# Generated models and the ω-rule: the nondeterministic case

*Michal Walicki* [*)] *Sigurd Meldal*
michal@ii.uib.no sigurd@ii.uib.no
University of Bergen
Department of Informatics
HiB, 5020 Bergen, NORWAY

**Abstract.** A language for specifying nondeterministic operations which generalizes the equational specification language is introduced. Then, various notions of generated multimodels are discussed and sufficient conditions for the existence of quasi-initial semantics of nondeterministic specifications are given. Two calculi are introduced: NEQ and NIP. The former is sound and complete with respect to the class of all multimodels. The latter is an extension of the former with the ω-rule. It is sound and complete with respect to one of the classes of the generated multimodels. The calculi reduce to the respective deterministic calculi whenever the specification involves only deterministic operations.

## 1. Introduction

The notion of *nondeterminism* arises naturally in describing concurrent systems. Various approaches to the theory and specification of such systems, for instance, CCS [23], CSP [16], process algebras [1], event structures [39], include the phenomenon of nondeterminism. But nondeterminism is also a natural concept in describing sequential programs, as witnessed for instance, by the powerdomain constructions of denotational semantics [28, 31, 29, 4] and by attempts to extend abstract data types with nondeterministic operations [3, 2, 25, 26].

As we argued elswhere [35, 38] the use of nondeterministic operators is an appropriate and useful abstraction tool, and more· nondeterminism is a *natural* abstraction concept whenever there is a hidden state or other components of a system description which are, methodologically, conceptually or technically, inaccessible at a particular level of abstraction. Whether the world really is nondeterministic or not we leave to the physicists and philosophers to ponder. A computer system in isolation certainly is deterministic: When started from a particular state (given in full detail) twice, both executions will demonstrate identical behavior. Possible sources of perceived nondeterminism lie only in the unpredictability of the environment such as hardware failures or human factors. Considering all such factors as parts of the total state given in full detail may obviate the perceived nondeterminism, but leads to undesirable complexity and is possible only in principle.

The primary argument in favor of accepting nondeterministic operators is instrumental, and identical to the credo of the abstract data type community: *One should specify a system only in such detail that any implementation satisfying the specification also satisfies the user, and no more.* It turns out that nondeterministic operators ease the process of specifying systems by allowing one to disregard irrelevant aspects – be they the external influ-

[*)] This work has been partially supported by the Architectural Abstraction project under NFR (Norway), by CEC under ESPRIT–II Basic Reearch Working Group No. 6112 COMPASS, by the US DARPA under ONR contract N00014-92-J-1928, N00014-93-1-1335 and by the US Air Force Office of Scientific Research under Grant AFOSR-91-0354.

ences or implementation details – and thus reducing the danger of overspecification resulting from technical rather than methodical reasons.

In recent years several authors have tried to provide algebraic specifications of nondeterminism with the semantics based on the notion of *multialgebras* [22, 15, 17, 24, 21]. It is also the approach we are taking here. Multialgebras seem to offer a natural generalization needed for the adequate treatment of nondeterminism. However, since a multialgebra models the nondeterministic terms as *sets* of individuals, the concept of reachability of such a model remains unclear. For instance, is an element which can be returned as one of the possible results of a nondeterministic operation, but which is not a unique interpretation of any (deterministic) term, to be considered reachable or not? The logical counterpart of the deterministic generated models is the induction principle. What form should it have if we want to apply it to multialgebras? And which multialgebraic models should we choose in order to obtain sound inductive reasoning? In this paper we intend to answer some of these questions.

Let SP denote an equational specification; Mod(SP) the class of its models, $\mathrm{Mod}_{\Sigma}(\mathrm{SP})$ the generated models, $T_{SP}$ an initial model; SP $\vdash_{CAL} e$ provability of an equation $e$ from the axioms of SP with a calculus CAL. Among the fundamental results for the deterministic equational specifications there are the following theorems:

1. $\mathrm{Mod(SP)} \models e$ iff $\mathrm{SP} \vdash_{EQ} e$ – soundness and completeness of equational calculus
2. $\mathrm{Mod}_{\Sigma}(\mathrm{SP}) \models e$ iff $\mathrm{SP} \vdash_{IP} e$ – soundness and completeness of induction principle
3. $T_{SP} \in \mathrm{Mod}_{\Sigma}(\mathrm{SP})$ – initial models are generated

We present the analogous theorems for the specifications involving nondeterministic operations. Section 2 introduces the specification language $\mathcal{L}$ and discusses briefly its main features. Section 3 defines the multialgebraic semantics of specifications in $\mathcal{L}$. Then we discuss various notions of generated multimodels (section 4), suggest a generalization of the initial semantics to the nondeterministic context and give the sufficient condition for the existence of such semantics (section 5), and present the results analogous to 3. (section 5), 1. (section 6), and 2. (section 7). All the constructions and theorems are *non-intrusive* in the sense that, when restricted to the deterministic context, they specialize to the above (and other) standard results.

The present paper is a self-contained version of some of the results from [36]. The reader interested in the motivation for some unorthodox decisions we have made here, as well as the explanations of some technical details is referred to [36] where he can also find the proofs omitted here.

## 2. The specification language

A specification is a pair $(\Sigma, \Pi)$, where the *signature* $\Sigma$ is a pair of a sets (S, F) of sorts S and operation symbols F (with arguments and result sorts in S). There exists a denumerable set $\mathcal{V}$ of variables for every sort. For any syntactic entity (term, formula, set of formulae) $\chi$, $\mathcal{V}[\chi]$ will denote the set of variables in $\chi$. Letters from the end of the latin alphabet, $x,y,z$ are used for variables. The set of terms over the signature $\Sigma$ and a variable set X is denoted $W_{\Sigma,X}$. We always assume that, for every sort S, the set of ground terms of sort S, $(W_{\Sigma})_S$, is not empty. [1]

Formulae of the language $\mathcal{L}$ are *clauses* and $\Pi$ is a set of such formulae. Each clause $C$ is a set of atomic formulae (i.e., the ordering and multiplicity of the atomic formulae

[1] We do not address the problem of empty sorts here and will present calculi which work under the assumption that sorts are not empty. We will usually give signatures with at least one constant for every sort, but other means of restricting the signatures [19, 10] or ensuring nonemptiness [10, 12] can be used instead. It seems also that the most flexible approach which generalizes calculus by introducing explicit variables [10, 11, 7] can be adapted to our framework.

do not matter), written as $a_1,\ldots,a_n$ (possibly within "{ }"). To ease readability we will sometimes write disjunction explicitly as $a_1\vee\ldots\vee a_n$. Comma indicates concatenation of clauses, and semicolon a collection. E.g., if $C$ is $\{c_1,\ldots,c_n\}$, $D$ is $\{d_1,\ldots,d_m\}$, then "$C$ ; $D$" denotes the conjunction of the two clauses, while "$C$, $D$" denotes the clause $\{c_1,\ldots,c_n,d_1,\ldots,d_m\}$. Usually, we use uppercase Latin letters for single clauses, uppercase Greek letters for sets of clauses, and lowercase Latin letters $c$, $t$ for terms.

An atomic formula (atomic clause) is either an *equation*, $t\doteq s$, an *inclusion*, $t\prec s$, or an *inequalities*, $t\#s$, of terms $t$, $s\in W_{\Sigma X}$. Equalities are understood to mean *necessary* equality (i.e. the two terms *always* return the same result). Inequalities are understood to mean *necessary* inequality (i.e. the two terms *never* return the same results). Equations and inclusions are called *positive* atoms, and inequalities *negative* atoms. This distinction expresses the fact that a clause $\{ t\#s, p\prec r \}$ is equivalent to the conditional formula $\neg(t\#s) \Rightarrow p\prec r$, stating that whenever $t$ intersects $s$, $p$ is included in $r$. A clause with exactly one positive atom is called a *Horn formula*, and a Horn formula with no negative atoms a *simple formula*.

All variables occurring in a clause are implicitly universally quantified over the whole clause. A clause is satisfied if, for every assignment to the variables, at least one of the atoms is true. For a specification SP=$(\Sigma, \Pi)$, $\mathcal{L}$(SP) is the restriction of $\mathcal{L}$ to $W_{\Sigma,\mathcal{V}}$.

A specification SP is interpreted in a multialgebra where the (nondeterministic) operations correspond to *set-valued functions*. The main point concerning the interpretation of $\mathcal{L}$ is related to the meaning of equality. Since terms are interpreted in a multialgebra as sets of the possible results, the formula $t=s$ is usually interpreted as the equality of *the sets corresponding to all possible results* of the operations. This gives a model which is mathematically plausible but does not correspond to our operational intuition. The equality $t=s$ does not guarantee that the result returned by some particular application of $t$ will actually be equal to the result returned by an application of $s$. It merely tells us that *in principle* (in all possible executions) any result produced by $t$ can also be produced by $s$ and vice versa.

Equality in our view, on the other hand, should be a *necessary* equality which must hold in every evaluation of a program (specification). *It does not correspond to set equality, but to identity of 1-element sets.* Thus the simple formula $t\doteq s$ should hold in a multistructure M iff both $t$ and $s$ are interpreted in M as one and the same set which, in addition, *has only one element*. (This is the reason for using the symbol $\doteq$ instead of usual =.) Equality is then a *partial equivalence relation* (not necessarily reflexive) and terms $t$ for which $t\doteq t$ holds are exactly the deterministic terms, denoted by $\mathcal{D}_{\mathrm{SP},X}$. More precisely, $t\in\mathcal{D}_{\mathrm{SP},X}$ if $t\doteq t$ holds in every model of the specification. (In particular, variables will always be interpreted deterministically.)

If it is possible to produce a computation where $t$ and $s$ return different results – and this is possible whenever they are nondeterministic – then the terms are not equal but, at best, equivalent They are equivalent if they are capable of returning the same results, i.e., if they are interpreted as the same set. This may be expressed using the inclusion relation: $s\prec t$ holds iff the set of possible results of $s$ is included in the set of possible results of $t$, and $s\succ\prec t$ iff each is included in the other.

A nondeterministic operation may then be specified by a series of inclusions – each defining one of its possible results. Such a specification establishes only a "lower bound" on the admitted nondeterminism.

*Example 2.1*

|  |  |  |  |  |
|---|---|---|---|---|
| S: | { Nat }, |  |  |  |
| F: | 0: |  | $\rightarrow$ Nat | (zero) |
|  | s_: | Nat | $\rightarrow$ Nat | (successor) |
|  | _⊔_: | Nat×Nat | $\rightarrow$ Nat | (binary nondeterministic choice) |

Π. 1. $0 \doteq 0$
2. $s(x) \doteq s(x)$
3. $1 \# 0$ (As usual, we abbreviate $s^n(0)$ as $n$.)
4. $0 \prec 0 \sqcup 1$ $1 \prec 0 \sqcup 1$ □

The first two axioms make zero and successor deterministic. A limited form of negation is present in $\mathcal{L}$ in the form of clauses with only negative atoms. Axiom 3. makes 0 distinct from 1. Axioms 4. make then $\sqcup$ a nondeterministic choice with 0 and 1 among its possible results. This, however, ensures only that in every model both 0 and 1 can be returned by $0 \sqcup 1$. In most models all other kinds of elements may be among its possible results as well, since no extension of the result set of $0 \sqcup 1$ will violate the inclusions of 4. If we are satisfied with this degree of precision, we may stop here and use only Horn formula. All the results in the rest of the paper apply to this special case. But to specify an "upper bound" of nondeterministic operations we also need disjunction of positive atoms. Before we illustrate this we have to consider another feature of the language.

The fact that equality is not reflexive for the nondeterministic terms reflects the intended interpretation of terms as *applications* of the respective operations. Every two syntactic occurrences of a term $t$ refer to *possibly distinct* applications of $t$. For nondeterministic $t$, the equality $t \doteq t$ does not hold because the first occurrence refers to an arbitrary application of $t$ and the second one to an arbitrary, hence possibly distinct, application of the same operation $t$. The equality says that arbitrary two applications return the same result. Now, if we write the axiom:

5. $0 \sqcup 1 \doteq 0 \vee 0 \sqcup 1 \doteq 1$

the two occurrences of $0 \sqcup 1$ refer to two arbitrary applications and, consequently, we obtain that either any application of $0 \sqcup 1$ equals 0 or else it equals 1, i.e., that $\sqcup$ is not really nondeterministic but merely underspecified. Since axioms 4. require that both 0 and 1 be among the results of $t$, addition of 5. will actually make the specification inconsistent.

What we are trying to say with the disjunction 5. is that *every application* of $0 \sqcup 1$ returns either 0 or 1, i.e., we need a means of identifying two occurrences of a nondeterministic term as referring to one and the same application. This can be done by *binding* both occurrences to a variable. We would write instead

5'. $x \# 0 \sqcup 1 \vee x \doteq 0 \vee x \doteq 1$

The axiom says: whenever $0 \sqcup 1$ returns $x$, then $x$ equals 0 or $x$ equals 1. Notice that such an interpretation presupposes that variables refer to unique, individual values. Variants of this interpretation are known as *call-time-choice* [14, 6], *singular* [30], *inside-out* or *IO* [8, 9] and correspond, roughly, to the call-by-value passing of deterministic parameters. This is the most common approach in the literature on the algebraic semantics of nondeterministic specification languages, in spite of the fact that it prohibits unrestricted substitution of terms for variables. Any substitution must now be guarded by the check that the substituted term yields a unique value, i.e., is deterministic. We return to this point in the section on reasoning where we introduce a calculus with the appropriately restricted substitution rules. (For a further discussion of the distinction between plural and singualr semantics of variables, see [33, 38].)

## 3. Multistructures and multimodels.

**Definition 3.2 (Multistructures).** For a signature $\Sigma$, M is a $\Sigma$-*multistructure* if

1. its carrier $|M|$ is an S-sorted set, and
2. for every $f: S_1 \times \ldots \times S_n \rightarrow S$ in $\mathbf{F}$, there is a corresponding function $f^M: S_1^M \times \ldots \times S_n^M \rightarrow \mathcal{P}^+(S^M)$.

A function $\Phi$: $A \to B$ (i.e., a family of functions $\Phi_S$: $S^A \to S^B$, for every $S \in \mathbf{S}$) is a *multihomomorphism* from a $\Sigma$-multistructure A to B if

H1. for each constant symbol $c \in \mathbf{F}$, $\Phi(c^A) \subseteq c^B$ and

H2. for every $f$: $S_1 \times \ldots \times S_n \to S$ in $\mathbf{F}$ and $\underline{a}_1 \ldots \underline{a}_n \in S_1^A \times \ldots \times S_n^A$:
$\Phi(f^A(\underline{a}_1 \ldots \underline{a}_n)) \subseteq f^B(\Phi(\underline{a}_1) \ldots \Phi(\underline{a}_n))$

If all inclusions in H1 and H2 are (set) equalities the homomorphism is *tight*, otherwise it is strictly loose (or just *loose*). □

Operations applied to sets refer to their unique pointwise extensions; $\mathcal{P}^+(S)$ is the set of non-empty subsets of S. Notice that for a constant $c$: $\to S$, 2. indicates that $c^M$ can be a *set* of several elements of sort $S$.

If there exists a multihomomorphims $\Phi$: $A \to B$ which is an inclusion we say that A is a *submultistructure* of B. (Optionally, we may only require that $\Phi$ is injective.) If, in addition, $\Phi$ is tight A is a *tight submultistructure* of B.

Since multihomomorphisms are defined on individuals and not sets they preserve singletons and are $\subseteq$-monotonic. We denote the class of $\Sigma$-multistructures by MStr($\Sigma$). It has the distinguished word structure:

**Definition 3.3 (Word multistructure).** The *word multistructure* $MW_\Sigma$ for a specification SP over signature $\Sigma$=(S,F) is defined as:

1. for each $S \in \mathbf{S}$, $S^{MW_\Sigma}$ is the set of all 1-element sets containing ground words of sort S,
2. for each $f$: $S_1 \times \ldots \times S_n \to S$ in $\mathbf{F}$, $\{t_i\} \in S_i^{MW_\Sigma}$ : $f^{MW_\Sigma}(\{t_1\},\ldots,\{t_n\}) = \{f(t_1,\ldots,t_n)\}$
□

According to 2. every ground term is interpreted in $MW_\Sigma$ as a singleton set. We will often treat such results as terms rather than 1-element sets (i.e., we do not take special pains to distinguish $MW_\Sigma$ and $W_\Sigma$).

With this definition $MW_\Sigma$ is not an initial $\Sigma$-structure since it is deterministic and there can exist several homomorphisms from it to a given multistructure. In [18] it is shown that it is (tightly) initial if we modify definition 3.2 by allowing a homomorphism $\Phi$: $A \to B$ to send single elements of $|A|$ to (nonempty) sets of elements of $|B|$ (i.e., letting $\Phi$: $|A| \to \mathcal{P}^+(|B|)$). We will retain the notion of homomorphism as in definition 3.2 because such a generalization is not needed. It merely "disguises" some strictly loose homomorphisms as tight ones since we may then have $c^A=\{\underline{c}\}$ and $\Phi(c^A)=\{\underline{c}_1,\ldots,\underline{c}_n\}=c^B$. (Underscored lower case letters, such as $\underline{a}$, $\underline{c}$, are used for the elements of a semantic domain.)

More importantly, it is a known fact [18, 24] that, in the general case, one should not expect the existence of initial multi*models*. In [18] Hußmann has shown that such multimodels may not exist even if the specification language is restricted to simple formulae. Therefore we allow general clauses in the specifications and will concentrate on the whole class of multimodels of a specification. For the discussion of the structure of this class and (the lack of) initiality results, the reader is referred to [18, 37, 36]. In the present context, the significance of the term structure is expressed in

**Lemma 3.4.** If M is a $\Sigma$-multistructure then, for every set of variables X and assignment $\beta$: $X \to |M|$, there exists a unique function $\beta[\_]$: $W_{\Sigma,X} \to \mathcal{P}^+(|M|)$ such that:

$\beta[x] = \{\beta(x)\}$ $\qquad$ $\beta[c] = c^M$ $\qquad$ $\beta[f(t_1,\ldots,t_n)] = \bigcup\{f^M(t1,\ldots tn) \mid ti \in \beta[t_i]\}$ □

In particular, for X=$\emptyset$, there is a unique interpretation function (not a multihomomorphism) $\mathcal{I}$: $W_\Sigma \to \mathcal{P}^+(|M|)$ satisfying the last two points of this definition. Observe that, as a consequence of the definition of multistructures, operations in M are $\subseteq$-monotonic, i.e., $\beta[s]\subseteq\beta[t] \Rightarrow \beta[f(s)]\subseteq\beta[f(t)]$. Next we define the class of multimodels of a specification:

**Definition 3.5 (Satisfiability).** Let M be an $\Sigma$-multistructure, $\beta$: $X\to|M|$ an assignment to a set of variables X. M satisfies an atom $a$ under $\beta$ iff

- $a$ is $s \prec t$ and $\beta[s] \subseteq \beta[t]$
- $a$ is $s \# t$ and $\beta[s] \cap \beta[t] = \emptyset$
- $a$ is $s \doteq t$ and $\beta[s] = \beta[t] = \{x\}$, for some $x \in |M|$.

M *satisfies* a clause $C$, $M \models C$, iff for each $\beta$, M satisfies at least one atom of $C$.
An SP-*multimodel* is an SP-multistructure which satisfies all the axioms of SP.
*MMod(SP)* denotes the class of multimodels of SP. □

The reason for using empty intersection (and not set inequality) as the interpretation of the negative atomic formulae $s\#t$ is the same as for using "elementwise" equality as the interpretation of $\doteq$. Since we avoid set equality in the positive sense, the most natural negative form seems to be the one we have chosen. Under the chosen interpretation, $\doteq$ denotes *necessary* equality and $\#$ *necessary* inequality. For deterministic terms $t$ and $s$, $s\#t$ is the same as $s\neq t$, i.e., deterministic negative atoms correspond exactly to equational (deterministic) conditions.

For nondeterministic terms the predicate $\#$ reflects our interest in *binding* such terms. According to the last definition variables may be assigned only single elements from the carrier $|M|$ and so always denote individual values. This means that, for instance, the clause $s\#t$ is equivalent to $\{ x\#s, x\#t \}$. Considering $x$ to be an individual and $t$ a set, $x\#t$ may be seen as the more familiar $x\notin t$. Thus we may treat our notation as an abbreviation for more elaborate formulae with two negated membership relations and one new variable (not occurring in the rest of the clause).

## 4. Generated multimodels.

Consider a deterministic specification over signature $\Sigma$, and let $Str(\Sigma)$ denote the class of $\Sigma$-structures, and $W_\Sigma$ the word structure over signature $\Sigma$.

**Definition 4.6.** $A\in Str(\Sigma)$ is *generated* iff the unique (interpretation) homomorphism $\mathcal{I}: W_\Sigma\to A$ is surjective. □

It is convenient to use a more general notion of generatedness. For specifications with constructors, sufficient completeness corresponds to models being generated by the constructor operations. In the context of nondeterministic specifications we may want to say that a model is to be generated by the deterministic operations only. This motivates

**Definition 4.7.** Let $\mathcal{T}\subseteq W_\Sigma$. $A\in Str(\Sigma)$ is *$\mathcal{T}$-generated* iff the unique homomorphism $\mathcal{I}: W_\Sigma\to A$ is surjective when restricted to $\mathcal{T}$, i.e., $\mathcal{I}[\mathcal{T}] = |A|$. □

In particular, "generated" will mean $W_\Sigma$-generated. We may allow $\mathcal{T}$ to comprise only terms of some (not all) sorts and speak about $\mathcal{T}$-generated structures meaning the structures where only the corresponding sorts are $\mathcal{T}$-generated.

The notion of generated structure does not generalize trivially to multistructures. As usual, there are several possibilities for extending the deterministic concepts to the nondeterministic context.

Since nondeterministic terms may be interpreted as sets of values rather than as individuals, the choices we are facing concern which of these values are to be considered as generated. The first definition comes from [17]:

**Definition 4.8.** A structure $M \in MStr(\Sigma)$ is *1-$\mathcal{T}$-generated* iff for every $\underline{m} \in |M|$ there exists a $t \in \mathcal{T}$: $\{\underline{m}\} = t^M$. □

According to this definition, every value in M must be denoted by some term $t$ which, in addition, is interpreted in M deterministically even if $t \doteq t$ is not a consequence of SP.

The next definition is less restrictive in this respect:

**Definition 4.9.** $M \in MStr(\Sigma)$ is *2-$\mathcal{T}$-generated* iff there exists a multihomomorphism $\hbar$: $MW_\Sigma \rightarrow M$ which is surjective when restricted to $\mathcal{T}$. □

It implies that all elements of the carrier of $M$ are denotable by some term – for every $\underline{m} \in |M|$ there exists a $t \in \mathcal{T}(\{t\} \in MW_\Sigma)$ such that $\hbar(t) = \underline{m}$. Here a nondeterministic term $c$ can be used by the homomorphism $\hbar$ as a pre-image of some individual, even if its interpretation $c^M$ may be a set of several values.

Both definitions focus on the elements of the structure and do not requre that also all sets be denotable. The last definition, from [15], allows most freedom in the generated multistructures:

**Definition 4.10.** A structure $M \in MStr(\Sigma)$ is *3-$\mathcal{T}$-generated* iff for every $\underline{m} \in |M|$ there exists a $t \in \mathcal{T}$: $\underline{m} \in t^M$. □

Equality from the first definition has been replaced here by the membership relation. According to this definition a structure is generated if every element is a member of a set denoted by some term.

**Definition 4.11.** M is a *n-$\mathcal{T}$-generated multimodel* (for n=1,2,3), $M \in MMod_{n\text{-}\mathcal{T}}(SP)$, if it is a n-$\mathcal{T}$-generated multistructure and satisfies the axioms of the specification SP. □

The following example illustrates the relation between the three definitions:

*Example 4.12*

Let SP = (({S}, {$a,c$: $\rightarrow$S}), { {$a \prec c$} }):

| | A1 | A2 | A3 |
|---|---|---|---|
| $a$ | $\{\underline{a}\}$ | $\{\underline{a}\}$ | $\{\underline{a}_1, \underline{a}_2, \ldots\}$ |
| $c$ | $\{\underline{a}\}$ | $\{\underline{a}, \underline{c}\}$ | $\{\underline{a}_1, \underline{a}_2, \ldots, \underline{c}_1, \underline{c}_2, \ldots\}$ |

A1 is 1-, 2-, and 3-generated, A2 is 2- and 3-generated, and A3 is only 3-generated. A2 is not 1-generated because there is no term (among $\{a,c\}$) which can be interpreted deterministically as $\{\underline{c}\}$ ($c$ cannot be such a term without violating the axiom). A3 is not 2-generated because there is no surjective homomorphism from $\{a,c\}$ onto the elements of $|A3|$. □

1-generatedness is characterised by the non-existence of substructures: M is 1-generated (for $\mathcal{T}$=$W_\Sigma$) if and only if it has no proper submultistructure. This notion corresponds to the idea of determmistic constructors – only the values which can be reached by some deterministic operation are considered generated. It reflects the intuition that programs operate on some definite universe of values. 3-generatedness stands on the opposite side allowing constructors to be nondeterministic – anything such operations can return is taken to be generated. 3-generated models may be convenient when some axioms are given for nondeterministic terms without reference to any (intended or actual) deterministic basis. This concept corresponds to the non-existence of tight substructures: M is 3-generated if and only if it has no proper tight submultistructure. 2-generatedness may seem a bit strange but it too has a plausible interpretation. It restricts nondeterminism, but preserves the information about underspecified nondeterministic operations. The axiom $a \prec c$ indicates only that $a$ must be a possible result of $c$. But nondeterminism of $c$ is not exhausted by this information and so 2-generatedness allows the generated model A2 to have an additional element $\underline{c}$ representing the possibility of $c$ returning something more than $\underline{a}$.

Obviously, 1-$\mathcal{T}$-generatedness is the smallest predicate and we have the trivial

**Lemma 4.13.** $\mathrm{MMod}_{1\text{-}\mathcal{T}}(\mathrm{SP}) \subseteq \mathrm{MMod}_{2\text{-}\mathcal{T}}(\mathrm{SP}) \subseteq \mathrm{MMod}_{3\text{-}\mathcal{T}}(\mathrm{SP})$. □

Which of the definitions one prefers may be a matter of taste, and we will not attempt to choose among them at this point. The choice will be suggested in section 7 on inductive reasoning. At the moment, we only observe that when $\mathcal{T} \subseteq \mathcal{D}_{\mathrm{SP}}$, they are all equivalent to the classical definition.

**Definition 4.14.** $\mathrm{M} \in \mathrm{MMod(SP)}$ is *DET-generated* iff it is $\mathcal{T}$-generated, for some $\mathcal{T} \subseteq \mathcal{D}_{\mathrm{SP}}$. □

We have dropped the numbering of generatedness because all three definitions are equivalent for $\mathcal{T} \subseteq \mathcal{D}_{\mathrm{SP}}$.

**Lemma 4.15.** Let $\mathcal{T} \subseteq \mathcal{D}_{\mathrm{SP}}$. Then $\mathrm{MMod}_{1\text{-}\mathcal{T}}(\mathrm{SP}) = \mathrm{MMod}_{2\text{-}\mathcal{T}}(\mathrm{SP}) = \mathrm{MMod}_{3\text{-}\mathcal{T}}(\mathrm{SP})$.
□

The difference between 1-$\mathcal{T}$-generated and DET-generated structures is that the former simply enforce the deterministic interpretation of (possibly nondeterministic) terms from $\mathcal{T}$ while the latter can only contain elements which are denoted by terms which have to be deterministic.

The DET-generated multialgebras have the important property that the homomorphisms from such structures are unique.

*Example 4.16*

Let SP contain only one operation $c: \to \mathrm{S}$. Any 1- or 2-generated multialgebra N1, N2 will have a 1-element S sort – $\mathrm{S}^{\mathrm{N}} = \{\underline{c}\}$. A 3-generated multialgebra N3 may have any number of elements in the sort S. Let $\mathrm{N} \in \mathrm{MStr(SP)}$ interpret $c$ nondeterministically, e.g. $c^N = \{\underline{c}_0, \underline{c}_1\}$. Obviously, there are two homomorphisms from the generated N1, N2 to N. Also, for any N3 there will be at least two homomorphisms to N.

Let SP1 contain, in addition, the operations $a, b: \to \mathrm{S}$, and the axioms

1. $a \prec c$ 2. $b \prec c$ 3. $a \doteq a$, 4. $b \doteq b$ 5. $a \# b$

and consider (1- and 2-)generated multimodels of SP1, M1 and M2, where

$$\mathrm{S}^{\mathrm{M1}} = \{\underline{a}, \underline{b}\}, \qquad \mathrm{S}^{\mathrm{M2}} = \{\underline{a}, \underline{b}, \underline{c}\},$$
$$c^{\mathrm{M1}} = \{\underline{a}, \underline{b}\} \qquad c^{\mathrm{M2}} = \{\underline{a}, \underline{b}, \underline{c}\}$$

Let $\mathrm{N} \in \mathrm{MMod(SP1)}$. For M2 (and M3 with more than 2 elements of sort S), a homomorphism to N will not be unique. But if there is a homomorphism $\phi: \mathrm{M1} \to \mathrm{N}$ then such a $\phi$ is unique since

$$\phi(a^{\mathrm{M1}}) = a^N \quad \phi(b^{\mathrm{M1}}) = b^N \quad \phi(c^{\mathrm{M1}}) = \phi(\{\underline{a}, \underline{b}\}) = \{\phi(a^{\mathrm{M1}}), \phi(b^{\mathrm{M1}})\} = \{a^N, b^N\} \subseteq c^N$$
□

From the first part of this example we can see that 1-generatedness of N1 does not guarantee uniqueness of the homomorphisms from N1. Uniqueness of $\phi$ is a consequence not of M1 being 1-generated but of the fact that it is DET-generated.

**Lemma 4.17.** Let M,N$\in$MMod(SP) and let M be DET-generated. There exists at most one homomorphism $\phi$: M$\rightarrow$N.

*Proof idea:*
Any homomorphism $\phi$: M$\rightarrow$N must satisfy $\phi(t^M) \subseteq t^N$, which for deterministic $t$ means identity. DET-generatedness of M implies then that the image of any homomorphism from M is uniquely determined by the image of the interpretations of the deterministic terms. □

## 5. Quasi-initial semantics.

The last lemma is important because it allows us to define quasi-initial semantics of nondeterministic specifications. Since we have allowed the use of disjunction in the specifications it should not be surprising that, in general, we do not have initial (multi)models. However, as shown for instance in [17, 18], even if we restrict the specifications to Horn formulae, or even to simple formulae, initial multimodels still need not exist.

Quasi-initial semantics assigns to a specification an initial covering of its model class. It is a natural generalization of initial semantics in that it effects a partition of the model class into components – each with an initial object which is the quasi initial object from the covering. It was used by several authors as the basis for the semantics of specifications using more expressive language than Horn formulae [20, 13, 32, 42].

The class MMod(SP) together with the multihomomorphisms is a pre-order, denoted (MMod(SP),$\rightarrow$), where N$\rightarrow$M iff there is a multihomomorphism from N to M.

**Definition 5.18 (Minimal, Quasi-Initial).** M$\in$MMod(SP) is

1. *minimal* in the pre-order (MMod(SP),$\rightarrow$) iff, for all N$\in$MMod(SP), N$\rightarrow$M implies M$\rightarrow$N,
2. *quasi-initial* iff it is minimal and for every N$\in$MMod(SP), there exists at most one homomorphism M$\rightarrow$N. □

**Definition 5.19 (Covering).**

1. A *covering* of a pre-order (MMod(SP),$\rightarrow$) is a subset **C**$\subseteq$MMod(SP) such that for each M$\in$MMod(SP) there is a C$\in$**C** with C$\rightarrow$M.
2. A covering is *minimal* if none of its proper subsets is a covering.
3. A covering **C** is *initial* iff it is minimal and contains only quasi-initial objects. □

The general result is

**Theorem 5.20.** If MMod(SP) is not empty then it has a minimal covering.

*Proof idea:*
Every non-empty chain in (MMod(SP),$\rightarrow$) has a lower bound. Zorn's lemma implies the conclusion. □

Unfortunately, we cannot ensure the existence of an initial covering in the general case. To obtain a sufficient condition for its existence we need an appropriate restriction on the form of the specifications. Lemma 4.17 indicates that if a minimal covering contains only DET-generated models then the covering is initial. To ensure the existence of DET-generated models we need

**Definition 5.21.** SP=$(\Sigma,\Pi)$ is *DET-complete* iff for every $t \in W_\Sigma$ there exists an $s \in \mathcal{D}_{SP}$ such that $s \prec t$ is valid. □

The restriction to DET-complete specifications was introduced, for slightly different reasons, in [17]. DET-generated structures constitute the semantic counterpart of DET-completeness in the sense that only DET-complete specifications are guaranteed to possess DET-generated multimodels.

**Lemma 5.22.** Let SP be DET-complete. Every $M \in$MMod(SP) has a DET-generated subalgebra N, i.e., such that there is an injective homomorphism $\iota$: N→M.

*Proof idea:*
For every $t \in W_\Sigma$ let $t^N = t^M \cap \mathcal{D}^M_{SP}$. DET-completeness of SP ensures that no such set is empty and the inclusion $|N| \subseteq |M|$ is the required homomorphism. □

Moreover, for DET-complete specifications, the DET-generated subalgebras of minimal models are exactly the quasi-initial models.

**Lemma 5.23.** Let SP be DET-complete, $M \in$MMod(SP) and N be DET-generated subalgebra of M. Then:

1. if M is quasi-initial then M is DET-generated,
2. if M is minimal then N is quasi-initial.

*Proof idea:*
1. Quasi-initiality of M forces the inclusion $|N| \subseteq |M|$ to be surjective.
2. Inclusion homomorphism $|N| \subseteq |M|$, minimality of M and lemma 4.17. □

Finally, for the DET-complete specifications we have:

**Theorem 5.24.** If SP is a DET-complete specification and MMod(SP)$\neq\emptyset$ then MMod(SP) has an initial covering.

*Proof idea:*
Take the DET-generated subalgebras of the elements of a minimal covering. □

## 6. The calculus for nondeterministic specifications.

In [36] we have introduced the calculus NEQ which is sound and complete with respect to the class MMod(SP). Its rules are given below:

R0: $\vdash C$ for any $C \in \Pi$

R1: a) $\vdash x \# y, x \doteq y$ b) $\vdash x \# t, x \prec t$ $x, y \in \mathcal{V}$

R2: $$\frac{\vdash C^x_t \quad ; \quad \vdash D, s \doteq t}{\vdash C^x_s, D}$$

R3: $$\frac{\vdash C^x_t \quad ; \quad \vdash D, s \prec t}{\vdash C^x_s, D}$$ $x$ not in a right-hand side of $\prec$ in $C$

R4: $\dfrac{\vdash C, s \preceq t \quad ; \quad \vdash D, s \# t}{\vdash C, D}$ (CUT) ($\preceq$ being either $\doteq$ or $\prec$)

R5: $\dfrac{\vdash C}{\vdash C, e}$ (WEAK)

R6: $\dfrac{\vdash C, x \# t}{\vdash C^x_t}$ (ELIM) $x \in \mathcal{V} - \mathcal{V}[t]$, at most one $x$ in $C$

$C^x_t$ denotes $C$ with $t$ substituted for $x$. We will write $\Pi \vdash C$ to indicate that $C$ is provable from $\Pi$ with the calculus NEQ. Few comments regarding the rules may be in order.

R1 expresses the relation between $\#$ and equality and inclusion. Since variables $x$ and $y$ are individuals, the two rules correspond to, respectively, $x{\neq}y \vee x{=}y$, and $x \notin t \vee x \in t$. They also capture the fact that '$\doteq$' is a partial equivalence relation and is reflexive only for variables (see PER below).

R2 is a paramodulation rule allowing replacement of deterministic terms (in the case when $s \doteq t$ holds in the second assumption). In particular, it allows derivation of the standard substitution rule when the substituted terms are deterministic, and prevents substitution of nondeterministic terms for variables.

R3 allows "specialization" of a clause by substituting for a term $t$ another term $s$ which is included in $t$. The restriction that the occurrences of $t$ which are substituted for don't occur in the right-hand side of $\prec$ in $C$ is needed to prevent, for instance, the unsound conclusion $\vdash p \prec s$ from the premises $\vdash p \prec t$ and $\vdash s \prec t$.

$s \# t$ implies both negation of $s \doteq t$ and of $s \prec t$. R4 allows us to resolve these complementary atoms.

R6 eliminates redundant bindings, namely those that bind an application of a term occurring at most once in the rest of the clause.

The following derived rules illustrate some consequences of NEQ.

PER: $\vdash x \doteq x$ variables are guaranteed to be deterministic

NE: $\dfrac{\vdash x \# t}{\vdash \emptyset}$ R6 terms are interpreted as non-empty sets

SUB: $\dfrac{\vdash C \quad ; \quad \vdash t \doteq t}{\vdash C^x_t}$ variables can be substituted by deterministic terms

The counterpart of soundness/completeness of the equational calculus is the following:

**Theorem 6.25.** NEQ is sound and complete wrt. MMod(SP):

$$\text{MMod(SP)} \models C \text{ iff } \Pi \vdash C. \qquad \Box$$

The proof of this theorem is a rather involved Henkin-style argument which we cannot even sketch here (see [34]). In the next section we consider extension of NEQ with an induction principle.

## 7. The nondeterministic ω-rule.

Let SP=($\Sigma$,$\Pi$) be a deterministic specification, $C$ be a formula in the associated specification language $\mathcal{K}$ with a free variable $x$ of sort $S$. Let $\mathcal{T}\subseteq W_\Sigma$ and Mod$_\mathcal{T}$(SP) be the class of $\mathcal{T}$-generated models of SP. The *ω-rule* is a semi-formal proof rule [e.g., 40, 41]: [2]

$$\frac{\Pi \vdash C^x_t \qquad \forall t\in\mathcal{T}}{\Pi \vdash C} \tag{IP}$$

Then, for instance for the equational specification language $\mathcal{K}$, one can prove the metatheorem

$$\text{Mod}_\mathcal{T}(\text{SP}) \models C \text{ iff } \Pi \vdash C^x_t \text{ for all } t\in\mathcal{T} \qquad \text{(IP-completness)}$$

The *inductive proof schema* specifies, in addition, the strategy for proving the premise of IP by means of some enumeration of all terms in $\mathcal{T}$. For example, if $\mathcal{T}$ can be built from a set of constants $C$ of sort $S$, and operations $F=\{f_1,\ldots,f_n: S\rightarrow S\}$ then the induction schema will be

$$\frac{\begin{array}{ll}\Pi \vdash C^x_c & \forall c\in C\\ \Pi\vdash C \Rightarrow \Pi\vdash C^x_{f(x)} & \forall f\in F\end{array}}{\Pi \vdash C} \tag{IS}$$

Obviously, soundness of IP implies soundness of IS but, in general, IS is not complete [27]. We will focus on the ω-rule – a particular induction schema must be constructed depending on the chosen set $\mathcal{T}$.

We first consider the question of which among the notions of generatedness we have introduced constitute a natural semantic counterpart of the possible nondeterministic induction. DET-generated models are certainly of special interest and we will consider them at the end of this section. 1- and 2-$\mathcal{T}$-generated models turn out to be ill-suited for the purpose. At least two reasons for this are that:

1. they are not guaranteed to exist, and inconsistency may be unprovable with the induction principle, and
2. even if they do exist, additional semantic and syntactic assumptions are needed in order to make the natural induction principle sound.

We elaborate on these two points with respect to 1-$\mathcal{T}$-generated models. The case of 2-$\mathcal{T}$-generated models is similar and merely involves even further complications.

*Example 7.26*

Let SP contain three constants $a$, $b$, $c$: $\rightarrow$ S, and the following three axioms:

$\Pi$: $\{ a\prec c \ ;\ b\prec c \ ;\ a\# b \}$

SP does not have a 1-$\{a,c\}$-generated model because in such a model both $a$ and $c$ must be interpreted deterministically, and then, due to the third axiom, $\underline{b}$ cannot be contained in the set $\{\underline{c}\}=\{\underline{a}\}$. □

1-$\{a,c\}$-generatedness is inconsistent with SP because it forces the elements $a$, $c$ to be interpreted as deterministic operations while SP requires $c$ to be nondeterministic. It might seem that we could prove inconsistency by adding the determinacy axiom for every term $t\in\mathcal{T}$. But this would lead to unsound reasoning because not *all* interpretations of the $\mathcal{T}$-terms have to be deterministic in a 1-$\mathcal{T}$-generated multimodel.

[2] We formulate the principle, as well as the theorems, for one variable $x$. The general formulations are obtained trivially by replacing the occurrences of single $x$ and $t$ by the tuples $x_1,\ldots,x_n$ and $t_1,\ldots,t_n$.

*Example 7.27*

Let SP1 contain three constants $a, b, c : \rightarrow S$, and no axioms. There are several 1-$\{a,b,c\}$-generated models of SP1:

| | A1 | A2 | A3 | A4 | A5 |
|---|---|---|---|---|---|
| $a$ | $\{\underline{a}\}$ | $\{\underline{a}\}$ | $\{\underline{a}\}$ | $\{\underline{b},\underline{c}\}$ | $\{\bullet\}$ |
| $b$ | $\{\underline{b}\}$ | $\{\underline{b}\}$ | $\{\underline{a},\underline{c}\}$ | $\{\underline{b}\}$ | $\{\bullet\}$ |
| $c$ | $\{\underline{c}\}$ | $\{\underline{a},\underline{b}\}$ | $\{\underline{c}\}$ | $\{\underline{c}\}$ | $\{\bullet\}$ |

The suggested determinacy axioms, $a \doteq a$, $b \doteq b$, $c \doteq c$ hold only for A1 and A5.
□

Thus, on the one hand we need the additional determinacy axioms and, on the other hand, they are too strong for the whole class of 1-$\mathcal{T}$-generated models. To remove this contradiction several additional restrictions on the form of specifications as well as the model class are needed. Hußmann, who in [17] uses 1-generated models, introduced for a limited version of the language $\mathcal{L}$ restrictions to DET-complete, DET-additive specifications and *maximally deterministic* models (in the last example only A1 and A5 would be maximally deterministic) which allow only a reduced version of the induction principle.

We therefore turn to 3-$\mathcal{T}$-generated models as being more promising. For every M in the class $\mathrm{MMod}_{3\text{-}\mathcal{T}}(\mathrm{SP})$ of 3-$\mathcal{T}$-generated models, all elements are in the result set of some term $t \in \mathcal{T}$. However, we cannot base our induction on the assumption that $C^x_t$ is provable for all $t \in \mathcal{T}$ since for nondeterministic $t$ such a substitution does not correspond to the validity of the clause $C$ for all elements of a generated domain. The correct formulation of the $\omega$-rule is the following:

$$\text{If } \Pi \vdash \{x \# t, C\} \text{ for all } t \in \mathcal{T} \text{ then } \Pi \vdash C \qquad \text{(NIP)}$$

Let NIP denote the calculus NEQ+NIP. Our main result is

**Theorem 7.28.** Let $C$ be an $\mathcal{L}(\mathrm{SP})$ clause:

$$\mathrm{SP} \vdash_{\mathrm{NIP}} C \quad \text{iff} \quad \mathrm{MMod}_{3\text{-}\mathcal{T}}(\mathrm{SP}) \models C \qquad \square$$

Soundness is an easy consequence of soundness of NEQ, the proof of completeness is a more elaborate application of the omitting type theorem from [5].

As a trivial consequence of lemma 4.15, the induction principle NIP can be used for the DET-generated models. In this case $\mathcal{T} \subseteq \mathcal{D}_{\mathrm{SP}}$, so NIP reduces to IP, because binding a deterministic term $t$ to $x$ in a clause $C$ is equivalent to the substitution $C^x_t$:

**Lemma 7.29.** Suppose that $\Pi \vdash t \doteq t$. Then:

$$\Pi \vdash C^x_t \quad \text{iff} \quad \Pi \vdash \{x \# t, C\} \qquad \square$$

## 8. Conclusion.

We have defined four notions of generated multistructures for nondeterministic specifications. Three of them admit generated structures with respect to arbitrary terms, and the fourth – DET-generatedness – models generated by the deterministic terms only. We have shown that in the case of DET-generatedness all definitions are equivalent, and hence, correspond also to the classical definition of the generated structures.

We have also given the sufficient condition of DET-completeness for the existence of quasi-initial semantics for nondeterministic specifications and shown that quasi-initial models are DET-generated.

We have presented two calculi for reasoning about nondetermism. NEQ is sound and complete with respect to all multimodels of a nondeterministic specification. Furthermore, when all terms are deterministic, it is a generalization of the equational calculus to conditional disjunctive equations. NIP is NEQ extended with the induction principle. We have stated soundness and completeness of NIP with respect to the class of 3-generated multimodels, as well as the equivalence of the principle with the classical induction in the case when the generating terms are deterministic.

Thus we have generalized the results on induction in the deterministic specifications to the case of nondeterminism. This extension is non-intrusive in the sense that, whenever an $\mathcal{L}$-specification happens to be deterministic, the results and constructions provided by the classical theory coincide with those introduced in this paper.

## Acknowledgment.

We thank an anonymous referee for useful suggestions, in particular, for drawing our attention to the equivalence of the definitions 4.8 and 4.10 to the respective notions of submultistructures.

## REFERENCES

[1] Bergstra, J.A., Klop, J.W., "Algebra of communicating processes," *Proc. of CWI Symposium on Mathematics and CS*, 89 - 138, Oct. 6-7 1986.

[2] Broy, M., Gnatz, R., Wirsing, M., "Semantics of Nondeterministic and Noncontinuous Constructs," LNCS, vol. 69, Springer, 1980, pp. 553 - 392.

[3] Broy, M., Wirsing, M., "On the Algebraic Specification of Nondeterministic Programming Languages," in *CAAP'81*, LNCS, vol. 112, Springer, 1981, pp. 162 - 179.

[4] Broy, M., "On the Herbrand Kleene universe for nondeterministic computations," in *Proc. MFCS'84*, LNCS, vol. 176, Springer, 1984.

[5] Chang, C.C., Keisler, H.J., *Model Theory*, Amsterdam, North-Holand, 1977.

[6] Clinger, W., "Nondeterministic call by need is neither lazy nor by name," *Proc. ACM Symp. LISP and Functional Programming*, 226-234, 1982.

[7] Ehrig, H., Mahr, B., *Fundamentals of Algebraic Specification*, vol. 1, Springer, 1985.

[8] Engelfriet, J., Schmidt, E.M., "IO and OI. 1," *Journal of Computer and System Sciences*, vol. 15, 328-353, 1977.

[9] Engelfriet, J., Schmidt, E.M., "IO and OI. 2," *Journal of Computer and System Sciences*, vol. 16, 67-99, 1978.

[10] Goguen, J.A., Meseguer, J., "Completeness of Many-Sorted Equational Logic," *SIGPLAN Notices*, vol. 16, no. 7, 1981.

[11] Goguen, J.A., Meseguer, J., "Universal realization, persistent interconnection and implementation of abstract modules," in *Proc., 9th Int. Coll. on Automata, Languages and Programming*, LNCS, vol. 140, Springer, 1982.

[12] Goguen, J.A., Meseguer, J., "Remarks on Remarks on Many-Sorted Equational Logic," *SIGPLAN Notices*, vol. 22, no. 4, 41-48, April 1987.

[13] Goguen, J.A., *What is unification? A categorical view of substitution, equation, and solution*, Tech. Rep. CSLI-88-124, Center for Study of Languages and Information, 1988.

[14] Hennessy, M.C.B., "The semantics of call-by-value and call-by-name in a nondeterministic environment," *SIAM J. Comput.*, vol. 9, no. 1, 1980.

[15] Hesselink, W.H., "A Mathematical Approach to Nondeterminism in Data Types," *ACM Transactions on Programming Languages and Systems*, vol. 10, 1988.

[16] Hoare, C.A.R., *Communicating Sequential Processes*, Prentice-Hall International Ltd., 1985.

[17] Hußmann, H., *Nondeterministic Algebraic Specifications*, Ph.D. thesis, Fakultät für Mathematik und Informatik, Universität Passau, 1990.

[18] Hußmann, H., *Nondeterminism in Algebraic Specifications and Algebraic Programs*, Birkhäuser, 1993.

[19] Huet, G., Oppen, D., "Equations and Rewrite Rules: A Survey," in *Formal Language Theory: Perspectives and Open Problems*, Academic Press, 1980.

[20] Kaplan, S., "Conditional Rewriting," in *Conditional Term Rewriting Systems*, LNCS, vol. 308, Springer, 1987.

[21] Kaplan, S., "Rewriting with a Nondeterministic Choice Operator," *Theoretical Computer Science*, vol. 56, 37-57, 1988.

[22] Kapur, D., *Towards a theory of abstract data types*, Ph.D. thesis, Laboratory for CS, MIT, 1980.

[23] Milner, R., *Calculi for Communicating Systems*, LNCS vol. 92, Springer, 1980.

[24] Mosses, P.D., "Unified Algebras and Institutions," in *Proc. of LICS'89, Fourth Annual Symposium on Logic in Computer Science,* 1989.

[25] Nipkow, T., "Non-deterministic Data Types: Models and Implementations," *Acta Informatica*, vol. 22, 629 - 661, 1986.

[26] Nipkow, T., "Observing nondeterministic data types," in *Recent Trends in Data Type Specification*, LNCS, vol. 332, Springer, 1987.

[27] Nourani, F., "On induction for program logic: Syntax, semantics, and inductive closure," *EATCS Bulletin*, vol. 13, 1981.

[28] Plotkin, G., "A power domain construction," *SIAM Jour. Comp.*, vol. 5, no. 3, 452 - 487, 1976.

[29] Plotkin, G., Apt, K.R., "Countable Nondeterminism and Random Assignment," *Tech. Rep. University of Edinburgh*, 1982.

[30] Søndergaard, H., Sestoft, P , *Non-Determinacy and Its Semantics*, Tech. Rep. 86/12, Datalogisk Institut, Københavns Universitet, January 1987.

[31] Smyth, M.B., "Power domains," *J. of Computer and System Sciences*, vol. 16, 1978.

[32] Volger, H., "The semantics of disjunctive deductive databases," in *CSL'89*, LNCS, vol. 440, Springer, 1989.

[33] Walicki, M.A., Meldal, S., "Singular and plural nondeterministic parameters," *SIAM Journ. of Computing (submitted)*, .

[34] Walicki, M.A., Meldal, S., "A complete calculus for the multialgebraic and functional semantics of nondeterminism," *ACM Transactions on Programming Languages and Systems (submitted)*, .

[35] Walicki, M., Meldal, S., "Sets and Nondeterminism," in *Proc. of the Workshop on Logic Programming with Sets: ICLP'93,* 1993.

[36] Walicki, M., *Algebraic Specifications of Nondeterminism*, Ph.D. thesis, University of Bergen, Department of Informatics, 1993.

[37] Walicki, M., Meldal, S., "Initiality + Nondeterminism Implies Junk," in *Proc. of NIK'93,* Haveraaen, M., Tapir, November 1993, pp. 129-138.

[38] Walicki, M., Meldal, S., "Multialgebras, Power Algebras and Complete Calculi of Identities and Inclusions," to be publsihed in *Recent Trends in Data Type Specifications,* LNCS 1995.

[39] Winskel, G., "An introduction to event structures," LNCS, vol. 354, Springer, 1988.

[40] Wirsing, M., *Algebraic Specification*, Tech. Rep. MIP-8914, Universität Passau, 1989.

[41] Wirsing, M., "Algebraic Specification," in *Handbook of Theoretical Computer Science*, vol. B, The MIT Press, 1990.

[42] Wolter, U., Löwe, M., "Beyond Conditional Equations," in *CAAP'92*, LNCS, vol. 581, Springer, 1992.