some aspects of Floyd's
bounded context grammars and Irons' structural connections
in regard to ambiguity in parsing."

3.112 Ross, D., and Rodriguez, J.
Theoretical foundation of the computer aided design system.
compilers
"The authors discuss plexes and precedence as a basis for
the AED system."
CR 6316.

3.113 Scheinberg, S.
Note on the Boolean properties of context-free languages.
Info and Control 3 (1960), 372-375.
context-free languages
CR 0925.

3.114 Schutzenberger, M. P.
Context-free languages and pushdown automata.
Info and Control 6 (Sept 1963), 246-264.
context-free languages

3.115 Schutzenberger, M. P.
Classification of Chomsky languages,
In: Formal Language Description Languages for Computer
Programming, T. B. Steel, Jr., (Ed.), Worth Holland
Publishing Co., Amsterdam, (1966), 100-104.
formal languages

3.116 Schutzenberger, M. P.
Some remarks on Chomsky's context-free languages.
context-free
3.117 Schutzenberger, M. P., and Chomsky, N.
The algebraic theory of context-free languages.
Computer Programming and Formal Systems, Braffort, P. and Hirshberg, D., (Eds.), North Holland Publishing Co.,
Amsterdam,
context-free languages

3.118 Shamir, E.
On sequential languages.
formal languages

3.119 Tarski, A.
Logic, Semantics, Metamathematics.
semantics, meta-languages
This is a collection of articles which are useful to the compiler writer if he is interested in the theory of semantics.

3.120 Unger, S. H.
A global parser for context-free phrase-structure grammars.
parser, phrase-structure, syntax-directed compiler
The author gives an algorithm for analyzing any, context-free phrase-structure grammar. Any sentence in the language is parsed by a program generated by the algorithm.

3.121 Walk, K.
Entropy and testability of context-free languages.
In Formal Language Description Languages for Computer Programming, T. B. Steel, Jr., (Ed.), North Holland Publishing Co., Amsterdam, (1966), 105-123.
context-free language

3.122 Wirth, N.
A basic course on compiler principles,
syntax-directed, compiler
An introduction to phrase-structure languages is presented as a basis for devising syntax-directed compilers. Both theory and applications are presented.
3.123 Wirth, N., and Weber, H.
EULER—a generalization of ALGOL and its formal definition:
Part I, II,
A method for defining programming languages (simple precedence grammars) is developed which introduces a
rigorous relationship between structure and meaning. A
generalization of ALGOL is described in detail to show that
block-structure, procedures, etc. can be adequately handled.
Part II contains a formal description of the language EULER.
An attempt is made to generalize ALGOL to create a simpler
and more flexible language.

3.124 Wolman, B. L.
Operators for manipulating language structures.
ACM Symposium on Symbolic and Algebraic Manipulations,
"The algorithmic theory of languages provides a language
structure capable of representing the syntactic and semantic
structure of statements in algebraic, procedural or
graphical languages. Utilizing the semantic sequencing
information in the structure, operators defined for atomic
forms may be applied to arbitrarily complex structures to
provide a powerful manipulation capability. The author
describes a system constructed on these bases."

3.125 Wood, D.
The theory of left factored languages: Part I, II.
"Part I is relevant for syntax directed top-down
compilation. Part II includes two appendixes that extend
the argument."

7.126 Wood, D.
"The normal form theorem—another proof.
Comp J 12, 2 (Ray 1969), 139-147.
"This article presents some theory applicable to
context-free languages."

3.127 Woods, W.A.
Context-sensitive parsing,
Comm ACM 13, 6 (July 1970), 437-444.
3.128 Younger, D. H.
Recognition and parsing of context free languages in time \( n^3 \) cubed,
# parsing, context free languages #
4. 0 RESOURCES ALLOCATION

4. 1 Aho, A. V., Sethi, R., and Ullman, J. D.
A formal approach to code optimization.
SIGPLAN 5, 7 (July 1970), 86-100.
# optimization #

4. 2 Allen, F. E.
Control flow analysis.
SIGPLAN 5, 7 (July 1970), 1-19.
# optimization #

4. 3 Allen, F. E.
Program optimization.
Annual Review in Automatic Programming, Vol 5, (1965),
239-279. Pergamon Press, N.Y.
# optimization #
"Machine independent and language independent methods of
optimizing the execution times of compiled programs are
described. The approach is based on the topological
characteristics of the program. Optimization techniques
include eliminating redundant instructions, folding, moving
instructions from one part of the program to another,
reducing the strength of operators, replacing tests, etc."

4. 4 Bagwell, T. Jr.
Local optimization.
SIGPLAN 5, 7 (July 1970), 52-66.
# optimization #

4. 5 Belady, L. A., and Kuehner, C. J.
Dynamic space-sharing in computer systems.
# allocation, processor #
"The authors explore the problem of optimizing program
execution in a space-shared environment. A relationship
between space-sharing, program behavior, and processor
efficiency is given."

4. 6 Bolas, R. J.
Optimization problem in extensible compilers.
SIGPLAN 5, 7 (July 1970), 127.
(abstract).
# optimization, compiler, extensible compilers #
4. 7 Breuer, M. A.
Generation of optimal code for expressions via factorization.
Comm ACM 12, 6 (June 1969), 333-340.
# compiler, optimization #
"The author presents methods for increasing the efficiency of the object code produced while compiling any given expression. Each expression is broken up into a set of sub-expressions each of which occurs in more than one other expression or sub-expression. These sub-expressions are put in a definite sequence such that computing occurs in correct sequence and storage requirements are reduced. The procedures used are heuristic in nature."

4. 8 Pusam, V. A., and Englund, D. E.
Optimization of expressions in Fortran.
Coma ACM 12, 12 (Dec 1963), 666-674.
# optimization #

4. 9 Cheatham, T. E., Jr., and Standish, T. A.
Optimization aspects of compiler-compilers.
SIGPLAN 5, 10 (Oct 1970), 10-17.
# compiler-compiler, optimization #

4. 10 Clark, E. R.
On the automatic simplification of source-language programs.
Coma ACM 10, 3 (Mar 1967), 160-166.
# language, compiler #
"The author describes methods to simplify programs automatically. Simplification is based on the form of the program and the knowledge obtained by a processor."
CR 11212.

4. 11 Cocke, J.
Global common subexpression elimination.
# optimization #

4. 12 Cocke, J., and Biller, R.
Some analysis techniques for optimizing computer programs.
Proc. of 2nd Hawaii Int'l Conference on System Sciences, Honolulu, (Jan 1969), 143-146.
# code optimization #

Stochastic evaluation of a static storage allocation.
Comm ACM 4, 10 (Oct 1961), 460-464.
# storage allocation #
"This article develops a method of evaluating the 'efficiency of a specific allocation of a given library.' The method described utilizes limits and summations."
4. 14 Collins, G. O.
Experience in automatic storage allocation.
Comm ACM 4, 10 (Oct 1961), 436-440.
# allocation #
"Collins discusses storage allocation as a part of a programming system, with some history of allocation techniques. This is a fairly general article which provides good overview of the subject. The author presents his point of view: that one must have a programming system powerful enough so that it can supervise and participate in the execution of the programs, in addition to being able to help prepare the programs."
CR 2150.

4. 15 Czaja, L., and Szore, P.
Storage allocation for ALGOL.
Algorytmy 4, 7 (1967), 91-111. (Russian),
# storage allocation #
CR 15191.

4. 16 Dantzig, G. B., and Reynolds, G. H.
Optimal assignment of computer storage by chain decomposition of partially ordered sets.
# resource allocation, optimization #

4. 17 Darden, S. C., and Heller, S. B.
Streamline your software development.
# optimization, resource allocation #
"This article presents a case history of the optimization of an ALGOL compiler."

4. 18 Dawkins, G. S.
Design-of a language for optimization.
# language, optimization #

4. 19 Day, W. R. E.
Compiler assignment of data items to registers.
# compilation, optimization #
"This paper presents three algorithms for assigning data items to registers. Optimization is discussed."
4. 20  
Degtyarev, Ye. K.
Analysis and optimization of the structure of an asynchronous digital computer.
Izv. Akad. USSR Tehn. Kibernet, (1965), (Russian),
# optimization #
CR 14353.

4. 21  
Denman, H. H.
Computer generation of optimized subroutines,
Proc ACM 14th Nat'l Conf. (1959), 40.
# optimization #
"The author presents a method by which a digital computer may be programmed to generate function-evaluation subroutines which have a fair amount of accuracy within an interval suited to the programmer's needs. The method utilizes Chebyshev polynomial approximation."

4. 22  
Cenning, P. J.
Resource allocation in multiprocess computer systems.
# allocation #
CR 15412.

4. 23  
Dennis, J. B.
Segmentation and the design of multiprogrammed computer systems,
3 ACM 12, 4 (Oct 1965), 589-602.
# compiler, allocation #
"This paper describes the problems inherent in multiprogramming: dynamic allocation, referencing of common information from many programs, etc. Also described are the concepts of name space vs. memory space and segmentation."

4. 24  
Semi-automatic allocation of data storage for PACT I.
# storage allocation #
"The general problem of storage allocation is discussed, along with specific problems encountered in constructing the storage allocation section of the PACT I compiler.

4. 25  
Elson, M., and Rake, S. T.
Code-generation technique for large-language compilers.
IBM Systems 3 9, 3 (1970).
# compiler, optimization #
"A technique for generating optimized code is presented. Optimization is both local and global. The program operates on a meta-machine dealing with tree structures which represent the text to be compiled. The approach readily lends itself to extendible languages and the modification of existing languages."
Fateman, R. J.  
Optimal code for serial and parallel computation.  
# code optimization #

Pinkelstein, M.  
A compiler optimization technique.  
Comp J 2, 1 (May 1968), 22-25.  
# optimization, compilers #

"The author introduces the concept of 'deferred store' and describes how it can be used in compilation. A more optimal machine code is obtained as a result of the usage of this technique when compiling programs."  
CR 15402.

Pirth, A. W. O.  
Optimization problems: solution by an analogue computer.  
# optimization #

"The author discusses optimization problems in general, paying special attention to linear programming. Constraints are utilized in the solution, which is done on an analogue computer. Three possible types of solutions and a good example are included."  
CR 1127.

Fitzwater, D. R.  
A storage allocation and reference structure.  
Comm ACM 7, 9 (Sept 1964), 542-545.  
# allocation, storage #

"A method is described for adding subscripted variable capability to autocoder type systems."  
CR 6933.

Poster, J. M.  
A syntax improving program.  
Comp 3 11, 1 (1968), 31-34.  
# compiler, syntax, parsing #

"The author describes a program which accepts a grammatical definition of a language as data and transforms it into an equivalent grammar that can be parsed by a simple parsing algorithm,"

Fotheringhaa, J.  
Dynamic storage allocation in the Atlas computer, including an automatic use of a backing store,  
Comm ACM 4, 10 (Oct 1961), 435-436.  
# storage allocation #
4. 32 Prailey, D. J.
Expression optimization using unary complement operators.
SIGPLAN 5, 7 (July 1970), 67-85.
# optimization #

4. 33 Gatwick, J. V.
Data storage in compilers.
BIT 4, 3 (1964), 137-140.
# storage, compilers #

4. 34 Gear, C. W.
High speed compilation of efficient object code.
Comm ACM 8, 8 (Aug 1965), 483-488.
# compilation, optimization #
"The author describes a method for partial optimization of code which is non-machine dependent. The intention is to find a middle ground between compiling and efficient object code."
CR 9000,

4. 35 Haddon, B. K., and Waile, W. M.
A compaction procedure for variable-length storage elements.
# allocation #
"The authors present a procedure for compacting the storage such that all of the free-space forms a single element."
CR 13547.

4. 36 Haynes, H. R., and Schutte, L. J.
Compilation of optimized syntactic recognizers from Floyd-Evans productions.
SIGPLAN 5, 7 (July 1970), 38-51.
# syntax analysis, optimization, compiler #

4. 37 Heising, W. P., and Larner, P. A.
A semi-automatic storage allocation system at loading time.
Comm ACM 4, 10 (Oct 1961), 446-449.
# allocation #
"This article is concerned with the referencing, loading, and overlaying of the segments of a modular program. The article describes a storage allocation system which has the following advantages: 1) no recompilation or assembly, 2) storage overlay can be planned after programming within certain limits, 3) the same program can be run on many machines, and 4) simplicity. Objectively, the author, in addition to listing the advantages also lists the disadvantages of the system."
CR 3895,
4. 38 Hellerman, R.
A control system for multiprogram use of core storage,
IBM Dickei 10, 501 (June 1961).
# allocation, storage #

4. 39 Hill, V., Langaaack, H., Schwarz, A. R., and Seegmueller, G.
Efficient handling of subscripted variables in ALGOL 60
compilers.
Proc 1962 Rome Symposium on Symbolic Languages in Data
# compiler, allocation #

4. 40 Holt, A. W.
Discussion of the problem of definition of storage
allocation.
Comm ACM 4, 5 (May 1961), 210-211.
# storage allocation #
"This discussion is concerned mostly with the transfer of
informational entities from one level of storage to another.
The term 'unallocated program' is defined."

4. 41 Holt, A. W.
Program organization and record keeping for dynamic storage
allocation.
Comm ACM 4, 10 (Oct 1961), 422-431.
# allocation #
"Holt gives a method of 'allocation interpretation' which
would divide a large program into units and then convert
floating code into fixed code, load the program into care
and perform computations by unit. The author discussed in
depth, goes into considerable detail, covering such topics
as program description with respect to inter- and
intra-programmatic structure, system records, and segment
records."
CR 2696, 4369.

4. 42 Horwitz, L. P., Karp, R. M., Miller, R. E., and Winograd, S.
Index register allocation.
3 ACM 13, 1 (Jan 1966), 43-61.
# allocation, optimization #
"A procedure for index register allocation is described.
The rules of this procedure are shown to yield an optimal
allocation for straight line programs."

4. 43 Buxtable, D. A. R.
On writing an optimizing translator for ALGOL 60.
In Introduction to Systems Programming, P. Wegner, (Rd.),
# translator, optimization #
4. 44 Ichbiah, J. D., and Horse, S. P.
A technique for generating almost optimal Floyd-Evans
productions for precedence grammars,
# precedence grammars, syntax-directed analysis #

4. 4s Iliffe, J. K., and Jodeit, J. G.
A dynamic storage allocation scheme.
Coap J 5, 3 (Oct 1962), 200-209.
# storage allocation, dynamic allocation #
"This article presents a system of semi-automatic storage
control which is based on the use of codewords. The
advantages of this system include simplification of array
indexing, the extension of problem-oriented languages, and
the combination of 'the normal functions of a loading
routine with the ability to allocate storage dynamically'."
CR 4175.

4. 46 Jensen, J., Mondrup, P., and Naur P.
A storage allocation scheme for ALGOL 60.
# allocation #
"This article describes a storage allocation scheme for the
DASR, a machine with a 2048 instruction core storage and a
magnetic drum. Dynamic block administration is illustrated
by ALGOL procedures, and various facets of storage
management are discussed."
CR 2614.

4. 47 Jodeit, J. G.
Storage optimization in programming systems.
# storage allocation, optimization #

4. 48 Katz, J. H.
Optimizing hit-time computer simulation.
# optimization #
"This paper presents techniques applicable to any general
purpose compiler, the results of which are to optimize
bit-time computer simulation. Among the properties these
techniques give to the Boolean compiler are improvement of
object code efficiency and the automatic selection of an
optimum set of subroutines for evaluating the given set of
Boolean functions, given a specified memory constraint,"
4. 49  Kelley, J. E., Jr.  
Techniques for storage allocation algorithms.  
Comm ACM 4, 10 (Oct 1961), 449-454.  
# allocation #  
"This article presents a few helpful techniques for  
approaching allocation problems. Among the methods  
discussed are dynamic programming and heuristic methods,  
The article itself is valuable in that it is general and  
that the techniques presented can be universally applied.*  
CR 2149.

4. so ,  Moulton, K. C.  
A fast storage allocator.  
Comm ACM 5, 10 (Oct 1965), 623-625.  
# allocation #  
"A fast bookkeeping method is described which is  
particularly appropriate for list structure operations is  
described. The system makes available blocks which are  
halved repeatedly when smaller blocks are needed."  

4. 51  LaFrance, J. A.  
Optimization of error-recovery in syntax-directed parsing  
algorithms.  
SIGPLAN 5, 7 (July 1970), 128.  
(Abstract),  
# optimization, parsing #  

4. 52  LaFrance, J. A.  
Optimization of error recovery in syntax-directed parsing  
algorithms.  
SIGPLAN 5, 12 (Dec 1970), 2-17.  
# optimization, parsing, syntax-directed translation #  

4. 53  Landin, P. J.  
The mechanical evaluation of expressions.  
Carp J-6 (1963), 308.  
# compiling #  
"Landin is concerned with the structural simplification of  
expressions."  
CR 6677.  

4. 54  Lowry, E. S., and Medlock, C. W.  
Object code optimization.  
# optimization, compiling #  
"The author discusses optimization techniques used by the  
OS/360 Fortran H compiler. Optimization techniques consist  
of combining common sub-expressions, moving loop independent  
computations out of loops, induction variable optimization  
and register allocation. The authors apply control flow and  
data flow analysis techniques to transform programs to  
improve object time efficiency."
4. 55
Luccio, F.
A comment on index register allocation.
Comm ACM 10, 9 (Sept 1967), 572-574.
# allocation #
"The author describes a technique for optimal index register allocation in straight line programs which has a smaller number of enumerations."

4. 56
Maher, R. J.
Problems of storage allocation in a multiprocessor multiprogrammed system.
Comm ACM 4, 10 (Oct 1961), 421-422.
# allocation #
"The author discusses the problems of allocation in the Burroughs B5000 Information Processing System. Mainly, the author discusses the actual routines used in the Burroughs ES000 and glosses over the actual problems that still exist by stating that work is being done on them."
CR 2148.

4. 57
McKeeman, W. M.
Peephole optimization,
Comm ACM 8, 7 (July 1965), 443-444,
# optimization #
"A simple method for discarding redundant instructions during the final stage of compilation is described and examples are given."
CR 8065.

4. 58
Medlock, C. W., and Lowry, E. W.
Global program optimization.
XBM (Confidential) TR 00.1330, (Sept 1965).
# optimization #

4. 59
Naur, P.
The performance of a system for automatic segmentation of programs within an ALGOL compiler (GIER ALGOL).
Comm ACM 8, 11 (Nor 1965), 671-676, 686.
# compiler 8
"The Gier ALGOL compiler for handling transfer or program segments from drum store to core at execution time is presented. The system is described and evaluated."
4. 60 Nievergelt, J.
On the automatic simplification of computer programs.
Comm ACM 8, 6 (June 1965), 366-370.
"This paper presents the problem of designing a program which will simplify other programs without knowing the meaning of the program but only its form. An attempt is made to find transformation which yield equivalent programs."
CR 8247.

4. 61 O'Neill, R. W.
A preplanned approach to a storage allocation compiler.
Comm ACM 4, 10 (Oct 1961), 417.
"This is a short discussion of considerations for designing a storage allocating compiler and touches on means for minimizing execution time."

4. 62 Painter, J. A.
Effectiveness of an optimizing compiler for arithmetic expressions.
SIGPLAN 5, 7 (July 1970), 101-126.

4. 63 Pollack, B. W.
Compiler techniques.
"This book presents a summary of the basic techniques necessary for the implementation of compilers, a wide variety of subjects is covered including syntax, parsing, resource allocation, detection and correction of errors, and details of compiler construction."

4. 64 Randell, R., and Kuehner, C. J.
Dynamic storage allocation systems.
"The authors present a method of characterizing dynamic storage allocation systems according to the functional capabilities provided and the techniques used."

4. 65 Aidgwap, R. K.
Compiling routines.
Proc ACM 7th Nat'l Conf., Toronto, (1952), 1-S.
"This paper demonstrates the time advantages in using a compiler to assemble library routines into a program instead of writing the program from scratch."
4. 66 Riskin, B. N.
Core allocation based on probability.
Comm ACM 4, 10 (Oct 1961), 454-459.
# allocation #
"A real-time system with multiple input sources (including a drum) presents some particular core allocation problems. This article discusses an efficient allocation technique for a real-time system."

4. 67 Roberts, A. E.
A general formulation of storage allocation.
Comm ACM 4, 10 (Oct 1961), 419-420.
# allocation #
"The author gives a 'formal picturization of a computer allocation process.' It is done with a given computer, M, which is associated to a ficticious M', which differs from M in that it has unbounded primary storage. The author discusses mappings of an M' program to M-admissible subprograms and a linking set of interludes. A general process for storage allocation is presented which would decouple a program into segments, mapping the segments into storage and provide linkages between segments."

4. 68 Rutledge, J. D.
Approach to definition of storage allocation.
# storage allocation #
"Rutledge presents a very general approach to the allocation-compilation process in this paper; it is designed to provoke discussion at a future ACM meeting on the subject."

4. 69 Sams, B. H.
The case for dynamic storage allocation,
# allocation #
"@Dynamic storage allocation and preplanned storage allocation are described and support is given to dynamic storage allocation as the preferred form of the two."

4. 70 Sams, B. H.
Dynamic storage allocation for an information retrieval system.
Comm ACM 4, 10 (Oct 1961), 431-433.
# allocation #
"When dynamic allocation is required throughout processing it can be handled by means of an allocation code which does the required book-keeping. Such a system is described for an information retrieval system."
4. 71 Sattley, K.
Allocation of storage for arrays in ALGOL 60.
# allocation, translator #
"The author presents a method of dynamic allocation of storage at runtime for ALGOL 60 arrays which have dimensions defined by variables. Some sample programs are given in ALGOL to illustrate the process of allocation."

4. 72 Schneider, V.
A system for designing fast programming language translators.
# translator, optimization #

4. 73 Sethi, R., and Ullman, J. D.
The generation of optimal code for arithmetic expressions.
3 ACM 17, 4 (Oct 1970), 715-728.
# optimization, resource allocation #

4. 74 Strachey, C., and Wilkes, M. V.
Some proposals for improving the efficiency of ALGOL 60.
# compiler, optimization #
CR 1929.

4. 75 Walter, K. G.
Compiler optimization of object programs.
# compiler, optimization #
"The author examines in detail a Fortran IV and an ALGOL 60 compiler. He presents some heuristic approaches to partitioning programs into pieces where it is possible to determine the effect of changes within the pieces on the entire program. The author concentrates on eliminating common sub-expressions and invariant expressions from explicit loops and recursive procedures."
CR 13630.

4. 76 Wegner, P.
Notes on the ACM Computer Optimization Symposium, Urbana.
# compiler #

4. 77 Uheeling, R. F.
Optimizers, their structure.
# optimization #
"The author takes a look at the philosophy of optimization."
CR 0953.
4. 78 Wieland, M.
Storage allocation for variables in ALGOL programs.
(German).
# storage allocation #

4. 79 Yershov, A. P.
ALPHA--an automatic programming system of high efficiency.
# compiler, optimization, translator #
"This paper describes the implementation of an extended
ALGOL 60 compiler on the Russian M-20 computer.
'Capabilities are described and details of optimization
techniques are given."
5. 0 ERRORS -- DETECTION AND CORRECTION

5. 1 Arden, R. W., Galler, B. A., and Graham, R. M.
An algorithm for equivalence declarations,
Comm ACM 4, 7 (July 1961), 310-314.
"This article describes an algorithm for providing a
storage assignment for each variable and array occurring in
any EQUIVALENCE statement, which is done by working with
one equivalence class of arrays at a time. Several figures
are included to aid the authors in explaining their
algorithm."
CR 1932.

5. 2 Blair, C. R.
A program for correcting spelling errors.
Info and Control 3 (May 1960), 60-67.
"error correction."

5. 3 Conway, R. W., and Maxwell, W. L.
CORC--the Cornell computing language.
Comm ACM 6, 6 (June 1963), 317-321.
"CORC is designed for use by the non-professional programmer
who is not highly concerned with the mechanics of a
computer. The compiler provides extensive diagnostics There
are only nine different types of statements, no
compiler-controlling declarations, and no decimal numbers.
CORC will correct spelling errors, grammatical errors, and
punctuation errors whenever possible."
CR 4778.

5. 4 Daverou, F.
A technique for computer detection and correction of
spelling errors.
Comm ACM 7, 3 (Mar 1964), 171-176.
"error detection, error correction."

5. 5 Evans, T., and Darley, D.
On-line debugging techniques: a survey.
"This paper is a survey of on-line debugging techniques used
in time-sharing systems. Also discussed are possible future
directions for work in this area."
CR 0751.
5. 6  
Freeman, D. N.  
Error corrections in CORC—the Cornell Computing language.  
Proc AFIPS 1964 FJCC, Vol 26, 15-34.  
"CORC is a teaching language used at Cornell which has extensive error correction procedures. The language is described briefly and the error-correction procedures are described in detail."  
CR 7626.

5. 7  
Irons, P. T.  
An error-correcting parse algorithm.  
Comm ACM 6, 11 (Nov 1963), 669-673.  
"This article presents an algorithm which corrects syntax in a program. The program is parsed until an incorrect statement is found. The program then makes a tentative correction, and continues making tentative corrections until one is found that will parse consistently. This algorithm may have some importance in the future in the area of pattern recognition."  
CR 5670.

5. 8  
LaFrance, J. A.  
Optimization of error-recovery in syntax-directed parsing algorithms.  
SIGPLAN 5, 7 (July 1970), 128.  
(Abstract).  
"optimization, parsing"  

5. 9  
LaFrance, J. A.  
Optimization of error recovery in syntax-directed parsing algorithms.  
SIGPLAN 5, 12 (Dec 1970), 2-17.  
"optimization, parsing, syntax-directed translation"  

5. 10  
Morgan, H. L.  
Spelling corrections in systems programs.  
Comm ACM 13, 2 (Feb 1970), 90-94.  
"error detection, error correction"  

5. 11  
Moulton, P. G., and Muller, M. E.  
DITRAN—a compiler emphasizing diagnostics.  
"The authors emphasize improvement of diagnostic capabilities of compilers. DITRAN (Diagnostic forTRAN) has extensive error checking capabilities,"  
CR 11927.
5. 12 Pollack, B. il.
Compiler techniques.
# compilers, translators, interpreters, processors #
"This book presents a summary of the basic techniques necessary for the implementation of compilers. A wide variety of subjects is covered including syntax, parsing, resource allocation, detection and correction of errors, and details of compiler construction."

5. 13 Rosen, S., Spurgeon, R. A., and Donnelly, J. K.
PUFPT—Perdue University fast Fortran translator.
Comm ACM 8, 11 (Nov 1965), 661-666.
# compiler #
"This paper describes a high-speed system for the complete Fortran IV language, including the subroutine library. The system included an elaborate diagnostic message routine."

5. 14 Weinberg, G. M., and Gressett, G. L.
An experiment in automatic verification of programs,
Comm ACM 6, 10 (Oct 1963), 610-613.
# compiler, error-detection #
"This paper discusses the effectiveness of a compiler at replacing explicit verification. The authors examine three levels of error, control, computation and format, and their detection. They come to the conclusion that 'a properly constructed compiler ... can replace an explicit program verification technique with great effectiveness, (with) many fringe benefits and low cost.'"
CR 5306.
6. 0

**COMPILER IMPLEMENTATION IN GENERAL**

6. 1

**Allard, R. W., Wolf, K. A., and Zealin, R. A.**
Some effects of the 6600 computer on language structures.
Comm ACM 7, 2 (Feb 1964), 112-119.

"This article describes an intermediate level language for
the CDC 6600 computer which reflects the structure of the
machine, Methods for implementing this language are
considered.
CR 5999.

6. 2

**Arden, B. W.**
On the construction of algorithm translators.
Proc ACM 14th Nat'l Conf. (1959), 23.

6. 3

**Arden, R. W., Galler, R. A., and Graham, R. M.**
The internal organization of the HAD translator.

"HAD is a language which somewhat resembles ALGOL 60. Its
translator has been designed for maximum translation speed
and efficiency, The translator is divided into three parts:
statement decomposition, storage allocation, and generation
of the object program. In each of the parts, emphasis is
placed on the use of tables for storage. The authors
explain each part in a fair amount of detail, giving an
easily attained insight to the make-up of this particular
compiler."

6. 4

**ACM Compiler Symposium.**
Papers presented at the ACM Compiler Symposium, November
17-14, 1960, Washington, D.C.
Comm ACM 4, 1 (Jan 1961), 3-84.

"The entire January 1961 issue of Coma ACM is devoted to
articles on various aspects of compilers."

6. 5

**Eackus, J. W., Bauer, P. L., Green, J., Katz, C., McCarthy,
J., Naur, P., Perlis, A. J., Rutishauser, H., Saaelson, K.,
Vauquois, B., Regstein, J. H., van Rijngaarden, A., and
Woodger, M.**
Revised report on the algorithmic language ALGOL 60.
Comm J 5, 4 (Jan 1963), 349-368.

"This report is the complete defining description of ALGOL
60. The topics discussed, in order, are: language
structure, basic symbols, identifiers, numbers, strings,
expressions, statements, and declarations. At the end are
examples of procedure declarations."
CR 4540.
6. 6 Barhieri, R., and Morrissey, J.
Computer compiler organization studies.
"The authors discuss compiler organizations to increase
efficiency of the system in the areas of better hardware
utilization, reduced compilation time, etc. Emphasis is
laid on incremental translation, re-usable compilers, and
the like."

6. 7 Barrett, W., and Mitchell, A. J.
An extended Autocode for PEGASUS,
Comp J 6, 3 (Oct 1963), 237-240.
"Extended Autocode was written for a Pegasus computer in a
language based on Pegasus Autocode. Important new features
of the Autocode include the ability to handle long
arithmetic statements, whereas before, only single-operator
arithmetic statements could be handled. Prior to the
conclusion, the author briefly describes the operation of
the compiler."
CR 5359.

6. 8 Blatt, J. M.
Comments from a Fortran user.
"Compilers are designated as either A or B types, depending
upon whether the chief use is for small problems coded by
people who are essentially not programmers or for large
problems which require efficient use of machine space."
CR 0632.

6. 9 Bobrow, D. G., (Ed),
Symbol manipulation languages and techniques.

6. 10 Breed, L. M., and Lathwell, R. H.
The implementation of APL/360.
In Interactive Systems for Experimental Applied Mathematics,
Klerer, M. and Reinfelds, J., (Eds.), Academic Press, N. Y.,
(1968), 390-399.

6. 11 Caracciolo Di Porino, A.,
On a research project in the field of languages for
processor construction,
# processor #
"Di Farina discusses the requirements for a programing
language for processor construction and for a meta-language
which will provide a complete formal description of a
language."

6. 12 Caracciolo Di Farina, A., and Cecchi Morandi, M.
Su uno schema di traduttore per l'ALGOL.
(An ALGOL translation scheme.)
*Atti del convegno sui linguaggi simbolici di programmazione,
AICA, (Jan 1962), 103-120. (Italian).*
# translator, semantics, language #

6. 13 Cardenas, A. F., and Rarplus, W. J.
Design and organization of a translator for a partial
differential equation language.
# translator #

6. 14 Cheatham, T. E.
The architecture of compilers.
(1964).*
# compiler #

6. 15 Cheatham, T. E., Collins, G. O., and Leoaard, G. P.
CL-I, an environment for a compiler.
*Comm ACM 4, 1 (Jan 1961), 23-28.*
# compiler #
"The authors found a need for psograrrer-program
intercommunication. They filled the need with a CL-1
programming system, which, in addition to the compiler,
incorporates a filing program, data and separate data
descriptions. The CL-1 environment provides a Monitor and a
master file setup for large-scale information processing
problems. It is an entire programming system, rather than
simple a compiler."
6. 16 Cocke, J., and Schwartz, J. T.
Programming languages and their compilers: preliminary notes.
2d rev. version.
Design characteristics of the WATFOR compiler.
SIGPLAN 5, 7 (July 1970), 25-36.
Formal structure of ALGCL and simplification of its description,
symbolic languages in data processing.
Gordon and Breach, N. Y., (1962), 75-82.
Design of a language-for optimization.
Design of a language-for optimization.
Design of a language-for optimization.
6. 21  Dijkstra, E. W.
On the design of machine independent programming languages.

"This article gives an approach to evaluating a language. Somewhere the points the author deems important are: 1) facilitation of the programmer as much as possible, 2) the importance of semantics definition, which has as reaction to an arbitrary process description in this language the actual execution of this process*, and 3) minimization of redundancy. The concern is mostly with the characteristics of languages and slightly concerned with what a translator needs to know about a language. It is mostly background material for a translator-writer."

CR 5696.

6. 22  Dijkstra, E. W.
Flaking a translator for ALGOL 60.
APIC Bull. 7 (Hay 1961),

"This article presents the author's experience in the construction of an ALGOL 60 translator. The approach used is general because the object program is not assumed to be machine language. Also, the translation process described is one that 'reads the ALGOL program from BEGIN to END, simultaneously producing * the corresponding object program'." CR 5677.

6. 23  Duncan, F. G.

"This paper describes two ALGOL compilers, both approximately the same size, both being written in User Code, both accepting identical versions of ALGOL 60. They differ in that one compiler has emphasis on fast compilation while the other is 'aimed at recognizing and giving special treatment to certain situations amenable to optimizations'." CR 3531,
6. 24  Elgot, C. C., and Robinson, A.
Random access stored-program machines, an approach to
programming languages,
# compiler, language #
"A class of machine models is introduced as a basis for
discussion. Address modification is discussed and the
relationship between problem-oriented languages and machine
languages is considered-
CR 8657.

6. 25  Ershov, A. P., and Rar, A. P.
SYGMA, a symbolic generator and macro-assembler.
In Symbolic Manipulation Languages and Techniques,
# generator, macro-assembler #
"The authors make an attempt to define a machine-oriented
programming system as a linguistic system with a number of
free parameters. The language is considered to be a
quadruple of 1) a set of syntactically admissible programs,
2) a programming processor, 3) a working processor with, 4)
its operational memory."
CR 14957.

6. 26  Evans, A.
An ALGOL 60 compiler.
87-124. Pergamon Press, N. Y.
# compiler #
"This paper is a thorough discussion of the internal
workings of an ALGOL translator used at Carnegie-Mellon
University. The compiler is partly based on Polish postfix
notation and the stack concept."
CR 7905.

6. 27  Evans, A., Jr.
An ALGOL 60 compiler,
# compiler #
CR 7905.

6. 28  Falkoff, A. D., and Iverson, K. E.
The APL/360 terminal system.
Research Report RC 1922, IBM Watson Research Center,
# compiler #
6. 29 Palkoff, A., D., and Iverson, K. E.
The APL/360 terminal system.
# compiler

6. 30 Feldman, J., and Gries, D.
Translator writing systems.
Coma ACM 11, 2 (Feb 1968), 77-113.
# compiler-compiler, translator, syntax, semantics
"This paper surveys critically the research efforts put into automating compiler writing. The paper includes the formal study of syntax and its application to translator writing, various approaches to automating semantic aspects of translator writing and other related topics such as the formal study of semantics, etc."
CR 14729.

6. 31 Yranciotti, R. G., and Lietzke, M. P.
The organization of the SHARE ALGOL 60 translator.
# translator, compiler
"This paper describes an ALGOL translator which operates under the Fortran Monitor System. The function of each phase, the general organization of the object code and the storage allocation scheme used for handling ALGOL block structure and dynamic array storage are described."

6. 32 Franklin, R. W.
Implementation of a compiler--GECOM.
# compiler
CR 5027.

6. 33 Garwick, J. V.
The definition of programming languages by their compilers.
In Formal Language Description Languages for Computer Programming, T. B. Steel, Jr., (Ed.), North Holland Publishing Co., Amsterdam, (1966), 139-147.
# language, compiler

6. 34 Garwick, J. V.
Data storage in compilers,
BIT 4, 3 (1964), 137-140.
# storage, compilers

6. 35 Garwick, J. V.
The definition of programming languages by the compiler.
# languages, compilers
6. 36 Gau, A. A,
Recursive processes and ALGOL translation.
# translation #

Programming languages, a NATO advanced study institute
summer school.
# languages, compilers #

6. 38 'Glass, R. L.
An elementary discussion of compiler/interpreter writing.
Computing Surveys 1, 1 (Mar 1969), 06-77.
# compiler, interpreter #
"An excellent overview of the problems involved in the
implementation of compilers is presented and interpreters Is
presented."

6. 39 Good, I. 3.
Number of possible strategies when writing compilers.
Comm ACM 11, 7 (July 1968), 474-474,
# compiling #
"The author gives a mathematical formula for the number of
strategies given K programming languages and J compilers, (J
< K)."

6. 40 Gorn, S.
Specification languages for mechanical languages and their
processors, a baker's dozen.
comm ACM 4, 12 (Dec 1961), 532-542.
# language, syntax #
"The author presents 13 languages, including the natural
languages, Backus Normal Form, trees, incidence matrices and
Turing machines. These languages provide different points
of view of the same problem and aid the the clarification of
problems in different ways."
CR 11417,

6. 41 Gorn, S.
The logical design of formal mixed languages.
# formal languages #

6. 42 Graham, R. N.
Notes on translation of algebraic languages.
In Summer Session on Advanced Programming, J. W. Carr, III,
# translation #
6. 43 Grau, A. A.
The structure of an ALGOL translator.
# translator #

6. 44 Grau, A. A.
A translator-oriented symbolic programming language.
3 ACM 9, 4 (Oct 1962), 480-487.
# translation #
"The author presents a target language which may be used as
an intermediate language in translation. Features of the
language include a small number of instruction types and
minima parenthesis structure. The author discusses the
operations and he ends with an application of this language
to the translation of ALGOL."
CR 3868.

6. 45 Grau, A. A.
Recursive processes and ALGOL translation.
# translation #
"The author describes a recursive translation process. The
approach used is the 'control push-down', which handles the
storage requirements of recursive subroutines used in the
transactor. The article includes a section of the
translation matrix actually used in the procedure."

6. 46 Green, J.
Symposium on languages for processor construction.
# processor #

6. 47 Gark, H., and Minker, J.
The design and simulation of an information processing
system.
3 ACM 8, 2 (Apr 1961), 260-270.
# compiler, processor #
"This article presents the design of an information
processing system which involves input/output,
interpretation, storage allocation, retrieval of data,
logical processing and correlation. These facets are
discussed, and the author concludes by noting some basic
problems of systems which handle language data."
6. 48 Hawkins, E. N., and Huxtable, D. H. R.
A multipass translation scheme for ALGOL 60.
# translator, optimization #
"A multi-pass translator produces more efficient code than a
one-pass translator; the authors give an in-depth
description of the one which they have written for the KDF
9. The main feature of this translation scheme is
efficiency in areas such as minimum running time and machine
storage requirements, 'The scheme operates in seven
distinct phases: 1) input, 2) syntactic check and reduction
of the input text to a form suitable for processing by the
later phases, 3) procedure classification, 4) storage
allocation, 5) index optimization, 6) translation and
formula optimization, and 7) final compilation and output'."

6. 49 Hellerman, H.
Experimental personalized array translator system.
Comm ACM 7, 7 (July 1964), 433-438.
# translator #
"The system uses a symbolic source language which contains
powerful statement types including numeric, Boolean
relational and selectional operators on operands which can
be arrays."
CR 6669.

6. 50 Hext, J. R.
Programming languages and compiling techniques.
# compiling, language #

6. 51 Higaan, B.
A comparative study of programming languages.
# syntax, -semantics, formal-languages, compiler #
"This book covers a wide variety of topics including formal
languages, macrogenerators, different programming languages,
list processing, etc."
CR 14510.

6. 52 Hopgood, P. R. A.
Compiling techniques.
pp.
# compilers #
"This book deals with modern techniques used in the design
and implementation of compilers. It covers data structures,
trees, graphs, arrays, tables, the description of languages,
lexical and syntactic analysis, code generation, storage
allocation and compiler-compilers. It is an excellent
introduction to the field."
6. 53  fngeraan, P. Z.
The parameterization of the translation process.
# translation #

6. 54  fngeraan, P. Z.
A syntax oriented translator.
# syntax, translation #
"This short monograph describes a single syntax-directed translator. It covers its definition, syntax, parsing and extensions and relationships to other translators.*
CR 11509.

A translation technique for languages whose syntax is expresible in extended Backus Normal Form,
# languages, translation #

6. 56  Irons, E. T.
A syntax directed compiler for ALGOL 60.
Comm ACM 4, 1 (Jan 1961), 51-06.
# syntax-directed, compiler, meta-language #
"Compilers not only translate one language into another but define the source language in terms of a second one, making it difficult to modify a compiler to reflect a language change. Irons has developed a compiler which keeps the two functions distinct, making modification simpler. The paper describes a compiling system consisting of a meta-language and a translator. Because of the separation of the two, extensions and modifications of the object language can be made more easily."

6. 57  Irons, E. T.
The structure and use of the syntax-directed compiler,
Annual Review in Automatic Programming, Vol 3, (1963),
207-227. Pergamon Press, N. Y.
# syntax-directed, compiler, meta-language #
"This paper describes the structure and use of a compiling system in which the translator is independent of the translation rules and hence is independent of both the object and source language. The author first presents the meta-language, then examples of translation performed by the meta-language, and ends with a description of the recognition procedure."
6. 58
Iverson, K. E.
A programming language.
# language #
"The author presents a programming language in detail and then applies the language to such topics as sorting and logical calculus. The book is in textbook format, with exercises at the end of each chapter."

6. 59
Jonas, R. W.
Generalized translation of programming languages,
# translation, language #
"The author describes a general translation language valid for both programming as well as natural languages. He also introduces the notion of semantical grammars."
CR 0050.

6. 60
Kanner, H.
An algebraic translator.
Comm ACM 2, 10 (Oct 1959), 19-22.
# translator #
"The author presents a translator which is similar to that of J. H. Wegstein (Comm ACM, Mar, 1959). A flowchart is included."

6. 61
Kanner, A., Kosinski, P., and Robinson, C. L.
The structure of yet another ALGOL compiler.
Comm ACM 8, 7 (July 1965), 427-438.
# compiler #
"A high-speed top-down method of syntax analysis is described which eliminates source string backup. Block structure and recursion are handled without interpretive methods. Techniques of code generation for expressions are also described."
CR 15194.

6. 62
Katzan, H., Jr.
Batch, conversational, and incremental compilers.
# compilers #

6. 63
1-level storage system.
# storage #
CR 4176.
6. 64  **Klerer, M., and Reinfelds, I.**

compiler, processors

"This volume presents a series of papers on interactive on-line system. It presents the user's point of view, components of interactive systems, automation of applied mathematics, and information on the implementation of interactive systems. It includes some information on the writing of interpreters."

6. 65  **Knuth, D. E.**

compiler

"An excellent work discussing many of the techniques used in the implementation of compilers."

6. 66  **Laning, J. H., and Zierler, N.**
A program for translation of mathematical equations for Whirlwind I.

Engineering Memo. E-364, HIT.

translation

6. 67  **Laurance, N.**
A compiler language for data structures.


compiler, language

"The language described is based on an implementation of the HAD compiler for the Philco 212. Data-structuring abilities of this language are based on the operator definition statements of HAD together with some simple extensions of the syntax."

6. 68  **Ledgard, H. F.**
Ten mini-languages in need of formal definitions.

SIGPLAN 5, 4 & 5 (Apr 1970), 14-37.

language, compilers

6. 69  **Lee, J. A. N.**
The anatomy of a compiler.


compiler, language, syntax

"This book discusses formal definition of syntax, syntactic analysis, various compiler generators and similar subject areas."

CR 14728.
6. 70 Lomet, D. B.
The construction of efficient deterministic language processors.
# translators #
CR 19075.

6. 71 Mancino, O. G., and Cecchi, M. M.
The internal structure of the FORTRAN CEP translator.
Comm ACM 8, 3 (Mar 1965), 149-151.
# translator, compiler #
"A short outline of the CEP computer is given followed by a
description of the internal structure of the translator.
Emphasis is on the compilation of expressions, input/output
lists and subscripted variables."
CR 8243.

6. 72 Maurer, W. D.
Programming.
# programming #

6. 73 Mayoh, R. fi.
Letter to the editor correcting E. T. Irons' A
syntax-directed compiler for ALGOL 60., Comm ACM 4, 1 (Jan
1961), 51-06.
Comm ACM 4, 6 (June 1961), 284.
# syntax-directed, compiler #
"Mayoh writes the editor of some possible corrections that
can be made to Irons' article in a previous issue."

6. 74 McCarthy, J.
A formal description of a subset of ALGOL.
In- Formal Language Description Languages for Computer
Programming, T. B. Steel, Jr., (Ed.), North Holland
Publishing Co, Amsterdam, (1963), 1-12.
# formal #

6. 75 McKeeman, W. M.
An approach to computer language design,
# compiler, language #
CR 13436.
6. 36  McKinnonwood, T. R.
A multi-access implementation of an interpretive text
processing language.
Psoc IFIP Congress (1968), Software I, Booklet B, 28-32.
language
CR 15782.

6. 77  Metcalfe, H. H.
A parametrized compiler based on mechanical linguistics.
Comm ACM 6, 7 (July 1963), 365.
(compiler, syntax-directed #
"(Abstract only). A workshop has developed four
syntax-directed compilers. One of these is discussed at
length.")
CR 6432, 8000.

6. 78  Metcalfe, H. H.
A parametrized compiler based on mechanical linguistics.
125-165. Pergamon Press, N. Y.
translator #
"This paper describes a technique for parameterizing a
compiler in such a way that it can easily be fitted to a new
machine through a translation algorithm. Modern linguistic
theory is used as a basis.")
CR 8432, 8000.

6. 39  Miller, A. E., and Goldman, M.
Organization and program of the BMEW/Checkout data
processor.
83-96.
(processor #
CR 1065.

6. 80  Mock, O. R.
Logical organization of the PACT I compiler.
(compiles #
"The author outlines the step-by-step process of producing a
compiler which translates PACT I into IBM 701 machine code.
Tape is used for storage during the compilation process."

6. 81  Moore, R. D.
An implementation of ALGOL 60 for the PF6000.
(storage allocation, compiler #
CR 7259.
6. 82
Naur, P.
The design of the GIER ALGOL compiler.

Typography, allocation

"This report gives a full description of an ALGOL 60 system for a small machine. Many different aspects of the system are discussed including storage allocation, procedure calls, storage problems within the translator and the methods used in writing the translator."

6. 83
Naur, P.
Program translation viewed as a general data processing problem.
Comm ACM 9, 3 (Mar 1966), 176-179.

Typography, translation

"The paper attempts to obtain a broader viewpoint toward compiler writing rather than considering it as a narrow field of computer science. The author deals with structure, reliability and techniques."

6. 84
Naur, P.
The design of the GIER ALGOL compiler.
BIT 3 (1963), 124-139, 145-166.

Typography, compiler

CR 7904.

6. 85
Noble, A. S., and Talnadge, R. B.
Design of an integrated programming and operating system, I and II.
IBM Systems J 2 (June 1963), 152-179.

Typography, compiler

6. 86
Opler, A.
Requirements for real-time languages.
Comm ACM 9, 3 (Mar 1966), 196-199.

Typography, compiling

The unique requirements of real-time programming are discussed with some attention being paid to special compilation and execution peculiarities."

6. 87
Opler, A., and Gray, M.
Design of a multiprogrammed algebraic compiler (processor).

Typography, compiler

6. 88
Opler, A., Caracciofo, A., and Gorn, S.
Symposium on languages for processor construction.

Typography, processor

CR 7257.
6. 89  
Paul, M.  
ALGOL 60 processors and a processor generator,  
"This paper describes the author's experience with  
processors using pushdown stacks. The general problem of  
formal language translation is also discussed."  
CR 7263.

6. 90  
Perlis, A. J.  
The synthesis of algorithmic systems.  
3 ACM 14, 1 (Jan 1967), 1-9.  
"This book presents a summary of the basic techniques  
necessary for the implementation of compilers. A wide  
variety of subjects is covered including syntax, parsing,  
resource allocation, detection and correction of errors, and  
details of compiler construction."

6. 91  
Pollack, B. W.  
Compiler techniques,  
"This book presents a summary of the basic techniques  
necessary for the implementation of compilers. A wide  
variety of subjects is covered including syntax, parsing,  
resource allocation, detection and correction of errors, and  
details of compiler construction."  

6. 92  
Randell, B., and Russel, L. J.  
ALGOL 60 implementation.  
"Major components of any programming language are identified  
as 1) the elementary statement form, 2) mechanisms for  
linking statements together and 3) mechanisms for data  
input/output. Many examples are given, often from list  
processing languages."

6. 93  
Raphael, B.  
The structure of programming languages,  
"Major components of any programming language are identified  
as 1) the elementary statement form, 2) mechanisms for  
linking statements together and 3) mechanisms for data  
input/output. Many examples are given, often from list  
processing languages."

6. 94  
Ross, D. T.  
AED Jr.: an experimental language processor.  
"Major components of any programming language are identified  
as 1) the elementary statement form, 2) mechanisms for  
linking statements together and 3) mechanisms for data  
input/output. Many examples are given, often from list  
processing languages."

6. 95  
Rutishauser, H.  
Panel on techniques for processor construction.  
Proc IFIP Congress, Munich, (1962), 524-531.  
"Major components of any programming language are identified  
as 1) the elementary statement form, 2) mechanisms for  
linking statements together and 3) mechanisms for data  
input/output. Many examples are given, often from list  
processing languages."
Ryder, K. L.
Note on an ALGOL 60 compiler for PEGASUS I.
Comp J (1963-64), 336-338.
# compiler #
"This note gives a short description of an ALGOL 60 compiler which implements most of ALGOL 60 including recursive facilities. Comparison with the PEGASUS autocode is given along with the effort involved and reasons for writing."
CR 5997.

Samelson, K.
Programming languages and their processing,
# syntax, translator, generator #
"Samelson's article gives an introduction to language structure, pushdown stacks and different forms of processors."
CR 7252.

Sattley, K.
Notes on construction of an ALGOL translator.
# translator #
CR 0143.

Schwartz, J. T., and Cocke, J.
Programming languages and their compilers, preliminary notes.
# languages, compilers #
"A lengthy, extremely good summary of the work done in the field?"

Sheridan, P.
The arithmetic translator-compiler of the IBM Fortran automatic coding system.
Comm ACM 2, 2 (Feb 1959), 9-21.
# translator, compiler, optimization #
"This article is a formal and detailed description of the translation of Fortran formulas into IBM 704 machine language."

Smith, J. W.
JOSS-IT: design philosophy.
# compiler design #
6.102 Steil, A. B.
Using the readily available algebraic language as a compiler environment.
"The author suggests a technique for using algebraic command language in writing compilers when a small special purpose language is to be implemented."

6.103 Sugimoto, M.
PL/1 reducer and direct processor.

6.104 Teichroev, D., and Lubin, J. P.
Computer simulation-discussion of the technique and comparison of languages.
"The purpose of this paper is to present a comparison of some computer simulation languages and some of their implementations."

A language design for concurrent processes.

6.106 Trundle, R. W. L.
LITHP—an ALGOL list-processor.
Comp J 9 (1966), 167-172.
"This paper describes a simple implementation of list processing which can be used on any machine having a suitable ALGOL compiler. The system consists of a special set of declarations."

6.107 Wegner, P.
Programming languages, information structures and machine organization.
"This book discusses machine language, machine organization, assembly techniques, macro systems, lambda calculus, the structure of procedure-oriented languages and the run-time representation of dynamic systems."
Introduction to system programming.  
# compilers #  
"This collection of articles includes two discussions of  
FORTRAN compilers, four of ALGOL compilers, and three of  
various commercial compilers. The topics of these articles  
include translation, optimization and stack techniques."  
CR 0640.

6.109  Wiseman, N. E., and Hiles, J. O.  
A ring structure processor for a small computer.  
Comp J 10 (Feb 1968), 338-346.  
# processor #

6.110  Yershov, A. P.  
ALPHA-'-an automatic programming system of high efficiency,  
# compiler, optimization, translator #  
"This paper describes the implementation of an extended  
ALGOL 60 compiler on the Russian M-20 computer.  
Capabilities are described and details of optimization  
techniques are given."
7. 0

DETAILS OF COMPILER CONSTRUCTION

7. 1
Anderson, J. P.
A note on some compiling algorithms.
Comm ACM 7, 3 (Mar 1964), 149-150.

"Two compiling generators for arithmetic expressions are discussed: one presently used in an experimental compiler and a suggested improvement."
CR 6315.

7. 2
Arden, B. W., Galler, B. A., and Graham, B. M.
An algorithm for translating Boolean expressions.
J ACM 9, 2 (Apr 1962), 222-239.

"This article gives a method for scanning Boolean expressions which 'fits into a general scheme for the translation of statements to machine language.' In this scheme, there is no redundant evaluation of an expression: once evaluation is known to be TRUE, the rest of the expression is skipped."
CR 4061.

7. 3
Baer, J. L., and Bovet, D. P.
Compilation of arithmetic expressions for parallel computations.
Proc IFIP (1968), Booklet B, 4-10.

7. 4
Barnett, M. P.
Indexing and the A-notation.
Comm ACR 6, 12 (Dec 1963), 740-745.

"The author discusses some methods of indexing sequentially stored elements of sparse multi-dimensional arrays in the A-notation. One technique used is dense storage versus a symmetric rectangular array."
CR 5668.

7. 5
Aarnett, M. P.
Low level language subroutines for use within Fortran.

"The author describes subroutines dealing with 'special arithmetic*, symbol manipulation, bit manipulation and visual display. It is his feeling that the use of such subroutines simplifies coding and eases the transition of programs from one computer to another."
CR 2144.
7. 6 Barron, D. W. 
Assemblers and loaders. 
"This short monograph presents a good introduction to the subject. It covers symbol tables, one- and two-pass assemblers, macro-assemblers, and meta-assemblers."
CR 19037.

7. 7 Batson, A. 
The organization of symbol tables. 
Comm ACM 9, 2 (Peb 1965), 111-112.
# symbol tables #
"This article describes techniques used in the Virginia ALGOL 60 compiler for symbol table organization. The primary consideration was making the recognition of identifiers and reserved words as rapid as possible."

7. 8 Bell, J. R. 
The quadratic quotient method: a hash code eliminating secondary clustering. 
# hash-coding #

7. 9 Pemer, R. W. 
Survey of modern programming techniques. 
Comp Rull. (Mar 1961), 127-135
# compiling #

7. 10 Platny, J. 
Symbolical record of time dependent logical relations and a way of their ordering. 
*compilation #.

7. 11 Bloom, B. H. 
Space/time trade-offs in hash-coding with allowable errors. 
Comm ACM 13, 7 (July 1970), 422-426.
# hash-coding #

7. 12 Bobrow, D. G., and Murphy, D. L. 
Structure of a LISP system using two-level store. 
# compiling #

7. 13 Bobrow, D., and Teitelman, W. 
Format-directed list processing in LISP. 
# translators #
7. 14  Bottenbruch, H.
Use of magnetic tape for data storage in the ORACLE-ALGOL translator.
"Because of its small memory size, the ORACLE-ALGOL translator makes use of magnetic tape for array storage during the translation process."

7. 15  Bottenbruch, H. H., and Grau, A. A.
On translation of Boolean expressions.
Comm ACM 5, 7 (July 1962), 384-386.
"This article centers around optimization of Boolean expressions and possible execution during translation of some operations. Several ALGOL examples are given and discussed."

7. 16  Bouman, C. A.
An advanced input-output system for a COBOL compiler.
"RCA created an I/O system called the file control processor to produce object programs in an efficient manner and to help implement the COBOL compiler on their 601 computer. The author describes an interpretive system called the File Control Processor which utilizes the technique of segregation. Some of the objectives of this system were minimum object time memory use, maximum object file speed, and ability to implement all types of batching."
CR 2612,

7. 17  Boyell, R. L.
The method of successive grids for reduction of function storage requirements.
"This article describes the use of grids for reduction of function storage requirements. The coarsest grid is used for storage of the first digit, and each succeeding digit is stored in a seemingly finer grid. The advantage of the grid method is, however, dependent on the size of the function table to be stored."
CR 4543.

7. 18  Boyle, J. M., and Grau, A. A.
An algorithmic semantics for ALGOL 60 identifier denotation.
JACM 17, 2 (Apr 1970), 361-382.
"language, semantics"
7. 19  
Bratman, H.
An alternate form of the UNCOL diagram.
Comm ACM 4, 3 (Mar 1961), 142.
* generator, compiler, translator *
"This is merely a clarification of the UNCOL diagrams appearing in Comm ACM (Aug. 1958), 12-4, and (Sept. 1958), 9-15. They show the transformations made by generators, translators, and compilers."
CR 1042.

7. 20  
Breed, L. M., and Lathwell, R. A.
The implementation of APL/360.
* compiler *

7. 21  
Preuer, M. A.
Generation of optimal code for expressions via factorization.
Comm ACM 12, 6 (June 1969), 333-340.
* compiler, optimization *
"The author presents methods for increasing the efficiency of the object code produced while compiling any given expression. Each expression is broken up into a set of sub-expressions each of which occurs in more than one other expression or sub-expression. These sub-expressions are put in a definite sequence such that computing occurs in correct sequence and storage requirements are reduced. The procedures used are heuristic in nature."

7. 22  
Brigham, R. C., and Bell, C. G.
A translation routine for the DEUCE computer.
Comp J 2 (1959), 76-84.
* translation *
"The authors have developed a mathematically-oriented programming language (SODA); both the language and its translation process are described in this paper."

7. 23  
Brooker, R. A.
A programming package for some general modes of arithmetic.
Comm ACM 7, 2 (Feb 1964), 119-127.
* language, compiler *
"This paper describes an interpretive system for computation with many different types (INTEGER, REAL, etc.) including matrices consisting of these types."
CR 6936.
   Some proposals for the realization of a certain assembly program.  
   *Comp 3* 3 (1960), 220-231.  
   "This paper is essentially a continuation of "An assembly program for a phrase-structure language" with emphasis on implementation."

7. 25  Burge, W. H.  
   Interpretation, stacks and evaluation.  
   In *Introduction to System Programming*, P. Wegner, (Ed),  
   "This paper is concerned with expressions which have a value or which describe things (AE's). The first part of the paper describes a method for evaluation; the second describes AE's which are equivalent to regular expressions and RNF expressions and interprets them in different ways."

7. 26  Burge, W. H.  
   The evaluation, classification and interpretation of expressions.  
   "This paper is concerned with expressions which have a value or which describe things (AE's). The first part of the paper describes a method for evaluation; the second describes AE's which are equivalent to regular expressions and RNF expressions and interprets them in different ways."

7. 27  Cart, J.  
   Recursive subscripting compilers and list-type memories.  
   *Comm ACM* 2, 2 (Feb 1959), 4-6.  
   "Carr develops a powerful method of handling algorithm which modify the contents of lists. He speaks of adding to, deleting from, and examining list structures. Recursion is mentioned as being particularly useful when dealing with lists."

7. 28  Carr, J. W., and Hanson, J. W.  
   Two subroutines for symbol manipulation with an algebraic compiler.  
   *Comm ACM* 4, 2 (Feb 1961), 102-103.  
   "Two subroutines, one for the decomposition of alphabetic words, the other for the combination of single alphabetic characters, make it possible to adapt languages to symbol manipulation work. The subroutines written for the IBM 650 are described."  
   CR 1214,
7. 29 Christiansen, C. 
On the implementation of AMBIT, a language for symbol manipulation. 
Comm ACM 9, 8 (Aug 1966), 570-573. 
"A brief description of the implementation technique of the AMBIT replacement rule is given. An algorithm for the AMBIT SCAN' is given which provides a rationale for the AMBIT language."

7. 30 Cleave, J. P. 
Algorithms for formula translation. 
Comp J 2 (1959), 53-66. 
"Cleave gives two algorithms for formula translation into a three-address code: one for explicit formulas and one for implicit formulas."

7. 31 Cocke, J. 
Global common subexpression elimination. 
"optimization"

7. 32 Coffman, P. G., and Eve, J. 
File structure using hash functions. 
Comm ACM 13, 7 (July 1970), 427-432. 
"hash-coding"

7. 33 Cohen, J. 
A use of fast and slow memories in list processing languages. 
Comm ACM 10, 2 (Feb 1967), 82-86. 
"The author describes a method of increasing the memory space utilization for list-structured data. Memory is divided into pages. Whenever an element of a page not currently in fast store is called, the program selects the least active page and interchanges it with the new page."

7. 34 Conway, M., and Speroni, J. 
Arithmetizing declarations: an application to COBOL. 
Coma ACM 6, 1 (Jan 1963), 24-27. 
"compiler-writing"

7. 35 Cook, D. P. 
Automatic use of random access backing store in ALGOL programs. 
Comp Bull. 11, 4 (Mar 1968), 301-302. 
"storage allocation"
Annotated Bibliography

7.36 Day, A. C.
Pull table quadrature searching for scatter storage.
# hash-coding#

7.37 Day, W. H. E.
Compiler assignment of data items to registers.
# compilation, optimization #
"This paper presents three algorithms for assigning data items to registers. Optimization is discussed."

7.38 Djikstra, E. W.
Solution of a problem in concurrent programing control.
*Comma ACM 8, 9* (Sept 1965), 569.
# compiling #
CR 9023.

7.39 Elson, M., and Rake, S. T.
Code-generation technique for large-language compilers.
# compiler, optimization #
"A technique for generating optimized code is presented, Optimization is both local and global. The program operates on a meta-machine dealing with tree structures which represent the text to be compiled, The approach readily lends itself to extendible languages and the modification of existing languages."

7.40 Ershov, A. P.
On programming of arithmetic operations.
*Comma ACM 1, 8* (Aug 1958), 3-6, and (Sept 1958), 16.
# compiling #
"An arithmetic operation can be described by a three-part general algorithm, Sane possible specific algorithms are discussed. The September article contains the figures which were left out of the August article."

7.41 Evans, A.
*An ALGOL 60 compiler,*
# compiler #
"This paper is a thorough discussion of the internal workings of an ALGOL translator used at Carnegie-Mellon University. The compiler is partly based on Polish postfix notation and the stack concept."
CR 7905.
Evans, A., Perlis, A. J., and VanZoeren, H.
The use of threaded lists in constructing a combined ALGOL and machine-like processor,
Comm ACM 4, 1 (Jan 1961), 36-41.

The authors discuss a method for providing both speed and full use of the machine in one ALGOL translator. Some possible extensions to ALGOL'60 are briefly discussed. The usage of threaded lists is presented as a possible method of having both 'rapid translation' and 'making full use of the machine's properties in the translated code' with a minimum loss of efficiency.

Fabian, V.
A recursive procedure for compiling expressions.
Chiffres 2 (Apr 1963), 275-281.

Floyd, R. W.
An algorithm for coding efficient arithmetic operations.

The article describes a formula translation scheme that 'reduces the number of store and fetch operations, evaluates constant sub-expressions during compilation, and recognizes many equivalent sub-expressions.' The author provides a series of flowcharts along with a detailed explanation of his technique.

Foster, J. M.
Automatic syntactic analysis.

This short monograph presents an excellent overview of the subject of grammars, parsing, and syntactic analysis. The author covers top-down and bottom-up parsing, universal parsing methods, transition matrices, precedence grammars as well as several other important topics.

Galler, R., and Fisher, M. J.
An improved equivalence algorithm.
7, 47  Galler, B. A., and Perlis, A. J.
Compiling matrix operations.

The authors contend that including linear algebra in
algebraic languages is not as difficult as thought, by
developing a translation process for handling matrix
operations. They propose a modification of ALGOL 60 which
would allow matrices and vectors as variables and give many
ALGOL examples.

CR 4638.

3. 48  Gear, G. W.
Optimization of the address -field compilation in the ILLIAC II assembler.
Comp J 6 (Jan 1964), 332.

The author describes a recursive translation process. The
approach used is the "control push-down", which handles the
storage requirements of recursive subroutines used in the
translator. The article includes a section of the
translation matrix actually used in the procedure.

7, 49  Grau, A. A.
Recursive processes and ALGOL translation.

The author describes a recursive translation process. The
approach used is the "control push-down", which handles the
storage requirements of recursive subroutines used in the
translator. The article includes a section of the
translation matrix actually used in the procedure.

7. 50  Gries, D.
The use of transition matrices in compiling.
Tech. Rept. No. CS 57, Computer Science Dept.,
Stanford Univ., Stanford, Calif. (Mar 1967), and
Comm ACH 11, 1 (Jan 1968), 26-34.

The author gives an algorithm for constructing an efficient
left-right recognizer from a suitable RNF grammar. The
algorithm uses a transition matrix and stack. The algorithm
is a practical one and may be used for the construction of compilers
CR 14284, 14508.

7, 51  Gries, D., Paul, M., and Wiehle, H. R.
Some techniques used in the ALCOR-ILLINOIS 7090.
Coma ACM 8, 8 (Aug 1965), 496-500.

The authors describe some of the lesser known but
significant techniques used in implementing the
ALCOR-Illinois 7090 compiler,
CR 8066.
7.52 Hamblin, C. L.  
Translation to and from Polish notation.  
Comp J (Oct 1962).  
# translation #

7.53 Hansen, W. J.  
Compact list representation, definition, garbage collection, and system implementation.  
# list processing #

7.54 Harrison, M. C.  
Data-structures and programming.  
# languages, compilers #

This lengthy work discusses many of the data structures commonly found in the implementation of systems programs, including compilers and interpreters?

7.55 Hawkins, P. N., and Huxtable, D. H. R.  
A multipass translation scheme for ALGOL 60.  
# translator, optimization #

A multi-pass translator produces more efficient code than a one-pass translator; the authors give an in-depth description of the one which they have written for the KDP 9. The main feature of this translation scheme is efficiency in areas such as minimum running time and machine storage requirements, 'The scheme operates in seven distinct phases: 1) input, 2) syntactic check and reduction of the input text to a form suitable for processing by the later phases, 3) procedure classification, 4) storage allocation, 5) index optimization, 6) translation and formula optimization, and 7). final compilation and output'.

7.56 Hempstead, G., and Schwartz, J. T.  
FACT loop expansion,  
# compiler #

This is a discussion of the coding involved in compiling FACT loops.

7.57 Hill, V., Langmaack, H., Schwarz, H. R., and Seegmueller, G.  
Efficient handling of subscripted variables in ALGOL 60 compilers.  
# compiler, allocation #
7. 58 Hoare, C. A. R.
The Elliot ALGOL input/output system.
Comp 3 5, 4 (Jan 1963), 345-348.
# compiler #
"This article describes the method of specifying input and
output of ALGOL programs run on the National-Elliot 803 and
the Elliot 503 computers. The system is 'set up so as to
have a minimum appearance of 'read' and 'print'. One of the
advantages of the system is in its output of data with
alphanumeric description, accomplished with one 'print'
statement - v
CR 4539.

7. 59 Homer, E. D.
An algorithm for selecting and sequencing statements as a
basis for a problem oriented programming system.
# compilers #
"This paper presents the basis for a problem oriented
computer programming system."
CR 11528.

7. 60 Hopgood, P. R. A.
Compiling techniques.
PP.
# compilers #
"This book deals with modern techniques used in the design
and implementation of compilers. It covers data structures,
trees, graphs, arrays, tables, the description of languages,
lexical and syntactic analysis, code generation, storage
allocation and compiler-compilers. It is an excellent
introduction to the field."

7. 61 Hopgood, P. R. A.
A solution to the table overflow problem for hash tables.
Comp Bull. 11 (Mar 1968), 297.
# hashing, resource allocation #

7. 62 Huskey, H. D.
Compiling techniques for algebraic expressions.
# compiling, translation #
CR 1648.
7. 63 Huskey, H. D., and Wattenburg, W. H.
A basic compiler for arithmetic expressions.
# compiler #
"This article describes briefly a technique for compiling
arithmetic expressions. It includes a test program and
appendix, wherein the compiler is given, written as a
Fortran program."

7. 64 Huskey, H. D., and Wattenburg, W. H.
Compiling techniques for Boolean expressions and conditional
statements in ALGOL 60.
Comm ACM 4, 1 (Jan 1961), 70-75.
# compiling #
"This paper gives a method of compiling Boolean expressions
which does not, as is usual, compile an object program that
performs all logical operations ..., but instead compiles a
program which tests for only a minimum of logical
expressions. The techniques are presented in several ALGOL
60 routines with accompanying commentary."

7. 65 Ingerman, P. Z.
Thunks.
Comm ACM 4, 1 (Jan 1961), 06-58.
# compiling #
"This article is concerned with efficient compilation of
Procedures. A thunk is the coding produced by the
translator associated with a variable which provides its
address; one is used for each parameter in each procedure
statement,"

7. 66 Ingerman, P. Z.
Dynamic declarations.
Comm ACM 4, 1 (Jan 1961), 59.
# mapping #
"This is a short paper describing a technique for mapping
one array into another."

7. 67 Ingerman, P. Z.
A new algorithm for algebraic translation.
Proc ACM 14th Nat'l Conf. (1959), 22.
# t-translation #

7. 68 Ingerman, P. Z.
Techniques for processor construction.
# processor #
7.69  Irons, E. T., and Feurzeig, W.  
Comments on the implementation of recursive procedures and blocks in ALGOL 60.  
# compiling, recursion mechanisms  
"This paper covers the problem of procedure entries and exits and the determination of recursion in a procedure.  
Several diagrams with explanatory notes help explain the processes for handling the problem."

7.70  Irwin, L.  
Implementing phrase-structure productions in PL/1.  
# phrase-structure  
"A simple technique is described for implementing productions of a context-free phrase-structure grammar in PL/1."

7.71  Jensen, J.  
Generation of machine code in ALGOL compilers.  
BIT S (1965), 235-245.  
# compiling  

7.72  Jensen, J., and Maur, P.  
An implementation of ALGOL 60 procedures.  
BIT 1, 1 (Jan 1961), 38-47.  
# compiler  
"This article describes a method of implementing ALGOL 60 procedures. One technique used is to represent each parameter by a subroutine. The link between the procedure body and the call information is formed by a fixed administratioe subroutine which is called in every tire an entry into a procedure is made."

7.73  Johnsen, R. L., Jr.  
Implementation of WEILAC for the IBM 704 and IBM 709 computers.  
# compiler  

7.74  Rain, R. Y.  
Block structures, indirect addressing and garbage collection.  
Comm ACM 12, 7 (July 1969), 395-398.  
# compiling  

7.75  Karp, R. M., and Miller, R. E.  
Properties of a model for parallel computations: determinacy, termination, queening.  
SIAM J (Nov 1966), 1340-1411.  
# compiling
3. 76 Keese, W. M., Jr., and Huskey, R. D.
An algorithm for the translation of ALGOL statements.
(Preprints).
$\#$ translation, compiling $\#$
CR 3587.

7. 77 Kelley, J. E., Jr.
Techniques for storage allocation algorithms,
Comm ACM 4, 10 (Oct 1961), 449-454.
$\#$ allocation $\#
"This article presents a few helpful techniques for
approaching allocation problems. Among the methods
discussed are dynamic programming and heuristic methods.
The article itself is valuable in that it is general and
that the techniques presented can be universally applied."$ CR 2749.

7. 78 Klerer, M.
Automatic dimensioning,
Comm ACM 10, 3 (Mar 1967), 165-166.
$\#$ compiling $\#

7. 79 Knight, K. R.
An ALGOL construction for procedures as parameters of
procedures,
Comm ACM 13, 4 (Apr 1970), 266.
$\#$ compiler implementation $\#

7. 80 Knuth, D. E.
The art of computer programming, Vol 1, Vol 2.
$\#$ compilers $\#
"An excellent work discussing many of the techniques used in
the-implementation of compilers."$ CR 6677.

7. 81 Landin, P. J.
The mechanical evaluation of expressions.
Comp J 6 (1963), 308.
$\#$ compiling $\#
"Landin is concerned with the structural simplification of
expressions,"$ CR 6677.

7. 82 Lauer, P.
Formal definition of ALGOL 60.
Tech. Rept. No. TR 25.088, TBM Labs., Vienna, Austria (Dec
1968).
$\#$ syntax, semantics $\#$
7. 83 Learner, A., and Lim, A. L.
A note on transforming context-free grammars to Wirth-Weber precedence form.
Comp J 13, 2 (May 1970), 142-144.
# context-free grammar #
"A technique is presented which will convert every CF grammar into an equivalent Wirth-Weber simple precedence grammar."

7. 84 Lietzke, M. P.
A method of syntax checking ALGOL 60.
# syntax #
"A syntax checker designed around ALGOL 60 is discussed. The checker is a set of mutually recursive processors tied together by bookkeeping subroutines. # #"
CR 6662.

7. 85 Lucas, P.
The structure of formula-translators.
IBM Laboratories, Vienna, Austria.
# formula translation #

7. 86 Madnick, S. E.
String processing techniques.
Comm ACM 10, 7 (July 1967), 420-424.
# storage allocation #

7. 87 Manelowitz, R.
ANCHOR-an algorithm for analysis of algebraic and logical expressions.
# compiler #

7. 88 Martin, D. P.
Boolean matrix methods for the detection of simple precedence grammars.
Comm ACM 11, 10 (Oct 1968), 685-687.
# grammars #
"The author describes a technique for computing the precedence relations of a context-free language using Boolean matrices. It translates the definitions of precedence into the representation of relations by Boolean matrices."
CR 0159.
7. 89  Maurer, W. D.
An improved hash-code for scatter storage.
"This is perhaps one of the best articles in existence on
hash-coding."

7, 90  Miller, L., Minker, J., Reed," W. G., and Shindle, W. E.
An multi-level file structure for information processing-
"This is a short article on the theory behind PACT."

7, 91  Miller, R. C., and Oldfield, B. G.
Producing computer instructions for the PACT I compiler,
"This is one of the best articles giving an introduction to
the techniques of hash-coding?

7, 92  Morris, R.
Scatter storage techniques.
Comm ACM 11, 1 (Jan 1968), 38-44.
"This is one of the best articles giving an introduction to
the techniques of hash-coding?

7, 93  Nakata, I.
A note on compiling algorithms for arithmetic expressions.
"The author describes a compiling algorithm which minimizes
the frequency of storing and recovering intermediate
results."

7, 94  Nather, R. P.
On the compilation of subscripted variables.
"This article discusses the compiler for the REX-4000 with
emphasis on the utilization of a complete evaluation of the
storage mapping function. By this method subscripted
variables were augmented so that their properties included
1) any number of dimensions, 2) they could be written as
arithmetic expressions, 3) other qualities listed by the
author at the end of the article."

3. 95  Naur, P.
Checking of operands in ALGOL compilers.
RIT 4 (1965), 151-163.
7. 96 Pacelli, A.
Tecniche di tradazione automatica.
(Compiling techniques.)
Atti del convegno sui linguaggi simbolici di programmazione,

7. 97 Petroni, L., and Vandoni, C. E.
Integer and signed constants in ALGOL.
Comm ACM 7, 12 (Dec 1964), 7, 1234-435.
Meta-language, syntax, semantics #
"The authors remark on the relationship between syntax and
semantics. The ALGOL 60 definition is criticized for being
divorced from its semantics."

7. 98 Pollack, B. W.
Compiler techniques,
"This book presents a summary of the basic techniques
necessary for the implementation of compilers. A wide
variety of subjects is covered including syntax, parsing,
resource allocation, detection and correction of errors, and
details of compiler construction."

7. 99 Randell, B., and Russel, L. J.
Single-scan techniques for the translation of arithmetic
expressions in ALGOL 60.
3 ACR 11, 2 (Apr 1964), 159-167.
"This paper concerns the use of a stack to store general
expressions in reverse Polish form. By changing the
procedure slightly type information is included in the
stack. Also some computation may be made at compile time."
CR 6.303.

7.100 Rotenberg, N., and Opler, A.
Variable width stacks.
Comm ACM 6, 10 (Oct 1963), 608-610.
"This article discusses variable width stacks to be used in
a compiler for a character addressable variable field
computer. With this type of computer substitution or
expansion of language elements is unnecessary. As an
illustration, a variable width stack and two character
stacks can scan algebraic expressions."
CR 5358.
7.101 Rntishauser, H.
Panel on techniques for processor construction.
Proc IFIP Congress, Munich, (1962), 524-531.
# compiler, translator #
"Various panel members discuss different aspects of compiler construction and describe some of the problems encountered by the compiler writer."

7.102 Ryan, J. T.
A direction-independent algorithm for determining the forward and backward compute point for a term or subscript during compilation.
# compilation #
"This paper describes an algorithm which determines the earliest and latest times when a subscript can be computed."

7.103 Sable, J. D.
Use of semantic structure in information systems.
Coma ACR 5, 1 (Jan 1962), 40-42.
# semantic analysis #
"This paper describes semi-automatic techniques applied to semantic analysis and how semantic structure, once determined, can be effectively used in information retrieval systems. The author diagrams the semantic structure of a vocabulary via three matrices: scope, reduced, and basis."

7.104 Samanskii, V. E., and Ellanskaja, L. V.
General scheme of the methods of block iteration.
Vycisl. Bat, (Kiev) Vyp., No. 1, (1965), 41-52. (Russian),
# compiling #
CR 15437.

7.105 Samelson, K., and Bauer, F. L.
Sequential formula translation.
Comb ACM 3, 2 (Feb 1960), 76-83.
# translator #
"A brief history of sequential formula translation is given and the specific elements of translation, including the evaluation of arithmetic expressions, are discussed. The last-in-first-out principle is presented."
CR 0219.

7.106 Samet, P. A.
The efficient administration of blocks in ALGOL.
Comp J 8 (1965), 21-23.
# compiler #
"A scheme for administration of ALGOL blocks is proposed, based on the use of block numbers rather than levels. It is claimed that this method simplifies the organization of procedure calls, including recursive calls."
7.103 Schmidt, L.
Implementation of a symbol manipulation for heuristic translation,
# translator #

7.108 Schorr, H.
Compiler writing techniques and problems.
# compiler, language, translator #

7.109 Schwarzenberger, P.
Syntax-oriented algorithms for personal data files.
# syntax-oriented #
CR 14488.

7.110 Sethi, R., and Ullman, J. D.
The generation of optimal code for arithmetic expressions.
# optimization, resource allocation #

7.111 Sheridan, P.
The arithmetic translator-compiler of the IBM Fortran automatic coding system.
Comm ACM 2, 2 (Feb 1959), 9-21.
# translator, compiler, optimization #
"This article is a formal and detailed description of the translation of Fortran formulas into IBM 784 machine language."

7.112 Standish, T. A.
A data definition facility for programming languages.
Computer Science Rept., Carnegie Institute of Tech.,
Pittsburgh, Pa., (May 1967).
# language #
"This dissertation describes a descriptive notation fat data structure which is embedded in a programming language in such a way that the resulting language behaves as a synthetic tool."
7.113 Stone, H. S.
One-pass compilation of arithmetic expressions for parallel processors.
# compilation, processor, parallel #
*This article describes a one-pass algorithm for the compilation of expressions such that the resulting expression structure is inherently parallel. This approach may increase compute speed?
CR 12741.

7.114 Swift, C. J.
Compiling connectives.
Comm ACM 3, 6 (June 1960), 345-346.
# compiling #
"The author describes the handling of the connectives 'and' and 'or' in the FACT language."
CF 0216.

7.115 SHARE Ad-Hoc Committee on Universal Languages.
The problem of programming communication with changing machines: a proposed solution, Part 1, Part 2.
# compilers #
"The authors suggest that a three-level concept of machine languages, problem oriented languages, and UNCOL (a universal computer oriented language). Generators would take any POL to UNCOL, and translators would change UNCOL to a specific machine language."

7.116 Thorlin, J. F.
Code generation for PIE (parallel instruction execution).
# code generation #

7.117 Watt, J. M.
The realization of ALGOL procedures and designational expressions.
Comp J 5, 4 (Jan 1963), 332-337.
# allocation, compiler #
"This paper describes methods for compiling recursive procedures and designational expressions in ALGOL 60. Storage allocation at run time and a method for organizing procedure linkage are discussed."
CR 4535.
7.118 Wegner, P.
Communication between independently translated blocks,
Comm ACM 5, 7 (July 1962), 376-381.
# languages, storage allocation #
This article is about communication between blocks in a common
intermediate language. Also discussed to a lesser degree is the problem of dynamic
storage allocation for fixed and variable length blocks.

7.119 Wegner, P.
An introduction to stack compilation techniques.
In Introduction to System Programming, P. Wegner, (Ed),
# compilation #

7.120 Wegstein, 3. H.
From formulas to computer oriented language.
Comm ACM 2, 3 (Mar 1959), 6-8.
# translation #
"This paper is concerned with the part of a compiler which
translates algebraic formulas into computer code, and it
describes a rather complex technique for breaking a formula
down into a sequence of sub-formulas. A flowchart is
included."

7.121 West, V. D.
On the compilation of arithmetic expressions.
(Letter).
# compilation #

7.122 Williams, P. A.
Handling identifiers as internal symbols in language
processors.
Comm ACM 2, 6 (June 1959), 21-24.
# hash-coding #
"This article presents a technique for hash-coding symbols."

7.123 Wolpe, H.
Algorithm for analyzing logical statements to produce a
truth function table.
Comm AC?! 1, 3 (Jan 1958), 4-13.
# compiling #
"This article describes a method of producing code
corresponding to a truth table based on a series of logical
conditions,"
8. 0 ADDITIONAL TOPICS

8. 1 Bahr, K.
FORMAC-FORTRAN preprocessor.
In Symbolic Mathematical Computation, 1,3 (Oct 1969), 34-47.
# pre-processor #

8. 2 Bennett, R. K., and Kvilekval, A.
SET, self extending translator.
# extendible, translator #

8. 3 Bennett, R. K., and Neumann, D. H.
Extension of existing compilers by sophisticated use of macros.
Comm ACM 7, 9 (Sept 1964), 541-542.
# extendible, macro-processor #

8. 4 Bobrow, D. G., and Weizenbaum, J.
List processing and extension of the language facility by embedding.
# processor, compiler, languages #

8. 5 Bolas, B. J.
Optimization problems in extensible compilers.
SIGPLAN S, 7 (July 1970), 127.
(abstract),
# optimization, compiler, extensible compilers #

8. 6 Book, E., and Bratman, H.
Using compilers to build compilers.
# compiler-compiler #

The CWIC/360 system, a compiler for writing and implementing compilers.
SIGPLAN 5, 6 (June 1970), 11-29.
# compiler-coapiler #
8. 8 Brooker, R. A. et. al.
The compiler-compiler.
Annual Review in Automatic Programing, Vol 3, (1963),
229-275. Pergamon Press, N. Y.

"This paper is a detailed specification of a system for
describing the form and meaning of the statements in a
phrase-structure language. The system operates in two
phases: 1) accepting and recording the definition of the
phrase-structure language, and 2) translating a source
program written in that language. A compiler given this
system can generate a compiler for an arbitrary
phrase-structure language."

8. 9 Brooker, R. A., and Rohl, J. S.
Simply partitioned data structures: the compiler-compiler
reexamined.
In Machine Intelligence I.

"The authors consider some of the problems that arise with
more complex types of data structures. The discussion
relates to a simple language model with nested block
structure and concentrates on two types of statements: data
declarations and assignments."

8. 10 Brooker, R. A., Morris, D., and Rohl, J. S.
Compiler compiler facilities in ATLAS autocode.
Comp J 9 (1967), 350-353.

"This paper describes how the phrase-structure facilities of
the compiler-compiler have been added to ATLAS Autocode."

8. 11 Brooker, R. A., Morris, D., and Rohl, J. S.
Experience with the compiler-compiler.

"The authors describe their experience with the
compiler-compiler, a compiler to facilitate the writing of
compilers. The articles describes the following: phrases
and formats, a parsing algorithm, phrase routines, and
format routines."

8. 12 Brown, P. J.
Using a macro-processor to aid software implementation.
Comp 3 12 (Nov 1969), 327-331.

# macro-processor #
8. 13  Brown, P. J.
Macro processors and their use in implementing software.
# macro processor

8. 14  Brown, P. J.
The ML/I macro processor.
# macro-processor
"The author describes a general purpose macro processor.
It's primary use is to allow any existing programming
language to be extended to suit a particular user's
requirements.*
CR 13442.

8. 15  Brown, P. J.
A survey of macro-processors.
37-88. Pergamon Press, R. Y.
# m&a-language

8. 16  Burkhardt, W. H.
Mentalanguage systems and extensible programming.
Proc. of 3rd Hawaii Int'l Conference on System Sciences,
Honolulu, (Jan 1970), 898-901.
# meta-language

8. 17  Burkhardt, W. H.
Metalanguage and syntax specification,
Coma ACM 8, 5 (May 1965), 304-305.
# syntax, meta-language
"Two meta-languages are described. One is adequate for
ALGOL, the other has additional meta-operators which allow
definition of BASIC Fortran."

8. 18  Cheatham, T. E.
The introduction of definitional facilities into higher
level programming languages.
# compiler
"This paper describes a method for implementing definitional
(macro) facilities into a high level language. The paper
concentrates on the compiler mechanisms necessary rather
than syntax specifications."
8. 19  
Cheatham, T. E.  
The TGS-II translator-generator system.  
Proc IPIP Congress, N. Y., (1965), 592-593.  
# translator, compiler-compiler #  
"This paper describes a system which is intended to be a  
general purpose compiling system which is efficient, general,  
and allows efficient implementation and documentation of  
modifications made to the language."

8. 20  
Cheatham, T. E.  
Preliminary description of the translator generator system,  
II.  
(1964).  
# translator, compiler-compiler #

8. 21  
Cheatham, T. I?::, Jr., and Standish, T. A.  
Optimization aspects of compiler-compilers.  
SIGPLAN 5, 10 (Oct 1970), 10-17.  
# compiler-compiler, optimization #

8. 22  
Computer Associates Compiler generator systems  
prograr descriptions,  
No. CA-63-4-50, (July 1563).  
(unavailable).  
# compiler-compiler #

8. 23  
Coulouris, G. F.  
The compiler processor project,  
# compiler-compiler #

8. 24  
Dove, R. K.  
Design highlights of CABAL--a compiler-compiler.  
# compiler,compiler #  
"The author describes the main features of CABAL designed  
for compiling language translators,  
CABAL includes  
co-routines for syntax processing, reductions for syntax  
parsing, full algebraic power for semantics processing,  
structures and operators for string manipulations, etc."  

8. 25  
Eastwood, D. P., and McIlroy, M. D.  
Macro compiler modification of SAP.  
Memorial Computing Lab., Bell Telephone Labs., Murray Hill,  
# compiler, macro-compiler #
8.26  Feldman, Z. A., and Curry, J.
The compiler-compiler in a time-sharing environment.
Lecture notes on Advanced Computer Organization.
Univ. of Michigan, Ann Arbor, Mich., (1967).
# compiler-compiler #

8.27  Feldman, J., and Cries, D.
Translator Writing systems.
Comm ACM 11, 2 (Feb 1968), 77-113.
# compiler-compiler, translator, syntax, semantics #
"This paper surveys critically the research efforts put into
automating compiler writing. The paper includes the formal
study of syntax and its application to translator writing,
varying approaches to automating semantic aspects of
translator writing and other related topics such as the
formal study of semantics, etc."
CR 14729.

8.28  Perentxy, E. M., and Gabura, J. R.
Syntactic directed processor writing system.
# syntax-directed, processor #
"The authors describe a processor writing system--MPL/I.
The processor produced by MPL/I is a PL/I program plus
syntax tables. The translator includes a driving channel
making use of a parsing method developed by B. Domolki."

8.29  Ferguson, D. E.
Evolution of the meta-assembly program.
Comm ACM 9, 3 (Mar 1966), 190-196.
# meta-assembler #
"A generalized assembler is described. How a meta-assembler
is made to function as an assembler is also described. The
paper concludes with a discussion of the implications for
compilers."

a.30  Foster, J. M., and Elcock, E. W.
Absys 1: an incremental compiler for assertions; an
introduction.
American Elsevier Publishing Co., N.Y.
# incremental compiler #
"The authors present some of the many features of the
language implemented with Absys 1, an on-line incremental
compiler,"
8. 31 Pu jino, K.
 Compiler generating system.
 Info. Processing in Japan, 7 (July 1967), 22-34.
 # compiler, generator #
 "The author presents a basic investigation and a method of realizing a compiler generating system. The aim of CGS is to develop a simple method requiring shorter time to make a compiler system with greater capabilities."

8. 32 Gardner, R. I.
 Development of a meta-compiler containing list-processing capabilities.
 Univ. of Calif., Los Angeles, AD-681451, (Dec 1968), 19 pp.
 # meta-compiler #

8. 33 Garwick, J. V.
 GARGOYLE, a language for compiler writing.
 Comm ACM 7, 1 (Jan 1964), 16.
 # compiler, language #
 "This paper describes a language for writing compilers which is syntax directed but which attempts to retain the advantages of assembly language."
 CR 5675.

8. 34 Gilbert, P., and McLellan, W. A.
 Compiler generation using formal specification of procedure-oriented machine languages.
 # formal, language #
 "The authors describe a compiler generation system which is rigorously based and which allows formal specification of both source language and machine language."
 CR 0016.

8. 35 Glennie, A. E.
 On the syntax machine and the construction of a universal compiler.
 # syntax, compiler #

8. 36 Green, J.
 Symposium on languages for processor construction.
 # processor #

8. 37 Gries, D.
 Internal notes on the compiler writing system.
 # compiler-compiler #
8. 38  Gries, D.
CIL: compiler implementation language.
SLAC Rept. No. 102, (Mar 1969), Stanford Linear Accelerator
Center, Stanford Univ., Stanford, Calif.
(Map 1969).
# compiler-coapiler #

8. 39  Halpern, M.
Toward a general processor for programming languages.
# language, translator, processor, metacompiler #
"The author states that any programming language is best
described as a body of macro instructions, and that macro
call constitutes a canonical form which describes
programming notations. A general processor that would
translate a number of languages is more economical than
building new compilers. The author also gives a program in
ALTBXT implemented by means of the prototype processor
described in the article."
CR 14053.

8. 40  Halpern, M.
XPOP: a meta-language without metaphysics,
# meta-language #
"The XPOP language is a compiler writing system which
consists of two parts, a skeleton compiler and a battery of
pseudo-operations for specifying the notation and compiling
peculiarities of the particular language. XPOP is
relatively unrestricted in type and properties of the
languages it can accept."
CR 7266.

8. 41  Hoare, C. A. R.
A programming language for processor construction.
Notes from NATO summer school lectures, (1966).
# processor, compiler #

8. 42  Holt, A. W.
General purpose programming systems.
# compiler #
"Holt describes an approach to a general-purpose compiler
which would be more truly 'general purpose' than existing
systems under that name."

8. 43  Huskey, H. D.
Languages for aiding compiler writing (panel discussion).
Proc Rome Symposium on Symbolic Languages in Data
# compilers #
ANNOTATED BIBLIOGRAPHY

8. 44  Irons, E. T.
Experience with an extensible language.
# extensible compiler #

Techniques and advantages of using the formal compiler
writing system FSL to implement a Formula ALGOL compiler.
# compiler-compiler #
"The FSL compiler writing system is described. The utility
and details of implementing a powerful ALGOL compiler are
discussed."

8. 46  Kay, A. C.
FLEX--a flexible extendable language.
Tech. Rept. 4-7, Univ. of Utah, Salt. Lake City, Utah, (June
1968).
# extendible #

8. 47  Kerr, R. H., and Clegg, J.
The Atlas ALGOL compiler--an ICT implementation of ALGOL
using the Brooker-Morris syntax directed compiler.
Comp J (1967).
# compiler #

8. 48  Knowlton, P. H.
The use of an algebraic language as both a source and target
language.
# compiler, language #
"This approach to compiler writing is very useful when only
minor extensions of a given language are to be made, Effort
and time to be spent on compiler writing are minimized. The
language L SIX is described."
CR 15768.

8. 49  Leavenworth, B. M.
Syntax macros and extended translation,
Comm ACM 9, 11 (Nov 1966), 790-793.
# syntax, translation #
"A translation approach is described which allows one to
extend the syntax and semantics of a given high-level base
language through the use of a new formalism called a
'syntax-macro'. Two types are discussed and examples are
given."

8. 50  Leonard, G., and Goodroe, J.
flora extendible machines.
Comm ACM 9, 3 (Mar 1966), 143-148.
# extendible, translators #
8. 51  Leroy, R.
A macro-generator for ALGOL.

8. 52  Liu, C. D., Chang, G. D., and Marks, R. E.
The design and implementation of a table driven compiler system,

8. 53  Lombardi, L. A.
Incremental computation,
Academic Press, N. Y.

8. 54  Marrs, D. L., Collins, C. W., and Hess, H. A.
A user-oriented macro-processor.
Proc ACM 23rd Nat'l Conf. (1968), 751-763.

8. 55  Maurer, W. D.
The compiled macro assembler.

8. 56  McIlroy, M. D.
Macro instruction extension of compiler language.
8. 57    McKeeman, W. M., Horning, J. J., and Wortman, D. B.
A compiler generator,
  # compiler, compiler-compiler #
"This book presents both an overview of the syntax-directed
precedence language approach to compiler writing and the
specific example of the XPL compiler which was developed at
Stanford University using this method."

8. 59    McKeeman, W. M., Nelson, E. C., and Wortman, D. B.
The XPL compiler generator system.
Proc AFIPS 1968 JCC, Vol 33, 617-635.
  # compiler-compiler, compiler generator #
"This paper describes the XPL language and a set of program
which constitute a translator writing system. XPL is
described by comparison with PL/1. The XPL language is
deliberately restricted to simple features which are useful
in writing translators."

8. 59    Mondschein, L.
VITAL compiler-compiler reference manual.
TN 191967-1, Lincoln Labs., MIT, Lexington, Mass. (Jan
1967).
  # compiler-compiler #

8. 60    Moore, S. A.
Data processing with the compiler compiler.
  # compiler-compiler, data structure #
"The author describes the use of a compiler-compiler for
data processing applications?
CR 0043.

8. 61    Morris, D., and Wilson, I.
A system program generator.
Computer Science Dept., Univ. of Manchester, Manchester,
  # generator, processor #

8. 62    Napper, R. B. E.
Some proposals for SNAP, a language with formal macro
facilities.
  # macroprocessor, language #
"The author describes proposals to apply full power of the
compiler-compiler to provide an extendable language system
which is one-pass and produces efficient object code." 
CR 15203.
8. Rapper, R. B. E.
An introduction to the compiler compiler.
# compiler-compiler #

8. Newley, M. C.
An efficient system for user extendible languages.
# extendible language #
"The author discusses the revisions that should be made for
introducing additional data types and data manipulations in
order to make a language more flexible. The author
describes a compiler extension scheme using ALGOL as an
element."

8. Nolan, M.
MPL: macro pre-compiling language.
# pre-compiler, macros #

8. Rorthcote, R. S.
The structure and use of a compiler-compiler system.
# compiler-compiler #
"This paper describes a compiler-capiler system in which
compilers produced by the system generate object codes in an
intermediate language form. The code along with appropriate
tables can be used as input to a computer without extensive
software."

8. O'Neill, J. T.
META PI--an on-line interactive compiler-compiler.
# compiler-compiler #
"The author describes a translator system implemented on the
RCA Spectra 70. The syntax analysis is top down. The code
is generated by a set of symbol manipulation routines
embedded in the syntax specifications."

8. Opler, A.
'TOOL'--a processor construction language.
# compiler #
"This is an extremely brief description of a Honeywell
system for compiler construction. Some features of TOOL
are: 1) separation of format definitions from procedural
statements, 2) provision for an open set of user defined
terms, and 3) provision for temporary or permanent
replacement of any system element. A short discussion
follows the article."
8. 69 Oppenheim, D. K., and Haggerty, D. P.  
META 5: a tool to manipulate strings of data.  
"This paper describes the META 5 programming system. The system is based on D. V. Schorr's META If system, META 5 is intended to be useful for any problem in which extensive and complex string manipulation is necessary. The system is not solely a compiler-writing system."
CR 11203,

8. 70 Pankhurst, R. J.  
GULP: a compiler-compiler for verbal and graphic languages.  
Proc ACM 23rd Nat'l Conf. (1968), 405-421.  
"A compact compiler-compiler system is described for computer-aided design. Effort is made to reduce storage requirements even at the expense of processing time. Definition and description of user language, syntax language, and semantic language are given. Several extensions of the system are suggested in conclusion."
CR 0147.

8. 71 Pearson, R. W.  
Genesis: a compiler-compiler.  

8. 72 Peccoud, M., Griffiths, M., and Peltier, M.  
Incremental interactive compilation.  
Proc IFIP Congress (1968), Software I, Booklet B, 33-37.

8. 73 Pollack, B. W.  
"This book presents a summary of the basic techniques necessary for the implementation of compilers. A wide variety of subjects is covered including syntax, parsing, resource allocation, detection and correction of errors, and details of compiler construction."
8. 74 Pratt, T. W., and Lindsay, R. K.
A processor-building system for experimental programming language.
"This paper describes an extension of the notation of a translator building system to that of a processor-building system. An operating example of one such system is described."

8. 75 Reynolds, J. C.
An introduction to the COGENT programming system.
*Proc ACM 20th Nat'l Conf. (1965)*, 422-436.
"COGENT is a list-processing compiler-compiler. It provides full generality with regard to input language and target language; the syntax and translation rules must be input to the system. COGENT may be used directly for algebraic manipulation, theorem proving and heuristic programming in addition to its function as a compiler-compiler."

8. 76 Reynolds, J. C.
COGENT programming manual.

8. 77 Richards, M.
BCPL: a tool for compiler writing and system programming.

8. 78 Rishel, W. Z.
Incremental compilers.

8. 79 Rose, G. P.
An extension of ALGOL-like Languages.
*Comm ACM 7*, 2 (Feb 1964), 52-61.
"This paper describes a method which can be used to extend the definition of ALGOL-like languages to include certain sets which are undefinable in BNF."
8. 80 Rosen, S.
A compiler-building system developed by Brooker and Harris.
Comm ACM 7, 7 (July 1964), 403-414.
# compiler-compiler #
"This paper gives a complete description of Brooker and
Morris' compiler building system."
CR 6667.

8. 81 Sandewall, E. J.
LIISP A: a LISP-like system for incremental computing.
# incremental computing #

8. 82 Schneider, F. W., and Johnson, G. D.
META-3: a syntax-directed compiler writing compiler to
generate efficient code.
Proc ACM 19th Nat'l Conf. (1964), D1.5-1.
# syntax-directed, metacompiler #
"A complete description of both the META III compiler and of
the compilation algorithm are given, BETA III is based on
META II which was developed by D. V. Schorre. The basic
compilation method is a recursive top-down scan with out
backtrack."
CR 6944.

8. 83 Schneider, V.
Some syntactic methods for specifying extendible programing
languages,
# syntax, extendible languages #

8. 84 Schneider, V.
A system for designing fast programming language
translators.
#-translator, optimization #

8. 85 Schorre, D. V.
META-II: a syntax oriented compiler writing language,
Proc ACM 19th Nat'l Conf. (1964), D1.3.
# syntax-directed, metacompiler #
"META II is a working compiler writing language which
consists of syntax equations resembling BNP and into which
are inserted instructions to output assembly language
commands. The paper describes both how the compiler system
was written and how it is used."
CR 6943.
8. 86 Schorre, V. A.
A syntax directed SMALGOL for the 1401.:
Comm ACM 6, 7 (July 1963), 365.
# syntax-directed #
"(Abstract only). A syntax-directed compiler is proposed
which would save space during compilation and could be
implemented on a small machine."

8. 87 Sibley, R. A.
The SLANG-system.
Comm ACM 4, 1 (Jan 1961), 75-84.
# compiler, translator #
"The SLANG system is being developed to facilitate the
application of automatic programming methods to compiler
writing. The object is to produce a system that, given
statements describing the compilation process for a
problem-oriented language and statements describing the
computer to be used, will output a compiler to do the job
and documentation for it."

8. 88 Snyder, R. E., and Pleischner, G. M.
The Vectran precompiler.
General Dynamics Dynamics/Astronautics, San Diego, Calif.,
Rept. No. GDA-ERR-AN-076, CPSTI Rept. No. AD-681 786, (Aug
# pre-compiler #

8. 89 Squires, B. E.
Compiler systems and meta-languages.
Rept. No. 208, Computer Science Dept., Univ. of Illinois,
# meta-languages #

8. 90 Strachey, C.
A general purpose macro-generator.
camp J 8, 3 (Oct 1965), 225-241.
# macro processor #

8. 91 Thompson, F. B., Lockemann, P. C., Dostert, B., and
Deverill, R. S.
REL: a rapidly extensible language system.
# extensible language #
8.92 Trout, R. G.
A compiler-compiler system.
"The author discusses compiler-compiler systems and describes a program which has been implemented on an Elliott 503 computer. This program was used for experimental construction of a Fortran compiler."
CR 13163.

8.93 Vandermey, J. E., Varney, R. C., and Patchen, R. E.
Symple--a general syntax-directed macro preprocessor.
"This paper describes a macro-processor that is independent of any particular assembly language. This processor is basically a string manipulator. Its output is presented to some compiler or assembler?"

8.94 Waite, W.
A language independent macro processor.
"This article is a description of an experiment which involved developing a list processing compiler through bootstrapping techniques. The article also discusses some problems of adapting the system to a new machine."
CR 7913.

8.95 Wilkes, M. V.
An experiment with a self-compiling compiler for a simple list-processing language.
"This article is a description of an experiment which involved developing a list processing compiler through bootstrapping techniques. The article also discusses some problems of adapting the system to a new machine."
9. 0  MISCELLANEOUS RELATED REFERENCES

9. 1  Abraham, P. W.
The LISP 2 programing language and system.
# compiler, language #
"The LISP 2 language is described along with the system in
which it is embedded. The system allows extension of itself
and includes a group of routines far implementing the system
on a new machine."
CR 0805.

9. 2  Anonymous,
Criteria to be applied in the standardization of a
programming language.
Comp Bull. 12, 2 (June 1968), 54-06.
# languages #

9. 3  Arden, B. W., Galler, B. A., and Graham, R. M.
Michigan algorithmic decoder.
# compiler #

9. 4  Arden, B., and Graham, R.
On CAT and the construction of translators.
Comm ACM 2, 7 (July 1959), 24-26.
# translator #
"The authors briefly describe the Generalized Algebraic
Translator's characteristics. The article provides an
algorithm to transform a statement to machine instructions
which may be of interest to a person writing a translator."

9. 5  Arden, B., and Graham, R. M.
Generalized algebraic translator.
Computation Center, University of Michigan, Ann Arbor,
(1959).
# translator #

9. 6  Arnold, R. F.
A compiler capable of learning.
# compiler, translator #
"The author has developed an alternative to interpretive
programming and ordinary compilers: a compiler which uses a
random program generator to produce an object program of the
same length as the source program and checks for a match by
interpretive routines."
9. 7 Backus, J. W. et. al.
The FORTRAN automatic coding system.
# language, compiler #
"The original FORTRAN compiler is described
section-by-section by its authors. A brief summary of
FORTRAN is also presented,"

9. 8 Ralzer, R. M., and Farber, D. J.
APAREL--a parse-request language.
# language #

9. 9 Bar-Hillel, Y.
Language and information.
# language #
CR 7178.

9. 10 Barron, D. W.
Assemblers and loaders.
# assemblers, loaders, systems #
"This short monograph presents a good introduction to the
subject. It covers symbol tables, one- and two-pass
assemblers, macro-assemblers, and meta-assemblers."
CR 19077.

9. 11 Barton, D. W., Buxton, 3. N., Hartley, D. P., Nixon, P., and
Strachey, C.
The main features of CPL.
Comp J 6, 2 (July 1963), 134-143.
# language #

9. 12 Bayer, R., Murphree, P., Jr., and Gries, D.
Users manual for the Alcor-Illinois 7090 ALGOL 60
translator, 2nd Ed.
U of Illinois, (Sept 1964).
# translator #

9. 13 Bennett, R. K.
A base for the definition of computer languages.
# languages #
CR 14509,

Data processing compilers for small card reading computers.
Proc ACM 14th Nat'l Conf. (1959), 63.
# compiler #
"An excellent guide to the various implementations of LISP and some comments on their specific realizations."

"compiling"

"compiler"

"This paper describes an ALGOL translator which has been implemented in the IBSYS Operating System. The system allows use of MAP procedures in ALGOL programs."

"MADCAP II is the latest version of the automatic programming compiler for MANIAC II.' This paper is an initial text for MADCAP II, explaining notation, algebra, progressing to code format, etc."

"compiler"

"compiler"

CR 0089.

ARNOTATED BIBLIOGRAPHY
9. 22  
Rrauholtz, T. G. M., Fraser, A. G., and Hunt, P. M.
NEBULA: a programming language for data processing.
Comp J 4, 3 (Oct 1961), 197-211.
# language #
"This is a general paper about NEBULA, written for people without previous experience with automatic programming languages. The authors describe basic properties of the language such as procedures and control statements. The authors conclude with an example of a program."

9. 23  
Rreitbard, G. Y., and Wiederhold, G.
The ACHE compiler.
Proc IFIP Congress (1968), Software I, Booklet B, 38-44.
# compiler #
CR 10661.

9. 24  
Brooker, R. A., and Clark, S. R.
An index directed compiler.
Comp J 10, 1 (1967), 60-63.
# compiler #
"The authors describe a system for discriminating between many possible combinations of operands and operators and invoking a routine to deal with them. The system is described with reference to the situation which may arise in a compiler."

9. 25  
Brown, S. A., Drayton, C. E., and Mittman, B.
A description of the APT language.
# language #
CR 5692.

9. 26  
Ruerket, J. B.
An experimental non-procedural language processor.
# processor #

9. 27  
Buxton, J. N., and Laski, J. G.
Control and simulation language.
# language #
This paper describes CSL, a programming language designed for use in complex logical problems, wherein predicate calculus is the basic approach. The article concludes with a description of compilation techniques." CR 3886.

9. 28  
Castle, J.
A command program compiler.
# compiler #
9. 29 Caves, W. E., and Toalinson, R. E.
The decision module compiler.
AD-679240, (May 1966), 298.
compiler 
"The authors describe the working and construction of DMC. DMC is an object time compiler whose end product is a complete running program."

9. 30 Chapin, N.
An implementation of IPL-V on a small computer.
ProcACM 19th Nat'l Conf. (1964), D1.2-1--D1.2-6.
compiler 
"This paper describes an implementation of IPL-V on the IBM 1620. The language is a list-processor and list processes are used in the implementation itself. Comparisons are made with other IPL-V implementations."

9. 31 Cheatham, T. E., and Warshall, S.
_translation of retrieval requests_ couched in a 'semiformal' English-like language.
translation 
"This paper is about the design of query translators which 'transform any user's description of the data desired into a characterization suitable for driving an automatic search procedure.' The authors include in the article an example of a language, QUERY, which they use to point out areas of difficulty in translation."

CL-S, an environment for a compiler.
compiler 
"The authors found a need for programmer-program intercommunication. They filled the need with a CL-1 programming system, which, in addition to the compiler, incorporates a filing program, data and separate data descriptions. The CL-1 environment provides a monitor and a raster file setup for large-scale information processing problems. It is an entire programming system, rather than simple a compiler."

9. 33 Chipps, J., Koschaann, M., Orgel, S., Perlis, A., and Smith, J.
A mathematical language compiler.
compiler 
"The authors list the components of all compilers and define the structure of a particular mathematical compiler."
9, 34 Christensen, C.
Examples of symbol manipulation in the AMBIT programming language.
A language
"This paper contains a brief informal description of AMBIT
and several examples of programs written in the language.
No implementation is described for the language because none
existed at the time of writing."
CR 10075.

9. 35 Christiansen, C.
On the implementation of AMBIT, a language for symbol
manipulation.
Comm ACM 9, 8 (Aug 1966), 570-573.
A language
"A brief description of the implementation technique of the
AMBIT replacement rule is given. An algorithm for the
'AMBIT SCAN' is given which provides a rationale for the
AMBIT language."

9. 36 Clippinger, R. P.
COBOL.
Colnp J 5, 3 (Oct. 1963), 177-140.
A language, compiler
"This article gives general information about extensions
made on COBOL between May, 1961, and April, 1962. Changes
discussed are generalized arithmetic verbs, sorting and
report writing."
CR 4095.

9. 37 Connors, T. B.
ADAM - a generalized data management system.
A language
"The ADAM system is a 'simulation program for data
management"3
CR 10822.

9. 38 Conway, M.
Proposal for an UNCOL.
Comm ACM 1, 10 (Oct 1958), S-8.
A compiler
"The language described is a proposed 'intermediate'
language into which any program-oriented language could be
coded and from which any machine language could be generated
by appropriate generators and translators."
9. 39  Conway, M. E.
Design of a separable transition-diagram compiler.
Comm ACM 6, 7 (July 1963), 396-408.

This paper presents a COBOL compiler design which describes a high speed, one-pass, syntax-directed compiler, which can be built in less than a year by two people with an assembler. The compiler uses coroutines, which perform lexical analysis, syntactical analysis, data structure analysis, and code generation. The author includes many illustrative figures.
CR 5024.

9. 40  Conway, M., and Speroni, J.
Arithmetizing declarations: an application to COBOL,
Comm ACM 6, 1 (Jan 1963), 24-27.

The compiler-writing
CR 5046.

9. 41  Conway, R. W., and Maxwell, W. L.
CORC--the Cornell computing language.
Comm ACM 6, 6 (June 1963), 317-321.

The language, compiler, error

CORC is designed for use by the non-professional programmer who is not highly concerned with the mechanics of a computer. The compiler provides extensive diagnostics. There are only nine different types of statements, no compiler-controlling declarations, and no decimal numbers. CORC will correct spelling errors, grammatical errors, and punctuation errors whenever possible.
CR 4778.

9. 42  Conway, R. W., Delfause, J. J., and Maxwell, W. L.
CLP--the Cornell list processor.

The highlights of CLP are presented along with examples of its main advantages. CLP is mainly a teaching language emphasizing simulation and list-processing.
CR 8254.

9. 43  Cook, D. P.
Automatic use of random access backing store in ALGOL programs.
Comp Bull. 11, 4 (Mar 1968), 301-302.

Storage allocation
CR 15410.
9. 44  
Coulouris, G. F., Goodey, T. J., Hill, R. W., Keeling, R. W., and Levin, D.  
The London CPL 1 compiler.  
*compiler, language*  
"The authors describe the compiler implemented on the London Atlas for the language CPL 1, which is very similar to CPL."  
CR 10663.

9. 45  
Cunningham, J. P.  
COBOL.  
Comm ACM 6, 3 (Mar 1963), 79-82.  
*language, compiler*  
"This article describes the documentation of COBOL. Included is a list of compilers being developed for COBOL 61."  
CR 5033.

9. 46  
Curtis, A. R., and Pyle, I. C.  
A proposed target language for compilers on ATLAS.  
Comp J 5, 2 (July 1962), 100-106.  
*language*  
"This article describes the target language BAS, which provides communication links between various parts of HARTRAN. One important new feature of BAS is that if storage for an array has not been assigned, it automatically assigns the requisite number of words in the appropriate part of store."  
CR 4178.

9. 47  
Dahl, O. J., and Nygaard, K.  
Basic concepts of SIMULA, and ALGOL-based simulation language.  
Norwegian Comp. Center, Oslo, Norway, (1967).  
*language*  
CR- 13635,  

9. 48  
De Vogelaere, R.  
Active language I.  
In Interactive Systems for Experimental Applied Mathematics,  
Klerer, M. and Reinfelds, J., (Eds.), Academic Press, N. Y.,  
(1968), 106-137.  
*language*  

9. 49  
Dijkstra, E. W.  
Recursive programming.  
Num. Rat, 2, (1960), 312-318.  
*compiling*  


AHNOTATED BIBLIOGRAPHY

9. 50  Cijkstra, E. W.
An ALGOL 60 translator for the X1.
Annual Review in Automatic Programming, Vol 3, (1963),
329-345. Pergamon Press, N. Y.
# translator #
"This article presents the structure of the object program
of an ALGOL 60 translator. A few of the features of this
particular translator are 1) reduction in size through use
of a discrimination vector rather than a transition matrix,
2) references to a subroutine coupler are numbered and the
translator punches only a number for these references, and
3) to a certain extent, it is independent of hardware
representation."  
CR 1391, 5676.

9. 51  Dijkstra, E. W.
ALGOL 60 translation.
ALGOL Bull. Suppl. No. 10, Stichting Mathematisch Centrum,
(Nov 1961).
# translation #

9. 52  Dolotta, T. A.
Les languages symboliques et leur edition.
(Symbolic languages and their editing.)
Chif res No. 3 (Sept 1962), 149-174. (French).
# language #
CR 4533.

9. 53  Duncan, P. G.
Comp 3 5 (July 1962), 130432.
# processors, compiling, optimization #
"This paper describes two ALGOL compilers, both
approximately the same size, both being written in user code,' both
accepting identical versions of ALGOL 60. They differ
in that one compiler has emphasis on fast compilation while
the other is 'aimed at recognizing and giving special
treatment to certain situations amenable to optimizations'."
CR 3571.

9. 54  Duncan, P. G., and Hurtable, D. R. R.
The DEUCE alphacode translator.
Comp J 3 (1961), 98.
# translator #
CR 0488.
Englund, D., and Clark, E.
The CLIP-translator.

"This article describes the translator for a language which is essentially an expanded ALGOL. The CLIP compiler is capable of reproducing itself and of writing JOVIAL-like language compilers. The authors discuss the translator: table packing, data generation, instruction generation. The article is not limited to discussion of CLIP--other methods of translation are touched upon throughout the article."

Ernst, H.
TCS an experimental multiprogramming system for the IBM 7090.

Ershov, A. P., and Rar, A. P.
SYGRA, a symbolic generator and macro-assembler.
In Symbolic Manipulation Languages and Techniques, North Holland Publishing Co., Amsterdam, (1968), 226-246.

"The authors make an attempt to define a machine-oriented programming system as a linguistic system with a number of free parameters. The language is considered to be a quadruple of 1) a set of syntactically admissible programs, 2) a programming processor, 3) a working processor with, 4) its operational memory."

Evans, A.
An ALGOL 60 compiler.

"This paper is a thorough discussion of the internal workings of an ALGOL translator used at Carnegie-Mellon University. The compiler is partly based on Polish postfix notation and the stack concept."

Evans, A., Jr.
An ALGOL 60 compiler.
9. 60 Falkoff, A. D., and Iverson, K. E.
The APL/360 terminal system.
Research Report RC 1922, IBM Watson Research Center,

9. 61 Falkoff, A. D., and Iverson, K. E.
The APL/360 terminal system.
In Interactive Systems for Experimental Applied Mathematics,
Klerer, M., and Reinfelds, J., (Eds.), Academic Press, N. Y.,
(1968), 22-37.

9. 62 Farber, D. J., Griswold, R. E., and Polonsky, I. P.
SNOROL, a string manipulation language.
ACM 11, 1 (Jan 1964), 21-30.

An ALGOL-based associative language.

9. 64 Floyd, R. W.
A descriptive language for symbol manipulation,

9. 65 Poster, D. M.
A simple list-processing interpreter,
Comp J 8 (1965), 120-129.

9. 66 Franklin, R. W.
Implementation of a compiler--GECOM.
Yreiburghouse, R. A.
The Multics PL/1 compiler.

Galler, R., and Perlis, A. J.
A proposal for definitions in ALGOL.

Gallie, T. M., Jr.
The Duke ALGOL compiler and syntactic routine method for syntax recognition.

Garvin, L., (Ed).
Natural language and the computer.

Garwick, J. V.
GARGOYLE, a language for compiler writing.
Comm ACW 7, 1 (Jan 1964), 16.

Garnick, J. V.
GPL, a truly general programming language.
Comm ACM 11, 9 (Sept 1968), 634-638.

Garwick, J. V.
The GPL compiler.
Proc IFIP (1968), Booklet B, 1-3.

Garwick, J. V., Bell, J., and Krider, L.
The GPL language,
Gau, A. A.
Recursive processes and ALGOL translation.

Gawlik, H. J.
MIRPAC, a compiler based on standard mathematical notation and plain English.
Comm ACM 6, 9 (Sept 1963), 545-547.

Genuys, F., (Ed).
Programming languages, a NATO advanced study institute summer school.

Ginsburg, S., and Spanier, E. H.
Bounded ALGOL-like languages,

Gorn, S.
An experiment in universal coding.

Gorn, S.
Common programming language task, Pt. I.

Graham, H. L., and Ingerman, P. Z.
A universal assembly mapping language.
ORACLE binary internal translator (ORBIT).
Comm ACM 4, 1 (Jan 1961), 19.
# translator #

Conclusions after using the PACT I advanced coding techniques.
3 ACW 3, 4 (Oct 1956), 309-313.
# compiler #
"The efficiency of the PACT I compiler and language is discussed, along with possible modifications and extensions of the system."

Alcor-Illinois 7090--an ALGOL compiler for the IBM 3090.
# compiler #

The design and simulation of an information processing system.
3 ACH 8, 2 (Apr 1961), 260-270.
# compiler, processor #
"This article presents the design of an information processing system which involves input/output, interpretation, storage allocation, retrieval of data, logical processing and correlation. These facets are discussed, and the author concludes by naming some basic problems of systems which handle language data."

9. A6 | Guzman, A., and McIntosh, H. V.
CONVERT.
Comm ACH 9, 8 (Aug 1966), 604-615.
# language #

9. A7 | Ralstead, M. H.
Machine-independent computer programming.
# compiling #
"This book is essentially a compilation of lecture notes from a course on NELIAC (a subset of ALGOL) taught by the author. A self-compiler was used in the course and most of the book is concerned with compilers and compiler systems."
9. 88 Halstead, M. H.
NELIAC,
Comm ACM 6, 3 (Mar 1963), 91-92.

"This article gives an account of current documentation on the NELIAC language." It does, however, briefly cover the topic of NELIAC compilers three features: self-compilation, relatively small and simple, relative speed. There is also a brief description of the language.

CR 5034.

9. 89 Hartaan, P. H.
A SMALGOL compiler for the ALUAC III-E at Oregon State University.
Comm ACM 6, 7 (July 1963), 365.

"(Abstract only). This version of SMALGOL has a one-pass translator using a push-down list. The compiler does not allow procedures but allows Boolean variables."

9. 90 Haynam, G. E.
An extended ALGOL-based language,

"This paper describes various ways in which ALGOL may be extended to provide any type of special facilities while retaining the generality of ALGOL."

9. 91 Flays, D. G.
Introduction to computable linguistics.

"This volume is intended as an introduction to the field of computational linguistics. It contains good coverage on such topics as algorithms, storage structures, representation of data in storage, look-up techniques, parsing strategies, and formal grammar theory."

9. 92 Rigman, B.
Towards an ALGOL translator.

"This article is a progress report on work being done on an ALGOL translator written in ALGOL. The process is to be done in five passes, and at the time of the writing, three passes had been completed. They are briefly described in this article. The translation process itself is then discussed."
Hoare, C. A. R.
Report on the Elliot ALGOL translator,
    # compiling #
Comp J 5, 2 (July 1962), 127-129.
    # translator #
"At the time of writing, the translator had not yet been completed. However, the method decided upon was as follows: the main aim is to be speed, to be accomplished by a translation system which accepts a source program in ALGOL, reads and translates it and transfers control to the translated program. If the length of the program dictates, a two-pass system will be incorporated?"
CR 3568.

Hackney, R. W.
ABS12 ALGOL: an extension to ALGOL 60 for industrial use.
Comp J 4, 4 (1962), 292-300.
    # language, compiler #
CR 5686.

Hornick, S. D.
IBM 709 tape matrix compiler.
Comm ACH 2, 9 (Sept 1959), 31-32.
    # compiler #
The tape matrix compiler performs matrix algebra on input which is given in a form which is much closer to matrix algebra notation than to coding. There is little discussion! of the inner workings of the compiler."
CR 0090.

Huskey, H. D., Halstead, M. H., and McArthur, R.
NELIAC—a dialect of ALGOL.
Comm ACM 3, 8 (Aug 1960), 463.
    # language #

Huxtable, D. H. R.
On writing an optimizing translator for ALGOL 60.
In Introduction to Systems Programming, P. Wegner, (Ed.),
    # translator, optimization #
CR 6307.

Tanov, T. I.
The logical schemes of algorithms.
In Problems of Cybernetics 1, 82-140.
    # compiling #
9. 99
International Computation Centre, (Eds).
Symbolic language in data processing, proceedings of the
# compiling #

9.100
International Standards Organization Survey of programing
languages and processors.
Coma ACM 6, 3 (Mar 1963), 98-99.
# languages, processors #
"This article is six pages of charts surveying programming
languages and processors, giving the following information:
language, author, Machine, minimum configuration and notes."

9.101
Isbitz, H.
CLIP, a compiler language for information processing.
System Development Corp., Santa Monica, Calff. (1959),
9 pp.
Proc ACM 14th Nat'l Conf. (1959), 73.
# compiler #
CR 0322.

9.102
Iverson, K. E.
A programming language,
# language #
"The author presents a programming language in detail and
then applies the language to such topics as sorting and
logical calculus. The book is in textbook format, with
exercises at the end of each chapter."

9.103
Kanner, H.
An algebraic translator.
Comm ACM 2, 10 (Oct 1959), 19-22.
# translator #
"The author presents a translator which is similar to that
of 3. H. Uegstein (Comm ACM, Mar, 1959). A flowchart is
included."

9.104
Kerr, R. H., and Clegg, J.
The Atlas ALGOL compiler--an ICT implementation of ALGOL
using the Brooker-Morris syntax directed compiler.
Comm J (1967).
# compiler #

9.105
Knowlton, K. C.
A programmer's description of L SIX,
Comm ACM 9, 8 (Aug 1966), 616-625.
# language #
9.106 Knuth, D. E.
RUNCIBLE, algebraic translation on a limited computer.
 translator, compiler #
"The RUNCIBLE compiler is described. It is designed for a
small to intermediate sized machine. The translation
process is largely described by a flowchart."

9.107 Knuth, D. E.
History of writing compilers.
 compilers #

9.108 Knuth, D. E.
A history of writing compilers.
 compilers #
"This paper describes the various components of compilers
and how different compilers have handled formula breakdown
and object code generation?
 CR 3133,

9.109 - Lauer, P.
Formal definition of ALGOL 60
Tech. Rept. No. TR 25.088, IBM Labs., Vienna, Austria (Dec
1968).
 syntax, semantics #

9.110 Leavens, R. M.
Fortran IV as a syntax language.
Comm ACM 7, 2 (Feb 1964), 72-80.
 language, syntax #
 CR 6000.

9.111 Ledgard, H. P.
Ten mini-languages in need of formal definitions.
SIGPLAN 5, 4 & S (Apr 1970), 14-37.
 language, compilers #

9.112 Lomet, D. B.
The construction of efficient deterministic language
processors,
 translators #
 CR 19078.

9.113 Lucas, P.
Definition of a subset of PL/1 by finite state local
vectors.
 language #
9,114 Lucas, P., and Walk, K.
On the formal description of PL/1.
# syntax #

9,115 Macleod, I. A.
SP/1--a FORTRAN integrated string processor.
# extendible language #

9,116 Markowitz, H. M., Hausner, B., and Karr, H. W.
SIMSCRIPT 1.5, a simulation programming language.
# language #
CR 12763.

9,117 Masterson, K. S.
Compilation for two computers with NELIAC.
# compiler-compiler #
"NELIAC is able to *bootstrap* itself and to generate a CDC 1604 compiler on a UNIVAC COUNTESS computer. A description of the characteristics of the compiler is given along with an operational description."
CR 3566.

9,118 McCarthy, J. et. al.
LISP 1.5 programmers manual.
Computation Lab Rept, MIT (1962).
# compiler #
"This is the original LISP 1.5 program manual and description. One of the appendices contains a description of the LISP compiler (which is written in LISP)."
CR 5689.

9,119 Nealy, G.
A generalized assembly system.
RM-3646-PR.
# translator #

9,120 Mendicino, S. F., Hughes, R. A., Martin, J. T., McMahon, F. H., Ranelletti, J. E., and Zwakenberg, R. G.
The LR/TRAN compiler.
# compiler #
LCC: a language for conversational computing.
In Interactive Systems for Experimental Applied Mathematics,
Klerer, M. and Reinfelds, J., (Eds.), Academic Press, N. Y.,
(1968), 203-214.
# language #

9.122 Mooers, C., and Deutsch, L. P.
TRAC, a text handling language.
Proc ACM 20th Nat'l Conf. (1965), 229-246.
# language, compiler #
"This paper is a description of the TRAC language. TRAC is
an extendible language which was designed for use with the
reactive typewriter. The paper also describes the design
decisions made in writing the system."

9.123 Mooers, C. N.
TRAC, a procedure-describing languages for the reactive
typewriter.
Coma ACM 9, 3 (War 1966), 215-219.
# language, extendible #
"A description of TRAC is given along with a processing
algorithm. TRAC is based on a generalization of the concept
of the 'macro'. TRAC has the ability to accept and store
definitions of procedures and thus indefinitely extend
itself."

9.124 Moraff, N.
Business and engineering enriched Fortran (BEEP).
# language #
"This paper describes an extension of Fortran to make it
usable for both business and engineering problems. The
extension is accomplished through the addition of numerous
CALL-able subroutines."

9.125 Norris, D., and Rohl, J. S.
The Atlas compiler system.
Comp J 10 (Nor 1967), 227-230.
# compiler #

9.126 Morris, D., and Wilson, I.
A system program generator.
Computer Science Dept., Univ. of Ranchester, Manchester,
# generator, processor #

9.123 Morrison, R. A.
Graphic language translation with a language independent
compiler.
# translation, compiler #
9.128  Napper, R. B. E.  
The third-order compiler: a context for free ran-machine communication.  
# compiler-compiler #  
"The author introduces the concept of third-order compilers which would provide to the compiler-writer facilities similar to those provided by the second-order compiler to the ordinary programmer."

Revised report on the algorithmic language ALGOL 60.  
Comm ACM 6, 1 (Jan 1963), 1-17.  
Comp J 5 (1963), 349-367.  
# language #  
"The report gives a complete defining description of the international algorithmic language ALGOL 60. The language is dissected very systematically, beginning with the structure, then continuing on to basic symbols, expressions, statements and declarations. Numerous examples and an alphabetical index of definitions of concepts and syntactic units is included."

Report on the algorithmic language ALGOL 60.  
# syntax, semantics, language #  
"This is a final report on ALGOL 60, and consists of a listing of the complete syntax of the language."

Information processing language V manual, 2nd ed.  
# language #

NUCLEOL--a minimal list processor,  
SIGSAM Bull. 12 (July 1969), 40-52.  
Publ. by ACM Special Interest Group on Symbolic and Alg. Manipul.  
# processor #
9.133 Opler, A., Farhman, D., Heit, M., King, W., O'Connor, E.,
Goldfinger, A., Landow, R., Ogle, J., and Slesinger, D.
Automatic translation of programs from one computer to
another.
(Preprints).
# translation #

9.134 Painter, J. A.
Semantic correctness of a compiler for an ALGOL-like
language.
# theory #

9.135 Paul, M.
Kolloquium fur sprachen und algorithmen.
# language #

9.136 Perlis, A. J.
A format language.
Comm ACM 7, 2 (Feb 1964), 89-97.
# language #
"This paper describes a format system for input/output in an
ALGOL-like language."

9.137 Perlis, A. J., and Iturriaga, R.
An extension to ALGOL for manipulating formulae.
Comm ACM 7, 2 (Feb 1964), 127-130.
# compiler #

Report on the algorithmic language ALGOL, by the ACM
Committee on Programming Language and the GMM Committee on
Programming.
Numer Math. 1 (1959), 41-60.
Comm ACM 1, 12 (1958), 8-22.
# language #
CR 0323, 3585.

INTERNAL translator (IT), a compiler for the 650.
Carnegie Inst. of Technology Computation Center, (Jan 1958).
# compiler #
9.140 Petrone, L.  
Un compilatore algebrico per l'USS 90.  
(An algebraic compiler for the USS 90.)  
Atti del convegno sui linguaggi simbolici di programmazione,  
AICA, (Jan 1962), 80-82. (Italian).
# compiler  
CR 3569.

9.141 Plaskow, J., and Schuman, S.  
The TRANGEN system on the M460 computer,  
AFcRL-66-516 (July 1966).
# compiler

9.142 Pollack, B. W.  
Compiler techniques.  
# compilers, translators, interpreters, processors  
"This book presents a summary of the basic techniques necessary for the implementation of compilers. A wide variety of subjects is covered including syntax, parsing, resource allocation, detection and correction of errors, and details of compiler construction.*

9.143 Porter, S. W., and Porter, C. B.  
NELIAC-1604-a compiler for the Control Data Corporation 1604 computer, Sept. 1961 version,  
# compiler

9.144 Porter, S. W., and Porter, C. B.  
NELIAC-a compiler for Burroughs 220 computer, Jan 1961 version.  
# compiler

9.145 Rabinowitz, I. N.  
Report on the algorithmic language FORTRAN II.  
Corm ACM 5, 6 (June 1962), 327-337.  
# translator, syntax  
"Most of this article is a description in modified BNF of the syntax of IBM 7090 FORTRAN II. A compiler which is directed by a syntax table and can compile any language by reference to an appropriate table is mentioned but the idea is not discussed at length."  
CR 3878.

9.146 Presser, L.  
The structure, specification, and evaluation of translators and translator writing systems.  
Rept. 68-51, Univ. of Calif., Los Angeles, Calif. (Oct 1968).  
# translators
9.147

Resnick, M., and Sable, J.

**INSCAN**: a syntax-directed language processor.


"INSCAN is a convenient tool for expressing the syntax of linear languages and for specifying the actions necessary to translate or otherwise process languages. It has been implemented at Auerbach. The INSCAN approach to language processor design separates the language scanning and translation function from the details of the post-translation processing and facilitates experimentation with the design of languages.

CR 15767.

9.148

Reviglio, G.

**La programmazione automatica**: il linguaggio algebrico ALGOL.

[Automatic programming: the algebraic language ALGOL.]


" language "

CR 4779.

9.149

Reynolds, J. C.

An introduction to the COGENT programming system.


"COGENT is a list-processing compiler-compiler. It provides full generality with regard to input language and target language; the syntax and translation rules must be input to the system. COGENT may be used directly for algebraic manipulation, theorem proving and heuristic programming in addition to its function as a compiler-compiler.*

9.150

Reynolds, J. C.

COGENT programming manual.


" language, compiler-compiler "

9.151

Reynolds, J. C.

**GEDANKEN**--a simple typeless language based on the principle of completeness and the reference concept.


" language, compiler "
9.152 Rich, R. P.
APT, a common computer language.
# language #
"The APT language here described is intended to serve as a common computer language for computational problems. In this article the author lists first terms, then statements. He points out that use of APT may cut the cost of compilation."

9.153 Roberts, L. G.
A graphical service system with variable syntax.
# syntax #

9.154 Rosen, S., and Goldberg, I. B.
ALTAC, the TRANSC algebraic translator.
Proc ACM 14th Nat'l Conf. (1959), 62.
# translator, compiler #

TAC, the TRANSC assembler-compiler.
Proc ACM 14th Nat'l Conf. (1959), 60.
# compiler #

PUPPT—Perdue University fast Fortran translator,
Comm ACM 8, 17 (Nov 1965), 661-666.
# compiler #
"This paper describes a high-speed system for the complete Fortran IV language, including the subroutine library. The system included an elaborate diagnostic message routine."

9.157 Ross, D. T.
AED bibliography.-
# processor #

9.158 Rousell, A. R.
A progress report on NEBULA.
Comp J 5, 3 (Oct 1962), 162-163.
# compiler #
"This article reviews some of the reasons for changing from COBOL to NEBULA, which include freedom of choice of data media and format for the media. One current problem of NEBULA is that it cannot handle large programs. However, a new compiler is planned to facilitate this problem."
CR 3882.
9.159 Rutledge, J. D.
On Tanov's program schemata.
ACM 12, 1 (Jan 1964), 1-9.
# formal
CR 8337.

9.160 Sammet, J. E.
A definition of COBOL 6 1.
# language

9.161 Sammet, J. E.
Programming languages: history and fundamentals.
# language

9.162 Sandewall, E. J.
LISP A: a LISP-like system for incremental computing.
# incremental computing

9.163 Schlesinger, S., and Sashkin, L.
POSE: a language for posing problems to a computer.
# language
"POSE utilizes Fortran formulas and logical representation but is very different from Fortran. POSE programs consist of a problem statement in equation form, the 'compiler' supplies the method of solution and performs all clerical chores. Presents the concept of 'solution-compiler'."
CR 12752.

9.164 Schorre, V. A.
A syntax directed SMALGOL for the 1401.
ACM 6, 7 (July 1963), 365.
# syntax-directed
"(Abstract only). A syntax-directed compiler is proposed which would save space during compilation and could be implemented on a small machine."

WATFOR--the University of Waterloo FORTRAN IV.
ACM 10, 1 (Jan 1967), 41-44.
# compiler, language
9.166  Shaw, C. J.
A specification of JOVIAL.
Comm ACM 6, 12 (Dec 1963), 721-735.
# compiler, syntax #
"This is a report on the state of JOVIAL as of 1963. It is mostly concerned with the current formal specification of the language."
CR 6322.

9.167  Shaw, C. J.
JOVIAL.
Dataaation 7, 6 (June 1961), 28-32.
# language #
"JOVIAL is a language aiding the programmer in the area of notation. The notation is more powerful and machine symbology has been greatly reduced as compared to other languages. In addition to discussing the language itself, the author briefly discusses JOVIAL compilers, which share the common feature of two sub-programs, the 'generator' and the 'translator'."
CR 1216.

9.168  Sibley, E. R.
The engineering assistant: design of a symbol manipulation system.
# language, compiler #

9.169  Siegal, H., and Painter, J.
The use of generators in TAC.
Proc ACM 14th Nat'l Conf. (1959), 61.
# generator, compiler #

9.170  Simon, H.
Experiments with a heuristic compiler.
The RAND Corp. P-2349, (June 1961).
# compiler, language #
"This article describes experiments in the construction of a heuristic compiler. The author begins with a general survey of the heuristic compiler and then goes on to describe routines for compiling programs: SDSC compiler, DSCN compiler, and general compiler."
CR 2904.

9.171  Simon, H.
The heuristic compiler.
# compiler #
CR 4775.
9.172 Smith, J. W.
Syntactic and semantic augments to ALGOL.
# syntax, semantics #
"Some possible extensions of ALGOL are proposed; most are
concerned with string manipulation."
CR 0214.

9.173 Stark, R.
ALTEXT—a multiple purpose language,
Lockheed Missiles and Space Company, Tech. Rept. 6-75-65-15,
(Mar 1965).
, # language #

9.134 Starynkevitch, D.
La programmation autonatipue des formules sur CAB 500.
(Automatic programming of formulas on the CAB 500.)
Elektr Daten 9 (1961), 1-S. (French).
# translation #
CR 1215.

9.175 Steel, T. B., Jr.
PACT IA.
3 ACR 4, 1 (Jan 1957), 8-11.
# compiler #
"PACT IA is a modified PACT I compiler for use on the IBM
704. Again, compilation is done in stages and requires
several tape passes."

9.176 Steel, T. B., Jr.
A first version of UNCOL.
# language #
"The specifications for a possible universal
computer-oriented language are presented."
CR 2142,

9.177 Steel, T. B., Jr.
UNCOL, Universal Computer Oriented Languages revisited.
Datamation 6, (Jan/ Feb 1960), 14-20.
# language #
CR 0764.

9.178 Steel, T. B., Jr., (Ed).
Formal language description languages for computer
programming.
# meta-languages, formal languages #
9,179 Stiegler, A. D.
A compiler for the Fortran assembly program.
Comm ACM 5, 11 (Nov 1962), 545.

A compiler called by a pseudo-operation can be used during the assembly process to compile algebraic expressions that are in the variable field of an assembly listing.*

9,180 Tapella, C. A.
A MELIAC compiler for the CDC 1604.

9,181 Taylor, R., and Harragan, D. A.
The Fortran system for ORION.

The ORION Fortran Orion system allows the compilation, assembly and editing of programs written in Fortran or a symbolic assembly language. The system is described. The system was written in Fortran and its utility in this respect is discussed.

CR 3627.

9,182 Taylor, W., Turner, L., and Uaychoff, R.A.
A syntactical chart of ALGOL 60.
Comm ACM 4, 9 (Sept 1961), 393.

The authors have prepared a syntax chart for ALGOL 60 which contains every basic symbol. The chart aided them in writing a compiler, and they suggest its use in checking the syntax of a program written in ALGOL 60.*

9,183 Teichroev, D., and Lubin, J. P.
Computer simulation-discussion of the technique and comparison of languages.

The purpose of this paper is to present a comparison of some computer simulation languages and some of their implementations.

CR 11466.

9,184 Thompson, F. B.
English for the computer.

This paper is a discussion of the use of natural English as a computer language. It develops a point of view which is realized in the DEACON system.

CR 12698.
9.185 Ushijman, K., Arita, I., and Otsuki, S.
A conversational processor for a structuring language.
Info. Processing in Japan, 8 (1968), 37-40.
"The authors discuss conversational processing on a
time-sharing system using a block-structured language.
Discussion is with particular reference to an experimental
ALGOL compiler implemented on an OKITAC 5090 computer."

9.186 Van der Poel, W. L.
List and string processing in general programming languages.
Symbol manipulation languages and techniques, 191-206.

9.187 van Wijngaarden, A,
Generalized ALGOL: symbolic languages in data processing.

9.188 Watt, J. R., and Wattenburg, W. H.
A NELIAC-generated 7090-1401 compiler.
"This brief article summarizes the results of a project in
which NELIAC was used to generate a compiler for the IBM
1401. The net results of the project were reduction in
programming time and improvement in documentation of the
system."
CR 1660.

9.189 Wegstein, J.
A general purpose pseudocode.
Proc ACM 9th Nat.1 Conf. (June 1954).

9.190 Weizenbaum, J.
Symmetric list processor.
Comm ACM 6, 9 (Sept 1963), 524-544.
Wells, M. B.

MADCAP, a scientific compiler for displayed formula textbook language.
Comm ACR 4, 1 (Jan 1961), 31-36.
"MADCAP is a scientific language designed so that input will approach textbook form and be easily readable. The compiler written to handle this language is described, along with the problems involved in compiling a textbook-type language."

# compiler #

Wells, M. B.

Recent improvements in MADCAP.
Comm ACM 6, 11 (Nov 1963), 674-678.
# compiler #

"This paper discusses improvements of MADCAP in three areas: complex display, a notation for integration, a notation for binomial coefficients. In logical control a notation for variably nesting loops has been developed. Finally, in subroutine, the main improvement discussed is the notation and use of procedures."
CR 5687.

Woodger, M., (Ed).

Supplement to the ALGOL 60 report.
# syntax, language #

"This report specifically lists the changes to ALGOL 60 which are incorporated in the revised report published in the same issue of the Comm ACM."

Yngve, V. H.

Towards better programming languages.
Proc ACM 17th Nat.1 Conf. (1962).
# language #

Yngve, V. H.

A model and an hypothesis for language structure.
# languages #
CR 1043.

Young, J. W., Jr.

Ronprocedural languages.
# languages #
9.197 Yushchenko, K. L., and Ristrova, L. P.  
A processor for an algebraic programming language for the Kiev computer.  
*Zh Prats z Obchisl Mat i Tekhn*, (1961), 30-41.  
Ref *Zh flat No. 2B* (Feb 1963), Rev. 2B421 (Ukrainian).  
# processor, language #  
CR 5356.

9.198 Zara, R. V.  
A semantic model for a language processor.  
*Proc ACM 22nd Nat'l Conf.* (1967), 323-339.  
# processor #  
CR 15148.

9.199 Zaremba, W. A.  
A *syntax* for ALGOL input/output formats.  
*Comp J* 12, 4 (Nov 1969), 342-348.  
# language, syntax #

9.200 Zemanek, H.  
Semiotics and programming languages.  
*Comm ACM* 9, 3 (Mar 1966), 139-143.  
# languages #  
"This article concerns the application of 'semiotics' to programming languages.  @Semiotics@ consists of three branches: syntactics, semantics and *pragmatics*."