(Co)Algebraic Specification with Base Sets, Recursive and Iterative Equations


Peter Padawitz
TU Dortmund

January 8, 2015

More details can be found in:

- *Algebraic Compiler Construction*
- *Fixpoints, Categories, and (Co)Algebraic Modeling*
- *From Modal Logic to (Co)Algebraic Reasoning* (with *Expander2*)
Abstract

We present some fundamentals of a uniform approach to specify, implement and reason about (co)algebraic models in a many-sorted setting that covers constant, polynomial and collection types. Three kinds of (infinite-)tree models (finite terms, coterms and continuous trees) yield concrete representations (and Haskell implementations) of initial resp. final models.

On the axiomatic side, a format for recursive equations, which define either constructors on a final model or destructors on an initial one, is introduced. We show how iterative equations, which define continuous trees, can be translated into recursive equations so that the unique solvability of the latter implies the unique solvability of the former.

As a prototypical example, recursive equations define the Brzozowski automaton whose states are regular expressions and which accepts regular languages. We show how this set of equations can be extended by equations representing a non-left-recursive grammar $G$ such that it defines an acceptor of the language of $G$. 
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>4</td>
</tr>
<tr>
<td>Semantics</td>
<td>12</td>
</tr>
<tr>
<td>Initial and final algebras</td>
<td>27</td>
</tr>
<tr>
<td>Recursive equations</td>
<td>48</td>
</tr>
<tr>
<td>Iterative equations</td>
<td>59</td>
</tr>
<tr>
<td>Context-free grammars with base sets</td>
<td>67</td>
</tr>
<tr>
<td>Constructing recursive from iterative equations</td>
<td>73</td>
</tr>
<tr>
<td>(Co-)Horn Logic</td>
<td>80</td>
</tr>
</tbody>
</table>
Let $S$ be a set of sorts.

An $S$-sorted set $A$ is a tuple $(A_s)_{s \in S}$ of sets.

We also write $A$ for the union of $A_s$ over all $s \in S$.

An $S$-sorted subset $B$ of $A$, written as $B \subseteq A$, is an $S$-sorted set with $B_s \subseteq A_s$ for all $s \in S$.

Given $S$-sorted sets $A_1, \ldots, A_n$, an $S$-sorted relation $r \subseteq A_1 \times \cdots \times A_n$ is an $S$-sorted set with $r_s \subseteq A_{1,s} \times \cdots \times A_{n,s}$ for all $s \in S$.

The $S$-sorted binary relation $\Delta_A = \{\Delta_{A,s} \mid s \in S\}$ is called the diagonal of $A^2$.

Given $S$-sorted sets $A$ and $B$, an $S$-sorted function $f : A \rightarrow B$ is an $S$-sorted set such that for all $s \in S$, $f_s$ is a function from $A_s$ to $B_s$.

$Set^S$ denotes the category of $S$-sorted sets and $S$-sorted functions.
Let $S$ and $BS$ be sets of sorts and base sets, respectively.

The set $\mathbb{T}(S, BS)$ of types over $S$ and $BS$

is inductively defined as follows:

- $S \subseteq \mathbb{T}(S, BS)$.  
  (sorts)
- $BS \subseteq \mathbb{T}(S, BS)$.  
  (base sets)
- For all $n > 0$, $e_1, \ldots, e_n \in \mathbb{T}(S, BS)$, $e_1 \times \cdots \times e_n \in \mathbb{T}(S, BS)$.  
  (product types)
  The nullary product is identified with the base set $1 = \{\epsilon\}$.
- For all $n > 0$, $e_1, \ldots, e_n \in \mathbb{T}(S, BS)$, $e_1 + \cdots + e_n \in \mathbb{T}(S, BS)$.  
  (sum types)
- For all $e \in \mathbb{T}(S, BS)$, $\text{word}(e)$, $\text{bag}(e)$, $\text{set}(e) \in \mathbb{T}(S, BS)$.  
  (collection types over $e$)
- For all $X \in BS$ and $e \in \mathbb{T}(S, BS)$, $e^X \in \mathbb{T}(S, BS)$.
  (power types over $e$)
- For all $e, e' \in \mathbb{T}(S, BS)$ with $e' \not\in BS$, $e^{e'} \in \mathbb{T}(S, BS)$.  
  (higher-order types over $e$)

A type is **first-order** if it does not contain higher-order types.

$\mathbb{T}_1(S, BS)$ denotes the set of first-order types over $S$ and $BS$. 

A type is **flat** if it is a sort, a base set or a collection or power type over a sort.

\[ \mathbb{FT}(S, BS) \] denotes the set of flat types over \( S \) and \( BS \).

**A signature** \( \Sigma = (S, BS, BF, F, P) \) consists of

- a finite set \( S \) of **sorts** (symbols for sets),
- a finite set \( BS \) of **base sets**, implicitly including \( 1 = \{\epsilon\} \) and \( 2 = \{0, 1\} \),
- a finite set \( BF \) of **base functions** \( f : X \to Y \) with \( X, Y \in BS \),
- a finite set \( F \) of **operations** (symbols for functions) \( f : e \to e' \) with \( e, e' \in \mathbb{T}(S, BS) \),
- a finite set \( P \) of **predicates** (symbols for relations) \( p : e \) where \( e \) is a finite product of sorts and base sets.

For all \( f : e \to e' \in F \), \( \text{dom}(f) = e \) resp. \( \text{ran}(f) = e' \) is the **domain** resp. **range** of \( f \).

For all \( p : e \in P \), \( \text{dom}(p) = e \) is the **domain** of \( p \).

Given signatures \( \Sigma \) and \( \Sigma' \), \( \Sigma \cup \Sigma' \) denotes the componentwise union of \( \Sigma \) and \( \Sigma' \).
$f \in F$ is a **constructor** if there are flat types $e_1, \ldots, e_n$ over $S$ and $BS$ such that $\text{dom}(f) = e_1 \times \cdots \times e_n$ and $\text{ran}(f) \in S$.

$f \in F$ is a **destructor** if there are non-power flat types $e_1, \ldots, e_n$ over $S$ and $BS$ and $X \in BS$ such that $\text{dom}(f) \in S$ and $\text{ran}(f) = (e_1 + \cdots + e_n)^X$.

$\Sigma$ is **constructive** resp. **destructive** if $F$ consists of constructors resp. destructors.

**Constructive signatures**

Let $X$ be a set of constants and $CS$ be a set of nonempty sets of constants.

**Nat** $\bowtie$ **natural numbers**

$$S = \{\text{nat}\}, \quad BS = \emptyset, \quad F = \{ \text{zero} : 1 \to \text{nat}, \text{succ} : \text{nat} \to \text{nat} \}.$$  

**List($X$)** $\bowtie$ **finite sequences of elements of** $X$

$$S = \{\text{list}\}, \quad BS = \{X\}, \quad F = \{ \text{nil} : 1 \to \text{list}, \text{cons} : X \times \text{list} \to \text{list} \}.$$
Syntax

\( \text{Reg}(CS) \) \( \cong \) regular expressions over \( CS \) and regular languages over \( X = \bigcup CS \)

\[
S = \{\text{reg}\}, \quad BS = \emptyset, \quad F = \{ \begin{align*}
\text{eps} : 1 &\to \text{reg}, \\
\text{mt} : 1 &\to \text{reg}, \\
\text{par} : \text{reg} \times \text{reg} &\to \text{reg}, \quad \text{(parallel composition)} \\
\text{seq} : \text{reg} \times \text{reg} &\to \text{reg}, \quad \text{(sequential composition)} \\
\text{iter} : \text{reg} &\to \text{reg} \} \cup \\
\{ \overline{C} : 1 &\to \text{reg} \mid C \in CS \} 
\]

The nullary constructor \( \overline{C} \) stands for a name of the set \( C \).

Destructive signatures

Let \( X \) and \( Y \) be sets of constants.

\( \text{coNat} \) \( \cong \) natural numbers with infinity

\[
S = \{\text{nat}\}, \quad BS = \emptyset, \quad F = \{\text{pred} : \text{nat} \to 1 + \text{nat}\}. 
\]
\( coList(X) \cong \text{finite or infinite sequences of elements of } X \) \((coList(1) \cong coNat)\)

\[
S = \{\text{list, pair}\}, \quad BS = \{X\}, \quad F = \{\text{split : list} \to 1 + \text{pair}, \\
\text{first : pair} \to X, \\
\text{rest : pair} \to \text{list}\}.
\]

\(DAut(X,Y) \cong \text{deterministic Moore automata with input from } X \text{ and output in } Y\)

\[
S = \{\text{state}\}, \quad BS = \{X,Y\}, \quad F = \{\delta : \text{state} \to \text{state}^X, \\
\beta : \text{state} \to Y\}.
\]

\(Acc(X) \cong DAut(X,2) \cong \text{deterministic acceptors of subsets of } X^*\)

\[
S = \{\text{reg}\}, \quad BS = \{X, 2\}, \quad F = \{\delta : \text{reg} \to \text{reg}^X, \\
\beta : \text{reg} \to 2\}.
\]

\(Stream(X) \cong DAut(1, X) \cong \text{streams over } X\)

\[
S = \{\text{list}\}, \quad BS = \{X\}, \quad F = \{\text{head : list} \to X, \\
\text{tail : list} \to \text{list}\}.
\]
Let $V$ be a $T(S, BS)$-sorted set of variables.

The $T(S, BS)$-sorted set $T_{\Sigma}(V)$ of $\Sigma$-terms over $V$ is inductively defined as follows:

- For all $e \in T(S, BS)$, $V_e \subseteq T_{\Sigma}(V)_e$.
- For all $X \in BS$, $X \subseteq T_{\Sigma}(V)_X$.
- For all $f : 1 \rightarrow e \in BF \cup F$, $f \in T_{\Sigma}(V)_e$.
- For all $n > 1$, $e_1, \ldots, e_n \in T(S, BS)$, $t \in T_{\Sigma}(V)_{e_1 \times \cdots \times e_n}$ and $1 \leq i \leq n$, $\pi_i t \in T_{\Sigma}(V)_{e_i}$.
- For all $n > 1$, $e_1, \ldots, e_n \in T(S, BS)$, $1 \leq i \leq n$ and $t \in T_{\Sigma}(V)_{e_i}$, $i t \in T_{\Sigma}(V)_{e_1 + \cdots + e_n}$.
- For all $n > 1$, $e_1, \ldots, e_n \in T(S, BS)$ and $t_i \in T_{\Sigma}(V)_{e_i}$, $1 \leq i \leq n$, $(t_1, \ldots, t_n) \in T_{\Sigma}(V)_{e_1 \times \cdots \times e_n}$.
- For all $f : e \rightarrow e' \in BF \cup F$ and $t \in T_{\Sigma}(V)_e$, $ft \in T_{\Sigma}(V)_{e'}$.
- For all $c \in \{$word, bag, set$\}$, $e \in T(S, BS)$ and $t \in T_{\Sigma}(V)^*_e$, $c(t) \in T_{\Sigma}(V)_{c(e)}$.
- For all $n > 0$, $e_i, e \in T(S, BS)$, $x_i \in V_{e_i}$ and $t_i \in T_{\Sigma}(V)_e$, $1 \leq i \leq n$, $\lambda x_1.t_1 | \cdots | x_n.t_n \in T_{\Sigma}(V)_{e_1 + \cdots + e_n}$.
- For all $e, e' \in T(S, BS)$, $t \in T_{\Sigma}(V)_{e e'}$ and $u \in T_{\Sigma}(V)_{e'}$, $t(u) \in T_{\Sigma}(V)_e$. 
• For all \( e \in \mathbb{T}(S, BS) \), \( t \in T_\Sigma(V)_e \) and \( u, v \in T_\Sigma(V)_e \), \( \text{ite}(t, u, v) \in T_\Sigma(V)_e \).

A \( \Sigma \)-term \( t \) that does not contain variables or \( \text{ite} \), then \( t \) is called **ground**.

\( T_\Sigma \) denotes the set of ground \( \Sigma \)-terms.

---

**The set \( F_\Sigma(V) \) of \( \Sigma \)-formulas over \( V \)**

is inductively defined as follows:

• \( \text{True}, \text{False} \in F_\Sigma(V) \).

• For all \( p : e \in P \) and \( t \in T_\Sigma(V)_e \), \( pt \in F_\Sigma(V) \). \hfill (\( \Sigma \)-atoms over \( V \))

• For all \( e \in \mathbb{T}(S, BS) \) and \( t, u \in T_\Sigma(V)_e \), \( t =_e u \in F_\Sigma(V) \). \hfill (\( \Sigma \)-equations over \( V \))

• For all \( \varphi \in F_\Sigma(V) \), \( \neg \varphi \in F_\Sigma(V) \).

• For all \( \varphi, \psi \in F_\Sigma(V) \), \( \varphi \land \psi, \varphi \lor \psi, \varphi \Rightarrow \psi, \varphi \Leftarrow \psi, \varphi \Leftrightarrow \psi \in F_\Sigma(V) \).

• For all \( x \in V \) and \( \varphi \in F_\Sigma(V) \), \( \forall x \varphi, \exists x \varphi \in F_\Sigma(V) \).
[0] =_{\text{def}} \emptyset \text{ and for all } n > 0, [n] =_{\text{def}} \{1, \ldots, n\}.

For all \( f : A \to B, f^* : A^* \to B^* \) is defined as follows:
\[ f^*(\varepsilon) = \varepsilon \text{ and for all } n > 0 \text{ and } (a_1, \ldots, a_n) \in A^n, f^*(a_1, \ldots, a_n) = (f(a_1), \ldots, f(a_n)). \]

Let \( A, B \) be sets and \( a = (a_1, \ldots, a_m), b = (b_1, \ldots, b_n) \in A^* \).

\[
\begin{align*}
    a &=_{\text{word}} b \iff_{\text{def}} a = b. \\
    a &=_{\text{bag}} b \iff_{\text{def}} \exists f : [n] \to [n] : (a_1, \ldots, a_n) = (b_{f(1)}, \ldots, b_{f(n)}), \\
    &\quad \text{i.e., } b \text{ is a permutation of } a. \\
    a &=_{\text{set}} b \iff_{\text{def}} \{a_1, \ldots, a_m\} = \{b_1, \ldots, b_n\}.
\end{align*}
\]

Let \( h : A \to B \).

\[
\begin{align*}
    B_{\text{fin}}(A) &=_{\text{def}} A/_{=_{\text{bag}}} \text{ and } B_{\text{fin}}(h) : B_{\text{fin}}(A) \to B_{\text{fin}}(B) \text{ maps } [a]_{=_{\text{bag}}} \text{ to } [h^*(a)]_{=_{\text{bag}}}. \\
    P_{\text{fin}}(A) &= \{C \subseteq A \mid |A| < \omega\} \text{ and } P_{\text{fin}}(h) : P_{\text{fin}}(A) \to P_{\text{fin}}(B) \text{ maps } C \text{ to } \{f(a) \mid a \in C\}.
\end{align*}
\]
Predicate lifting

For all $e \in T_1(S, BS)$, the functor $F_e : Set^S \to Set$ is inductively defined as follows:

For all $S$-sorted sets $A, B$, $S$-sorted functions $h : A \to B$, $s \in S$, $X \in BS$, $n > 1$ and $e, e_1, \ldots, e_n \in T_1(S, BS)$,

- $F_s(A) = A_s$,
- $F_X(A) = X$,
- $F_{e_1 + \ldots + e_n}(A) = F_{e_1}(A) + \cdots + F_{e_n}(A)$,
- $F_{e_1 \times \ldots \times e_n}(A) = F_{e_1}(A) \times \ldots \times F_{e_n}(A)$,
- $F_{\text{word}(e)}(A) = F_e(A)^*$,
- $F_{\text{bag}(e)}(A) = B_{\text{fin}}(F_e(A))$,
- $F_{\text{set}(e)}(A) = P_{\text{fin}}(F_e(A))$,
- $F_{e^X}(A) = F_e(A)^X$,
- $F_s(h) = h_s$, (projection functor)
- $F_X(h) = id_X$, (constant functor)
- $F_{e_1 + \ldots + e_n}(h) = F_{e_1}(h) + \cdots + F_{e_n}(h)$,
- $F_{e_1 \times \ldots \times e_n}(h) = F_{e_1}(h) \times \ldots \times F_{e_n}(h)$,
- $F_{\text{word}(e)}(h) = F_e(h)^*$,
- $F_{\text{bag}(e)}(h) = B_{\text{fin}}(F_e(h))$,
- $F_{\text{set}(e)}(h) = P_{\text{fin}}(F_e(h))$,
- $F_{e^X}(h) = F_e(h)^X$.

We mostly write $A_e$ instead of $F_e(A)$.
Relation lifting

Given an $S$-sorted relation $R \subseteq A \times B$, $R$ is extended to a $T_1(S, BS)$-sorted relation inductively as follows:

Let $s \in S$, $e_1, \ldots, e_n, e \in T_1(S, BS)$ and $X \in BS$.

\begin{align*}
R_X &= \Delta_X, \\
R_{e_1+\ldots+e_n} &= \{(((a_i), (b_i)) \in (\bigsqcup_{i=1}^n A_{e_i}) \times \bigsqcup_{i=1}^n B_{e_i} \mid (a, b) \in R_{e_i}, 1 \leq i \leq n\}, \\
R_{e_1 \times \ldots \times e_n} &= \{(((a_1, \ldots, a_n), (b_1, \ldots, b_n)) \in (\prod_{i=1}^n A_{e_i}) \times \prod_{i=1}^n B_{e_i} \\
&\quad \mid \forall 1 \leq i \leq n : (a_i, b_i) \in R_{e_i}\}, \\
R_{\text{word}(e)} &= \bigcup_{n \in \mathbb{N}} \{(((a_1, \ldots, a_n), (b_1, \ldots, b_n)) \in A_e^* \times B_e^* \\
&\quad \mid \forall 1 \leq i \leq n : (a_i, b_i) \in R_{e}\}, \\
R_{\text{bag}(e)} &= \bigcup_{n \in \mathbb{N}} \{[[[a_1, \ldots, a_n]_\text{bag}, [[[b_1, \ldots, b_n]_\text{bag}}) \in \mathcal{B}_{\text{fin}}(A_e) \times \mathcal{B}_{\text{fin}}(B_e) \\
&\quad \mid \forall 1 \leq i \leq n : (a_i, b_i) \in R_{e}\}, \\
R_{\text{set}(e)} &= \{(C, D) \in \mathcal{P}_{\text{fin}}(A_e) \times \mathcal{P}_{\text{fin}}(B_e) \mid \forall c \in C \exists d \in D : (c, d) \in R_{e}, \\
&\quad \forall d \in D \exists c \in C : (c, d) \in R_{e}\}, \\
R_{e \times X} &= \{(f, g) \mid \forall x \in X : (f(x), g(x)) \in R_{e}\}.
\end{align*}
Let $\Sigma = (S, BS, BF, F, P)$ be a signature.

**A $\Sigma$-algebra $A$**

consists of

- an $S$-sorted set, called the **carrier** of $A$ and often also denoted by $A$,
- for each $f : e \to e' \in F$, a function $f^A : A_e \to A_{e'}$,
- for each $p : e \in P$, a subset $p^A$ of $A_e$.

Suppose that all function and relation symbols of $\Sigma$ have first-order domains and ranges. Let $A, B$ be $\Sigma$-algebras.

An $S$-sorted function $h : A \to B$ is a **$\Sigma$-homomorphism** if for all $f : e \to e' \in F$, $h_{e'} \circ f^A = f^B \circ h_e$, and for all $p : e \in P$, $h_e(p^A) \subseteq p^B$.

$Alg_\Sigma$ denotes the category of $\Sigma$-algebras and $\Sigma$-homomorphisms.

⇒ A $\Sigma$-homomorphism $h$ is iso in $Alg_\Sigma$ iff $h$ is bijective and for all $p : e \in P$, $p^B \subseteq h_e(p^A)$. 
Let $U_S$ be the forgetful functor from $\text{Alg}_\Sigma$ to $\text{Set}^S$.

For all $f : e \to e' \in F$, $\overline{f} : F_e U_S \to F_{e'} U_S$ with $\overline{f}(A) = \text{def } f^A$ for all $A \in \text{Alg}_\Sigma$ is a natural transformation:

\[
\begin{array}{c}
A_e \xrightarrow{f^A} A_{e'} \\
\downarrow h_e & \quad \downarrow h_{e'} \\
B_e \xrightarrow{f^B} B_{e'}
\end{array}
\]

Given a category $\mathcal{K}$ and an endofunctor $F$ on $\mathcal{K}$,

- an $F$-algebra or $F$-dynamics is a $\mathcal{K}$-morphism $\alpha : F(A) \to A$,
- an $F$-coalgebra or $F$-codynamics is a $\mathcal{K}$-morphism $\alpha : A \to F(A)$.

$\text{Alg}_F$ and $\text{coAlg}_F$ denote the categories of $F$-algebras resp. $F$-coalgebras where

- an $\text{Alg}_F$-morphism from $\alpha : F(A) \to A$ to $\beta : F(B) \to B$ is a $\mathcal{K}$-morphism $h : A \to B$ with $h \circ \alpha = \beta \circ F(h)$,
A constructive signature $\Sigma = (S, BS, BF, F, P)$ induces a functor

$$H_\Sigma : Set^S \rightarrow Set^S :$$

For all $A, B \in Set^S$, $h \in Set^S(A, B)$ and $s \in S$,

$$H_\Sigma(A)_s = \bigsqcup_{f:e \rightarrow s \in F} A_e,$$
$$H_\Sigma(h)_s = \bigsqcup_{f:e \rightarrow s \in F} h_e.$$
For all $s \in S$ and $f : e \to s \in F$,

\[
H_{\Sigma}(A)_s \xrightarrow{A'_s} [f^A]_{f : e \to s \in F} A_s
\]

$\ell_f$

$\alpha' = \alpha_s \circ \ell_f$

Examples

\[
\begin{align*}
H_{Nat}(A)_{nat} &= 1 + A_{nat}, \\
H_{List(X)}(A)_{list} &= 1 + (X \times A_{list}), \\
H_{Reg(CS)}(A)_{reg} &= 1 + 1 + CS + A_{reg}^2 + A_{reg}^2 + A_{reg}.
\end{align*}
\]
$h : A \to B$ is a $\Sigma$-homomorphism $\iff h$ is an $\text{Alg}_{H\Sigma}$-morphism from $\alpha(A)$ to $\alpha(B)$:

\[
\begin{array}{ccc}
A_e & \xrightarrow{f^A} & A_s \\
\downarrow h_e & & \downarrow h_s \\
B_e & \xrightarrow{f_B} & B_s
\end{array}
\iff
\begin{array}{ccc}
H\Sigma(A)_s & \xrightarrow{\alpha(A)_s} & A_s \\
\downarrow H\Sigma(h)_s & & \downarrow h_s \\
H\Sigma(B)_s & \xrightarrow{\alpha(B)_s} & B_s
\end{array}
\]

$h : \alpha \to \beta$ is an $\text{Alg}_{H\Sigma}$-morphism $\iff h$ is a $\Sigma$-homomorphism from $A(\alpha)$ to $A(\beta)$:

\[
\begin{array}{ccc}
H\Sigma(A)_s & \xrightarrow{\alpha_s} & A_s \\
\downarrow H\Sigma(h)_s & & \downarrow h_s \\
H\Sigma(B)_s & \xrightarrow{\beta_s} & B_s
\end{array}
\iff
\begin{array}{ccc}
A_e & \xrightarrow{f^A(\alpha)} & A_s \\
\downarrow h_e & & \downarrow h_s \\
B_e & \xrightarrow{f^A(\beta)} & B_s
\end{array}
\]
A destructive signature \( \Sigma = (S, BS, BF, F, P) \) induces a functor

\[
H_\Sigma : \text{Set}^S \to \text{Set}^S:
\]

For all \( A, B \in \text{Set}^S \), \( h \in \text{Set}^S(A, B) \) and \( s \in S \),

\[
H_\Sigma(A)_s = \prod_{f : s \to e \in F} A_e,
\]

\[
H_\Sigma(h)_s = \prod_{f : s \to e \in F} h_e.
\]

\( \text{Alg}_\Sigma \) and \( \text{coAlg}_{H_\Sigma} \) are equivalent categories:

Let \( A \in \text{Alg}_\Sigma \) and \( \alpha : H_\Sigma(A) \to A \in \text{coAlg}_{H_\Sigma} \).

The \( H_\Sigma(A) \)-coalgebra \( A' : H_\Sigma(A) \to A \) and the \( \Sigma \)-algebra \( \alpha' \) are defined as follows:

For all \( s \in S \) and \( f : s \to e \in F \),

\[
A_s \xrightarrow{A'_s} H_\Sigma(A)_s \xrightarrow{H_\Sigma(h)_s} A_e
\]

\[
f^{\alpha'} = \pi_f \circ \alpha_s
\]
Examples

\[
\begin{align*}
H_{\text{coNat}}(A)_{\text{nat}} &= 1 + A_{\text{nat}}, \\
H_{\text{coList}}(X)(A)_{\text{list}} &= 1 + (X \times A_{\text{list}}), \\
H_{\text{DAut}}(X,Y)(A)_{\text{state}} &= A^X_{\text{state}} \times Y.
\end{align*}
\]

Haskell implementation of \( \text{Alg}_{\Sigma} \)

Let \( \Sigma = (S, BS, \emptyset, F, \emptyset) \) be a signature,
\( BS = \{X_1, \ldots, X_k\} \), \( S = \{s_1, \ldots, s_m\} \) and \( F = \{f_1 : e_1 \rightarrow e'_1, \ldots, f_n : e_n \rightarrow e'_n\} \).

Each \( \Sigma \)-algebra is an element of the following Haskell datatype:

```
data Sigma x1 ... xk s1 ... sm = Sigma {f1 :: e1 -> e1',...,
                                   fn :: en -> en'}
```

Examples

```
data Nat nat = Nat {zero :: nat, succ :: nat -> nat}
data List x list = List {nil :: list, cons :: x -> list -> list}
```
data Reg cs reg = Reg {eps,mt :: reg, con :: cs -> reg,  
par,seq :: reg -> reg -> reg,  
itra :: reg -> reg}
data Conat nat = Conat {pred :: nat -> Maybe nat}
data Colist x list = Colist {split :: list -> Maybe (x,list)}
data DAut x y state = DAut {delta :: state -> x -> state,  
beta :: state -> y}

Evaluation of terms and formulas

Let \( V \) be a \( \mathcal{T}(S, BS) \)-sorted set of variables, \( A \) be a \( \Sigma \)-algebra and \( A^V \) be the set of valuations of \( V \) in \( A \), i.e., \( \mathcal{T}(S, BS) \)-sorted functions from \( V \) to \( A \).

For all \( g \in A^V \), \( e \in \mathcal{T}(S, BS) \), \( a \in A_e \), \( x \in V_e \) and \( z \in V \).

\[
g[a/x](z) =_{def} \begin{cases} 
a & \text{if } z = x, 
g(z) & \text{otherwise}. \end{cases}
\]
The $T(S, BS)$-sorted extension $g^* : T_\Sigma(V) \to A$ of $g$

is defined as follows:

- For all $x \in V$, $g^*(x) = g(x)$.
- For all $x \in X \in \cup \mathcal{BS}$, $g^*(x) = x$.
- For all $n > 1$, $e_1, \ldots, e_n \in T(S, BS)$, $t = (t_1, \ldots, t_n) \in T_\Sigma(V)_{e_1 \times \ldots \times e_n}$ and $1 \leq i \leq n$, $g^*(\pi_i t) = g^*(t_i)$.
- For all $n > 1$, $e_1, \ldots, e_n \in T(S, BS)$, $1 \leq i \leq n$ and $t \in T_\Sigma(V)_{e_i}$, $g^*(\iota_i t) = (g^*(t), i)$.
- For all $n \in \mathbb{N}$ and $t_1, \ldots, t_n \in T_\Sigma(V)$, $g^*(t_1, \ldots, t_n) = (g^*(t_1), \ldots, g^*(t_n))$.
- For all $f : e \to e' \in F$ and $t \in T_\Sigma(V)_e$, $g^*(f(t)) = f^A(g^*(t))$.
- For all $c \in \{\text{word, bag, set}\}$, $c(t) \in T_\Sigma(V)_{c(e)}$, $g^*(c(t)) = [g^*(t)]_{=c}$.
- For all $n > 0$, $e_i, e \in T(S, BS)$, $x_1 \in V_{e_1}, \ldots, x_n \in V_{e_n}$, $t_1, \ldots, t_n \in T_\Sigma(V)_e$ and $(a, i) \in A_{e_1+\ldots+e_n}$, 
  \[ g^*(\lambda x_1.t_1| \ldots | x_n.t_n)(a, i) = g[a/x_i]^*(t_i). \]
- For all $e, e' \in T(S, BS)$, $t \in T_\Sigma(V)_{e e'}$ and $u \in T_\Sigma(V)_{e'}$, $g^*(t(u)) = g^*(t)(g^*(u))$. 
• For all \( e \in \mathbb{T}(S, BS) \), \( t \in T_\Sigma(V)_2 \) and \( u, v \in T_\Sigma(V)_e \),

\[
g^*(\text{ite}(t, u, v)) = \begin{cases} 
  g^*(u) & \text{if } g^*(t) = 1, \\
  g^*(v) & \text{otherwise}.
\end{cases}
\]

A \( \Sigma \)-term \( t \) is **first-order** if the range of each subterm of \( t \) is first-order.

For all \( e \in \mathbb{T}(S, BS) \) and first-order \( \Sigma \)-terms \( t \), we define:

\[
t^A : A^V \rightarrow A_e \\
g \mapsto g^*(t)
\]

\( \bar{t} : \_^V \rightarrow \mathcal{F}_e \mathcal{U}_S \) with \( \bar{t}_A =_{\text{def}} t^A \) for all \( A \in \text{Alg}_\Sigma \) is a natural transformation:

\[
\begin{array}{ccc}
A^V & \xrightarrow{t^A} & A_e \\
\downarrow h^V & & \downarrow h_e \\
B^V & \xrightarrow{t^B} & B_e
\end{array}
\]
(1) is equivalent to the **Substitution Lemma**: 

For all $g \in A^V$, $\Sigma$-homomorphisms $h : A \to B$ and first-order $\Sigma$-terms $t$,

$$(h \circ g)^*(t) = (h \circ g^*)(t).$$

A interprets a $\Sigma$-formula $\varphi$ over $V$ by the set $\varphi^A \subseteq A^V$ of valuations that satisfy $\varphi$ and is inductively defined as follows:

For all $e \in T(S, BS)$, $p : e \in P$, $t, u \in T_\Sigma(V)_e$, $\varphi, \psi \in Fo_\Sigma(V)$, $s \in S \cup BS$ and $x \in V_s$,

- $\mathsf{True}^A = A^V$,
- $\mathsf{False}^A = \emptyset$,
- $p(t)^A = \{g \in A^V \mid g^*(t) \in p^A\}$,
- $(\neg \varphi)^A = A^V \setminus \varphi^A$,
- $(\varphi \land \psi)^A = \varphi^A \cap \psi^A$,
- $(\varphi \lor \psi)^A = \varphi^A \cup \psi^A$,
- $(\varphi \Rightarrow \psi)^A = (\psi \Leftarrow \varphi)^A = (\neg \varphi \lor \psi)^A$,
$(\psi \iff \varphi)^A = (\varphi \Rightarrow \psi)^A \cap (\varphi \Leftarrow \psi)^A,$

$(\forall x \varphi)^A = \{ g \in A^V \mid \forall a \in A_s : g[a/x] \in \varphi^A \},$

$(\exists x \varphi)^A = \{ g \in A^V \mid \exists a \in A_s : g[a/x] \in \varphi^A \}.$

A satisfies $\varphi \in Fo_{\Sigma}(V)$, written as $A \models \varphi$, if $\varphi^A = A^V$.

The **Substitution Lemma** implies:

For all **negation-free** $\Sigma$-formulas $\varphi$, $g \in A^V$ and $\Sigma$-homomorphisms $h : A \rightarrow B$,

$$g \in \varphi^A \Rightarrow h \circ g \in \varphi^B.$$
An $S$-sorted binary relation $R$ on $A$ is a $\Sigma$-congruence on $A$ if for all $f : e \to e' \in F$ and $(a, b) \in R_e,$ $(f^A(a), f^A(b)) \in R_{e'}$.

If $\Sigma$ is destructive, then $\Sigma$-congruences are also called $\Sigma$-bisimulations.

An $S$-sorted subset $B$ of $A$ is a $\Sigma$-invariant (or $\Sigma$-subalgebra of $A$) if for all $f : e \to e' \in F$ and $a \in A_e,$ $f^A(a) \in A_{e'}$.

A $\Sigma$-algebra $A$ satisfies the **induction principle** if for all $S$-sorted subsets $B$ of $A$, $A \subseteq B$ iff $B$ contains a $\Sigma$-invariant.

| A is initial in $Alg_\Sigma$ $\iff$ A satisfies the induction principle and for all $\Sigma$-algebras $B$ there is a $\Sigma$-homomorphism from $A$ to $B$. |

A $\Sigma$-algebra $A$ satisfies the **coinduction principle** if for all $S$-sorted binary relations $R$ on $A$, $R \subseteq \Delta_A$ iff $R$ is contained in a $\Sigma$-congruence.

| A is final in $Alg_\Sigma$ $\iff$ A satisfies the coinduction principle and for all $\Sigma$-algebras $B$ there is a $\Sigma$-homomorphism from $B$ to $A$. |
Terms for constructive signatures

Let $\Sigma = (S, BS, BF, F)$ be a constructive signature.

$T_\Sigma$ is a $\Sigma$-algebra:

For all $f : e \to s \in F$ and $t \in T_{\Sigma,e}$, $f^{T_\Sigma}(t) =_{\text{def}} ft$.

Let $\sim$ be the least $FT(S, BS)$-sorted equivalence relation on $T_\Sigma$ such that

1. for all $n > 1$, $e_1, \ldots, e_n \in FT(S, BS)$ and $t_i, t'_i \in T_{\Sigma,e_i}$, $1 \leq i \leq n$,

$$t_1 \sim_{e_1} t'_1 \land \cdots \land t_n \sim_{e_n} t'_n \implies (t_1, \ldots, t_n) \sim_{e_1 \times \cdots \times e_n} (t'_1, \ldots, t'_n),$$

2. for all $n > 1$, $e \in FT(S, BS)$ and $t_i, t'_i \in T_{\Sigma,e}$, $1 \leq i \leq n$,

$$t_1 \sim_{e} t'_1 \land \cdots \land t_n \sim_{e} t'_n \implies \text{word}(t_1, \ldots, t_n) \sim_{\text{word}(s)} \text{word}(t'_1, \ldots, t'_n),$$

3. for all $n > 1$, $e \in FT(S, BS)$, $f : [n] \to [n]$ and $t_i, t'_i \in T_{\Sigma,e}$, $1 \leq i \leq n$,

$$t_1 \sim_{e} t'_1 \land \cdots \land t_n \sim_{e} t'_n \implies \text{bag}(f(t_1), \ldots, f(t_n)) \sim_{\text{bag}(s)} \text{bag}(t'_1, \ldots, t'_n),$$
• for all $m, n > 0$, $e \in \mathbb{F}^T(S, BS)$, $t_i \in T_{\Sigma,e}$, $i \in [m]$, and $t'_i \in T_{\Sigma,e}$, $1 \leq i \leq n$,

\[
\forall 1 \leq i \leq m \exists 1 \leq j \leq n : t_i \sim_e t'_j \land \forall 1 \leq j \leq n \exists 1 \leq i \leq m : t_i \sim_e t'_j
\]

implies $\text{set}(t_1, \ldots, t_m) \sim_{\text{set}(s)} \text{set}(t'_1, \ldots, t'_n)$,

• for all $s \in S$, $f : e \rightarrow s \in F$ and $t, t' \in T_{\Sigma,e}$, $t \sim_e t'$ implies $ft \sim_s ft'$,

• for all $X \in BS$, $\sim_X = \Delta_X$.

For simplicity, we identify $T_{\Sigma}$ with $T_{\Sigma}/\sim$.

**$T_{\Sigma}$ is initial in $\text{Alg}_\Sigma$.**

For all $\Sigma$-algebras $A$, the unique $\Sigma$-homomorphism

$$
\text{fold}^A : T_{\Sigma} \rightarrow A
$$

is defined inductively as follows:

For all $f : e \rightarrow s \in F$, $t \in T_{\Sigma,e}$, $c \in \{\text{word, bag, set}\}$, $e' \in S \cup BS$ and $t' \in T^*_{\Sigma,e'}$,

\[
\text{fold}^A_e(ft) = f^A(\text{fold}^A_e(t)),
\]

\[
\text{fold}^A_{c(e')}(c(t')) = [\text{fold}^A_{c》(t')])_{c}.
\]
Haskell implementation of $T_{\Sigma}$ and fold

All collection types are implemented by Haskell’s list type.

Let $BS = \{X_1, \ldots, X_k\}$, $S = \{s_1, \ldots, s_m\}$ and 

$$F = \{c_{ij} : e_{ij} \to s_i \mid 1 \leq i \leq m, 1 \leq j \leq n_i\},$$

i.e., $Alg_{\Sigma}$ is implemented by the following datatype:

```hs
data Sigma x1 ... xk s1 ... sm =
  Sigma {c11 :: e11 -> s1,...,c1n_1 :: e1n_1 -> s1,
         ...
         cm1 :: em1 -> sm,...,cmn_m :: emn_m -> sm}
```

The following datatypes provide the carriers of $T_{\Sigma}$:

```hs
data S1T x1 ... xk = C11 e11 | ... | C1n_1 e1n_1
  ...

data SmT x1 ... xk = Cm1 em1 | ... | Cmn_m emn_m
```
The algebra $T_\Sigma$ is then defined as follows:

\[
\sigmaT :: \Sigma x_1 \ldots x_k (S_1T x_1 \ldots x_k) \ldots (S_mT x_1 \ldots x_k) \\
\sigmaT = \Sigma C_{11} \ldots C_{1n_1} \ldots C_{m1} \ldots C_{mn_m}
\]

Let $1 \leq i \leq m$.

\[
foldSi :: \Sigma x_1 \ldots x_k s_1 \ldots s_m \rightarrow S_iT x_1 \ldots x_k \rightarrow s_i \\
foldSi \ alg \ ti = \text{case } ti \ of \ C_{i1} t \rightarrow c_{i1} \ alg \, \$\ foldE_{i1} \ alg \, t \\
\quad \ldots \\
\quad C_{in_i} t \rightarrow c_{in_i} \ alg \, \$\ foldE_{in_i} \ alg \, t
\]

\[
foldWordSi, foldBagSi, foldSetSi :: \Sigma x_1 \ldots x_k s_1 \ldots s_m \rightarrow [S_iT x_1 \ldots x_k] \rightarrow [s_i] \\
foldWordSi = \text{map } foldSi \\
foldBagSi = \text{map } foldSi \\
foldSetSi = \text{map } foldSi
\]
Let $1 \leq i \leq k$.

foldxi :: Sigma x1 ... xk s1 ... sm -> xi -> xi
foldxi _ = id

foldE1x...xEn :: Sigma x1 ... xk s1 ... sm -> (E1T,...,EnT) -> (E1,...,En)
foldE1x...xEn alg (t1,...,tn) = (foldE1 alg t1,...,foldEn alg tn)

Examples

data NatT = Zero | Succ NatT

natT :: Nat NatT
natT = Nat Zero Succ

foldNat :: Nat nat -> NatT -> nat
foldNat alg t = case t of Zero -> zero alg
Succ t -> succ alg $ foldNat alg t
data `ListT x = Nil | Cons x (ListT x)`

data `ListT x = Nil | Cons x (ListT x)`

`listT :: List x (ListT x)`
`listT = List Nil Cons`

`foldList :: List x list -> ListT x -> list`
`foldList alg t = case t of Nil -> nil alg
Cons x t -> cons alg x $ foldList alg t`

data `RegT cs = Eps | Mt | Con cs | Par (RegT cs) (RegT cs) | Seq (RegT cs) (RegT cs) | Iter (RegT cs)`

data `RegT cs = Eps | Mt | Con cs | Par (RegT cs) (RegT cs) | Seq (RegT cs) (RegT cs) | Iter (RegT cs)`

`regT :: Reg cs (RegT cs)`
`regT cs = Reg Eps Mt Con Var Par Seq Iter`
foldReg :: Reg cs reg -> RegT cs -> reg
foldReg alg t = case t of
    Eps -> eps alg
    Mt -> mt alg
    Con c -> con alg c
    Par t u -> par alg (foldReg alg t) $ foldReg alg u
    Seq t u -> seq alg (foldReg alg t) $ foldReg alg u
    Iter t -> iter alg $ foldReg alg t

Coterms for destructive signatures

Let $\Sigma = (S, BS, BF, F)$ be a destructive signature and

$$Lab^*_\Sigma = \{(d, x, i) | d : s \to (e_1 + \cdots + e_n)^X \in F, x \in X, 1 \leq i \leq n\} \cup \mathbb{N}.$$  

For all $d : s \to e^X$, $a \in A_s$ and $x \in X$, $d^A_x(a) =_{def} d^A(a)(x)$.

$coT^*_\Sigma$ denotes the greatest $\mathbb{FT}(S, BS)$-sorted set of prefix closed partial functions

$$t : Lab^*_\Sigma \rightarrow 1 + \{\text{word, bag, set}\} + \cup BS$$
such that the following conditions hold true:

- For all $s \in S$, $t \in coT_{\Sigma,s}$, $d : s \to (e_1 + \cdots + e_n)^X \in F$ and $x \in X$, $t(\epsilon) = \epsilon$ and there is $1 \leq i \leq n$ such that $(d, x, i) \in \text{def}(t)$, $\lambda w.t((d, x, i)w) \in coT_{\Sigma,e_i}$ and for all $(d, x, i), (d, x, j) \in \text{def}(t)$, $\text{dom}(d) = s$ and $i = j$.

- For all $c \in \{\text{word, bag, set}\}$, $s \in S \cup BS$ and $t \in coT_{\Sigma,c(s)}$, $t(\epsilon) = c$ and there is $n \in \mathbb{N}$ such that for all $1 \leq i \leq n$, $\lambda w.t(iw) \in coT_{\Sigma,s}$, and $\text{def}(t) \cap \text{Lab}_\Sigma = [n]$.

- For all $X \in BS$, $coT_{\Sigma,X} = X$ (here identified with the set $1 \to X$ of functions).

The elements of $coT_\Sigma$ are called $\Sigma$-coterm. 
A $\Sigma$-coterm with destructors $f_1, \ldots, f_8$ that map into sum types.

Each root of a subcoterm is labelled with its sort.

Each leaf is labelled with a base element. Three dots stand for an infinite coterm.
For all $t \in coT_\Sigma$, let $def_1(t) = def(t) \cap Lab_\Sigma$.

Let $\sim$ be the greatest $\mathbb{FT}(S, BS)$-sorted equivalence relation on $coT_\Sigma$ such that

- for all $s \in S$, $t \sim_s t'$ and $d \in def_1(t)$, $\lambda w. t(dw) \sim \lambda w. t'(dw)$,
- for all $s \in S \cup BS$ and $t \sim_{\text{word}(s)} t'$, $D = def 1(t) = def_1(t')$ and for all $i \in D$, $\lambda w. t(iw) \sim_s \lambda w. t'(iw)$,
- for all $s \in S \cup BS$ and $t \sim_{\text{bag}(s)} t'$, $D = def 1(t) = def_1(t')$ and there is $f : [n] \sim [n]$ such that for all $i \in D$, $\lambda w. t(iw) \sim_s \lambda w. t'(f(i)w)$,
- for all $s \in S \cup BS$, $t \sim_{\text{set}(s)} t'$ and $i \in def_1(t)$ there is $j \in def_1(t')$ such that $\lambda w. t(iw) \sim_s \lambda w. t'(jw)$, for all $s \in S \cup BS$, $t \sim_{\text{set}(s)} t'$ and $j \in def_1(t')$ there is $i \in def_1(t)$ such that $\lambda w. t(iw) \sim_s \lambda w. t'(jw)$,
- for all $X \in BS$, $\sim_X = \Delta_X$.

For simplicity, we identify $coT_\Sigma$ with $coT_\Sigma/\sim$. 
\( \text{coT}_\Sigma \) is a \( \Sigma \)-algebra:

For all \( s \in S, t \in \text{coT}_\Sigma, d : s \to (e_1 + \cdots + e_n)^X \in F, x \in X \) and \( w \in \text{Lab}^*_\Sigma, \)

\[
(d, x, i) \in \text{def}(t) \quad \Rightarrow \quad d^{\text{coT}_\Sigma}(t)(x)(w) = t((d, i, x)w).
\]

**Example 1**

Let \( L = \{(\delta, x) \mid x \in X\} \). \( \text{coT}_{\text{DAut}(X,Y)} \) consists of all functions from \( L^* + \text{L}^*\beta \) to \( 1 + Y \), that for all \( w \in L^* \) map \( w \) to \( \epsilon \) and \( w\beta \) to an element of \( Y \):

\[
\text{coT}_{\text{DAut}(X,Y)} \cong 1^{L^*} \times Y^{L^*\beta} \cong Y^{L^*\beta} \xrightarrow{L^*\beta \approx X^*} Y^{X^*}.
\]

Hence \( \text{coT}_{\text{DAut}(X,Y)} \) is \( \text{DAut}(X, Y) \)-isomorphic to the \( \text{DAut}(X, Y) \)-algebra \( \text{Beh}(X, Y) \) of behavior functions that is defined as follows:

\[
\text{Beh}(X, Y)_{\text{state}} = Y^{X^*}.
\]

For all \( f : X^* \to Y, x \in X \) und \( w \in X^* \),

\[
\delta^{\text{Beh}(X,Y)}(f)(x)(w) = f(xw) \quad \text{and} \quad \beta^{\text{Beh}(X,Y)}(f) = f(\epsilon).
\]
For all \( \Sigma \)-algebras \( A \), the unique \( \Sigma \)-homomorphism \( \text{unfold}^A : A \to \text{coT}_\Sigma \) is defined as follows: For all \( s \in \mathbb{F}T(S, BS) \), \( a \in A_s \), \( (d, x, i) \in \text{Lab}_\Sigma \), \( w \in \text{Lab}_\Sigma^* \) and \( k \in \mathbb{N} \),
\[ \text{unfold}_s^A(a)(\epsilon) = \epsilon, \]
\[ \text{unfold}_s^A(a)((d, x, i)w) = \begin{cases} 
\text{unfold}_{e_i}^A(b)(w) & \text{if } d : s \to (e_1 + \cdots + e_n)^X \in F \\
& \text{and } d^A(a)(x) = (b, i), \\
\text{undefined} & \text{otherwise}, \\
\text{unfold}_s^A(a_k)(w) & \text{if } \exists c \in \{\text{word, bag, set}\}, \ e \in S \cup BS : \\
& s = c(e), \ a = [(a_1, \ldots, a_n)]_c = c \\
& \text{and } 1 \leq k \leq n, \\
\text{undefined} & \text{otherwise}. 
\end{cases} \]

**Example 2**

Let \( A \) be a \( DAut(X, Y) \)-algebra, \( \xi : Beh(X, Y) \to coT_{DAut(X,Y)} \) be the isomorphism of Example 1 and \( \text{unfold}B : A \to Beh(X, Y) \) be defined as follows:

For all \( a \in A_{state}, x \in X \) and \( w \in X^* \),
\[ \text{unfold}B^A(a)(\epsilon) = \beta^A(a), \]
\[ \text{unfold}B^A(a)(xw) = \text{unfold}B^A(\delta^A(a)(x))(w). \]
Since $\text{unfold} B$ is $\text{DAut}(X, Y)$-homomorphic,

$$\text{unfold}^A = \xi \circ \text{unfold}^B_A.$$ 

**Haskell implementation of $\text{coT}_\Sigma$ and $\text{unfold}$**

Again, all collection types are implemented by Haskell’s list type.

Let $BS = \{X_1, \ldots, X_k\}$, $S = \{s_1, \ldots, s_m\}$ and

$$F = \{d_{ij} : s_i \to e_{ij} \mid 1 \leq i \leq m, 1 \leq j \leq n_i\},$$

i.e., $\text{Alg}_\Sigma$ is implemented by the following datatype:

```haskell
data Sigma x1 ... xk s1 ... sm =
    Sigma {d11 :: s1 -> e11,...,d1n_1 :: s1 -> e1n_1,
              ...
              dm1 :: sm -> em1,...,dmn_m :: sm -> emn_m}
```
The following datatypes provide the carriers of $\text{co}T_\Sigma$:

\[
\begin{align*}
\text{data } S_1C \ x_1 \ldots \ x_k &= S_1C \{d_{11C} :: e_{11} | \ldots | d_{1n_1C} :: e_{1n_1}\} \\
&\ldots \\
\text{data } S_mC \ x_1 \ldots \ x_k &= S_mC \{d_{m1C} :: e_{m1} | \ldots | d_{mn_mC} :: e_{mn_m}\}
\end{align*}
\]

The algebra $\text{co}T_\Sigma$ is then defined as follows:

\[
\begin{align*}
\sigma_C :: \Sigma x_1 \ldots x_k (S_1C \ x_1 \ldots x_k) \ldots (S_mC \ x_1 \ldots x_k) \\
\sigma_C &= \Sigma d_{11C} \ldots d_{1n_1C} \ldots d_{m1C} \ldots d_{mn_mC}
\end{align*}
\]

Let $1 \leq i \leq m$.

\[
\begin{align*}
\text{unfold}_{Si} :: \Sigma x_1 \ldots x_k s_1 \ldots s_m \rightarrow si \rightarrow S_iC \ x_1 \ldots x_k \\
\text{unfold}_{Si} \ \text{alg} \ \text{ai} &= S_iC \ (\text{unfold}_{Ei1} \ \text{alg} \ \$ \ di1 \ \text{alg} \ \text{ai}) \\
&\ldots \\
&\ (\text{unfold}_{Ein_i} \ \text{alg} \ \$ \ din_i \ \text{alg} \ \text{ai})
\end{align*}
\]

\[
\begin{align*}
\text{unfoldWord}_{Si}, \text{foldBag}_{Si}, \text{foldSet}_{Si} :: \Sigma x_1 \ldots x_k s_1 \ldots s_m \\
&\rightarrow [si] \rightarrow [S_iT x_1 \ldots x_k]
\end{align*}
\]
unfoldWordSi = map unfoldSi
unfoldBagSi = map unfoldSi
unfoldSetSi = map unfoldSi

Let $1 \leq i \leq k$ and $n > 1$.

unfoldxi :: Sigma $x_1 \ldots x_k$ $s_1 \ldots s_m$ -> $x_i$ -> $x_i$
unfoldxi _ = id

unfoldE^xi :: Sigma $x_1 \ldots x_k$ $s_1 \ldots s_m$ -> ($x_i$ -> E) -> $x_i$ -> EC
unfoldE^xi alg f x = unfoldE alg $ f x

data Sum_n e_1 \ldots e_n = S_1 e_1 \mid \ldots \mid S_n e_n
Let $1 \leq i \leq n$.

\[
\text{unfoldE}_1+\ldots+\text{E}_n :: \Sigma x_1 \ldots x_k s_1 \ldots s_m \to \text{Sum}_n \text{E}_1 \ldots \text{E}_n \\
\to \text{Sum}_n \text{E}_1^C \ldots \text{E}_n^C
\]

\[
\text{unfoldE}_1+\ldots+\text{E}_n \text{ alg } a = \text{case } a \text{ of } S_1 a \to \text{unfoldE}_1 \text{ alg } a \\
\ldots \\
S_n a \to \text{unfoldE}_n \text{ alg } a
\]

**Examples**

\[
\text{data } \text{ConatC} = \text{ConatC} \{ \text{predC} :: \text{Maybe ConatC} \}
\]

\[
\text{conatC} :: \text{Conat } \text{ConatC} \\
\text{conatC} = \text{Conat } \text{predC}
\]

\[
\text{unfoldConat} :: \text{Conat } \text{n} \to \text{nat} \to \text{nat} \to \text{ConatC} \\
\text{unfoldConat} \text{ alg } \text{nat} = \text{ConatC} \$ \text{ do } \text{nat} \leftarrow \text{pred} \text{ alg } \text{nat} \\
\quad \text{Just } \$ \text{unfoldConat} \text{ alg } \text{nat}
\]
data `ColistC` `x` = `ColistC` {`splitC` :: `Maybe` (`x`,`ColistC` `x`)}

colistC :: Colist `x` (ColistC `x`)
colistC = Colist `splitC`

unfoldColist :: Colist `x` `list` -> `list` -> ColistC `x`
unfoldColist `alg` `list` = ColistC $ do (`x`,`list`) <- `split` `alg` `list`
                                Just (`x`,unfoldColist `alg` `list`)

data `StateC` `x` `y` = `StateC` {`deltaC` :: `x` -> State `x` `y`, `betaC` :: `y`}

dAutC :: DAut `x` `y` (StateC `x` `y`)
dAutCot = DAut `deltaC` `betaC`

unfoldDAut :: DAut `x` `y` state -> state -> StateC `x` `y`
unfoldDAut `alg` `state` = StateC (unfoldDAut `alg` . `delta` `alg` `state`)  
                                 (`beta` `alg` `state`)
Realization of elements of final algebras

Given a $\Sigma$-algebra $A$, a final $\Sigma$-algebra $\text{Fin}$, $a \in A$ and $f \in \text{Fin}$,

\[(A, a) \text{ realizes } f \text{ iff } \text{unfold}^A(a) = f.\]

Example 3

Let $A$ be the following $\text{Acc}({\mathbb{Z}})$-algebra:

\[
eo :: \text{DAut Int Bool Bool}
\]
\[
eo = \text{DAut (\ \text{\state} \to \text{if state then even else not . even) id}}
\]

and

\[
f : \mathbb{Z}^* \to 2
\]
\[
(x_1, \ldots, x_n) \mapsto \sum_{i=1}^n x_i \text{ is even}
\]

\[
g : \mathbb{Z}^* \to 2
\]
\[
(x_1, \ldots, x_n) \mapsto \sum_{i=1}^n x_i \text{ is odd}
\]
Since $h : A \to \text{Beh}(\mathbb{Z}, 2)$ with $h(1) = f$ and $h(0) = g$ is $\text{Acc}(\mathbb{Z})$-homomorphic,

\[ h = \text{unfold}^{eo}. \]

Hence $(A, 1)$ realizes $f$ and $(A, 0)$ realizes $g$. 
Recursive equations

Given a constructive signature $C\Sigma = (S, BS, BF, C)$ and a destructive signature $D\Sigma = (S, BS', BF', D)$, $\Psi = (C\Sigma, D\Sigma)$ is called a bisignature.

Let $\Sigma = C\Sigma \cup D\Sigma$. A set

$$E = \{dc(x_1, \ldots, x_{n_c}) = t_{d,c} \mid c : e_1 \times \cdots \times e_{n_c} \rightarrow s \in C, \ d : s \rightarrow e \in D\}$$

of $\Sigma$-equations is a system of recursive $\Psi$-equations if the following conditions hold true:

- For all $d \in D$ and $c \in C$, $\text{freeVars}(t_{d,c}) \subseteq \{x_1, \ldots, x_{n_c}\}$.
- $C$ is the union of disjoint sets $C_1$ and $C_2$.
- For all $d \in D$, $c \in C_1$ and subterms $du$ of $t_{d,c}$, $u$ is a variable and $t_{d,c}$ is a term without elements of $C_2$.
  \Rightarrow \text{no nesting of destructors, but possible nestings of constructors of } C_1
- For all $d \in D$, $c \in C_2$, subterms $du$ of $t_{d,c}$ and paths $p$ of (the tree representation of) $t_{d,c}$, $u$ consists of destructors and a variable and $p$ contains at most one occurrence of an element of $C_2$.
  \Rightarrow \text{no nesting of constructors of } C_2, \text{ but possible nestings of destructors}
Let $E$ be a system of recursive $\Psi$-equations and $A$ be a $C \Sigma$-algebra. An **inductive solution of $E$ in $A$** is a $\Sigma$-algebra $B$ with $B|_{C \Sigma} = A$ that satisfies $E$.

(1) If $C_2$ is empty, then $E$ has a unique inductive solution in every initial $C \Sigma$-algebra.

Let $E$ be a system of recursive $\Psi$-equations and $A$ be a $D \Sigma$-algebra. A **coinductive solution of $E$ in $A$** is a $\Sigma$-algebra $B$ with $B|_{D \Sigma} = A$ that satisfies $E$.

(2) $E$ has a unique coinductive solution in every final $D \Sigma$-algebra.

Moreover, $T_{C \Sigma} \in Alg_{D \Sigma}$, $coT_{D \Sigma} \in Alg_{C \Sigma}$ and $fold^{coT_{D \Sigma}} = unfold^{T_{C \Sigma}}$. 

![Diagram]

\[
T_{C \Sigma} \xrightarrow{inc} T_{C \Sigma}(coT_{D \Sigma}) \xrightarrow{inc} coT_{D \Sigma} \\
\downarrow id \quad \quad \quad \quad \quad \downarrow id \\
\leftarrow \quad \quad \quad \quad \quad \leftarrow \\
T_{C \Sigma} \xrightarrow{fold^{coT_{D \Sigma}}} \leftarrow T_{C \Sigma}(coT_{D \Sigma}) \xrightarrow{fold^{coT_{D \Sigma}}} coT_{D \Sigma}
\]
Example 4

Let

\[ C\Sigma = (\{\text{list}\}, \emptyset, \emptyset, \{\text{evens}, \text{odds}, \text{exchange}, \text{exchange}' : \text{list} \to \text{list}\}), \]

\[ \Psi = (C\Sigma, \text{Stream}(X)) \]

and \( s \in V \). The equations

\[
\begin{align*}
\text{head(evens}(s)) & = \text{head}(s), & \text{tail(evens}(s)) & = \text{evens}(\text{tail}(\text{tail}(s))), \\
\text{head(odds}(s)) & = \text{head}(\text{tail}(s)), & \text{tail(odds}(s)) & = \text{odds}(\text{tail}(\text{tail}(s))), \\
\text{head(exchange}(s)) & = \text{head}(\text{tail}(s)), & \text{tail(exchange}(s)) & = \text{exchange}'(s), \\
\text{head(exchange}'(s)) & = \text{head}(s), & \text{tail(exchange}'(s)) & = \text{exchange}(\text{tail}(\text{tail}(s)))
\end{align*}
\]

form a system \( E \) of recursive \( \Psi \)-equations.

\( \text{evens}(s) \) und \( \text{odds}(s) \) list the elements of \( s \) at even resp. odd positions.

\( \text{exchange}(s) \) exchanges the elements at even positions with those at odd positions.

\[ (2) \implies E \text{ has a unique coinductive solution in the final Stream}(X)\text{-algebra.} \]
Example 5

Let $CS$ be a set of nonempty sets of constants, $X = \bigcup CS$,

$$D\Sigma = (\{\text{reg}\}, \{2, X\},$$

$$\{\text{max}, \ast : 2 \times 2 \to 2\} \cup \{\_ \in C : X \to 2 \mid C \in CS\},$$

$$\{\delta : \text{reg} \to \text{reg}^X, \beta : \text{reg} \to 2\}$$

$\Psi = (\text{Reg}(CS), D\Sigma)$, $C \in CS$ and $t, u \in V$. The equations

$$\delta(\text{eps}) = \lambda x. m t,$$

$$\delta(\text{mt}) = \lambda x. m t,$$

$$\delta(\overline{C}) = \lambda x. \text{ite}(\chi(C)(x), \text{eps}, \text{mt})$$

$$\delta(\text{par}(t, u)) = \lambda x. \text{par}(\delta(t)(x), \delta(u)(x)),$$

$$\delta(\text{seq}(t, u)) = \lambda x. \text{ite}(\beta(t), \text{par}(\text{seq}(\delta(t)(x), u), \delta(u)(x))$$

$$\text{seq}(\delta(t)(x), u)),$$

$$\delta(\text{iter}(t)) = \lambda x. \text{seq}(\delta(t)(x), \text{iter}(t)),$$

$$\beta(\text{eps}) = 1,$$

$$\beta(\text{mt}) = 0,$$

$$\beta(\overline{C}) = 0,$$
Recursive equations

\[ \beta(par(t, u)) = \max\{\beta(t), \beta(u)\}, \]
\[ \beta(seq(t, u)) = \beta(t) \ast \beta(u), \]
\[ \beta(iter(t)) = 1. \]

form the system \(BRE\) of recursive \(\Psi\)-equations.

(1) \(\implies\) \(BRE\) has a unique inductive solution \(A\) in the initial \(Reg(CS)\)-algebra \(T_{Reg(CS)}\).

\(Bro(CS) = \text{def } A|_{Acc(X)}\) is called the Brzozowski automaton.

(2) \(\implies\) \(BRE\) has a unique coinductive solution \(B\) in the final \(Acc(X))\)-algebra \(Pow(X)\), which is defined as follows:

For all \(L \subseteq X^*\) and \(x \in X\),

\[ Pow(X)_{state} = \mathcal{P}(X^*), \]
\[ \delta^{Pow(X)}(L)(x) = \{w \in X^* \mid xw \in L\}, \]
\[ \beta^{Pow(X)}(L) = \begin{cases} 0 & \text{falls } \epsilon \in L, \\ 1 & \text{sonst.} \end{cases} \]
$\text{Lang}(X) = B|_{\text{Reg}(CS)}$ is defined as follows:

For all $L, L' \subseteq X^*$ and $C \in CS$,

- $\text{eps}^{\text{Lang}(X)} = \{\epsilon\}$,
- $\text{mt}^{\text{Lang}(X)} = \emptyset$,
- $\overline{C}^{\text{Lang}(X)} = C$,
- $\text{par}^{\text{Lang}(X)}(L, L') = L \cup L'$,
- $\text{seq}^{\text{Lang}(X)}(L, L') = L \cdot L'$,
- $\text{iter}^{\text{Lang}(X)}(L) = L^*$.

$$ = \implies \text{fold}^{\text{Lang}(X)} = \text{unfold}^{\text{Bro}(CS)} : \text{T}_{\text{Reg}(CS)} \rightarrow \mathcal{P}(X^*)$$

$$ = \implies \text{For all } t \in \text{T}_{\text{Reg}(CS)}, \ (\text{Bro}(CS), t) \text{ realizes the characteristic function of the language } \text{fold}^{\text{Lang}(X)}(t) \text{ of } t.$$
Bro(CS) can be optimized to Norm(CS) by simplifying its states with respect to semiring axioms between each two transition steps:

For all \( t \in T_{Reg(CS)} \), \( \delta^{Norm(CS)}(t) = \text{def} \ reduce \circ \delta^{Bro(CS)}(t) \).

Let \( \Psi = (C\Sigma, D\Sigma) \) be a bisignature, \( C\Sigma = (S, BS, BF, C) \), \( D\Sigma = (S, BS', BF', D) \), \( A \) be a \((C\Sigma \cup D\Sigma)\)-algebra and \( \sim \) be an \( S \)-sorted relation on \( A \).

The \( C \)-equivalence closure \( \sim_C \) of \( \sim \) is the least \( S \)-sorted equivalence relation on \( A \) that contains \( \sim \) and satisfies the following condition: For all \( c : e \to s \in C \) and \( a, b \in A_e \),

\[
a \sim_C b \quad \text{implies} \quad c^A(a) \sim_C c^A(b).
\]

\( \sim \) is a \( D\Sigma \)-congruence up to \( C \) if for all \( d : s \to e \in D \) and \( a, b \in A_s \),

\[
a \sim b \quad \text{implies} \quad d^A(a) \sim_C d^A(b).
\]

\( A|_{D\Sigma} \) is final in \( Alg_{D\Sigma} \),

\( \sim \) is a \( D\Sigma \)-congruence up to \( C \),

there is a system of recursive \( \Psi \)-equations \[ \implies \sim_C \text{ is a } D\Sigma \text{-congruence.} \] (3)
Example 6

Let \( \Psi \) be as in Example 5 and \( V = \{x, y, z\} \),

\[
\sim = \{ (g^*(seq(x, par(y, z))), g^*(par(seq(x, y), seq(x, z))) \mid g : T_{Reg(CS)}(V) \rightarrow Pow(X) \}
\]

is an \( Acc(X) \)-congruence up to \( C \).

\[
\Rightarrow \quad \text{Since } Pow(X) \text{ is final in } Alg_{Acc(X)}, \ (3) \text{ implies that } \sim_C \text{ is } Acc(X)\text{-congruence.}
\]

\[
\Rightarrow \quad \text{Since } Pow(X) \text{ satisfies the coinduction principle, } \sim \subseteq \Delta_{Pow(X)} \text{ and thus }
\]

\[
Pow(X) \models seq(x, par(y, z)) = par(seq(x, y), seq(x, z)).
\]

Given a bisignature \( \Psi \), we have seen that a system \( E \) of recursive \( \Psi \)-equations defines

- destructors on constructors inductively or
- constructors on destructors coinductively.

Similarly,
• the rules of a **structural operational semantics** (SOS) or a **transition system specification**
• or a **distributive law** \( \lambda : TD \to DT \) of an endofunctor \( T \) over an endofunctor \( D \)

provide both

• an **inductive definition** of a semantics (destructors; \( D \)) of the syntax (constructors; \( T \)) of some language and
• a **coinductive definition** of the constructors on the language’s behavioral model, given by the destructors.

Can \( \lambda \) be derived from \( \Psi \) such that \((C\Sigma \cup D\Sigma)\)-algebras satisfying \( E \) correspond to \( \lambda \)-bialgebras?

With regard to their domain and range types, functions that come as inductive or coinductive solutions of systems of recursive \( \Psi \)-equations are destructors or constructors, respectively.
Recursion schemas that define functions with more general domain or range types have been studied mainly in category-theoretical settings like distributive laws or adjunctions. For instance, in Ralf Hinze, *Adjoint Folds and Unfolds*, functions are defined as adjoint (co)extensions of folds or unfolds.

We think that most examples investigated in category-theoretical settings can be presented as systems of recursive $\Psi$-equations. Maybe, in some cases, the syntactic conditions given here must be weakened, but in many cases, they will already be weak enough – due to our powerful term language that involves polynomial as well as power and collection types.

Here are some modeling formalisms where coinductive definability has already been studied in detail:

- basic process algebra
  - Rutten, *Processes as Terms: Non-well-founded Models for Bisimulation*

- stream expressions and infinite sequences
  - Rutten, *A Coinductive Calculus of Streams*

- tree expressions and infinite trees
  - Silva, Rutten, *A Coinductive Calculus of Binary Trees*
• arithmetic expressions and valuations, CCS and transition trees  
  ☢ Hutton, *Fold and Unfold for Program Semantics*

• stream function expressions and causal stream functions  
  ☢ Hansen, Rutten, *Symbolic Synthesis of Mealy Machines from Arithmetic Bitstream Functions*
Iterative equations

Let \( \Sigma = (S, BS, BF, F) \) be a constructive signature and \( V \) be an \( S \)-sorted set.

An \( S \)-sorted function

\[
E : V \rightarrow T_{\Sigma}(V)
\]

with \( E(V) \cap V = \emptyset \) is called a system of iterative \( \Sigma \)-equations.

Let \( A \) be a \( \Sigma \)-algebra and \( A^V \) be the set of \( S \)-sorted functions from \( V \) to \( A \).

\( g \in A^V \) solves \( E \) in \( A \) if \( g^* \circ E = g \).

Iterative equations are uniquely solvable in the following tree model:

\( CT_{\Sigma} \) denotes the greatest \( \mathbb{FT}(S, BS) \)-sorted set of prefix closed partial functions

\[
t : \mathbb{N}^* \rightarrow F + \{\text{word, bag, set}\} + \cup BS
\]

such that

- for all \( s \in S \) and \( t \in CT_{\Sigma,s} \) there are \( n > 0 \) and \( e_1, \ldots, e_n \in \mathbb{FT}(S, BS) \) with

\[
t(\varepsilon) : e_1 \times \cdots \times e_n \rightarrow s \in F, \text{ def}(t) \cap \mathbb{N} = [n] \text{ and } \lambda w. t(iw) \in CT_{\Sigma,e_i} \text{ for all } 1 \leq i \leq n,
\]
Iterative equations

• for all $c \in \{\text{word, bag, set}\}$, $s \in S \cup BS$ and $t \in CT_{\Sigma,c(s)}$ there is $n_t \in \mathbb{N}$ with $t(\epsilon) = c$, $\text{def}(t) \cap \mathbb{N} = [n_t]$ and $\lambda w.t(iw) \in CT_{\Sigma,s}$ for all $1 \leq i \leq n_t$,

• for all $X \in BS$, $CT_{\Sigma,X} = X$ (again identified with the set $1 \rightarrow X$ of functions).

Let $\sim$ be the greatest $\mathcal{FT}(S, BS)$-sorted equivalence relation on $CT_{\Sigma}$ such that

• for all $s \in S$ and $t \sim_s t'$, $t(\epsilon) = t'(\epsilon)$ and for all $i \in \mathbb{N}$, $\lambda w.t(iw) \sim \lambda w.t'(iw)$,

• for all $s \in S \cup BS$ and $t \sim_{\text{word}(s)} t'$, $n_t = n_{t'}$ and for all $i \in [n_t]$, $\lambda w.t(iw) \sim_s \lambda w.t'(iw)$,

• for all $s \in S \cup BS$, $t \sim_{\text{bag}(s)} t'$ and $f : [n_t] \rightarrow [n_{t'}]$, $n_t = n_{t'}$ and for all $i \in [n_t]$, $\lambda w.t(f(i)w) \sim_s \lambda w.t'(iw)$,

• for all $s \in S \cup BS$, $t \sim_{\text{set}(s)} t'$, $i \in [n_t]$ and $j \in [n_{t'}]$ there are $k \in [n_{t'}]$ and $l \in [n_t]$ such that $\lambda w.t(iw) \sim_s \lambda w.t(kw)$ and $\lambda w.t(lw) \sim_s \lambda w.t'(jw)$,

• for all $X \in BS$, $\sim_X = \Delta_X$.

For simplicity, we identify $CT_{\Sigma}$ with $CT_{\Sigma}/\sim$.

The elements of $CT_{\Sigma}$ are called $\Sigma$-trees.
Iterative equations

$CT_{\Sigma}$ is a $\Sigma$-algebra:

For all $f : e \rightarrow s \in F$, $t = (t_1, \ldots, t_n) \in CT_{\Sigma,e}$ and $w \in \mathbb{N}^*$,

$$f^{CT_{\Sigma}}(t)(w) =_{def} \begin{cases} f & \text{if } w = \epsilon, \\ t_i(v) & \text{if } \exists i \in \mathbb{N} : iv = w. \end{cases}$$

$f^{CT_{\Sigma}}(t)$ is also written as $ft$ and $f^{CT_{\Sigma}}(\epsilon)$ as $f$.

Let $\Sigma_\perp = (S, BS, BF, F \cup \{ \perp_s : 1 \rightarrow s \mid s \in S \})$ and $\leq$ be the least reflexive, transitive and $\Sigma$-congruent $S$-sorted relation on $CT_{\Sigma_\perp}$ such that for all $s \in S$ and $t \in CT_{\Sigma_\perp,s}$, $\perp_s \leq t$.

Kleene’s fixpoint theorem $\implies$

$CT_{\Sigma_\perp}$ is initial in $CAlg_{\Sigma}$,

the category of $\omega$-continuous $\Sigma$-algebras as objects and strict and $\omega$-continuous $\Sigma$-homo-morphisms.
Elgot’s Theorem (see Goguen et al., *Initial Algebra Semantics and Continuous Algebras*)

Each system of iterative \( \Sigma \)-equations has a unique solution in \( CT_\Sigma \).

\( \Sigma \) induces the destructive signature \( co\Sigma \) with \( H_\Sigma = H_{co\Sigma} \):

\[
co\Sigma = (S, BS, BF, \{d_s : s \to \bigsqcup_{f:e \to s \in F} e \mid s \in S\} \cup \{\pi_i : e_1 \times \cdots \times e_n \to e_i \mid n > 1, e_1, \ldots, e_n \in FT(S, BS), 1 \leq i \leq n\})
\]

Here each product type \( e_1 \times \cdots \times e_n \) is regarded as an additional sort. The projections \( \pi_i : e_1 \times \cdots \times e_n \to e_i, 1 \leq i \leq n \), provide its destructors.

\( CT_\Sigma \) is a \( co\Sigma \)-algebra:

For all \( s \in S \) and \( t \in CT_{\Sigma,s} \) such that \( t(\epsilon) \) is \( n \)-ary,

\[
d_s^{CT_\Sigma}(t) =_{def} ((\lambda w.t(1w), \ldots, \lambda w.t(nw)), t(\epsilon)).
\]
CT\_\Sigma is final in Alg\_\text{co}\Sigma.

For all co\Sigma-algebras A, the unique \Sigma-homomorphism unfold\_A : A \rightarrow CT\_\Sigma is defined as follows: For all s \in S, a \in A_s, i \in \mathbb{N} and w \in \mathbb{N^*},

\[
\begin{align*}
\text{unfold}_A(a)(\epsilon) & = f, \\
\text{unfold}_A(a)(iw) & = \begin{cases} \\
\text{unfold}_A(a_i)(w) & \text{if } \pi_1(d_s^A(a)) = (a_1, \ldots, a_n) \land 1 \leq i \leq n, \\
\text{undefined} & \text{otherwise.}
\end{cases}
\end{align*}
\]

CT\_\Sigma \cong coT\_\text{co}\Sigma.
A coΣ-coterm
... and the corresponding $\Sigma$-tree:

Let $E : V \to T_{\Sigma}(V)$ be a system of iterative $\Sigma$-equations.

The co$\Sigma$-algebra $T^E$ is defined as follows: For all $s \in S$, $f : e \to s \in F$, $t \in T_{\Sigma}(V)_e$ and $x \in V_s$,

\[
T^E_s = T_{\Sigma}(V)_s,
\]
\[
d^E_s(ft) = (t, f),
\]
\[
d^E_s(x) = d^E_s(E(x)).
\]
\textbf{Iterative equations}

\[ \text{unfold}^{T^E} \circ \text{inc}_V : V \to CT_\Sigma \text{ solves } E \text{ in } CT_\Sigma. \tag{4} \]

\[ g : V \to CT_\Sigma \text{ solves } E \text{ in } CT_\Sigma \text{ iff } g^* : T^E \to CT_\Sigma \text{ is co}\Sigma\text{-homomorphim.} \tag{5} \]

\[(4) \land (5) \implies \text{ Each system of iterative } \Sigma\text{-equations has a unique solution in } CT_\Sigma.\]

An alternative proof of this result is given in Example 8 below.

\textbf{Example 7} \quad \Psi = (\Sigma, \text{co}\Sigma)

For all \( e \in T(S, BS) \), let \( x_e \) be a variable that is not contained in \( V \).

\[ DC = \{ d_s(f(x)) = \iota_f(x) \mid s \in S, \ f : e \to s \in F \} \]

is a system of recursive \( \Psi \)-equations.

\[(2) \implies DC \text{ has a unique coinductive solution in } CT_\Sigma. \tag{6} \]
Context-free grammars with base sets

A context-free grammar $G = (S, BS, Z, R)$ (with base sets) consists of

- a set $S$ of sorts (also called nonterminals),
- a set $BS$ of nonempty base sets,
- a set $Z$ of terminals,
- a set $R$ of rules $s \rightarrow w$ with $s \in S$ and $w \in (S \cup BS \cup Z)^*$.

The abstract syntax of $G$ is the constructive signature $\Sigma(G) = (S, BS, \emptyset, F, \emptyset)$ with

$$ F = \{ f_r : e_{i_1} \times \ldots \times e_{i_k} \rightarrow s \mid r = (s \rightarrow e_1 \ldots e_n) \in R, \ s \in S, \ \{i_1, \ldots, i_k\} = \{1 \leq i \leq n \mid e_i \in S \cup BS\} \}. $$

The elements of the term algebra $T_{\Sigma(G)}$ are usually called syntax trees.
Let \( X = Z \cup \bigcup BS \).

The \( \Sigma(G) \)-word algebra \( Word(G) \) recovers the concrete from the abstract syntax:

For all \( s \in S \), \( Word(G)_s = X^* \). For all \( r = (s \rightarrow e_1 \ldots e_n) \in R \),

\[
\begin{align*}
\phi_r^{Word(G)} & : (X^*)^k \rightarrow X^* \\
(w_{i_1}, \ldots, w_{i_k}) & \mapsto v_1 \ldots v_n 
\end{align*}
\]

where \( \{i_1, \ldots, i_k\} = \{1 \leq i \leq n \mid e_i \in S \cup BS\} \) and for all \( 1 \leq i \leq n \),

\[
\phi_i = \begin{cases} 
  e_i & \text{if } e_i \in Z, \\
  w_i & \text{if } e_i \in S \cup BS.
\end{cases}
\]

The language \( L(G) \) of \( G \) is the set of words over \( X \) that result from folding a syntax tree in \( Word(G) \):

\[
L(G) = \text{fold}^{Word(G)}(T_{\Sigma(G)}).
\]
Let $U$ be the forgetful functor from $Alg_{\Sigma(G)}$ to $Set^S$ and $(M : Set^S \to Set^S, \eta, \epsilon)$ be a monad that encapsulates the compiler output and – in the case of incorrect input – produces error messages.

A generic compiler for $G$ is a natural transformation

$$compile_G : X^* \to MU$$

such that for all $w \in X^*$,

$$M(\lambda v. v = w)(compile_G^{Word(G)}(w)) = M(\lambda v. 1)(compile_G^{Word(G)}(w)). \quad (7)$$

A mapping $compile_G : X^* \to MU$ is a natural transformation iff for all $\Sigma(G)$-homomorphisms $h : A \to B$,

$$M(h) \circ compile_G^A = compile_G^B; \quad (8)$$

in particular,

$$M(\text{fold}^A) \circ compile_G^{T_{\Sigma(G)}} = compile_G^A.$$

Hence a the application $compile_G^A$ of the generic compiler to the “target language” $A$ decomposes into the parser for $G$, $compile_G^{T_{\Sigma(G)}}$, and the folding in $A$ of the syntax trees ($\Sigma(G)$-terms) produced by the parser.
The correctness of $\text{compile}^B_G$ with respect to a source model, given by a further $\Sigma(G)$-algebra $Sem$, and a target model $Mach$ ("abstract machine") is usually expressed as the commutativity of the following diagram:

\[
\begin{array}{c}
T_{\Sigma(G)} \xrightarrow{\text{fold}^A} A \\
\downarrow \text{fold}^{Sem} \quad (9) \quad \downarrow \text{execute} \\
Sem \xrightarrow{\text{encode}} Mach
\end{array}
\]

$\text{execute}$ executes a "target program" $a \in A$ on the abstract machine $Mach$.

$\text{encode}$ expresses the source model in terms of the target model.
In the sequel, we regard the constructors \( \text{par} \) and \( \text{seq} \) of \( \text{Reg}(CS) \) as operations of mutable arity and thus write

- \( \text{par}(t_1, \ldots, t_n) \) instead of \( \text{par}(t_1, \text{par}(t_2, \ldots, \text{par}(t_{n-1}, t_n) \ldots)) \) and
- \( \text{seq}(t_1, \ldots, t_n) \) instead of \( \text{seq}(t_1, \text{seq}(t_2, \ldots, \text{seq}(t_{n-1}, t_n) \ldots)) \).

\( \text{par}(t) \) and \( \text{seq}(t) \) stand for \( t \).

\( G \) induces an iterative system of \( \text{Reg}(CS) \)-equations:

\[
E_G : S \rightarrow T_{\text{Reg}(CS)}(S)
\]

\[
s \mapsto \text{par}(\overline{w_1}, \ldots, \overline{w_k})
\]

where \( \{w_1, \ldots, w_k\} = \{w \in (S \cup CS)^* \mid s \rightarrow w \in R\} \)

and for all \( n > 1, e_1, \ldots, e_n \in S \cup CS \) and \( s \in S \),

\[
\overline{e_1 \ldots e_n} = \text{seq}(\overline{e_1}, \ldots, \overline{e_n}),
\]

\[
\overline{s} = s.
\]

\( E_G \) is called the system of equations for \( G \).
The function $sol_G : S \rightarrow \mathcal{P}(X^*)$ with $sol_G(s) = L(G)_s$ for all $s \in S$ solves $E_G$ in $\text{Lang}(X)$.

(10)

$sol_G$ is the least solution of $E_G$ in $\text{Lang}(X)$, i.e., for all solutions $g$ of $E_G$ in $\text{Lang}(X)$ and all $s \in S$, $sol_G(s) \subseteq g(s)$. 
Constructing recursive from iterative equations

Let $\Psi = (C\Sigma, D\Sigma)$, $C\Sigma = (S, BS, BF, C)$, $\Sigma = C\Sigma \cup D\Sigma$ and $V \in Set^S$.

$$C\Sigma_V = (S, BS \cup \{V_s \mid s \in S\}, BF, C \cup \{\text{in}_s : V_s \rightarrow s \mid s \in S\}),$$

$$\Psi_V = (C\Sigma_V, D\Sigma),$$

$$\Sigma_V = C\Sigma_V \cup D\Sigma.$$

Let $E : V \rightarrow T_{C\Sigma}(V)$ be a system of iterative $C\Sigma$-equations, $\text{rec}(E)$ be a system of recursive $\Psi_V$-equations and $A$ be a $\Sigma$-algebra.

$\text{rec}(E)$ simulates $E$ in $A$ if for all solutions $g : V \rightarrow A$ of $E$, the $\Sigma_V$-algebra $A_g$ with $A_g|_\Sigma = A$ and $\text{in}^{A_g}_s = g_s$ for all $s \in S$ satisfies $\text{rec}(E)$.

Suppose that $\text{rec}(E)$ simulates $E$ in $A$ and $A$ is final in $\text{Alg}_{D\Sigma}$. Then $E$ has a unique solution in $A$. \hfill (11)

\textbf{Proof.} Let $g, h : V \rightarrow A$ solve $E$ in $A$. We extend $A$ to $\Sigma_V$-algebras $A_1, A_2$ by defining $\text{in}^{A_1}_s = g_s$ and $\text{in}^{A_2}_s = h_s$ for all $s \in S$. By assumption, both $A_1$ and $A_2$ satisfy $\text{rec}(E)$. Since $A|_{D\Sigma}$ is final in $\text{Alg}_{D\Sigma}$, (2) implies that the coinductive solution of $\text{rec}(E)$ in $A|_{D\Sigma}$ is unique. Hence $A_1 = A_2$ and thus for all $s \in S$, $g_s = \text{in}^{A_1}_s = \text{in}^{A_2}_s = h_s$. \hfill \Box
\( \sigma_V : V \to T_{\Sigma V} \) denotes the substitution with \( \sigma_V(x) = in_s x \) for all \( x \in V_s \) und \( s \in S \). For all \( \Sigma_V \)-algebras \( A \),

\[
(in^A)^* = fold^A \circ \sigma^*_V : T_{\Sigma}(V) \to A,
\]

where \( in^A = (in^A_s : V_s \to A_s)_{s \in S} \).

**Example 8** \( \Psi = (C\Sigma, coC\Sigma) \)

Let \( C\Sigma = (S, BS, BF, C) \) be a constructive signature, \( D\Sigma = co\Sigma \) and \( E : V \to T_{C\Sigma}(V) \) be a system of iterative \( C\Sigma \)-equations.

\[
rec(E) = \{ d_s(in_s(x)) = \iota_c(\sigma^*_V(t)) \mid s \in S, x \in V_s, E(x) = ct \}
\]

is a system of recursive \( \Psi_V \)-equations.

By (6), the system \( DC \) of recursive \( \psi \)-equations has a unique coinductive solution \( A \) in \( CT_{C\Sigma} \).

Let \( g : V \to A \) be a solution of \( E \) in \( A \). For all \( s \in S, x \in V_s \) with \( E(x) = ct \),

\[
in^{A_g}_s(x) = g(x) = g^*(E(x)) = g^*(ct) = c^A(g^*(t)),
\]

and thus for all \( S \)-sorted sets \( V' \) of variables and \( h : V' \to A_g \).
Constructing recursive from iterative equations

\[ h^*(d_s(in_s x)) = d_{Ag}^A(inpAg(x)) \]

\[ \overset{(13)}{=} d_A^A(c^A(g^*(t))) \]

\[ \overset{(6)}{=} \iota_c(g^*(t)) = \iota_c((inpAg)^*(t)) \]

\[ \overset{(12)}{=} \iota_c(foldA^Ag(\sigma^*_V(t))) = \iota_c(h^*(\sigma^*_V(t))) = h^*(\iota_c(\sigma^*_V(t))) \]

Hence \( A_g \) satisfies \( \text{rec}(E) \), i.e.,

\[ \text{rec}(E) \text{ simulates } E \text{ in } A. \]

Since \( A \) is final in \( Alg_{co\Sigma} \), (4) and (11) imply that \( E \) has a unique solution in \( A \).

Example 9 \( \Psi = (Reg(CS), D\Sigma) \)

Let \( G = (S, BS, Z, R) \) be a non-left-recursive context-free grammar (i.e., there are no derivations of the form \( s \xrightarrow{+} G sw \)), \( CS = BS \cup \{ \{z\} | z \in Z \} \) and \( \text{reduce} \) be a function that simplifies regular expressions by applying semiring axioms.

Then for all \( s \in S \) there are \( k_s, n_s > 0, C_{s,1}, \ldots, C_{s,n_s} \in CS \) and \( \text{Reg}(CS) \)-terms \( t_{s,1}, \ldots, t_{s,n_s} \) over \( S \) such that

\[ (reduce \circ E_G^*)^{k_s}(s) = \text{par}(\text{seq}(\overline{C_{s,1}}, t_{s,1}), \ldots, \text{seq}(\overline{C_{s,n_s}}, t_{s,n_s})) \]

or

\[ \text{(14)} \]
\[(reduce \circ E^*_G)^k(s) = par(seq(\overline{C_{s,1}}, t_{s,1}), \ldots, seq(\overline{C_{s,n_s}}, t_{s,n_s}), \text{eps}).\]  
\[(15)\]

\(S_{\text{eps}}\) denotes the set of all \(s \in S\) such that case (15) holds true.

Let \(\text{Reg}(CS)'\) be the extension of \(\text{Reg}(CS)\) by the \(S\) of sorts of \(G\) as a further base set and the constructor \(\text{in} =_{\text{def}} \text{in}_{\text{reg}} : S \rightarrow \text{reg}\) as a further operation.

Let \(D\Sigma\) be defined as in Example 5, \(\Psi_S = (\text{Reg}(CS)', D\Sigma)\) and \(\Sigma = \text{Reg}(CS)' \cup D\Sigma\).

Using the notations of (14) and (15), we obtain the following system of recursive \(\Psi_S\)-equations:

\[
\text{rec}(E_G) = \{\delta(\text{in}(s)) = \lambda x.\sigma^*_S(\text{par}(\text{ite}(\chi(C_{s,1})(x), t_{s,1}, mt), \ldots, \\
\text{ite}(\chi(C_{s,n_s})(x), t_{s,n_s}, mt))) | s \in S\} \cup \\
\{\beta(\text{in}(s)) = 1 | s \in S_{\text{eps}}\} \cup \\
\{\beta(\text{in}(s)) = 0 | s \in S \setminus S_{\text{eps}}\}
\]

Let \(X = \bigcup CS\). By Example 5, the system \(\text{BRE}\) of recursive \(\Psi\)-equations has a unique coinductive solution \(A\) in \(\text{Pow}(X)\).
Let $g : S \to A$ be a solution of $E_G$ in $A$. For all $n \in \mathbb{N}$,

$$g^* = g^* \circ (\text{reduce} \circ E^*)^n.$$  \hspace{1cm} (16)

Let $h : V \to A_g$. Hence for all $s \in S$,

$$h^*(\text{in}(s)) = \text{in}^{A_g}(s) = g(s) = g^*(s) \overset{(16)}{=} g^*((\text{reduce} \circ E^*_G)^{k_s}(s))$$  \hspace{1cm} (17)

By (12),

$$g^* = (\text{in}^{A_g})^* = \text{fold}^{A_g} \circ \sigma^*_S : T_{\text{Reg}(CS)}(S) \to A.$$  \hspace{1cm} (18)

Hence for all $s \in S \setminus S_{\text{eps}}$,

$$h^*(\delta(\text{in}(s))) = \delta^A(h^*(\text{in}(s))) \overset{(17)}{=} \delta^A(g^*((\text{reduce} \circ E^*_G)^{k_s}(s))) = \ldots$$

$$= \delta^A(\bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i}))) = \lambda x.\delta^A(\bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i}))(x))$$

$\overset{\text{Def.} \delta^A}{=} \lambda x.\{w \in X^* \mid xw \in \bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i}))\} = \ldots$

$$= g^*(\lambda x.\text{par}(\text{ite}(\chi(C_{s,1})(x), t_{s,1}, mt), \ldots, \text{ite}(\chi(C_{s,n_s})(x), t_{s,n_s}, mt)))$$

$\overset{(18)}{=} \text{fold}^{A_g}(\sigma^*_S(\lambda x.\text{par}(\text{ite}(\chi(C_{s,1})(x), t_{s,1}, mt), \ldots, \text{ite}(\chi(C_{s,n_s})(x), t_{s,n_s}, mt))))$

$$= h^*(\sigma^*_S(\lambda x.\text{par}(\text{ite}(\chi(C_{s,1})(x), t_{s,1}, mt), \ldots, \text{ite}(\chi(C_{s,n_s})(x), t_{s,n_s}, mt))))$$

and
Constructing recursive from iterative equations

\[ h^*(\beta(in(s))) = \beta^A(h^*(in(s))) = \beta^A(g^*((reduce \circ E^*_G)^{k_s}(s))) = \ldots \]
\[ = \beta^A(\bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i}))) \ \text{Def. } \beta^A 0 = h^*(0), \]

and for all \( s \in S_{eps} \),

\[ h^*(\delta(in(s))) = \delta^A(h^*(in(s))) = \delta^A(g^*((reduce \circ E^*_G)^{k_s}(s))) = \ldots \]
\[ = \delta^A(\bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i})) \cup \{\epsilon\}) = \lambda x.\delta^A(\bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i})) \cup \{\epsilon\})(x) \]
\[ = \lambda x.\{w \in X^* \mid xw \in \bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i})) \cup \{\epsilon\}\} = \ldots \]
\[ = g^*(\lambda x.\text{par}(\text{ite}(\chi(C_{s,1})(x), t_{s,1}, mt), \ldots, \text{ite}(\chi(C_{s,n_s})(x), t_{s,n_s}, mt))) \]
\[ = h^*(\sigma^*_S(\lambda x.\text{par}(\text{ite}(\chi(C_{s,1})(x), t_{s,1}, mt), \ldots, \text{ite}(\chi(C_{s,n_s})(x), t_{s,n_s}, mt))) \]

and

\[ h^*(\beta(in(s))) = \beta^A(h^*(in(s))) = \beta^A(g^*((reduce \circ E^*_G)^{k_s}(s))) = \ldots \]
\[ = \beta^A(\bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i})) \cup \{\epsilon\}) \ \text{Def. } \beta^A 1 = h^*(1). \]

Hence \( A_g \) satisfies \( \text{rec}(E_G) \), i.e.,

\[ \text{rec}(E_G) \text{ simulates } E_G \text{ in } A. \] (19)
Constructing recursive from iterative equations

\[(10) \land (11) \land (19) \Rightarrow \text{sol}_G \text{ is the only solution of } E_G \text{ in } A.\]

\(\text{rec}(E_G)\) suggests the following extension of \(\text{Bro}(CS)\) to a \(\text{Reg}(CS)'\)-Algebra \(\text{Bro}(CS)\):

For all \(s \in S\),

\[
\delta^{\text{Bro}(CS)'}(\text{in}(s)) = \lambda x.\sigma^*_S(\text{par}(\text{ite}(x \in C_{s,1}, t_{s,1}, mt), \ldots, \text{ite}(x \in C_{s,n_s, t_{s,n_s}, mt}))),
\]

\[
\beta^{\text{Bro}(CS)'}(\text{in}(s)) = \text{if } s \in S_{\text{eps}} \text{ then } 1 \text{ else } 0.
\]

Let \(\text{Lang}(X)' = A_{\text{sol}_G}|_{\text{Reg}(CS)'}\) and \(\Sigma = \text{Reg}(CS)' \cup D\Sigma\).

\(\text{Bro}(CS)\)' agrees with the \(\Sigma\)-algebra \(T_{\text{Reg}(CS)'}\) (see (2)). Hence

\[
\text{fold}^{\text{Lang}(X)'} = \text{unfold}^{\text{Bro}(CS)'} : \text{Bro}(CS)' \rightarrow \text{Pow}(X)
\]

and thus \(\text{fold}^{\text{Lang}(X)'}\) is \(\text{Acc}(X)\)-homomorphic. Hence for all \(s \in S\),

\[
\text{unfold}^{\text{Bro}(CS)'}(\text{in}(s)) = \text{fold}^{\text{Lang}(X)'}(\text{in}(s)) = \text{in}^{\text{Lang}(X)'}(s)
\]

\[
= \text{in}^{A_{\text{sol}_G}}(s) = \text{sol}_G(s) = L(G)s,
\]

i.e., \((\text{Bro}(CS)', \text{in}(s))\) realizes the characteristic function of the language \(L(G)s\) of words over \(X\) that are derivable from \(s\) via the rules of \(G\).
(Co-)Horn Logic

(Co-)Horn clauses

Let $\Sigma = (S, BS, BF, F, P)$ and $\Sigma' = (S, BS, BF, F, P \cup P')$ be signatures and $C$ be a $\Sigma$-algebra.

$Alg_{\Sigma', C}$ denotes the full subcategory of $Alg_\Sigma$ consisting of all $\Sigma'$-algebras $A$ with $A|_{\Sigma} = C$.

$Alg_{\Sigma', C}$ is a complete lattice: For all $A, B \in Alg_{\Sigma', C}$,

$$A \leq B \iff_{\text{def}} \forall p \in P': p^A \subseteq p^B.$$ 

For all $A \subseteq Alg_{\Sigma', C}$ and $p : e \in P'$,

$$p^\perp = \emptyset, \quad p^\top = A_e, \quad p^{\cup A} = \bigcup_{A \in A} p^A \quad \text{and} \quad p^{\cap A} = \bigcap_{A \in A} p^A.$$
A $\Sigma'$-formula $\varphi$ is **negation-free w.r.t.** $\Sigma$ if $\varphi$ does not contain $\Rightarrow$, $\Leftarrow$ or $\iff$ and all subformulas of $\varphi$ with a leading negation symbol belong to $F_{0\Sigma}(V)$.

A **Horn clause** for $P'$ is a $\Sigma'$-formula $p(t) \Leftarrow \varphi$ such that $p \in P'$ and $\varphi$ is negation-free w.r.t. $\Sigma$.

Let $AX$ be a set of Horn clauses for $P'$.

The **$AX$-step function** $\Phi : Alg_{\Sigma',C} \rightarrow Alg_{\Sigma',C}$ is defined as follows:

For all $A \in Alg_{\Sigma',C}$ and $p \in P'$,

$$p^{\Phi(A)} \overset{\text{def}}{=} \{ g^*(t) \mid p(t) \Leftarrow \varphi \in AX, \; g \in \varphi^A \}.$$  

$\Phi$ is monotone and thus by the Fixpoint Theorem of Knaster and Tarski, $\Phi$ has the least fixpoint

$$\text{lfp}(\Phi) = \cap \{ A \in Alg_{\Sigma',C} \mid \Phi(A) \leq A \}.$$  

Consequently,

$$\text{lfp}(\phi) \models p(x) \iff \bigvee_{p(t) \Leftarrow \varphi \in AX} \exists \; \text{var}(t, \varphi) : (x = t \land \varphi).$$
A co-Horn clause for $P'$ is a $\Sigma'$-formula $p(t) \Rightarrow \varphi$ such that $p \in P'$ and $\varphi$ is negation-free w.r.t. $\Sigma$.

Let $AX$ be a set of co-Horn clauses for $P'$.

The $AX$-step function $\Phi : Alg_{\Sigma',C} \rightarrow Alg_{\Sigma',C}$ is defined as follows:

For all $A \in Alg_{\Sigma',C}$ and $p : e \in P'$,

$$p^{\Phi(A)} =_{def} C_e \backslash \{g^*(t) \mid pt \Rightarrow \varphi \in AX, \ g \in C^V \backslash \varphi^A\}.$$ 

$\Phi$ is monotone and thus by the Fixpoint Theorem of Knaster and Tarski, $\Phi$ has the greatest fixpoint

$$gfp(\Phi) = \bigsqcup \{A \in Alg_{\Sigma',C} \mid A \leq \Phi(A)\}.$$ 

Consequently,

$$gfp(\phi) \models p(x) \iff \bigwedge_{p(t) \Rightarrow \varphi \in AX} \forall \text{var}(t, \varphi) : (x \neq t \lor \varphi).$$

*** to be continued ***