T. Mossakowski and H.-J. Kreowski (Eds.): WADT 2010,
LNCS 7137, pp. 13--24. Springer, Heidelberg (2012)

My ADT Shrine

Hans-Dieter Ehrich

Institut für Informationssysteme
Technische Universität Braunschweig
D-38092 Braunschweig
HD.Ehrich@tu-bs.de

Abstract. The 20th WADT 2010 is put into perspective by giving afterglows of the 1st WADT 1982 in Langscheid near Dortmund, and the 10th WADT 1994 in Santa Margherita near Genova. First encounters with pioneers in the field are recalled, in particular with the ADJ group who initiated the initial-algebra approach. The author's contributions at that time are put in this context. Around 1982, the emphasis of his work moved to databases and information systems, in particular conceptual modeling. His group used a triple of layers to model information systems, data—objects—systems, where the focus of interest now was on objects and systems. The interest in data issues paled in comparison. There were cases, however, where benefits could be drawn from the early work on ADTs and the foundations established in this field.

1 Opening the Shrine

In March 1982, Udo Lipeck and the author organized a workshop on Algebraic Specification in Langscheid, Sorpesee, near Dortmund. It took place in the *Heimvolkshochschule Sorpesee*, an institution of adult education. Figure 1 shows its logo at that time.

There were 29 participants coming from 9 universities, 8 German and 1 Dutch. Table 1 shows the list grouped by universities and ordered by alphabet.

As for the Dortmund group: at the time of the workshop, Volker Lohberger had left for Essen, Gregor Engels for Osnabrück, and Udo Pletat for Stuttgart. Klaus Drosten, the author, Martin Gogolla, and Udo Lipeck were about to leave for Braunschweig. Many of the other participants moved as well more or less shortly after the workshop. So the group felt that the event should be repeated in order to remain in contact.

This way, the Langscheid workshop became the 1st WADT. It was followed by WADTs in Passau (2nd:1983), Bremen (3rd:1984), Warberg Castle near Braunschweig (4th:1986), Gullane near Edinburgh (5th:1987), Berlin (6th:1988), Wusterhausen near Berlin (7th 1990), Dourdan near Paris (8th:1991), and Caldes de Malavella near Barcelona (9th:1992).

The 10*th* WADT was held from May 30 to June 6, 1994, in Santa Margherita Ligure near Genova, together with the 5th COMPASS Workshop. Figure 2 gives a view of the beautiful location (left) and the organizer, Egidio Astesiano (right), on an excursion to Genova.

T. Mossakowski and H.-J. Kreowski (Eds.): WADT 2010, LNCS 7137, pp. 13–24, 2012.

Fig. 1. Site of the 1^{st} WADT

Table 1. Participants of the 1st WADT

Aachen	Herbert Klaeren, Heiko Petzsch
Berlin	Hartmut Ehrig, Werner Fey, Horst Hansen, Klaus-Peter Hasler, Hans-Jörg Kreowski, Michael Löwe, Peter Padawitz, Michaela Reisin
Bonn	Christoph Beierle, Peter Raulefs, Angelika Voß
Bremen	Herbert Weber
Dortmund	Klaus Drosten, Hans-Dieter Ehrich, Gregor Engels, Martin Gogolla, Udo Lipeck, Volker Lohberger, Udo Pletat, Axel Poigné
Karlsruhe	Heinrich C. Mayr
Leiden	Jan Bergstra
München	Harald Ganzinger, Peter Pepper, Martin Wirsing
Saarbrücken	Claus-Werner Lermen, Jacques Loeckx

90 participants were registered at the workshop, the author refrains from listing them. The program offered 62 presentations. Selected papers were published after the conference in the LNCS 906 volume entitled *Recent Trends in Data Type Specification.* It was edited by Egidio Astesiano, Gianna Reggio and Andrzej Tarlecki, and it was published in 1995 after the conference.

The author's contribution to this volume, coauthored by Amilcar Sernadas, was a paper entitled *Local Specification of Distributed Families of Sequential Objects* [12]. So he was away from the ADT field by then—in fact, already for nearly ten years.

Fig. 2. Left: Santa Margherita Ligure, the site of the 10^{th} WADT; Right: Egidio Astesiano speaking to participants on an excursion to Genova

It should be mentioned that the 10^{th}WADT/5^{th}COMPASS Workshop was the starting point of the "Common Framework Initiative for algebraic specification and development" (CoFI, the homepage is [5]) to unify and standardize the algebraic specification languages that were around at that time. There were quite a few. At least the main concepts to be incorporated were thought to be clear—although it was realized that it might not be so easy to agree on a common language to express these concepts. And so it was. The result of the efforts are published in two LNCS volumes [1,22].

Actually, WADT proceedings were continuously published in the Springer LNCS series from the 1987 Gullane meeting on, with a precursor in 1984 when the Bremen WADT proceedings were published in the Springer *Informatik Fachberichte*[1].

The author was very pleased when he was invited to give a retrospect lecture at the 20^{th} anniversary WADT in Schloss Etelsen near Bremen. Figure 3 gives a view of that beautiful place. The author hadn't attended any of the WADT's since 1995, the 11^{th} one, in Oslo.

For his presentation, the author was very generously given one full hour. But still, there was a need to concentrate on what was felt most essential and what could be of interest for the audience. So the author had to select and to simplify more than he had wished to. Also, his selection of topics was quite personal, guided by the material that he had readily available or could easily retrieve, and it was certainly also guided by vanity. So the selection is neither complete nor fairly balanced.

Since this paper is an elaboration of that presentation, it naturally suffers from the same problems: oversimplification, overselectivity, vanity bias. The author is aware that much more work along the ideas outlined here was going on at that time, all over the world. Historical completeness and balance is not among his intentions, though.

[1] Informatics Technical Reports.

Fig. 3. Schloss Etelsen near Bremen, site of the 20^{th} WADT

2 Early Treasures

2.1 ADT Roots

It was in the mid to late nineteen hundred and seventies when the author became aware of the seminal ADJ[2] papers [20] and [19].

In [20], ADJ address programming language semantics, not abstract data types yet: the syntax of a programming language $\mathcal{L}$ is represented as an algebraic signature $L = (S, \Sigma)$ where S is a set of sorts and $\Sigma = \{\Sigma_{x,s}\}_{x \in S^*, s \in S}$ is an $S^* \times S$-sorted operator signature, i.e., a collection of operators $\omega \in \Sigma_{x,s}$ also written $\omega : s_1 \times \ldots \times s_n \to s$, where $x = s_1 \ldots s_n \in S^*$ and $s \in S$.[3] Interpretation is given in the category $\mathbf{Alg}^c_\Sigma$ of continuous Σ-algebras where fixpoint equations can be solved. This category has initial algebras which form an isomorphism class, with the term algebra T_Σ as a natural representative. T_Σ may be seen as the abstract syntax of the programming language. "Abstract" here means concentrating on the structure of the syntax and disregarding the concrete symbols written by the programmer. Mathematical semantics is given implicitly

[2] The acronym ADJ denoted a group of authors consisting of varying subsets of {Jim Thatcher, Eric Wagner, Jesse Wright, Joseph Goguen}; the group aimed at establishing an adjunction between category theory and computer science.

[3] Throughout this paper, a coherent notation is used which may deviate from the notations in the papers referred to.

by writing equations over Σ that are supposed to hold. A set of Σ-equations E determines the subcategory $\mathbf{Alg}^{c}_{\Sigma,E}$ of algebras satisfying these equations. This subcategory also has initial algebras, with a natural representative $\mathrm{T}_{\Sigma,E}$, the quotient term algebra $T_\Sigma/\sim_E$ where $\sim_E$ is the congruence relation induced by E. By initiality, there is a unique morphism $\mu : T_\Sigma \to T_{\Sigma,E}$. This initial morphism gives meaning to each syntactic construct by interpreting it in the semantic algebra. This way, it establishes the semantics $[\![\mathcal{L}]\!]$ of the programming language $\mathcal{L}$. This approach has been termed *algebraic* semantics[4].

In [19], ADJ carried this elegant approach over to the specification of abstract data types, concentrating on finitary algebras. The essentials are well-known.

- a *data type* is an algebra
- an *abstract* data type is an isomorphism class of algebras
- the *syntax* of an abstract data type is given by a *specification* $D = (S, \Sigma, E)$ where S is a set of sorts, Σ is an $S^* \times S$-sorted set family of operators, and E is a set of Σ-equations
- the semantics of such a specification is defined in two steps
  1. the *Σ-equations* E determine a category of Σ-algebras satisfying these axioms
  2. the isomorphism class of *initial* (Σ, E)*-algebras* (or any representative in it) is suggested as *the* abstract data type specified "up to isomorphism" by the equational specification D.
- under natural conditions, there is an *operational* semantics in the form of a term rewriting system that operates precisely in an initial algebra (the term normal form algebra).

This is the basis. There are extensions and ramifications in many respects.

Fig. 4. Left: IBM Research Yorktown Heights in 1981; right, from left: Jim Thatcher, Eric Wagner, NN, the author, Wolfgang Wechler at MFCS'81 in Štrbské Pleso in the High Tatras (then Czechoslovakia, now Slowakia)

[4] As opposed to denotational, operational, and axiomatic sematics.

Figure 4 shows photographs of early encounters with members of the ADJ group. Their ideas spread over to Europe, Germany in particular, and gave rise to the 1982 Langscheid workshop, to become the 1^{st} WADT, in order to give the emerging discussions about abstract data types a forum.

Early cooperation of ADJ with the Berlin group is witnessed by [14]. ADJ's influence on the author's own work is best shown in what he considers to be a special treasure in his shrine, namely [6]—not the first but probably the most important of his early publications in prestigious journals.

In this paper, the concepts of implementation and parameterization of abstract data types are explored on the syntactic level of specifications $D = (S, \Sigma, E)$. A *specification morphism* $f : D_1 \to D_2$ is a pair (f_S, f_Σ) where $f_S : S_1 \to S_2$ is a map, and $f_\Sigma : \Sigma_1 \to \Sigma_2$ is an $S^* \times S$-indexed set family of maps $\{f_{\Sigma,x,s} : \Sigma_{1,x,s} \to \Sigma_{2,x,s}\}_{x \in S^*, s \in S}$ such that, for every equation $l = r$ in E_1, $f(l) = f(r)$ is in E_2^*, the closure of E_2. We trust that the reader has an idea how f is applied to the left and right hand side terms of an equation (to every term operator recursively), and what the "closure" of an equation system is[5].

Referring to Figure 5, specification D_1 *implements* specification D_0 if there is a specification D_2 with two specification morphisms $f : D_1 \to D_2$ and $t : D_0 \to D_2$, such that D_0 is embedded "truely" into D_2 wrt t, and D_1 is embedded "fully" into D_2 wrt f. As for the precise definitions and their variants, we refer to the paper.

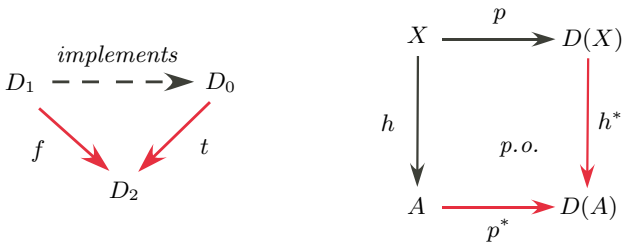

Fig. 5. Implementation (left) and parameterization (right)

A parameterized abstract data type $p : X \hookrightarrow D$ is given by an inclusion morphism p embedding a formal parameter part X into a specification D, written $D(X)$. A parameter assignment is given by a morphism $h : X \to A$ where A is an actual parameter specification. h says which formal parameter sorts and operators are to be replaced by which actual ones, while equations in X act as constraints that have to be obeyed by the actual parameter A. The result of substituting X by A is given by a pushout in the category of specifications and specification morphisms: the specification $D(A)$ where X is replaced by A, with p^* showing the embedding of the actual parameter in the result, and h^* showing the extension of h to the entire specification.

The author's approach to parameter replacement was an adaptation and simplification of Ehrig, Pfender and Schneider's categorial treatment of graph

[5] The expert reader may wonder what is meant here: the deductive closure or the entailment closure. It does not matter, they coincide in equational theories.

replacements [15]. (Co)limits as a mathematical tool for describing system integration had been used at about the same time by Joseph Goguen [17]. The semantics of CLEAR [4] makes extensive use of colimit constructions, also pushouts for handling parameter replacement, albeit on the level of semantic algebras.

There were also other approaches to abstract data type specification, some using categories and some not, some using initial semantics and some not. [2] gives a nice overview of abstract data type models at that time.

2.2 Algebraic Domain Equations

The 1983 paper [11], coauthored by the author and Udo Lipeck, sadly didn't find due attention, but the author always kept a warm remembrance of it. The idea was stimulated by Dana Scott's work on data types as lattices where data types (domains) are characterized implicitly as solutions of domain equations [23]. Was there an analogue in the realm of algebraic data types? What could the term "domain equation" possibly mean here? The authors' basic idea was to remodel a domain equation as a parametric specification $p : X \hookrightarrow D(X)$ with a further morphism $e : X \to D(X)$ with the idea that the targets of p and e had to be merged.

$$X = D(X) \quad \text{vs.} \quad X \overset{p}{\underset{e}{\rightrightarrows}} D(X)$$

In order to give a flavour of the approach, here are a couple of examples. With appropriate definitions of X, $\times$, $+1$, p and e (the details of which can be found in [11]), we have, for example, natural numbers specified by $X \rightrightarrows X + 1$, stacks with data of sort S specified by $X \rightrightarrows X \times S + 1$, binary trees with data of sort S at each node specified by $X \rightrightarrows X \times X \times S + 1$, binary trees with data of sort S at the leaves specified by $X \rightrightarrows X \times X + S$, etc.[6].

In order to give an idea of how it works, we expand the first example, natural numbers, specified by $X \overset{p}{\underset{e}{\rightrightarrows}} X + 1$.[7] X has one sort which we also denote by X, and one constant $x_0 :\to X$. "+1" adds one new sort N, one new constant $n_0 :\to N$, and a "construction" operator $\sigma : X \to N$ which connects the sorts. Let $p : X \to X + 1$ be the embedding sending all items in X to the ones in $X + 1$ with the same denotaton. Let $e : X \to X + 1$ send sort X to sort N and constant x_0 to constant n_0. Then the coequalizer of p and e identifies the two sorts, let us denote the resulting sort by $\mathbb{N}$, and also the two constants, let us denote the resulting constant by 0. If we suggestively rename the operator σ by *succ*, then its signature is $succ : \mathbb{N} \to \mathbb{N}$. So we have the basic specification of the natural numbers with one sort, constant 0 and the successor operator.[8]

[6] However, there are no interesting counterparts of Scott's reflexive domain $X = X^X$ and his powerset domain $X = 2^X$.

[7] This example is a simplification of example 7.1 in [11].

[8] A slightly more complex version of this example adds to "+1" a "projection" operator $\pi : N \to X$ and the (not too far fetched) equation $\pi(\sigma(x)) = x$. In the coequalizer, these then become the predecessor operator $pred : \mathbb{N} \to \mathbb{N}$ and the equation $pred(succ(n)) = n$. There is no equation for $pred(0)$, so this value is left undefined.

The semantics of algebraic domain equations roughly works as follows. As is well-known, each specification morhism $f : D_1 \to D_2$ gives rise to adjoint functors $F : D_1\text{-alg} \to D_2\text{-alg}$, the free functor, and $\bar{F} : D_2\text{-alg} \to D_1\text{-alg}$, the forgetful functor. An Algebraic Domain Equation $X \overset{p}{\underset{e}{\rightrightarrows}} D(X)$ then defines the pair $X\text{-alg} \underset{\bar{E}}{\overset{P}{\rightleftarrows}} D\text{-alg}$ of functors.

Let (c, Q) be the coequalizer of p and e, $X \overset{p}{\underset{e}{\rightrightarrows}} D \xrightarrow{c} Q$. Referring to the functors $X\text{-alg} \underset{\bar{E}}{\overset{P}{\rightleftarrows}} D\text{-alg} \xleftarrow{\bar{C}} Q\text{-alg}$, the *fixpoints* of $\bar{E}P$ are precisely the algebras $A\bar{C}$ for algebras $A \in Q\text{-alg}$. These are the *solutions* of an Algebraic Domain Equation $X \overset{p}{\underset{e}{\rightrightarrows}} D(X)$. Apparently, any initial Q-algebra is an *initial* solution of the Algebraic Domain Equation.

Fascinating work must be mentioned that has been influential in one or the other way, although there is no room for giving detailed references. Smyth's and Plotkin's work on a categorial solution for recursive domain equations [24] was of great interest when working on the above. Burstall's and Goguen's work on the ADT specification language CLEAR [3,4], and Goguen's work on OBJ beginning with [18] were continuing sources of inspiration, also later on when the author worked on database conceptual modeling languages.

Initial semantics was a great idea that guided the author quite a bit in the early ADT days. It soon became apparent, though, that initiality was not enough. In 1984, Martin Gogolla, Klaus Drosten, Udo Lipeck, and the author published a paper [16] supporting all forms of error handling: error introduction, error propagation, and error recovery; there is an initial semantics which coincides with the term-rewriting semantics if the latter is finite Church-Rosser; a specification-correctness criterion allows for non-initial models—loosely speaking, they are "initial on the ok values" and "loose on the error values". This means that carrier sets are partitioned into an ok part and an error part, and the initial morphisms must be injective on the former while there are no constraints for the latter. In specification, operators are classified into those which introduce errors in normal situations and those which preserve ok states. Two types of variables are introduced, one for non-error situations only and the others for ok states and exceptional states.

3 Drifting Away

When moving from Dortmund to Braunschweig where the author took the chair of databases and information systems, conceptual modeling became his major interest. ADTs were largely unknown in this community, and there was disappointment in the beginning that they were of little help. The author expressed that in his contribution to the 3rd WADT 1984 in Bremen [7].

This marked the beginning of the author's drifting away from the ADT area. Rather, *object* concepts caught his interest. Corresponding concepts have been

dealt with in the ADT community as well, probably first by Joseph Goguen [17] who made objects fundamental in his OBJ series of languages [18]. Although OBJ has found practical applications in interesting projects, the author thinks it is fair to say that the ADT-related object concepts did not find their way into everyday practice. Practice in conceptual modeling, and in software engineering at large, came to be heavily dominated by UML which did not succeed in raising much enthusiasm among ADT theorists—so far.

But of course, it is not easy to abandon an Old Love; the author made attempts to use ADT theory for database aspects. An example is [8]. In brief, the ideas are as follows:

- the *data* in a database are organized in ADTs—but they are usually standard, not open to user specification
- database objects are identified by *keys* specified, for instance, in the context of relational tables; keys are usually immutable
- database keys are organized in ADTs which are *extensions* of the data-level ADTs
- for their specification, a *final* algebra approach is appropriate, relative to the initial-algebra data layer.

Ironically, the term *abstract data type* had always been used in the programming area in the sense that later would become standard as *object*, namely a stateful unit of state-based operations reading and writing data.

So the author's migration *from data types to object types* (this is the title of [13]) was, in a way, a turn back to the programming roots. Although substantial agreement has been achieved in practice, there is still no coherent "theory of objects" which copes with all aspects, including interaction, aggregation, inheritance, types, classes, specification, implementation, correctness, verification, ..., and which can provide a sufficiently rich and reliable basis for designing, implementing, and using object-oriented languages and systems.

The problem is not that there is no theory. The problem is that there are so many. And that they are so diverse.

4 Visits Home

Writing textbooks takes time. So the book [9][9] appeared long after initiating the project, when all three authors had abandoned the ADT area for years. They were relieved when the book was finished, and they did not find the push to work on an English version. Unfortunately.

So the author was quite glad when Jacques Loeckx asked him years later to coauthor an English textbook on abstract data types where the basic material in [9] could be reused [21][10]. In order to find wider distribution, this textbook does not use categories. Figure 6 shows the covers of the textbooks.

[9] Authored by the author, Martin Gogolla and Udo Lipeck.
[10] Authored by Jacques Loeckx, the author and Markus Wolf.

Fig. 6. Textbooks [9] and [21]

Another visit home—in the sense of using categories and cooperating with an ADT pioneer, not exactly in the sense of working on an ADT subject—was [10], coauthored with Joseph Goguen and Amílcar Sernadas. It is an amalgamation of two approaches, Joseph's *objects–as–sheaves* approach and Amilcar's and the author's *objects as observed processes* approach. So it put two of the many theories of objects into relationship.

A moment of commemoration is in order. Joseph Amadee Goguen passed away on July 3, 2006. We owe him a lot.

5 Closing the Shrine

After having been away from the WADT community for most of the last 20 years, the author felt during WADT 2010 that the subjects covered were not that far away from what he had been doing in the more theoretical parts of his work. Partly, at least. So he could as well have stayed...

For the future, the author feels inspired by the following citation from a tragedy of German classic literature: Johann Wolfgang von Goethe, *Die natürliche Tochter*. Trauerspiel, 1. Akt, Herzog.[11]

> "...
> Und heute noch, verwahrt im edlen Schrein,
> Erhältst du Gaben, die du nicht erwartet.
> ..."
>
> "... / And still today, coffered in a noble shrine /
> You receive gifts that you did not expect. / ..." [12]

Indeed—that happens all the time. For instance when the author was invited to the 20^{th} WADT. He was surprised. And pleased that the WADT community remembered him. So he now feels encouraged to keep in closer touch.

[11] The Natural Daughter, Tragedy, 1st Act, Duke.
[12] Translation: the author.

And it goes on:

"...
Hier ist der Schlüssel! Den verwahre wohl!
Bezähme deine Neugier! Öffne nicht,
Eh' ich dich wieder sehe, jenen Schatz.
Vertraue niemand, sei es, wer es sei.
Die Klugheit rät's, der König selbst gebeut's.
..."

"... / Here is the key! Keep it well under lock! / Restrain your curiosity! Do not open / That shrine before we meet again / Trust nobody, whoever it may be / Judiciousness suggests it, the king himself demands it. / ..."

The conclusion is that the author shall not reopen the shrine. Never! Well, ... certainly not before the 30^{th} WADT!

Acknowledgements. Warmest thanks are due to the organizers of the 20^{th} anniversary WADT and the editors of this volume. In the final reviewing process, the three anonymous referees were very friendly and helped a lot to improve the paper. Many thanks to them! It goes without saying that the author admits responsibility for all remaining deficiencies.

References

1. Bidoit, M., Mosses, P.D. (eds.): CASL User Manual. LNCS, vol. 2900. Springer, Heidelberg (2004)
2. Broy, M., Wirsing, M., Pair, C.: A Systematic Study of Models of Abstract Data Types. Theoretical Computer Science 33, 139–174 (1984)
3. Burstall, R.M., Goguen, J.A.: Putting Theories Together to Make Specifications. In: Proc. 5th IJCAI, pp. 1045–1058. MIT, Cambridge (1977)
4. Burstall, R.M., Goguen, J.A.: The Semantics of CLEAR, a Specification Language. In: Bjorner, D. (ed.) Abstract Software Specifications. LNCS, vol. 86, pp. 292–331. Springer, Heidelberg (1980)
5. Common Framework Initiative (CoFI), `http://www.informatik.uni-bremen.de/cofi/wiki/index.php/CoFI`
6. Ehrich, H.-D.: On the Theory of Specification, Implementation, and Parametrization of Abstract Data Types. Journal of the ACM 29, 206–277 (1982)
7. Ehrich, H.-D.: Algebraic (?) Specification of Conceptual Database Schemata (extended abstract). In: Kreowski, H.-J. (ed.) Recent Trends in Data Type Specification. Informatik-Fachberichte, vol. 116. Springer, Berlin (1985)
8. Ehrich, H.-D.: Key Extensions of Abstract Data Types, Final Algebras, and Database Semantics. In: Pitt, D., Abramsky, S., Poigné, A., Rydeheard, D. (eds.) Category Theory and Computer Programming. LNCS, vol. 240, pp. 412–433. Springer, Heidelberg (1986)
9. Ehrich, H.-D., Gogolla, M., Lipeck, U.W.: Algebraische Spezifikation Abstrakter Datentypen. Teubner, Stuttgart (1989)
10. Ehrich, H.-D., Goguen, J.A., Sernadas, A.: A Categorial Theory of Objects as Observed Processes. In: de Bakker, J.W., de Roever, W.-P., Rozenberg, G. (eds.) REX 1990. LNCS, vol. 489, pp. 203–228. Springer, Heidelberg (1991)

11. Ehrich, H.-D., Lipeck, U.: Algebraic Domain Equations. Theoretical Computer Science 27, 167–196 (1983)
12. Ehrich, H.-D., Sernadas, A.: Local Specification of Distributed Families of Sequential Objects. In: Astesiano, E., Reggio, G., Tarlecki, A. (eds.) Abstract Data Types 1994 and COMPASS 1994. LNCS, vol. 906, pp. 219–235. Springer, Heidelberg (1995)
13. Ehrich, H.-D., Sernadas, A., Sernadas, C.: From Data Types to Object Types. J. Inf. Process. Cybern. EIK 26(1-2), 33–48 (1990)
14. Ehrig, H., Kreowski, H.-J., Thatcher, J.W., Wagner, E.G., Wright, J.B.: Parameterized Data Types in Algebraic Specification Languages, pp. 157–168. Springer, Berlin (1980)
15. Ehrig, H., Pfender, M., Schneider, H.J.: Graph-grammars: An algebraic approach. In: Proceedings of the 14th Annual Symposium on Switching and Automata Theory (SWAT 1973), pp. 167–180. IEEE Computer Society, Washington, DC, USA (1973)
16. Gogolla, M., Drosten, K., Lipeck, U., Ehrich, H.-D.: Algebraic and Operational Semantics of Specifications Allowing Exceptions and Errors. Theoretical Computer Science 34, 289–313 (1984)
17. Goguen, J.A.: Objects. International Journal of General Systems, 1563-5104 1, 237–243 (1974)
18. Goguen, J.A.: Some Design Principles and Theory for OBJ-0. In: Yeh, R. (ed.) LNCS, vol. 75, pp. 425–475. Prentice-Hall (1979)
19. Goguen, J.A., Thatcher, J.W., Wagner, E.G.: An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types. In: Yeh, R. (ed.) Current Trends in Programming Methodology IV, pp. 80–149. Prentice-Hall (1978)
20. Goguen, J.A., Thatcher, J.W., Wagner, E.G., Wright, J.B.: Initial Algebra Semantics and Continuous Algebras. Journal of the ACM 24, 68–95 (1977)
21. Loeckx, J., Ehrich, H.-D., Wolf, M.: Specification of Abstract Data Types. J. Wiley & Sons and B.G.Teubner Publishers (1996)
22. Mosses, P.D. (ed.): CASL Reference Manual. LNCS, vol. 2960. Springer, Heidelberg (2004)
23. Scott, D.S.: Data Types as Lattices. SIAM J. Comp. 5, 522–587 (1976)
24. Smyth, M.B., Plotkin, G.D.: The Category-Theoretic Solution of Recursive Domain Equations. In: Proc. 18th IEEE FOCS, pp. 13–17 (1977)