# ALSI - EINE HÖHERE PROGRAMMIERSPRACHE ZUR TRANSFORMATION VON ALGOL 68-VERBUNDEN IN SIMULA-KLASSEN

U. KASTENS

## 1. Einführung

Die vergleichende Betrachtung von ALGOL 68 und SIMULA zeigt, daß diese Programmiersprachen in einigen grundlegenden Sprachelementen wie den Datenstrukturen verwandte Eigenschaften haben. Es liegt daher nahe, die Möglichkeiten der Transformation zwischen diesen Sprachen zu untersuchen. Das Resultat solcher Bemühungen ist die Definition der höheren Programmiersprache ALSI (ALGOL 68, SIMULA)[Ka73], die einen großen Teil von ALGOL 68 mit dem Verbundkonzept als Kern enthält und die Bedingung der Obersetzbarkeit in SIMULA erfüllt.

Im Folgenden sollen zunächst ALGOL 68- und SIMULA-Datenstrukturen vergleichend gegenübergestellt werden. Dabei beziehe ich mich noch auf die ursprüngliche Definition von ALGOL 68 in [Wi69] nicht auf die 1973 revidierte Fassung.

## 2. Gegenüberstellung der Datenstrukturen

Die Definition der elementaren Datenobjekte stimmt in beiden Programmiersprachen nahezu überein. Das Konzept der zusammengesetzten Objekte ist in ALGOL 68 "orthogonal" in die übrigen Spracheigenschaften integriert, während bei SIMULA das Klassenkonzept historisch aus den Problemen der Simulation diskreter Systeme entwickelt und auf ALGOL 60 aufgepflanzt wurde.

In ALGOL 68 werden Objekte nach ihrer "Art" (mode) unterschieden, die im Wesentlichen festlegt, welche Operationen auf ihnen ausführbar sind. Wir wenden im Folgenden diese Terminologie auch auf SIMULA an.

### 2.1 Verbunde

Unter Verbunden verstehen wir zusammengesetzte Objekte, auf deren Komponenten direkt durch Angabe einer Komponentenbezeichnung zugegriffen werden kann. In ALGOL 68 unter-

liegt die Art der Komponenten keinen speziellen Einschränkungen, mit der Ausnahme, daß ein Verbundobjekt kein Objekt von gleicher Art enthalten kann. Die Art eines Verbundobjektes wird bestimmt durch die Komponentenarten und -bezeichnungen und die Anzahl und Reihenfolge der Komponenten.

Das folgende Programmstück zeigt die Vereinbarung eines Verbundobjekts und Zuweisungen an dessen Komponenten

```
struct (real radius, struct (real x,y) mittelpunkt) k:= (1,(0,0));
radius of k:= 1.5;
mittelpunkt of k:= (1.0, 2.0)
```

In ALGOL 68 kann durch Artvereinbarung für Verbundarten (wie auch für andere Arten) eine Bezeichnung eingeführt werden.

Die Vereinbarungen

```
struct kreis = (real radius, punkt mittelpunkt);
struct punkt = (real x,y);
kreis k:=(1,(0,0))
```

haben zusammen die gleiche Bedeutung wie die Vereinbarung im obigen Beispiel.

Diese Artbezeichnungen sind nur als Kurzschreibweise für die vollständige Artangabe zu verstehen; sie sind kein unterscheidendes Merkmal der Art. Es ist deshalb möglich, daß Arten, die verschieden angegeben werden, gleich sind:

```
struct m1 = (real k, ref m2 e)
struct m2 = (real k, ref m1 e)
```

Der Obersetzer muß solche, und auch komplexere Fälle erkennen (Artidentifizierungsalgorithmen von Koster [Ko69] und Zosel [Zo71]). Zusammengesetzte Objekte werden extern durch Angabe einer Liste der Komponenten notiert. Da weder die Art noch Komponentenbezeichnungen in dieser Notation angegeben werden, ist es Aufgabe des Obersetzers, aus dem Kontext zu ermitteln, von welcher Art das Objekt ist.

In SIMULA ist die Wiedergabe von ALGOL 68-Verbunden möglich durch entsprechende Spezialisierung des mächtigeren Klassenkonzeptes.

Das oben betrachtete Programmstück kann etwa so formuliert werden:

```
class kreis (radius, mittelpunkt);
   real radius; ref (punkt) mittelpunkt;;
class punkt (x,y); real x,y;;

ref (kreis) k;
   k:-new kreis(1, new punkt (0,0));
   k.radius:=1.5;
   k.mittelpunkt.x:=1.0;
   k.mittelpunkt.y:=2.0;
```

Enthält ein Verbund einen anderen Verbund als Komponente, so ist das in SIMULA durch einen entsprechenden Verweis auszudrücken. Dabei wird ausgenutzt, daß das "räumliche Enthalten" und das "Verweisen" gleichwertige Implementierungen desselben Prinzips sind. Außer dem Generierungsoperator gibt es in SIMULA keine Operation, die ein Klassen-Objekt als Einheit behandelt. Das Zuweisen von Verbundwerten erfolgt deshalb komponentenweise. Da die explizite Vereinbarung von Klassen mit Zuordnung einer Klassenbezeichnung vorgeschrieben ist, existiert das Problem der Artidentifikation hier nicht.

Die Anwendungsmöglichkeiten von Klassen gehen über die Realisierung von Verbundobjekten hinaus, denn innerhalb einer Klassenvereinbarung können Anweisungen angegeben und Prozeduren vereinbart werden, die dann selbst Verbundkomponenten sind. Mit diesen Hilfsmitteln kann z.B. eine einfache Kellerorganisation folgendermaßen implementiert werden:

```
class keller (laenge); integer laenge;
begin array a (1:laenge);
      integer pegel; boolean voll, leer;
      real procedure pop;
      begin pop:=a(pegel);
            pegel:=pegel-1;
            leer:=pegel=0
      end;
      procedure push (x); real x;
      begin pegel:=pegel+1;
            a (pegel):=x;
            voll:=pegel=laenge
      end;
      pegel:=0; voll:=false; leer:=true
end
```

Ein solches Objekt wird z.B. durch die Anweisung

```
s:-new keller (100)
```

generiert. Zugriffe auf Elemente dieses Kellerobjektes können dann z.B. so formuliert werden

```
            if not s.voll then s.push (3.4)
  oder      if not s.leer then y:=s.pop
```

Die Methode, mit der der Keller implementiert wurde, braucht außerhalb der Klassenvereinbarung nicht zur Kenntnis genommen zu werden - es hätte ebenso gut eine lineare Liste statt einer Reihung verwendet werden können (vergleiche Parnas [Pa72]).

Nach dem gleichen Prinzip werden in SIMULA die Standardklassen SIMULATION und SIMSET definiert, die Datenstrukturen und Operationen zur Simulation bzw. Listenverarbeitung bereitstellen.

Anwendungen dieser Art legen es nahe, eine Klassenvereinbarung aufzufassen als die Definition einer Datenstruktur zusammen mit den auf ihr ausführbaren Operationen in geschlossener Form. In ALGOL 68 ist ein analoges Konzept nicht vorhanden. Der Aufbau einer ähnlichen Konstruktion macht dort erhebliche formulierungstechnische Schwierigkeiten. Verbunde können zwar Prozedurvariablen als Komponenten enthalten; die Prozeduren müssen aber jeweils bei der Generierung eines solchen Verbundes von außen zugewiesen werden.

Im letzten Beispiel haben wir den Anweisungsteil der Klassenvereinbarung nur zur Initialisierung der Komponenten verwendet. Durch geeigneten Aufbau des Anweisungsteiles kann eine Klasse eine oder mehrere gleichartige Koroutinen definieren; z.B. das folgende vereinfachte Produzent-Konsument-System:

```
class produzent
begin initialisiere ;
      detach;
      while bedingung do
      begin produziere;
            resume (k)
      end
end
```

```
class konsument
begin initialisiere;
      detach;
      while bedingung do
      begin verbrauche;
            resume (p)
      end
end
```

Eine weitere Anwedungsmöglichkeit von SIMULA-Klassen liegt in der Erzeugung partiell parametrisierter Prozeduren. Die im folgenden Beispiel definierte Klasse erzeugt abhängig von den Parametern entsprechende Polynomfunktionen, die z.B. in einer Integrationsprozedur ausgewertet werden:

```
class polynom (grad, koeffizieten);
      integer grad; array koeffizienten;
begin real procedure polynomwert (x); value x; real x;
      begin real s; integer i;
            s:=koeffizienten (grad);
            for i:=grad-1 step-1 until 0 do
            s:=s*x+koeffizienten (i);
            polynomwert:=s
      end
end
```

Ein Aufruf der Integrationsprozedur könnte dann z.B. lauten

```
integral (new polynom (5, koeff ).polynomwert, 0.5, 2.1)
```

Derartige Konstruktionen sind in SIMULA möglich, da alle Objekte unbeschränkte Lebensdauer haben, während in ALGOL 68 als Kellersprache solche partiell parameterisierbaren Prozeduren nicht formuliert werden können.

## 2.2 Vereinigungsarten

Für eine Reihe von Anwendungen ist es erforderlich, Variable zu haben, die Objekte von unterschiedlicher Art als Inhalte besitzen können. In ALGOL 68 wird die Art einer solchen Variablen ausgedrückt als ein Bezug auf die Vereinigung der Arten, die die möglichen Werte annehmen können, z.B.

```
ref union (real, int, ref formel)
```

In der ursprünglichen Definition von ALGOL 68 sind die über den einzelnen Arten definierten Operatoren nicht auf Objekte der union-Art anwendbar, vielmehr muß man erst mit einer speziellen Zuweisung ("conformity relation") das Objekt an eine Variable der betreffenden Teilart zuweisen.

In SIMULA wird die Vereinigung verschiedener Arten auf Klassen beschränkt. Mit Hilfe von Präfixen in den Klassenvereinbarungen können Hierarchien von Klassen definiert werden. Wollen wir z.B. eine Listenstruktur aufbauen aus Elementen, die unterschiedliche Komponenten haben wie die Klassen

```
class listelem1 (nachf, k); ref (listelem ) nachf; integer k;;
class listelem2 (nachf, c); ref (listelem ) nachf; character c;
```

so können wir die beiden Klassen zu einer "Oberklasse" zusammenfassen:

```
            class listelem (nachf); ref (listelem) nachf;;
listelem    class listelem1(k); integer k;;
listelem    class listelem2(c); character c;;
```

Die in beiden Unterklassen gleiche Komponente (nachf) kann in der Oberklasse definiert werden. Der Zugriff auf Komponenten, die in der Unterklasse definiert sind erfordert, daß entweder die Klassenbezeichnung als"Qualifikation"angegeben oder das Objekt - wie in ALGOL 68 - an eine geeignete Variable zugewiesen wird.

```
l qua listelem1. k oder
l1:-l; l1.k
```

Nach dem gleichen Schema ist eine Hierarchie von Standard-Klassen zur Organisation der Ein- und Ausgabe definiert. Jedem Datei-Typ ist eine Klasse zugeordnet, die die jeweiligen Bearbeitungsprozeduren als Komponenten enthält:

```
            class file ...
       file class directfile ...
       file class infile ...
       file class outfile ...
    outfile class printfile ...
    outfile class punchfile ...
```

Die Artvereinigung aus ALGOL 68 kann in SIMULA durch die Definition einer Oberklasse ohne Komponenten dargestellt werden:

```
union m = (m1, m2)
```

entspricht den Klassenvereinbarungen

```
class m;;
m class m1....
m class m2....
```

## 2.3 Reihungen

In SIMULA wurde das Array-Konzept unverändert aus ALGOL 60 übernommen. ALGOL 68-Reihungen weichen davon vor allem nur in den folgenden Punkten ab:

- Der Deskriptor einer Reihung ist auf der Programmebene insoweit zugänglich, daß die Indexgrenzen abgefragt und im Falle von flexiblen Grenzen auch verändert werden können.
- Durch Erzeugen eines veränderten Deskriptors können die Indexgrenzen "getrimmt" bzw. Reihungsausschnitte definiert werden.
- Operationen mit Reihungen als Einheiten, insbesondere Zuweisungen sind möglich.

- Reihungen können explizit als Folgen von Elementen angegeben werden, z.B.

```
[1:4] int a = (1,2,3,4)
[1:2, 1:3] int b = ((1,2,3),(4,5,6))
```

## 2.4 Namen

In der Algol 68-Terminologie werden unter "Namen" Verweise auf Bezugsobjekte im Sinne von Variablenadressen verstanden. Das Namenskonzept basiert auf einer strengen Hierarchie: Mit der Referenzstufe eines Namens ist die Anzahl der indirekten Adressierungsschritte festgelegt, die nötig ist um ein Bezugsobjekt zu erreichen, das kein Name ist. Die Referenzstufe eines Namens und die Art des Bezugsobjektes bestimmen die Art des Namens.

Allen Objekten - auch Namen und Konstanten - können durch Vereinbarungen Bezeichnungen zugeordnet werden. Namen von Komponenten strukturierter Objekte sind ebenfalls durch Bezeichnungen identifizierbar, z.B.:

```
ref int i = k of v
ref int j = a [4]
```

Die Möglichkeit, Namen zu definieren, die auf Namen verweisen, wirft das Problem der "Verschleppung" von Namen auf, deren Bezugsobjekte nicht mehr existieren. Das folgende Programmstück ist ein einfaches Beispiel dafür:

```
ref ref int ii  = ref int;
begin ref int k = int:=5;
      ii  := k
end;
print ( ii )
```

In der Zuweisung ii:=k wird der k zugeordnete Name an ii zugewiesen, dessen Gültigkeitsbereich den von k umfaßt. Um das Kellerprinzip aufrecht zu erhalten, muß ein Zugriff auf das nicht mehr existierende Objekt (in print(ii)) verhindert werden. Hierzu wären ineffiziente Laufzeitprüfungen nötig; es wird deshalb die stärkere Forderung gestellt: Der Name eines Objekts darf nicht an einen Namen zugewiesen werden, der eine größere Lebensdauer hat, als das Objekt selbst.

In SIMULA ist kein so strenges Referenzstufen-Konzept definiert wie in ALGOL 68. Für Klassenobjekte existieren Namen der Referenzstufe 1 und Variable dafür. Bei Objekten

einfacher Art wurde das Variablen-Prinzip aus ALGOL 60 übernommen. Wegen dieser unterschiedlichen Definition ist es erforderlich, bei der Transformation alle Objekte durch Klassen zu implementieren, um einheitliche Referenzstufen darzustellen. Das Problem der Verschleppung von Namen existiert in SIMULA nicht, da alle Objekte unbegrenzte Lebensdauer haben.

Der Vergleich auf Identität ist in ALGOL 68 und SIMULA die einzig mögliche Operation mit Namen als Operanden.

## 3. Transformation der Datenstrukturen

Bei der Transformation von ALGOL 68-Datenstrukturen in SIMULA-Klassen ist die Realisierung des Referenzstufenkonzeptes, der Identifizierung von Namen und der Artanpassungsoperationen die wichtigste Aufgabe. Die übrigen Zuordnungen können im allgemeinen durch Textsubstitution vorgenommen werden.

Die Darstellung beliebig langer Verweisketten durch Referenzvariable bereitet keine prinzipiellen Schwierigkeiten, da sie aus mehreren Verweisen kettenartig zusammengesetzt werden können.

Objekte der Referenzstufen 0 und 1 können durch gleichartige Klassenobjekte realisiert werden. Ihre externe Identifizierung erfolgt durch eine Referenzvariable. Benannte Objekte der Referenzstufe 0 werden dann als schreibgeschützte Variable aufgefaßt. Diese Unterscheidung kann vom Übersetzer vorgenommen werden. Auf gleiche Weise ist das Problem zu lösen, daß in SIMULA Verbundkomponenten nicht wiederum Verbundkonstante sein können: Es wird stattdessen eine Referenzvariable als Komponente angegeben, deren Wert ein unveränderlicher Verweis auf ein entsprechendes Objekt ist.

Objekte der Referenzstufe 2 werden dargestellt durch Klassenobjekte, die als einzige Komponente eine Referenzvariable von der entsprechenden Art besitzen.

Damit ergibt sich das folgende Abbildungsschema: Sei m eine beliebige Art. So entsprechen der ALGOL 68-Artvereinbarung

```
mode vb = struct (m ko, ref m k1, ref ref m k2)
```

die Klassenvereinbarungen

```
class vb;
begin ref (m) ko; ref (ref m) k1; ref (ref m) k2;
      >Zuweisungsprozedur<
      >Initialisierung<
end
```

und

```
class refvb;
begin ref (vb) verweis; end
```

Die im Klassenrumpf definierte Zuweisungsprozedur führt die komponentenweise Zuweisung eines Verbundes aus.

Namen von Referenzstufen höher als 2 können nach dem gleichen Prinzip (durch Einführung weiterer Klassen) dargestellt werden. Bei unserer Implementierung haben wir davon abgesehen, da man im allgemeinen mit den Referenzstufen 0,1 und 2 auskommt.

Die in ALGOL 68 definierten Artanpassungsoperationen können weitgehend durch Operationen auf den Klassenobjekten realisiert werden: z.B. entspricht dem "Dereferenzieren" von Referenzstufe 2 auf 1 das Verfolgen der Referenz, die durch die Komponente "verweis" gegeben ist, "Vereinigen"(uniting) wird durch entsprechende Qualifikation des Objektes erreicht, "Löschen" (voiding) durch Zuweisung an geeignete Variable, die "universelle Artanpassung" (hipping) und "Weiten" durch Generierung geeigneter neuer Objekte. In unserer Implementierung haben wir auf die Transformation der Anpassungsoperationen "proceduring" und "rowing" verzichtet, und - wie in SIMULA - gefordert, daß Verbundarten durch eine Artbezeichnung identifiziert werden, um so die Verfahren zu vereinfachen, mit denen die jeweils nötige Artanpassung festgestellt wird.

## 4. Transformation der Programmstrukturen

Eine Reihe von Sprachelementen sind in ALGOL 68 und SIMULA identisch oder können leicht auf entsprechende Eigenschaften abgebildet werden:

z.B. Zuweisungen, Formeln, bedingte Formeln, Fallunterscheidungen (Transformation in SIMULA-Switch-Konstruktion bzw. "if-Kaskaden"), Wiederholungsanweisungen, Sprünge, Generatoren, Vereinbarungen und Prozeduren.

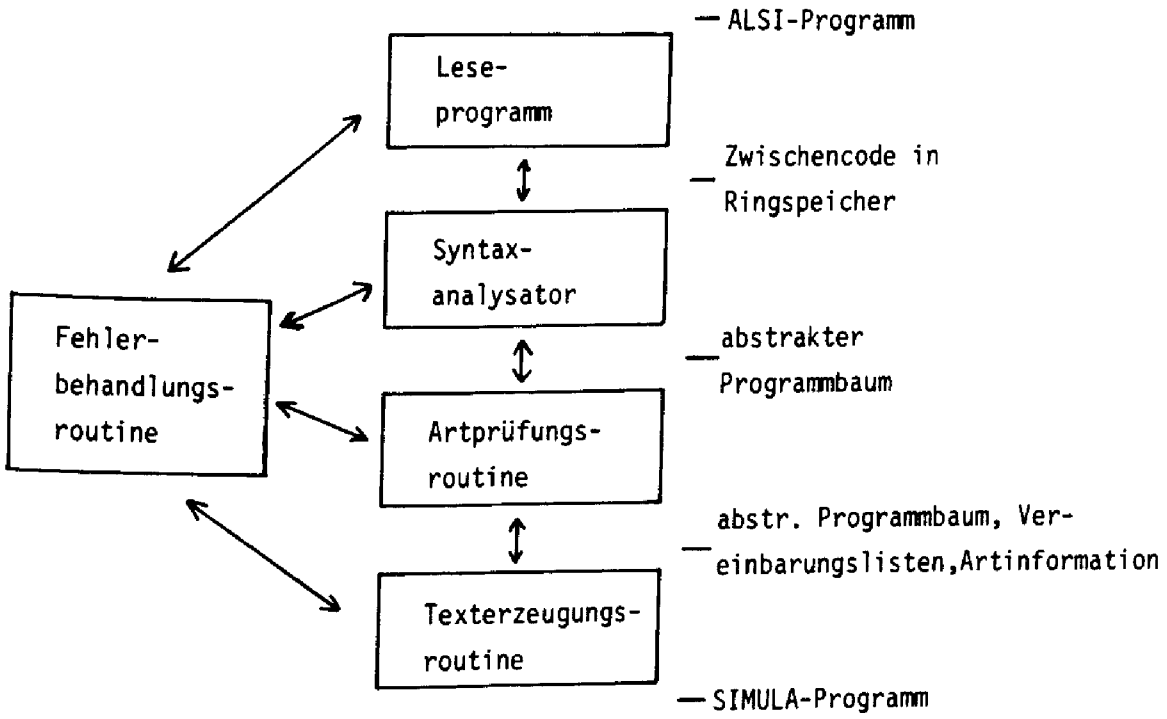
Es ist bemerkenswert, daß die Parameterübergabe aus ALGOL 68 aufgrund der gewählten Transformation der Datenobjekte ohne Schwierigkeiten durch die Parameterübergabe "by-reference" (dem Wertaufruf auf Referenzstufe 1) in SIMULA wiedergegeben werden kann.

Auf die Abbildung von ALGOL 68-Sprachelementen, die in SIMULA nicht in entsprechender Bedeutung existieren wurde verzichtet (so z.B. auf Identität von Ausdruck und Anweisung, Operatordefinitionen, parallele Blöcke, ALGOL 68-E/A und Prozeduren als Datenobjekte). In Prinzip könnten auch diese Spracheigenschaften auf SIMULA transformiert werden, z.B. indem man Prozeduren als Klassen darstellt und dann wie Verbunde als Datenobjekte behandeln kann.

## 5. Aufbau des Übersetzers

Die im Vorangehenden kurz erläuterten Sprachelemente sind zu der in [Ka73] definierten Sprache ALSI zusammengefaßt. Für diese Sprache wurde in SIMULA ein Übersetzer geschrieben, der ALSI-Programme in SIMULA übersetzt.

Er ist in fünf Koroutinen gegliedert, die wie in der folgenden Abbildung dargestellt zusammenwirken:




Das Leseprogramm führt die lexikalische Analyse durch und trägt einen Zwischencode in einen Ringspeicher ein. Der Syntaxanalysator arbeitet im "recursive-descendent"- Verfahren (ohne "backtracking") und baut aus dem Zwischencode einen abstrakten Programmbaum auf. Die Artprüfungsroutine fügt Vereinbarungslisten in den Baum ein, ermittelt die Arten der Objekte und führt die Artanpassung durch.

In der Texterzeugungsroutine werden aus dem abstrakten Programmbaum die SIMULA-Programmtexte aufgebaut. Weil das erzeugte SIMULA-Programm die gleiche Struktur wie das Quellprogramm hat, können die Texte von rekursiven Prozeduren modular aus Teiltexten zusammengesetzt werden. Die Routine zur Fehlerbehandlung kann von allen anderen Routinen aktiviert werden. Sie gibt die Meldungen aus, bestimmt bei syntaktischen Fehlern den Aufsetzpunkt und reaktiviert die entsprechende Syntaxprozedur.

Für das Zusammenwirken der Koroutinen wurde die Strategie verfolgt: Jede Routine ist

solange aktiv bis sie genügend Information beschafft hat, mit der die nächste Routine sinnvoll arbeiten kann. Dadurch wird gewährleistet, daß sich immer nur das notwendige Minimum an Informationen über das Eingabe-Programm im Speicher befindet.

Der Umfang des ALSI-Obersetzer-Programmes und die Obersetzungszeiten liegen in der gleichen Größenordnung wie beim SIMULA-Obersetzer. Die erzeugten SIMULA-Programme haben - abgesehen von den komplexeren Klassenvereinbarungen - etwa das gleiche Ausmaß wie die ALSI-Quellprogramme; sie benötigen wegen der nicht effizienten Darstellung von einfachen Werten und Reihungen durch Klassen mehr Rechenzeit (der erhöhte Speicherbedarf bei der Ausführung schlägt sich wegen der häufiger nötigen Speicherbereinigung auch in der Rechenzeit nieder).

## 6. Literatur

[DMN71] O.J. Dahl, B. Myhrhaug, K. Nygaard: SIMULA 67 COMMON BASE LANGUAGE, Norwegian Computing Center, pub. S-22, 1971

[Ka 73] U. Kastens: ALSI - Eine höhere Programmiersprache zur Transformation von ALGOL 68-Verbunden in SIMULA-Klassen, Universität Karlsruhe, Fakultät für Informatik, Diplomarbeit, März 1973

[Ko 69] C.H.A. Koster: On INfinite Modes, Algol Bulletin AB 30.3.3, Feb. 1969

[Pa 72] D.L. Parnas: On the Criteria To Be Used in Decomposing Systems into Modules, CACM, Dec. 1972

[Ro 73] H.Rohlfing: SIMULA - Eine Einführung, Bibliographisches Institut, Hochschultaschenbücher, 1973

[Wi 69] A. van Wijngaarden (Editor), B.J. Mailloux, J.E.L. Peck, C.H.A. Koster: Report on the Algorithmic Language ALGOL 68, Numerische Mathematik, 14, 79-218 (1969),Springer-Verlag

[Zo 71] M.E. Zosel: A Formal Grammar for the Representation of Modes and its Application to ALGOL 68, University of Washington, Computer Science Group, Dissertation, 1971

Anschrift des Verfassers: Dipl.-Inform. Uwe Kastens, Universität Karlsruhe, Fakultät für Informatik, D-7500 Karlsruhe, Zirkel 2