

# Literaturverzeichnis

- [Alb90] ALBLAS, H.: *Concurrent incremental attribute evaluation*. In: *Proceedings of the International Conference on Attribute Grammars and Their Applications*, Band 461 der Reihe LNCS, Seiten 343–358. Springer-Verlag, 1990.
- [Alb91] ALBLAS, H.: *Attribute evaluation methods*. In: ALBLAS, H. und B. MELICHAR (Herausgeber): *Proceedings of the International Summer School on Attribute Grammars, Applications, and Systems (Prag, Tschechische Republik)*, Band 545 der Reihe LNCS. Springer-Verlag, Juni 1991.
- [AM91] ALBLAS, H. und B. MELICHAR: *International Summer School on Attribute Grammars, Applications, and Systems (Prag, Tschechische Republik)*, Band 545 der Reihe LNCS. Springer-Verlag, Juni 1991.
- [Apt90] APT, K.R.: *Logic Programming*. In: LEEUWEN, J. VAN (Herausgeber): *Handbook of Theoretical Computer Science, Vol. B*, Kapitel 10, Seiten 493–574. Elsevier, 1990.
- [ASU86] AHO, A.V., R. SETHI und J.D. ULLMAN: *Compilers — Principles, Techniques, and Tools*. Addison-Wesley, 1986.
- [AU62] AHO, A.V. und J.D. ULLMAN: *The Theory of Parsing, Translation and Compiling, Vol. I*. Prentice-Hall, 1962.
- [AU73] AHO, A.V. und J.D. ULLMAN: *The Theory of Parsing, Translation and Compiling, Vol. II*. Prentice-Hall, 1973.
- [Bar84] BARENDREGT, H.P.: *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, 1984.
- [Boc76] BOCHMANN, G.: *Semantic evaluation from left to right*. *Comm. Assoc. Comput. Mach.*, 19:55–62, 1976.
- [Boy96] BOYLAND, J.T.: *Conditional attribute grammars*. *ACM Trans. on Progr. Lang. and Systems*, 18:73–108, 1996.
- [CF82] COURCELLE, B. und P. FRANCHI-ZANNETTACCI: *Attribute grammars and recursive program schemes*. *Theoret. Comput. Sci.*, 17:163–191, 235–257, 1982.

- [Cha90] CHAPMAN, N. P.: *Defining, analysing, and implementing communication protocols using attribute grammars*. Formal Aspects of Computing, 2:359–392, 1990.
- [Chu33] CHURCH, A.: *A set of postulates for the foundation of logic*. Am. Math. Soc., 2:33–34, 346–366, 839–864, 1932,1933.
- [DJ90] DERANSART, P. und M. JOURDAN: *Attribute Grammars and Their Applications, International Conference WAGA, Paris*, Band 461 der Reihe LNCS. Springer-Verlag, September 1990.
- [DJI88] DERANSART, P., M. JOURDAN und B. LORHO: *Attribute grammars*, Band 323 der Reihe LNCS. Springer-Verlag, 1988.
- [DM85] DERANSART, P. und J. MALUSZYNSKI: *Relating logic programs and attribute grammars*. J. Logic Programming, 2:119–155, 1985.
- [DPSS77] DUSKE, J., R. PARCHMANN, M. SEDELLO und J. SPECHT: *IO-macrolanguages and attributed translations*. Inf. Control, 35:87–105, 1977.
- [DRT81] DEMERS, A., T. REPS und T. TEITELBAUM: *Incremental evaluation for attribute grammars with application to syntax-directed editors*. Conference Record of the 8th ACM Symposium on Principles of Programming Languages, Seiten 105–116, 1981.
- [EF82] ENGELFRIET, J. und G. FILE: *Passes, sweeps and visits*. Memorandum INF-82-6, Twente University of Technology, Enschede, The Netherlands, 1982.
- [EF89] ENGELFRIET, J. und G. FILE: *Passes, sweeps, and visits in attribute grammars*. J. Assoc. Comput. Mach., 36:841–869, 1989.
- [Eng80] ENGELFRIET, J.: *Some open questions and recent results on tree transducers and tree languages*. In: BOOK, R.V. (Herausgeber): *Formal language theory; perspectives and open problems*, Seiten 241–286. New York, Academic Press, 1980.
- [Eng84] ENGELFRIET, J.: *Attribute grammars: attribute evaluation methods*. In: LORHO, B. (Herausgeber): *Methods and tools for compiler construction*, Seiten 103–138. Cambridge University Press, 1984.
- [Eng85] ENGELFRIET, J.: *Formele Talen en Automaten 2 (in Niederländisch)*. Vorlesungsmitschrift, Universität Leiden, Die Niederlande, 1985.
- [EV85] ENGELFRIET, J. und H. VOGLER: *Macro tree transducers*. J. Comput. Syst. Sci., 31:71–145, 1985.
- [FHV93] FÜLÖP, Z., F. HERRMANN, S. VAGVÖLGYI und H. VOGLER: *Tree transducers with external functions*. Theoret. Comput. Sci., 108:185–236, 1993.
- [Fil83] FILE, G.: *Theory of attribute grammars*. Doktorarbeit, Twente University of Technology, 1983.

- [Fis68] FISCHER, M.J.: *Grammars with macro-like productions*. Doktorarbeit, Harvard University, Massachusetts, 1968.
- [Fra82] FRANCHI-ZANNETTACCI, P.: *Attributs semantiques et schemas de programmes*. Doktorarbeit, Universite de Bordeaux I, 1982.
- [Fül81] FÜLÖP, Z.: *On attributed tree transducers*. Acta Cybernetica, 5:261–279, 1981.
- [Hud89] HUDAK, P.: *Conception, evolution, and application of functional programming languages*. ACM Computing Surveys, 21:359–411, 1989.
- [Jaz81] JAZAYERI, M.: *Simpler construction for showing the intrinsically exponential complexity of the circularity problem for attribute grammars*. J. Assoc. Comput. Sci., 28:715–720, 1981.
- [JOR75] JAZAYERI, M., W. F. OGDEN und W. C. ROUNDS: *The intrinsically exponential complexity of the circularity problem for attribute grammars*. Comm. Assoc. Comput. Sci., 18:697–706, 1975.
- [Kas80] KASTENS, U.: *Ordered attribute grammars*. Acta Informatica, 13:229–256, 1980.
- [Kas90] KASTENS, U.: *Übersetzerbau*. Oldenbourg-Verlag, 1990.
- [Kes96] KESSELER, M.: *The implementation of functional languages on parallel machines with distributed memory*. Doktorarbeit, Universität Nijmegen, 1996.
- [Knu68] KNUTH, D.E.: *Semantics of context-free languages*. Math. Syst. Theory, 2:127–145, 1968. Corrections in *Math. Syst. Theory*, 5:95–96, 1971.
- [Knu90] KNUTH, D. E.: *The genesis of attribute grammars*. In: *Attribute Grammars and their Applications, Proceedings of the International Conference WAGA, Paris, Frankreich*, Band 461 der Reihe *LNCS*, Seiten 1–12. Springer-Verlag, September 1990.
- [Kow74] KOWALSKI, R.: *Predicate logic as a programming language*. Information Processing, 74:569–574, 1974.
- [KW76] KENNEDY, K. und S. K. WARREN: *Automatic generation of efficient evaluators for attribute grammars*. 3rd ACM Symposium on Principles of Programming Languages, Proceedings, Seiten 32–49, 1976.
- [Llo87] LLOYD, J.W.: *Foundations of Logic Programming*. Springer-Verlag, 1987. Second, extended edition.
- [LMW86] LOECKX, J., K. MEHLHORN und R. WILHELM: *Grundlagen der Programmiersprachen*. B. G. Teubner Stuttgart, 1986.
- [Loo90] LOOGEN, R.: *Implementierung funktionaler Programmiersprachen*. Nummer 232 in *Informatik Fachberichte*. Springer-Verlag, 1990.

- [Loo93] LOOGEN, R.: *Integration funktionaler und logischer Programmiersprachen, Semantik und Implementierung*. Oldenbourg-Verlag, 1993.
- [Lor84] LORHO, B.: *Methods and Tools for Compiler Construction*. Cambridge University Press, 1984.
- [LRS74] LEWIS, P.M., D.J. ROZENKRANTZ und R.E. STEARNS: *Attributed translations*. J. Comput. Syst. Sci., 9:279–307, 1974.
- [Mah88] MAHN, U.: *Attributierte Grammatiken und Attributierungsalgorithmen*, Band 157 der Reihe *Informatik-Fachberichte*. Springer-Verlag, 1988.
- [MV95] MÖSSLE, A. und H. VÖGLER: *Efficient call-by-value evaluation strategy of primitive recursive program schemes*. Technischer Bericht TUD/FI95/19, Technische Universität Dresden, 1995.
- [MW88] MAIER, D. und D. S. WARREN: *Computing with Logic*. Benjamin Cummings Publishing Company, 1988.
- [Nol93] NOLL, T. persönliche Mitteilung, 1993.
- [NV94] NOLL, T. und H. VÖGLER: *Top-down parsing with simultaneous evaluation of noncircular attribute grammars*. Fundamenta Informaticae, 20:285–332, 1994.
- [odA88] AKKER, R. OP DEN: *Parsing attribute grammars*. Doktorarbeit, Universität Twente, Enschede, Die Niederlande, Dezember 1988.
- [odAMT91] AKKER, R. OP DEN, B. MELICHAR und J. TARHIO: *Attribute evaluation and parsing*. Technischer Bericht INF 81-91, Universität Twente, Enschede, Die Niederlande, April 1991.
- [Oud93] OUDE LUTTIGHUIS, P.: *Parallel algorithms for parsing and attribute evaluation*. Doktorarbeit, Universität Twente, Enschede, Die Niederlande, 1993.
- [Paa95] PAAKKI, J.: *Attribute grammar paradigms — a high level methodology in language implementation*. ACM Computing Surveys, 27:196–255, 1995.
- [Pav93] PAVLO, P.: *LR Parsing L-attribute Grammars*. Doktorarbeit, Tschechische Technische Universität, Prag, Tschechische Republik, März 1993.
- [Pey87] PEYTON JONES, S.L.: *The Implementation of Functional Programming Languages*. Series in Computer Science. Prentice-Hall, 1987.
- [PL92] PEYTON JONES, S.L. und D. LESTER: *Implementing functional languages, a tutorial*. Prentice-Hall, 1992.
- [Rob79] ROBINSON, J.A.: *Logic: Form and Function*. Edinburgh University Press, 1979.
- [RT89] REPS, T. W. und T. TEITELBAUM: *The synthesizer generator — a system for constructing language-based editors*. Springer-Verlag, 1989.

- [Saa78] SAARINEN, M.: *On constructing efficient evaluators for attribute grammars*. LNCS, 62:382–397, 1978.
- [Sch89] SCHÖNING, U.: *Logik für Informatiker*. Wissenschaftsverlag, 1989.
- [Sch95] SCHMITZ, L.: *Syntazbasierte Programmierwerkzeuge*. Teubner-Verlag, 1995.
- [Thi94] THIEMANN, P.: *Grundlagen der funktionalen Programmierung*. Teubner-Verlag, 1994.
- [vEK76] EMDEN, M. H. v. und R. A. KOWALSKI: *The semantics of predicate logic as a programming language*. J. Assoc. Comput. Sci. 23 (176), 733–742, 1976.
- [Vog91] VOGLER, H.: *Functional description of the contextual analysis in block-structured programming languages: a case study of tree transducers*. Science of Computer Programming, 16:251–275, 1991.
- [WM92] WILHELM, R. und D. MAURER: *Übersetzerbau — Theorie, Konstruktion, Generierung*. Springer-Verlag, 1992.

# Index

- abgeleitete Auswertungsordnung, 162
- Abhängigkeit, 31
- Abhängigkeitsgraph, 31, 35
- Ableitung, 12
- Ableitungsbaum, 13, 35
- Ableitungsrelation, 12
- absolutely noncircular, 117
- abstrakter Syntaxbaum, 17, 101
- AddGoto*, 133, 137
- AddPlan*, 133, 136
- äquivalente Attributgrammatiken, 29
- äquivalente Grammatiken, 12
- Äquivalenz, 29
- Aktivierungsanzahl, 172
- AlgorithmusAddGoto*, 133
- Algorithmus *AddGoto*, 137
- Algorithmus *AddPlan*, 133, 136
- Algorithmus des KW-Auswerters, 124
- Algorithmus *evaluate*, KW-Auswerter, 124
- Algorithmus *FirstGoto*, 135
- Algorithmus *MakeEvaluator*, 134
- Algorithmus *MakeInitStates*, 135
- Algorithmus *MakePlan*, 131
- Algorithmus *P-evaluate*, 53
- Algorithmus pure pass evaluator, 53
- Algorithmus pure sweep evaluator, 51
- Algorithmus pure visit evaluator, 49
- Algorithmus *S-evaluate*, 52
- Algorithmus simple pass evaluator, 56
- Algorithmus simple sweep evaluator, 55
- Algorithmus simple visit evaluator, 54
- Algorithmus *sP-evaluate*, 57
- Algorithmus *sS-evaluate*, 56
- Algorithmus *sV-evaluate*, 55
- Algorithmus *V-evaluate*, 50
- Algorithmus zur Berechnung der *is*-Graphen, 120
- Algorithmus zur Berechnung der *is-sets*, 38
- Algorithmus zur Berechnung der *is/si*-Graphen, 164
- Algorithmus zur Erstellung einer Konstruktionsinformation, 134
- Algorithmus zur Konstruktion eines Plans, 131
- Anfangsauswertungszustand, 122
- Anfangskonfiguration, 151
- Anfangszustand, 144
- angewandte Produktion, 14
- OAG*-Instruktion, 171
- Argumentposition, 91
- Attribut, 1, 4, 22, 26
- Attributauswerter, 47–49, 117, 119
- Attributauswertung, 47
- Attributbeschreibung, 22
- Attributgrammatik, 1, 5, 21–23, 58
- Attributgrammatik zu einem Logik-Programm, 89, 97–99
- Attributinstanz, 26
- Attributvorkommen, 22, 26
- Ausdrucksstärke, 70, 74
- Ausgaberegister, 144
- Außen-Argumentposition, 91
- Außenattributvorkommen, 22
- auswertbare semantische Regel, 130
- Auswertungsordnung, 162
- Auswertungsplan, 122
- Auswertungszustand, 121
- Auswertungszustandsgraph, 132
- AZS*-Graph, 132
- Bedeutung einer Attributgrammatik, 29
- Bedeutungsattribut, 22
- Berechnung der *is-sets*, 38
- Berechnungsterme, 104, 105

- beschriebene Sprache, 29
- beschriebene string-to-value Übersetzung, 29
- beschriebene Übersetzung, 29
- Beschriftung eines Knotens, 13, 14, 16
- Besuchsreihenfolge, 54
- betroffene Attributinstanz, 175
- betroffener Knoten, 175
- Beweisbaum, 84
- Bezeichnung einer Klasse, 58
- Blatt, 13, 14
- Bochmann-Normalform, 21
- brother-graph, 61
- compare, 142
- definite Hornformel, 82
- Definition einer Attributgrammatik, 21
- Definition einer Programmiersprache, 5
- deklarative Semantik, 83
- Dekoration, 26
- Dewey-Notation, 14
- dynamisch konstruierter Attributauswerter, 47
- eindeutige kontextfreie Grammatik, 14
- Eindeutigkeit des Rechenergebnisses, 106
- Eingabemenge, 121, 122
- Eingabesymbol, 144
- Eingangszustand, 121
- Eingangszustandsknoten, 132
- Einsatz von Attributgrammatiken, 1, 5
- Einzelschrittrelation eines RKT, 145
- Endzustand, 144
- Epsilon-Produktion, 14, 178
- Erstellung einer Konstruktionsinformation, 122
- Ertrag eines Teilbaums, 130
- erweiterter Abhängigkeitsgraph, 35
- Etikettierung, 121
- Etikettierung eines Knotens, 121
- evaluate*, KW-Auswerter, 124
- expand, 142
- Faktformel, 82
- FirstGoto*, 135
- Fixpunktsemantik, 83
- Folgekonfiguration, 152
- freie Variable, 100
- freies Attributvorkommen, 100
- Front der Blätter, 14
- funktionale Programmierung, 101
- Funktionsgleichungssystem, 101
- Funktionsymbol, 21, 82
- Funktionsweise eines KW-Auswerters, 121
- geordnete Attributgrammatik, 161, 165
- geordnete Partition, 54
- gewünschte Dekoration, 27
- Gleichungsmenge, 102
- gleichzeitiges Parsing und Attributauswertung, 141
- globaler Informationstransport, 1, 25, 26
- Graph, 37
- graphische Notation, 13
- Grundbeweisbaum, 84
- Grundinstanz, 84
- Hasse-Diagramm, 60
- Hauptfunktionssymbol, 102
- Hollerith-Zahlen, 4
- Hornformel, 82
- Identität, 11
- Identitätsoperation, 23
- immediate computation paradigm, 160
- induzierte Reduktionsrelation, 105
- Informationstransport, 1, 91
- inherites Attribut, 4, 22
- initiale Registerbelegung, 144
- Inklusionsdiagramm, 59
- inkrementell ausführbarer Code, 159
- inkrementelle Attributauswertung, 161
- inkrementelle Codeerzeugung, 159
- inkrementeller Algorithmus, 161
- inkrementeller Attributauswerter, 175
- Innen-Argumentposition, 91
- Innenattributvorkommen, 22
- innerer Knoten, 14
- Instanzen, 84
- Instanzen einer Hornformel, 84
- Instanzen eines Literals, 84
- Instanzen eines Terms, 84

- Instanziierung des Schemas für KW-Auswerter, 122  
 Interpretationsfunktion, 21  
*is*-Graph, 35, 37, 161  
*is*-Graph eines Ableitungsbaumes, 35  
*is*-Graph eines Nichtterminalsymbols, 35  
*is-set*, 35, 38  
*is/si*-Graph, 162  
 item, 142  
 iter-OAG-Instruktion, 171  
  
*K*-Rangalphabet, 15  
*K*-sortierte Menge, 14  
 Kantenbeschriftungsfunktion, 132  
 Kellerautomat, 142  
 Kellerbodensymbol, 144  
 Kellerfeld, 142  
 Kellersymbol, 144  
 Kellertransduktor mit Zuweisungen, 144  
 Kennedy-Warren-Algorithmus, 119  
 Kettenregel, 23  
 Klasse der stark nichtzirkulären Attributgrammatiken, 119  
 Klasse von Attributgrammatiken, 58, 59  
 Knoten, 13, 14  
 Knotenbeschriftungsfunktion, 132  
 Kollektion, 35  
 Kollektion der *is*-Graphen, 35  
 Konfiguration eines RKT, 143, 145  
 konfluent, 116  
 konfluente Relation, 11  
 Konstruktion einer Attributgrammatik, 97–99  
 Konstruktion einer semantischen Bedingung, 90, 92  
 Konstruktion einer semantischen Regel, 90, 92  
 Konstruktion eines Logikprogramms, 87  
 Konstruktion eines Plans, 129–130  
 Konstruktion eines RKT, 147–149  
 Konstruktion eines *sg-f* Programms, 111–112  
 Konstruktionsinformation, 122  
 kontextfreie Grammatik, 12  
 kontextsensitive Nebenbedingungen, 159  
 Korrektheitsbeweis, 159  
  
 KW-Algorithmus, 119  
 KW-Algorithmus, erste Phase, 129  
 KW-Algorithmus, zweite Phase, 123  
 KW-Auswerter, 119, 123  
 KW-Auswertungsplan, 122  
 KW-Instruktion, 122  
 KW-Plan, 122  
  
 $\lambda$ -Kalkül, 101  
 language based environment, 159  
 Literal, 81  
 Logik-Programm, 82, 83, 94  
 Logik-Programmierung, 81  
 lokale Attribute, 1  
 lokaler Informationstransport, 1, 25  
 lokales Attributvorkommen, 22  
  
 macro tree transducer, 116  
*MakeEvaluator*, 134  
*MakeInitStates*, 135  
*MakePlan*, 131  
 MapToPlanIndex, 172  
 Menge der Auswertungszustände, 121  
 Menge der Eingabemengen, 122  
 menügesteuerter top-down Entwurf, 159  
 Mitgliedschaftstest, 117  
 Mitgliedschaftstest von *ANC*, 117, 120  
  
 Nachfolgerknoten, 14  
 negatives Literal, 81  
 Nichtdeterminismus, 48  
 nichtdeterministischer Attributauswerter, 48–49  
 Nichtterminalsymbol, 12, 35  
 Nichtzirkularität, 33  
 noethersch, 116  
 noethersche Relation, 11  
 Normalform, 11  
 Notation einer Hornformel, 82  
 Notation, allgemein, 11  
 Notwendigkeit, 31  
  
 OAG-Auswertungsplan, 169  
*OAG-Plan*, 169  
 oben und unten erweiterter Abhängigkeitsgraph, 162  
 ordered attribute grammar, 161, 165



- orientierbares Logik-Programm, 94
- P-evaluate*, 53
- p*-lokales Attribut, 1
- Parsing, 141
- Partition, 54
- pass, 49, 59, 77
- Phase 1 des KW-Algorithmus, 129
- Phase 2 des KW-Algorithmus, 123
- positives Literal, 81
- Prädikatssymbol, 21, 82
- Produktion, 12
- Prozedur *evaluate*, 170
- prozedurale Semantik, 83
- pure, 49
- Pure *k*-pass Attributgrammatik, 53–54
- Pure *k*-sweep Attributgrammatik, 52–53
- Pure *k*-visit Attributgrammatik, 50–51
- Pure multi-pass Attributgrammatik, 53–54
- Pure multi-sweep Attributgrammatik, 52–53
- Pure multi-visit Attributgrammatik, 50–51
- Pure pass Attributauswerter, 53
- Pure sweep Attributauswerter, 52
- Pure visit Attributauswerter, 49
- Rang, 16
- Rangalphabet, 16
- Reduktionsrelation, 101, 104, 105
- Reduktionssemantik, 101, 106
- Registerbelegung, 145
- Registerbeschreibung, 144
- Registerkellertransduktor, 142, 144
- Registername, 144
- Registervorkommen, 144
- Registerwert, 144
- Registerzuweisung, 144
- Rekursionsargument, 101
- Restriktion, 11
- Richtungsetikettierung, 91, 93
- RKT einer unkonditionalen, simple 1-pass Attributgrammatik, 148–149
- Ruhezustand, 121
- Ruhezustandsknoten, 132
- S-evaluate*, 52
- Selektorterm, 93
- Semantik, 83
- Semantikgebung, 83
- semantische Bedingung, 1, 22
- semantische Gleichung, 27
- semantische Regel, 1, 22
- semantischer Bereich, 21
- semantischer Test, 27
- sg-f* Programm, 101
- sichere Richtungsetikettierung, 93
- simple, 49
- Simple *k*-pass Attributgrammatik, 57
- Simple *k*-sweep Attributgrammatik, 55–56
- Simple *k*-sweep Produktion, 72
- Simple *k*-visit Attributgrammatik, 54–55
- Simple multi-pass Attributgrammatik, 57
- Simple multi-sweep Attributgrammatik, 56
- Simple multi-visit Attributgrammatik, 54–55
- Simple pass Attributauswerter, 57
- Simple sweep Auswerter, 56
- Simple visit Attributauswerter, 55
- SIMPLE, Bsp., 102
- simultan rekursiv, 101
- Sorte, 14
- Sortenmenge, 102
- sortierte Menge, 14, 102
- sP-evaluate*, 57
- sprachbasierte Programmierumgebung, 159
- Sprache, 12, 29
- sS-evaluate*, 56
- stark nichtzirkuläre Attributgrammatik, 117–118
- startseparierte Grammatik, 13
- Startsymbol, 12
- statisch konstruierter Attributauswerter, 48
- statischer Attributauswerter, 117
- Stelligkeit, *siehe* Rang
- Striktheit der Inklusionen, 61–69
- string-to-value Übersetzung, 29, 70, 141, 147

- Struktureditor, 159  
Substitution, 84  
*sV-evaluate*, 55  
sweep, 49, 59, 77  
syntaxgesteuerte Übersetzung, 108  
syntaxgesteuerter Editor, 160  
syntaxgesteuertes funktionales Programm,  
101, 102  
synthetisches Attribut, 4, 22
- Terminalsymbol, 12  
terminierende Relation, 11  
Terminterpretation, 86  
top-down Parsing, 141  
totale Ordnung, 162  
Transitionsfunktion, 144  
transitive Hülle, 11  
tree-traversal, 47
- Übersetzung, 9, 29, 86, 106  
Übersetzung eines RKT, 146  
Übersetzung eines syntaxgesteuerten funk-  
tionalen Programms, 106  
Übersetzung in ein *sg-f* Programm, 108  
Übersetzung in Logik-Programm, 86  
Übersicht über Attributauswerter, 49  
unkonditionale Attributgrammatik, 23, 101  
Unvergleichbarkeit von Klassen, 61–69
- V-evaluate*, 50  
Variable, 102  
Variablensubstitution, 84  
Vergleich von Klassen, 59  
Verzahnung von Parsing und Attributaus-  
wertung, 141  
visit, 49, 59, 77  
Vorgängerknoten, 14
- Wertmenge, 21  
Wurzel, 13, 14
- X*-Ableitungsbaum, 13
- Z*-Ableitungsbaum, 13  
Zielformel, 82  
Zirkularität, 30, 32, 36  
Zirkularitätstest, 37–41  
Zirkularitätstest für geordnete Attribut-  
grammatiken, 162  
Zirkularitätstest für stark nichtzirkuläre  
Attributgrammatiken, 117  
zulässige Dekoration, 27–28, 30

# Bezeichnungsindex

- 1V-AG, 59
- A, 22
- $\mathcal{A}$ , 142, 145
- $A(p)$ , 22
- $A(t)$ , 26
- $A(x)$ , 26
- $\mathcal{A}_0$ , 144
- abstract*( $t$ ), 17
- AG, 41, 48
- AG(*p1-pass*), 60
- AG(*p1-sweep*), 60
- AG(*p1-visit*), 60
- AG(*pm-pass*), 60
- AG(*pm-sweep*), 60
- AG(*pm-visit*), 60
- AG(*s1-pass*), 60
- AG(*s1-sweep*), 60
- AG(*s1-visit*), 60
- AG(*sm-pass*), 60
- AG(*sm-sweep*), 60
- AG(*sm-visit*), 60
- AG(*xy-z*), 58
- $\alpha_0$ , 22
- ANC, 117
- Argpos, 91
- Ass, 144
- Att, 22
- ausen*( $p$ ), 22
- AZS, 121
  
- B, 22, 144
  
- C, 22
- $\mathcal{C}$ , 70
- $C(p)$ , 22
- card*( $M$ ), 11
  
- D, 21, 144
  
- $D(p)$ , 31
- $D(p)[D_0; D_1, \dots, D_n]$ , 162
- $D(p)[D_1, \dots, D_n]$ , 35
- $\delta$ , 144
  
- E, 31, 132
- $\varepsilon$ , 12
- evaluate*, 170
- expand**, 148
  
- F, 82
- $\mathcal{F}$ , 102
- $f$ , 12
- $f[m/m']$ , 12
- flag*, 121, 169
  
- G, 21
- $G_0$ , 1, 12
- $\Rightarrow_{G_0}$ , 12
- $\Gamma$ , 144
- Gl, 102
- goto*, 122, 169
  
- H, 82
- $\Rightarrow_H$ , 105
  
- I, 22
- I-Att, 22
- id*, 23
- innen*( $p$ ), 22
- inodet*( $t$ ), 13
- inp*, 121
- is*( $t$ ), 35
- is*( $X$ ), 118
- is-set*( $X$ ), 35
- is/si*( $X$ ), 162
  
- K, 14, 21
- $K_G$ , 122

- $\kappa$ , 14, 21  
*Kom*, 145  
 KW, 119  
  
*L*, 12, 81  
*L(G)*, 29  
*L-AG*, 59  
 $\underline{\text{label}}_t(x)$ , 13  
*LR(0)*-item, 142  
  
 [m], 11  
*max(M)*, 11  
*min(M)*, 11  
  
*N*, 12, 21  
*IN*, 11  
*IN+*, 11  
*node(t)*, 13  
  
*OAG*, 161, 165  
 $\Omega$ , 14, 21  
  
*P*, 12, 21  
*P(M)*, 11  
*P<sub>AA</sub>(G)*, 123  
*P<sub>f</sub>(M)*, 11  
*P<sub>AA</sub>*, 47  
 $\Phi$ , 21, 82  
 $\varphi$ , 16, 21, 84  
 $\Pi(x)$ , 71  
*PL*, 159  
*plan*, 122, 169  
 $\underline{\text{prod}}_t(x)$ , 14  
*profit*, 130  
 $\Psi$ , 21, 82  
  
*q*, 22  
*Q<sub>f</sub>*, 144  
  
*R*, 22  
*R(p)*, 22  
*rank*, 16  
*rank( $\sigma$ )*, 16  
*re*, 91  
**read**, 148  
**ready**, 149  
*Reg*, 144  
 RKT, 144  
  
*root(t)*, 13  
  
*S*, 22  
*S-Att*, 22  
*sel-s*, 93  
 $\Sigma$ , 12, 21, 144  
 $\sigma$ , 16  
*sort*, 14  
*SReg*, 144  
**start**, 148  
  
*T*, 15, 162  
*t*, 13  
*t(x)*, 13  
 $T_{\beta}^Z$ , 106  
 $T_{\Omega}$ , 15  
 $T_{\Omega}(V)$ , 15  
 $T_{\Phi}(V)$ , 81  
 $T_X$ , 162  
 $T_{sv}$ , 70, 71  
 $\tau(A)$ , 146  
 $\tau(G)$ , 29  
 $\tau_{sg}$ , 109  
 $\tau_{sv}(G)$ , 29, 70  
  
*U*, 82  
  
*V*, 31, 132  
*VAL*, 145  
 $\underline{\text{val}}_{in}$ , 144  
 $\underline{\text{val}}_t$ , 26  
 $v_{p,j}$ , 54  
  
*W*, 22, 144  
  
 $\sim_X$ , 35  
 $\rightarrow_X$ , 35  
*x*. - 1, 14  
*x.0*, 14  
*x.j*, 14  
*xVar*, 102  
  
*yout*, 144  
*yield(t)*, 14  
*yVar*, 102  
  
*Z*, 12, 21

- 
- $\rightarrow$ , 11
  - $\rightarrow_X$ , 35
  - $\rightarrow [M']$ , 11
  - $\rightarrow^*$ , 11
  - $\rightarrow^+$ , 11
  - $\rightarrow^0$ , 11
  - $\rightarrow^{n+1}$ , 11
  - $\Rightarrow$ , 12
  - $\Rightarrow_{G_0}$ , 12
  - $\Rightarrow_H$ , 105
  - $\vdash_{\mathcal{A}}$ , 145
  - $\#$ , 144

# Bücher aus dem Umfeld

## **Programmieren mit Fortran 90**

Eine umfassende Einführung für Studenten und Praktiker

von Hans-Peter Bäumer

1997. X, 484 S. mit 2 Disketten. Kart.

ISBN 3-528-05208-2

*Aus dem Inhalt:* Sprachelemente - Zeichenverarbeitung - Festpunktzahlen, Gleitpunktzahlen, Kontrollstrukturen - Felder - Unterprogramme und Module - Abgeleitete Datentypen, Datenstrukturen und anwenderdefinierte Operatoren

Dieses praxisorientierte Lehrbuch richtet sich sowohl an den Anfänger als auch an den fortgeschrittenen Programmierer, der mit dem aktuellen Standard Fortran 90 erfolgreich Software entwickeln möchte. Es ist zum einen ein kompaktes, aus langjähriger didaktischer Erfahrung entstandenes Lehrbuch, zum anderen läßt es sich für den fortgeschrittenen Anwender als detailliertes Handbuch verwenden. Einführende Kapitel zu grundlegenden Sprachelementen, zur FORTRAN Ein- und Ausgabe und zur Zeichenverarbeitung führen den Leser zum ersten Schwerpunkt, der Arithmetik mit Festpunkt- und Gleitpunktzahlen. Datenstrukturen und das Programmieren in geeigneten Programmbausteinen werden eingehend behandelt. Desweiteren ist dem Werk ein Anhang mit der Beschreibung der 113 Standardunterprogramme beigelegt. Textaufgaben und Programmieraufgaben zu den einzelnen Kapiteln helfen dem Leser, sowohl seine neu erworbenen Kenntnisse als auch seine Programmierfähigkeiten zu kontrollieren. Alle im Text verwandten Programmbeispiele sind auf zwei Disketten dem Buch beigelegt.