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

Peter Padawitz
TU Dortmund, Germany

June 2, 2016

(actual version: http://fldit-www.cs.uni-dortmund.de/∼peter/IFIP2014.pdf)

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.

Contents

Syntax

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 $r_s \subseteq A_{1,s} \times \ldots \times A_{n,s}$ 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** $f : A \to B$ 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 $e \in \mathbb{T}(S, BS)$, $word(e), bag(e), set(e) \in \mathbb{T}(S, BS)$. (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 $e' \notin BS$, $e^{e'} \in \mathbb{T}(S, BS)$. (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 $f : e \to e' \in F$, $dom(f) = e$ resp. $ran(f) = e'$ is the **domain** resp. **range** of f.

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

Given signatures Σ and Σ', $\Sigma \cup \Sigma'$ denotes the componentwise union of Σ and Σ'.

$f \in F$ is a **constructor** if there are flat types $e_1, \dots, e_n$ over S and BS such that $dom(f) = e_1 \times \cdots \times e_n$ and $ran(f) \in S$.

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

Σ 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 ⇝ natural numbers

$$S = \{nat\}, \quad BS = \emptyset, \quad F = \{ \begin{array}{l} zero : 1 \to nat, \\ succ : nat \to nat \end{array} \}.$$

List(X) ⇝ finite sequences of elements of X

$$S = \{list\}, \quad BS = \{X\}, \quad F = \{ \begin{array}{l} nil : 1 \to list, \\ cons : X \times list \to list \end{array} \}.$$

$Reg(CS)$ ⇝ regular expressions over CS and regular languages over $X = \bigcup CS$

$$\begin{aligned} S = \{reg\}, \quad BS = \emptyset, \quad F \;=\; \{\; & eps : 1 \to reg, \\ & mt : 1 \to reg, \\ & par : reg \times reg \to reg, && \text{(parallel composition)} \\ & seq : reg \times reg \to reg, && \text{(sequential composition)} \\ & iter : reg \to reg \;\} \cup && \text{(iteration)} \\ \{\; & \overline{C} : 1 \to reg \mid C \in CS \;\} \end{aligned}$$

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

Destructive signatures

Let X and Y be sets of constants.

$coNat$ ⇝ natural numbers with infinity

$$S = \{nat\}, \quad BS = \emptyset, \quad F = \{pred : nat \to 1 + nat\}.$$

$coList(X)$ ⇝ finite or infinite sequences of elements of X $(coList(1) \mathrel{\widehat{=}} coNat)$

$$S = \{list, pair\}, \quad BS = \{X\}, \quad F = \{ \begin{array}{l} split : list \to 1 + pair, \\ first : pair \to X, \\ rest : pair \to list \ \}. \end{array}$$

$DAut(X, Y)$ ⇝ deterministic Moore automata with input from X and output in Y

$$S = \{state\}, \quad BS = \{X, Y\}, \quad F = \{ \begin{array}{l} \delta : state \to state^X, \\ \beta : state \to Y \ \}. \end{array}$$

$Acc(X) \mathrel{\widehat{=}} DAut(X, 2)$ ⇝ deterministic acceptors of subsets of X^*

$$S = \{reg\}, \quad BS = \{X, 2\}, \quad F = \{ \begin{array}{l} \delta : reg \to reg^X, \\ \beta : reg \to 2 \ \}. \end{array}$$

$Stream(X) \mathrel{\widehat{=}} DAut(1, X)$ ⇝ streams over X

$$S = \{list\}, \quad BS = \{X\}, \quad F = \{ \begin{array}{l} head : list \to X, \\ tail : list \to list \ \}. \end{array}$$

Let V be a $\mathbb{T}(S,BS)$-sorted set of variables.

The $\mathbb{T}(S,BS)$-sorted set $T_\Sigma(V)$ of **Σ-terms over** V

is inductively defined as follows:

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

- For all $e \in \mathbb{T}(S, BS)$, $t \in T_\Sigma(V)_2$ and $u, v \in T_\Sigma(V)_e$, $ite(t, u, v) \in T_\Sigma(V)_e$.

A Σ-term t that does not contain variables or ite, then t is called **ground**.

T_Σ denotes the set of ground Σ-terms.

The set $Fo_\Sigma(V)$ of **Σ-formulas over V**

is inductively defined as follows:

- $True, False \in Fo_\Sigma(V)$.
- For all $p : e \in P$ and $t \in T_\Sigma(V)_e$, $pt \in Fo_\Sigma(V)$. (Σ-atoms over V)
- For all $e \in \mathbb{T}(S, BS)$ and $t, u \in T_\Sigma(V)_e$, $t =_e u \in Fo_\Sigma(V)$. (Σ-equations over V)
- For all $\varphi \in Fo_\Sigma(V)$, $\neg\varphi \in Fo_\Sigma(V)$.
- For all $\varphi, \psi \in Fo_\Sigma(V)$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$, $\varphi \Leftarrow \psi$, $\varphi \Leftrightarrow \psi \in Fo_\Sigma(V)$.
- For all $x \in V$ and $\varphi \in Fo_\Sigma(V)$, $\forall x\varphi$, $\exists x\varphi \in Fo_\Sigma(V)$.

Semantics

$[0] =_{def} \emptyset$ and for all $n > 0$, $[n] =_{def} \{1, \dots, n\}$.

For all $f : A \to B$, $f^* : A^* \to B^*$ is defined as follows:
$f^*(\epsilon) = \epsilon$ and for all $n > 0$ and $(a_1, \dots, a_n) \in A^n$, $f^*(a_1, \dots, a_n) = (f(a_1), \dots, f(a_n))$.

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

$$
\begin{aligned}
a =_{word} b &\Leftrightarrow_{def} a = b. \\
a =_{bag} b &\Leftrightarrow_{def} \exists\, f : [n] \xrightarrow{\sim} [n] : (a_1, \dots, a_n) = (b_{f(1)}, \dots, b_{f(n)}), \\
&\qquad \text{i.e., } b \text{ is a permutation of } a. \\
a =_{set} b &\Leftrightarrow_{def} \{a_1, \dots, a_m\} = \{b_1, \dots, b_n\}.
\end{aligned}
$$

Let $h : A \to B$.

$\mathcal{B}_{fin}(A) =_{def} A/{=_{bag}}$ and $\mathcal{B}_{fin}(h) : \mathcal{B}_{fin}(A) \to \mathcal{B}_{fin}(B)$ maps $[a]_{=_{bag}}$ to $[h^*(a)]_{=_{bag}}$.

$\mathcal{P}_{fin}(A) = \{C \subseteq A \mid |A| < \omega\}$ and $\mathcal{P}_{fin}(h) : \mathcal{P}_{fin}(A) \to \mathcal{P}_{fin}(B)$ maps C to $\{f(a) \;\; a \in C\}$.

Predicate lifting

For alle $e \in \mathbb{T}_1(S, BS)$, 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 $e, e_1, \dots, e_n \in \mathbb{T}_1(S, BS)$,

$$
\begin{array}{lll}
F_s(A) = A_s, & F_s(h) = h_s, & \text{(projection functor)} \\
F_X(A) = X, & F_X(h) = id_X, & \text{(constant functor)} \\
F_{e_1+\dots+e_n}(A) = F_{e_1}(A) + \dots + F_{e_n}(A), & F_{e_1+\dots+e_n}(h) = F_{e_1}(h) + \dots + F_{e_n}(h), & \\
F_{e_1\times\dots\times e_n}(A) = F_{e_1}(A) \times \ldots \times F_{e_n}(A), & F_{e_1\times\dots\times e_n}(h) = F_{e_1}(h) \times \ldots \times F_{e_n}(h), & \\
F_{word(e)}(A) = F_e(A)^*, & F_{word(e)}(h) = F_e(h)^*, & \\
F_{bag(e)}(A) = \mathcal{B}_{fin}(F_e(A)), & F_{bag(e)}(h) = \mathcal{B}_{fin}(F_e(h)), & \\
F_{set(e)}(A) = \mathcal{P}_{fin}(F_e(A)), & F_{set(e)}(h) = \mathcal{P}_{fin}(F_e(h)), & \\
F_{e^X}(A) = F_e(A)^X, & F_{e^X}(h) = F_e(h)^X. &
\end{array}
$$

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 $\mathbb{T}_1(S, BS)$-sorted relation inductively as follows:

Let $s \in S$, $e_1, \ldots, e_n, e \in \mathbb{T}_1(S, BS)$ and $X \in BS$.

$$
\begin{aligned}
R_X &= \Delta_X, \\
R_{e_1+\cdots+e_n} &= \{((a,i),(b,i)) \in (\textstyle\coprod_{i=1}^n A_{e_i}) \times \coprod_{i=1}^n B_{e_i} \ \mid \ (a,b) \in R_{e_i},\ 1 \le i \le n\}, \\
R_{e_1\times\cdots\times e_n} &= \{((a_1,\ldots,a_n),(b_1,\ldots,b_n)) \in (\textstyle\prod_{i=1}^n A_{e_i}) \times \prod_{i=1}^n B_{e_i} \\
&\qquad \mid \forall\, 1 \le i \le n : (a_i,b_i) \in R_{e_i}\}, \\
R_{word(e)} &= \textstyle\bigcup_{n\in\mathbb{N}}\{((a_1,\ldots,a_n),(b_1,\ldots,b_n)) \in A_e^* \times B_e^* \\
&\qquad \mid \forall\, 1 \le i \le n : (a_i,b_i) \in R_e\}, \\
R_{bag(e)} &= \textstyle\bigcup_{n\in\mathbb{N}}\{([(a_1,\ldots,a_n)]_{=_{bag}},[(b_1,\ldots,b_n)]_{=_{bag}}) \in \mathcal{B}_{fin}(A_e) \times \mathcal{B}_{fin}(B_e) \\
&\qquad \mid \forall\, 1 \le i \le n : (a_i,b_i) \in R_e\}, \\
R_{set(e)} &= \{(C,D) \in \mathcal{P}_{fin}(A_e) \times \mathcal{P}_{fin}(B_e) \ \mid \ \forall\, c \in C\ \exists\, d \in D : (c,d) \in R_e, \\
&\qquad \forall\, d \in D\ \exists\, c \in C : (c,d) \in R_e\}, \\
R_{e^X} &= \{(f,g) \mid \forall\, x \in X : (f(x),g(x)) \in R_e\}.
\end{aligned}
$$

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

A Σ-**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 Σ have first-order domains and ranges. Let A, B be Σ-algebras.

An S-sorted function $h : A \to B$ is a Σ-**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_Σ denotes the category of Σ-algebras and Σ-homomorphisms.

⇴ A Σ-homomorphism h is iso in Alg_Σ 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 Alg_Σ to Set^S.

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

$$\begin{array}{ccc} A_e & \xrightarrow{f^A} & A_{e'} \\ {\scriptstyle h_e}\downarrow & & \downarrow{\scriptstyle h_{e'}} \\ B_e & \xrightarrow{f^B} & B_{e'} \end{array}$$

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 $\alpha : A \to F(A)$.

Alg_F and $coAlg_F$ denote the categories of F-algebras resp. F-coalgebras where

- an 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 $coAlg_F$-**morphism** from $\alpha : A : F(A)$ to $\beta : B \to F(B)$ is a $\mathcal{K}$-morphism $h : A \to B$ with $F(h) \circ \alpha = \beta \circ h$.

A constructive signature $\Sigma = (S, BS, BF, F, P)$ induces a functor

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

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

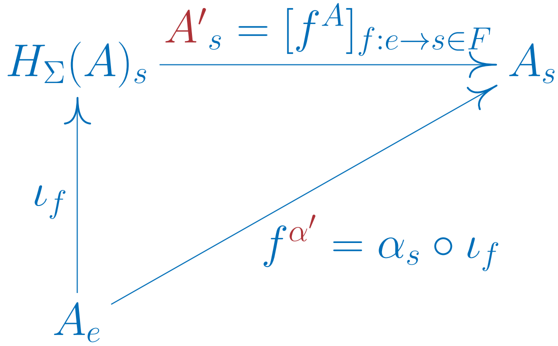
$$\begin{aligned} H_\Sigma(A)_s &= \coprod_{f:e \to s \in F} A_e, \\ H_\Sigma(h)_s &= \coprod_{f:e \to s \in F} h_e. \end{aligned}$$

Alg_Σ and Alg_{H_Σ} are equivalent categories:

Let $A \in Alg_\Sigma$ and $\alpha : A \to H_\Sigma(A) \in Alg_{H_\Sigma}$.

The H_Σ-algebra $A' : A \to H_\Sigma(A)$ and the Σ-algebra α' are defined as follows:

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



Examples

$$
\begin{aligned}
H_{Nat}(A)_{nat} &= 1 + A_{nat}, \\
H_{List(X)}(A)_{list} &= 1 + (X \times A_{list}), \\
H_{Reg(CS)}(A)_{reg} &= 1 + 1 + CS + A_{reg}^2 + A_{reg}^2 + A_{reg}.
\end{aligned}
$$

$h : A \to B$ is a Σ-homomorphism $\iff$ h is an Alg_{H_Σ}-morphism from $\alpha(A)$ to $\alpha(B)$:

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

$h : \alpha \to \beta$ is an Alg_{H_Σ}-morphism $\iff$ h is a Σ-homomorphism from $A(\alpha)$ to $A(\beta)$:

$$\begin{array}{ccc} H_\Sigma(A)_s & \xrightarrow{\alpha_s} & A_s \\ {\scriptstyle H_\Sigma(h)_s}\downarrow & & \downarrow{\scriptstyle h_s} \\ H_\Sigma(B)_s & \xrightarrow[\beta_s]{} & B_s \end{array} \quad\iff\quad \begin{array}{ccc} A_e & \xrightarrow{f^{A(\alpha)}} & A_s \\ {\scriptstyle h_e}\downarrow & & \downarrow{\scriptstyle h_s} \\ B_e & \xrightarrow[f^{A(\beta)}]{} & B_s \end{array}$$

A destructive signature $\Sigma = (S, BS, BF, F, P)$ induces a functor

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

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

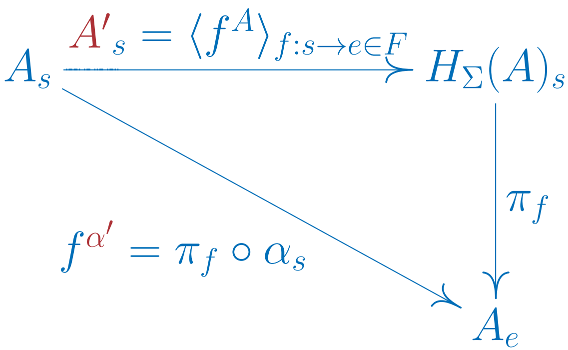
$$H_\Sigma(A)_s = \prod_{f:s\to e\in F} A_e,$$
$$H_\Sigma(h)_s = \prod_{f:s\to e\in F} h_e.$$

Alg_Σ and $coAlg_{H_\Sigma}$ are equivalent categories:

Let $A \in Alg_\Sigma$ and $\alpha : H_\Sigma(A) \to A \in coAlg_{H_\Sigma}$.

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

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



Examples

$$
\begin{aligned}
H_{coNat}(A)_{nat} &= 1 + A_{nat}, \\
H_{coList(X)}(A)_{list} &= 1 + (X \times A_{list}), \\
H_{DAut(X,Y)}(A)_{state} &= A_{state}^{X} \times Y.
\end{aligned}
$$

Haskell implementation of Alg_Σ

Let $\Sigma = (S, BS, \emptyset, F, \emptyset)$ be a signature,
$BS = \{X_1, \dots, X_k\}$, $S = \{s_1, \dots, s_m\}$ and $F = \{f_1 : e_1 \to e_1', \dots, f_n : e_n \to e_n'\}$.

Each Σ-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,
                            iter :: 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 $\mathbb{T}(S,BS)$-sorted set of variables, A be a Σ-algebra and A^V be the set of **valuations of V in A**, i.e., $\mathbb{T}(S,BS)$-sorted functions from V to A.

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

$$g[a/x](z) =_{def} \begin{cases} a & \text{if } z=x, \\ g(z) & \text{otherwise.} \end{cases}$$

The $\mathbb{T}(S, BS)$-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 $x \in X \in \cup BS$, $g^*(x) = x$.
- For all $n > 1$, $e_1, \dots, e_n \in \mathbb{T}(S, BS)$, $t = (t_1, \dots, t_n) \in T_\Sigma(V)_{e_1 \times \dots \times e_n}$ and $1 \leq i \leq n$, $g^*(\pi_i t) = g^*(t_i)$.
- For all $n > 1$, $e_1, \dots, e_n \in \mathbb{T}(S, BS)$, $1 \leq i \leq n$ and $t \in T_\Sigma(V)_{e_i}$, $g^*(\iota_i t) = (g^*(t), i)$.
- For all $n \in \mathbb{N}$ and $t_1, \dots, t_n \in T_\Sigma(V)$, $g^*(t_1, \dots, t_n) = (g^*(t_1), \dots, g^*(t_n))$.
- For all $f : e \to e' \in F$ and $t \in T_\Sigma(V)_e$, $g^*(f(t)) = f^A(g^*(t))$.
- For all $c \in \{word, bag, set\}$, $c(t) \in T_\Sigma(V)_{c(e)}$, $g^*(c(t)) = [g^*(t)]_{=_c}$.
- For all $n > 0$, $e_1, \dots, e_n, e \in \mathbb{T}(S, BS)$, $x \in V_{e_1} \cup \dots \cup V_{e_n}$, $t_i \in T_\Sigma(V)_e$, $1 \leq i \leq n$, and $(a, i) \in A_{e_1 + \dots + e_n}$,

$$g^*(\lambda x.(t_1 | \dots | t_n))(a, i) = g[a/x]^*(t_i).$$

- For all $e, e' \in \mathbb{T}(S, BS)$, $t \in T_\Sigma(V)_{e^{e'}}$ 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 $u, v \in T_\Sigma(V)_e$,

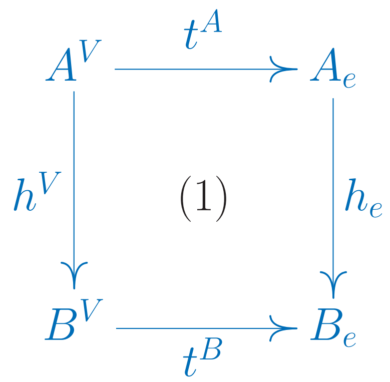
$$g^*(ite(t,u,v)) = \begin{cases} g^*(u) & \text{if } g^*(t) = 1, \\ g^*(v) & \text{otherwise.} \end{cases}$$

A Σ-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 Σ-terms t, we define:

$$\begin{aligned} t^A : A^V &\rightarrow A_e \\ g &\mapsto g^*(t) \end{aligned}$$

$\overline{t} : \_^V \rightarrow F_e U_S$ with $\overline{t}_A =_{def} t^A$ for all $A \in Alg_\Sigma$ is a natural transformation:



(1) is equivalent to the *Substitution Lemma*:

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

$$(h \circ g)^*(t) = (h \circ g^*)(t). \tag{2}$$

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

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

$$\begin{array}{lcl}
True^A & = & A^V, \\
False^A & = & \emptyset, \\
p(t)^A & = & \{g \in A^V \mid g^*(t) \in p^A\}, \\
(\neg\varphi)^A & = & A^V \setminus \varphi^A, \\
(\varphi \wedge \psi)^A & = & \varphi^A \cap \psi^A, \\
(\varphi \vee \psi)^A & = & \varphi^A \cup \psi^A, \\
(\varphi \Rightarrow \psi)^A & = & (\psi \Leftarrow \varphi)^A = (\neg\varphi \vee \psi)^A,
\end{array}$$

$$
\begin{aligned}
(\psi \Leftrightarrow \varphi)^A &= (\varphi \Rightarrow \psi)^A \cap (\varphi \Leftarrow \psi)^A, \\
(\forall x \varphi)^A &= \{g \in A^V \mid \forall\, a \in A_s : g[a/x] \in \varphi^A\}, \\
(\exists x \varphi)^A &= \{g \in A^V \mid \exists\, a \in A_s : g[a/x] \in \varphi^A\}.
\end{aligned}
$$

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** Σ-formulas φ, $g \in A^V$ and Σ-homomorphisms $h : A \to B$,

$$g \in \varphi^A \quad \Rightarrow \quad h \circ g \in \varphi^B.$$

Initial and final algebras

An S-sorted binary relation R on A is a **Σ-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 Σ is destructive, then Σ-congruences are also called **Σ-bisimulations**.

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

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

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

A Σ-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 Σ-congruence.

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

Terms for constructive signatures

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

T_Σ is a Σ-algebra:

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

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

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

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

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

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

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

$$t_1 \sim_e t'_1 \wedge \cdots \wedge t_n \sim_e t'_n \text{ implies } bag(f(t_1), \ldots, f(t_n)) \sim_{bag(s)} bag(t'_1, \ldots, t'_n),$$

- for all $m, n > 0$, $e \in \mathbb{FT}(S, BS)$, $t_i \in T_{\Sigma,e}$, $i \in [m]$, and $t'_i \in T_{\Sigma,e}$, $1 \leq i \leq n$,

$$\forall\, 1 \leq i \leq m\ \exists\, 1 \leq j \leq n : t_i \sim_e t'_j \ \wedge\ \forall\, 1 \leq j \leq n\ \exists\, 1 \leq i \leq m : t_i \sim_e t'_j$$
$$\text{implies}\ \ set(t_1, \dots, t_m) \sim_{set(s)} set(t'_1, \dots, t'_n),$$

- for all $s \in S$, $f : e \to s \in F$ 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_Σ with $T_\Sigma/\!\sim$.

T_Σ is initial in Alg_Σ.

For all Σ-algebras A, the unique Σ-homomorphism

$$fold^A : T_\Sigma \to A$$

is defined inductively as follows:

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

$$\begin{aligned} fold^A_s(ft) &= f^A(fold^A_e(t)), \\ fold^A_{c(e')}(c(t')) &= [fold^A_{e'}(t')]_{=_c}. \end{aligned}$$

Haskell implementation of T_Σ and *fold*

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

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

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

i.e., Alg_Σ is implemented by the following datatype:

```
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_Σ:

```
data S1T x1 ... xk = C11 E11T | ... | C1n_1 E1n_1T
...
data SmT x1 ... xk = Cm1 Em1T | ... | Cmn_m Emn_mT
```

The algebra T_Σ is then defined as follows:

```
sigmaT :: Sigma x1 ... xk (S1T x1 ... xk) ... (SmT x1 ... xk)
sigmaT = Sigma C11 ... C1n_1 ... Cm1 ... Cmn_m
```

Let $1 \leq i \leq m$.

```
foldSi :: Sigma x1 ... xk s1 ... sm -> SiT x1 ... xk -> si
foldSi alg ti = case ti of Ci1 t -> ci1 alg $ foldEi1 alg t
                           ...
                           Cin_i t -> cin_i alg $ foldEin_i alg t

foldWordSi,foldBagSi,foldSetSi :: Sigma x1 ... xk s1 ... sm
                                  -> [SiT x1 ... xk] -> [si]
foldWordSi = map . foldSi
foldBagSi  = map . foldSi
foldSetSi  = 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)

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)

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

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

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

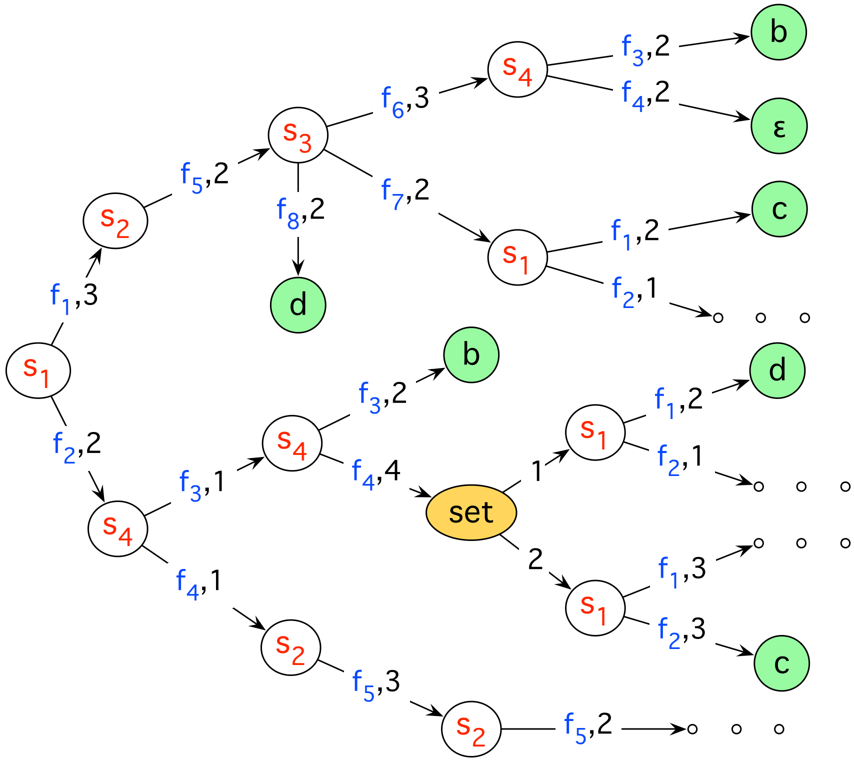
coT_Σ denotes the greatest $\mathbb{FT}(S, BS)$-sorted set of prefix closed partial functions

$$t : Lab_\Sigma^* \multimap\!\!\to 1 + \{word, bag, set\} + \cup BS$$

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 $1\le i\le n$ such that $(d,x,i)\in def(t)$, $\lambda w.t((d,x,i)w)\in coT_{\Sigma,e_i}$ and for all $(d,x,i),(d,x,j)\in def(t)$, $dom(d)=s$ and $i=j$.
- For all $c\in\{word,bag,set\}$, $s\in S\cup BS$ and $t\in coT_{\Sigma,c(s)}$, $t(\epsilon)=c$ and there is $n\in\mathbb{N}$ such that for all $1\le i\le n$, $\lambda w.t(iw)\in coT_{\Sigma,s}$, and $def(t)\cap Lab_\Sigma=[n]$.
- For all $X\in BS$, $coT_{\Sigma,X}=X$ (here identified with the set $1\to X$ of functions).

The elements of coT_Σ are called **Σ-coterms**.



A Σ-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_Σ 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 $t \sim_{word(s)} t'$, $D =_{def} def_1(t) = def_1(t')$ and for all $i \in D$, $\lambda w.t(iw) \sim_s \lambda w.t'(iw)$,
- for all $s \in S \cup BS$ and $t \sim_{bag(s)} t'$, $D =_{def} def_1(t) = def_1(t')$ and there is $f : [n] \stackrel{\sim}{\to} [n]$ such that for all $i \in D$, $\lambda w.t(iw) \sim_s \lambda w.t'(f(i)w)$,
- for all $s \in S \cup BS$, $t \sim_{set(s)} t'$ 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 $s \in S \cup BS$, $t \sim_{set(s)} t'$ 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_Σ with $coT_\Sigma/\sim$.

coT_Σ is a Σ-algebra:

For all $s \in S$, $t \in coT_{\Sigma,s}$, $d : s \to (e_1 + \cdots + e_n)^X \in F$, $x \in X$ and $w \in Lab_\Sigma^*$,

$$(d,x,i) \in def(t) \quad \Rightarrow \quad d^{coT_\Sigma}(t)(x)(w) = t((d,i,x)w).$$

Example 1

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

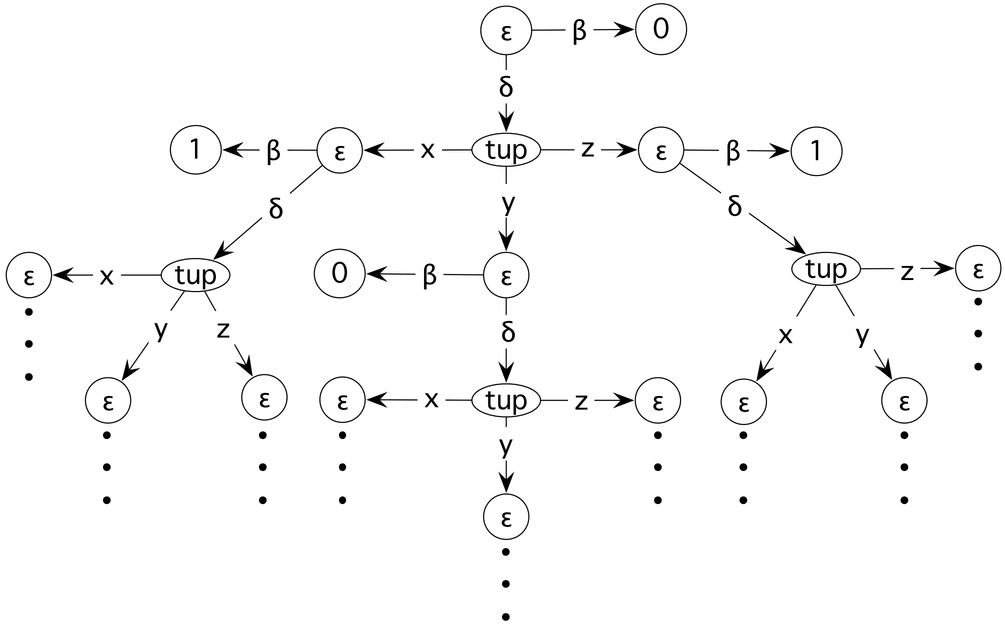
$$coT_{DAut(X,Y)} \;\cong\; 1^{L^*} \times Y^{L^*\beta} \;\cong\; Y^{L^*\beta} \;\stackrel{L^*\beta \cong X^*}{\cong}\; Y^{X^*}.$$

Hence $coT_{DAut(X,Y)}$ is $DAut(X,Y)$-isomorphic to the $DAut(X,Y)$-algebra $Beh(X,Y)$ of **behavior functions** that is defined as follows:

$$Beh(X,Y)_{state} \;=\; Y^{X^*}.$$

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

$$\delta^{Beh(X,Y)}(f)(x)(w) = f(xw) \quad \text{and} \quad \beta^{Beh(X,Y)}(f) = f(\epsilon).$$



A $DAut(\{x,y,z\},Y)$-coterm of sort state

coT_Σ is final in Alg_Σ.

For all Σ-algebras A, the unique Σ-homomorphism $unfold^A : A \to coT_\Sigma$ is defined as follows: For all $s \in \mathbb{FT}(S,BS)$, $a \in A_s$, $(d,x,i) \in Lab_\Sigma$, $w \in Lab_\Sigma^*$ and $k \in \mathbb{N}$,

$$
\begin{aligned}
unfold_s^A(a)(\epsilon) &= \epsilon, \\
unfold_s^A(a)((d,x,i)w) &= \begin{cases} unfold_{e_i}^A(b)(w) & \text{if } d : s \to (e_1 + \cdots + e_n)^X \in F \\ & \quad \text{and } d^A(a)(x) = (b,i), \\ \text{undefined} & \text{otherwise}, \end{cases} \\
unfold_s^A(a)(kw) &= \begin{cases} unfold_s^A(a_k)(w) & \text{if } \exists\, c \in \{word, bag, set\},\ e \in S \cup BS : \\ & \quad s = c(e),\ a = [(a_1,\dots,a_n)]_{=c} \\ & \quad \text{and } 1 \le k \le n, \\ \text{undefined} & \text{otherwise.} \end{cases}
\end{aligned}
$$

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 $unfoldB : A \to Beh(X,Y)$ be defined as follows:

For all $a \in A_{state}$, $x \in X$ and $w \in X^*$,

$$
\begin{aligned}
unfoldB^A(a)(\epsilon) &= \beta^A(a), \\
unfoldB^A(a)(xw) &= unfoldB^A(\delta^A(a)(x))(w).
\end{aligned}
$$

Since $unfoldB$ is $DAut(X,Y)$-homomorphic,

$$unfold^A = \xi \circ unfoldB^A.$$

Haskell implementation of coT_Σ and $unfold$

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

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

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

i.e., Alg_Σ is implemented by the following datatype:

```
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 coT_Σ:

```
data S1C x1 ... xk = S1C {d11C :: E11C | ... | d1n_1C :: E1n_1C}
...
data SmC x1 ... xk = SmC {dm1C :: Em1C | ... | dmn_mC :: Emn_mC}
```

The algebra coT_Σ is then defined as follows:

```
sigmaC :: Sigma x1 ... xk (S1C x1 ... xk) ... (SmC x1 ... xk)
sigmaC = Sigma d11C ... d1n_1C ... dm1C ... dmn_mC
```

Let $1 \leq i \leq m$.

```
unfoldSi :: Sigma x1 ... xk s1 ... sm -> si -> SiC x1 ... xk
unfoldSi alg ai =  SiC (unfoldEi1 alg $ di1 alg ai)
                       ...
                       (unfoldEin_i alg $ din_i alg ai)

unfoldWordSi,foldBagSi,foldSetSi :: Sigma x1 ... xk s1 ... sm
                                    -> [si] -> [SiT x1 ... xk]
```

```
unfoldWordSi = map . unfoldSi
unfoldBagSi  = map . unfoldSi
unfoldSetSi  = map . unfoldSi
```

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

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

unfoldE^xi :: Sigma x1 ... xk s1 ... sm -> (xi -> E) -> xi -> EC
unfoldE^xi alg f = unfoldE alg . f

data Sum_n e1 ... en = S1 e1 | ... | Sn en
```

Let $1 \leq i \leq n$.

```
unfoldE1+...+En :: Sigma x1 ... xk s1 ... sm -> Sum_n E1 ... En
                                             -> Sum_n E1C ... EnC
unfoldE1+...+En alg a = case a of S1 a -> unfoldE1 alg a
                                  ...
                                  Sn a -> unfoldEn alg a
```

Examples

```
data ConatC = ConatC {predC :: Maybe ConatC}

conatC :: Conat ConatC
conatC = Conat predC

unfoldConat :: Conat nat -> nat -> ConatC
unfoldConat alg nat = ConatC $ do nat <- pred alg nat
                                  Just $ unfoldConat alg 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 -> StateC 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 Σ-algebra A, a final Σ-algebra Fin, $a \in A$ and $f \in Fin$,

(A, a) **realizes** f iff $unfold^A(a) = f$.

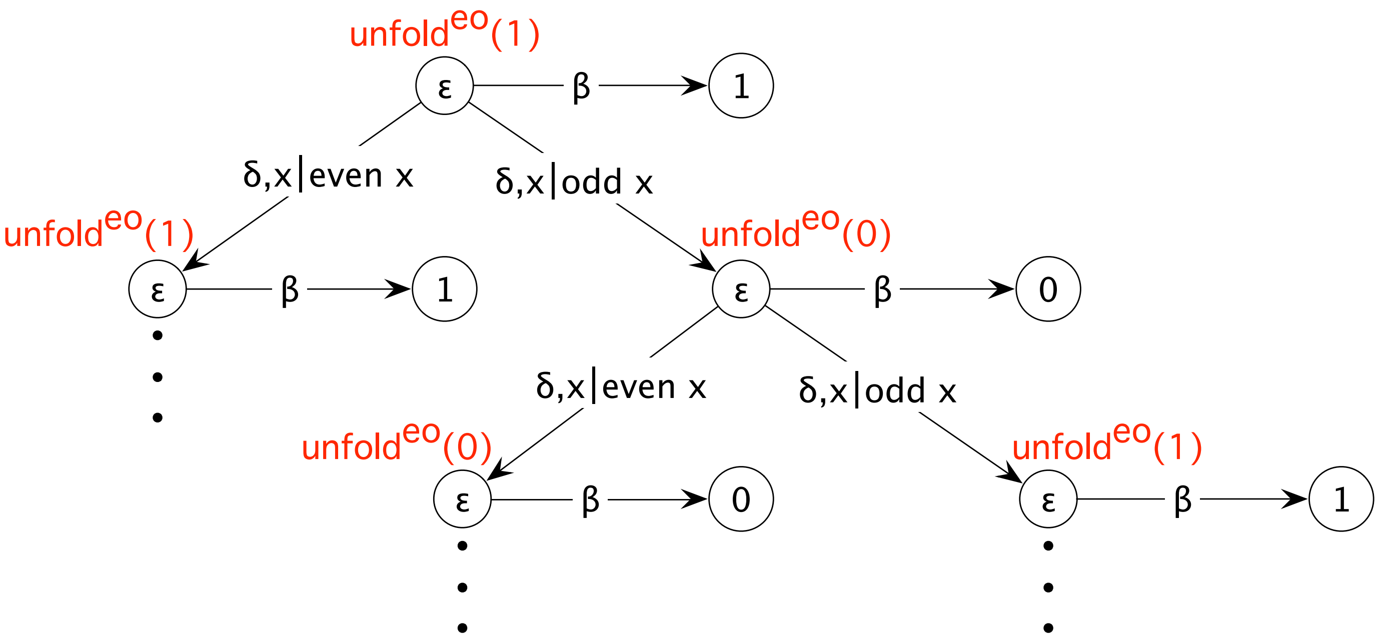
Example 3

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

```
eo :: DAut Int Bool Bool
eo = DAut (\state -> if state then even else not . even) id
```

and

$$\begin{array}{rcl} f : \mathbb{Z}^* & \to & 2 \\ (x_1, \ldots, x_n) & \mapsto & \sum_{i=1}^n x_i \textit{ is even} \end{array} \qquad \begin{array}{rcl} g : \mathbb{Z}^* & \to & 2 \\ (x_1, \ldots, x_n) & \mapsto & \sum_{i=1}^n x_i \textit{ is odd} \end{array}$$



Since $h : A \to Beh(\mathbb{Z}, 2)$ with $h(1) = f$ and $h(0) = g$ is $Acc(\mathbb{Z})$-homomorphic,

$$h = unfold^{eo}.$$

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

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

of Σ-equations is a **system of recursive Ψ-equations** if the following conditions hold true:

- For all $d \in D$ and $c \in C$, $freeVars(t_{d,c}) \subseteq \{x_1, \ldots, x_{n_c}\}$.
- 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.

  $\Rightarrow$ 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.

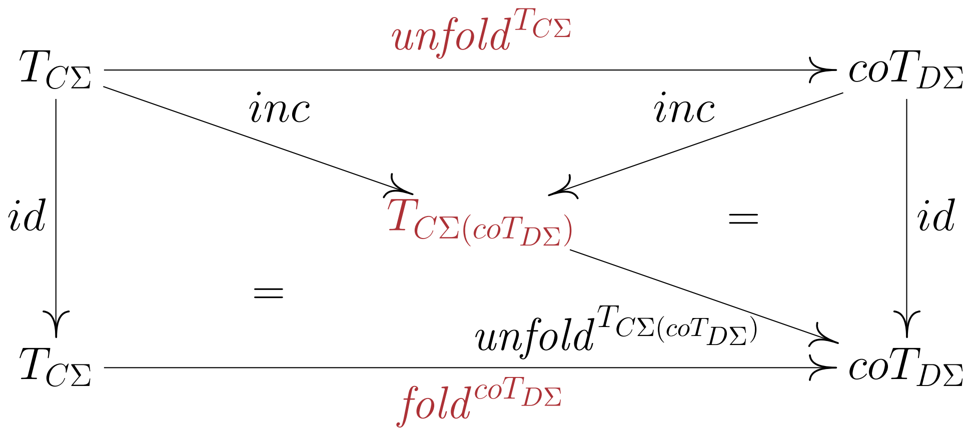
  $\Rightarrow$ no nesting of constructors of C_2, but possible nestings of destructors

Let E be a system of recursive Ψ-equations and A be a $C\Sigma$-algebra. An **inductive solution of E in A** is a Σ-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 Ψ-equations and A be a $D\Sigma$-algebra. A **coinductive solution of E in A** is a Σ-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 $fold^{coT_{D\Sigma}} = unfold^{T_{C\Sigma}}$.



Example 4

Let

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

$\Psi = (C\Sigma, Stream(X))$ and $s \in V$. The equations

$$\begin{array}{lcllcl}
head(evens(s)) & = & head(s), & tail(evens(s)) & = & evens(tail(tail(s))), \\
head(odds(s)) & = & head(tail(s)), & tail(odds(s)) & = & odds(tail(tail(s))), \\
head(exchange(s)) & = & head(tail(s)), & tail(exchange(s)) & = & exchange'(s), \\
head(exchange'(s)) & = & head(s), & tail(exchange'(s)) & = & exchange(tail(tail(s)))
\end{array}$$

form a system E of recursive Ψ-equations.

$evens(s)$ und $odds(s)$ list the elements of s at even resp. odd positions.
$exchange(s)$ exchanges the elements at even positions with those at odd positions.

(2) $\Longrightarrow$ E has a unique coinductive solution in the final $Stream(X)$-algebra.

Example 5

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

$$\begin{aligned} D\Sigma \;=\; & (\{reg\}, \{2, X\}, \\ & \{max, * : 2 \times 2 \to 2\} \cup \{\_ \in C : X \to 2 \mid C \in CS\}, \\ & \{\delta : reg \to reg^X,\ \beta : reg \to 2\}), \end{aligned}$$

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

$$\begin{aligned} \delta(eps) &= \lambda x.mt, \\ \delta(mt) &= \lambda x.mt, \\ \delta(\overline{C}) &= \lambda x.ite(\chi(C)(x), eps, mt) \\ \delta(par(t,u)) &= \lambda x.par(\delta(t)(x), \delta(u)(x)), \\ \delta(seq(t,u)) &= \lambda x.ite(\beta(t), par(seq(\delta(t)(x), u), \delta(u)(x)) \\ & \qquad\qquad seq(\delta(t)(x), u)), \\ \delta(iter(t)) &= \lambda x.seq(\delta(t)(x), iter(t)), \\ \beta(eps) &= 1, \\ \beta(mt) &= 0, \\ \beta(\overline{C}) &= 0, \end{aligned}$$

$$\begin{aligned}\beta(par(t,u)) &= max\{\beta(t),\beta(u)\},\\ \beta(seq(t,u)) &= \beta(t)*\beta(u),\\ \beta(iter(t)) &= 1.\end{aligned}$$

form the system BRE of recursive Ψ-equations.

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

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

(2) $\Longrightarrow$ 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 $x \in X$,

$$\begin{aligned}Pow(X)_{state} &= \mathcal{P}(X^*),\\ \delta^{Pow(X)}(L)(x) &= \{w \in X^* \mid xw \in L\},\\ \beta^{Pow(X)}(L) &= \begin{cases} 0 & \text{falls } \epsilon \in L,\\ 1 & \text{sonst.}\end{cases}\end{aligned}$$

$Lang(X) = B|_{Reg(CS)}$ is defined as follows:

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

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

(2) $\Longrightarrow$ $fold^{Lang(X)} = unfold^{Bro(CS)} : T_{Reg(CS)} \to \mathcal{P}(X^*)$

$\Longrightarrow$ For all $t \in T_{Reg(CS)}$, $(Bro(CS), t)$ realizes the characteristic function of the language $fold^{Lang(X)}(t)$ 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 $t \in T_{Reg(CS)}$, $\delta^{Norm(CS)}(t) =_{def} reduce \circ \delta^{Bro(CS)}(t)$. ❑

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 $a, b \in A_s$,

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

$$\left.\begin{array}{l} A|_{D\Sigma} \text{ is final in } Alg_{D\Sigma}, \\ \sim \text{ is a } D\Sigma\text{-congruence up to } C, \\ \text{there is a system of recursive } \Psi\text{-equations} \end{array}\right\} \Longrightarrow \sim_C \text{ is a } D\Sigma\text{-congruence.} \qquad (3)$$

Example 6

Let Ψ be as in Example 5 and $V = \{x, y, z\}$,

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

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

$\Longrightarrow$ Since $Pow(X)$ is final in $Alg_{Acc(X)}$, (3) implies that $\sim_C$ is $Acc(X)$-congruence.

$\Longrightarrow$ Since $Pow(X)$ satisfies the coinduction principle, $\sim \subseteq \Delta_{Pow(X)}$ and thus

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

❑

Given a bisignature Ψ, we have seen that a system E of recursive Ψ-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 λ be derived from Ψ such that $(C\Sigma \cup D\Sigma)$-algebras satisfying E correspond to **λ-bialgebras**?

With regard to their domain and range types, functions that come as inductive or coinductive solutions of systems of recursive Ψ-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 Ψ-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

$$E : V \to T_\Sigma(V)$$

with $img(E) \cap V = \emptyset$ is called a **system of iterative Σ-equations**.

Let A be a Σ-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_Σ denotes the greatest $\mathbb{FT}(S, BS)$-sorted set of prefix closed partial functions

$$t : \mathbb{N}^* \multimap\!\!\rightarrow F + \{word, bag, set\} + \cup BS$$

such that

- for all $s \in S$ and $t \in CT_{\Sigma,s}$ there are $n > 0$ and $e_1, \dots, e_n \in \mathbb{FT}(S, BS)$ with $t(\epsilon) : e_1 \times \cdots \times e_n \to s \in F$, $def(t) \cap \mathbb{N} = [n]$ and $\lambda w.t(iw) \in CT_{\Sigma,e_i}$ for all $1 \le i \le n$,

- for all $c \in \{word, bag, set\}$, $s \in S \cup BS$ and $t \in CT_{\Sigma,c(s)}$ there is $n_t \in \mathbb{N}$ with $t(\epsilon) = c$, $def(t) \cap \mathbb{N} = [n_t]$ 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 $1 \to X$ of functions).

Let $\sim$ be the greatest $\mathbb{FT}(S, BS)$-sorted equivalence relation on CT_Σ 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 $t \sim_{word(s)} t'$, $n_t = n_{t'}$ and for all $i \in [n_t]$, $\lambda w.t(iw) \sim_s \lambda w.t'(iw)$,
- for all $s \in S \cup BS$, $t \sim_{bag(s)} t'$ and $f : [n_t] \stackrel{\sim}{\to} [n_t]$, $n_t = n_{t'}$ and for all $i \in [n_t]$, $\lambda w.t(f(i)w) \sim_s \lambda w.t'(iw)$,
- for all $s \in S \cup BS$, $t \sim_{set(s)} t'$, $i \in [n_t]$ 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_Σ with $CT_\Sigma/{\sim}$.

The elements of CT_Σ are called **Σ-trees**.

CT_Σ is a Σ-algebra:

For all $f : e \to s \in F$, $t = (t_1, \dots, t_n) \in CT_{\Sigma,e}$ 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 $\Sigma_\perp = (S, BS, BF, F \cup \{\perp_s : 1 \to s \mid s \in S\})$ and $\leq$ be the least reflexive, transitive and Σ-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 $\Longrightarrow$

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

the category of ω-continuous Σ-algebras as objects and strict and ω-continuous Σ-homomorphisms.

Elgot's Theorem (see Goguen et al., *Initial Algebra Semantics and Continuous Algebras*)

Each system of iterative Σ-equations has a unique solution in CT_Σ.

Σ induces the destructive signature $co\Sigma$ with $H_\Sigma = H_{co\Sigma}$:

$$\begin{aligned} co\Sigma \;=\; (S, BS, BF, \; & \{d_s : s \to \textstyle\coprod_{f:e\to s\in F} e \mid s \in S\} \cup \\ & \{\pi_i : e_1 \times \cdots \times e_n \to e_i \mid n > 1,\; e_1,\dots,e_n \in \mathbb{FT}(S,BS), \\ & \qquad 1 \le i \le n\}) \end{aligned}$$

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

CT_Σ is a $co\Sigma$-algebra:

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

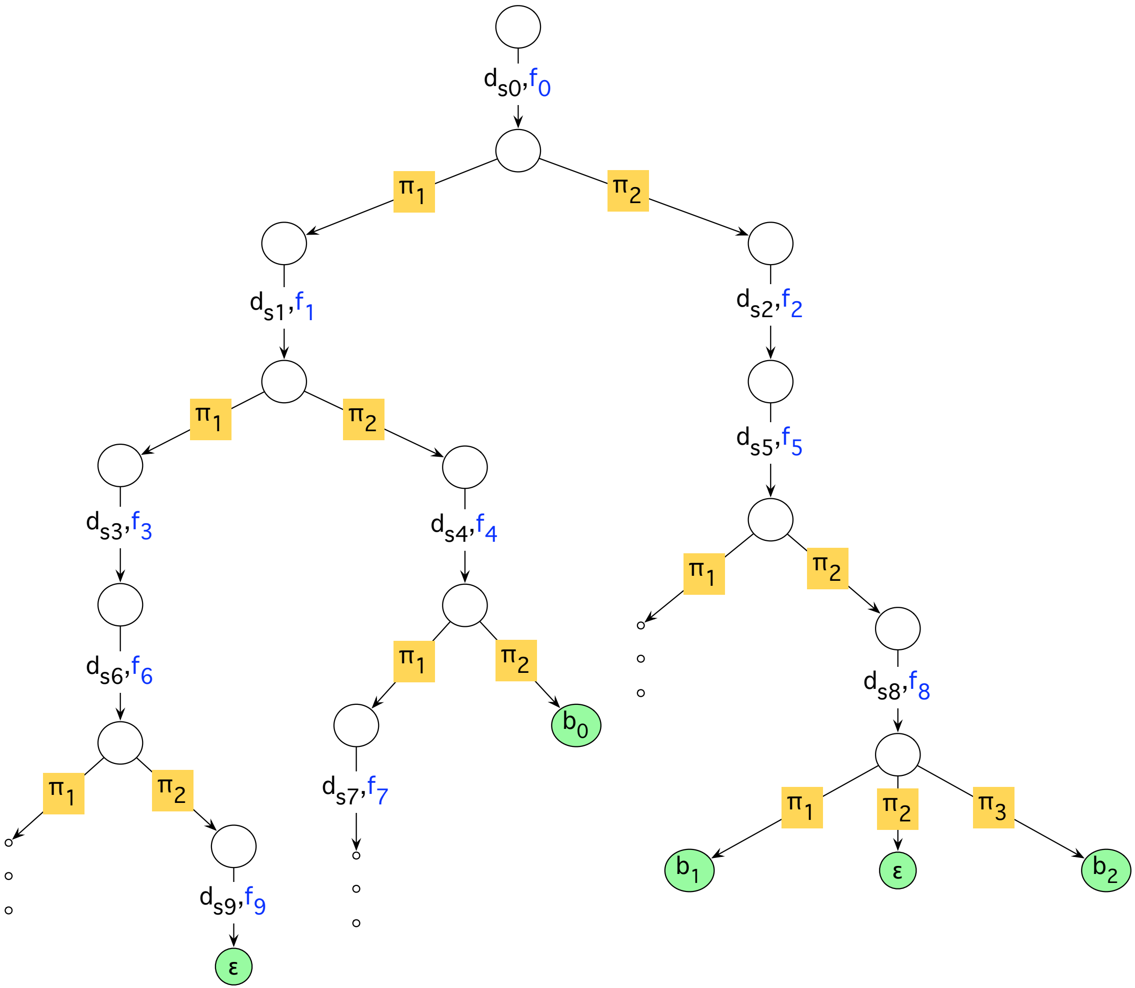
$$d_s^{CT_\Sigma}(t) \;=_{def}\; ((\lambda w.t(1w), \dots, \lambda w.t(nw)), t(\epsilon)).$$

CT_Σ is final in $Alg_{co\Sigma}$.

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

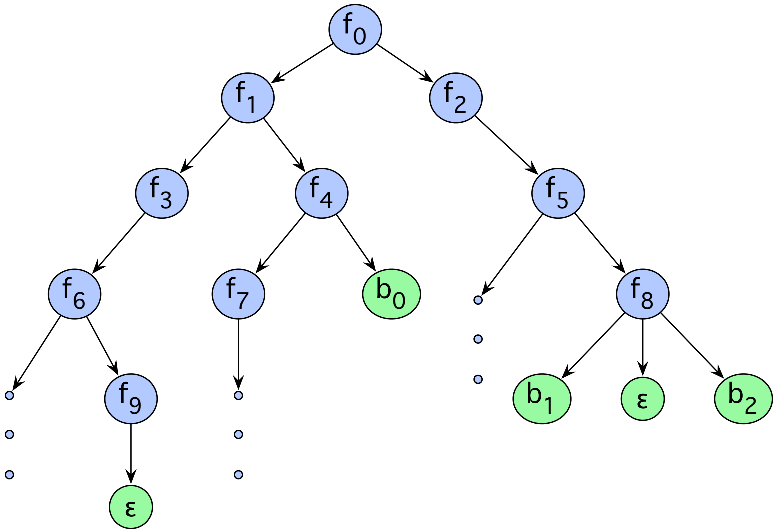
$$\begin{array}{rcl} unfold^A(a)(\epsilon) & = & f, \\ unfold^A(a)(iw) & = & \begin{cases} unfold^A(a_i)(w) & \text{if } \pi_1(d_s^A(a)) = (a_1,\dots,a_n) \wedge 1 \leq i \leq n, \\ \text{undefined} & \text{otherwise.} \end{cases} \end{array}$$

$CT_\Sigma \cong coT_{co\Sigma}$.



A coΣ-*coterm*

... and the corresponding Σ-tree:



Let $E : V \to T_\Sigma(V)$ be a system of iterative Σ-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 $x \in V_s$,

$$\begin{aligned} T^E_s &= T_\Sigma(V)_s, \\ d^{T^E}_s(ft) &= (t, f), \\ d^{T^E}_s(x) &= d^{T^E}_s(E(x)). \end{aligned}$$

$unfold^{T^E} \circ inc_V : V \to CT_\Sigma$ solves E in CT_Σ. (4)

$g : V \to CT_\Sigma$ solves E in CT_Σ iff $g^* : T^E \to CT_\Sigma$ is $co\Sigma$-homomorphic. (5)

(4) $\wedge$ (5) $\Longrightarrow$ Each system of iterative Σ-equations has a unique solution in CT_Σ.

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

Example 7 $\Psi = (\Sigma, co\Sigma)$

For all $e \in \mathbb{T}(S, BS)$, 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 Ψ-equations.

(2) $\Longrightarrow$ DC has a unique coinductive solution in CT_Σ.	(6)

Context-free grammars with base sets

A **context-free grammar** $G = (S, BS, R)$

consists of

- a set S of **sorts** (also called **nonterminals**),
- a set BS of nonempty **base sets** whose singletons are called **terminals** and are identified with their respective unique element,
- a set R of **rules** $s \to w$ with $s \in S$ and $w \in (S \cup BS)^*$.

Let Z be the set of terminals of G. The following function $typ : (S \cup BS)^* \to \mathcal{T}(S, BS)$ removes all elements of Z from words over $S \cup BS$ and translates the latter into the corresponding product types:

- $typ(\epsilon) = 1$.
- For all $s \in S \cup BS \setminus Z$ and $w \in (S \cup BS)^*$, $typ(sw) = s \times typ(w)$.
- For all $x \in Z$ and $w \in (S \cup BS)^*$, $typ(xw) = typ(w)$.

The constructive signature

$$\Sigma(G) = (S, BS, \{f_{s\to w} : typ(w) \to s \mid s \to w \in R\})$$

is called the **abstract syntax of G of G**.

Finite ground $\Sigma(G)$-terms are called **syntax trees of G**.

Let $X = \bigcup BS$.

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

- For all $s \in S$, $Word(G)_s =_{def} X^*$.
- For all $w \in Z^*$ and $r = (s \to w) \in R$, $f_r^{Word(G)}(\epsilon) =_{def} w$.
- For all $n > 0$, $w_0 \dots w_n \in Z^*$, $e_1, \dots, e_n \in S \cup BS \setminus Z$, $r = (s \to w_0 e_1 w_1 \dots e_n w_n) \in R$ and $(v_1, \dots, v_n) \in (X^*)^n$,
$$f_r^{Word(G)}(v_1, \dots, v_n) =_{def} w_0 v_1 w_1 \dots v_n w_n.$$

The **language** $L(G)$ **of G** is the set of words over X that result from folding syntax trees in $Word(G)$:

$$L(G) \ =_{def} \ fold^{Word(G)}(T_{\Sigma(G)}).$$

According to [2], generic compilers for G can be formulated in category-theoretic terms as follows:

Let $(M : Set^S \to Set^S, \eta, \epsilon)$ be a monad that encapsulates the compiler output or, in the case of incorrect input, returns error messages, $\mathcal{P} : Set^S \to Set^S$ be the (S-sorted) powerset functor, $M \times M = \_ \times \_ \circ \Delta \circ M$,

$$\oplus : M \times M \to M \quad \text{and} \quad set : M \to \mathcal{P}$$

be natural transformations and

$$E \;=\; \{m \in img(M) \mid set(m) = \emptyset\}$$

such that for all sets A, B, $m, m', m'' \in M(A)$, $e \in E$, $f : A \to M(B)$, $h : A \to B$ and $a \in A$,

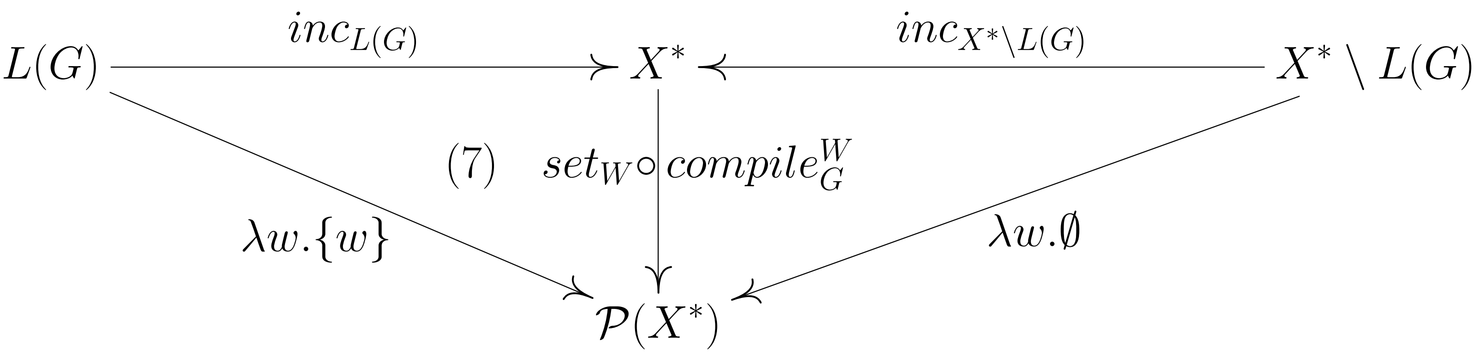
$$\begin{aligned}
(m \oplus m') \oplus m'' &= m \oplus (m' \oplus m''), \\
M(h)(e) &= e, \\
M(h)(m \oplus m') &= M(h)(m) \oplus M(h)(m'), \\
set_A(m \oplus m') &= set_A(m) \cup set_A(m'), \\
set_A(\eta_A(a)) &= \{a\}, \\
set_B(m \gg= f) &= \bigcup\{set_B(f(a)) \mid a \in set_A(m)\}.
\end{aligned}$$

Let $const(X^*)$ be the functor that maps each object and morphism of $Alg_{\Sigma(G)}$ to the S-sorted set $(X^*)_{s\in S}$ and the S-sorted function $(id_{X^*})_{s\in S}$, respectively, U be the forgetful functor from $Alg_{\Sigma(G)}$ to Set^S and $W = Word(G)$.

A natural transformation

$$compile_G : const(X^*) \to MU$$

is a **generic compiler for** G if $set_W \circ compile_G^W$ is the following coproduct extension:



Such a compiler is generic because it has two parameters: a $\Sigma(G)$-algebra $\mathcal{A}$ that represents a target language and the monad M (together with $\oplus$ and set) that determines which target objects and error messages, respectively, are to be returned.

Let $parse_G = compile_G^{T_{\Sigma(G)}}$ and $unparse_G =_{def} fold^{Word(G)}$.

Since $compile_G$ is a natural transformation and for all $\Sigma(G)$-algebras $\mathcal{A}$,

$$fold^{\mathcal{A}} : T_{\Sigma(G)} \to \mathcal{A}$$

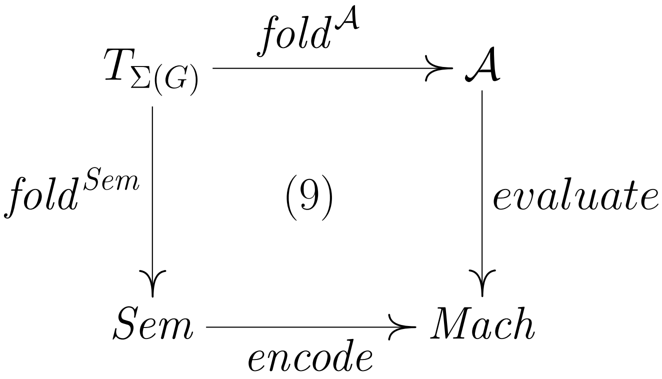
is $\Sigma(G)$-homomorphic,

$$compile_G^{\mathcal{A}} \quad = \quad X^* \xrightarrow{parse_G} M(T_{\Sigma(G)}) \xrightarrow{M(fold^{\mathcal{A}})} M(\mathcal{A}). \tag{8}$$

Hence the restriction of $parse_G$ to $L(G)$ is a right inverse of $unparse_G$:

$$set_W \circ M(unparse_G) \circ parse_G \circ inc_{L(G)} = set_W \circ M(fold^W) \circ parse_G \circ inc_{L(G)}$$
$$\stackrel{(8)}{=} set_W \circ compile_G^W \circ inc_{L(G)} \stackrel{(7)}{=} \lambda w.\{w\}.$$

Following the classical notion of compiler correctness [1, 3], we call $compile_G^{\mathcal{A}}$ **correct w.r.t. a source model** Sem **and a target model** $Mach$ ("abstract machine") if there are functions $execute : \mathcal{A} \to Mach$ and $encode : Sem \to Mach$ such that the following diagram commutes:



evaluate runs a "target program" $a \in A$ on the abstract machine *Mach*, while *encode* expresses the source model in terms of the target model.

The initiality of $T_{\Sigma(G)}$ allows us to reduce the proof that (9) commutes to the extension of *encode* and *evaluate* to $\Sigma(G)$-homomorphisms. For this purpose, *Mach* must be extended to a $\Sigma(G)$-algebra. This can often be done by establishing a target signature Σ' such that $T_{\Sigma'}$ concides with $\mathcal{A}$, each constructor of $\Sigma(G)$ corresponds to a Σ'-term, *Sem* is a Σ'-algebra and *evaluate* folds Σ'-terms in *Sem*. The mapping of $\Sigma(G)$-constructors to Σ'-terms may then determine a definition *encode* such that both *encode* and *evaluate* become $\Sigma(G)$-homomorphic. In this way, [3] shows the correctness of a compiler that translates imperative programs into data flow graphs.

In the sequel, we regard the constructors *par* and *seq* of $Reg(CS)$ as operations of mutable arity and thus write

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

$par(t)$ and $seq(t)$ stand for t.

G induces an iterative system of $Reg(CS)$-equations:

$$\begin{aligned} E_G : S &\rightarrow T_{Reg(CS)}(S) \\ s &\mapsto par(\overline{w_1}, \dots, \overline{w_k}) \end{aligned}$$

where $\{w_1, \dots, w_k\} = \{w \in (S \cup CS)^* \mid s \rightarrow w \in R\}$
and for all $n > 1$, $e_1, \dots, e_n \in S \cup CS$ and $s \in S$,

$$\begin{aligned} \overline{e_1 \dots e_n} &= seq(\overline{e_1}, \dots, \overline{e_n}), \\ \overline{s} &= s. \end{aligned}$$

E_G is called the **system of equations for G**.

The function $sol_G : S \to \mathcal{P}(X^*)$ with $sol_G(s) = L(G)_s$ for all $s \in S$ solves E_G in $Lang(X)$. (10)

sol_G is the least solution of E_G in $Lang(X)$, i.e., for all solutions g of E_G in $Lang(X)$ 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 $V \in Set^S$.

$$\begin{aligned} C\Sigma_V &= (S, BS \cup \{V_s \mid s \in S\}, BF, C \cup \{in_s : V_s \to s \mid s \in S\}), \\ \Psi_V &= (C\Sigma_V, D\Sigma), \\ \Sigma_V &= C\Sigma_V \cup D\Sigma. \end{aligned}$$

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

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

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

Proof. Let $g, h : V \to A$ solve E in A. We extend A to Σ_V-algebras A_1, A_2 by defining $in_s^{A_1} = g_s$ and $in_s^{A_2} = h_s$ for all $s \in S$. By assumption, both A_1 and A_2 satisfy $rec(E)$. Since $A|_{D\Sigma}$ is final in $Alg_{D\Sigma}$, (2) implies that the coinductive solution of $rec(E)$ in $A|_{D\Sigma}$ is unique. Hence $A_1 = A_2$ and thus for all $s \in S$, $g_s = in_s^{A_1} = in_s^{A_2} = h_s$. ❑

$\sigma_V : V \to T_{\Sigma_V}$ denotes the substitution with $\sigma_V(x) = in_s x$ for all $x \in V_s$ und $s \in S$. For all Σ_V-algebras A,

$$(in^A)^* = fold^A \circ \sigma_V^* : T_\Sigma(V) \to A, \tag{12}$$

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

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 Ψ_V-equations.

By (6), the system DC of recursive ψ-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 $E(x) = ct$,

$$in_s^{A_g}(x) = g(x) = g^*(E(x)) = g^*(ct) = c^A(g^*(t)), \tag{13}$$

and thus for all S-sorted sets V' of variables and $h : V' \to A_g$,

$$h^*(d_s(in_s x)) = d_s^{A_g}(in_s^{A_g}(x)) \stackrel{(13)}{=} d_s^A(c^A(g^*(t))) \stackrel{(6)}{=} \iota_c(g^*(t)) = \iota_c((in_s^A)^*(t))$$
$$\stackrel{(12)}{=} \iota_c(fold^{A_g}(\sigma_V^*(t))) = \iota_c(h^*(\sigma_V^*(t))) = h^*(\iota_c(\sigma_V^*(t))).$$

Hence A_g satisfies $rec(E)$, i.e.,

$rec(E)$ simulates E 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 $s \stackrel{+}{\longrightarrow}_G sw$), $CS = BS \cup \{\{z\} \mid z \in Z\}$ and $reduce$ be a function that simplifies regular expressions by applying semiring axioms.

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

$$(reduce \circ E_G^*)^{k_s}(s) = par(seq(\overline{C_{s,1}}, t_{s,1}), \dots, seq(\overline{C_{s,n_s}}, t_{s,n_s})) \quad (14)$$

or

$$(reduce \circ E_G^*)^{k_s}(s) = par(seq(\overline{C_{s,1}}, t_{s,1}), \dots, seq(\overline{C_{s,n_s}}, t_{s,n_s}), eps). \tag{15}$$

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

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

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

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

$$\begin{aligned} rec(E_G) \;=\; & \{\delta(in(s)) = \lambda x.\sigma_S^*(par(ite(\chi(C_{s,1})(x), t_{s,1}, mt), \dots, \\ & \qquad ite(\chi(C_{s,n_s})(x), t_{s,n_s}, mt))) \mid s \in S\} \cup \\ & \{\beta(in(s)) = 1 \mid s \in S_{eps}\} \cup \\ & \{\beta(in(s)) = 0 \mid s \in S \setminus S_{eps}\} \end{aligned}$$

Let $X = \bigcup CS$. By Example 5, the system BRE of recursive Ψ-equations has a unique coinductive solution A in $Pow(X)$.

Let $g : S \to A$ be a solution of E_G in A. For all $n \in \mathbb{N}$,

$$g^* \;=\; g^* \circ (reduce \circ E^*)^n. \tag{16}$$

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

$$h^*(in(s)) = in^{A_g}(s) = g(s) = g^*(s) \stackrel{(16)}{=} g^*((reduce \circ E_G^*)^{k_s}(s)) \tag{17}$$

By (12),

$$g^* = (in^{A_g})^* = fold^{A_g} \circ \sigma_S^* : T_{Reg(CS)}(S) \to A. \tag{18}$$

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

$$\begin{aligned}
&h^*(\delta(in(s))) = \delta^A(h^*(in(s))) \stackrel{(17)}{=} \delta^A(g^*((reduce \circ E_G^*)^{k_s}(s))) = \dots \\
&= \delta^A(\textstyle\bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i}))) = \lambda x.\delta^A(\bigcup_{i=1}^{n}(C_{s,i} \cdot g^*(t_{s,i})))(x) \\
&\stackrel{Def.\ \delta^A}{=} \lambda x.\{w \in X^* \mid xw \in \textstyle\bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i}))\} = \dots \\
&= g^*(\lambda x.par(ite(\chi(C_{s,1})(x), t_{s,1}, mt), \dots, ite(\chi(C_{s,n_s})(x), t_{s,n_s}, mt))) \\
&\stackrel{(18)}{=} fold^{A_g}(\sigma_S^*(\lambda x.par(ite(\chi(C_{s,1})(x), t_{s,1}, mt), \dots, ite(\chi(C_{s,n_s})(x), t_{s,n_s}, mt)))) \\
&= h^*(\sigma_S^*(\lambda x.par(ite(\chi(C_{s,1})(x), t_{s,1}, mt), \dots, ite(\chi(C_{s,n_s})(x), t_{s,n_s}, mt))))
\end{aligned}$$

and

$$h^*(\beta(in(s))) = \beta^A(h^*(in(s))) \stackrel{(17)}{=} \beta^A(g^*((reduce \circ E_G^*)^{k_s}(s))) = \dots$$
$$= \beta^A(\bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i}))) \stackrel{Def.\ \beta^A}{=} 0 = h^*(0),$$

and for all $s \in S_{eps}$,

$$h^*(\delta(in(s))) = \delta^A(h^*(in(s))) \stackrel{(17)}{=} \delta^A(g^*((reduce \circ E_G^*)^{k_s}(s))) = \dots$$
$$= \delta^A(\bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i})) \cup \{\epsilon\}) = \lambda x.\delta^A(\bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i})) \cup \{\epsilon\})(x)$$
$$\stackrel{Def.\ \delta^A}{=} \lambda x.\{w \in X^* \mid xw \in \bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i})) \cup \{\epsilon\}\}$$
$$= \lambda x.\{w \in X^* \mid xw \in \bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i}))\} = \dots$$
$$= g^*(\lambda x.par(ite(\chi(C_{s,1})(x), t_{s,1}, mt), \dots, ite(\chi(C_{s,n_s})(x), t_{s,n_s}, mt)))$$
$$\stackrel{(18)}{=} fold^{A_g}(\sigma_S^*(\lambda x.par(ite(\chi(C_{s,1})(x), t_{s,1}, mt), \dots, ite(\chi(C_{s,n_s})(x), t_{s,n_s}, mt))))$$
$$= h^*(\sigma_S^*(\lambda x.par(ite(\chi(C_{s,1})(x), t_{s,1}, mt), \dots, ite(\chi(C_{s,n_s})(x), t_{s,n_s}, mt))))$$

and

$$h^*(\beta(in(s))) = \beta^A(h^*(in(s))) \stackrel{(17)}{=} \beta^A(g^*((reduce \circ E_G^*)^{k_s}(s))) = \dots$$
$$= \beta^A(\bigcup_{i=1}^{n_s}(C_{s,i} \cdot g^*(t_{s,i})) \cup \{\epsilon\}) \stackrel{Def.\ \beta^A}{=} 1 = h^*(1).$$

Hence A_g satisfies $rec(E_G)$, i.e.,

$$rec(E_G) \text{ simulates } E_G \text{ in } A. \tag{19}$$

$(10) \wedge (11) \wedge (19) \quad \Rightarrow \quad sol_G$ is the only solution of E_G in A.

$rec(E_G)$ suggests the following extension of $Bro(CS)$ to a $Reg(CS)'$-Algebra $Bro(CS)'$:

For all $s \in S$,

$$\delta^{Bro(CS)'}(in(s)) = \lambda x.\sigma_S^*(par(ite(x \in C_{s,1}, t_{s,1}, mt), \dots, ite(x \in C_{s,n_s}, t_{s,n_s}, mt))),$$
$$\beta^{Bro(CS)'}(in(s)) = if \;\; s \in S_{eps} \; then \; 1 \; else \; 0.$$

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

$Bro(CS)'$ agrees with the Σ-algebra $T_{Reg(CS)'}$ (see (2)). Hence

$$fold^{Lang(X)'} \;\; = \;\; unfold^{Bro(CS)'} : Bro(CS)' \to Pow(X)$$

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

$$unfold^{Bro(CS)'}(in(s)) = fold^{Lang(X)'}(in(s)) = in^{Lang(X)'}(s)$$
$$= in^{A_{sol_G}}(s) = sol_G(s) = L(G)_s,$$

i.e., $(Bro(CS)', in(s))$ 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 clauses

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

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

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

$$A \leq B \;\Leftrightarrow_{def}\; \forall\, p \in P' : p^A \subseteq p^B.$$

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

$$p^\perp = \emptyset, \quad p^\top = A_e, \quad p^{\sqcup\mathcal{A}} = \bigcup_{A \in \mathcal{A}} p^A \text{ and } p^{\sqcap\mathcal{A}} = \bigcap_{A \in \mathcal{A}} p^A.$$

A Σ'-formula φ is **negation-free w.r.t.** Σ if φ does not contain $\Rightarrow$, $\Leftarrow$ or $\Leftrightarrow$ and all subformulas of φ with a leading negation symbol belong to $Fo_\Sigma(V)$.

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

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

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

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

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

Φ is monotone and thus by the Fixpoint Theorem of Knaster and Tarski, Φ has the least fixpoint

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

Consequently,

$$lfp(\phi) \models p(x) \Leftrightarrow \bigvee_{p(t)\Leftarrow\varphi\in AX} \exists\, var(t,\varphi) : (x = t \wedge \varphi).$$

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

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

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

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

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

Φ is monotone and thus by the Fixpoint Theorem of Knaster and Tarski, Φ has the greatest fixpoint

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

Consequently,

$$gfp(\phi) \quad \models \quad p(x) \Leftrightarrow \bigwedge_{p(t) \Rightarrow \varphi \in AX} \forall\, var(t, \varphi) : (x \neq t \vee \varphi).$$

*** to be continued ***

References

[1] F.L. Morris, *Advice on Structuring Compilers and Proving Them Correct*, Proc. ACM POPL (1973) 144-152 *link*

[2] P. Padawitz, *Übersetzerbau*, TU Dortmund 2016, *link*

[3] J.W. Thatcher, E.G. Wagner, J.B. Wright, *More on Advice on Structuring Compilers and Proving Them Correct*, Theoretical Computer Science 15 (1981) 223-249 *link*