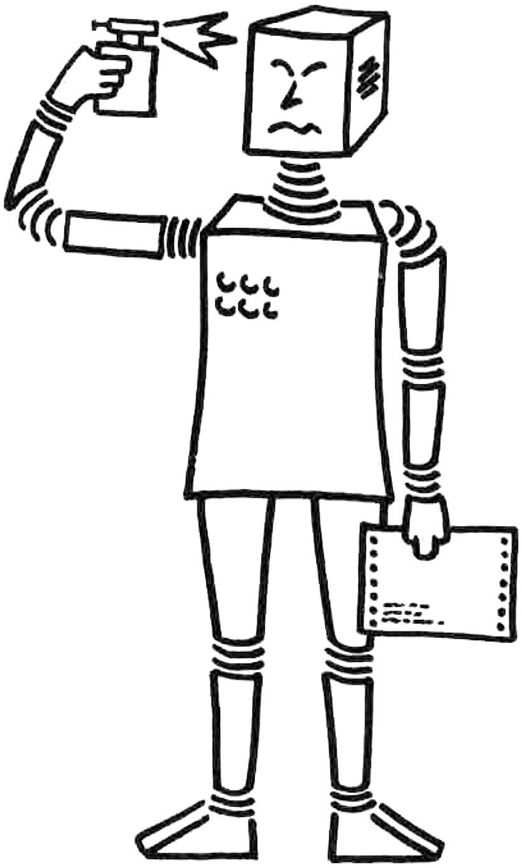
SEKI-PROJEKT

SEKI MEMO

Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern 1, W. Germany



Algebraic Domain Equations for Specifications Containing Inequational Axioms

Gerd Krützer

MEMO SEKI-84-04

ALGEBRAIC DOMAIN EQUATIONS FOR SPECIFICATIONS CONTAINING INEQUATIONAL AXIOMS

from

Gerd Krützer

Abstract

Algebraic domain equations (ade's) provide a means for implicitly or recursively specifying parameterized data types. A unique semantics is available provided the respective ade's are defined over a category of specifications which are solely based on equational axioms. We extend this approach by showing that there exists an appropriate semantics even if the respective specifications contain as well equational as inequational axioms.

Keywords:

Abstract data types, parameterization, specifications with inequalities, algebraic domain equations.

Contents

0. Introduction

Ehrich and Lipeck introduced in their paper [E/L 81] some sort of recursive domain equations for the specification of parameterized abstract data types. Though they were dealing with the 'initial-algebra' - approach to data types they pointed out some analogies to Scott's theory of 'Domain Equations'. Since they were working with algebras rather than with some paritally ordered sets Ehrich and Lipeck called their work 'Algebraic Domain Equations'.

They started with a category of algebraic specifications with equations called **Spec**. Then a parameterized specification is some injective **Spec** - morphism p: $\mathbf{X} \to \mathbf{D(X)}$. (Here $\mathbf{D(X)}$ indicates that the formal parameter $\mathbf{X}$ is fully contained in the target specification $\mathbf{D}$). Parameter passing is defined by the pushout of some diagram in **Spec**.

Then each parameterized specification defines a forgetful functor $\bar{P}$: $\mathbf{Alg}_{\mathbf{D(X)}} \to \mathbf{Alg}_{\mathbf{X}}$. The parameterized data type defined by p: $\mathbf{X} \to \mathbf{D(X)}$ is the (strongly) persistent left-adjoint functor P: $\mathbf{Alg}_{\mathbf{X}} \to \mathbf{Alg}_{\mathbf{D(X)}}$ which sends each $\mathbf{X}$-algebra to its free extension in $\mathbf{Alg}_{\mathbf{D(X)}}$. Here (strong) persistency of P means that for each $\mathbf{X}$-algebra A there holds ($|\bar{P}oP|(A) = A$) $|\bar{P}oP|(A) \cong A$.

Now an algebraic domain equation (ade) is an equation of the form

$$\mathbf{X} \overset{(p,e)}{\underset{\Rightarrow}{=}} \mathbf{D(X)}$$

where p,e: $\mathbf{X} \to \mathbf{D(X)}$ are **Spec** - morphisms such that p: $\mathbf{X} \to \mathbf{D(X)}$ is a parameterized specification.

But all what we have done until now is to give the syntactic requirements. What is the semantics of such an equation?

Let's take a short look to a 'domain equation' in Scott's theory of domains. Let D = $\mathbf{K}$(D) be a domain equation. Here $\mathbf{K}$(D) is a domain-expression containting the domain-variable D and the names of some basic domains all connected by domain constructors like '+' (sum), 'x' (product) or '→' (function-space). This is the syntactic aspect. Then by using the inverse-limit construction we proceed in the following way

- we define a domain D_0 defined by the basic domains connected as in $\mathbf{K}(D)$.

- we look for a retraction-pair (i_0, j_0)

  $i_0: D_0 \to \hat{\mathbf{K}}(D_0)$

  $j_0: \hat{\mathbf{K}}(D_0) \to D_0$

- we proceed by iterating (for k>1)

  $D_{k+1} = \hat{\mathbf{K}}(D_k)$

  $i_{k+1} := \hat{\mathbf{K}}(i_k): \hat{\mathbf{K}}(D_k) \to \hat{\mathbf{K}}(D_{k+1})$

  $j_{k+1} := \hat{\mathbf{K}}(j_k): \hat{\mathbf{K}}(D_{k+1}) \to \hat{\mathbf{K}}(D_k)$

- then the solution for $D = \mathbf{K}(D)$ is the so called inverse limit

$$D_\infty = \{<d_0, d_1, \ldots> \mid \forall_{k=0}^{\infty} d_k = j_k(d_{k+1})\}$$

Now the solution or the sematics of the above domain equation is a fixedpoint of the endofunctor $\hat{\mathbf{K}}$: $\underline{\underline{CPO}} \to \underline{\underline{CPO}}$.
And thus the argumentation of Ehrich and Lipeck is such that the consider the semantics of the solution of the algebraic domain equation $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D(X)}$ should be a fixedpoint of some endofunctor. The only relevant endofunctors which may be constructed from the data given by $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D(X)}$ are

$P \circ E: \underline{\underline{Alg}}_{\mathbf{D(X)}} \to \underline{\underline{Alg}}_{\mathbf{D(X)}}$

and

$E \circ P: \underline{\underline{Alg}}_{\mathbf{X}} \to \underline{\underline{Alg}}_{\mathbf{X}}$.

The second choice would not make much sense since we intended to extend our argument-type. But this cannot be achieved by applying the forgetful functor $\bar{E}$ which takes each $\mathbf{D(X)}$ algebra to it's $\mathbf{X}$-reduct.
So we decide to use $P\circ\bar{E}$. But there may be a lot of fixedpoints for $P\circ\bar{E}$. Now in Scott's approach each domain equation denoted just one domain and not a class of domains. In analogy to this Ehrich and Lipeck choose as unique solution for the algebraic domain equation $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D(X)}$ the initial $\mathbf{Q}$-algebra $I_{\mathbf{Q}}$ where $(q,\mathbf{Q})$

is the coequalizer of p,e according to the diagram

$$X \overset{p}{\underset{e}{\rightrightarrows}} D(X) \xrightarrow{q} Q.$$

What we want to do here is to look whether algebraic domain equations can be used for other sorts of specifications namely those which contain <u>inequalities</u> in their axiom sets. For example Hornung [H/R ?] uses specifications with positive conditional equations and simple inequalities and gives an initial and terminal semantics of parameterized abstract data types. Milner [Mi 77] and Möller [Mö 82] use inequalities in the framework of partially odered algebras.

We shall proceed in the following way:
In Chapter I we shall recall the main concepts of algebraic domain equations as far as they are needed for our purposes.
In Chapter II we shall present some results about the category of specifications with inequalities as they are introduced in Hornung´s paper [H/R ?].
In Chapter III we give the context in which algebraic domain equations can be applied to specifications with inequalities.

I. ALGEBRAIC DOMAIN EQUATIONS

Algebraic domain equations as we use them here are defined by using parameterized specifications. We have already outlined the general framework in the introduction. Thus in the current chapter we shall give the formal background. The basic results presented here are given for the category **Spec** (specifications with equational axioms); the extension to specifications with inequalities will be given in Chapter II.

I.1 Specifications and ADT's

Following the well known results of the ADJ - group we take a specification **S** to be a triple

S = <S,Σ,E>

where **S** is the set of sorts

Σ is the set of operations

and E is the set of equations (axioms).

The pair $\underline{\Sigma}$:= <S,Σ> is called the signature of this specification.

The signature belongs to a syntactical level in the sense that it determines the form of the specified data type. Each algebra A which is supposed to be a model for this specification must have a carrier-set A_s for each sort $s \in S$ and an operation $\sigma: A_{s1} \times \ldots \times A_{sn} \to A_s$ for each $\sigma \in \Sigma_{s1,\ldots,sn,s}$.

The set E of equations consists of pairs <L,R> where L,R are terms built from operations (of the signature) and variables (from an S - sorted variable - set X). In the categorical view we want to choose a category whose objects are specifications. But we still need to say which morphisms should connect various specifications.

Thus we first say what a signature morphism is.

I.1.1 Definition

Let $\underline{\Sigma}$ = <S,Σ> and $\underline{\Sigma}'$ = <S',Σ'> be two signatures.

Then a signature-morphism is a pair $f=<f_{sort},f_{op}>$ with

(i) a sort-mapping $f_{sort}: S \to S'$

and

(ii) an operation mapping $f_{op}: \Sigma \to \Sigma'$

such that if $\sigma \in \Sigma_{s1,\ldots,sn,s}$ then

$$f_{op}(\sigma) \in \Sigma'_{fsort(s1),\ldots,fsort(sn),fsort(s)}$$

With this definition we can build the category of signatures as objects and signature morphisms as morphisms. We denote this category by **Sig**.

We are now able to define specification- morphisms which in a sense take care of a proper translation of equations from one specification to another. So we come along with the following

I.1.2 Definition

Let $\mathbf{S} = <S,\Sigma,E>$ and $\mathbf{S}' = <S',\Sigma',E'>$ be specifications.

Then a specification-morphism

$f : \mathbf{S} \to \mathbf{S}'$

is a signature-morphism $f=<f_{sort},f_{op}>$ such that

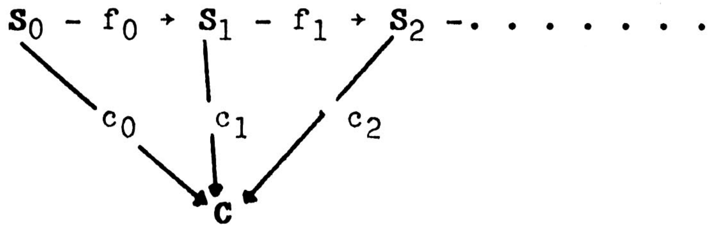
$\forall e \in E. f(e) \in E'$

In this sense a specification-morphism corresponds to the theory-morphism as in [B/G 80].

So we get the category with specifications as objects and specification-morphisms as morphisms. This category is a very important one and as already pointed out, we denote it by **Spec**.

Now some technical results about this category, which we shall use later on. These results can be found in the paper of Ehrich and Lipeck [E/L 81] and thus detailed proofs will be omitted.

Most of these results have to do with the cocompleteness-property of **Spec**. Cocompleteness as we need it here means that for any family $(\mathbf{S}_i | i \in I)$ of specifications in **Spec** there is a unique object $\mathbf{C}$ in **Spec** and a family of **Spec**-morphisms $c_i: \mathbf{S}_i \to \mathbf{C}$ such that the following diagram commutes



Informally speaking this means that whenever we have specifications $\mathbf{S}_0$, $\mathbf{S}_1$,... connected by specification morphisms f_i: $\mathbf{S}_i \to \mathbf{S}_{i+1}$ we can determine a unique specification **C** in which all specifications $\mathbf{S}_i$ can be ´embedded´ (without loss of ´information´) by specification morphisms c_i: $\mathbf{S}_i \to \mathbf{C}$. Furhtermore this ´embedding´ respects the connection between the specifications $\mathbf{S}_i$, $\mathbf{S}_{i+1}$ due to the commutativity property of the diagram above.

This means $\forall i \varepsilon I. c_{i+1} o f_i = c_i$

The importance of the cocompleteness of Spec lies in the fact that for $(\mathbf{S}_i | i \varepsilon I)$ there is a unique specification **C** which in a sense ´contains´ all the structure carried by $(\mathbf{S}_i | i \varepsilon I)$.

Now the results:

I.1.3 Theorem [E/L 81]

Spec has coequalizers.

This means that for any pair of Spec-morphisms f,g: $\mathbf{S} \to \mathbf{S}'$ there exists a unique specification **C** and a unique morphism h: $\mathbf{S}' \to \mathbf{C}$ with $hof = hog$.

C and h are such that for any specification $\mathbf{C}'$ and morphism h': $\mathbf{S}' \to \mathbf{C}'$ with $h'of = h'og$ there exists a unique morphism r: $\mathbf{C} \to \mathbf{C}'$ such that $h' = roh$.

I.1.4 Thoerem [E/L 81]

Spec has coproducts.

This means that for any family $(\mathbf{S}_i | i \varepsilon I)$ of specifications there exists a unique object **C** and a family of (coproduct-injections)

i_k: $\mathbf{S}_k \to \mathbf{C}$. These are such that for any object **D** and any family $(d_k:\mathbf{S}_k \to \mathbf{D} \mid k\varepsilon K)$ there always exists a unique Spec-morphism $h: \mathbf{C} \to \mathbf{D}$ such that $\forall k \varepsilon I.\ h \circ i_k = d_k$.

I.1.5 Theorem

Spec is cocomplete.
This is a consequence of the preceeding two theorems.

For further discussion it is useful to show the construction of coequalizers and coproducts in Spec.
The coproduct of two specifications $\mathbf{S} = \langle S,\Sigma,E\rangle$ and $\mathbf{S}' = \langle S',\Sigma',E'\rangle$ is simply the triple built from the disjoint union of the components of **S** and **S'**. We denote it by $\mathbf{S} + \mathbf{S}' := \langle S+S',\Sigma+\Sigma',E+E'\rangle$ ('+' means disjoint union).
Given the two specifications **S** and **S'** as above. Furthermore let $f,g: \mathbf{S} \to \mathbf{S}'$ be specification-morphisms. Then the coequalizer of f and g is built in the following way:
First take $\underline{R}(S')$ to be the least equivalence relation on S' generated by the set $\{\langle f_{sort}(s),g_{sort}(s)\rangle \mid s\varepsilon S\}$
Then take $\underline{R}(\Sigma')$ to be the least equivalence relation generated by the set $\{\langle f_{op}(\sigma),g_{op}(\sigma)\rangle \mid \sigma\varepsilon\Sigma\}$ which respects $\underline{R}(S')$. This means: By $\underline{R}(S')$ the set S' is divided into equivalence classes. Then give each equivalence class a unique new name. This leads to a new sort-set S". Let $\sigma \varepsilon \Sigma_{s1,\ldots,sn,s}$. Then the sorts $s_1,\ldots,s_n,s$ are mapped to the new sorts (in S") denoted by $[f_{sort}(s_1)],\ldots,[f_{sort}(s_n)],[f_{sort}(s)]$ for the respective equivalence classes of $\underline{R}(S')$.)
We had already $\sigma \varepsilon \Sigma_{s1,\ldots,sn,s}$. The corresponding coequalizer-operation falls into the equivalence class $[f_{op}(\sigma)]$ belonging to the new operation set $\Sigma'[f_{sort}(s_1)],\ldots,[f_{sort}(s_n)],[f_{sort}(s)]$.
To get the new operation-set Σ" we have to rename the equivalence classes generated by $\underline{R}(\Sigma')$ with unique new operation names. The equation-set E' is then renamed according to the previous renaming of sort- and operation-sets.

The coequalizer morphism h (in I.1.3) is then simply defined by

(i) sort-mapping h_{sort}

$$\forall s' \varepsilon S'.\ h_{sort}(s') := [s']$$

(ii) operation-mapping

$$\forall \sigma' \varepsilon \Sigma'_{s1' \ldots sn', s'}.\ h_{op}(\sigma') := [\sigma']$$

Now one can easily verify $h \circ f = h \circ g$.

I.2 Algebras: Models for Specifications

Specifications are the syntactical description for adt's. They determine in a sense the 'form' of adt's. On the semantical level we have to consider models for specifications. Here the ADJ-group uses heterogeneous algebras. We shall shortly review their interpretation by giving the most important definitions and theorems (without proof) as far as we need them here. For more detailed information consult for example [ADJ 82].

Let Spec = $<S,\Sigma,E>$ be a specification with signature $\underline{\Sigma} = <S,\Sigma>$.

I.2.1 Definition

A Σ-algebra A is given by:

(i) a set A_s for each sort $s \varepsilon S$

($\bigcup_{s \varepsilon S} A_s$ is called the carrier-set)

(ii) a mapping $\sigma_A: A_{s1} \times A_{s2} \times \ldots \times A_{sn} \rightarrow A_s$ for each operation $\sigma \varepsilon \Sigma_{s1,\ldots,sn,s}$

I.2.2 Definition

Let A, B be $\underline{\Sigma}$ - algebras (the respective carriers will be denoted by the name of the algebras!). A $\underline{\Sigma}$- algebra-homomorphism is a mapping h: A $\rightarrow$ B such that

$$\forall \sigma \varepsilon \Sigma_{s1,\ldots,sn,s}\ \forall a_1,\ldots,a_n \ \varepsilon\ A_{s1} \times \ldots \times A_{sn}.$$
$$h(\sigma_A(a_1,\ldots,a_n)) = \sigma_B(h(a_1),\ldots,h(a_n))$$

The category with $\underline{\Sigma}$-algebras as objects and $\underline{\Sigma}$-algebra-homomorphisms as morphisms is denoted $\underline{\underline{Alg}}_{\Sigma}$.

I.2.3 Definition

A Σ-term is defined by

(i) each constant $c \in \Sigma_{\lambda,s}$ is a Σ-term (of sort s)
(λ denotes the empty word!)

(ii) Let $t_1,\dots,t_n$ be Σ-terms of sorts $s_1,\dots,s_n$ and $\sigma \in \Sigma_{s1,\dots,sn,s}$. Then $\sigma(t_1,\dots,t_n)$ is a Σ-term of sort s.

Σ-algebras can be interrelated with certain structure-preserving mappings, called Σ-algebra-homomorphisms.

But Σ-terms may not always give what we need. Thus we introduce terms with variables.

Let $X := \bigcup_{s \in S} X_s$ be a countable set of variables.

Then we define Σ-terms which (eventually) contain variables from X. These terms will be denoted $\Sigma(X)$-terms.

I.2.4 Definition

The following terms are considered to be $\Sigma(X)$-terms.

(i) Each Σ-term is a $\Sigma(X)$-term

(ii) Each variable $x \in X_s$ is a $\Sigma(X)$-term of sort s.

(iii) Let $t_1,\dots,t_n$ be Σ-terms of sorts $s_1,\dots,s_n$ and $\sigma \in \Sigma_{s1,\dots,sn,s}$. Then $\sigma(t_1,\dots,t_n)$ is a $\Sigma(X)$-term.

Terms can be used to construct carrier-elements of so-called term-algebras. These term-algebras have an interesting property which makes them adequate candidates for unique models for specifications: They are initial in $\mathbf{Alg}_\Sigma$.

I.2.5 Definition

A Σ-algebra A is initial in $\mathbf{Alg}_\Sigma$, if for each Σ-algebra B there exists a unique Σ-algebra-homomorphism h_B: A $\to$ B.

Now we have to give a description of the term-algebra determined by Σ.

I.2.6 Definition

The term-algebra $T\Sigma$ determined by the signature $\underline{\Sigma}$ is defined by the following conditions:

(i) The carrier set for a sort $s \in S$ is the set of $\underline{\Sigma}$-terms of sort s.

(ii) Let $\sigma \in \Sigma_{s1,\dots,sn,s}$ and $t_1,\dots,t_n$ be $\underline{\Sigma}$-terms of sorts $s_1,\dots,s_n,s$. Then $\sigma_{T\Sigma}$ is defined by

$$\sigma_{T\Sigma}(t_1,\dots,t_n) := \sigma(t_1,\dots,t_n).$$

I.2.7 Theorem

$T\Sigma$ is initial in $\underline{\underline{Alg}}_\Sigma$.

I.2.7 Definition

Let $\underline{\Sigma}(X)$ be the signature with variables from definition I.2.4. Then the algebra $T\Sigma(X)$ is defined by the following conditions:

(i) $T\Sigma(X)_s$ is the set of all $\Sigma(X)$-terms as in definition I.2.4 (for each sort $s \in S$).

(ii) Let $t_1,\dots,t_n$ be terms from $T\Sigma(X)_{s(1)},\dots,T\Sigma(X)_{s(n)}$ and $\sigma \in \Sigma_{s1,\dots,sn,s}$.
Then

$$\sigma_{T\Sigma(X)}(t_1,\dots,t_n) := \sigma(t_1,\dots,t_n)$$

According to this definition $T\Sigma(X)$ is a $\underline{\Sigma}$-algebra namely the free $\underline{\Sigma}$-algebra generated by the set X.
Terms of $T\Sigma(X)$ can be evaluated in a $\underline{\Sigma}$-algebra A if we assign to each variable an element of A.

I.2.8 Definition

Let $T\Sigma(X)$ be the free $\underline{\Sigma}$-algebra generated by X and A be a $\underline{\Sigma}$-algebra.
Then an assignment from elements of A to X is a mapping

$$\Theta: X \to A$$

with $\Theta := (\Theta_s: X_s \to A_s \mid s \in S)$

Such an assignment determines the so-called evaluation-mapping for $\underline{\Sigma}(X)$-terms in a $\underline{\Sigma}$-algebra A.

I.2.9 Theorem

Let Θ be an assignment and A be a $\underline{\Sigma}$-algebra.
Let the evaluation-mapping $\bar{\Theta}$: $T_{\Sigma(X)} \to A$ with
$\bar{\Theta} = (\bar{\Theta}_s : T\Sigma(X)_s \to A_s \mid s \in S)$ be defined by
(i) $\forall x \in X_s .\ \bar{\Theta}_s(x) = \Theta_s(x)$
(ii) Let $t_1,\ldots,t_n$ be elements of $T\Sigma(X)_{s1},\ldots,T\Sigma(X)_{sn}$ and
$\sigma \in \Sigma_{s1,\ldots,sn,s}$. Then
$\bar{\Theta}(\sigma(t_1,\ldots,t_n)) := \sigma_A(\Theta_{s1}(t_1),\ldots,\Theta_{sn}(t_n))$.
Then $\bar{\Theta}$: $T\Sigma(X) \to A$ is a $\underline{\Sigma}$-homomorphism.

This theorem shows that interpretations of terms fit into the algebraic framework, because they are $\underline{\Sigma}$-algebra homomorphisms. Now we must say what equations are considered to be and what it means to say: an equation is satisfied in an algebra.

I.2.10 Definition

A $\underline{\Sigma}$-equation is a pair E=<L,R> with $L,R \in T\Sigma(X)_s$ for a sort $s \in S$.

I.2.11 Definition

Let E = <L,R> be a $\underline{\Sigma}$-equation of sort $s \in S$ and A be a $\underline{\Sigma}$-algebra.
Then A satisfies E if for all assignments Θ: X $\to$ A the evaluation $\bar{\Theta}$: $T\Sigma(X) \to A$ gives
$\bar{\Theta}_S(L) = \bar{\Theta}_S(R)$.

This definition means that an equation is valid in an algebra, if all interpretations of left- and right-hand sides of E in A have the same value as result.
Let A be a $\underline{\Sigma}$-algebra. Then a congruence relation $\equiv$ on A is a family $\equiv := (\equiv_s \subseteq A_s \times A_s \mid s \in S)$ such that each $\equiv_s$ is an equivalence

on A_s and respects the operations in the sense that if
$\sigma \ \varepsilon \ \Sigma_{s1,\ldots,sn,s}$ and $a_1,\ldots,a_n \ \varepsilon \ A_{s1}\times\ldots\times A_{sn}$ then
$\sigma_A([a_1],\ldots,[a_n]) = [\sigma_A(a_1,\ldots,a_n)]$.
([a] denotes the equivalence-class of a).
Let **S** = $<S,\Sigma,E>$ be a specification. Then we define a congruence relation on $T\Sigma$ by using the equations E and assignments θ: $X \rightarrow T\Sigma$ (X is the variable set of E) in the following way:

I.2.12 Definition

The congruence relation on $T\Sigma$ generated by E is a family $\equiv_E = (\equiv_{E,s} \subseteq T\Sigma_s \times T\Sigma_s \mid s\varepsilon S)$ of least congruences defined by $\equiv_{E,s}$.

(i) Let E = <L,R> be an equation in E of sort S. Then

$$\bar\theta(L) \equiv_{E,s} \bar\theta(R)$$

(ii) Let $\sigma \ \varepsilon \ \Sigma_{s1,\ldots,sn,s}$ and $t_i, t_i' \ \varepsilon \ T\Sigma_{si}$ $(i=1,\ldots,n)$ with $t_i \equiv_{E,si} t_i'$.
Then

$$\sigma(t_1,\ldots,t_n) \equiv_{E,s} \sigma(t_1',\ldots,t_n')$$

(iii) $\forall t\varepsilon T\Sigma_s.\ t\equiv_{E,s} t$

(iv) $\forall t,t'\varepsilon T\Sigma_s.\ t \equiv_{E,s} t' \Rightarrow t' \equiv_{E,s} t$

(v) $\forall t,t',t''\varepsilon T\Sigma_s.\ (t\equiv_{E,s} t' \,\&\, t' \equiv_{E,s} t'') \Rightarrow t \equiv_{E,s} t''$

The congruence-classes [t] for $t \ \varepsilon \ T\Sigma_s$ are built by

$$[t]_{\equiv E,s} := \{t'\varepsilon T\Sigma_s \mid t \equiv_{E,s} t'\}$$

Then we can build the quotient-term algebra $T\Sigma/_{\equiv E}$ in the following way:

I.2.13 Definition

Let **S** = $<S,\Sigma,E>$ be a specification. Then the quotient-term-algebra $T\Sigma/_{\equiv E}$ is defined by

(i) For each $s\varepsilon S$ the carrier-set $T\Sigma/_{\equiv E,s}$ is the set
$T\Sigma/_{\equiv E,s} := \{[t]_{\equiv E} \mid t \ \varepsilon \ T\Sigma_s\}$

(ii) Let $\sigma \ \varepsilon \ \Sigma_{s1,\ldots,sn,s}$ and $t_i \ \varepsilon \ T\Sigma_{si}$ $(i = 1,\ldots,n)$.
Then

$$\sigma_{T\Sigma/\equiv E}([t_1],\ldots,[t_n]) := [\sigma(t_1,\ldots,t_n)]$$

I.2.14 Theorem

Let **S** = <S,Σ,E> be a specification and $T\Sigma/_{\equiv E}$ the quotient-termalgebra defined by **S**.
Then $T\Sigma/_{\equiv E}$ is initial in $\underline{Alg}_{\Sigma,E}$ (and is uniquley determined up to isomorphism!).
So we are prepared to say what an adt is considered to be in the ADJ-philosophy:

I.2.15 Definition

Let **S** = <S,Σ,E> be a specification.
Then by the abstract data type specified by **S** we mean the isomorphism-class of the quotient-termalgebra $T\Sigma/_{\equiv E}$.

I.3 Parameterized Specifications and Parameterized Data Types

We now show how ´new´ data types can be constructed from ´old´ ones in the ADJ-approach by using the ´parameterization-technique. On the syntactical level parameterization means that we start with a formal parameter specification **X** and ´embed´ it into a resulting specification **D** via an injective Spec-morphim p: **X** →
The formal parameter has very little structure such that there is a (eventually) large class of specifications in Spec which will fit this structure and can therefore serve as actual syntactical parameters. Parameterization means that one specification is built from one or more parameter-specifications by eventually extending the structure provided by the parameters with new sorts , new operations and new equations. This is so far the syntactical view.
On the semantical level parameterization means transformation of algebras of one category into algebras of another (resultant) category together with transformation of algebra-homomorphisms.

This should be done in such a way that the structure of the parameter-algebra will not be lost. This means that by a certain 'reduction' of the resultant algebra we get an algebra that has the same structure as the parameter-algebra. The transformation of 'old' structures (category of parameter algebras) into 'new' (extended) structures (category of parameterized algebras) will be carried out by functors (according to the category-theoretical viewpoint used in the ADJ-approach). Analogously the 'reduction' will be carried out by so-called forgetful functors. These functors 'forget' in a sense all of the additional structure of the resultant algebras and 'concentrate' only on the structure of the 'old' parameter algebra.

According to the philosophy that an adt should be uniquely determined the resultant (parameterized) algebra is the 'free'-extension of the parameter algebra. 'Free extension' means that the elements of the 'old' carriers A_s (for the parameter algebra) become by transformation (with the respective functor) elements of the new carrier $B_{p(s)}$ (if B is the resultant algebra and p: $\mathbf{X} \to \mathbf{D}$ the parameterized specification belonging to the transformation).

The following definitions and theorems formalize the above ideas. The results are taken from [E/L 81].

Remark: In the sequel if p: $\mathbf{X} \to \mathbf{D}$ is a Spec-morphism, then
$p(s) := p_{sort}(s)$ $(s \in S)$ and $p(\sigma) := p_{op}(\sigma)$ $(\sigma \in \Sigma_{s1,..,sn,s})$.

I.3.1 Definition

A parameterized specification is an injective Spec-morphism

$$p: \ X \to \mathbf{D}$$

$\mathbf{X}$ is called the formal parameter of p.

In the sequel we use a special kind of parameterized specifications, namely (strongly) persistent specifications. This property is mainly connected with the transformation (of data types) specified by the parameterized specifications. The transformation is expressed by using certain functors between categories of algebras. So we introduce here the functors with which we are concerned in parameterization namely forgetful and (strongly) persistent functors.

Remark: If e: $\mathbf{X} \to \mathbf{D}$ is a $\underline{\underline{Spec}}$-morphism then the respective persistent functor belonging to e will be denoted by the (upper case letter) E and the respective forgetful functor will be denoted by $\bar{E}$.

I.3.2 Definition

Let e: $\mathbf{X} \to \mathbf{D}$ be a $\underline{\underline{Spec}}$-morphism and B be a **D**-algebra. Furthermore let sig(**X**),sig(**D**) and sorts(**X**),sorts(**D**) denote the signatures and sorts of the specifications **X** and **D**.
Then the forgetful functor E: $\underline{\underline{Alg}}_{\mathbf{D}} \to \underline{\underline{Alg}}_{\mathbf{X}}$ sends each **D**-algebra B to the **X**-algebra A defined by

(i) $\forall s \varepsilon sorts(X).\ A_s := B_{e(s)}$

(ii) Each operation $\sigma\ \varepsilon\ sig(\mathbf{X})_{s1,\ldots,sn,s}$ is defined by the image-operation under e.
σ_A: $A_{s1} \times \ldots \times A_{sn} \to A_s$ is defined to be the operation
$e_{op}(\sigma)_B$: $B_{e(s1)} \times \ldots \times B_{e(sn)} \to B_{e(s)}$
The algebra A is called the sig(**X**)-reduct of B.

(iii) Let B, B′ be **D**-algebras and h: $B \to B'$ be a **D**-algebra-homomorphism.
Let A, A′ be the respective sig(**X**)-reducts of B and B′ defined by $\bar{E}$.
Then $\bar{E}$ transforms h into an **X**-algebra-homomorphism
g: $A \to A'$ by
$\forall s \varepsilon sorts(X).\ g_s := h_{e(s)}$

In the following discussion we denote the object part of a category $\underline{C}$ by $|\underline{C}|$ and the morphism part by $/\underline{C}/$. If A,B ε $|\underline{C}|$ then $\underline{C}$(A,B) denotes the set of all morphisms from A to B.
Furthermore if $\underline{C}$, $\underline{D}$ are categories and F: $\underline{C} \to \underline{D}$ is a functor then we shall denote its object part by $|F|$: $|\underline{C}| \to |\underline{D}|$ and its

morphism part by $/F/: /\underline{C}/ \to /\underline{D}/$.
Now we turn to the definition of persistent functors between categories of algebras. Persistent functors are used to construct parameterized data types from parameter data types. They perform this transformation in such a way that they 'remember' the structure of the parameter-algebra. The structure of the 'remembered' algebra can then be rediscovered by application of a forgetful functor.

I.3.3 Definition

Let p,e: $\mathbf{X} \to \mathbf{D}$ be Spec-morphisms.
Then a persistent functor P (determined by p) is a functor

$$P: \underline{Alg}_{\mathbf{X}} \to \underline{Alg}_{\mathbf{D}}$$

such that

$$\forall A \in |\underline{Alg}_{\mathbf{X}}| . |\bar{E} \circ P|(A) \cong A$$

('≅' means isomorphy and 'o' denotes the composition of functors)

P is strongly persistent iff

$$\forall A \in |\underline{Alg}_{\mathbf{X}}| . |\bar{E} \circ P|(A) = A$$

Now we can see what it means to say that a persistent functor 'remembers' the structure of its argument (or parameter)-algebra namely

$$|\bar{E} \circ P|(A) \cong A$$

or

$$|\bar{E} \circ P|(A) = A.$$

We see that we can always rediscover the structure of the argument and thus no relevant 'information' is lost by application of a persistent functor.
What is left for the moment is to give the respective working definitions for parameterized data types (pdt's) and the semantics of a parameterized specification.
For short: a pdt or data type constructor consists namely of a persistent functor and forgetful functor. The standard-semantics of a parameterized specification is given by a persistent functor P_{free}: $\underline{Alg}_{\mathbf{X}} \to \underline{Alg}_{\mathbf{D}}$ which transforms each $\mathbf{X}$-algebra into its free

extension and by the forgetful functor P: $\underline{\underline{Alg}}_{\mathbf{D}} \to \underline{\underline{Alg}}_{\mathbf{X}}$.
Constructing the free extension of an **X**-algebra A by application of a persistent functor P_{free}: $\underline{\underline{Alg}}_{\mathbf{X}} \to \underline{\underline{Alg}}_{\mathbf{D}}$ with $|P_{free}|(A) := A'$ means:

The old carriers A_s ($s \in sorts(\mathbf{X})$) are bijectively transformed to the 'new' carriers $A'_{p(s)}$ (neither new elements are added to A_s in $A'_{p(s)}$ nor 'old' elements are mapped onto the same image).

and

new carriers, new operations and new equations are eventually added in A'.

These are the key ideas contained in the following sequence of definitions and theorems.

I.3.4 Definition

As a working definition for pdt's we choose
A parameterized data type consists of a parameterized specification

$$p: \mathbf{X} \to \mathbf{D}$$

and the (strongly) persistent functor P_{free}: $\underline{\underline{Alg}}_{\mathbf{X}} \to \underline{\underline{Alg}}_{\mathbf{D}}$ that takes each **X**-algebra A to its free extension over A with respect to p such that

$$|\bar{P} \circ P_{free}|(A) \cong A \quad (|\bar{P} \circ P_{free}|(A) = A)$$

I.3.5 Definition

Let p: **X** $\to$ **D** be a parameterized specification.
Then by the standard-semantics of p we mean the pair (p, P_{free}).

I.3.6 Definition

Let p: **X** $\to$ **D** be a parameterized specification.
The p is called (strongly) persistent if its underlying standard-semantics has this property.

We shall proceed by clarifying parameter-passing in Spec and by building instances of pdt's. The rest of the chapter contains some category-theoretical results on the relation between Spec and the category of Spec-models, the algebras, which are contained in Cat (category of categories with functors as morphisms).
As a parameterized specification is intended to be used as a method of systematically constructing new specifications from old ones we have to indicate what parameter passing means.
In general we bind an actual (syntactical) parameter to the formal parameter in p: **X** → **D** by a morphism f: **X** → **A** and then complete the resulting diagram

$$\begin{array}{ccc} \mathbf{X} & \xrightarrow{p} & \mathbf{D} \\ f\downarrow & & \\ \mathbf{A} & & \end{array}$$

by giving it a unique meaning as the pushout of the diagram

$$\begin{array}{ccc} \mathbf{X} & \xrightarrow{p} & \mathbf{D} \\ f\downarrow & & \downarrow f' \\ \mathbf{A} & \xrightarrow{p'} & \mathbf{B} \end{array}$$

(where f'op = p'of)

I.3.7 Definition

Let p: **X** → **D** be a parameterized specification.
Then an actual syntactical parameter is a pair (f,**A**) such that f: **X** → **A** is a Spec-morphism.

I.3.8 Definition

Let (f,**A**) be an actual syntactical parameter for p: **X** → **D**. Then the result of parameter passing (**A** for **X**) is the pushout of the diagram

$$\begin{array}{ccc} \mathbf{X} & \xrightarrow{p} & \mathbf{D} \\ f\downarrow & & \downarrow f' \\ \mathbf{A} & \xrightarrow{p'} & \mathbf{D}' \end{array}$$

The preceding definition means that the result of parameter-passing is determined by the equivalence of

(i) embedding **X** into **D** and then replacing the formal part in **D** by the actual parameter **A** via using f

and

(ii) substituting **A** for **X** via f and then embedding **A** into **D** (by changing **D** to **D'**).

Again by the requirement that the meaning of parameter- passing should be the pushout of the above diagram we know that the result is uniquely determined. The parameter-passing-mechanism on the syntactical level corresponds in a sense to the following mechanism on the semantical level.

Let (f,**A**) be an actual parameter for the parameterized specification p: **X** → **D**.

Then by using the standard-semantics (p, P_{free}) we indicated that each **X**-algebra will be transformed into a **D**-algebra (free-extension) by using the (strongly) persistent functor P_{free}. Now we take an **A**-algebra and then transform it into a **D'**-algebra by using the standard-semantics of p': A → D', namely (p', P'_{free}) which sends the actual **A**-algebra to its free extension in $\underline{Alg}_{D'}$.

I.3.9 Definition

Let p: **X** → **D** be a parameterized specification with actual parameter (f,**A**). Let the result of parameter passing be the pushout of the following diagramm:

$$\begin{array}{ccc} \mathbf{X} & \xrightarrow{p} & \mathbf{D} \\ f\downarrow & & \downarrow f' \\ \mathbf{A} & \xrightarrow{p'} & \mathbf{D'} \end{array}$$

Then an actual parameter for (p, P_{free}) is a triple (f,**A**,A) where (f,**A**) is the actual parameter for p and A is an **A**-algebra. The result of parameter-passing is indicated to be the commutativity of the following diagram.

$$\begin{array}{ccc} \mathbf{Alg_X} & \xrightarrow{\text{Pfree}} & \mathbf{Alg_D} \\ \bar{F}\uparrow & & \uparrow F' \\ \mathbf{Alg_A} & \xrightarrow{\text{P'free}} & \mathbf{Alg_{D'}} \end{array}$$

Remark: If P_{free} is (strongly) persistent then P'-free is. (technical result in [E/L 81])

We adopt the following convention for the structure of <u>formal-parameter-specifications</u>.

<u>I.3.10 Convention</u>

Let p: $\mathbf{X} \to \mathbf{D}$ be a parameterized specification.
Then the formal parameter $\mathbf{X}$ is one of the following specifications (alternatives are enclosed in brackets { }):

<u>sorts</u>: X

<u>opns</u>: $\{\bar{x}: \to X\}$ (X eventually contains a constant)

$\equiv_X$: X×X → BOOL

<u>eqns</u>: $x \equiv_X x$ = true

$x \equiv_X x'$ = $x' \equiv_X x$

$((x \equiv_X x'\ \&\ x' \equiv_X x'') \Rightarrow x \equiv_X x'')$ = true

Furthermore we shall give each parameterized specification a <u>unique</u> <u>name</u>. This will help us to identify various specifications by their names and to use them as parameters in other parameterized specifications. Thus if p: $\mathbf{X} \to \mathbf{D}$ is a parameterized specification with simple parameter $\mathbf{X}$ we identify the resulting specification by the (unique) name P. The equation

$\mathbf{P} = K(\mathbf{X}_1,\ldots\mathbf{X}_n)$ where K is a <u>constructor</u> (that is a combination of the parameters $\mathbf{X}_1,\ldots,\mathbf{X}_n$ via operations such as '+', '×')

means, that we name the resulting specification by **P**. The formal parameter will be denoted by the keyword <u>formal</u>.
We make the following convention:

I.3.11 Convention

Let p: **X** → **D** be a parameterized specification.
Then the equation **P** = K(**X**) denotes the following specification:

sorts: P ...
  formal X
opns: $\{\bar{p}: \to P\}$
  $\equiv_P$: P×P → BOOL
  . other .
  . opera- .
  . tions .
  formal $\{\bar{x}: \to x\}$
  $\equiv_X$: X×X → BOOL
eqns: $p \equiv_P p = true$
  $p \equiv_P p' = p' \equiv_P p$
  $((p \equiv_P p' \,\&\, p' \equiv_P p'') \Rightarrow p \equiv_P p'') = true$
  . other .
  . equations .
  formal $x \equiv_X x$
  $x \equiv_X x' = x' \equiv_X x$
  $((x \equiv_X x' \,\&\, x' \equiv_X x'') \Rightarrow x \equiv_X x'') = true$

In the case of more than one formal parameter e.g. **P** = K($\mathbf{X}_1, \ldots, \mathbf{X}_n$) this concept can be easily extended by using a tuple of parameterized specifications (p_i: $\mathbf{X}_i$ → **D**| i=1,...,n) and then defining one **Spec**-morphism by this triple.
What we still need is to extend the parameter-passing concept in the case when the parameters are themselves parameterized specifications for parameterized data types. In this case the following definitions will be used:

I.3.11 Definition

Let p: **X** → **D** be a parameterized specification with actual parameter (f,**A**). Let $\hat{p}$: **Y** → **A** be a parameterized specification. Let the result of passing (f,**A**) for **X** be the pushout of the following diagram

$$\begin{array}{ccccc} & & \mathbf{X} & \xrightarrow{p} & \mathbf{D} \\ & & f\downarrow & & \downarrow f' \\ Y & \xrightarrow{\hat{p}} & \mathbf{A} & \xrightarrow{p'} & \mathbf{D}' \end{array}$$

Then the result of passing $\hat{p}$: $\mathbf{Y} \to \mathbf{A}$ for $\mathbf{X}$ via f is given by $p' \circ \hat{p}$.

I.3.12 Definition

Let (p, P_{free}) be (strongly) persistent standard-semantics for p: $\mathbf{X} \to \mathbf{D}$ and $(f, \mathbf{A}, A)$ be an actual parameter. Furthermore let $(\hat{p}, \hat{P}_{free})$ be a (strongly) persistent standard-semantics for $\hat{p}$: $\mathbf{Y} \to \mathbf{A}$.
Then the result of parameter passing is the (strongly) persistent parameterized data type given by
$(p' \circ \hat{p},\ P'_{free} \circ \hat{P}_{free})$.
Now we give some examples for parameterized specifications using the conventions I.3.10 and I.3.11.

Example 1

$\mathbf{P} = \mathbf{X}_1 \times \mathbf{X}_2 \times \ldots \times \mathbf{X}_n \quad (n \in \mathbf{N})$

sorts: P, BOOL
formal $X_1, \ldots, X_n$
opns: $\equiv_p$: $P \times P \to BOOL$
$<\ldots>$: $X_1 \times \ldots \times X_n \to P$
$[i]$: $P \to X_i \quad (i = 1, \ldots, n)$
formal $\equiv_{xi}$: $X_i \times X_i \to BOOL \quad (i = 1, \ldots, n)$
eqns: $p \equiv_p p = true$ — $\equiv_p$ is an equivalence relation on P
$p \equiv_p p' = p' \equiv_p p$
$((p \equiv_p p' \ \&\ p' \equiv_p p'') \Rightarrow p \equiv_p p'') = true$
$[i]<x_1, \ldots, x_n> = x_i \quad (i = 1, \ldots, n)$
$<x_1, \ldots, x_n> \equiv_p <x_1', \ldots, x_n'> = (x_1 \equiv_{x1} x_1' \ \&\ldots\&\ x_n \equiv_{xn} x_n')$
formal $x_i \equiv_{xi} x_i$
$x_i \equiv_{xi} x_i' \ x_i = x_i' \equiv x_i \qquad i = 1, \ldots, n$
$((x_i \equiv_{x'i} \ \&\ x_i' \equiv_{xi} x_i'') \Rightarrow x_i \equiv_{xi} x_i'') = true$

Example 2

$\mathbf{P} = \mathbf{X}_1 \times \mathbf{X}_2 \times \dots \times \mathbf{X}_n + 1$ (n-fold product with constant)

sorts: P, BOOL

  formal $X_1, \dots, X_n$

opns: $\bar{p}: \to P$

$\equiv_p: P \times P \to BOOL$

$<\dots>: X_1 \times \dots \times X_n \to P$

$[i]: P \to X_i \quad (i = 1, \dots, n)$

  formal $\bar{x}_i: \to X_i \quad (i = 1, \dots, n)$

$\equiv_{xi}: X_i \times X_i \to BOOL$

eqns: $p \equiv_p p = true$

$p \equiv_p p' = p' \equiv_p p$

$((p \equiv_p p' \ \& \ p' \equiv_p p'') \Rightarrow p \equiv_p p'') = true$

$<x_1, \dots, x_n> \equiv_p \bar{p} = false$

$\bar{p} \equiv_p <x_1, \dots, x_n> = false$

$[i]\ \bar{p} = \bar{x}_i$

  formal $x_i \equiv_{xi} x_i = true$

$x_i \equiv_{xi} x_i' = x_i' \equiv_{xi} x_i \quad (i = 1, \dots, n)$

$((x_i \equiv_{xi} x_i' \ \& \ x_i' \equiv_{xi} x_i'') \Rightarrow x_i \equiv_{xi} x_i'') = true$

Remark

In the following sections we shall omit the 'equivalence'-part of the specification that is the operations '≡' and the equations stating the equivalence property of '≡'.

Now some examples for parameter passing.
We use the following specification **NAT** for 'natural numbers':

sorts: NAT, BOOL

opns: $0: \to NAT$

$suc: NAT \to NAT$

$\equiv_{NAT}: NAT \times NAT \to BOOL$

eqns: $0 \equiv_{NAT} 0 = true$

$0 \equiv_{NAT} suc(n) = false$

$suc(n) \equiv_{NAT} 0 = false$

$suc(n) \equiv_{NAT} SUC(N') = n \equiv_{NAT} n'$

(This is a quite redundant specification of **NAT** but it is useful in showing parameter-passing!)

Let's turn to our next example:

We want to build **Q = NAT×NAT.**

So we take the specification from Example 1 with

$\mathbf{P} = \mathbf{X}_1 \times \mathbf{X}_2$ and the two Spec-morphisms p_1, p_2 characterized by:

p_1: $\mathbf{X}_1 \to \mathbf{NAT}$, $\equiv_{X1} \to \equiv_{NAT}$, $\bar{x}_1 \to 0$

p_2: $\mathbf{X}_2 \to \mathbf{NAT}$, $\equiv_{x2} \to \equiv_{NAT}$, $\bar{x}_2 \to 0$

We take the tuple $p = \langle p_1, p_2 \rangle$ to lead to the parameterized specification

p: $\mathbf{X}_1 \times \mathbf{X}_2 \to \mathbf{NAT} \times \mathbf{NAT}$

The resulting specification is given by:

<u>sorts</u>: Q, NAT, BOOL

<u>opns</u>: $\bar{q}: \to Q$

$\equiv_Q: Q \times Q \to BOOL$

$\langle\rangle: NAT \times NAT \to Q$

$[1]: Q \to NAT$

$[2]: Q \to NAT$

.

. <u>NAT-opns</u>

.

<u>eqns</u>: $\bar{q} \equiv_Q \bar{q}$

$[1](\bar{q}) = 0$

$[2](\bar{q}) = 0$

$\langle n_1, n_1' \rangle \equiv_Q \langle n_2, n_2' \rangle = (n_1 \equiv_{NAT} n_2 \ \& \ n_1' \equiv_{NAT} n_2')$

.

. <u>equations</u> <u>for</u> <u>NAT</u> .

.

We can now use this specification for building

Q × Q ≙ (NAT × NAT) × (NAT × NAT)

Q + 1 ≙ (NAT × NAT) + 1

etc.

I.4 Algebraic Domain Equations over Spec

We have pointed out in the introduction that an algebraic domain equation is a pair of **Spec**-morphisms
$\mathbf{X} \overset{(p,e)}{\Longrightarrow} \mathbf{D(X)}$.

Now we need two functors P: $\mathbf{Alg_X} \to \mathbf{Alg_D}$ and E: $\mathbf{Alg_D} \to \mathbf{Alg_X}$ to define the endofunctor PoE: $\mathbf{Alg_D} \to \mathbf{Alg_D}$ and then look for fixedpoints.
As certain fixedpoints of PoE should give meaning to 'recursive domain equations' in an algebraic framework it should be obvious that these 'equations' should be defined by a pair of Spec-morphisms p,e: $\mathbf{X} \to \mathbf{D}$. Here p is a parameterized specification and e is used to define the forgetful functor E: $\mathbf{Alg_D} \to \mathbf{Alg_X}$. The following sequence of definitions and theorems formalizes the ideas we have given in the introduction. Detailed proofs will be omitted and can be found elsewhere ([E/L 81], [K 83]).

I.4.1. Definition

Let **X** and **D** be specifications.
An algebraic domain equation (ade), denoted by

$\mathbf{X} \overset{(p,e)}{\Longrightarrow} \mathbf{D}$

consists of a pair of **Spec**-morphisms

p,e: $\mathbf{X} \to \mathbf{D}$

where p is a (strongly) persistent parameterized specification and e describes a forgetful functor.

Remarks: (i) The double-arrow $\overset{(,)}{\Longrightarrow}$ in $\mathbf{X} \overset{(p,e)}{\Longrightarrow} \mathbf{D}$ is used to indicate that p,e are directed from **X** to **D**.

(ii) By I.3.2. we know that each Spec-morphism defines a forgetful functor. As we have indicated in convention I.3.11. each parameterized specification should use unique name s (for sort, constants, equality). Then for e it suffices to map the formal parameter components to the (new) components in the resulting specification which have unique names.

So if the formal parameter is denoted by X and the target specification by P then

$e(X) := P$
$e(\bar{X}) := \bar{p}$
$e(\equiv_X) := \equiv_p$

We have two endofunctors described by p and e, namely

$E \circ P: \mathbf{Alg_X} \to \mathbf{Alg_X}$

and

$P \circ E: \mathbf{Alg_D} \to \mathbf{Alg_D}$

We shall use this later on but the two functors are <u>related</u> in the following way:

I.4.2. Lemma

Let $A \in |\mathbf{Alg_X}|$. Then the following assumption holds:
If A is fixedpoint of $\bar{E} \circ P$ then $|P|(A)$ is a fixedpoint of $P \circ \bar{E}$
and
if B is a fixedpoint of $P \circ \bar{E}$ then $|\bar{E}|(B)$ is a fixedpoint of $\bar{E} \circ P$.

I.4.3. Lemma

B is a fixed point of $P \circ \bar{E}$ iff $|\bar{E}|(B) \cong |\bar{P}|(B)$ and there exists an $\mathbf{X}$-algebra A such that $|P|(A) \cong B$.

The next theorem is very important. It states (informally spoken) that each fixedpoint of $P \circ \bar{E}$ has a certain (syntactically determined) form and that we can restrict the search for fixedpoints to a subcategory of $\mathbf{Alg_D}$. The argumentation is the following:
Let $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D}$ be an ade. We know by I.1.3. that each pair of morphisms in **Spec** has a <u>coequalizer</u>. This coequalizer leads to a renaming of sorts, operations and equations in the specification **D** as shown in the construction of coequalizers is **Spec** (the section following theorem I.1.5.). So we know that (p,e) has a

coequalizer in $\underline{\underline{Spec}}$. This coequalizer consists of a unique specification **Q** and a unique $\underline{\underline{Spec}}$-morphism q: **D** → **Q** (see theorem I.1.3.).

Let coeq(p,e)=(q,**Q**) denote this relation formally. Now it turns out that if $B \in |\underline{\underline{Alg}}_{\mathbf{D}}|$ is a fixedpoint of $P \circ \bar{E}$ then there exists an **Q**-algebra C such that $|\bar{Q}|(C) \cong B$. That means that we may restrict our search for fixedpoints of $P \circ \bar{E}$ to coequalizer-algebras ($\underline{\underline{Alg}}_{\mathbf{Q}}$). And here in coequalizer algebras an identification (renaming) between p- and e-components of **D** has been made. Now we turn to the theorem.

I.4.4. Theorem

Let $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D}$ be an ade and let (q,**Q**) = coeq(p,e).

If B is a fixedpoint of $P \circ \bar{E}$ then there exists a unique **Q**-algebra C such that $B \cong |\bar{Q}|(C)$.

Proof outline:

Our argumentation here is the following:

(i) (q,**Q**) = coeq(p,e). Now the functor **alg**: $\underline{\underline{Spec}} \to \underline{\underline{Cat}}^{op}$ sends each $\underline{\underline{Spec}}$-morphism r: **S** → **T** to the forgetful functor $\bar{R}$:= alg-r: $\underline{\underline{Alg}}_{\mathbf{R}} \to \underline{\underline{Alg}}_{\mathbf{S}}$. **alg** transforms coequalizers in $\underline{\underline{Spec}}$ to equalizers in $\underline{\underline{Cat}}$. If (q,**Q**) = coeq(p,e) in $\underline{\underline{Spec}}$ then $(\bar{Q},\underline{\underline{Alg}}_{\mathbf{Q}})$ = eq($\bar{P}$,$\bar{E}$) in $\underline{\underline{Cat}}$ (where eq() denotes the equalizer).

Thus for each **Q**-algebra C $|\bar{P} \circ \bar{Q}|(C) = |\bar{E} \circ \bar{Q}|(C)$ holds. (equalizer-property!).

(ii) Since B is a fixed-point of $P \circ \bar{E}$ we know that $|\bar{P}|(B) \cong |\bar{E}|(B)$. But we can even construct a **D**-algebra $B' \cong B$ such that $|\bar{P}|(B') = |\bar{E}|(B')$. This looks similar to the equalizer property $|P \circ \bar{Q}|(C) = |E \circ \bar{Q}|(C)$.

(iii) What is still missing is an algebra $C \in |\underline{\underline{Alg}}_{\mathbf{Q}}|$ suchthat $B' \cong |\bar{Q}|(C)$. We show how we can construct a suitable **Q**-algebra C out of B' such that this property holds.

Then we shall have $B \cong B' \cong |\bar{Q}|(C)$ and we are ready.

I.4.5 Corollary

B is a fixedpoint of PoE iff the following conditions hold:

(a) $B \cong |P|(A)$ for a **X**-algebra A

(b) $B \cong |\bar{Q}|(C)$ for a **Q**-algebra C

Each fixedpoint B of PoE has the property $|P|(B) \cong |E|(B)$. This means that all the carriers $B_{p(s)}$, $B_{e(s)}$ $(s \in S_{\mathbf{X}})$ are isomorphic and may therefore be identified. This identification is just what the coequalizer of p and e leads to. Moreover the identification process via coequalizers in **Spec** yields a unique category $\mathbf{Alg}_{\mathbf{Q}}$ which consists in a sense of all **D**-algebras in which isomorphic P and E-parts are identified. Motivated by the introduction and by Theorem I.4.4. we consider solutions of ade's $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D}$ to be **Q**-algebras ((q,**Q**)=coeq(p,e)) which have the fixedpoint property for $Po\bar{E}$, when they are reduced by $\bar{Q}$.

I.4.6. Definition

Let $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D}$ be an ade and (q,**Q**) = coeq(p,e). Then a solution of $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D}$ is a **Q**-algebra C, such that

$$|\bar{Q}|(C) \cong |Po\bar{E}|(|\bar{Q}|(C))$$

($|\bar{Q}|C$ is a fixedpoint of $Po\bar{E}$!)

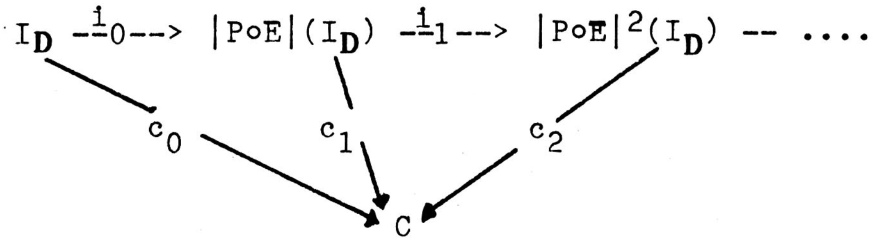
Now according to this definition there may be a variety of **Q**-algebras C which are solutions of $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D}$. But when we started we had in mind to use ade's for implicit specifications of parameterized data types. There ade's should define an (up to isomorphism) unique data type. So the question is: Is there a uniquely determined **Q**-algebra C which is a solution for $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D}$? Indeed:

The initial **Q**-algebra denoted by $I_{\mathbf{Q}}$ is uniquely determined and is a solution of $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D}$!

The following theorems and definitions show the development of this result. We proceed by using an analogon to Scott's inverse limit construction.

I.4.7. Therorem

Let $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D}$ be an ade and let $I_{\mathbf{D}}$ denote the initial **D**-algebra. Furthermore let i_0: $I_{\mathbf{D}} \to |P \circ E|(I_{\mathbf{D}})$ be the unique initial homomorphism and let $i_{k+1} := /P \circ \bar{E}/(i_k)$. Then the colimit **D**-algebra of the diagram



is a fixedpoint of $\bar{E} \circ P$.

Remark: By Theroem I.4.4. there exists a unique (up to isomorphism) **Q**-algebra $A \varepsilon |\underline{Alg}_{\mathbf{Q}}|$ such that

$$C \cong |\bar{Q}|(A).$$

Now we want to prove that the initial **Q**-algebra $I_{\mathbf{Q}}$ is a solution of $\mathbf{X} \overset{(p,e)}{\Rightarrow} \mathbf{D}$. I shall outline the argumentation.

(i) From the fixedpoint property we know: if $B \varepsilon |\underline{Alg}|_{\mathbf{D}}$ is a fixed-point of $P \circ E$ then $|E|(B) \cong |P|(B)$. According to this
fact we get a category $\underline{Iso}$ with:

objects: The class of all pairs (B, β) where B is a fixedpoint of $P \circ \bar{E}$ and β: $|\bar{E}|(B) \to |\bar{P}|(B)$ is the isomorphism for $|\bar{E}|B \cong |\bar{P}|(B)$.

morphisms: f: $(B_1, \beta_1) \to (B_2, \beta_2)$ where f: $B_1 \to B_2$ is an $\underline{Alg}_{\mathbf{D}}$-morphism such that $|P|(f) \circ \beta_1 = \beta_2 \circ |\bar{E}|(f)$.

Note that $\underline{Iso}$ is a full subcategory of $\underline{Alg}_{\mathbf{D}}$. We can consider $\underline{Iso}$ to be the category of all fixedpoints of $P \circ \bar{E}$.

(ii) Theorem I.4.4 shows that for each fixedpoint B of $P \circ \bar{E}$ we can construct a $B' \in \mathbf{Alg_D}$ such that $B \cong B'$ and $|E|(B') = |F|(B')$.

This fact leads to the category Equ with:

objects: all pairs $(B', id_{|E|(B')})$ from Iso.

morphisms: all $f: (B_1, id_{|E|(B1)}) \to (B_2, id_{|E|(B2)})$ such that

$$/\bar{E}/(f) = /\bar{F}/(f). \text{ (because } /\bar{F}/(f).id = \bar{F}/(f) = /\bar{E}/(f) = id \circ /\bar{E}/(F)\text{)}$$

On the other hand since $Q: \mathbf{Alg_Q} \to \mathbf{Alg_D}$ is an equalizer of $\mathbf{E}$ and $\mathbf{F}$, the characterizing property of the morphisms and objects in Equ (where $|E|(B) = |F|(B)$ and $/E/(F) = /F/(F)$) is similar to $\bar{Q}$'s characteristic property namely $\bar{E} \circ \bar{Q} = \bar{F} \circ \bar{Q}$.

These observations culminate in the assumption that Equ and $\mathbf{Alg_Q}$ are isomorphic as categories (provided the range of $\bar{Q}$ is restricted.)

(iii) We had $((c_n | n \in \mathbf{N}_0), C)$ as the colimit of diagram 1 in Theorem I.4.7. The object C was isomorphic to $|P \circ \bar{E}|(C)$. Let $\gamma: |P \circ \bar{E}|(C) \to C$ be this isomorphism. Then the pair $(C, /F/(\gamma))$ is initial in Iso. Since Iso and Equ are equivalent as categories there exists a $\mathbf{D}$-algebra C' isomorphic to C such that $(C', id_{|E|(C')})$ is an object of Equ. Equivalence of Iso and Equ imply that initiality of $(C, /F/(\gamma))$ is respected. Isomorphy of Equ and $\mathbf{Alg_Q}$ then implies that $I_{\mathbf{Q}}$ is a solution of $\mathbf{X} \overset{(p,e)}{\Longrightarrow} \mathbf{D}$ since isomorphy respects initiality. Thus the argumentation is closed.

Now we state the main theorem.

I.4.8. Theorem

Let $\mathbf{X} \overset{(p,e)}{\Longrightarrow} \mathbf{D}$ be an ade where $(q, \mathbf{Q}) = coeq(p,e)$. Then the initial $\mathbf{Q}$-algebra $I_{\mathbf{Q}}$ is a solution.

II. SPECIFICATIONS WITH INEQUALITIES

Hornung [H/R ?] uses specifications with inequalities and gives an initial and a terminal semantics for parameterized data types. We shall use these specifications in the connection with algebraic domain equations. In this chapter we shall define the respective category and we shall outline some properties of this category which are needed to define algebraic domain domain equations.

II.1 Definition

Let $\underline{\Sigma}$ = <S,Σ> be a signature.
A Σ-inequality is a pair <t,t'> with t,t' ε TΣ(X). We write t≠t'.

II.2 Definition

Let $\underline{\Sigma}$ be as above.
A positive conditional Σ-equation is a tuple of pairs

$$<<t_{11},t_{12}>,<t_{21},t_{22}>,\ldots,<t_{n1},t_{n2}>,<t_{n+11},t_{n+12}>>$$

It will be written as

$$t_{11}=t_{12} \;\&\; t_{21}=t_{22} \;\&\; \ldots \;\&\; t_{n1}=t_{n2} \Rightarrow t_{n+11}=t_{n+12}$$

$(t_{ij} \; \varepsilon \; T\Sigma(X)_{si} \; (i=1,\ldots,n+1, \; j=1,2))$

Now we shall define the category of specifications which contain equalities and inequalities in their axiom-part. What is important here is that we have to take care of the definition of our specification-morphisms. It doesn't suffice to use the 'normal' Spec-morphisms in our context since inequalities appear in the axiom part of the respective specifications. We have to send equalities to equalities and inequalites to inequalities.

II.3 Definition

The category of specifications with equalities and inequalities denoted by SpecI is defined by:

$|\underline{\underline{SpecI}}|$: specifications $\mathbf{S} = <S,\Sigma,E>$ with $E = EQ \cup NE$ such that EQ is a set of positive conditional equations as given in II.2 and NE is a set of inequalities as given in II.1

$/\underline{\underline{SpecI}}/$: $f: \mathbf{S} \to \mathbf{S}'$ is a signature morphism $f = <f_{sort}, f_{op}>$ such that
$\forall\, e \in EQ.\ f(e) \in EQ'$
$\forall\, e' \in NE.\ f(e') \in NE'$
(for $\mathbf{S} = <S,\Sigma,EQ \cup NE>$, $\mathbf{S}' = <S',\Sigma',EQ' \cup NE'>$)

For more technical reasons we introduce three relations on $T\Sigma$ which are generated by EQ and NE.

II.3.1 Definition

(i) Let $\mathbf{S} = <S,\Sigma,E> \in |\underline{\underline{SpecI}}|$ with $E = EQ \cup NE$.
Then $\rho_{EQ} \subseteq T\Sigma^2$ is the least (under inclusion) Σ-congruence on $T\Sigma$ such that

if $\forall\ l_1=r_1 \ \&\ldots\&\ l_n=r_n \Rightarrow l_{n+1}=r_{n+1} \in EQ$

then $\bigwedge_{i=1}^{n}(\sigma(l_i),\sigma(r_i)) \in \rho_{EQ} \Rightarrow (\sigma(l_{n+1}),\sigma(r_{n+1})) \in \rho_{EQ}$

(for all $\sigma \in Subst_{\Sigma(X)}$, $(l_i,r_i) \in T\Sigma(X)^2_{si}$ $(i=1\ldots n)$).
(This means that if all premises belong to the congruence the so does the conclusion for any correct substitution)

(ii) ρ_N is the least (under inclusion) relation on $T\Sigma$ which satisfies
$\forall\, s \in S\ \forall (l,r) \in T\Sigma(X)_s\ \forall \sigma \in Subst_{\Sigma(X)}.(l,r) \in NE \Rightarrow (\sigma(l),\sigma(r)) \in \rho_N$

(iii) ρ_{NE} is the least (under inclusion) relation on $T\Sigma$ such that
(a) $\rho_N \subseteq \rho_{NE}$
(b) ρ_{NE} is symmetric
(c) $(r,s) \in \rho_{EQ}\ \&\ (s,t) \in \rho_{NE} \Rightarrow (r,t) \in \rho_{NE}$

Next we introduce the set of distinguishing sorts of a specification in **SpecI**.

II.3.2 Definition

Let **S** = <S,Σ,E> be a specification in SpecI.
The set of distinguishing sorts denoted by DIS is defined by

$$DIS := \{ s \in S \mid \exists\, t,t' \in T\Sigma_S.\ (t,t') \in \rho NE\}$$

Now we shall use I.1.5 to define specifications with some special properties namely consistency, completeness and simplicity.
A specification is consistent, if there are no terms $t,t' \in T\Sigma_S$ which are as well equal ($t,t' \in \rho_{EQ}$) as unequal ($t,t' \in \rho_{NE}$).
Completeness means that each pair of terms $(t,t') \in T\Sigma_s^2$ belongs either to ρ_{EQ} or to ρ_{NE} (for $s \in DIS$).
Furthermore simplicity means that for each equation all the premises are terms of distinguishing sorts.

II.3.3 Definition

Let **S** = <S,Σ,E> ε |SpecI|.

(i) **S** is consistent: $\Leftrightarrow \rho_{EQ} \cap \rho_{NE} = \emptyset$

(ii) **S** is complete: $\Leftrightarrow \forall s \in DIS.\ \rho_{EQ.s} \cup \rho_{NE.s} = T\Sigma_s^2$

(iii) **S** is simple: $\Leftrightarrow \forall\ l_1=r_1 \& \ldots \& l_n=r_n \Rightarrow l_{n+1}=r_{n+1} \in EQ.\quad l_i,r_i \in \bigcup_{d \in DIS} T\Sigma(X)_d \quad (i=1..n)$

As we already pointed out in chapter I the concept of algebraic domain euqations depends on the assumption that the underlying category of specifications is cocomplete. In order to show that ade's can be defined over SpecI we have to prove that this category is again cocomplete. This task is done in the following sequence of assumptions about SpecI.

II.4 Lemma

Spec I has coproducts.

Proof:
The disjoint union of specifications (named as triples of sets) is the coproduct.

II.5 Lemma

Spec I has coequalizers.

Proof:
Let **S** = <S,Σ,E>, **S'** = <S',Σ',E'> (E=EQ u NE, E'=EQ' u NE') be two specifications in SpecI. Futhermore let f,g: **S** → **S'** be two morphisms relating **S** and **S'**. Then the coequalizer-construction follows the usual construction in Spec (equationally defined specifications). Since we have already given an explicit outline of coequalizers in Spec we mention only briefly what to do:

(a) Define the relation $R_{sort} := \{<f_{sort}(s), g_{sort}(s)> \mid s \in S\} \subseteq S' \times S'$ $R_{op} := \{<f_{op}(\sigma), g_{op}(\sigma)> \mid \sigma \in \Sigma\} \subseteq \Sigma' \times \Sigma'$ and define the least equivalences $\mathbf{A}(R_{sort}) \subseteq S'^2$, $\mathbf{A}(R_{op}) \subseteq \Sigma'^2$ defined by these relations.

(b) Rename each equivalence-class in S' by a unique new sort-name. The resulting sort-set will be denoted by $\hat{S}$. In a similar way rename each equivalence-class in Σ' by a unique new operation name. The resulting operation-set will be denoted by $\hat{\Sigma}$.

(c) By (b) we have defined a signature-morphism q': <S',Σ'> → <S",Σ">. Now q' is used to define the specification-morphism q in which is the coequalizer of f and g. Let DIS (DIS') be the sets of distinguishing sorts in S (S').
Now f(DIS) $\subseteq$ DIS' and g(DIS) $\subseteq$ DIS'. Furthermore f(EQ) $\subseteq$ EQ', g(EQ) $\subseteq$ EQ' and f(NE) $\subseteq$ NE', g(NE) $\subseteq$ NE'.

Therefore we can be sure that we shall not identify equalities

and inequalities when we translate the axioms in E´ by q´ and thus having q. The resulting axiom-set will be denoted by $\hat{E} = \hat{EQ} \cup \hat{NE}$ with $\hat{EQ} \cap \hat{NE} = \emptyset$. The set of distinguishing sorts in the new specification will be denoted by $\hat{DIS}$.
Thus the new specification is $\mathbf{Q} := <\hat{S},\hat{\Sigma},\hat{E}>$ and q: $\mathbf{S}' \to \mathbf{Q}$ is the coequalizer of f and g. q is a SpecI-morphism since it transforms distinguishing sorts into distinguishing sorts, equalities to equalities, inequalities into inequalities.

<u>II.6 Lemma</u>

SpecI is cocomplete.

<u>Proof</u>:
Consequence of II.4 and II.5.

In Definition II.3.3 we have pointed out some important properties of specifications, namely consistency, completeness and simplicity.
In the sequel the following questions will be important: Imagine that $\mathbf{S},\mathbf{S}' \in |\text{SpecI}|$ are specifications which are (a) consistent, (b) complete or (c) simple. Suppose f,g: $\mathbf{S} \to \mathbf{S}'$ are two SpecI-morphisms. Can we guarantee that the coequalizer-object in

$$\mathbf{S} \underset{g}{\overset{f}{\rightrightarrows}} \mathbf{S}' \overset{q}{\dashrightarrow} \mathbf{Q}$$

is again (a) consistent, (b) complete or (c) simple?
The answer is: we can guarantee those properties to hold for the coequalizer-object!
In the following lemmata these answers are worked out in detail.

<u>II.7 Lemma</u>

Let $\mathbf{S} = <S,\Sigma.E>$, $\mathbf{S}' = <S',\Sigma',E'> \in |\text{SpecI}|$ be specifications and let f,g: $\mathbf{S} \rightarrow \mathbf{S'}$ be two <u>SpecI</u>-morphisms.

Furthermore let the pair (q,**Q**) denote the coequalizer of f and g (with $\mathbf{Q} = <\hat{S},\hat{\Sigma},\hat{E}>$ as in the proof of II.5) according to the diagram

$$\mathbf{S} \overset{f}{\underset{g}{\rightrightarrows}} \mathbf{S'} \overset{q}{\dashrightarrow} \mathbf{Q}$$

If **S** and **S'** are consistent then **Q** is.

<u>Proof</u>:

We have to show that $\rho_{\hat{EQ}} \cap \rho_{\hat{NE}} = \emptyset$.

By consistency of **S** and **S'** we have $\rho_{EQ} \cap \rho_{NE} = \emptyset = \rho_{EQ'} \cap \rho_{NE'}$ and furthermore since f and g are <u>SpecI</u>-morphisms:

$$f(\rho_{EQ}) \cap f(\rho_{NE}) = \emptyset = g(\rho_{EQ}) \cap g(\rho_{NE})$$

Now the only way to generate inconsistency would be the application of q.

But since q is a <u>SpecI</u>-morphism we have

$$\forall\ e' \varepsilon EQ'.\ q(e)\ \varepsilon\ \hat{EQ}$$

and

$$\forall\ e'\ \varepsilon\ NE'.\ q(e')\ \varepsilon\ \hat{NE}$$

But q sends DIS'-terms to $\hat{DIS}$-terms and non DIS'-terms to non-$\hat{DIS}$-terms. Thus is must be the case that

$\rho_{\hat{EQ}} \cap \rho_{\hat{NE}} = \emptyset$ and **Q** is consistent.

<u>II.8 Lemma</u>

Let f,g: $\mathbf{S} \rightarrow \mathbf{S'}$ and (q,**Q**) be as in II.7.

If **S** and **S'** are complete then **Q** is.

<u>Proof</u>:

That **S** and **S'** are complete specifications means that

(i) $\forall\ s \varepsilon DIS.\ \rho_{EQ,s} \cup \rho_{NE,s} = T\Sigma_s^2$

and

(ii) $\forall\ s' \varepsilon DIS'.\ \rho_{EQ',s'} \cup \rho_{NE',s'} = T\Sigma'^2_{s'}$

Now we have to show that

$$\forall \hat{s} \varepsilon \hat{DIS}.\ \rho_{\hat{EQ},\hat{s}} \cup \rho_{\hat{NE},\hat{s}} = \hat{T\Sigma}^2_{\hat{s}}$$

This is equivalent to

(*) $\forall s' \varepsilon DIS'.\ q(\rho_{EQ'},s') \cup q(\rho_{NE'}.s') = \hat{T\Sigma}^2_{q(s')}$

Now for each $S' \varepsilon DIS' - (f_{sort}(DIS) \cup g_{sort}(DIS))$ the assumption (*) is clearly satisfied. But again for each $s' \varepsilon DIS' \cap (f_{sort}(DIS) \cup g_{sort}(DIS))$ the assumption (*) holds since q is surjective.

II.9 Lemma

Let f,g: **S** → **S'** and (q,**Q**) be as in II.7.
If **S**,**S'** are simple then **Q** is.

Proof:

Simplicity means that all terms in the premises of $\hat{EQ}$ have to be of distinguishing sorts $\hat{DIS}$. But this is obvious due to the fact that q transforms DIS' to $\hat{DIS}$. Therefore terms of dstinguishing sorts in $T\Sigma'$ are translated into terms of distinguishing sorts in $\hat{T\Sigma}$. Thus **Q** is simple.

II.10 Corollary

Let f,g: S → **S'** and (q,**Q**) be as in II.7. If **S**,**S'** are consistent, complete and simple then **Q** is.

Proof:

Obvious from II.7 - II.10.

Specifications are syntactic entities which are used to define data types. In the algebraic approach data types are viewed as heterogenous algebras. In normal equationally defined specifications the models are those algebras which satisfy the equations. In the case of Spec1 where specifications may contain equalities and inequalities we have to ensure that two terms $(t,t') \varepsilon \rho_{NE}$ (which are different in the term-algebra $T\Sigma$) will not be identified in the respective model A e.g. $t_A \neq t'_A$. (Here

t_A, t'_A are the interpretations of t,t′ in the algebra A).
Thus we use a slightly different definition for the category of **S**-models (**S** ε |$\underline{\underline{SpecI}}$|).

II.11 Definition

Let **S** = <S,Σ,E> be a specification in $\underline{\underline{SpecI}}$. Then the category of **S**-algebras $\underline{\underline{Alg}}_{\mathbf{S}}$ is defined by

|$\underline{\underline{Alg}}_{\mathbf{S}}$| : all $\underline{\Sigma}$-algebras A with

(i) A satisfies the equations in EQ = E - NE

(ii) $\forall\, t,t' \varepsilon T\Sigma.\ (t,t')\varepsilon\rho_{NE} \Rightarrow t_A \neq t'_A$

/$\underline{\underline{Alg}}_{\mathbf{S}}$/ : all $\underline{\Sigma}$-homomorphisms on |$\underline{\underline{Alg}}_{\mathbf{S}}$|.

For consistent specifications the respective category of models is always non-empty and contains an initial object.

II.12 Lemma

Let **S** = <S,Σ,E> be a specification in $\underline{\underline{SpecI}}$. If **S** is consistent then $\underline{\underline{Alg}}_{\mathbf{S}}$ is nonempty and $T\Sigma/_{\rho EQ}$ is the initial object in $\underline{\underline{Alg}}_{S}$.

Proof: see [H/R ?]

For the following discussion we consider a slightly modified notion of parameterized data types. It is modified in the sense that we take directly into account free (initial) extensions of argument algebras as results of applying data type constructors.

II.13 Convention

In the following discussion let **S** = <S,Σ,E>, **S1** = <S1,Σ1,E1> and **S′** = <S′,Σ′,E′> be $\underline{\underline{SpecI}}$-objects such that
S ∩ S1= Σ ∩ Σ1= E ∩ E1 = ∅ and **S′** := **S** + **S1** := <S+S1,Σ+Σ1,E+E1>
(where ′+′ denotes the disjoint union)

According to this convention we are able to say what we mean by parameterized specification.

II.14 Definition

Let **S**,**S1**,**S'** be as in II.13 where E1 does not contain inequalities.
Then a parameterized specification is an injective ~~SpecL~~-morphism p: **S** → **S'**.

This is similar to the definition used in [E/L 81]. For our purposes it suffices to turn to a special case of II.14 namely that p: **S** → **S'** is the inclusion-morphism between **S** and **S'** (S' is an extension of **S**).

II.15 Definition (alternative to II.14)

Given the premises of II.13 we define a parameterized specification p: **S** → **S'** to be the inclusion-morphism between **S** and **S'**.

II.16 Definition

Let **S**,**S1**,**S'** be given as in II.13 Let p: **S** → **S'** be a parameterized specification.
Then by a parameterized data type (specified by p) we mean a (strongly) persistent functor P: $\underline{\underline{Alg}}_{\mathbf{S}} \rightarrow \underline{\underline{Alg}}_{\mathbf{S'}}$ such that
$\forall A \in \underline{\underline{Alg}}_{\mathbf{S}}. \ |\mathrm{F} \circ P|(A) \cong A \quad (|\mathrm{F} \circ P|(A) = A)$

II.17 Definition

Given the premises in II.16 with the parameterized specification p: **S** → **S'**.
Then by the standard semantics of p we mean a pair (p,P) where P: $\underline{\underline{ALG}}_{\mathbf{S}} \rightarrow \underline{\underline{ALG}}_{\mathbf{S'}}$ is a functor as given in II.16 (p,P) is (strongly) persistent, if P is.

II.18 Definition

Let **S** = <S,Σ,E>, **S1** = S + <S1,Σ1,E1> be consistent specifications in **SpecI**. Let A ε |**Alg**$_{\mathbf{S}}$|.

$\equiv_{E1,A}$ is the smallest Σ + Σ1 congruence on T(Σ+Σ1) such that

(a) $\equiv_A \subseteq \equiv_{E1,A}$

(b) $\forall l_1=r_1 \& l_2=r_2 \& \ldots \& l_n=r_n \Rightarrow l_{n+1}=r_{n+1} \varepsilon E_1 \ \forall \sigma \varepsilon Subst_{\Sigma(X)}.$
$\forall_{i=1} i.\ \sigma(l_i) \equiv_{E1,A} \sigma(r_i) \Rightarrow \sigma(l_{n+1}) \equiv_{E1,A} \sigma(r_{n+1})$

II.19 Definition

Let **S** = <S,Σ,E> and **S′** = <S′,Σ′,E′> be consistent specifications with S ⊆ S′, Σ ⊆ Σ′, E ⊆ E′.

S′ is an i-extension of **S** iff $T\Sigma'/_{\equiv EQ'}|_\Sigma \cong T\Sigma/_{\equiv EQ}$ (this means that no new elements and new identifiers are introduced by extending $T\Sigma/_{\equiv EQ}$.)

II.20 Theorem

Let **S** = <S,Σ,E>, **S1** = <S1,Σ1,E1>, **S′** = <S′,Σ′,E′> = <S+S1,Σ+Σ1,E+E1> be consistent specifications in SpecI such that p: **S** → **S′** is a parameterized specification.
Let P: **Alg**$_{\mathbf{S}}$ → **Alg**$_{\mathbf{S'}}$ be the following functor

(i) ∀Aε|**Alg**$_{\mathbf{S}}$|. $|P|(A) := T\Sigma'/_{\equiv E1,A}$

(ii) ∀A,Bε|**Alg**$_{\mathbf{S}}$| ∀hε**Alg**$_{\mathbf{S}}$(A,B)./P/(h): |P|(A) → |P|(B)

Then $|P|(T\Sigma_{\equiv EQ}) = T\Sigma'/_{\equiv EQ'}$ and **S′** is an i-extension of **S**.

Proof ([H/R ?] 3.2.3 and 3.2.4).

III. CONSISTENT SPECIFICATIONS AND ALGEBRAIC DOMAIN EQUATIONS

As a 'minimal' additional requirement for our further discussion we require our specifications to be consistent. Thus it cannot be the case that two terms in our term-algebra $T\Sigma$ are as well equal as unequal. We turn our attention to a subcategory of SpecI namely the category of consistent specifications.
Thus we make the following definition.

III.1 Definition and Lemma

Let SpecIC (consistent specifications with inequalities) be defined by:

|SpecIC|: all specifications **S** = <S,Σ,EQ u NE> (ε |SpecIC|) such that $\rho_{EQ} \cap \rho_{NE} = \emptyset$
(consistent specifications)

/SpecIC/: all f:**S** → **S**´ (ε /SpecI/) with **S** = <S,Σ,EQ u NE>, **S**´ = <S´,Σ´,EQ´ u NE´> such that
$f_{sort}^{-1}(DIS') = DIS$ (bijective) and $f(\rho_{NE}) = \rho_{NE'}$
Then SpecIC is a subcategory of SpecI.

Proof:

(i) SpecIC is a category since for each **S** ε |SpecIC| there clearly exists an identity morphism $id_{\mathbf{S}}$: **S** → **S** and for two morphisms
f: **S** → **S**´, g: **S**´ → **S**"
(**S** = <S,Σ,EQ u NE>, **S**´,Σ´,EQ´ u NE´>,
S" = <S",Σ",EQ" u NE">) there exists the composition morphism gof: **S** → **S**" since

(*) $g_{sort}^{-1}(DIS") = DIS'$ and $f_{sort}^{-1}(DIS') = DIS$
$\Rightarrow (g_{sort} \circ f_{sort})^{-1}(DIS") = f_{sort}^{-1} \circ g_{sort}^{-1}(DIS")$
$= f_{sort}^{-1}(DIS') = DIS$

and

(**) $g(\rho_{NE'}) = \rho_{NE''}$ and $f(\rho_{NE}) = \rho_{NE'}$

$\Rightarrow g \circ f(\rho_{NE}) = g(\rho_{NE'})$

$= \rho_{NE''}$

as required by the definition of /**SpecIC**/.

(ii) We still have to show that **SpecIC** is a subcategory of **SpecI**, but by definition of **SpecIC** we have

(*) $|\mathbf{SpecIC}| \subseteq |\mathbf{SpecI}|$

and

(**) $/\mathbf{SpecIC}/ \subseteq /\mathbf{SpecI}/$

Now what we want to show is that algebraic domain equations can be defined over **SpecIC** and that they have again a unique solution defined by means of a coequaliyer-algebra.
Thus we first say what an algebraic domain equation is in **SpecIC**:

III.1.1 <u>Definition</u>

An algebraic domain equation over **SpecIC** is a pair

$p,e\colon \mathbf{S} \to \mathbf{S'}$ (written) $\mathbf{S} \overset{(p,e)}{\Rightarrow} \mathbf{S'}$

where $p\colon \mathbf{S} \to \mathbf{S'}$ is a parameterized specification and $P\colon \mathbf{Alg}_{\mathbf{S}} \to \mathbf{Alg}_{\mathbf{S'}}$ the respective strongly persistent functor. The morphism e defines again the forgetful functor $alg\text{-}e\colon \mathbf{ALG}_{\mathbf{S'}} \to \mathbf{Alg}_{\mathbf{S}}$.
As Ehrich and Lipeck indicated in [E/L 82] an approach to ade's which uses another category of specifications than **Spec** must satisfy the following requirements:

R1: <u>The respecitive category of specifications must be cocomplete.</u>

R2: <u>Each specification **S** in the respective category of specifications must be such that $\mathbf{Alg}_{\mathbf{S}}$ has an initial object.</u>

R3: <u>Let $f\colon \mathbf{S} \to \mathbf{S'}$ be a **SpecIC** morphism. Then the respective forgetful-functor</u>

alg-f: $\underline{\underline{Alg}}_{\mathbf{S}'} \to \underline{\underline{Alg}}_{\mathbf{S}}$
has a left-adjoint.

R4: The functor
alg: $\underline{\underline{SpecIC}} \to \underline{\underline{Cat}}^{op}$
respects coequalizers and pushouts (that is alg respects colimits)

We proceed by showing that requirements **R1** - **R4** are satisfied by our construction of SpecIC and that the application of ade's on consistent specifications with inequalities works well.

III.2 Lemma

$\underline{\underline{SpecIC}}$ is cocomplete.

Proof:
By Lemma II.6 we know that $\underline{\underline{SpecI}}$ is cocomplete.
But since $\underline{\underline{SpecIC}}$ is a subcategory of $\underline{\underline{SpecI}}$ we conclude that again $\underline{\underline{SpecIC}}$ is cocomplete.

Now we look for our second requirement **R2.**

III.3 Lemma

Let $\mathbf{S}$ = <S,Σ,EQ u NE> ε $|\underline{\underline{SpecIC}}|$
Then $\underline{\underline{Alg}}_{\mathbf{S}}$ has an initial object.

Proof:
This assumption is exactly Lemma II.12. Let's now turn to the second half of our requirements which deal with properties of the functor alg: $\underline{\underline{SpecIC}} \to \underline{\underline{Cat}}^{op}$ which is defined by

III.4 Definition

alg: $\underline{\underline{SpecIC}} \to \underline{\underline{Cat}}^{op}$ is the following functor:

$\forall$ **S** ε |SpecIC|.|alg|(**S**) := $\underline{Alg}_{\mathbf{S}}$

$\forall$ f:**S** $\rightarrow$ **S** ε /SpecIC/.alg-f: $Alg_{\mathbf{S}'} \rightarrow Alg_{\mathbf{S}}$

is the forgetful functor which takes each Σ',E'-algebra A' to it's Σ,E-reduct in $\underline{Alg}_{\mathbf{S}}$.

(**S** = <S,Σ,E>, **S'** = <S',Σ',E'>).

Now we have to show that for each f: **S** $\rightarrow$ **S'** (which is a <u>parameterized</u> specification) (f ε /SpecIC/) the forgetful functor alg-f: $Alg_{\mathbf{S}'} \rightarrow Alg_{\mathbf{S}}$ has a left-adjoint. It suffices to show that there exists a functor F: $\underline{Alg}_S \rightarrow \underline{Alg}_{S'}$ which takes each **S**-algebra A to it's <u>free</u> <u>i-extension</u> |F|(A) (as given by the construction in Theorem II.20). Since F is determined by f we shall often use free-f instead of F.

III.5 <u>Lemma</u>

Let f: **S** $\rightarrow$ **S'** be a (strongly) persistent parameterized specification in SpecIC.

The the forgetful functor

alg-f: $\underline{Alg}_{\mathbf{S}'} \rightarrow \underline{Alg}_{\mathbf{S}}$

has a left adjoint

F := free-f: $\underline{Alg}_{\mathbf{S}} \rightarrow \underline{Alg}_{\mathbf{S}'}$

<u>Proof</u>:

Define F := |free-f|: $\underline{Alg}_{\mathbf{S}} \rightarrow \underline{Alg}_{\mathbf{S}'}$ as given by II.20.

That means that each **S**-algebra A is sent to it's free i-extension in $\underline{Alg}_{\mathbf{S}'}$.

Now for each A ε |$\underline{Alg}_{\mathbf{S}}$| there exists a homomorphism

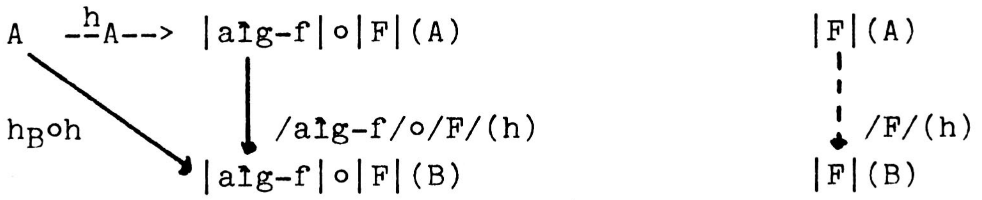
h_A: A $\rightarrow$ |alg-f∘F|(A)

(clearly due to Theorem II.20 we have that |alg-f∘F|(A) $\cong$ A)

Now given A,B ε |$\underline{Alg}_{\mathbf{S}}$| and h: A $\rightarrow$ B (Σ-homomorphism) we define the <u>morphism-part</u> of F

/F/(h): |F|(A) $\rightarrow$ |F|(B)

such that the following diagram commutes



(Clearly the definition is unique according to the fact that $h_B o h$ is well defined.

Thus

F: $\mathbf{Alg}_{S}$ → $\mathbf{Alg}_{S'}$

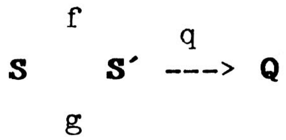
is a left-adjoint of alg-f.

Now let's turn to our last requirement namely that alg: SpecIC → CAT^{op} respects colimits. We show this by proving that alg sends coequalizers in SpecIC to equalizers in Cat^{op} and by indicating that pushouts in SpecIC are sent to pullbacks in Cat^{op}.

III.6 Lemma

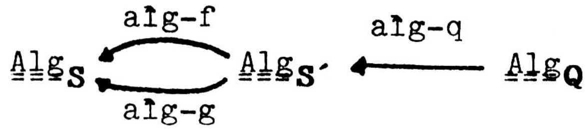
Let f,g: **S** → **S'** be morphisms in SpecIC.
Then alg: SpecIC → Cat^{op} respects to coequalizers of f and g.

Proof:
Since SpecIC is cocomplete we know that the coequalizers of f and g does exist in SpecIC. It will be denoted by
coeq(f,g) =: (q,**Q**).
This situation is represented by diagram **D1**



Now we must show that alg sends coequalizers in SpecIC to equalizers in Cat^{op}. This means that the diagram **D2** must be an equalizer-diagram in Cat^{op}.

D2



By our definition of SpecIC we know that no SpecIC-morphism does introduce any new inequalities into it's target specification. (The inequalities in the source may only be renamed by the respective morphims.) Thus according to diagram **D2** the respective relations ρ_{NE}, $\rho_{NE'}$ and $\rho_{NE''}$ are all isomorphic.

(Here ρ_{NE} is generated by the inequalities in **S**

$\rho_{NE'}$ is generated by the inequalities in **S′**

$\rho_{NE''}$ is generated by the inequalities in **Q**)

Thus the inequalities will not cause troubles in our analysis. Now for proving that diagram **D2** belongs to an equalizer-situation in CatoP we have to observe what the forgetful functors alg-q, alg-g, alg-f do with the carrier-sets of algebras in the respective source-categories.

(i) alg-q:

Let C be a **Q**-algebra and let B be a **S′**-algebra such that

$B = |alg\text{-}q|(C)$

(*) Then we have for each $s' \in sorts(\mathbf{S'})$ with

$s' \notin \{f_{sort}(s) \mid s \in sorts(\mathbf{S})\} \cup \{g_{sort}(s) \mid s \in sorts(\mathbf{S})\}$

the fact that

$B_{s'} := C_{s'}$

(**) For each $s' \in sorts(\mathbf{S'})$ with

$s' \in \{f_{sort}(s) \mid s \in sorts(\mathbf{S})\} \cup \{g_{sort}(s) \mid s \in sorts(\mathbf{S})\}$

we have

$B_{s'} := C_{\hat{s}}$ where

$\hat{s} := [f_{sort}(s), g_{sort}(s)]$

according to the coequalizer construction for f,g.

(ii) alg-f, alg-g:

Now let $A, A' \in |\mathbf{Alg}_{\mathbf{S}}|$ with $a := |\ alg\text{-}f|(B)$ and

$A' := |alg\text{-}g|(B)$. According to (1) we have
$\forall\, s \in sorts(\mathbf{S}).\ A_s := B_{fsort(s)} = B_{gsort(s)} =: A'_s$.
Therefore

$$
\begin{aligned}
|alg\text{-}f| \circ |alg\text{-}q|(C) &= |alg\text{-}f|(B) \\
&= |alg\text{-}g|(B) \\
&= |alg\text{-}g| \circ |alg\text{-}q|(C)
\end{aligned}
$$

as required for the equalizer-property of **D2.** Moreover ($\mathbf{Alg_Q}$,alg-q) is uniquely determined by construction.
It follows that ($\underline{Alg}_{\mathbf{Q}}$,alg-q) is the equalizer for diagram **D2.**

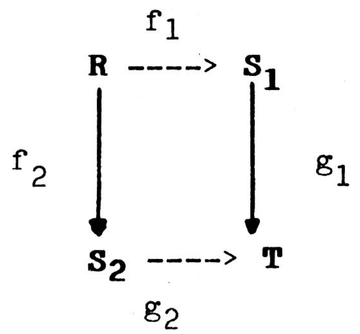
III.7 Lemma

Let f_1: $\mathbf{R} \to \mathbf{S_1}$ and f_2: $\mathbf{R} \to \mathbf{S_2}$ be morphisms in SpecIC.
Then alg respects the pushout of f_1, f_2.

Proof:

Since SpecIC is cocomplete the pushout of f_1, f_2 exists in SpecIC. Diagram **D3** shows the respective situation:

D3:



We have to show that diagram **D4** corresponds to a pullback situation in CatoP.

D4:

$$
\begin{array}{ccccc}
 & & \widehat{\mathrm{alg}}\text{-}f_1 & & \\
 & \mathbf{Alg}_{\mathbf{R}} & \longleftarrow & \mathbf{Alg}_{\mathbf{S1}} & \\
\widehat{\mathrm{alg}}\text{-}f_2 & \uparrow & & \uparrow & \widehat{\mathrm{alg}}\text{-}g_1 \\
 & \mathbf{Alg}_{\mathbf{S2}} & \longleftarrow & \mathbf{Alg}_{\mathbf{T}} & \\
 & & \widehat{\mathrm{alg}}\text{-}g_2 & &
\end{array}
$$

Again by definition of SpecIC we have that $\mathbf{R}, \mathbf{S}_1, \mathbf{S}_2, \mathbf{T}$ have isomorphic sets of inequalities. Thus this will not cause further troubles.

But then it is clear that we may restrict our attention to the functor

$\mathrm{alg}: \mathbf{Spec} \rightarrow \mathbf{Cat}^{op}$

(which is used by Ehrich and Lipeck in their original work on ade´s).

And moreover we know that alg respects pushout. Thus we may conclude that $\widehat{\mathrm{alg}}$ does.

Now we come to our main result, namely that ade´s are defineable over **SpecIC** and that they have again a unique solution defined by means of a coequalizer-algebra.

III.8 Theorem

Let $\mathbf{S} \overset{(p,e)}{\Longrightarrow} \mathbf{S}'$ be an algebraic domain equation as defined in II.1.1. Then this equation has a unique solution namely

$|\bar{Q}|(I_Q)$

(where $(q,Q) = \mathrm{coeq}(p,e)$)

Proof:

The proof is a consequence of the proceeding lemmata.

Bibliography

[ADJ 82]: Thatcher, J.W.; Wagner, E.G.; Wright, J.B.
Data Type Specification: Parameterization and the Power of Specification Techniques
ACM Toplas
Vol. 4, No. 4, 1982, pp. 711-732

[B/G 80]: Burstall, R.M.; Crogue, J.A.
The Semantics of Clear,
A Specification Language
Internal Report CSR-65-80, 1980
Department of Computer Science
University of Edinburgh

[E/L 81]: Ehrich, H.-D.; Lipeck, U.
Algebraic Domain Equations
Forschungsbericht Nr. 125, 1981
Abteilung Informatik
Universität Dortmund

[E/L 82]: Ehrich, H.-D.; Lipeck, U.
Ergänzung zu 'Algebraic Domain Equations', 1982
Abteilung Informatik
Universität Dortmund

[H/R ?]: Hornung, G.; Raulefs, P.
Initial and Terminal Algebra Semantics of Parameterized Abstract Data Type Specifications with Inequalities, ?
Institut für Informatik III
Universität Bonn.

[K 83]: Krützer, G.
An Approach to Parameterized

Continuous Data Types
Memo-SEKI-83-11
Fachbereich Informatik
Universität Kaiserslautern

[Mi 77]: Milner, R.
Flowgraphs and Flow Algebras
Internal Report CSR 5-77, 1977
Department of Computer Science
University of Edinburgh

[Mö 82]: Möller, B.
Unendliche Objekte und Geflechte
TUM-I8213,1982
Institut für Informatik
Technische Universität München

[Sc 72]: Scott, D.S.
Contiuous Lattices
Springer Lecture Notes in Mathematics
Vol. 274, 1972, pp. 97-136

[Sc 76]: Scott, D.S.
Data Types as Lattices
SIAM Journal of Computing
Vol. 5, 1976, pp. 522-587