1. Cleaning up the Mess of Recursive Data Structures

Typical "non-cyclic" recursive data structures include lists, trees, stacks, and so on. The seminal paper about them, both in terms of their formalization in an untyped setting and also in terms of decidability, is Derek Oppen's JACM, 27, 403-411, 1980 paper "Reasoning about Recursively Defined Data Structures".

The mess. The whole area is conceptually a mess.

The reason is the following. The function symbols of any such recursive data structure are split into:

1. **Free constructors** \( \mathcal{F} \), which build the data structure
   and obey no axioms, and

2. **Selector functions** \( \mathcal{S} \), which decompose a data structure
   into smaller components

Of course we want the selector functions to be defined
functions that disappear after evaluating each term

\( \Delta \) to its canonical form, so that the specification is

sufficiently complete, so that the so-called constructors

are really so.
The problem is that even for simple data structures such as lists, this is impossible in a may-sorted setting and, a fortiori, even more impossible in an unsorted setting such as the one that Oyappen uses, where he gives strange equations like

$$\text{head}(\text{nil}) = \text{nil}$$

that would not even type in a typed setting.

Let us see the impossibility for lists, but the same can be shown for trees or for stacks. The constructors are \text{nil} and \text{cons}" \text{Cons} : \text{El} \times \text{List} \rightarrow \text{List}. And the selectors are \text{head} : \text{List} \rightarrow \text{El} and \text{tail} : \text{List} \rightarrow \text{List}, called \text{CAR} and \text{CDR} for \text{LISP} lists.

The problem is that there is no satisfactory way of defining \text{head} (\text{nil}) as a constructor term of sort \text{El}.

The only possible way is to postulate the existence of a constant, say, \text{no-head} of sort \text{El}, belonging to \text{El}.

But no suppose we view lists as a parameterized data type \text{List} [X]. Then how is this constant going to be interpreted for each non-empty set \text{El}? And in what sense will it be a constructor?

The Inconsistent Way Out. Current approaches to satisfying the recursiveness of recursive data structures adopt the way out of declaring the value of selector expressions for problematic
cases such as head(mil) to be underspecified. Specifically, this means that the parameterized data type

\( (\Sigma \text{LIST}
, \text{LIST}(X)) \)

does not have a single expansion of, say, a set of elements \( A \) to \( \text{LIST}(A) \), but as many as elements \( m \), \( \text{head}_m \in A \), each provide a different model (a different expansion).

This is clearly a mess. Suppose \( A = \text{NJ} \). Now consider the QF-formulas:

\[ x + 1 = \text{head}(\text{mil}), \quad 0 = \text{head}(\text{mil}) \]

\[ 1 + 1 = \text{head}(\text{mil}), \quad x + 1 + 1 = \text{head}(\text{mil}) \]

\[ x + 1 \neq \text{head}(\text{mil}), \quad 0 = \text{head}(\text{mil}), \quad x + 1 \neq \text{head}(\text{mil}) \]

It turns out all are satisfiable if just any choice of \( \text{no-head}_m \in \text{NJ} \) is allowed. But of course if another tool has assumed that \( \text{no-head}_{\text{NJ}} = 0 \), then some formulas (2nd, 4th, and 6th) are satisfiable and others not.

The upset is that it is, any day's given what an SMT solver will give as an answer to questions about the satisfiability of QF formulas such as the above, particularly if the have not adopted the convention of leaving the value of \( \text{head}(\text{mil}) \) underspecified but have instead adopted some other convention.
Furthermore, lists are supposed to be a data type, and in any data type with its salt any expression should be either completely and uniquely defined, or be a type error, so there should be no ambiguity about validity of formulas. But validity of formulas becomes totally ambiguous in underspecified "data types."

**The Ignored Solution.** The entire problem was solved by Mereguer and Goguen a long time ago: J. Mereguer and J. A. Goguen, "Order-Sorted Algebraic Solves the Constructor-Selector, Multiple Representation, and Coercion Problem," Information and Computation, 103, 114-158 (1993). The trivial observation is that head and tail are partial functions that become total in an order-sorted setup in which we add a nilist, subset of non-empty lists. In general, if \( e : A_1 \rightarrow A_n \rightarrow B \) in a constructor, we can add a subset \( B_c < B \) of selector operators \( \text{sel-c}_i : B_c \rightarrow A_i \), \( 1 \leq i \leq n \), defined by equations

\[
\text{sel-c}_i (C(X_1 : A_1, \ldots, X_n : A_n)) = X_i : A_i
\]

This is of course not the only option. For example, we could define an alternative solution for lists of the form:
funct. \text{LIST}\{X::\text{TRIV}3\} is

\begin{align*}
\text{sorts} & \quad \text{List}\{X\} \quad \text{Elt}\{X\}. \\
\text{subsort} & \quad X \subseteq \text{Elt}\{X\}. \\
\text{op} & \quad \text{nil} : \rightarrow \text{List}\{X\} [\text{ctn}]. \\
\text{op} & \quad \text{head} : \text{List}\{X\} \rightarrow \text{Elt}\{X\}. \\
\text{op} & \quad \text{tail} : \text{List}\{X\} \rightarrow \text{List}\{X\}. \\
\text{var} & \quad E : X \subseteq \text{Elt}, \quad \text{var} \quad L : \text{List}\{X\}. \\
\text{eq} & \quad \text{head}(E \cdot L) = L. \\
\text{eq} & \quad \text{head}(\text{nil}) = \text{no-head}. \\
\text{eq} & \quad \text{tail}(E \cdot L) = L. \\
\text{eq} & \quad \text{tail}(\text{nil}) = \text{nil}. \\
\end{align*}

end-func

The key point however, is that in both cases subsorts allow a full definition of the desired data type, so that once we have fully specified such a data type there is no ambiguity whatsoever exists about validity of formulas in a theory of recursive data structures so defined. We will see that this also applies to parameterized ones because they have always a generic model, so any QF formulas either valid or not.
A precise notion of recursive data structure solving the problem

**Definition.** A recursive data structure is a possibly parameterized order-sorted equational theory of the form:

$$P \xrightarrow{\gamma} B$$

where:

1. $P$ consists of $n$ disjoint copies of the TRIV theory (the case $m=0$ yields unparameterized recursive data structures like NAT in constructors $0$ and $s$ and selector $s$). Let $E_1, \ldots, E_m$ be the sorts of $P$.

2. $\Sigma_B$ is a disjoint union $\Sigma_B = \Sigma_B \cup \Delta_B$ where each $c \in \Sigma$ is called a constructor, and each $s \in \Delta_B$ is called a selector.

3. The equations $E_B$ of $B$ are all of the form either:

   3.1 $\text{sel}(c x_1, \ldots, x_m) = x_i$ for some $1 \leq i \leq n$, $m$

   3.2 $\text{sel}(a) = b$, for $a, b$ constants in $\Sigma$.

4. The equations $E_B$ are confluent, terminating and sufficiently complete (in the parameterized sense explained in J. Meseguer, "Order-Sorted Parameterization and Induction," Springer LNCS 5700, 43-80, 2009).

5. Each sort $E_i$, $1 \leq i \leq m$, in $P$ is a minimal element in the sort poset $(S_B, \leq_B)$ of $B$, and for any $c: w \to sE_i$, $s \neq E_i$, $1 \leq i \leq m$. 
Here is a crucial observation:

**Theorem.** If \( P \rightarrow^* B \) is a recursive data structure specification, then \( B \) has the finite variant property.

**Proof.** All constructor operators \( C(x_1, \ldots, x_n) \) have themselves only variants. Given a selector set \( \Sigma \) only possible variants are:

\[
(\text{sel}(X), \text{id}), \quad \langle X : \{X \mapsto C(x_1, \ldots, x_n)\} \rangle, \quad (b, \{x \mapsto a\}).
\]

**Definition.** Given a recursive data structure specification \( P \rightarrow^* B \), say with possibly parameterized) theory \( P \rightarrow^* B \), any with \( P = \text{TRIV} \cup \{\text{Ed} \rightarrow \text{Ed} \} \cup \cdots \cup \text{TRIV} \cup \{\text{Ed} \rightarrow \text{Ed}^m\} \) its semantics in the parameterized data type \( (\Sigma_B, \Phi_f) \)

where \( \Phi_f = \{\Phi_f[A_1, \ldots, A_n] \mid A_i \in \text{Set}_{\neg \phi}, 1 \leq i \leq n\} \).

**Theorem.** For \( (\Sigma_B, \Phi_f) \) the semantics of a recursive data structure, \( \Phi_f[A_1, \ldots, A_n] \) is a generic model for \( Q \) if satisfiability for \( A_i \) such that \( |A_i| = x_0, \ 1 \leq i \leq n \).
Proof. Suppose $Y \in QF(E_\beta)$ is satisfiable in some $\mathcal{F}_\beta[B_1, B_\alpha]$ with assignment $a \in \delta(Y \rightarrow \mathcal{F}_\beta[B_1, B_\alpha])$ where $Y = \varphi(Y)$. Define $[C_1, C_\alpha]$ as follows:

$$C_i = \{ b \in B_i \mid b \leq a(i)! \in \varepsilon_\beta \land y \in Y \} \cup \{ b_\alpha \}$$

where $b_\alpha \in B_i$ is arbitrarily chosen to make sure $C_i \neq \emptyset$.

Note that then we have a retract right inverse to the inclusion $j : [C_1, C_\alpha] \hookrightarrow [B_1, B_\alpha]$, so that $j : r = 4[C_1, C_\alpha]$, with the $C_1, C_\alpha$ finite non-empty sets. Furthermore, we have a factorization

$$\xymatrix{T_{E_\beta}(Y) \ar@{->}[r]^a \ar@{-->}[dr] & \mathcal{F}_\beta[B_1, B_\alpha] \ar@{-->}[d]^{\mathcal{F}_\beta(j)} \ar@{->}[r] & \mathcal{F}_\beta[C_1, C_\alpha]}
$$

so that $\mathcal{F}_\beta[C_1, C_\alpha], a = Y$. But then, we can always include each $C_i$ into a set $A_i$ with $|A_i| = \aleph_0$ to get an inclusion $[C_1, C_\alpha] \hookrightarrow [A_1, A_\alpha]$ and a retract map $r'$ with $j' : r' = 4[C_1, C_\alpha]$, and since $\mathcal{F}_\beta$ is a functor this gives an a submodel inclusion $\mathcal{F}_\beta[C_1, C_\alpha] \hookrightarrow \mathcal{F}_\beta[A_1, A_\alpha]$. So that by lemma, we have $\mathcal{F}_\beta[A_1, A_\alpha], a = Y$, as desired. \(\square\)

Note that for any $[A'_1, A'_\alpha]$ with $|A'_i| = \aleph_0$, we have a bijection $h : [A'_1, A'_\alpha] \approx [A_1, A_\alpha]$ and therefore an isomorphism $\mathcal{F}_\beta[A'_1, A'_\alpha] \approx \mathcal{F}_\beta[A_1, A_\alpha]$.
Corollary. If $P \rightarrow B$ is a (possibly unparametered) recursive data structure specification, then satisfiability of QF $\Sigma_B$-formulas in the theory $(\Sigma_B, \Gamma_J)$ is decidable.

Proof. If $P = \emptyset$ the result follows trivially from $B$ being FVP and $\Sigma$ being free constructor so that $\Gamma_J$ is DS-compact. If $P$ has sorts $\text{Elts}_1, \ldots, \text{Elts}_m$, instantiate them to $m$-rename copies of the theory $\text{NAT}$ currying of the constructor signature $o : \rightarrow \text{NAT}$, $s_i : \text{NAT} \rightarrow \text{NAT}$, renamed as $O_i : \rightarrow \text{NAT}_i$, $S_i : \text{NAT} \rightarrow \text{NAT}_i$, $1 \leq i \leq m$. Recall from Van Valen that this just means replacing $P$ by the theory:

\[ \text{NAT}_1 \otimes \cdots \otimes \text{NAT}_m = \begin{array}{c}
O_1 \rightarrow \text{NAT}_1 \\
\ldots \\
O_m \rightarrow \text{NAT}_m
\end{array} \]

In this way we get the theory $B \left[ x_1 \mapsto \text{NAT}_1, \ldots, x_n \mapsto \text{NAT}_m \right]$, which is both FVP and DS-compact and has decidable satisfiability of QF formulas for its initial algebra. Now note that

\[ (\Sigma_B, \Gamma_J; B[x_1 \mapsto \text{NAT}_1, \ldots, x_n \mapsto \text{NAT}_m] | \Sigma_B) \]

is precisely of the form $\Gamma_J [\text{NAT}_1, \ldots, \text{NAT}_m]$. That is, the result follows by restricting the con-sat satisfiability of $\Gamma_J; B[x_1 \mapsto \text{NAT}_1]$ to the sub-language $\text{QF}(\Sigma_B)$. $\square$
Exercise. Consider the parametrized data type \( L[X] \) of lists defined in \( \text{VarSet} \), and the theory inclusion (view) view \( \text{Triv2Test} \) from \( \text{TRIV} \) to \( \text{TOSET} \)

sort \( \text{Eet} \) to \( \text{Eet} \).

Then from \( \text{LIST}[X::\text{TRIV}] \) we can obtain

from \( \text{ORD-LIST}[T::\text{TOSET}] \) is

protecting \( \text{LIST}[\text{Triv2Test}] \)

endfun

where we have replaced the \( \text{TRIV} \) parameter theory by \( \text{TOSET} \).

1. Prove that \( \text{LIST}[\text{NIL}\leq] \) the ordered lists with \( \text{NIL}\leq \) the standard order on the naturals is a generic model of \( \text{ORD-LIST}[T] \). How would we exploit this fact in proving correctness of a parametrized/polyinstantiable sorting module like \( \text{INSERT-SORT}[T::\text{TOSET}] \)?

2. Generalized this result to any recursive data structure

   a) with \( P \) a single copy of \( \text{TRIV} \)

   b) with \( P \) have \( n \geq 1 \) copies of \( \text{TRIV} \) and one or several of them replaced by \( \text{TOSET} \) as above.
3. A More General Notion of Optimally Intersectable Signatures

we can relax the conditions in the Addendum to Lecture 24 to arrive at the following more general notion:

Definition: The ordered signed signatures \( \Sigma_i \) and \( \Sigma_j \) are optimally intersectable, denoted as \([\Sigma_i \land \Sigma_j]_k\), if for some component in \((\Sigma_i, \leq_i)\) of a sort \( S_i \in \Sigma_i, 1 \leq i \leq 2\), either condition (1) or (2) holds, and condition (3) holds:

1. either \( [\Sigma_i]_i \cap [\Sigma_j]_j \neq \emptyset \)

2. \( [k;k]_k \) has a top sort, \([S_k]_k \subseteq [S_i]_i \forall k \in \{1,2,3\} \)

1.1 \( \forall k \in \{1,2\}, [S_k]_k \) has a top sort, \( [S_k]_k \subseteq [S_i]_i \forall k \in \{1,2,3\} \)

1.2 \( \leq k \subseteq [S_k]_k \supseteq [S_k]_k \supseteq [S_i]_i \)

1.3. (Downward closure). \( \forall s \in [S_i]_i \forall s' \in [S_k]_k, s' \leq k s \Rightarrow s \in [S_k]_k \)

1.4 Uniqueness \( [S_i]_i \cap S_j = [S_j]_j \cap S_i = [S_k]_k \)

2. either \( [S_i]_i \cap [S_j]_j = \{s_0\}, [S_i]_i \subseteq \{1,2,3\} \land \exists k \in \{1,2,3\} \) such that \( s_0 \) is top element of \( [S_k]_k \)
and furthermore:

\[(U) \quad \exists i \in I \quad S_j \cap S_i = \{s_0\}\]

and 3. If \( f \in \text{fun}(\Sigma_1) = \text{fun}(\Sigma_2) \) (resp. \( f \in \text{med}(\Sigma_1) \cap \text{med}(\Sigma_2) \))

Then \( \exists i, j \in \{1, 2, 3\} \) such that:

3.1. If \( (s_1, s_2, s_3) \in F_i(f) \) (resp. \( (s_1, s_2, s_3) \in P_i(f) \))

then:

\[F_i(f) = F_j(f) \cap \left( \prod_{i \in I} [s_i] \times \prod_{i \in I} [s_i] \right)\]

(resp. \( P_i(f) = P_j(f) \cap \left( \prod_{i \in I} [s_i] \times \prod_{i \in I} [s_i] \right) \))

\[\forall l \leq m, \quad [s_l]_i \subseteq [s_l]_j, \quad 1 \leq i \leq n, \quad \text{and} \quad [s_l]_i \subseteq [s_l]_j, \quad 1 \leq l \leq m.\]

These conditions can be illustrated with the following example:
Exercise, Prove:

1. If $\Sigma_1$ and $\Sigma_2$ are optimally intersectable and $\Sigma_0 = \Sigma_1 \cap \Sigma_2$, then

$$
\begin{align*}
\text{Mod}(\Sigma_1) & \xleftarrow{-1_{\Sigma_1}} \text{Mod}(\Sigma_0) \\
-1_{\Sigma_1} & \downarrow \downarrow -1_{\Sigma_0} \\
\text{Mod}(\Sigma_0) & \xleftarrow{-1_{\Sigma_0}} \text{Mod}(\Sigma_2)
\end{align*}
$$

is a pullback.

2. If $\Sigma_0$ consists only of ory (\(\text{fun}(\Sigma_1) \cap \text{fun}(\Sigma_2) = \emptyset\) and \(\text{prod}(\Sigma_1) \cap \text{prod}(\Sigma_2) = \emptyset\)), then

$$(\Sigma_1 \cup \Sigma_2, \text{fun})$$

is a correct ascent map.