Abstract: The questions of specification and particularly extensions of abstract data types are investigated within the framework of many-sorted logic and model theory. A new constructive method of extensions of abstract specification is presented defining a notion of a first order constructor. This is useful for step-wise development of specifications. A first order relative completeness and consistency theorem is proved guaranteeing that the extension method is sound. This is a syntactic preservation theorem for type extension. A corresponding semantic preservation theorem follows as a simple generalization of a known result of model theory.

Induction schemas for constructors are introduced as important rules for proving properties of objects specified by first order axioms within a typed language. It is conjectured that such set of schema constitute a minimal sufficient set for proving inductive properties that hold in certain natural models.

Implementations are considered as faithful interpretation between theories and relation to Hoare's method of proving correctness of data representation is briefly discussed.
1. INTRODUCTION

Algebraic approaches to data types have been studied in Gutt (75), Zill (75), GTW (76), EKP (78), Ehri (78), Wand (79), Nour (79), and others. Here, we wish to consider the questions of specification, extension and implementation from the point of view of first order logic. We believe that there are some fundamental points which could be better understood in light of the material in this paper. Thus, even if one's inclination is algebraic, this treatment cannot but enhance one's detailed understanding. Preliminary versions of this work were first reported in Nour (78) and Nour (78a). A more detailed version appears in Nour (79).

There has been much effort in the past to use the notions of logic, particularly proof-theoretic notions, to reason about the computation performed by programs, and to attempt to prove programs behave correctly according to some correctness criteria. Since the abstract data type methodology was introduced, similar ideas are being attempted for programs manipulating abstract data. Hoar (72) is the first of such efforts. The ALPHARD methodology [WLS (76)] is another much further developed approach.

There is also work done in considering a programming language as being predicate logic plus control structure [Kowa (74)]. There is an implemented language which uses ideas along these lines with Resolution Logic [Robi (65)]. This is the language PROLOG of CKPR (72). All such research efforts will benefit from a good understanding of the logical foundations of data abstraction techniques. This work may also be relevant to the "first order programming logic" of CM (79).

We investigate the question of data type specification, particularly that of extension using first order many-sorted theories and model theory. Possible applicability as well as the limitations of such an approach are studied. Other independent research on first order or model-theoretic approaches to data types are reported in BDPP (79) and BMM (79).

The first two sections define some notions and present some known results of model theory and logic which apply to the problem at hand. Sections 3 and 4 explore the limitations of possible first order approaches and suggest ways of getting around the difficulties. These sections are essentially based on known results of model theory and are expository in nature.

In section 5 we define a notion of extension for many-sorted theories in which a notion of first order constructor signature is defined.

The main new theorem of this paper is a relative completeness and consistency theorem for many-sorted first order logic which is required for data type extensions. The theorem makes use of a property we have defined based on a notion due to Henkin. We define the notion of an S-sorted theory being Henkin relative to a restriction to a sub-theory with fewer sorts and operations. It is used to prove a syntactic preservation theorem for data types. We also prove a general-
ization of a known result of model theory to many sorts to show a semantic pre-
servation of the data type extended to a new type.

Finally, we conclude with a notion of implementation, we suggest the use of
faithful interpretation between theories as being appropriate, and we briefly
discuss how Hoare's work may be seen in this light.

The author wishes to thank Dr. Joseph Goguen of SRI and UCLA Computer Science
and Professor Donald A. Martin of UCLA Mathematics Department for their comments
on an earlier version.

2. MANY-SORTED THEORIES AND MODELS

In this section we consider data type specifications from the point of view
of first order logic. Some basic notions of logic and model theory are outlined
below. However, we will not cover such basic notions as quantifiers, well-formed
formulae, and first order theories.

As in the algebraic approach to data abstraction, it is more convenient to
use a many-sorted language. The use of many-sorted languages does not add to the
logical power of systems, for by using a unary predicate \( P_s(x) \) for "\( x \) is of type
\( s \)" one can in principle reduce to ordinary (single-sorted) logic. For more
details see KK (71), for example. We shall give a brief account of the notions
which are useful to us here.

First, we extend the notion of signature to include predicate letters as
follows. Let \( S \) be a non-empty set, whose members are called sorts. An \( S \)-sorted
signature is a collection of symbols which fall into two disjoint classes:

(a) A set of predicate symbols: for each \( n>0 \) and each \( n \)-tuple \( <s_1,\ldots,s_n> \)
of sorts in \( S \), there is a set (possibly empty) of \( n \)-place predicate
symbols each of which is said to be of rank \( (s_1,\ldots,s_n) \).

(b) A set of function symbols: for each \( n>0 \) and each \( (n-1) \)-tuple
\( <s_1,\ldots,s_n,s> \) of sorts in \( S \), there is a set (possibly empty of \( n-
place function symbols, each of which is said to be of rank
\( <s_1,\ldots,s_n,s> \) of arity \( (s_1,\ldots,s_n) \) and of sort \( s \). Constants are func-
tions of empty arity and rank \( <\lambda,s> \), where \( \lambda \) is the empty string.

In addition, there is up to a countable supply of variables of each sort, \( <X_s> \)
for \( s \) in \( S \), disjoint from the other symbols in the signature.

An \( S \)-sorted language consists of an \( S \)-sorted signature plus the sentential
connectives \( \land, \lor, \neg, \rightarrow \) and the quantifiers \( \exists_s, \forall_s \) one for each sort \( s \in S \). We
often identify the notion of signature and language for convenience.

The notions of terms and well-formed formulae (wff) are easily extended to
many-sorted languages. The details may be found in KK (71).

An \( S \)-sorted structure \( A \) of signature \( \Sigma \) consists of:

1) A set \( A_s \), called the carrier of sort \( s \), for each \( s \in S \),

2) For every predicate symbol \( p \) of rank \( (s_1,\ldots,s_n) \), a relation
p_{A^s} x_1 \ldots x_{n^s} \in \mathcal{A}

(3) For every function symbol \( f \in \mathcal{L} \) of rank \( <s_1, \ldots, s_n, s> \), a function
\[
f_{A^s} : x_1 \ldots x_{n^s} \rightarrow A^s.
\]
In particular, if \( f \) is a constant symbol of sort \( s \), \( f_{A^s} \in A^s \).

Let \( A \) be an \( S \)-sorted structure. Let \( \theta \) be an assignment, \( \theta = \{ \theta : X \rightarrow A^s \} \), \( A \theta \), and let \( \phi \) be a formula with free variables among \( x_1, \ldots, x_n \), where \( x_i \) is of sort \( s_i \). Write \( \mathcal{A} \models \phi(a/x) \) for "\( \phi \) is true in \( \mathcal{A} \) when \( x \) is assigned \( a \)\), where \( x = (x_1, \ldots, x_n) \) and \( a = (a_1, \ldots, a_n) \), \( a_i \in A_{s_i} \) if \( x_i \) is of sort \( s_i \). We say that \( \mathcal{A} \) satisfies \( \phi \) with assignment \( \theta \).

For a sentence (closed formula) \( \beta \), we say \( \mathcal{A} \) models \( \beta \), written \( \mathcal{A} \models \beta \), provided the sentence is true in \( \mathcal{A} \). A formula \( \beta \) is said to be satisfiable if it has a model.

A first order many-sorted theory is a first order theory whose language is many-sorted. In order to present a first order many-sorted theory \( T \) we need only give the signature \( \mathcal{L} \) and a set of sentences \( \mathcal{A} \) called the non-logical axioms of \( T \). By a model of a theory \( T \) we mean a structure of the same signature as \( T \) which models the axioms of \( T \).

Two structures \( A \) and \( A' \) of the same signature are isomorphic iff there is a 1-1 function \( h \) mapping \( A \) onto \( A' \) such that:

(i) For each n-placed predicate \( p_{A^s} \) of \( A \) and the corresponding predicate \( p_{A'} \)
    of \( A' \), \( p_{A'}(a_1, \ldots, a_n) \) iff \( p_{A^s}(h(a_1), \ldots, h(a_n)) \).

(ii) For each n-placed function \( f_{A^s} \) of \( A \) and the corresponding function \( f_{A'} \)
    of \( A' \), \( h(f_{A^s}(a_1, \ldots, a_n)) = f_{A'}(h(a_1), \ldots, h(a_n)) \).

(iii) For each constant \( a \) of \( A \) and the corresponding constant \( a' \) of \( A' \),
    \( h(a) = a' \).

The function \( h \) above is called an isomorphism of \( A \) onto \( A' \) (or between \( A \) and \( A' \)). Write \( A \approx A' \) to denote \( A \) is isomorphic to \( A' \).

Two structures \( A \) and \( B \) are said to be elementarily equivalent denoted by \( A \equiv B \)
iff for every sentence \( \beta \), \( A \models \beta \) iff \( B \models \beta \). The following is a useful well-known result for the discussion that follows and will be stated without proof.

**Theorem 2.1.** Let \( A \) and \( B \) be two structures. If \( A \equiv B \), then \( A \approx B \). If \( A \) is finite, the converse also holds. \( \Box \)

For any structure \( A \), the set of sentences true in \( A \) is called the theory of the structure \( A \), written \( \text{Th}(A) \). The following well-known theorem ties a few important notions together.

**Theorem 2.2.** For a consistent theory \( T \), the following are equivalent:

(a) \( T \) is complete

(b) every two models of \( T \) are elementarily equivalent

(c) for every model \( A \) of \( T \), \( T \) is equivalent to \( \text{Th}(A) \). \( \Box \)
Theorem 2.3. (Loewenheim-Skolem-Tarski). If a theory $T$ has an infinite model, then it has infinite models of all cardinalities. $\Box$

We generalize the notions of extension, restriction and expansion of one-sorted logic to many sorted languages and structures as follows. A first order theory $T$ is an extension of a theory $T_0$ if the signature of $T$ contains that of $T_0$ and every theorem of $T_0$ is a theorem of $T$. $T_0$ is called a restriction of $T$. Let $\Sigma_0$ and $\Sigma$ be two signatures such that $\Sigma_0$ is a $\Sigma_0$-reduct of $A$ if it can be obtained from $A$ by omitting some functions and predicates of $A$. We say that $A$ is a $\Sigma$-expansion of $A_0$.

3. SPECIFICATIONS AS FIRST ORDER THEORIES

In this section we define a first order notion of presentation, and contrast it with an algebraic presentation. A first order presentation is a triple $P=<(S, \Sigma, C)>$, where $\Sigma$ is an $S$-sorted signature in the sense of section 2, including predicate letters, and $C$ is a set of first order axioms over the language given by $\Sigma$. A first order presentation $P$ defines a first order $S$-sorted theory $T$ of signature $\Sigma$ and axioms $C$.

Let $A$ be a structure of signature $\Sigma$ which models $T$. Then $A$ is a representation of a data type, i.e., it satisfies the properties presented by $P$. But are there other structures which also model $T$? By the Loewenheim-Skolem-Tarski theorem there are. So, the problem is how does one uniquely characterize the object or perhaps the class of objects which satisfy the set of properties put forth in $C$? Can we get a characterization which is unique up-to-isomorphism? The algebraic approach used the "initiality" property to tie the models down.

In a first order framework there are some difficulties, besides the Loewenheim-Skolem-Tarski theorem. That is, even if we restrict ourselves to countable models of theories, there can be non-isomorphic models of the same theory. This is usually expressed by saying that first order theories are not in general $\omega$-categorical, where $\omega$ is the cardinality of the set of natural numbers.

There is a weak notion of characterization using the notion of elementary equivalence which should be noted. Recall from Theorem 2.2 that if the theory $T$ presented by $P$ is a complete theory, then any two models of $T$ are elementary equivalent. Thus, if we have a complete theory, we get a characterization. But, by Theorem 2.1 this is weaker than "up-to-isomorphism".

Another alternative, which appears useful at first, is to consider a data type to be a structure $A$, and then give an axiomatization for $\text{Th}(A)$. That surely gives us enough to prove everything about $A$. Furthermore, such theory is clearly consistent and complete. However, if we consider the structure $\mathbb{N}$ of number theory by a well-known result of Tarski [Tars(36)] $\text{Th}(\mathbb{N})$ is not a recursive set. One may consider a recursively enumerable subset of the theory of structure in order to
be able to axiomatize it. But that is not in general quite adequate, since the sub-theory may not be complete and thus not even give the weak characterization of up-to-elementary-equivalence.

There are some more advanced model-theoretic notions which shed further light on the question of characterization. In fact, there are special cases in which one can get characterizations which to some extent escape the consequences of the Skolem-Löwenheim-Tarski theorem. If it is the case that the presentation has a prime model [see CK (73)] then one gets a nice characterization by considering the prime model to be the abstract data type specified. However, this author is not aware of any general methods of constructing prime models.

4. USE OF SPECIFIC MODELS AS ABSTRACT DATA TYPES

We have seen that unique characterizations are difficult to come by in first order logic. Nevertheless, one may wish to present a set of first order properties and be willing to consider any object which embodies the properties. For example, one may well consider the standard structure of arithmetic or its isomorphism class as the data type specified by the Peano axioms (see Section 7).

It has been standard practice in computer science to state first order axioms formalizing the properties that a program written in a language must satisfy, solely for the purpose of being able to prove things about the program (see Mw (77) for a survey), all along assuming that the axioms describe a model, although perhaps not uniquely, as long as the axioms describing the properties are consistent. This is often when one already has a model in mind, but wishes to have a domain of discourse to reason about the model and use logical deductions to conclude other properties which are true in the given model. How does one pick models otherwise?

Given a consistent theory T there is a way of building a model for it through syntactic material. Since in data type specifications we are seeking an approach which does not commit us to a concrete representation and since it seems that if a first order theory approach is adopted, one may have to accept a specific model as the data type, a model which is built from the syntax of the theory directly appears to be the right compromise. We will briefly outline a method to construct a model for a consistent theory. The model is usually called a canonical structure. More details may be found in CK (73) or Shoe (67), for the one-sorted languages.

Let $t_1$ and $t_2$ be variable-free terms of a given theory $T$; define $t_1 \equiv t_2$ if $T \vdash t_1 = t_2$. It is easily shown that $\equiv$ is an equivalence relation, by the properties of equality. Now, let $[A]$ be the set of all equivalence classes of $\equiv$; let $[a]$ denote the equivalence class of $a$ and define

\[
\begin{align*}
    f_A([a_1], \ldots, [a_n]) &= [f_{a_1} \ldots a_n] \\
p_A([a_1], \ldots, [a_n]) &\text{ iff } T \vdash p(a_1 \ldots a_n)
\end{align*}
\]
for each $f^r_A$ of arity $(s_1 \ldots s_n)$, and $p^r_A$ of rank $(s_1 \ldots s_n)$, $s_1$ of sort $s_1$. Now, we can show by induction that each variable-free term $t$ is represented by $[t]$ in the structure $A$. Furthermore, an atomic formula $\phi$ is true in $A$ iff $T \vdash \phi$. To show that this last property holds for all closed formulas, we must require that the theory $T$ be a Henkin theory, i.e., that for every closed instantiation $3\forall \psi$ there is a constant $c$ such that $T \vdash 3\forall \psi \rightarrow (\chi/c)$. For otherwise there may be a theorem asserting that an individual has a certain property without there being such an individual in $|A|$. Also, there may be a closed formula such that neither $T \vdash \beta$ nor $T \vdash \neg \beta$. So, we must start with a complete theory $T$. Now, it is a well-known theorem that for a complete Henkin theory the canonical structure of $T$ is a model of $T$, and that every consistent theory has a consistent and complete Henkin extension. However, there is no algorithm for constructing such complete and consistent extension. This follows from the construction that goes to prove that a consistent theory has a complete and consistent extension [Lindenbaum's Lemma – see, for example, Mend (64)].

The conclusion of this lengthy discussion is that if the theory given by the data type presentation is complete, the natural choice for a model is the canonical structure. This is analogous to accepting a canonical term algebra as the data type in the terminology of the algebraic approaches. Here, we take the view that for a complete theory the canonical structure is the abstract data type. If the presentation does not define a complete theory, then one must choose a model based on the experience with the object whose properties are given by the presentation.

5. CONSTRUCTIVE FIRST ORDER EXTENSIONS

Having the discussion of the previous sections in mind, we wish to investigate the possibility of altering a class of data types characterized by a first order many sorted theory presentation to a new class with additional properties and sorts.

We are interested in the process of building new first order specifications from existing ones in a sensible manner. This is in the spirit of structured specification, and will be done by theory extension in the sense of mathematical logic. However, the usual extension methods in logic deal with one-sorted signatures and, therefore, are only useful for extending by new functions or predicates but not by new sorts.

The notion of theory extension we will use is based on extension by definition as outlined below. Given a theory $T_0$, to enrich it with a new function through extension by definition, the following steps must be taken.

(i) add the new function symbol, say $f$, to the language of $T_0$;
(ii) add a defining axiom for $f$ to the axioms of $T_0$. 
This step requires a defining formula $\phi$ of the language $T_0$. The defining axiom is of the form $y = f(x_1, \ldots, x_n) + \phi(y)$ where the defining formula $\phi$ has its free variables among $\{x_1, \ldots, x_n, y\}$. In order for this construction to be well-defined two conditions must be satisfied:

\begin{align*}
(\text{c1}) & \quad T_0 \vdash \exists y \phi \\
(\text{c2}) & \quad T_0 \vdash \phi \land \phi(y'/y) = y = y'
\end{align*}

In the above $(y'/y)$ is the formula obtained from $\phi$ by replacing all free occurrences of $y$ by $y'$. (c1) proves that there is an instance of the defining formula (c2) is a uniqueness condition. The theory thus obtained is called an extension by definition of $T_0$. New predicates can also be defined using definitional extensions. But, we are more interested in introducing new functions and particularly new types. The introduction of new types requires some additional machinery which we now set out to develop.

We are interested in defining a data type from a known type by adding some new sorts and operations. New predicates are just as easily added, but we leave out the details. There is no inherent difficulty with introducing the new sorts simultaneously, but the exposition may be slightly more complicated. Certainly the same can be achieved by introducing the new sorts one at a time. The extension process is described below.

Let $T_0$ be an $S_0$-sorted theory of signature $\Omega$. Let $\Sigma_N$ be the signature by which $\Omega$ is enriched by the extension process, $\Omega$ is assumed disjoint from $\Sigma_N$. Define $\Sigma$ to be an $S$-sorted signature, such that $S = S_0 \cup S_N$, where $S_N$ is the set of new sorts to be added, and $\Omega = \Omega \cup \Sigma_N$. Furthermore, let $\Phi \Sigma_N$ be a set of function symbols (and constants) of sorts in $S_N$ which we call the constructor signature.

Our basic philosophy is that in writing specifications one has a set of constructors in mind for a canonical many sorted structure. So, the choice of $\Phi$ should be evident if one knows what is being specified. This is clarified in the example worked out later. We have taken a similar approach with equational specifications in our (79).

There does not seem to be a natural way of defining constructors syntactically. Therefore, we are forced to make the definition relative to a model, as in the algebraic case. Let $T$ be a theory of signature $\Sigma$. Let $A$ be a model of $T$ which we have accepted as the data type specified by $T$. Then we say that a subset $\Phi$ of $\Sigma$ is a constructor signature for $T$ relative to $A$ if $\Phi$ can inductively generate the universe of $A$ and no proper subset of it can. For example, 0 and successor function are a simple example of a constructor signature for number theory relative to the standard model of arithmetic.

Our extension process begins by changing the signature from $\Omega$ to $\Omega \cup \Phi$ and the set of sorts for $S_0$ to $S$. Then a set of first order axioms is given describ-
ing the properties of functions in \( \Phi \) explicitly. \( \Phi \) must include at least one constant, \( c \). We now make precise the type of axioms that we require for \( \Phi \). Let \( T^*_0 \) be a theory of sort \( S \) and signature \( \Sigma_0 \) obtained by extending \( T_0 \) with axioms \( E \) to be defined below.

We only allow extensions by a finite set of constructors \( \Phi \). For ease of presentation we assume that the constructors of each given sort of nontrivial arity have equal arity. This restriction can be easily removed. The set of axioms \( E \) by which \( T_0 \) is explicitly extended is characterized as follows:

1. For each \( g, g' \) in \( \Phi \) of rank \( <s_1, \ldots, s_n, s> \) and \( <s_1, \ldots, s_m, s> \), respectively, such that \( g \) and \( g' \) are distinct, \( E \) include the axiom
   \[
   g(x_1, \ldots, x_n) \neq g'(y_1, \ldots, y_m)
   \]
   where \( x_i \) and \( y_i \) are variables of sort \( s_i \); special cases of such axioms have either or both of \( m \) and \( n \) equal to zero, stating that constants of each sort are distinct, and constants are distinct from terms involving constructors of nontrivial arity;

2. Let \( g_1, \ldots, g_n \) be constructors of sort \( s \) in \( \Phi \) of nontrivial arity. Let \( c_1, \ldots, c_m \) be constants of sort \( s \), then \( E \) includes the axiom
   \[
y \neq c_1 \land \ldots \land y \neq c_m \land \exists x_1 \ldots \exists x_k. y = g_1(x_1, \ldots, x_k) \lor \ldots \lor y = g_n(x_1, \ldots, x_k)
   \]
   where \( y \) is a variable of sort \( s \) and \( g_i \)'s have rank \( <s_1, \ldots, s_k, s> \);
   \( x_i \)'s are variables of sort \( s_i \);

3. For every \( g \in \Phi \) of nontrivial rank \( <s_1, \ldots, s_n, s> \) \( E \) includes an axiom
   \[
g(x_1, \ldots, x_n) = g(x'_1, \ldots, x'_n) \iff x_1 = x'_1 \land \ldots \land x_n = x'_n;
   \]
   \( x_i \) and \( x'_i \) are variables of sort \( s_i \), \( i = 1, \ldots, n \).

4. \( E \) includes an axiom only if it can be admitted by (i) - (iii).

Note: variables not quantified explicitly are assumed universally quantified.

An important example of a special case of the above is the sub-theory \( Q \) of arithmetic [see TMR (71)] with explicit axioms for the constructors \( 0 \) and successor of the standard model of arithmetic as its only axioms according to (i) - (iv) above.

Having given explicit axioms for the constructors of the new sort, other functions may be defined through an extension by definition or recursively defined enrichment of the theory which is already explicitly extended by new sorts and the above axioms for constructors in \( \Phi \).

To summarize, we start with an \( S_0 \)-sorted theory \( T_0 \) of signature \( \Sigma_0 \). Extend \( T_0 \) to a theory of signature \( \Sigma_0 \cup \Phi \), where \( \Phi \) is a subset of the signature by which \( \Sigma_0 \) gets ultimately extended and consists of at least one constant \( c \), and other functions. The axioms of \( T_0 \) are augmented by a set of axioms for the functions in \( \Phi \) according to the rules (i) - (iv). The resulting theory is an \( S \)-sorted theory of signature \( \Sigma_0 \cup \Phi \), with \( S = S_0 \cup \Sigma_0 \). Call this theory \( T^*_0 \). Now, other operations may be added through extension by definition or recursively defined enrich-
ments of $T^*$ to take the signature from $\Sigma_0 \cup \Phi$ to $\Sigma$, extending $T_0^*$ to $T$. We shall refer to $T$ as first order extension by constructors of $T_0$. $T_0^*$ will be referred to as a constructive extension of $T_0$.

As an example of a many-sorted constructive extension let us sketch the example of stack data type as constructive theory extension from a given parameter theory for a data type $d$, since this is a must for any paper on data types! Assume a first order specification of a parameter data type $d$ is given by $T_0 = (S_0, \Sigma_0, C_0)$, presenting a first order theory $T_0$. Now, enrich $T_0$ with $\Phi = \{\lambda, \text{PUSH}\}$. Add the following axioms to those of $T_0$:

1. $\text{PUSH}(d,s) = \text{PUSH}(d',s') \iff d = d' \land s = s'$  
2. $\lambda \neq \text{PUSH}(d,s)$  
3. $s \neq \lambda \iff \exists d \exists s'(s = \text{PUSH}(d,s'))$

In the above $d$ is a variable of sort $d$ and $s, s'$ are variables of sort stack. For convenience we have not attached a type to the existential quantifier, letting its type to be determined from the variable it bounds. The axioms explicitly define the properties of the constructors. (1) asserts that PUSH is an injective operation; (2) asserts that the empty stack ($\lambda$) is not constructable from the other constructor operation. (3) asserts that the complement of (2) holds, i.e., any other term of type stack can be generated from a sequence of PUSH operations.

Let $T_0^*$ be the theory thus obtained. Now, enrich the signature of $T_0^*$ with $\{\text{TOP}, \text{POP}\}$. The following are defining axioms for TOP and POP as we shall verify later.

1. $\text{TOP}(s) = d \iff \exists s((s = \text{PUSH}(d,s'))) \land s \neq \lambda$  
2. $\text{POP}(s) = s' \iff \exists d(s = \text{PUSH}(d,s'))) \land s \neq \lambda$  
3. $\text{TOP}(\lambda) = \perp$  
4. $\text{POP}(\lambda) = \lambda$

In the above, $d$ is a variable of type $d$, $s, s'$ are variables of type stack. $\perp$ is a special constant of type $d$.

Next, we show that $d_1 - d_4$ indeed give us an extension by definition of $T_0^*$. This is done by checking that the conditions (c1) and (c2) of each extension is satisfied. To show (c1) holds, i.e., the existence of a defining formula, it is sufficient to show that

1. $T_0^* \vdash \exists d \exists s'(s = \text{PUSH}(d,s'))$ when $s \neq \lambda$  
2. $T_0^* \vdash \exists s \exists d(s = \text{PUSH}(d,s'))$ when $s \neq \lambda$  
3. $T_0^* \vdash \exists s'(s = \text{PUSH}(d,s')) \land \exists s''(s = \text{PUSH}(d',s'')) \land s \neq \lambda \rightarrow d = d'$  
4. $T_0^* \vdash \exists d(s = \text{PUSH}(d,s')) \land \exists d(s = \text{PUSH}(d,s'')) \land s \neq \lambda \rightarrow s' = s''$
These are easily verified by checking the axioms a1 - a3 of constructors.

6. CONSISTENCY AND COMPLETENESS OF EXTENSIONS

In any reasonable definition of new types from existing types, one must ensure there are no inconsistencies introduced by the definition. In our formulation of specifications in this paper we have been using syntactic notions throughout, i.e., theories as opposed to models are used. Syntactic notions often lead to more complicated formulations, but they are required for deductive reasoning. In this section we define syntactic notions of completeness and consistency for first order presentations and then prove that first order extension by constructors is indeed consistent and complete in the sense to be defined.

Definition 6.1. Let $T_0$ be an $S_0$-sorted first order theory, and $T$ an $S$-sorted extension of $T_0$, $S_0 \subseteq S$. We say that $T$ is a relatively complete extension of $T_0$ if for every term of the form $f(t_1, \ldots, t_n)$, with $f$ of rank $<s_1, \ldots, s_n, s> \in S_0$, $T \vdash f(t_1, \ldots, t_n) = c$, where $c$ is a constant of sort $s \in S_0$ and $t_i$ are of sort $s_i \in S$. Say that $T$ is a relatively consistent extension of $T_0$ if the $c$ above is unique. □

Note that if $T$ is a relatively complete and consistent extension of $T_0$, then $T_0$ itself must be trivially relatively complete and consistent (with respect to itself). This notion is similar to sufficient completeness idea of Gutt (75) but here we have a first order many-sorted language.

The above definition clearly gives a desirable and reasonable requirement to be imposed on extensions. For one does not wish to introduce new terms into the old types which do not correspond to values of old types in any model. Nor to be able to assert new theorems about the old constants. Furthermore, the extension requires that the base theory have no redundant terms.

Relative completeness may not hold if one is not careful in defining extensions. For example, one must take account of the error elements that arise when operations of the extended signature are applied to a new constant and the resulting type is one of the old types. $TOP(\lambda)$ is one such instance in the stack example. One must ensure there is a constant of type $\delta$ to which $TOP(\lambda)$ can be reduced. In the example we gave, we defined $TOP(\lambda)=\bot$, where we assumed $\bot$ is a special constant of type $\delta$.

The main theorem concerning relatively complete and consistent extensions is given below. But, before stating that theorem, we present our notion of relative Henkin property.

Definition 6.2. Let $T$ be an $S$-sorted extension of an $S_0$-sorted theory $T_0$, and $\phi$ a formula in the language of $T$. We say that $T$ is Henkin relative to $T_0$ if for every instantiation $E\forall y \phi$ in language of $T$ such that $y$ is a variable of sort $s \in S_0$ free in $\phi$, $T \vdash E\forall y \phi \leftrightarrow \phi(c/y)$, for some constant $c$ of language of $T_0$.

Note that if $T$ is Henkin, it clearly is Henkin relative to $T_0$. Now, we are ready to state our theorem.
Theorem 6.3. Let \( T \) be an \( S \)-sorted first order extension by constructors of an \( S_0 \)-sorted theory \( T_0 \), with \( T \) a constructive extension of \( T_0 \). Let \( T_0 \) be relatively complete and consistent (with respect to itself). If \( T \) is Henkin relative to \( T_0 \), then \( T \) is a relatively complete and consistent extension of \( T_0 \).

Proof: Let \( f \) be a function symbol of rank \( <s_1,...,s_n,s> \), \( s_i \in S_0 \), \( s \in S \), where \( S_0 \) is the set of sorts of \( T_0 \), and \( S \) that of \( T \). By the extension assumption there is a formula \( \phi \) of language of \( T_0 \) (the extension of \( T_0 \) by constructors) such that \( T_0 \models \exists y \phi(x_1,...,x_n,y) \) and \( \exists y \phi(x_1,...,x_n,y) \cup \phi(x_1,...,x_n,y'/y) \models y=y' \). Now, for distinct variables \( x_1,...,x_n \) of sorts \( s_1,...,s_n \), respectively, \( s_i \in S_0 \) and \( y \) a variable of sort \( s \in S_0 \), let \( f(x_1,...,x_n)=y \Rightarrow \phi(x_1,...,x_n,y) \) be the defining axiom for \( f \) in the extended theory \( T \). From the conditions of extension we have
\[
T \models \exists y \phi(x_1,...,x_n,y).
\]
Since \( T \) is assumed Henkin relative to \( T_0 \) this implies
\[
T_0 \models \exists y \phi(x_1,...,x_n,y) \] for some constant \( c \) of language \( T_0 \). From the defining axiom for \( f \) we have that \( T \models f(x_1,...,x_n)=c \Leftrightarrow \phi(x_1,...,x_n,c/y) \). Since \( T \) is an extension of \( T_0 \) we may conclude \( T \models f(x_1,...,x_n)=c \). Thus \( T \) is a relatively complete extension of \( T_0 \). That \( T \) is a relatively consistent extension follows by observing that by condition (c2) of extension by definition \( T_0 \models \phi(x_1,...,x_n,y) \cup \phi(x_1,...,x_n,y'/y) \models y=y' \)

Therefore, the \( c \) above must be unique. This completes the proof. \( \Box \)

This is essentially a syntactic way of saying that the existing data type specifications are preserved under extensions.

We can also show a semantic preservation of the data type specified by the presentation of \( T_0 \) in a data type specified by the presentation of \( T \), the extension of \( T_0 \). This is done by a simple generalization of a result of model theory to many-sorted structures and languages.

Theorem 6.4. Let \( T_0 \) be an \( S_0 \)-sorted first order theory of signature \( \Sigma \). Let \( T \) be an \( S \)-sorted theory of Signature \( \Sigma \), such that \( T \) is a first order extension (by constructors) of \( T_0 \). Let \( A \) be a model of \( T \). Then the \( \overline{S} \)-reduct of \( A \) is a model of \( T_0 \).

Proof: We show that for every closed formula \( \phi \) of \( T_0 \), \( A \models \phi \Rightarrow \overline{S}_0 \models \phi \). This is done by induction. First note that by definition of reduct the \( S_0 \)-sorted carrier is the same in both \( A \) and \( A_0 \), so the same individuals represent the constants of \( A \) and \( A_0 \), as far as the sorts in \( S_0 \) are concerned. This gives us the basis of the induction. Next, for \( \phi \) an atomic formula \( p(a_1,...,a_n) \) of \( T_0 \), where \( p \) has the rank \( (s_1...s_n) \), \( s_i \in S_0 \), \( A \models p(a_1,...,a_n) \Rightarrow \overline{S}_0 \models p(a_1,...,a_n) \). If \( \phi=\exists \beta \) for \( \beta \) atomic, then
\[
A \models \exists \beta \Rightarrow \overline{S}_0 \models \exists \beta ( \text{since } A_0 \models \exists \beta ) \]
and, therefore, \( A_0 \) models \( \phi \). For \( \phi=\beta \lor \beta' \), \( A \models \phi \iff A \models \beta \lor \beta' \). By induction this implies \( A_0 \models \beta \lor \beta' \), i.e., \( A_0 \models \beta \lor \beta' \) (other connectives are proved similarly). If \( \phi=\exists x \beta \), since \( \exists x \beta_0 \), \( A \models \phi \) implies there is a constant \( c \) of sort \( s \in S_0 \), such that \( A \models \beta(c/x) \). Then by induction \( A_0 \models \beta(c/x) \Rightarrow A_0 \models \exists x \beta \). \( \Box \)

7. INDUCTION SCHEMATA FOR CONSTRUCTORS AND PRADO MODELS

To make practical use of the concepts and formulation presented in the pre-
vious sections in proving properties of specified data types we need to introduce induction schemas into the picture, much like Peano's axiomatization of number theory. We believe that in order to prove computationally interesting properties of first order specifications, it is sufficient to introduce induction schemas for constructors only. For example, the successor S is the only non-trivial constructor for first order number theory. Peano's axioms include the following induction schema for the constructor S: $(\phi(0) \land \forall x(\phi(x) \rightarrow \phi(Sx))) \rightarrow \forall x\phi(x)$, where $\phi$ is any formula of number theory in which $x$ is free. A corresponding axiom schema of induction for the constructor PUSH of the stack example is $(\phi(\lambda) \land \forall x(\phi(x) \rightarrow \phi(PUSH(d,x)))) \rightarrow \forall x\phi(x)$, where $d$ is a variable of the sort pushed onto the stack and $x$ is a variable of sort stack, and $\phi$ is a formula of the language given by the signature of stack, in which $x$ is free.

Although the general concept of induction schema for constructors should be intuitively clear there seems to be some notational difficulties in the formulation of the most general induction schema for constructors. Instead, we make some simplifying assumptions in order to present the main idea. Suppose we have chosen a many-sorted structure $A$ as the data type presented by a first order many-sorted theory of signature $\Sigma$. Assume that the carrier of each sort $s$ in $S$ has only one constructor of non-trivial arity and at least one constant of each sort. We require that there be at most finitely many such constants. Now, for each constructor of rank $<s_1, \ldots, s_n, s>$ of non-trivial arity, with $s=s_i$ for some $i$ in $\{1, \ldots, n\}$, we can define an induction schema of the following kind.

Let $c_1, \ldots, c_m$ be constants of sorts $s$, and let $\phi$ be any formula of the language given by $\Sigma$. Let $z$ be a variable of sort $s$ free in $\phi$, and $x_i$ a variable of sort $s_i$. Then an instance of an induction schema on base $\{c_1, \ldots, c_m\}$ for $f$ is given by $(\phi(c_1) \land \cdots \land \forall x_1 \cdots \forall x_{n-1} \forall z(\phi(z) \rightarrow \phi(f(z_1, \ldots, x_{n-1}, z)))) \rightarrow \forall z\phi(z)$.

We propose such schemas as important proof rules for specifications, be they first order or equational. We note that the general concept of structural induction has its roots in Curry and Feys (58). Burstall (67) devised similar notions for proving properties of programs. What is novel in the present work is that certain structural induction schemas, namely schemas for constructors, are singled out as being sufficient for proving the inductive properties of specifications (along with the axioms given by the presentation). We are not substantiating this claim with a formal argument here. We have given a formulation of this question and the related theorem for algebraic presentations in Nour (79b). Canonical term algebras simplify the problem there.

An interesting point is that the so-called Peano models [see Henk (60)] have exactly the properties one finds in initial algebras. That is uniqueness-up-to-isomorphism and existence of a unique homomorphism to similar models. Thus, if one proceeds by axiomatizing data types the way we have done in this paper then
one can consider a many-sorted version of Peano models as the data type specified
and get exactly the characterization present in the initial algebra approach.
Furthermore, the induction schemas for constructors will be sufficient to prove
inductive properties that hold in such Peano models (plus the axioms of the pre-
sentation). More details will be reported in a forthcoming report of this author.

8. EQUATIONAL EXTENSION BY DEFINITION

There is a special case of extension by definition in which the defining
axioms are restricted to be of the form \( f(x_1, \ldots, x_n) = t \), where \( t \) is a term with
free variables among \( x_1, \ldots, x_n \) only. This is an abbreviation for \( f(x_1, \ldots x_n) =
\gamma \iff \gamma = t \). In such cases the conditions for existence and uniqueness ((e1) and (e2)
of the previous section) are always trivially satisfied, for \( t \) instantiates \( \exists \gamma, \gamma \),
and \( \gamma = \gamma \iff \gamma = \gamma \). The step from \( T_0 \) to \( T \) has to be taken as before. It is the
final extension from \( T \) to \( T^* \) that we are dealing with here. So, the problem is
that of enriching \( T_0 \) using equational defining axioms. Such extensions and
enrichments were discussed in Nour (78) and (78a). More recently, Cartwright and
McCarthy [CM (79)] defined a "first order programming logic" the key idea behind
which is that "recursive definitions of partial functions can be interpreted as
equations extending a first order theory of program domain," thus the concepts
we have investigated may prove helpful in studies of semantics of programming
languages within first order logic.

9. IMPLEMENTATIONS OF FIRST ORDER SPECIFICATIONS

There are two notions in logic and model theory which might be used to form-
alize a concept similar to implementation. One is the notion of interpretation
between theories which is essentially syntactic. The other is definability of a
structure within another. This is a model-theoretic notion. If we have partic-
cular structures chosen to represent the abstract data type, then it might make
sense to use the latter notion. Otherwise, we must use the former notion (inter-
pretation) which is syntactic. We will give an intuitive exposition of interpre-
tation between theories as it relates to the question of implementation. For
definability of a structure within another we refer the reader to CK (73).

The notion of implementation is in essence a change of representation. So,
given first order presentations \( P = \langle S, E, C \rangle \) and \( P' = \langle S', E', C' \rangle \) we wish to define a
syntactic notion of implementation. Recall that we identify the notions of sig-
nature and language for first order many-sorted theories. Write \( T(P) \) for theory
presented by \( P \).

We must first define a syntactic translation of \( E \) into \( E' \). For the above
translation, a mapping \( h \) must be defined which given any \( E \)-sentence \( \beta \),
\( \beta \epsilon T(P) \iff h(\beta) \epsilon T(P') \). Furthermore, if \( \beta \epsilon T(P) \iff h(\beta) \epsilon T(P') \) holds the interpretation
mapping \( h \) is said to be faithful. For example, one can show that the theory of
the structure \( \langle N, 0, S \rangle \) where \( N \) is the set of natural numbers is interpretable into
the theory of the structure \( <\mathbb{Z},+,.> \), where \( \mathbb{Z} \) is the set of integers. Furthermore, one can show that there is a faithful interpretation of \( \text{Th}(<\mathbb{N},0,S>) \) into \( \text{Th}(<\mathbb{Z},+,.>) \).

A detailed analysis of such approaches and their practicality is beyond the scope of the present paper. However, it seems to us that a careful formulation of the notion of data representation in Hoar (72) must make use of such notions of implementation. For Hoare's notion of data representation relies on a syntactic translation from one representation into another and then checks to see if an assertion (a first order sentence) in one representation is provable as a theorem of the other representation, once the assertion is translated into the language of that representation.

To summarize, we propose that if \( T(P) \) and \( T(P') \) are first order theories presented by \( P \) and \( P' \) (the specifications), then an implementation of \( T(P) \) in \( T(P') \) is a faithful interpretation of \( T(P) \) in \( T(P') \).

Also, it turns out that theory extension plays an important role in interpretation between theories [see TMR (71)] confirming that extensions and implementations are intimately related.

**REFERENCES**


