Kuwait J. Sci. Eng. **24**(1) 1997

Operational Semantics Oriented Specification

PETER BACHMANN

Cottbus University of Technology, P.O. Box 10 13 44, D-03013 Cottbus
e-mail: pb@informatik.tu-cottbus.de

ABSTRACT

The main idea of an Operational Semantics Oriented Specification is to specify algorithms instead of functions. Here, the term represents a state of a calculation as well as a data-object. A confluent transition relation on terms describes the behaviour of the computation. Such a transition relation may be the model of a specification. Weak and strong models are distinguished where the strong models take the sequence of the specification rules into consideration. By a combination of rewriting, with a special form of narrowing, a prototype of models is given which allows the carrying out of specifications. It is shown how confluence in hierarchical specifications can be guaranteed also for nonterminating transition relations.

INTRODUCTION

Formal specification methods have already been used for many years for the description of software systems and during the design process. In software practice, the formal specification is now mainly a tool for the exact description of the input-output behaviour of the system and constraints which must be fulfilled. As a result, there exists a sound basis for communication between the designer and the programmer of the implementation. Then, the implementation has to fulfill the specification.

Formal specification methods are based upon logic and algebra. The algebraic specification was introduced in connection with the notion of Abstract Data Types (ADT) in the seventies (Goguen *et al.* 1975; Zilles 1974; Guttag *et al.* 1978) and has been a subject of scientific investigations since that time.

The main aim of an algebraic specification consists not only of the exact description of a certain system. It has been shown that algebraic specifications are often hard to read and only understandable if enough comments are included. Two important aspects make the specifications much more useful:

Firstly, the specification method can be used in the design process as a guide. The formal specification of a certain problem forces the researcher to bring informal ideas into a precise form. This leads to a decomposition of the whole problem into subproblems and to a stepwise solution according to the rules of a top-down design. This aspect is mentioned very seldom in the literature (Ehrich *et al.* 1989; Wirsing 1989).

Secondly, operational semantics of specifications allow the test (proof) of properties of the specification in order to detect errors or bottlenecks already at the level of the design, therefore, one can check whether the specification meets the expectation.

By operational semantics we mean mechanisms which allow the carrying out of a specification. The two main methods are term-rewriting and narrowing. It is known from universal algebra that term-rewriting is complete w.r.t. proving equations in varieties. This holds too for conditional rewriting in quasivarieties. However, this only means that for every theorem a rewriting proof exists. The main problem is to find this proof. If the theory is provable then an attempt is made to construct such a system of rewriting rules for which the proof can be done in a straight forward manner. This can be done for terminating and confluent systems. By means of the Knuth-Bendix-Completion (Knuth & Bendix 1970), under certain conditions such a terminating and confluent system of rewriting rules can be constructed on the basis of the given axioms of the theory. For the application of Knuth-Bendix-Completion termination proofs for rewriting rules are very important.

Narrowing is a method of finding solutions of a given set of equations. It can be considered a generalization of the resolution principle. In order to use conditional term-rewriting in the general case, narrowing must be combined with rewriting.

In formal specifications, term-algebras are often used as models. The initial algebra approach takes advantage of the fact that the set of congruence classes of terms build an initial algebra within any variety or quasivariety. The congruence classes are constructed by the stable and congruent closure of a set of equations in the case of a variety. A term $f(t_1, \dots, t_n)$ has therefore a two-fold meaning. Firstly, it represents a value of the carrier, namely its congruence class, and secondly, it represents the application of function f onto the congruence classes of $t_1, \dots, t_n$. This is a very clear and mathematically well-founded concept. However, it includes some problems, especially with respect to practical applications. One of the main problems is the characterization of partial functions. Within the usual initial algebra approach, partial functions cannot be described. This would lead to the introduction of new data. Therefore, the above-mentioned extensions have been done. It seems, however, that in these approaches the understanding of the models of a specification is very complicated, even for a theoretically well-trained specialist. Moreover, the deduction of an operational semantics from the given specification is only possible under special conditions which are undecidable, in general.

The idea of OSOS consists in using the operational semantics directly in order to describe models. As before, terms represent values as well as applications of a function. However, there is a special kind of term which only represents values, namely constructor terms built by means of constructor symbols. Constructor symbols cannot be used to represent functions. To do this, separate function symbols must be introduced. A term headed by a function symbol represents the application of the corresponding function. This application then creates a (finite or infinite) sequence of states which describes the stepwise computation process. Each state is also described by a term. Thus the initial state of a function application is this term itself. The transition from one term in this sequence to its successor means one step in the computation. The state transistion is, in general, a nondeterministic relation, e.g. more than one successor state may exist. However, we demand that the transition relation is confluent. In this way, a set of finite or infinite term sequences belongs to each term. Because of confluence, all the finite sequences have the same last term. It represents the result of the computation. This result is well defined if the last term is a constructor term, i.e. a value. Otherwise, the application of the function is undefined. However, there are different kinds of undefinedness. If the state sequence is finite and the last term is no constructor term, then the computation terminates with an error value which is represented by the last term. If the state sequence does not terminate, then this infinite sequence is, of course, the whole description of the computation which is also undefined. This concept allows the specification of nonstrict functions which may be defined also if some arguments are not defined.

A specification Σ assigns to every function symbol f a sequence

$$\Sigma(f) = (P_1, r_1) \dots (P_{n_f}, r_{n_f}).$$

Here, P_i are premises consisting of equations and r_i are terms. First, we will consider two kinds of models: *weak* models where the order of the sequence is unimportant and *strong* models where the order is taken into account. It is shown that in both cases all partial recursive functions can be specified.

If we interpret a specification as a set of conditional *rewrite*-rules, then we get a third kind of model. Since we have to allow that variables of the premises do not occur in the left hand side of rewrite-rules, we need techniques to solve equations. This will be done by L-narrowing, closely related to lazy-narrowing.

Confluence is a natural property. However, like classical algebraic specifications, it is not so simple to verifiy. Moreover, the usual methods for confluence proofs do not work in our case since we allow, in general, nonterminating transition relations as models. In section 5, we introduce semi-models where the confluence is dropped, and we present sufficient conditions and a method to prove that under these conditions any semi-model is confluent, i.e. it becomes a model. Hierarchical specifications turn out to be a practical class where these confluence proofs can be done on the basis of Noetherian induction.

DOMAINS AND OPERATIONAL SEMANTICS

OSOS - signatures and domains

Let S be a set of so-called *sorts*. With S^* we denote the set of finite sequences (finite words) on S, ε denotes the empty sequence, and S^*S denotes the set of nonempty sequences.

A *signature* is a triple (S, F, ar) where S is a set of sorts, F is a set of *function symbols*, and $ar : F \to S^*S$ is an *arity function* which assigns to each function symbol f an *arity* $ar(f) = s_1 \dots s_n s \in S^*S$. As usual, we collect all function symbols of arity σs in a set $F_{\sigma s} := \{f \mid f \in F \text{ and } ar(f) = \sigma s\}$.

By means of a given signature we can build terms of sort s. By $\mathcal{T}_s(F)$ we denote the set of all terms of sort s built on the basis of the set F. If the set F is understood by the context, then we simply write $\mathcal{T}_s$. If $\sigma = s_1 \dots s_n$ then $\mathcal{T}_\sigma := \{t_1 \dots t_n \mid t_i \in \mathcal{T}_{s_i}\}$ is the set of all words of terms of length n such that the i-th term is of sort s_i. The elements of $\mathcal{T}_\sigma$ we underline, e.g. $\underline{t} \in \mathcal{T}_\sigma$.

The set $\mathcal{T}_s$ of terms of sort s is defined inductively by:

(i) if $c \in F_s$ then $c \in \mathcal{T}_s$
(ii) if $\underline{t} \in \mathcal{T}_\sigma$ and $f \in F_{\sigma s}$ then $\underline{t}f \in \mathcal{T}_s$

By $\mathcal{T}_s^*$ we denote the set of all finite sequences of terms of the same sort s, $\mathcal{T}_s^\omega$ denotes the set of all denumerably infinite sequences of terms of the same sort s, and $\mathcal{T}_s^\infty := \mathcal{T}_s^* \cup \mathcal{T}_s^\omega$ denotes the set of all denumerable (finite or infinite) sequences. If the sort does not matter, then we write $\mathcal{T} := \{\mathcal{T}_s \mid s \in S\}, \mathcal{T}^*, \mathcal{T}^\omega$, or $\mathcal{T}^\infty$ respectively.

An *OSOS - signature* is a signature $(S, C \cup F, ar)$ where $C \cap F = \emptyset$.

The symbols of set C are called *constructors* and the symbols of set F are called *operators*.

The *domain* of our semantics consists of all terms of $\mathcal{T}(F \cup C)$ where the elements of $\mathcal{T}(C)$ are called *data* and the elements of $\mathcal{T}(F \cup C) \setminus \mathcal{T}(C)$ are called *error-values* (in short: errors).

Operational semantics

Let us consider any relation $\rightarrow \subseteq \mathcal{T} \times \mathcal{T}$. By $\rightarrow^*$ we denote the transitive and reflexive closure of $\rightarrow$, by $\leftarrow$ the reverse relation $\rightarrow$ ($t \leftarrow t'$ iff $t' \rightarrow t$), and by $\circ$ we mean the concatenation of relations. A term t is called *irreducible* if there is no term t' such that $t \rightarrow t'$.

We can uniquely extend $\rightarrow$ on the set $\mathcal{T}_\sigma$ by

$$t\underline{t} \rightarrow t'\underline{t}' \quad \text{if} \quad (t \rightarrow t' \text{ and } \underline{t} = t') \quad \text{or} \quad (t = t' \quad \text{and} \quad \underline{t} \rightarrow \underline{t}').$$

Relation $\rightarrow$ induces two relations $\Downarrow \subseteq \mathcal{T} \times \mathcal{T}$ and $\downarrow \subseteq \mathcal{T} \times \mathcal{T}$ where

$$\begin{array}{lll} t' \Downarrow t'' & \text{iff} & \exists t : \ t' \rightarrow^* t \quad \text{and} \quad t'' \rightarrow^* t, \\ t \downarrow t' & \text{iff} & t \rightarrow^* t' \quad \text{and} \quad t' \text{ is irreducible.} \end{array}$$

An irreflexive relation $\rightarrow \subseteq \mathcal{T} \times \mathcal{T}$ is called a *transition* iff

$$\begin{array}{ll} \text{(i)} & \rightarrow \subseteq \bigcup \{ (\mathcal{T}_s(F \cup C) \setminus \mathcal{T}_s(C)) \times \mathcal{T}_s \mid s \in S \} \\ \text{(ii)} & \text{if} \quad \underline{t} \rightarrow^* \underline{t}' \quad \text{then} \quad \underline{t}f \rightarrow^* \underline{t}'f \\ \text{(iii)} & \leftarrow^* \circ \rightarrow^* \subseteq \Downarrow. \end{array}$$

Condition (i) demands that transitions can only occur between terms of the same sort where constructor terms are always irreducible. Condition (ii) demands that any transition done at any subterm induces a transition of the whole term too. Finally, condition (iii) means confluence (or, in other words, the Church-Rosser property): if there are any two terms, say t' and t'', which can be got by transitions from a common term t, i.e. $t' \leftarrow^* \circ \rightarrow^* t''$ then $t' \Downarrow t''$ follows which allows the carrying out of transitions to a common term. Confluence guarantees that the order of the used transitions does not matter.

An *operational semantics* O is a mapping which assigns to each symbol $f \in F \cup C$ a function $f^O : \mathcal{T}_\sigma \rightarrow \wp(\mathcal{T}_s^\infty)$ if $ar(f) = \sigma s$ such that there is a transition relation $\rightarrow_O$ where

$$\forall \underline{t} : \in f^O(\underline{t}) \qquad \text{iff} \quad t_0 = \underline{t}f \quad \text{and} \quad \forall i : t_i \rightarrow_O t_{i+1}.$$

We say that $\rightarrow_O$, $\Downarrow_O$, and $\downarrow_O$ are *associated with* operational semantics O. Indeed, O is fully characterized by $\rightarrow_O$. If $t_0 t_1 t_2 \ldots \in f^O(\underline{t})$ then there are transitions $t_0 \rightarrow_O t_1 \rightarrow_O t_2 \rightarrow_O \ldots$. Therefore, the sequence $t_0 t_1 t_2 \ldots$ describes a possible way of computation starting with term t_0.

Each operational semantics defines an *abstract semantics* O^* where $f^{O^*} : \mathcal{T}_\sigma \xrightarrow{\circ} \mathcal{T}_s$ if $ar(f) = \sigma s$ by

$$f^{O^*}(\underline{t}) = \begin{cases} t & \text{if} \quad \underline{t}f \downarrow_O t \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Here, $\xrightarrow{\circ}$ denotes a partial function. If $f^{O^*}(\underline{t})$ is defined, then $\underline{t}f \downarrow f^{O^*}(\underline{t})$.

We call a term t *strict* with respect to the operational semantics O if there is a term t' such that $t \downarrow_O t'$. Since $\rightarrow_O$ is confluent, there exists at most one term t' with $t \downarrow_O t'$. $\underline{t}f$ may be a strict term also if $\underline{t}$ is nonstrict.

For any relation $\rightarrow$ the induced relation $\Downarrow$ is reflexive and symmetric, but not always transitive. However, if $\rightarrow$ is confluent, i.e. condition (iii) is fulfilled, then $\Downarrow$ is transitive, too. Therefore, it becomes an equivalence. As a result the relation $\Downarrow_O$ associated with operational semantics O is an equivalence. Moreover, by condition (ii) this equivalence becomes a congruence, i.e. if $\underline{t} \Downarrow_O \underline{t}'$ then $\underline{t}f \Downarrow_O \underline{t}'f$.

Equivalent terms describe, in principle, the same elements of the domain. Since the constructor terms are always irreducible there cannot be two different equivalent data. If one term of an equivalence class is strict, then all terms of this class are strict too.

SPECIFICATIONS AND MODELS

The syntax of specifications

Now we will explain in which way specifications of operators, i.e. elements of set F can be described. The meaning of such a specification is a class of models where each model is a certain operational semantics. Special models, defined by term-rewriting and narrowing, will be of particular interest.

The syntax of the specifications described here is meant to be the theoretical basis. Therefore, it is not very legible. A user-friendly specification language, called LOSOS, is presented in Bachmann (1994).

In addition to the sets F and C we use *variables* in order to build terms. Let X be a set of variables, the elements of which are called *standard* variables. Each standard variable has a sort associated by the arity function $ar : X \rightarrow S$. Similiar to function symbols we collect all variables of arity s into the set $X_s := \{x \mid ar(x) = s\}$. Moreover, we introduce another set $X_{\mathbb{N}} := \{x_n \mid n \in \mathbb{N}\}$ (disjoint with X) of *positional* variables ($\mathbb{N}$ denotes the set of natural numbers). As abbreviations we introduce $X_n := \{x_1, \ldots, x_n\} \subset X_{\mathbb{N}}$, $\underline{x}_n := x_1 \ldots x_n$ or, shortly: $\underline{x}$ if n is known by the context. A positional variable x_i can admit any sort. So, the terms which include variables are built now by

(i) if $t \in F_s \cup C_s \cup X_s \cup X_{\mathbb{N}}$ then $t \in \mathcal{T}_s$

(ii) if $\underline{t} \in \mathcal{T}_\sigma$ and $f \in F_{\sigma s}$ then $\underline{t}f \in \mathcal{T}_{\sigma s}$.

Additionally, we use the two special symbols $\asymp$ and Λ and the two special sorts e and l in order to build terms

(iii) if $t \in \mathcal{T}_{s_1}$ and $t' \in \mathcal{T}_{s_2}$ then $tt' \asymp \in \mathcal{T}_e$

(iv) if $\underline{t} \in \mathcal{T}_e^*$ then $\underline{t}\Lambda \in \mathcal{T}_l$.

A term $tt' \asymp$ is called an equation and a term $\underline{t}\Lambda$ is called an equation system. Note that Λ itself is a term too; the empty equation system.

Positions in a term can uniquely be described by occurrences, a finite subset of $\mathbb{N}_+^*$, where $\mathbb{N}_+ := \mathbb{N} \backslash \{0\}$. For a term t, the set $occ(t)$ of all occurrences is inductively defined by:

$$\text{if } \underline{t}f \in \mathcal{T}_s \text{ where } \underline{t} = t_1 \ldots t_n \text{ then}$$
$$occ(\underline{t}f) := \{\epsilon\} \cup \{i\alpha \mid 1 \leq i \leq n \text{ and } \alpha \in occ(t_i)\}.$$

If $t \in \mathcal{T}$ and $u \in occ(t)$ then $t(u) \in F \cup C \cup X \cup X_{\mathbb{N}} \cup \{\asymp, \Lambda\}$ means the symbol at occurrence u in t and t/u means the subterm of t at u, defined by:

$$\underline{t}f(\varepsilon) := f,\ \underline{t}f/\varepsilon := \underline{t}f, \text{ and } \underline{t}f(iu) := t_i(u),\ \underline{t}f/iu := t_i/u.$$

The set of all occurrences which are labelled by a function symbol is denoted by $focc(t)$:

$$focc(t) := \{u \mid t(u) \in F\},$$

and the set of all variables occurring in a term t is denoted by $var(t)$:

$$var(t) := \{t(u) \mid u \in occ(t) \quad \text{and} \quad t(u) \in X\}.$$

By means of occurrences we can simply describe term-replacement. If we replace the subterm t/u in t by t' then we get the term $t[u \leftarrow t']$, formally

$$\underline{t}f[\varepsilon \leftarrow t'] := t' \ \text{ and } \underline{t}f[iu \leftarrow t'] := t_1 \ldots t_i[u \leftarrow t'] \ldots t_n f.$$

Although the positional variables can admit any sort of set S, within a term such a variable has only one sort. This concept is formalized by binding sorts to positional variables. Binding is defined as follows:

if $t(ui) = x_k \in X_{\mathbb{N}}$ and $t(u) \in F_{\sigma s} \cup C_{\sigma s}$ where $\sigma = s_1 \ldots s_n$ then s_i is bound to x_k and

$$\text{if } (x_k t \asymp \in \mathcal{T}_e \text{ or } t x_k \asymp \in \mathcal{T}_e) \text{ where } t \in \mathcal{T}_s \text{ then } s \text{ is bound to } x_k.$$

Now, we can state our *restriction*:

Within a term, only one sort can be bound to each positional variable.

This allows us to define specifications for operators f of an *OSOS-signature*.

A specification, say Σ, is a function which assigns to each operator $f \in F_{s_1 \ldots s_n s}$ a finite sequence of pairs

$\Sigma(f) = (P_1, r_1)(P_2, r_2) \ldots (P_{n_f}, r_{n_f})$

where

$$\forall i : 1 \leq i \leq n_f \rightarrow P_i \in \mathcal{T}_l(F \cup C \cup X \cup X_n) \text{ and } r_i \in \mathcal{T}_s(F \cup C \cup X \cup X_n),$$

and if the positional variable x_k occurs in any term then the sort s_k is bound to it. Each pair (P_i, r_i) is called an (f, i)-rule.

In order to make the syntax of specifications more legible within the theoretical considerations we use, additionally, a slightly less formal form of notation which corresponds to the language LOSOS.

A term $\underline{t}f$ will be written as $f(t_1 \ldots t_n)$ and, if f is a binary operator (like $+$) we use infix notation and write $t_1 f t_2$ instead of $f(t_1, t_2)$ (like $t_1 + t_2$ instead of $+(t_1, t_2)$). An equation $tt' \asymp$ will be written as $t = t'$, an equation system $t_1 t_1' \asymp \ldots t_k t_k' \asymp \Lambda$ will be written as a sequence $t_1 = t_1' : \ldots : t_k = t_k'$ and, finally, an f-specification will be written as

$$f(x_1, \ldots, x_n) : P_1 \Rightarrow r_1 : \ldots : P_{n_f} \Rightarrow r_{n_f}.$$

Note that in an f-specification the number n_f of pairs may be zero and, if it is not so, then any equation system P_i may be empty! Each premiss of an f-specification can contain positional variables from X_n, where n is the number of arguments of symbol f, as well as any standard variable.

Models

A model of a specification must be an operational semantics. A specification may have, in general, several models. First, we define some different kinds of models. In the next chapter we describe how special models can be constructed by means of rewriting and narrowing.

A *substitution* is a function $\varphi : X \to \mathcal{T}$ where $\varphi(x) \in \mathcal{T}_s$ if $x \in X_s$. For substitutions, we write $x\varphi$ instead of $\varphi(x)$. Every substitution can be uniquely extended to a function $\varphi^* : \mathcal{T} \to \mathcal{T}$ by

$$x\varphi^* := x\varphi, \quad \underline{t}\varphi^* := t_1\varphi^* \ldots t_n\varphi^*, \quad \text{and } \underline{t}f\varphi^* := \underline{t}\varphi^* f.$$

We identify φ^* and φ and omit the $*$.

Substitutions $\varphi,\ \psi$ can be concatenated by: $t(\varphi \circ \psi) := (t\varphi)\psi$.

The *domain* of a substitution φ is the set of all variables which are changed: $dom\varphi := \{x \ : \ x\varphi \neq x\}$.

A substitution φ is called *ground* if $\forall x \in X : x\varphi \in \mathcal{T}(F \cup C)$, i.e. $x\varphi$ does not contain any variables and therefore, $dom\varphi = X$.

A given reflexive and transitive relation $\to^*$ can be extended to substitutions in the following way

$$\varphi \to^* \varphi' \quad \text{iff} \quad \forall x \in X : x\varphi \to^* x\varphi'.$$

Similarly, the relations $\Downarrow$ and $\downarrow$ can be extended to substitutions where a substitution, say φ, is irreducible if for all substitutions φ' with $\varphi \to^* \varphi'$ the equality $\varphi = \varphi'$ follows. A substitution φ is therefore called *strict* if a substitution φ' with $\varphi \downarrow \varphi'$ exists.

Furthermore, we need special substitutions of kind $\Delta : X_{\mathbb{N}} \to \mathcal{T}$. If $f \in F_{s_1 \ldots s_n s}$ then we define for each term $\underline{t}f$ the substitution

$\Delta_{\underline{t}f} : X_n \to \mathcal{T} \quad$ where $x_i\Delta_{\underline{t}f} := t_i$.

Now, let us assume that a certain operational semantics O is given.

For any equation $tt' \asymp$ we define the set of *solutions* as

$$sol_O(tt' \asymp) := \{\varphi \ : \ \varphi \quad \text{is ground, strict, and} \quad t\varphi \Downarrow_O t'\varphi\},$$

and for any equation system $\underline{t}\Lambda$ we define the solutions inductively as the intersections of the set of solutions of all equations contained in the system, by

$$sol_O(\Lambda) := \{\varphi \ : \ \varphi : X \to \mathcal{T}\} \quad \text{and} \quad sol_O(t\underline{t}\Lambda) := sol_O(t) \cap sol_O(\underline{t}\Lambda).$$

Of course, the solutions of an equation system depend on the given operational semantics O.

Moreover, let us take any specification Σ where $\Sigma(f) = (P_1, r_1) \ldots (P_n, r_n)$, any operational semantics O, and any term $\underline{t}f$ where $f \in F_{s_1 \ldots s_n s}$ is a certain operator. We define two sets of *images* of $\underline{t}f$ under Σ and O by:

$$w - im_{\Sigma,O}(\underline{t}f) := \{r_k\Delta_{\underline{t}f}\varphi \ : \ \varphi \in sol_O(P_k\Delta_{\underline{t}f}) \quad \text{and} \quad 1 \leq k \leq n_f\}$$

and $s - im_{\Sigma,O}(\underline{t}f) := \{r_k\Delta_{\underline{t}f}\varphi \ : \ \varphi \in sol_O(P_k\Delta_{\underline{t}f})$ and $\forall i : 1 \leq i < k \to sol_O(P_i\Delta_{\underline{t}f}) = \emptyset\}$.

Of course, $s - im_{\Sigma,O}(\underline{t}f) \subseteq w - im_{\Sigma,O}(\underline{t}f)$.

By means of these preparations we define an x-model ($x \in \{w, s\}$) of a specification Σ as any operational semantics, say O, where the associated transition relation $\to_O$ meets the proposition:

$$\text{if } \ \underline{t}f \to_O t \ \text{ then } \ t \in x - im_{\Sigma,O}(\underline{t}f) \ \text{ or } \ (t = \underline{t}'f \text{ and } \underline{t} \to_O \underline{t}').$$

Since an operational semantics is uniquely defined by a transition relation, we also say that a transition relation which fulfils the condition above is an x-model.

In such a sense, the empty relation is always an x-model. It is a remarkable fact that any specification has always a model and is, therefore, satisfiable. However, the empty model is indeed very general. It is better to focus on a stronger notion of models. To do this, we investigate the behaviour of solutions in a bit more detail.

As mentioned above: two equivalent terms describe the same element of our domain. If there is a transition $t \to^* t'$ then, of course, $t \Downarrow t'$ and therefore, both terms are equivalent.

Two substitutions φ, φ' are equivalent if $\varphi \Downarrow \varphi'$. If φ is a solution of an equation system P and $tt' \asymp$ is any equation from P then $t\varphi \Downarrow t'\varphi$ holds. Since φ and φ' are equivalent, $t\varphi \Downarrow t\varphi'$ and $t'\varphi \Downarrow t'\varphi'$ hold too. By transitivity we get $t\varphi' \Downarrow t'\varphi'$. That means, φ' is a solution of P.

This shows that a model is mainly characterized by the relation $\Downarrow$. If $\to\; = \;\emptyset$ is the empty transition then the induced relation $\Downarrow$ becomes the identity, i.e. $\Downarrow\; = \;\{(t,t) \mid t \in \mathcal{T}\}$. If $\to \subseteq \to'$ then $\Downarrow \subseteq \Downarrow'$. But, the converse implication does not hold. We define a relation $\precsim$ between transitions by

$$\to \precsim \to' \qquad \text{iff} \qquad \Downarrow \subseteq \Downarrow'$$

$\precsim$ is, in general, not an antisymmetric relation. It induces an equivalence $\sim := \precsim \cap \succsim$ where

$$\to \sim \to' \qquad \text{iff} \qquad \Downarrow = \Downarrow' \text{ holds.}$$

A *maximal x-model* is an x-model $\to$ such that for every x-model $\to'$ the implication

$$\text{if} \qquad \Downarrow \subseteq \Downarrow' \qquad \text{then} \qquad \Downarrow = \Downarrow' \text{ holds.}$$

By Zorn's lemma, maximal x-models always exist. Moreover, one can extend each model to a maximal one. However, this extension is clearly not always unique. Let us consider the following example:

We take only one sort s, i.e. $S = \{s\}$, and we have the two constructors $C_s = \{a, b\}$ and the only unary function symbol $F_{ss} = \{f\}$. An f-specification could be:

$$f(x_1) \;:\Rightarrow x.$$

Since we have only one rule in this specification, all w-models are also s-models. There are four nonequivalent maximal s-models:

M1:	$af^{i+1} \to af^i,\; bf^{i+1} \to af^i$ for $i \in I\!N$,
M2:	$af^{i+1} \to bf^i,\; bf^{i+1} \to bf^i$ for $i \in I\!N$,
M3:	$af^{i+1} \to af^i,\; bf^{i+1} \to bf^i$ for $i \in I\!N$.
M4:	$af^{i+1} \to bf^i,\; bf^{i+1} \to af^i$ for $i \in I\!N$.

Here, af^i means a term inductively defined by:

$$af^0 = a, \quad af^{i+1} = aff^i.$$

These three models induce the following equivalence classes:

M1:	$\{af^i \;:\; i \in I\!N\} \cup \{bf^{i+1} \;:\; i \in I\!N\}, \{b\}$,
M2:	$\{a\}, \{af^{i+1} \;:\; i \in I\!N\} \cup \{bf^i \;:\; i \in I\!N\}$,
M3:	$\{af^i \;:\; i \in I\!N\}, \{bf^i \;:\; i \in I\!N\}$.
M4:	$\{af^{2i} \;:\; i \in I\!N\} \cup \{bf^{2i+1} \;:\; i \in I\!N\}, \{af^{2i+1} \;:\; i \in I\!N\} \cup$ $\{bf^{2i} \;:\; i \in I\!N\}$.

Let us change the f-specification into

$$f(x_1) \;: x_1 = x : x = a \Rightarrow x \;: x_1 = x : x = b \Rightarrow a.$$

For this specification, M1 is also a maximal model but M2, M3, and M4 are not more models since a and b cannot be in the same equivalence class. All other maximal models are now equivalent to M1, e.g. the models

M5: $af^{i+1} \to af^i,\ af^{i+1} \to bf^{i+1},\ bf^{i+1} \to af^i$ for $i \in I\!N$,
M6: $af^i \to af^j,\ bf^i \to af^j$ for $i, j \in I\!N$ and $j < i$.

The second specification looks strange. A much simpler specification would be:

$$f(x_1)\ :\Rightarrow a.$$

Note, however, that now M5 is no longer a model of the last specification. Nevertheless, this transition is equivalent to M6.

For the existence of a unique maximal model it seems to be sufficient that for all terms $\underline{t}f$ any two images of $\underline{t}f$ under Σ and O are equivalent, i.e. that the implication

$$t, t' \in x - im_{\Sigma,O}(\underline{t}f)\ \Rightarrow\ t \Downarrow_O t'$$

holds. However, this is not so. For instance, let us consider the specification

$$f(x_1)\ : f(x) = x_1 \Rightarrow x.$$

Here, M3 as well as M4 are nonequivalent models. However, for both of them our implication above holds.

REWRITING AND L-NARROWING: AN APPROACH FOR THE OPERATIONAL SEMANTICS OF OSOS

For a given specification, say Σ, we construct now a transition relation $\to_\Sigma$, called *rewriting*, which turns out to be an s-model of Σ if $\to_\Sigma$ is confluent. That means, in order to determine a t' such that $\underline{t}f \to_\Sigma t'$, we need information about the set $x - im_{\Sigma,\Sigma}(\underline{t}f)$. At least one element of this set must be known. Such an element will be found by *narrowing*.

As already mentioned, we are dealing with rules of the form

$$\Sigma(f) = (P_1, r_1) \dots (P_{n_f}, r_{n_f}).$$

Next, we go on to define the application of rules. In principle, an (f, i)-rule (P_i, r_i) can be applied at occurrence u of a goal $G \in \mathcal{T}_l$ if

(i) the term $\underline{x}f$ matches with G/u and
(ii) the premise P_i holds after the match.

Due to our special kind of rules, the condition (i) is fulfilled if $G(u) = f$. The match is then described by the substitution $\Delta_{G/u}$, i.e. $G/u = \underline{x}\Delta_{G/u}f$. But in order to check whether the premise holds after the match we have to derive the modified P_i to Λ. This modification is done by $P_i\Delta_{G/u}$. We postpone the check and apply the (f, i)-rule onto G at u if $G(u) = f$ in the hope that $P_i\Delta_{G/u}$ holds. The application of (f, i) onto G at u replaces in G the subterm G/u by the result r_i of rule (f, i) regarding the match. Additionally, in order to avoid conflicts of variables, we rename in r_i each standard variable x of rule (f, i) by the auxillary variable $x^{(k)}$ using the substitution $\rho^{(k)}$ before the replacement is carried out. The substitution $\rho^{(k)}$ can be formally defined by $dom\rho^{(k)} = X_s$ and $\forall x \in X_s\ :\ x\rho^{(k)} = x^{(k)}$ where k is a certain natural number. So, we have $r_i\rho^{(k)}\Delta_{G/u}$ as the new subterm at u and the rewritten goal is $G[u \leftarrow r_i\rho^{(k)}\Delta_{G/u}]$. The combination of the rewritten goal with the modified premise $P_i\rho^{(k)}\Delta_{G/u}$ implements the postponement of the premise check. The new goal is now in detail

$$G[u \leftarrow r_i\rho^{(k)}\Delta_{G/u}] \bowtie (P_i\rho^{(k)}\Delta_{G/u}) \text{ where}$$
$$(\underline{t}\Lambda \bowtie \underline{t}'\Lambda) := \underline{tt}'\Lambda.$$

Since G does not contain any positional variables and u is a position in G we can shift the substitution $\Delta_{G/u}$ to the right and we get

$$G[u \leftarrow r_i\rho^{(k)}\Delta_{G/u}] \bowtie (P_i\rho^{(k)}\Delta_{G/u}) \;=\; (G \bowtie P_i\rho^{(k)})[u \leftarrow r_i\rho^{(k)}]\Delta_{G/u}.$$

This application of the (f,i)-rule at u of G is called an *L-narrowing* step. It should bring us a goal closer to the application of a *reflecting* step, where an equation e within the goal is removed by finding a most general unifier of e ($mgu(e)$). If goal G has length n, $1 \le i \le n$, and $\mu = mgu(G/i)$ then by a reflecting step at i we get the new goal

$$G\mu[i \leftarrow \epsilon] \;=\; G[i \leftarrow \epsilon]\mu.$$

The $mgu\ \mu$ is something important because it may describe a part of the solution of the goal. Most general unifiers used in reflecting steps are therefore composed together.

After this preparation we are in the position to define the derivations.

Let G be an equation system of length n, called *goal*, δ an idempotent substitution, and k a natural number.
Reflecting step:
If $1 \le i \le n$ and $\mu = mgu(G/i)$ then
$$\langle k : G, \delta\rangle \overset{(i)}{\longmapsto} \langle k : G[i \leftarrow \epsilon]\mu, \delta\mu\rangle.$$
L-narrowing step:
If $u \in focc(G)$, $(P_i, r_i)\ is\ a\ (G(u), i) - rule$, $\hat{P}_i = P_i\rho^{(k)}$, and $\hat{r}_i = r_i\rho^{(k)}$ then
$$\langle k : G, \delta\rangle \overset{(u:i)}{\longmapsto} \langle k+1 : (G \bowtie \hat{P}_i)[u \leftarrow \hat{r}_i]\Delta_{G/u}, \delta\rangle.$$
If α is any sequence consisting of elements (i) or $(u : i)$ then we define the derivation
$$\langle k : G, \delta\rangle \overset{\alpha}{\longmapsto} \langle l : Q, \sigma\rangle$$
by
$$\langle k : G, \delta\rangle \overset{\varepsilon}{\longmapsto} \langle k : G, \delta\rangle \quad \text{and}$$
$$\langle k : G, \delta\rangle \overset{s\alpha}{\longmapsto} \langle l : Q, \sigma\rangle \quad \text{iff} \quad \langle k : G, \delta\rangle \overset{s}{\longmapsto} \langle k' : G', \delta'\rangle \overset{\alpha}{\longmapsto} \langle l : Q, \sigma\rangle$$
where s is either an element (i) or $(u : i)$.
The sequence α is called derivation history.

Unfortunately, these notations are overloaded with technical overhead. But we can improve the legibility by dropping some parts which can be understood by the context. Firstly, we omit the number k. This number is only used in order to avoid conflicts of variables. Now we always take into consideration that by an L-narrowing step only *fresh* auxillary variables are introduced. This is done changing the premise P_i and the result r_i of rule (f,i) into $\hat{P}_i$ and $\hat{r}_i$. Secondly, we also omit the indices of the standard substitution Δ since they are mentioned before. In the abbreviated version we have for the reflecting step:

$$\langle G, \delta\rangle \overset{(i)}{\longmapsto} \langle G[i \leftarrow \epsilon]\mu, \delta\mu\rangle$$

and for the L-narrowing step:

$$\langle G, \delta\rangle \overset{(u:i)}{\longmapsto} \langle (G \bowtie \hat{P}_i)[u \leftarrow \hat{r}_i]\Delta, \delta\rangle.$$

Finally, if we are not interested in the derivation history we write simply

$$\langle G,\delta\rangle \longmapsto \langle Q,\sigma\rangle \text{ instead of } \langle G,\delta\rangle \stackrel{\alpha}{\longmapsto} \langle Q,\sigma\rangle.$$

Roughly described, L-narrowing is a stepwise derivation process α which transforms a pair $\langle G,\delta\rangle$ into a pair $\langle Q,\sigma\rangle$ where G, Q are equation systems and δ, σ are substitutions. In general, there are several derivation histories α such that

$$\langle G,\lambda\rangle \stackrel{\alpha}{\longmapsto} \langle \Lambda,\sigma\rangle$$

where λ is the identical substitution ($x\lambda = x$ for all variables x) and Λ is the empty equation system. However, for each given goal G and derivation history α there exists at most one derivation

$$\langle G,\delta\rangle \stackrel{\alpha}{\longmapsto} \langle Q,\sigma\rangle.$$

If such a derivation exists then the correspondig history α can be found by a breadth-first search. However, if such an α with a corresponding derivation does not exist then there are two different cases. Either, after some steps, it is possible to decide that there is no such derivation, or the procedure does not terminate. In the latter case the existence of such history is unknown.

In Bachmann (1994) it is shown:
If there is a derivation $\langle G,\lambda\rangle \stackrel{\alpha}{\longmapsto} \langle \Lambda,\varphi\rangle$ then there is also a derivation $\langle G\varphi,\lambda\rangle \stackrel{\alpha}{\longmapsto} \langle \Lambda,\psi\rangle$ with $G\varphi\psi = G\varphi$. This means, that the goal $G\varphi$ can be reduced to an empty system by narrowing with the same steps as before where the now derived substitution ψ does not have any influence on $G\varphi$. Therefore, in Bachmann (1994), the substitution φ was called a *solution* of G, and let us call it in this paper a narrowing-solution.

In general, a narrowing solution may contain variables. In order to avoid such a case we define for specifications Σ and goals G:

$$nsol_\Sigma(G) = \{\varphi\sigma \;:\; \langle G,\lambda\rangle \longmapsto \langle \Lambda,\varphi\rangle, \sigma : X \to \mathcal{T}(F \cup C), \quad \text{and} \quad \varphi\sigma \quad \text{is strict}\}.$$

$\varphi\sigma$ is always a ground substitution.

As an additional concept we define the set of *narrowing images* $n{-}im_\Sigma(\underline{t}f)$ of a given specification Σ and a term $\underline{t}f$ where $\Sigma(f) = (P_1,r_1)(P_2,r_2)\ldots(P_{n_f},r_{nf})$ by

$$\begin{aligned} n{-}im_\Sigma(\underline{t}f) := \{r_k\Delta_{\underline{t}f}\psi \;:\; & \psi \in nsol_\Sigma(P_k\Delta_{\underline{t}f}) \\ & \text{and } \forall i : 1 \le i < k \to nsol_\Sigma(P_i\Delta_{\underline{t}f}) = \emptyset\}. \end{aligned}$$

Now, we define the rewrite relation $\to_\Sigma$ on the basis of a given specification Σ by

$$\underline{t}f \to_\Sigma t \qquad \text{iff} \qquad t \in n{-}im_\Sigma(\underline{t}f) \quad \text{or} \quad (t = \underline{t}'f \text{ and } \underline{t} \to_\Sigma \underline{t}').$$

Is the so-defined rewrite relation always a transition? Conditions (i) and (ii) are fulfilled by the definition of the rewrite relation directly. However, condition (iii), namely the confluence, is a harder problem. We will deal with confluence in section 5 in detail.

Now, we investigate under which conditions the inclusion

$$n{-}im_\Sigma(\underline{t}f) \subseteq x{-}im_{\Sigma,\Sigma}(\underline{t}f)$$

holds, i.e. the rewrite relation is an x-model provided that it is confluent.

If $\varphi \in nsol_\Sigma(G)$ then, in general, this does not mean that $\varphi \in sol_\Sigma(G)$. Let us consider the following simple example:

$$a :\Rightarrow b \quad :\Rightarrow c.$$

Every ground substitution φ is a narrowing solution of $a = c$, but $sol_\Sigma(a = c) = \emptyset$, since $a \rightarrow_\Sigma b$ and therefore, $a \Downarrow_\Sigma c$ does not hold. This reason consists in the fact that, for the rewrite relation the leftmost possible (f, i)-rule, is always used, but in the definition of $nsol_\Sigma(G)$ this is not considered. Therefore, we restrict the used derivation histories. We do this by the redefinition of the L-Narrowing-step, which is now recursively defined by

$$\langle G, \delta\rangle \overset{(u:i)}{\longmapsto} \langle G[u \leftarrow \hat{r}_i]\Delta\varphi, \delta\varphi\rangle \text{ if}$$

$$\varphi \in nsol_\Sigma(P_i\Delta) \text{ and } \forall 1 \leq k < i \rightarrow \varphi \notin nsol_\Sigma(P_k\Delta).$$

Now, if $\varphi \in nsol_\Sigma(G)$ then $G\varphi$ does not contain any variables and, as a consequence of lemma 4.7 of Bachmann (1994) we get a derivation $\langle G\varphi, \lambda\rangle \longmapsto \langle \Lambda, \psi\rangle$ where each L-narrowing step can be replaced by a rewrite step. This means, $\varphi \in sol_\Sigma(G)$.

What about the inclusion $n - im_\Sigma(\underline{t}f) \subseteq s - im_{\Sigma,\Sigma}(\underline{t}f)$? In order for this inclusion to hold we have to be sure that if $sol_\Sigma(G)$ is not empty then at least one of its elements can be found by narrowing.

If $\varphi \in sol_\Sigma(G)$ then there exists a derivation $\langle G\varphi, \lambda\rangle \longmapsto \langle \Lambda, \psi\rangle$ where $G\varphi\psi = G\varphi$. As a consequence of lemma 5.7 of Bachmann (1994) we can conclude that there exists a derivation $\langle G, \lambda\rangle \longmapsto \langle \Lambda, \delta\rangle$ where $\varphi = \delta\sigma$ for a certain σ in case that no narrowing step is applied onto the part of $G\varphi$ which is added to G by the substitution φ. Since each solution φ is strict there also exists an equivalent irreducible solution φ' and a derivation $\langle G\varphi', \lambda\rangle \longmapsto \langle \Lambda, \psi\rangle$ where no narrowing step is applied onto the φ'-part of $G\varphi'$, i.e. $\varphi' \in nsol_\Sigma(G)$.
Altogether, we can state that

$$n - im_\Sigma(\underline{t}f) \subseteq s - im_{\Sigma,\Sigma}(\underline{t}f).$$

That the restriction to strict substitutions is important is shown by the following specification:

$$a :\Rightarrow f(a).$$

The equation $h(x, f(x)) = h(f(x), x)$ does not have any narrowing solution but

$$h(\,f(a)) \Downarrow h(f(a), a)\text{holds.}$$

In order to become an x-model, the rewrite-relation $\rightarrow_\Sigma$ must be confluent. However, we cannot expect that $\rightarrow_\Sigma$ always becomes a maximal model. Let us consider any f-specification of form

$$f(x_1) \ : g(x_1, x, y) = 0 \Rightarrow a \ :\Rightarrow b$$

where g means a partial recursive function which can be specified by our method as shown in section 3.3. It is well known (Matijasevich 1970) that the set of roots of $g(t, x, y)$ for some terms t are denumerable but not decidable. If, now, $g(t, x, y)$ does not have any roots, then narrowing may be a nonterminating process and there is no computable strategy, in general, to determine that $nsol_\Sigma(g(t, x, y) = 0) = \emptyset$. It follows that $b \notin n - im_\Sigma(\underline{t}f)$.

CONFLUENCE AND HIERACHIES

The confluence of models

In our definition of a model, the transition relation is confluent by condition (iii) $\leftarrow^*_O \circ \rightarrow^*_O \subseteq \Downarrow$. However, this demand complicates the achievement of an overview of all models of a given specification. Now we are going to weaken the notion of a model, and introduce an *x-semi-model*, which simplifies the situation by dropping confluence. Then, we present sufficient conditions such that confluence is guaranteed and therefore, an x-semi-model becomes an x-model.

The main problem to overcome is the proof of confluence of a semi-model. The known confluence proofs for conditional rewriting are based either on terminating rules (as in Ganzinger 1987) or on very strict syntactical conditions, (as in Bergstra & Klop 1982). The principle used is to prove strong confluence or local confluence. Strong confluence is sufficient for confluence, and local confluence together with termination is also sufficient for confluence (Hußmann 1985). However, nontermination is inherent to our approach and strong confluence turned out to be too strict. Therefore, we need another concept.

If $\rightarrow \subseteq \mathcal{T} \times \mathcal{T}$ is any relation then we define $\rightarrow^n$ inductively by:

$$\rightarrow^0 := \{(t,t) \ : \ t \in \mathcal{T}\}, \ \ \rightarrow^{n+1} := \rightarrow^n \circ \rightarrow .$$

This means $\rightarrow^n$ describes n steps of $\rightarrow$. By $\rightarrow^{n*}$ we denote the reflexive closure of $\rightarrow$, i.e. $\rightarrow^{n*}$ describes n or zero steps.

We call an element $t \in \mathcal{T}$ *confluent* if the implication

$$t \rightarrow^* t_1, t \rightarrow^* t_2 \quad \Longrightarrow \quad t_1 \Downarrow t_2$$

holds.

Proposition 1. If t is confluent and $t \rightarrow^* t'$ then t' is confluent.

Proof. trivial and omitted.

Corollary 2. $\rightarrow$ is confluent iff $\forall t \in \mathcal{T} : \ t$ is confluent.

To a relation $\rightarrow \subseteq \mathcal{T} \times \mathcal{T}$ we associate an operator $\omega : \wp(\mathcal{T} \times \mathcal{T}) \rightarrow \wp(\mathcal{T} \times \mathcal{T})$ defined as:

$$\begin{array}{lll}
\omega(S) := \{(s,t) \ : & s \Downarrow t & \text{and} \\
& \forall s',t' : & s \rightarrow s', t \rightarrow t' \Rightarrow \\
& & ((s,t') \in S \text{ or } \exists s'' : \ s \rightarrow s'' \text{ and } (s'',t') \in S) \text{ and} \\
& & ((s',t) \in S \text{ or } \exists t'' : \ t \rightarrow t'' \text{ and } (s',t'') \in S) \text{ and} \\
& & ((s,t') \in S \text{ or } (s',t) \in S \text{ or } (s',t') \in S)\}.
\end{array}$$

Lemma 3. If $S \subseteq S'$ then $\omega(S) \subseteq \omega(S')$.

Proof. trivial and omitted.

This means that ω is monotonic in the cpo $(\wp(\mathcal{T} \times \mathcal{T}), \subseteq)$ and, by the fixed point theorem of Bourbaki (e.g. Cohn 1965) the operator ω has a greatest fixed point, say $\mathcal{T}_\omega$.

Lemma 4. If $(s,t) \in \mathcal{T}_\omega$ and $s \rightarrow^m s_m, t \rightarrow^n t_n$ then $s_m \Downarrow t_n$.

Proof. By induction on (m,n).

<u>Case 1.</u> $m = 0$, i.e. $s = s_m$.

<u>Case 1.1.</u> $n = 0$, i.e. $t = t_n$ and, trivially $s_m \Downarrow t_n$.

<u>Case 1.2.</u> $n > 0$, i.e. $t \rightarrow t' \rightarrow^{n-1} t_n$.

Since $\mathcal{T}_\omega = \omega(\mathcal{T}_\omega)$, there exists an s' such that $s \rightarrow^{1*} s'$ and $(s',t') \in \mathcal{T}_\omega$ and, by induction $s' \Downarrow t_n$. Consequently, $s \Downarrow t_n$, i.e. $s_m \Downarrow t_m$.

Case 2. $m > 0$, i.e. $s \to s' \to^{m-1} s_m$.
Case 2.1. $n = 0$, this corresponds to case 1.2.
Case 2.2. $n > 0$, i.e. $t \to t' \to^{n-1} t_n$.
Now, $(s, t') \in \mathcal{T}_\omega$ or $(s', t) \in \mathcal{T}_\omega$ or $(s', t') \in \mathcal{T}_\omega$ and, by induction $s_m \Downarrow t_n$ in all three cases. We consider the special set Ω defined by:

$$\Omega := \{(s,t) \ : \ s, t \text{ confluent and } s \Downarrow t\}.$$

Lemma 5. $\mathcal{T}_\omega \subseteq \Omega$.
Proof. Let $(s,t) \in \mathcal{T}_\omega$. We show that s is confluent.

We assume that $s \to^* s', s \to^* s''$. Because of $(s,t) \in \mathcal{T}_\omega$ it follows $s \Downarrow t$ and therefore, there exists an t' such that $s \to^* t', t \to^* t'$. By lemma 4 we now get $s' \Downarrow t'$ and therefore, there exists a t'' with $s' \to^* t'', t' \to^* t''$, i.e. $t \to t''$. Again by lemma 4, we get finally, $s'' \Downarrow t''$.

In order to show the reverse order of the implication of lemma we present an important proof principle.
Lemma 6. If $S \subseteq \omega(S)$ then $S \subseteq \mathcal{T}_\omega$.
Proof. Let $\mathcal{C}$ be a maximal chain in the cpo $(\wp(\mathcal{T} \times \mathcal{T}), \subseteq)$, such that

$$\forall X \in \mathcal{C} : S \subseteq X \text{ and } X \subseteq \omega(X).$$

$\bigcup \mathcal{C}$ is the least upper bound of $\mathcal{C}$. That means that for all $X \in \mathcal{C}$ we have $X \subseteq \bigcup \mathcal{C}$ and since ω is monotonic, $\omega(X) \subseteq \omega(\bigcup \mathcal{C})$. On the other hand, $X \subseteq \omega(X)$ follows from $X \in \mathcal{C}$ and so, $X \subseteq \omega(\bigcup \mathcal{C})$. That means $\omega(\bigcup \mathcal{C})$ is an upper bound for $\mathcal{C}$ and $\bigcup \mathcal{C} \subseteq \omega(\bigcup \mathcal{C})$ which would extend the chain $\mathcal{C}$. But this is impossible since $\mathcal{C}$ is maximal. It remains $\bigcup \mathcal{C} = \omega(\bigcup \mathcal{C})$. Therefore, $\bigcup \mathcal{C}$ is a fixed point and $\bigcup \mathcal{C} \subseteq \mathcal{T}_\omega$, since $\mathcal{T}_\omega$ is the greatest fixed point. It was $S \in \mathcal{C}$, i.e. $S \subseteq \bigcup \mathcal{C}$ and, consequently, $S \subseteq \mathcal{T}_\omega$.
Lemma 7. $\Omega \subseteq \mathcal{T}_\omega$
Proof. Let $(s,t) \in \Omega$ and s', t' be any two terms with $s \to s', t \to t'$. Now, s', t' are confluent and $s \Downarrow t'$, $s' \Downarrow t$, and $s' \Downarrow t'$, i.e. $(s,t) \in \omega(\Omega)$.
Corollary 8. $\Omega = \mathcal{T}_\omega$

Next, we drop the confluence from the definition of a transition. An irreflexive relation $\to \subseteq \mathcal{T} \times \mathcal{T}$ is called a *semi-transition* iff

$$\begin{array}{ll} \text{(i)} & \to \subseteq \bigcup\{(\mathcal{T}_s(F \cup C) \backslash \mathcal{T}_s(C)) \times \mathcal{T}_s \mid s \in S\} \\ \text{(ii)} & \text{if } \ \underline{t} \to^* \underline{t}' \ \text{ then } \ \underline{t}f \to^* \underline{t}'f. \end{array}$$

A semi transition $\to_O$ is called an *x-semi-model* of specification Σ iff

$$\text{if } \ \underline{t}f \to_O t \ \text{ then } \ t \in x - im_{\Sigma,O}(\underline{t}f) \ \text{ or } \ (t = \underline{t}'f \text{ and } \underline{t} \to_O \underline{t}').$$

Of course, each x-model is a x-semi-model too.

We say that the condition **Cx** is fulfilled by the x-semi-model O ($x \in \{w, s\}$) of a specification Σ if the three parts **Cxa, Cxb**, and **Cxc** are satisfied, where:

Cxa: If $\underline{s}, \underline{t}$ are confluent terms, $\underline{s} \Downarrow \underline{t}$, $\underline{s}f \to_O s = r\Delta_{\underline{s}f}\varphi$, and $\underline{t}f \to_O t = r'\Delta_{\underline{t}f}\psi$then $r = r'$ and $\forall x \in var(r\Delta_{\underline{s}f}) : x\varphi \Downarrow_O x\psi$.

Cxb: If $\underline{t}$ is a confluent term and $\underline{t}f \to_O t = r\Delta_{\underline{t}f}\psi$
then $\forall x \in var(r\Delta_{\underline{t}f}) : x\psi$ is confluent and Δ_t is confluent.

Cxc: If $\underline{s}, \underline{t}$ are confluent terms, $\underline{s} \Downarrow \underline{t}$, and $\underline{t}f \rightarrow_O t = r\Delta_{\underline{t}f}\psi$ then there is a φ with $\underline{s}f \rightarrow_O s = r\Delta_{\underline{s}f}\varphi$.

The following lemma shows that the condition **Cx** guarantees that every x-semi-model becomes an x-model.

Lemma 9. Let $S := \Omega \cup \{(\underline{s}f, \underline{t}f) \ : \ (\underline{s}, \underline{t}) \in \Omega\}$.

If **Cx** is fulfilled by the x-semi-model$O$of specificationΣ then $S \subseteq \Omega$.

Proof. Remark: instead of $\rightarrow_O$ and $\Downarrow_O$ we write $\rightarrow$ and $\Downarrow$ respectively.

We show that $S \subseteq \omega(S)$. Then, by lemma 6. and corollary 8. we have $S \subseteq \mathcal{T}_\omega = \Omega$.

Let $(s,t) \in S$. If $(s,t) \in \Omega$ then $(s,t) \in \omega(\Omega) \subseteq \omega(S)$ by lemma 7. and lemma 3.

We assume now that $s = \underline{s}f, t = \underline{t}f$ where $(\underline{s}, \underline{t}) \in \Omega$.

Let s', t' be any two terms with $s \rightarrow s', t \rightarrow t'$.

<u>Case 1.</u> $s' = \underline{s}'f$ and $\underline{s} \rightarrow \underline{s}'$.

That means $\underline{s}'$ is confluent, $\underline{s}' \Downarrow \underline{t}$, and therefore, $(s', t) \in S$.

<u>Case 1.1.</u> $t' = \underline{t}'f$ and $\underline{t} \rightarrow \underline{t}'$.

That means analogously, $(s, t') \in S$.

<u>Case 1.2.</u> $t' = r\Delta_{\underline{t}f}\psi$.

By **Cxc** and **Cxa** there exists an s'' with $s \rightarrow s'' = r\Delta_{\underline{s}f}\varphi$ and $\forall x \in var(r\Delta_{\underline{s}f}) : x\varphi \Downarrow x\psi$. From this we conclude $s'' \Downarrow t'$.

If r is just a variable then t' as well as s'' are confluent by **Cxb** and the fact that $\underline{t}, \underline{s}$ are confluent. Consequently, $(s'', t') \in S$.

If r is not a variable then by **Cxb**: $(\Delta_{s''}, \Delta_{t'}) \in \Omega$. This also means $(s'', t') \in S$.

<u>Case 2.</u> $s' = r\Delta_{\underline{s}f}\varphi$.

<u>Case 2.1.</u> $t' = \underline{t}'f$ and $\underline{t} \rightarrow \underline{t}'$.

This is analogous to case 1.1 and we get $(s, t'), (s', t'') \in S$.

<u>Case 2.2.</u> $t' = r'\Delta_{\underline{t}f}\psi$.

By **Cxa** we know that $r = r'$ and the proof runs analogously to the cases 1.2 and 2.1 and we get $s' = s'', t' = t''$ and $(s', t') \in S$.

Altogether, we have $(s,t) \in \omega(S)$, since $s \Downarrow t$.

By induction on the structure of terms we get:

Corollary 10. If the condition **Cx** *is fulfilled by the x-semi-model O then O is an x-model.*

Of course, the condition **Cx** is very general and therefore there is no hope for a general verification method. In the next subsection, we present an important class of specifications and show that **Cx** can be replaced by a more practicable condition.

Hierachical specifications

The concept of hierarchical specifications is not new and has already been used in many papers, (e.g. Bergstra & Klop 1982; Kaplan 1984), but, with some additional restrictions which are too strong for us. The general concept is that a hierarchy of specifications induces a well-founded quasi-ordering $\precsim$ on terms which allows proof of properties by Noetherian induction:

If for a set $W \subseteq \mathcal{T}$ the implication $(\prec t) \subseteq W \Rightarrow t \in W$ holds then $W = \mathcal{T}$. Here, $(\prec t) := \{t' \mid t' \prec t\}$.

A hierarchical $OSOS$-Signature consists of a chain of inclusions

$$C = F^{(0)} \subset F^{(1)} \subset \cdots \subset F^{(m)}.$$

The *level* $l(f)$ of a function symbol $f \in F$ is the natural number $n = l(f)$ such that $f \in F^{(n)} - F^{(n)-1}$. For every constructor symbol $c \in C$ we set $l(c) = 0$.

Such a hierarchical signature induces a well-founded quasi-ordering on the signature F by:

$$f \precsim g \qquad \text{iff} \qquad l(f) \leq l(g).$$

The chain of symbols induces a chain of term-sets $\mathcal{T}^{(k)} := \mathcal{T}(F^{(k)} \cup X)$.

A specification Σ is said to be hierarchical if the signature F is hierarchical and for each rule

$$\Sigma(f) = (P_1, r_1) \cdots (P_{n_f}, r_{n_f})$$

the condition

$$P_i \subseteq \mathcal{T}^{l(f)-1}, r_i \in \mathcal{T}^{l(f)}, \text{ and } var(r_i) \subseteq var(P_i) \cup X_{\mathbb{N}}$$

is fulfilled.

Next, we introduce a special class of x-semi-models and modify the condition **Cx** to **Hx**. It turns out that each of such a special x-semi-model which fulfills **Hx** is an x-model of an hierarchical specification.

We define:

$$hsol_O(P\Delta_{\underline{t}f}) := \{\varphi \ : \ \varphi \in sol_O(P\Delta_{\underline{t}f}); \text{and } \forall x \in var(P\Delta_{\underline{t}f}) : \ x\varphi \text{ is confluent}\}.$$

By $x{-}him_{\Sigma,O}$ we mean the functions which are defined analogously to $x{-}im_{\Sigma,O}$ but on the basis of $hsol_O$, i.e.

$$w{-}him_{\Sigma,O}(\underline{t}f) := \{r_k\Delta_{\underline{t}f}\varphi \ : \ \varphi \in hsol_O(P_k\Delta_{\underline{t}f}) \quad \text{and} \quad 1 \leq k \leq n_f\},$$

$$s{-}him_{\Sigma,O}(\underline{t}f) := \{r_k\Delta_{\underline{t}f}\varphi \ : \ \varphi \in hsol_O(P_k\Delta_{\underline{t}f}) \text{ and } \forall i : 1 \leq i < k \rightarrow hsol_O (P_i\Delta_{\underline{t}f}) = \emptyset\}.$$

An x-semi-model O of specification Σ is called an *x-h-semi-model* if for all $t \in \mathcal{T}$:

$$\text{if } x{-}him_{\Sigma,O}(t) \neq \emptyset \text{ then } \exists t' \in x{-}him_{\Sigma,O}(t) : \ t \rightarrow_O t'.$$

The condition **Hx** is defined as:

Hsa: If $\underline{t}$ is confluent, (P, r) is an f-rule, and $\varphi, \psi \in hsol_O(P\Delta_{\underline{t}f})$ then $\forall x \in var(r\Delta_{\underline{t}f}) : x\varphi \Downarrow_O x\psi$.

Hwa: **Hwa** and if $\underline{t}$ is a confluent term, $(P, r), (P', r')$ are f-rules, and $hsol_{\Sigma,O}(P\Delta_{\underline{t}f}) \cap hsol_{\Sigma,O}(P'\Delta_{\underline{t}f}) \neq \emptyset$ then $r = r'$.

Hxb: If $\underline{t}$ is confluent and $\underline{t}f \rightarrow_O t$ then Δ_t is confluent.

By S_k we denote the set

$$S_k := \Omega \cup \{(\underline{s}f, \underline{t}f) \ : \ (\underline{s}, \underline{t}) \in \Omega \text{ and } f \in F^{(k)}\}.$$

S_k is called *downward confluent* if $\forall i < k : \ S_i \subseteq \Omega$.

Lemma 11. Let S_k be downward confluent. If **Hx** *is fulfilled by the x-h-semi-model O of the specification Σ then $S_k \subseteq \Omega$.*

Proof. It is similar to the proof of lemma 9. i.e. we show that $S_k \subseteq \omega(S_k)$.
Let $(s,t) \in S_k$. If $(s,t) \in \Omega$ then $(s,t) \in \omega(\Omega) \subseteq \omega(S)$ by lemma 7. and lemma 3.
We assume now that $s = \underline{s}f, t = \underline{t}f$ where $(\underline{s}, \underline{t}) \in \Omega$.
Let s', t' be any two terms with $s \to s', t \to t'$.

Case 1. $s' = \underline{s}'f$ and $\underline{s} \to \underline{s}'$.
That means $\underline{s}'$ is confluent, $\underline{s}' \Downarrow \underline{t}$, and therefore, $(s', t) \in S_k$.

Case 1.1. $t' = \underline{t}'f$ and $\underline{t} \to \underline{t}'$.
That means analogously, $(s, t') \in S_k$.

Case 1.2. $t' = r\Delta_{\underline{t}f}\psi$.
It means that $\psi \in hsol_O(P\Delta_{\underline{t}f})$ where (P, r) is any f-rule. Let $uv \asymp$ be any equation from P. Then, $u\Delta_{\underline{t}f}\psi \Downarrow v\Delta_{\underline{t}f}\psi$, $u\Delta_{\underline{s}f}\psi \Downarrow u\Delta_{\underline{t}f}\psi$, and $v\Delta_{\underline{t}f}\psi \Downarrow v\Delta_{\underline{s}f}\psi$. All these terms are confluent since we have a hierarchical specification and S_k is downward confluent. Therefore, $\Downarrow$ is transitive for these terms and we get $u\Delta_{\underline{s}f}\psi \Downarrow v\Delta_{\underline{s}f}\psi$, i.e. $\psi \in hsol_O(P\Delta_{\underline{s}f})$ and $x - him_{\Sigma,O}(P\Delta_{\underline{s}f}) \neq \emptyset$ follows. From this we conclude the existence of an $\varphi \in hsol_O(P\Delta_{\underline{s}f})$ with $s \to s'' = r\Delta_{\underline{s}f}\varphi$ since O is an x-h-semi-model, and $s'' \Downarrow t'$.
If r is just a variable then t' as well as s'' are confluent and, consequently, $(s'', t') \in S_k$.
If r is not a variable then by **Hxb**: $(\Delta_{s''}, \Delta_{t'}) \in \Omega$. This also means $(s'', t') \in S_k$.

Case 2. $s' = r\Delta_{\underline{s}f}\varphi$.

Case 2.1. $t' = \underline{t}'f$ and
$\underline{t} \to \underline{t}'$.
This is analogous to case 1.1 and we get $(s, t'), (s', t'') \in S$.

Case 2.2. $t' = r'\Delta_{\underline{t}f}\psi$.
By **Hxa** we know that $r = r'$ and $\forall x \in var(P\Delta_{\underline{s}f}) : x\varphi \Downarrow x\psi$. Now the proof runs analogously to the cases 1.2 and 2.1 and we get $s' = s'', t' = t''$ and $(s', t') \in S_k$.
Altogether, we have $(s,t) \in \omega(S_k)$, since $s \Downarrow t$.

If S_k is downward confluent then $S_k \subseteq \Omega$ and so, S_{k+1} is downward confluent too. By induction it follows:

Corollary 12. If **Hx** *is fulfilled by the x-h-semi-model O then O is an x-model.*

It remains to give a verification method for the condition **Hx**. This can also be done using Noetherian induction. Here, we take the recursive-path-ordering $\precsim$ on terms induced by the quasi-ordering of the signature. This recursive-path-ordering is described, for instance, in Dershowitz (1987), and is a well-founded quasi-ordering.

We sketch the verification for the example of the specification of partial recursive functions.

Hxa is obviously fulfilled by all the rules

$$\begin{array}{rl}
\underline{zero}(x_1,\ldots,x_n): & \Rightarrow 0;\\
\underline{pos}_i(x_1,\ldots,x_n): & \Rightarrow x_i;\\
h(x_1,\ldots,x_n): & \Rightarrow f(g_1(x_1,\ldots,x_n),\ldots,g_m(x_1,\ldots,x_n));\\
h(x_1,\ldots,x_n,x_{n+1}): & x_{n+1}=0 \Rightarrow f(x_1,\ldots,x_n);\\
: & x_{n+1}=succ(y) \Rightarrow g(x_1,\ldots,x_n,y,h(x_1,,\ldots,x_n,y));\\
lf(x_1,\ldots,x_n,x_{n+1}): & f(x_1,\ldots,x_n,x_{n+1})=0 \Rightarrow x_{n+1}\\
: & f(x_1,\ldots,x_n,x_{n+1})=succ(y) \Rightarrow lf(x_1,\ldots,x_n,succ(x_{n+1})).
\end{array}$$

The only problem we get is with respect to **Hxb**.
We can assume that the symbols $f, g_1, \ldots, g_m$ have a lower level in the hierarchy than the symbol h, and using the recursive path ordering we get for all terms $t_1, \ldots, t_{n+1}, t$:

$$\begin{array}{rl}
f(g_1(t_1,\ldots,t_n),\ldots,g_m(t_1,\ldots,t_n)) & \prec h(t_1,\ldots,t_n),\\
f(t_1,\ldots,t_n) & \prec h(t_1,\ldots,t_n,t_{n+1}), \text{ and}\\
g(t_1,\ldots,t_n,y,h(t_1,,\ldots,t_n,t)) & \prec h(t_1,\ldots,t_n,t_{n+1})
\end{array}$$

where $y\varphi = t$ means any confluent solution of $t_{n+1} = succ(y)$ such that $t \prec t_{n+1}$.
If we assume that all terms s less than the left-hand sides of these rules are confluent and $t \prec t_{n+1}$, then **Hxb** holds too. The additional condition $t \prec t_{n+1}$ is fulfilled if, for instance, the x-h-semi-model uses constructor terms as solutions.

If we restrict the rewrite relation to

$$\underline{t}f \rightarrow_\Sigma t \quad \text{iff} \quad t \in in\!-\!im_\Sigma(\underline{t}f) \quad \text{or} \quad (t = \underline{t}'f \text{ and } \underline{t} \rightarrow_\Sigma \underline{t}')$$

where

$$\begin{array}{rl}
in\!-\!im_\Sigma(\underline{t}f) := \{r_k\Delta_{\underline{t}f}\psi : & \psi \in nsol_\Sigma(P_k\Delta_{\underline{t}f})\\
& \text{and } \forall x \in var(r_k\Delta_{\underline{t}f}) : x\psi \text{ is irreducible}\\
& \text{and } \forall i : 1 \le i < k \rightarrow nsol_\Sigma(P_i\Delta_{\underline{t}f}) = \emptyset\},
\end{array}$$

then this is an x-h-semi-model. Moreover, an irreducible term is always confluent. That means, if such a rewrite relation fulfilles **Hx** then it is an x-model. Clearly, for the specification of partial recursive functions above, the rewrite relation is an x-model.

CONCLUSIONS

As a general result, it turned out that the proposed method can be used in order to specify algorithms in a very abstract way. Additionally, rewriting, together with narrowing provides a prototype of a model which can be carried out on the basis of a given specification.

The following open questions must still be answered in order to make this method useful in practice:

Efficiency:
The efficiency of the prototype model, based on rewriting and narrowing, is very low if general breadth-first strategies are used. The reason consists not in the several rewrite and narrowing steps as the implementation has shown. Therefore, optimization techniques for breadth-first strategies must be developed or adapted from other narrowing methods.

Higher-order specifications
For practical purpose, higher order specifications in connection with polymorphism become more and more important. Therefore, operational semantics for higher order specifications must be investigated.

Extended proof methods
More effort must be made to extend the presented method in order to prove properties, such as the existence of a unique maximal model.

ACKNOWLEDGMENT

This research was sponsered by Kuwait University research grant number SM-081.

REFERENCES

Bachmann, P. 1994. LOSOS, a Language for Operational Semantics Oriented Specification. Technical Report, Kuwait University.

Bidoit, M. & Choppy, C. 1985. ASSPECIQUE: an intregrated environment for algebraic specifications. Proceedings of the Joint Conference on Theory and Practice of Software Development, Berlin, LNCS **186: 246-260.**

Bachmann, P., Drescher, T. & Nieke, S. 1990. SPECI-90 Nutzerhandbuch. TU Dresden.

Burstall, R. M. & Goguen, J.A. 1977. Putting theories together to make specifications. Proceedings of the fifth International Joint Conference on Artificial Intelligence, Cambridge, Massachusetts. pp 1045-1058.

Bergstra, J.A., Heering, J. & Klint, R. 1990. Module Algebra. Journal of the ACM **37(2)**: 335-372.

Bergstra, J.A. & Klop, J.W. 1982. Conditional rewrite rules: confluence and termination. Report IW 198/82, Mathematisch Centrum Amsterdam.

Bergstra, J.A. & Klop, J.W. 1986. Process Algebra: specifications and verification in bisimulation semantics. Proceedings of CWI Symposium of Mathematics and Computer Science II, North Holland, Amsterdam: 61-64.

Bachmann, P., Oberschelp, W., Thalheim, B. & Vossen, G. 1990. The Design of RAD: Towards an Interactive Toolbox for Database Design. Aachener Informatik-Berichte, No. 90-28.

Broy, M., & Wirsing, M. 1982. Partial abstract data types. Acta Informatica **18**: 47-64.

Cohn, P.M. 1965. Universal Algebra. Harper and Row, New York, N.Y.

Dershowitz, N. 1987. Termination of Rewriting. Journal of Symbolic Computation **3**: 69-116.

Ehrich, H.-D., Gogolla, M. & Lipeck, U.W. 1989. Algebraische Spezifikation abstrakter Datentypen. B. G. Teubner, Stuttgart.

Ehrig, H. & Mahr, B. 1990. Fundamentals of Algebraic Specification 2. Module Specifications and Constraints. EATCS Monographs on Theoretical Computer Science **21,** Springer, Verlaag.

Fey, W. 1989. Pragmatics, Concepts, Syntax, Semantics and Correctness Notions of ACT TWO. Dissertation, Berlin.

Fribourg, L. 1985. SLOG: a logic programming language interpreter based on clausel superposition and rewriting. Proceedings of IEEE Symposium on Logic Programming: 172-184.

Ganzinger, H. 1987. Ground term confluence in parametric conditional equational specifications, Proceeding of STACS 87, Springer **LNCS 247:** 216-227.

Geupel, O. 1988. Terminationsbeweise bei Termersetzungssystemen. Diplomarbeit, TU Dresden.

Gogolla, M. 1987. On parametric algebraic specifications with clean error handling. TAPSOFT 87, Pisa, **LNCS 249: 81-95.**

Goguen, J. A., Thatcher, J. W., Wagner, E. G. & Wright, J. B. 1975. Abstract data types as initial algebras and the correctness of data representations. Proceedings of Conference on Computer Graphics, Pattern Recognition and Data Structures: 89-93.

Guttag, J. V., Horowitz, & Muser, D.R. 1978. The design of data type specifications. In: **Yeh, R.T. (Ed).** Current Trends in Programming Methodology, Prentice Hall, Englewood Cliffs **4: 60-79.**

Huet, G. & Hullot, J.-M. 1982. Proofs by induction in equational theories with constructors. Journal Computational System Science **25 (2): 239-266.**

Huet, G. 1980. Confluent reductions: Abstract Properties and Applications to Term Rewriting Systems. ACM Journal **27(4)**: 214-256.

Hußmann, H. 1985. Rapid prototyping for algebraic specifications: RAP system user's manual. Report MIP-8504, Universität Passau.

Hußmann, H. 1989. Nichtdeterministische algebraische Spezifikation. Dissertation, Universität Passau.

Kaplan, S. 1984. Conditional rewrite rules.Theoretical Computer Science **33: 175-193.**

Knuth, D.E. & Bendix, P.B. 1970. Simple word problems in universal algebras. In: **Leech, J.(Ed)** Computational Problems in Abstract Algebra Pergamon Press, Oxford.

Kuske, D. 1991. Narrowing-Strategien. Diplomarbeit, TU Dresden.

Lescanne, P. 1983. Computer Experiments with the REVE term rewriting systems generator. Proceedings of the tenth ACM Symposium on Principles of Programming Languages.

Matijasevich, Y. 1970. Enumerable sets are diophantine. Dokladi Academi Nauk **191**: 279-282.

Möller, B. 1985. On the algebraic specification of infinite objects – ordered and continuous models of algebraic types. Acta Informatica **22**: 537-578.

Möller, B. 1987. Higher-order algebraic specifications. Fakultät für Mathematik und Informatik der TU München, Habilitationsschrift.

Qian, Z. 1990. Higher-order order sorted algebras. Proceedings of the second International Conference on Algebraic and Logic Programming, Nancy, LNCS **315**: 211-231.

Reichel, H. 1984. Structural Induction on Partial Algebras, Akademie Verlag, Berlin.

Smolka, G., Nutt, W., Goguen, J.A. & Meseguer, J., 1987. Order-sorted equational computation. Universität Kaiserslautern, SEKI-Bericht SR-87-14.

Wirsing, M. 1989. Algebraic Specification. Bericht MIP-8914, Universität Passau.

Zilles, S. N. 1974. Algebraic specification of abstract data types. Computation Structure Group memo 119, Laboratory for Computer Science, MIT.

(Accepted 29 January 1996)

المواصفات التشغيلية الموجة نحو الدلاليات

بيتر باخمان

جامعة كوتوبس للعلوم التكنولوجية

ص. ب 101344 و D - 03013 كوتبوس/المانيا

خلاصة

إن الفكرة الرئيسية في الطريقة المقترحة هي توصيف الخوارزميات بدلا من الدوال. ويمثل الحد هنا حالة حساب، إضافة إلى إنه يمثل عنصر بيانات. وتصف علاقة التحول المندمج للعناصر سلوك عملية الحوسبة. ومن الممكن أن تكون علاقة التحول هذه نموذجا لمواصفات. ويمكن التمييز بين النماذج الضعيفة والقوية حيث تأخذ النماذج القوية تسلسل قواعد المواصفات في الحسبان. إن القوة التعبيرية لطريقة المواصفات المقترحة تسمح بوصف جميع الدوال الارتدادية الجزئية. وبعملية دمج إعادة الكتابة مع طريقة خاصة للتضييق تنتج نماذج أولية تسمح بتحديد المواصفات.ونعرض هنا أيضا كيف أن الاندماج في المواصفات المتسلسلة الهرمية يمكن ضمانه في علاقات التحول غير المنتهية. وأخيرا نقدم عرضا موجزا لتنفيذ الطريقة المقترحة.