Abstract Syntax, Abstractly

Interlude: We will now abstract from the particular languages (e.g., the toy calculator language of expressions, the language of regular expressions, ...) to a theory of *all* abstract languages and *all* possible definitional interpreters (*eval* functions) on them. The starting point is to first specify the symbols in an arbitrary language, and then specify the constraining rules about how these symbols are to be used. The next step will be to provide meaning to the symbols, and to show how to combine meanings of the fragments of syntax in giving meaning to all abstract syntax trees built over a set of symbols.

[Note: For simplicity, we will consider the case where all expressions are of one single "sort". The corresponding semantics which we develop will be on so-called single-sorted or homogeneous algebras. A straight-forward generalisation on the discipline of how to use symbols in building abstract syntax trees is called "typing", and the corresponding semantics are developed on heterogeneous or multi-sorted algebras.]

Signatures and Signed/Ranked Algebras

Definition: A *Signature* Σ is a (non-empty) set of symbols, and
for *each* symbol, its "arity", which is a natural number ≥ 0

Notation: Given a signature Σ, let $\Sigma^{(k)}$ denote its subset of symbols with arity k.

Example: For boolean expressions, the signature consists of the symbols:

`T`, `F` with arity 0
$\neg$ with arity 1
$\wedge$, $\vee$ with arity 2.

Example: For integral numeric expressions the signature contains:
the (denumerable set of) numerals with arity 0
the symbols `+` , `*` with arity 2
and the symbol `-` (unary minus) with arity 1.

Definition: The set $Tree_\Sigma$ of abstract syntax trees

Suppose Σ is a given signature.
The set $Tree_\Sigma$ is inductively defined as follows:
Base cases: for each 0-ary symbol c in Σ, a labelled node $\bullet c$ is in $Tree_\Sigma$
Induction cases: for each $k > 0$,
for each k-ary symbol f in Σ,
for any trees $t_1, \ldots, t_k \in Tree_\Sigma$,
the tree consisting of
a labelled node $\bullet f$ at the root
with k sub-trees $t_1, \ldots, t_k$ below the root node

Abstract Syntax, Abstractly

Interlude: We will now abstract from the particular languages (e.g., the toy calculator language of expressions, the language of regular expressions, ...) to a theory of *all* abstract languages and *all* possible definitional interpreters (*eval* functions) on them. The starting point is to first specify the symbols in an arbitrary language, and then specify the constraining rules about how these symbols are to be used. The next step will be to provide meaning to the symbols, and to show how to combine meanings of the fragments of syntax in giving meaning to all abstract syntax trees built over a set of symbols.

[Note: For simplicity, we will consider the case where all expressions are of one single "sort". The corresponding semantics which we develop will be on so-called single-sorted or homogeneous algebras. A straight-forward generalisation on the discipline of how to use symbols in building abstract syntax trees is called "typing", and the corresponding semantics are developed on heterogeneous or multi-sorted algebras.]

Signatures and Signed/Ranked Algebras

Definition: A *Signature* Σ is a (non-empty) set of symbols, and for *each* symbol, its "arity", which is a natural number ≥ 0

Notation: Given a signature Σ, let $\Sigma^{(k)}$ denote its subset of symbols with arity k.

Example: For boolean expressions, the signature consists of the symbols:

`T, F`	with arity 0
$\neg$	with arity 1
$\wedge, \vee$	with arity 2.

Example: For integral numeric expressions the signature contains:
the (denumerable set of) numerals with arity 0
the symbols `+` , `*` with arity 2
and the symbol `-` (unary minus) with arity 1. $a - b = a + (-b)$

Definition: The set $Tree_\Sigma$ of abstract syntax trees

Suppose Σ is a given signature.
The set $Tree_\Sigma$ is inductively defined as follows:
Base cases: for each 0-ary symbol c in Σ, a labelled node $\bullet c$ is in $Tree_\Sigma$
Induction cases: for each $k > 0$,
for each k-ary symbol f in Σ,
for any trees $t_1, \ldots, t_k \in Tree_\Sigma$,
the tree consisting of
a labelled node $\bullet f$ at the root
with k sub-trees $t_1, \ldots, t_k$ below the root node

is in $Tree_\Sigma$.

Note: The roots of $t_1, \ldots, t_k$ are the children ordered 1...k of the new root node $\bullet f$ These t_i need not necessarily be distinct.

Well-formed trees are those with nodes labelled by symbols in Σ, which respect the arities of the symbols, i.e., where every node labelled with a k-ary symbol will have k children.

Note: If the signature Σ has no 0-ary symbols, then the set $Tree_\Sigma$ is empty.
If the signature Σ has no k-ary symbols for any $k > 0$ (i.e., all symbols have arity 0), then the set $Tree_\Sigma$ contains only leaf nodes, and its cardinality is $|\Sigma|$. If $\Sigma^{(0)}$ is finite, then so is $Tree_\Sigma$, and if $\Sigma^{(0)}$ is denumerable, so is $Tree_\Sigma$.
If Σ is denumerable, and contains at least one symbol of arity 0 and at least one symbol of arity $k > 0$, then $Tree_\Sigma$ is countably infinite.

Giving meaning to the symbols of Σ

Definition: A Σ-algebra $\mathscr{A} = \langle A, \ldots \rangle$ consists of
a set A — called the carrier set,
and an interpretation of each symbol in Σ:
for each 0-ary symbol in $\Sigma^{(0)}$, associate some element of A
for each k-ary symbol in Σ, associate a *total function* in $[A^k \to A]$ (that is, from $A \times \ldots \times A$ to A)

Notes:
- *Not all* elements of A need have a 0-ary symbol in Σ associated with them. That is, the carrier set may contain values for which there is no corresponding abstract syntax. A good example is the set of irrational real numbers from which there will be innumerable elements without a corresponding syntactic description.
- Different 0-ary symbols in Σ do not necessarily have to be associated with different elements — two *different* symbols can be mapped to the *same* value. Similarly for k-ary symbols, two different symbols can be mapped to the same total function.
- Two Σ-algebras can have the same carrier set A, but may differ on the interpretation of the symbols in the signature Σ. These would be considered *different* Σ-algebras.

Observe that so long as we respect the arities when associating meaning to each symbol, we are free to pick *any* meaning for these symbols. If we pick meanings that conform to the usual interpretations of the symbols, we call the Σ-algebra *standard;* otherwise we call it *nonstandard*.

Exercise: Pick some sensible signature, e.g., integer expressions over numerals, + and *, or boolean expressions, as given above, or any other signature, say the symbols for core regular expressions.

For each signature, give standard examples of ∑-algebras.

Exercise: Give for each signature, *nonstandard* examples of ∑-algebras

<u>Notation:</u>
If $\mathscr{A} = \langle A, \ldots \rangle$ is a ∑-algebra, and $a \in A$ is associated with 0-ary symbol c in ∑, we write $c_{\mathscr{A}}$ to denote element a.
Similarly, if f is a k-ary symbol in ∑, and in ∑-algebra $\mathscr{A}$, f is associated with a total function g: $[A^k \to A]$, we write $f_{\mathscr{A}}$ to denote the function g.

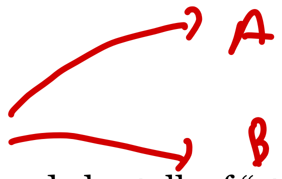
Abstract syntax = Trees treated as a ∑-algebra

Tree$_\Sigma$ is the ∑-algebra with carrier set $Tree_\Sigma$ and
• every 0-ary symbol c in ∑ is interpreted as the labelled node •c, which is in $Tree_\Sigma$, and
• every k-ary symbol f in ∑ is interpreted as the *tree-forming function* that takes k trees $t_1, \ldots, t_k$ in $Tree_\Sigma$ and creates the tree

```
    •f
   /  \
  t1 ... tk
```

in $Tree_\Sigma$ by sticking the trees $t_1, \ldots, t_k$ below a *new* root node labelled f.

Note that every k-ary symbol f in ∑ (for $k \geq 0$) is interpreted as either an element in $Tree_\Sigma$ or as a total function in $(Tree_\Sigma)^k \to Tree_\Sigma$.



∑ -Homomorphisms

We no discuss assigning meaning to abstract syntax trees, and also talk of "structure -preserving" functions that map the meaning of (every) tree in one ∑-algebra to the meaning of that same tree in another ∑-algebra.

Given signature ∑, the class of ∑-homomorphisms are the "structure-preserving" functions between ∑-algebras, such that the association of meanings to symbols in ∑ (according to the algebras) is preserved.

Definition: Suppose ∑ is a given signature.
Suppose $\mathscr{A} = \langle A, \ldots \rangle$ and $\mathscr{B} = \langle B, \ldots \rangle$ are both ∑-algebras. That is, each consists of a carrier set and interpretations of each symbol in ∑ which respect the arities of the symbols.
A <u>total function</u> h: $A \to B$ between the carrier sets of these algebras is called a <u>∑-*homomorphism*</u> if
- *for all* 0-ary symbols c in ∑: $h(c_{\mathscr{A}}) = c_{\mathscr{B}}$
- *for all* k-ary symbols ($k > 0$) f in ∑,
  for all $a_1, \ldots, a_k$ in A,

$$h(f_{\mathscr{A}}(a_1, \ldots, a_k)) = f_{\mathscr{B}}(h(a_1), \ldots, h(a_k))$$

Note that a $\sum$-homomorphism is a recursive function that maps every constructed element in the first algebra's carrier set obtained by applying $f_{\mathscr{A}}$ to an arbitrary tuple $a_1, \ldots, a_k$ of values in the first algebra's carrier set to the corresponding construction in the second algebra obtained by applying $f_{\mathscr{B}}$ to the tuple of the respective images in the second algebra's carrier set under h of the elements $a_1, \ldots, a_k$

Note that there is no reason that a $\sum$-homomorphism *should* exist between any two given $\sum$-algebras $\mathscr{A} = \langle A, \ldots \rangle$ and $\mathscr{B} = \langle B, \ldots \rangle$
Equally it may be possible for there to be *multiple* $\sum$-homomorphisms between two $\sum$-algebras $\mathscr{A} = \langle A, \ldots \rangle$ and $\mathscr{B} = \langle B, \ldots \rangle$

Exercise: Let $\mathscr{A} = \langle A, \ldots \rangle$ be any $\sum$-algebra. Show that the identity function $id_A : A \to A$ is a $\sum$-homomorphism.

Exercise: Suppose $h, h'$are two different $\sum$-homomorphisms between two $\sum$-algebras $\mathscr{A} = \langle A, \ldots \rangle$ and $\mathscr{B} = \langle B, \ldots \rangle$. On what elements $a \in A$ must $h(a) = h'(a)$? On what elements $a \in A$ may $h(a) \neq h'(a)$?

Initiality Theorem

The significance of the following theorem is that once we fix the meanings of the symbols in a given signature $\sum$ by picking a particular $\sum$-algebra $\mathscr{B} = \langle B, \ldots \rangle$, the meaning of every abstract syntax tree t in $Tree_{\Sigma}$ is determined.

Theorem: Suppose $\sum$ is a given non-trivial signature (i.e., at least one symbol has arity 0). For any $\sum$-algebra, $\mathscr{B} = \langle B, \ldots \rangle$, there is a unique $\sum$-homomorphism

$i_{\mathscr{B}}: Tree_{\Sigma} \to B$

from the $\sum$-algebra $\mathbf{Tree}_{\Sigma}$ to the $\sum$-algebra $\mathscr{B}$.

Proof — Define $i_{\mathscr{B}}$ to be $\sum$-homomorphism by construction.
Let $i_{\mathscr{B}}: Tree_{\Sigma} \to B$ as follows.

- for each 0-ary symbol c in $\sum$: $i_{\mathscr{B}}(\bullet c) = c_{\mathscr{B}}$
- for each k-ary symbol (k >0) f in $\sum$,

$$i_{\mathscr{B}}\left(\begin{array}{c} \bullet f \\ / \quad \backslash \\ t_1 \ldots \quad t_k \end{array} \right) = f_{\mathscr{B}}(i_{\mathscr{B}}(t_1) ,\ldots, i_{\mathscr{B}}(t_k))$$

<u>Uniqueness</u> of $i_{\mathscr{B}}$
Suppose $j: Tree_{\Sigma} \to B$ is a $\sum$-homomorphism.

Proof by induction on ($\mathtt{ht}\ t$) that for *all* t in $Tree_\Sigma$, $i_{\mathcal{B}}(t) = j(t)$

<u>Base cases</u> ($\mathtt{ht}\ t$) = 0.

t is of the form $\bullet c$

for each 0-ary symbol c in $\sum$: $i_{\mathcal{B}}(\bullet c) = c_{\mathcal{B}}$ // defn of $i_{\mathcal{B}}$

$= j(\bullet c)$ // j is a $\sum$-homomorphism

// from $\mathbf{Tree}_\Sigma$ to $\mathcal{B}$

<u>Induction Hypothesis</u>:

Assume that for all t' in $Tree_\Sigma$ such that ($\mathtt{ht}\ t'$) $\leq n$,

$i_{\mathcal{B}}(t') = j(t')$

<u>Induction Step</u>: Consider any t s.t. ($\mathtt{ht}\ t$) = n+1

t must be of the following form for some k-ary symbol f in $\sum$ (k > 0)

```
    •f
   /  \
 t1 ... tk
```

with ($\mathtt{ht}\ t_i$) $\leq n$ $(1 \leq i \leq k)$

Now, by definition,

for each k-ary symbol f in $\sum$, (where k >0)

```
iB(  •f  )
    /  \
  t1 ... tk
```

$= f_{\mathcal{B}}(i_{\mathcal{B}}(t_1),\ldots, i_{\mathcal{B}}(t_k))$ // definition of $i_{\mathcal{B}}$

$= f_{\mathcal{B}}(j(t_1),\ldots,j(t_k))$. // by **IH** on *each* of $t_1 \ldots t_k$

```
j(  •f  )      // j is a ∑-homomorphism from Tree∑ to B
   /  \
 t1 ... tk
```

Introducing Variables

We now proceed to deal with abstract syntax trees which may contain variables. We assume a denumerable set $\mathcal{X}$ of variables ranged over by typical (meta)variables $x, x', x_i, y, y', y_i, z, z', z_i \ldots \in \mathcal{X}$. We assume that the sets $\sum$ and $\mathcal{X}$ are disjoint (no common symbol).

Definition: The set $Tree_\Sigma(\mathcal{X})$ of abstract syntax trees possibly with variables.
Suppose $\sum$ is a given signature.
The set $Tree_\Sigma(\mathcal{X})$ is inductively defined as follows:
<u>Base cases</u>:

- for each 0-ary symbol c in $\sum$, a labelled node $\bullet c$ is in $Tree_\Sigma(\mathcal{X})$
- for each $x \in \mathcal{X}$, a labelled node $\bullet x$ is in $Tree_\Sigma(\mathcal{X})$

Induction cases: for each $k > 0$,
for each k-ary symbol f in $\sum$,
for any trees $t_1, \ldots, t_k$ in $Tree_\Sigma(\mathcal{X})$,
the tree consisting of
a labelled node $\bullet f$ at the root
with k sub-trees $t_1, \ldots, t_k$ below the root node
is in $Tree_\Sigma(\mathcal{X})$.

Note that there is a new collection of base cases, those of the variables. Observe that every t in $Tree_\Sigma$ is also in $Tree_\Sigma(\mathcal{X})$. Note also that the variables can only appear at the leaves of trees in $Tree_\Sigma(\mathcal{X})$. Note also that the introduction of the new base cases causes a percolation to the induction cases as well.

Observe that $Tree_\Sigma(\mathcal{X})$ is a $\sum$-algebra with carrier set $Tree_\Sigma(\mathcal{X})$ — instead of $Tree_\Sigma$ — with

- every 0-ary symbol c in $\sum$ interpreted as the labelled node $\bullet c$, which is in $Tree_\Sigma(\mathcal{X})$,

and

- every k-ary symbol f in $\sum$ is interpreted as the *tree-forming function*

that takes k trees $t_1, \ldots, t_k$ in $Tree_\Sigma(\mathcal{X})$ and creates the tree

```
    •f
   /  \
  t1 ... tk
```

in $Tree_\Sigma(\mathcal{X})$ by sticking the trees $t_1, \ldots, t_k$ below a *new* root node labelled f.

Note that every k-ary symbol f in $\sum$ (for $k \geq 0$) is interpreted as either an element in $Tree_\Sigma(\mathcal{X})$ or as a total function in $(Tree_\Sigma(\mathcal{X}))^k \to Tree_\Sigma(\mathcal{X})$.

How does the Initiality Theorem change to deal with abstract syntax trees that may contain variables? First, we introduce the concept of an A-valuation, which gives values from a set A to variables. Then we introduce the notion of a function extension.

Definition: Let A be any set and $\mathcal{X}$ be the set of variables. A function $\rho \in [\mathcal{X} \to A]$ is called an A-*valuation*.

Definition: Suppose $X \subset X'$, and for some set A, let $f : X \to A$ and $f' : X' \to A$. We say that f' *extends* f (or f' *is an extension of* f) if for each $x \in X$: $f'(x) = f(x)$.

Unique Homomorphic Extension Theorem
Suppose $\sum$ is any given signature. Let $\mathcal{B} = \langle B, \ldots \rangle$ be any $\sum$-algebra.
Let $\rho \in [\mathcal{X} \to B]$ be *any* B-valuation. Then there exists a *unique* $\sum$-*homomorphism* $\hat{\rho} \in [Tree_\Sigma(\mathcal{X}) \to B]$ that is an *extension* of $\rho \in [\mathcal{X} \to B]$.

Proof — Define $\hat{\rho}$ to be $\sum$-homomorphism that extends ρ by construction.

Let $\hat{\rho}: Tree_{\Sigma}(\mathcal{X}) \to B$ as follows.

- for each 0-ary symbol c in Σ: $\hat{\rho}(\bullet c) = c_{\mathcal{B}}$ // Σ-homomorphism
- for each $x \in \mathcal{X}$: $\hat{\rho}(\bullet x) = \rho(x)$ // Extension of ρ
- for each k-ary symbol (k >0) f in Σ, // Σ-homomorphism

$$\hat{\rho}\left(\begin{array}{c} \bullet f \\ / \quad \backslash \\ t_1 \dots\ t_k \end{array}\right) = f_{\mathcal{B}}(\hat{\rho}(t_1), \dots, \hat{\rho}(t_k))$$

<u>Uniqueness</u> of $\hat{\rho}$ (relative to ρ)

Suppose $j: Tree_{\Sigma}(\mathcal{X}) \to B$ is a Σ-homomorphism that extends ρ

Proof by induction on (`ht` t) that for *all* t in $Tree_{\Sigma}(\mathcal{X})$, $\hat{\rho}(t) = j(t)$

<u>Base cases</u> (`ht` t) = 0.

<u>Subcase</u> t is of the form $\bullet c$

for each 0-ary symbol c in Σ: $\hat{\rho}(\bullet c) = c_{\mathcal{B}}$ // defn of $\hat{\rho}$

$= j(\bullet c)$ // j is a Σ-homomorphism

// from $\mathbf{Tree}_{\Sigma}(\mathcal{X})$ to $\mathcal{B}$

<u>Subcase</u> t is of the form $\bullet x$

for each $x \in \mathcal{X}$: $\hat{\rho}(\bullet x) = \rho(x)$ // defn of $\hat{\rho}$

$= j(\bullet x)$ // j extends ρ

<u>Induction Hypothesis</u>:

Assume that for all t' in $Tree_{\Sigma}(\mathcal{X})$ such that (`ht` t') ≤ n,

$\hat{\rho}(t') = j(t')$

<u>Induction Step</u>: Consider any t s.t. (`ht` t) = n+1

t must be of the following form for some k-ary symbol f in Σ (k > 0)

$$\begin{array}{c} \bullet f \\ / \quad \backslash \\ t_1 \dots\ t_k \end{array}$$

with (`ht` t_i) ≤ n $(1 \le i \le k)$

Now, by definition,

for each k-ary symbol f in Σ, (where k >0)

$$\hat{\rho}\left(\begin{array}{c} \bullet f \\ / \quad \backslash \\ t_1 \dots\ t_k \end{array}\right)$$

$= f_{\mathcal{B}}(\hat{\rho}(t_1), \dots, \hat{\rho}(t_k))$ // definition of $\hat{\rho}$

$= f_{\mathcal{B}}(j(t_1), \dots, j(t_k))$. // by **IH** on *each* of $t_1 \dots t_k$ in $Tree_{\Sigma}(\mathcal{X})$

$$j\left(\begin{array}{c} \bullet f \\ / \quad \backslash \\ t_1 \dots\ t_k \end{array}\right)$$ // j is a Σ-homomorphism from $\mathbf{Tree}_{\Sigma}(\mathcal{X})$ to $\mathcal{B}$

Relevance Lemma: Suppose $\sum$ is any given signature. Let $\mathscr{B} = \langle B, \dots \rangle$ be *any* $\sum$-algebra. Let $t \in \mathbf{Tree}_{\Sigma}(\mathscr{X})$ be *any* (abstract syntax) tree. Let $X = vars(t)$ be the set of variables that appear in t. Let $\rho_1, \rho_2 \in [\mathscr{X} \to B]$ be *any* two B-valuations which coincide on X, i.e., for all $x \in X$: $\rho_1(x) = \rho_2(x)$.
Then $\widehat{\rho_1}(t) = \widehat{\rho_2}(t)$.
Proof is by induction on the structure (`ht`) of t.
(Note that `ht(`t`)`, $\widehat{\rho_i}(t)$ and $vars(t)$ are all defined on the structure of t).

Exercise: Prove the Relevance Lemma.