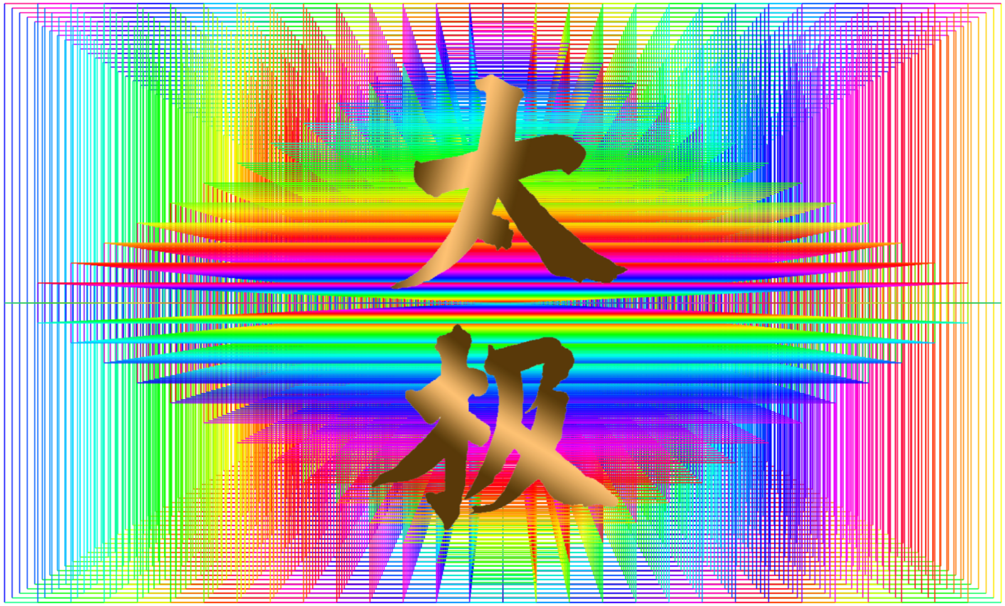


FCAM - old stuff

Peter Padawitz
TU Dortmund, Germany
March 17, 2023

(actual version: https://fldit-www.cs.tu-dortmund.de/∼peter/DialgOldStuff.pdf)

Contents

Notational conventions

Given a constructive or destructive signature $\Sigma = (S, F, P)$, $\mu\Sigma$ and $\nu\Sigma$ denote the initial or final Σ-algebra, respectively.

We write f^μ and f^ν for the interpretation of $f \in F$ in $\mu\Sigma$ and $\nu\Sigma$, respectively.

Binary trees

Let X be a set.

$$\begin{aligned}
S &= \{btree\}, \\
F &= \{empty : 1 \to btree,\ join : btree \times X \times btree \to btree\}, \\
F' &= \{split : btree \to 1 + (btree \times X \times btree)\}, \\
F'' &= \{root : btree \to X,\ left, right : btree \to btree\}, \\
Bintree(X) &= (S, \{X\}, F, \emptyset), \\
coBintree(X) &= (S, \{X\}, F', \emptyset), \\
infBintree(X) &= (S, \{X\}, F'', \emptyset).
\end{aligned}$$

- For all $A \in Set^S$,
  $H_{Bintree(X)}(A)_{btree} = H_{coBintree(X)}(A)_{btree} = 1 + A_{btree} \times X \times A_{btree}$ and
  $H_{infBintree(X)}(A)_{btree} = A_{btree} \times X \times A_{btree}$.
- $\mu Bintree(X)_{btree} \cong T$ where T is the least set of expressions such that $\bot \in T$ and for all $x \in X$ and $t, u \in T$, $x(t, u) \in T$.
- $empty = \bot$ and for all $x \in X$ and $t, u \in T$, $join(t, x, u) = x(t, u)$.

- $\nu coBintree(X)_{btree} \cong T'$ where T' is the set of partial functions $t : 2^* \to X$ such that for all $w \in 2^*$,
  - if $t(w0)$ is defined, then $t(w)$ is defined,
  - if $t(w1)$ is defined, then $t(w0)$ is defined.
- For all $t \in T'$,

$$split(t) = \begin{cases} * & \text{if } t = \Omega, \\ (\lambda w.t(0w), t(\epsilon), \lambda w.t(1w)) & \text{otherwise.} \end{cases}$$

- $\nu infBintree(X)_{btree} \cong X^{2^*}$.
- For all $t \in X^{2^*}$, $root(t) = t(\epsilon)$, $left(t) = \lambda w.t(0w)$ and $right(t) = \lambda w.t(1w)$.

Trees

Let X be a set.

$$\begin{array}{rcl} S &=& \{tree, trees\}, \\ F &=& \{join : X \times trees \to tree,\ \alpha : 1 \to trees, \\ && \quad cons : tree \times trees \to trees\}, \\ F' &=& \{root : tree \to X,\ subtrees : tree \to trees, \\ && \quad split : trees \to 1 + tree \times trees\}, \\ Tree(X) &=& (S, \mathcal{I}, F) \ = \ Tree(X, 1), \\ coTree(X) &=& (S, \mathcal{I}, F') \ = \ coTree(X, 1) \end{array}$$

(see chapter 8).

- For all $A \in Set^S$, $H_{Tree(X)}(A)_{tree} = H_{coTree(X)}(A)_{tree} = X \times A_{trees}$ and $H_{Tree(X)}(A)_{trees} = H_{coTree(X)}(A)_{trees} = 1 + (A_{tree} \times A_{trees})$.
- $\mu Tree(X)_{tree} \cong T$ and $\mu Tree(X)_{trees} \cong T^*$ where T is the least set of expressions such that for all $x \in X$ and $ts \in T^*$, $x \in T$ and $x(ts) \in T$.
- $\alpha = \epsilon$
  and for all $x \in X$, $t \in T$ and $ts \in T^*$, $join(x, ts) = x(ts)$ and $cons(t, ts) = t : ts$.

- $\nu\,coTree(X)_{tree} \cong T'$ and $\nu\,coTree(X)_{trees} \cong (T')^{\infty}$ where T' is the set of partial functions $t : (\mathbb{N} \cup \{\omega\})^* \to X$ such that for all $w \in (\mathbb{N} \cup \{\omega\})^*$ and $i \in \mathbb{N}$,
  - $t(\epsilon)$ is defined,
  - if $t(w0)$ is defined, then $t(w)$ is defined,
  - if $t(w(i+1))$ is defined, then $t(wi)$ is defined,
  - if $t(w\omega)$ is defined, then for all $i \in \mathbb{N}$, $t(wi)$ is defined.
- For all $t \in T'$, $root(t) = t(\epsilon)$ and

$$subtrees(t) = \begin{cases} * & \text{if } t = \Omega, \\ \lambda i.\lambda w.t(iw) & \text{otherwise.} \end{cases}$$

- For all $ts \in (T')^{\infty}$,

$$split(ts) = \begin{cases} * & \text{if } ts = \epsilon, \\ (ts(0), \lambda i.ts(i+1)) & \text{otherwise.} \end{cases}$$

Sample recursive functions

Recursion: Length of a finite list

The function $length : X^* \to \mathbb{N}$ satisfies the equations

$$length(nil) = 0 \quad (1)$$
$$length(cons(x,s)) = length(s) + 1 \quad (2)$$

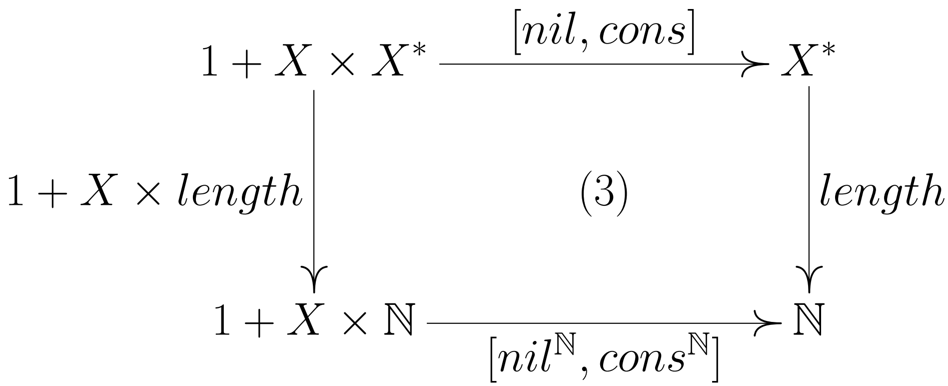
Define $\mathcal{K} = Set$ and $L = R = Id_{Set}$.

By (2), the kernel of *length* is compatible with *cons*:

$$length(s) = length(s')$$
$$\Rightarrow length(cons(x,s)) = length(s) + 1 = length(s') + 1 = length(cons(x,s')).$$

Hence *length* is *List*(X)-recursive and thus by Lemma KER (1), *length* agrees with $fold^{\mathbb{N}}$ where $nil^{\mathbb{N}} = 0$ and $cons^{\mathbb{N}} = \lambda(x,n).n+1$.

The validity of (1) and (2) is equivalent to the commutativity of (3):



Recursion and product: Fibonacci numbers

The function $fib : \mathbb{N} \to \mathbb{N}$ satisfies the equations

$$\begin{aligned} fib(zero) &= 0 \\ fib(succ(zero)) &= 1 \\ fib(succ(succ(n))) &= fib(n) + fib(succ(n)) \end{aligned}$$

Again, these equations do not imply that the kernel of fib is a Σ-congruence.

We regard the composition $fib \circ succ$ as a further function $fib' : \mathbb{N} \to \mathbb{N}$ and transform the above equations into a mutually recursive definition of fib and fib':

$$\langle fib, fib' \rangle(zero) = (0, 1) \tag{1}$$

$$\langle fib, fib' \rangle(succ(n)) = (fib'(n), fib(n) + fib'(n)) \tag{2}$$

Define $\mathcal{K} = Set^2$ and for all $A, B \in Set$, $L(A)_{nat} = (A_{nat}, A_{nat})$ and $R(A, B)_{nat} = A_{nat} \times B_{nat}$.

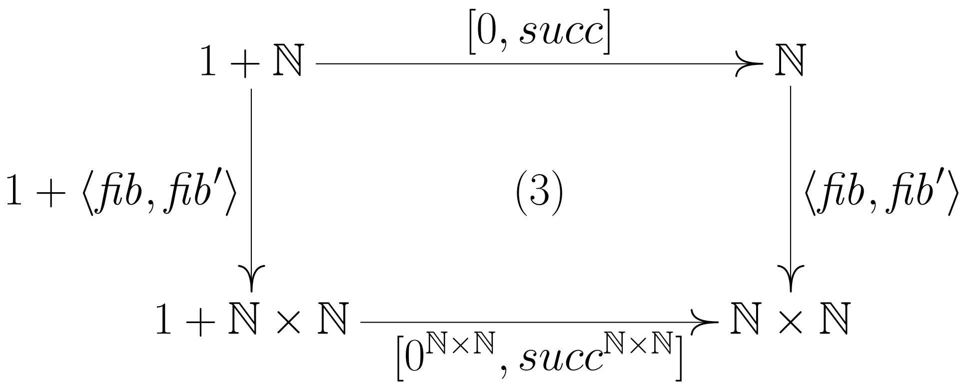
By (1) and (2), the kernel of $(fib, fib')^\# = \langle fib, fib' \rangle : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ is compatible with $succ$:

$$
\begin{aligned}
&(fib(m), fib'(m)) = \langle fib, fib' \rangle(m) = \langle fib, fib' \rangle(n) = (fib(n), fib'(n)) \\
&\Rightarrow \langle fib, fib' \rangle(succ(m)) = (fib(succ(m)), fib'(succ(m))) = (fib'(m), fib(m) + fib'(m)) \\
&\qquad = (fib'(n), fib(n) + fib'(n)) = (fib(succ(n)), fib'(succ(n))) = \langle fib, fib' \rangle(succ(n)).
\end{aligned}
$$

Hence $(fib, fib') : (\mathbb{N}, \mathbb{N}) \to (\mathbb{N}, \mathbb{N})$ is Nat-recursive and thus by Lemma KER (1), $\langle fib, fib' \rangle$ agrees with $fold^{\mathbb{N} \times \mathbb{N}}$ where

$$
\begin{aligned}
0^{\mathbb{N} \times \mathbb{N}} &= (0, 1), \\
succ^{\mathbb{N} \times \mathbb{N}} &= \lambda(m, n).(n, m + n).
\end{aligned}
$$

The validity of (1) and (2) is equivalent to the commutativity of (3):



Recursion and currying: Concatenation of finite lists

The function $conc : X^* \times X^* \to X^*$ satisfies the equations

$$\begin{aligned} conc(nil, s) &= s & (1)\\ conc(cons(x,s), s') &= cons(x, conc(s,s')) & (2) \end{aligned}$$

Define $\mathcal{K} = Set$ and for all $A \in Set$, $L(A)_{list} = A_{list} \times X^*$ and $R(A)_{list} = A_{list}^{X^*}$.

Let $Z = (X^*)^{X^*}$. By (2), the kernel of $conc^\# : X^* \to Z$ is compatible with $cons$:

$$\begin{aligned} & conc^\#(s) = conc^\#(s') \\ & \Rightarrow conc^\#(cons(x,s)) = \lambda s''.conc(cons(x,s),s'') = \lambda s''.cons(x, conc(s,s'')) \\ & \quad = \lambda s''.cons(x, conc^\#(s)(s'')) = \lambda s''.cons(x, conc^\#(s')(s'')) \\ & \quad = \lambda s''.cons(x, conc(s',s'')) = \lambda s''.conc(cons(x,s'),s'') = conc^\#(cons(x,s')). \end{aligned}$$

Hence $conc$ is $List(X)$-recursive and thus by Lemma KER (1), $conc^\#$ agrees with $fold^Z$ where $nil^Z = \lambda s.s$ and $cons^Z = \lambda(x,f).\lambda s.cons(x, f(s))$.

The validity of (1) and (2) is equivalent to the commutativity of (3):

$$\begin{array}{ccc} 1 + X \times X^* & \xrightarrow{[nil, cons]} & X^* \\ {\scriptstyle 1 + X \times conc^\#}\downarrow & (3) & \downarrow{\scriptstyle conc^\#} \\ 1 + X \times Z & \xrightarrow[{[nil^Z, cons^Z]}]{} & Z \end{array}$$

Recursion and identity: Folding a finite list from the right

Let A be a set and $Z = (X \times A \to A) \to A \to A$.
The function $foldr : X^* \to (X \times A \to A) \to A \to A$ satisfies the equations

$$foldr(nil)(f)(a) \;=\; a \qquad (1)$$

$$foldr(cons(x,s))(f)(a) \;=\; f(e, foldr(s)(f)(a)) \qquad (2)$$

Define $\mathcal{K} = Set$ and $L = R = Id_{Set}$.

By (2), the kernel of $foldr$ is compatible with $cons$:

$$\begin{aligned} & foldr(s) = foldr(s') \\ & \quad \Rightarrow \; foldr(cons(x,s)) = \lambda f.\lambda a.f(e, foldr(s)(f)(a)) = \lambda f.\lambda a.f(x, foldr(s')(f)(a)) \\ & \quad\quad\; = foldr(cons(x,s')). \end{aligned}$$

Hence *foldr* is $List(X)$-recursive and thus by Lemma KER (1), *foldr* agrees with $fold^Z$ where for all $f : X \times A \to A$, $a \in A$, $x \in X$ and $g \in Z$,

$$
\begin{array}{rcl}
nil^Z(f)(a) & = & a, \\
cons^Z(x, g)(f)(a) & = & \lambda s. g(f)(a)(x : s).
\end{array}
$$

The validity of (1) and (2) is equivalent to the commutativity of (3):

$$
\begin{CD}
1 + X \times X^* @>{[nil, cons]}>> X^* \\
@V{1 + X \times foldr}VV \quad (3) \quad @VV{foldr}V \\
1 + X \times Z @>>{[nil^Z, cons^Z]}> Z
\end{CD}
$$

Recursion and identity: Filter a finite list

Let $Z = (X \to 2) \to X^*$. The function $filter : X^* \to Z$ satisfies the equations

$$
\begin{array}{rcll}
filter(nil)(f) & = & nil & (1) \\
filter(cons(x, s))(f) & = & if \ f(x) \ then \ filter(s)(f) \ else \ x : filter(s)(f) & (2)
\end{array}
$$

Define $\mathcal{K} = Set$ and $L = R = Id_{Set}$.

By (2), the kernel of *filter* is compatible with *cons*:

$$\begin{aligned}
&\mathit{filter}(s) = \mathit{filter}(s') \\
&\Rightarrow \mathit{filter}(\mathit{cons}(x,s)) = \lambda f.\mathit{if}\ \ f(x)\ \mathit{then}\ \mathit{filter}(s)(f)\ \mathit{else}\ x\!:\!\mathit{filter}(s)(f) \\
&\quad = \lambda f.\mathit{if}\ \ f(x)\ \mathit{then}\ \mathit{filter}(s')(f)\ \mathit{else}\ x\!:\!\mathit{filter}(s')(f) = \mathit{filter}(\mathit{cons}(x,s')).
\end{aligned}$$

Hence *filter* is $List(X)$-recursive and thus by Lemma KER (1), *filter* agrees with $fold^Z$ where for all $f : X \to 2$, $x \in X$ and $g \in Z$, $nil^Z(f) = nil$ and

$$cons^Z = \lambda(x,g).\lambda f.\lambda s.g(f)(x\!:\!s).$$

The validity of (1) and (2) is equivalent to the commutativity of (3):

$$\begin{array}{ccc}
1 + X \times X^* & \xrightarrow{[nil,\, cons]} & X^* \\
{\scriptstyle 1 + X \times \mathit{filter}}\downarrow & (3) & \downarrow{\scriptstyle \mathit{filter}} \\
1 + X \times Z & \xrightarrow[{[nil^Z,\, cons^Z]}]{} & Z
\end{array}$$

Recursion and currying: Replication

Let X be a set. The function $repl : \mathbb{N} \times X \to X^*$ satisfies the equations

$$\begin{aligned} repl(zero, e) &= nil & (1) \\ repl(succ(n), e) &= cons(e, repl(n, e)) & (2) \end{aligned}$$

where $nil = nil^{\mu List(X)}$ and $cons = cons^{\mu List(X)}$ (see Lists and Streams).

Define $\mathcal{K} = Set$ and for all $A \in Set$, $L(A)_{nat} = A \times X$ and $R(A)_{nat} = A^X$.

Let $Z = (X^*)^X$. By (2), the kernel of $repl^\# : \mathbb{N} \to Z$ is compatible with $succ$:

$$\begin{aligned} & repl^\#(m) = repl^\#(n) \\ & \Rightarrow repl^\#(succ(m)) = \lambda e.cons(e, repl^\#(m)(e)) = \lambda e.cons(e, repl(m, e)) \\ & \quad = \lambda e.cons(e, repl(n, e)) = \lambda e.cons(e, repl^\#(n)(e)) = repl^\#(succ(n)). \end{aligned}$$

Hence $repl$ is Nat-recursive and thus by Lemma KER (1), $repl^\#$ agrees with $fold^Z$ where

$$\begin{aligned} 0^Z &= \lambda e.\epsilon, \\ succ^Z &= \lambda f.\lambda e.(e : f(e)). \end{aligned}$$

The validity of (1) and (2) is equivalent to the commutativity of (3):

$$\begin{array}{ccc} 1+\mathbb{N} & \xrightarrow{[0,\,succ]} & \mathbb{N} \\ {\scriptstyle 1+repl^{\#}}\downarrow & (3) & \downarrow{\scriptstyle repl^{\#}} \\ 1+Z & \xrightarrow[{[0^Z,\,succ^Z]}]{} & Z \end{array}$$

We have shown that there is a unique interpretation in $\mu List(X)$ of an additional constructor $repl : \mathbb{N} \times X \to list$ such that the corresponding extension of $\mu List(X)$ satisfies the equations for repl given in chapter 14.

Let $\Sigma = (S, F \cup \{repl\}, \{=: list \times list\})$, $\Sigma' = (S, F \cup \{repl\}, \emptyset)$ and AX be a set of Σ-Horn clauses such that for all $A \in Alg_{\Sigma,AX}$, $=^A$ is a Σ-congruence, and AX includes the equations for repl given in chapter 14.

Let $A = lfp(\Sigma, \mu\Sigma', AX)$. By Theorem ABSINI, $A/{=^A}$ is initial in $Alg_{\Sigma,AX}$. Since the initial $List(X)$-algebra is a (Σ, AX)-algebra, we conclude from Lemma CONEXT that (Σ, AX) is a conservative extension of $(List(X), \emptyset)$.

Recursion and identity: Subtrees of a cobintree

Let $Z = (\nu coBintree(X) \to \nu coBintree(X))$. The function

$$subtree : 2^* \to Z$$

satisfies the equations

$$
\begin{aligned}
& subtree(\alpha)(t) = t && (1)\\
fork(t) = (u,e,u') \;\Rightarrow\; & subtree(cons(0,s))(t) = subtree(s)(u) && (2)\\
fork(t) = (u,e,u') \;\Rightarrow\; & subtree(cons(1,s))(t) = subtree(s)(u') && (3)
\end{aligned}
$$

Define $\mathcal{K} = Set$ and $L = R = Id_{Set}$.

By (1)-(3), the kernel of *subtree* is compatible with *fork*.

Hence *subtree* is $List(2)$-recursive and thus by Lemma KER (1), *subtree* agrees with $fold^Z$ where for all $s \in 2^*$, $f \in Z$ and $t \in \nu coBintree(X)$,

$$
\begin{aligned}
\alpha^Z &= id,\\
cons^Z(b,f)(t) &= \begin{cases} f(u) & \text{if } b = 0 \text{ and } fork(t) = (u,e,u'),\\ f(u') & \text{if } b = 1 \text{ and } fork(t) = (u,e,u'). \end{cases}
\end{aligned}
$$

The validity of (1)-(3) is equivalent to the commutativity of (4):

$$\begin{array}{ccc} 1+2\times 2^* & \xrightarrow{[\alpha, cons]} & 2^* \\ {\scriptstyle 1+2\times subtree}\downarrow & (4) & \downarrow{\scriptstyle subtree} \\ 1+2\times Z & \xrightarrow[{[\alpha^Z, cons^Z]}]{} & Z \end{array}$$

Recursion and product: Check balancing (see [51])

Let $T = \mu Bintree(X)_{btree}$. The functions $depth : T \to \mathbb{N}$ and $bal : T \to 2$ satisfy the equations

$$\begin{aligned} \langle height, bal\rangle(empty) &= (0, True) & (1) \\ \langle height, bal\rangle(join(t,x,u)) &= (max(height(t), height(u)) + 1, \\ & \quad bal(t) \wedge bal(u) \wedge height(t) = height(u)) & (2) \end{aligned}$$

Define $\mathcal{K} = Set^2$ and for all $A, B \in Set$, $L(A)_{btree} = (\mathbb{N}, 2)$ and $R(A,B)_{btree} = A_{btree} \times B_{btree}$.

By (1) and (2), the kernel of

$$(height, bal)^{\#} = \langle height, bal\rangle : T \to \mathbb{N} \times 2$$

is compatible with *join*. Hence $(height, bal) : (T, T) \to (\mathbb{N}, 2)$ is $Bintree(X)$-recursive and thus by Lemma KER (1), $\langle height, bal\rangle$ agrees with $fold^{\mathbb{N}\times 2}$ where

$$
\begin{array}{rcl}
empty^{\mathbb{N}\times 2} & = & (0, True), \\
join^{\mathbb{N}\times 2} & = & \lambda((m,b),x,(n,c)).(max(m,n)+1, b \wedge c \wedge m = n).
\end{array}
$$

The validity of (1) and (2) is equivalent to the commutativity of (3):

$$
\begin{array}{ccc}
1 + T \times X \times T & \xrightarrow{[empty, join]} & T \\
\Big\downarrow {\scriptstyle 1+\langle height, bal\rangle} & (3) & \Big\downarrow {\scriptstyle \langle height, bal\rangle} \\
1 + (\mathbb{N}\times 2) \times X \times (\mathbb{N}\times 2) & \xrightarrow[{[empty^{\mathbb{N}\times 2}, join^{\mathbb{N}\times 2}]}]{} & \mathbb{N}\times 2
\end{array}
$$

Recursion and identity: Flatten a finite tree (see [65])

The functions $flatten : \mu Tree(X)_{tree} \to X^*$ and $flattenL : \mu Tree(X)_{trees} \to X^*$ satisfy the equations

$$
\begin{array}{rcll}
flatten(join(x,ts)) & = & x : flattenL(ts) & (1) \\
flattenL(\alpha) & = & \alpha & (2) \\
flattenL(cons(t,ts)) & = & flatten(t) +\!\!+ flattenL(ts) & (3)
\end{array}
$$

Define $\mathcal{K} = Set$ and $L = R = Id_{Set}$.

Since $S = \{tree, trees\}$, *flatten* and *flattenL* provide the *tree*- or *trees*-component of a Set^2-morphism function $flatten' : (\mu Tree(X)_{tree}, \mu Tree(X)_{trees}) \to (X^*, X^*)$.

By (1)-(3), the kernel of *flatten* is compatible with *join* and *cons*.

Hence $flatten'$ is $Tree(X)$-recursive and thus by Lemma KER (1) (1), $flatten'$ agrees with $fold^{X^*}$ where $join^{X^*} = \lambda(x, s).(x : s)$, $\alpha^{X^*} = \epsilon$ and $cons^{X^*} = \lambda(s, s').(s +\!\!+ s'))$.

The validity of (1)-(3) is equivalent to the commutativity of (4) and (5):

$$\begin{array}{ccc}
X \times \mu Tree(X)_{trees} & \xrightarrow{\;join\;} & \mu Tree(X)_{tree} \\
{\scriptstyle X \times flattenL}\downarrow & (4) & \downarrow{\scriptstyle flatten} \\
X \times X^* & \xrightarrow[\;join^{X^*}\;]{} & X^*
\end{array}$$

$$
\begin{array}{ccc}
1 + (\mu Tree(X)_{tree} \times \mu Tree(X)_{trees}) & \xrightarrow{[\alpha, cons]} & \mu Tree(X)_{trees} \\
\Big\downarrow {\scriptstyle 1 + (flatten \times flattenL)} & (5) & \Big\downarrow {\scriptstyle flattenL} \\
1 + (X^* \times X^*) & \xrightarrow[{[\alpha^{X^*}, cons^{X^*}]}]{} & X^*
\end{array}
$$

Corecursion: Addition on $\mathbb{N}_\infty$ (see [77])

Define $L : Set^2 \to Set$ and $R : Set \to Set^2$ as follows: For all $A, B \in Set$ and $g, h \in Mor(Set)$, $L(A,B) = A + B$, $L(g,h) = g + h$, $R(A) = (A, A)$ and $R(g) = (g, g)$.

Let $C = (\mathbb{N}' \times \mathbb{N}', \mathbb{N}')$. Then $L(C) = \mathbb{N}' \times \mathbb{N}' + \mathbb{N}'$ and $R(\nu\Sigma) = R(\mathbb{N}') = (\mathbb{N}', \mathbb{N}')$.

Moreover, $L(C)$ is a *coNat*-algebra: For all $m, n \in \mathbb{N}'$,

$$
\begin{aligned}
pred^{L(C)}(m,n) &= \begin{cases} \epsilon & \text{if } m = n = 0, \\ (0, n-1) & \text{if } m = 0 \wedge n \in \mathbb{N}' \setminus \{0\}, \\ (m-1, n) & \text{if } m \in \mathbb{N}' \setminus \{0\}, \end{cases} \\
pred^{L(C)}(n) &= pred(n).
\end{aligned}
$$

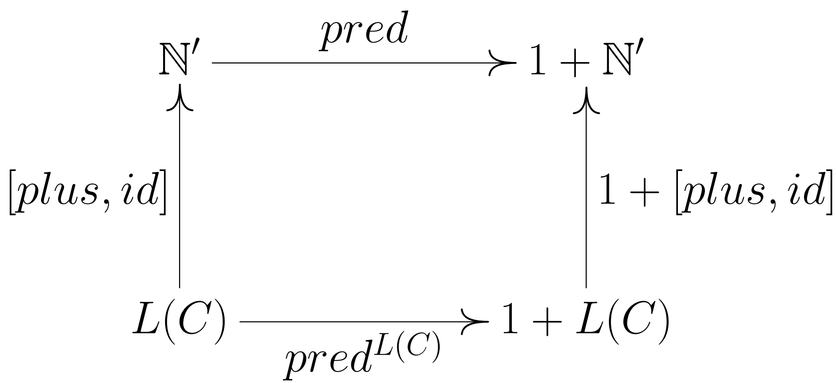
Let the arrow $join' : (1 + nat) + nat \to 1 + nat$ be interpreted as follows:

For all $A \in Alg_{coNat}$, $a \in A_{nat}$ and $i \in \{1,2\}$, $join'(\epsilon, 1) = \epsilon$ and $join'(a,i) = a$.

A function $plus : \mathbb{N}' \times \mathbb{N}' \to \mathbb{N}'$ satisfies the equation

$$\begin{aligned} pred(plus(m,n)) \;=\; & join'(\lambda(\iota_1(x_1).\lambda(\iota_1(y_1).\epsilon|\iota_2(y_2).y_2)(pred(n)), \\ & \iota_2(x_2).plus(x_2,n))(pred(m))) \end{aligned} \quad (1)$$

iff $(plus, id)^* = [plus, id] : L(C) \to \mathbb{N}'$ is $coNat$-homomorphic, i.e., the following diagram commutes:



Hence by Lemma COREC1, equations (1)-(3) have a unique solution $plus$.

Corecursion and coproduct: A blinker

Suppose that $on, off \in X$. The functions $blink : 1 \to X^{\mathbb{N}}$ and $blink' : 1 \to X^{\mathbb{N}}$ satisfy the equations

$$\langle head, tail\rangle(blink) \;=\; (on, blink') \quad (1)$$

$$\langle head, tail\rangle(blink') \;=\; (off, blink) \quad (2)$$

Define $\mathcal{K} = Set^2$ and for all $A, B \in Set$, $R(A)_{list} = (A_{list}, A_{list})$ and $L(A, B)_{list} = A_{list} + B_{list}$.

Let $Q = 1 + 1$. By (1) and (2), the image of $(blink, blink')^* = [blink, blink'] : Q \to X^{\mathbb{N}}$ is compatible with *head* and *tail*.

Hence $(blink, blink') : Q \to (X^{\mathbb{N}}, X^{\mathbb{N}})$ is $Stream(X)$-corecursive and thus by Lemma IMG (1), $[blink, blink']$ agrees with $unfold^Q$ where $\langle head^Q, tail^Q \rangle(*, 1) = (on, (*, 2))$ and $\langle head^Q, tail^Q \rangle(*, 2) = (off, (*, 1))$.

The validity of (1) and (2) is equivalent to the commutativity of (3):

$$\begin{array}{ccc} X^{\mathbb{N}} & \xrightarrow{\langle head, tail \rangle} & X \times X^{\mathbb{N}} \\ {\scriptstyle [blink, blink']} \uparrow & (3) & \uparrow {\scriptstyle X \times [blink, blink']} \\ Q & \xrightarrow[\langle head^Q, tail^Q \rangle]{} & X \times Q \end{array}$$

Corecursion and coproduct: Exchange stream elements (see [162])

The function $exch : X^{\mathbb{N}} \to X^{\mathbb{N}}$, which exchanges each two consecutive elements of a stream, satisfies the equations

$$\begin{aligned} head(exch(s)) &= head(tail(s)) \\ \langle head, tail\rangle(tail(exch(s))) &= (head(s), exch(tail(tail(s)))) \end{aligned}$$

We regard the composition $tail \circ exch$ as a further function

$$exch' : X^{\mathbb{N}} \to X^{\mathbb{N}}$$

and transform the above equations into a mutually recursive definition of $exch$ and $exch'$:

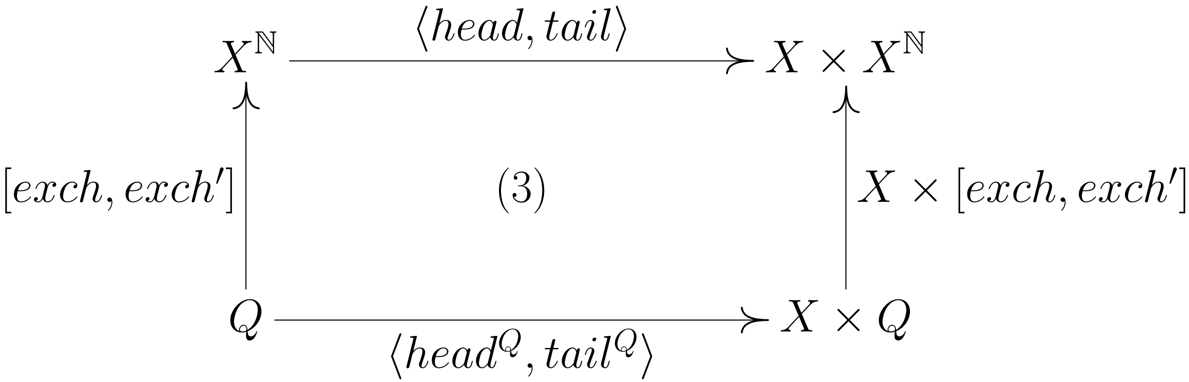
$$\begin{aligned} \langle head, tail\rangle(exch(s)) &= (head(tail(s)), exch'(s)) && (1) \\ \langle head, tail\rangle(exch'(s)) &= (head(s), exch(tail(tail(s)))) && (2) \end{aligned}$$

Define $\mathcal{K} = Set^2$ and for all $A, B \in Set$, $R(A)_{list} = (A_{list}, A_{list})$ and $L(A, B)_{list} = A_{list} + B_{list}$.

Let $Q = X^{\mathbb{N}} + X^{\mathbb{N}}$. By (1) and (2), the image of $(exch, exch')^* = [exch, exch'] : Q \to X^{\mathbb{N}}$ is compatible with $head$ and $tail$.

Hence $(exch, exch') : (X^{\mathbb{N}}, X^{\mathbb{N}}) \to (X^{\mathbb{N}}, X^{\mathbb{N}})$ is *Stream*-recursive and thus by Lemma IMG (1), $[exch, exch']$ agrees with $unfold^Q$ where for all $s \in X^{\mathbb{N}}$,
$\langle head^Q, tail^Q\rangle(s, 1) = (head(tail(s)), (s, 2))$ and
$\langle head^Q, tail^Q\rangle(s, 2) = (head(s), (tail(tail(s)), 1))$.

The validity of (1) and (2) is equivalent to the commutativity of (3):



Corecursion and coproduct: Alternation of successors and squares (see [65])

The functions $nats : \mathbb{N} \to X^{\mathbb{N}}$ and $squares : \mathbb{N} \to X^{\mathbb{N}}$ satisfy the equations

$$\langle head, tail\rangle(nats(n)) = (n, squares(n)) \quad (1)$$

$$\langle head, tail\rangle(squares(n)) = (n * n, nats(n + 1)) \quad (2)$$

Define $\mathcal{K} = Set^2$ and for all $A, B \in Set$, $R(A)_{list} = (A_{list}, A_{list})$ and $L(A, B)_{list} = A_{list} + B_{list}$.

Let $Q = \mathbb{N} + \mathbb{N}$. By (1) and (2), the image of

$$(nats, squares)^* = [nats, squares] : Q \to X^{\mathbb{N}}$$

is compatible with *head* and *tail*.

Hence $(nats, squares) : (\mathbb{N}, \mathbb{N}) \to (X^{\mathbb{N}}, X^{\mathbb{N}})$ is *Stream*-recursive and thus by Lemma IMG (1), $[nats, squares]$ agrees with $unfold^Q$ where for all $n \in \mathbb{N}$,
$\langle head^Q, tail^Q\rangle(n,1) = (n,(n,2))$ and $\langle head^Q, tail^Q\rangle(n,2) = (n*n,(n+1,1))$.

The validity of (1) and (2) is equivalent to the commutativity of (3):

$$\begin{array}{ccc}
X^{\mathbb{N}} & \xrightarrow{\langle head, tail\rangle} & X \times X^{\mathbb{N}} \\
{\scriptstyle [nats, squares]}\uparrow & (3) & \uparrow{\scriptstyle X \times [nats, squares]} \\
Q & \xrightarrow[\langle head^Q, tail^Q\rangle]{} & X \times Q
\end{array}$$

Corecursion and coproduct: Insertion into a stream (see [162])

The function $insert : X \times X^{\mathbb{N}} \to X^{\mathbb{N}}$ satisfies the equation

$$\begin{array}{rcl}
\langle head, tail\rangle(insert(x,s)) & = & if\ \ x \le head(s)\ \ then\ (x,s) \\
& & \qquad\qquad\qquad else\ (head(s), insert(x, tail(s)))
\end{array}$$

This equation does not imply that the image of *insert* is compatible with *head* and *tail*.

Therefore, we transform them into equations for *insert and* the identity on $X^{\mathbb{N}}$:

$$\begin{aligned} \langle head, tail\rangle(insert(x,s)) &= if\ x \leq head(s) \\ &\quad then\ (x, id(s))\ else\ (head(s), insert(x, tail(s))) && (1) \\ \langle head, tail\rangle(id(s)) &= (head(s), id(tail(s))) && (2) \end{aligned}$$

Define $\mathcal{K} = Set^2$ and for all $A, B \in Set$, $R(A)_{list} = (A_{list}, A_{list})$ and $L(A,B)_{list} = A_{list} + B_{list}$.

Let $Q = (X \times X^{\mathbb{N}}) + X^{\mathbb{N}}$. By (1)-(3), the image of

$$(insert, id)^* = [insert, id] : Q \to X^{\mathbb{N}}$$

is compatible with *head* and *tail*.

Hence $(insert, id) : (X \times X^{\mathbb{N}}, X^{\mathbb{N}}) \to (X^{\mathbb{N}}, X^{\mathbb{N}})$ is *Stream*-corecursive and thus by Lemma IMG (1), $[insert, id]$ agrees with $unfold^Q$ where for all $x \in X$ and $s \in X^{\mathbb{N}}$,

$$\langle head^Q, tail^Q\rangle(x,s) = \begin{cases} (x,s) & \text{if } x \leq head(s), \\ (head(s), (x, tail(s))) & \text{otherwise,} \end{cases}$$
$$\langle head^Q, tail^Q\rangle(s) = (head(s), tail(s)).$$

The validity of (1)-(3) is equivalent to the commutativity of (4):

$$\begin{array}{ccc}
X^{\mathbb{N}} & \xrightarrow{\langle head, tail \rangle} & X \times X^{\mathbb{N}} \\
{\scriptstyle [insert, id]} \uparrow & (4) & \uparrow {\scriptstyle X \times [insert, id]} \\
Q & \xrightarrow[\langle head^Q, tail^Q \rangle]{} & X \times Q
\end{array}$$

Corecursion and coproduct: Concatenation of colists (see [77])

The function $conc : X^\infty \times X^\infty \to X^\infty$ satisfies the equations

$$split(s) = \epsilon \wedge split(s') = \epsilon \;\Rightarrow\; split(conc(s, s')) = \epsilon \tag{1}$$

$$split(s) = \epsilon \wedge split(s') = (x, s'') \;\Rightarrow\; split(conc(s, s')) = (x, id(s'')) \tag{2}$$

$$split(s) = (x, s'') \;\Rightarrow\; split(conc(s, s')) = (x, conc(s'', s')) \tag{3}$$

Define $\mathcal{K} = Set^2$ and for all $A, B \in Set$, $R(A)_{list} = (A_{list}, A_{list})$ and $L(A, B)_{list} = A_{list} + B_{list}$.

Let $Q = X^\infty \times X^\infty + X^\infty$. By (1)-(3), the image of $(conc, id)^* = [conc, id] : Q \to X^\infty$

is compatible with $split$: Let $h = [conc, id]$.

$$\begin{array}{l}
split(s) = \epsilon \wedge split(s') = \epsilon \;\Rightarrow\; split(h(s,s')) = \epsilon = h(\epsilon),\\
split(s) = \epsilon \wedge split(s') = (x, s'')\\
\qquad\qquad\qquad\qquad \Rightarrow\; split(h(s,s')) = (x, h(s'')) = (h(x), h(s'')) = h(x, s''),\\
split(s) = (x, s'') \;\Rightarrow\; split(h(s,s')) = (x, h(s'', s')) = (h(x), h(s'', s')) = h(x, (s'', s')),
\end{array}$$

i.e., the image of h is compatible with $split$. Hence $(conc, id)$ is $coList(X)$-corecursive and thus by Lemma IMG (1), $(conc, id)$ agrees with $unfold^Q$ where for all $s, s' \in X^\infty$,

$$\begin{array}{rcl}
split^Q(s,s') & = & \begin{cases} * & \text{if } split(s) = split(s') = \epsilon,\\ (x, (s, s'')) & \text{if } split(s) = \epsilon \wedge split(s') = (x, s''),\\ (x, (s'', s')) & \text{if } split(s) = (x, s''), \end{cases}\\
split^Q(s) & = & split(s).
\end{array}$$

The validity of (1)-(3) is equivalent to the commutativity of (4):

$$\begin{array}{ccc}
X^\infty & \xrightarrow{\;split\;} & 1 + X \times X^\infty\\
{\scriptstyle [conc, id]}\uparrow & (4) & \uparrow{\scriptstyle 1 + X \times [conc, id]}\\
Q & \xrightarrow[\;split^Q\;]{} & 1 + X \times Q
\end{array}$$

Corecursion and coproduct: Flatten a cotree

Let $T = \nu coTree(X)$ (see Trees). The functions $flatten : T \to X^\infty$ and $flattenL : T^\infty \to X^\infty$ satisfy the equations

$$split(flatten(t)) = (root(t), flattenL(subtrees(t))) \quad (1)$$
$$split(ts) = \epsilon \;\Rightarrow\; split(flattenL(ts)) = \epsilon \quad (2)$$
$$split(ts) = (u, us)$$
$$\Rightarrow\; split(flattenL(ts)) = (root(u), flattenL(conc(subtrees(u), us)) \quad (3)$$

where $conc : T^\infty \times T^\infty \to T^\infty$ is defined as in chapter 12.

Define $\mathcal{K} = Set^2$ and for all $A, B \in \mathcal{L}$, $R(A)_{list} = (A_{list}, A_{list})$ and $L(A, B)_{list} = A_{list} + B_{list}$.

By (1)-(3), the image of

$$(flatten, flattenL)^* = [flatten, flattenL] : T + T^\infty \to X^\infty$$

is compatible with *split*.

Hence $(flatten, flattenL) : (T, T^\infty) \to (X^\infty, X^\infty)$ is $coList(X)$-corecursive and thus by Lemma IMG (1), $[flatten, flattenL]$ agrees with $unfold^{T+T^\infty}$ where for all $t \in T$ and $ts \in T^\infty$,

$$split^{T+T^\infty}(t) = (root(t), subtrees(t)),$$
$$split^{T+T^\infty}(ts) = \begin{cases} * & \text{if } split(ts) = \epsilon, \\ (u, us) & \text{if } split(ts) = (root(u), conc(subtrees(u), us)). \end{cases}$$

The validity of (1)-(3) is equivalent to the commutativity of (4):

$$\begin{array}{ccc} X^\infty & \xrightarrow{\quad split \quad} & 1 + X \times X^\infty \\ \uparrow {\scriptstyle [flatten, flattenL]} & (4) & \uparrow {\scriptstyle 1 + X \times [flatten, flattenL]} \\ T + T^\infty & \xrightarrow[\quad split^{T+T^\infty} \quad]{} & 1 + X \times (T + T^\infty) \end{array}$$

Corecursion and identity: Mirror a cobintree (see [74, ?])

Let $T = \nu coBintree(X)_{btree}$. The function $mirror : T \to T$ satisfies the equations

$$split(t) = \epsilon \;\Rightarrow\; split(mirror(t)) = \epsilon \quad (1)$$

$$split(t) = (u, x, u') \;\Rightarrow\; split(mirror(t)) = (mirror(u'), x, mirror(u)) \quad (2)$$

Define $\mathcal{K} = Set$ and $R = L = Id_{Set}$.

??? Extend *mirror* to the sets X and 1. Then (1) and (2) read as follows:

$$split(t) = \epsilon \;\Rightarrow\; split(mirror(t)) = \epsilon = mirror(\epsilon),$$

$$split(t) = (u, x, u')$$

$$\Rightarrow\; split(mirror(t)) = (mirror(u'), mirror(x), mirror(u)) = mirror(u', x, u),$$

Hence the image of *mirror* is compatible with *split*.

Hence *mirror* is *coBintree*(X)-corecursive and thus by Lemma IMG (1), *mirror* agrees with $unfold^T$ where for all $t \in T$,

$$split^T(t) = \begin{cases} * & \text{if } t = \Omega, \\ (\lambda w.t(1w), t(\epsilon), \lambda w.t(0w)) & \text{otherwise.} \end{cases}$$

The validity of (1) and (2) is equivalent to the commutativity of (3):

$$\begin{array}{ccc} T & \xrightarrow{\quad split \quad} & 1 + T \times X \times T \\ {\scriptstyle mirror}\uparrow & (3) & \uparrow{\scriptstyle 1 + mirror \times X \times mirror} \\ T & \xrightarrow[\quad split^T \quad]{} & 1 + T \times X \times T \end{array}$$

Since T is a final algebra, properties of $mirror^T$ like $mirror^T \circ mirror^T = id_T$ are shown by algebraic coinduction (see, e.g., [**?**]).

Restriction with a greatest invariant: Length of a colist

Let $C = \{length\}$. $\nu coList'$ is isomorphic to the $coList'$-coalgebra $B =_{def} Tree_{coList,C}(BA)$ of C-labelled $coList$-trees over BA.

B_{list} can be represented as the union of $\mathbb{N}'$ and the set of partial functions $s : \mathbb{N} \to X \times \mathbb{N}'$ such that $s(0)$ is defined and for all $i \in \mathbb{N}$, if $s(i+1)$ is defined, then $s(i)$ is defined.

With respect to this interpretation, the destructors of $coList'$ are interpreted as follows: $B_1 = \{\omega\}$ and for all $s \in B_{list}$,

$$split^B(s) = \begin{cases} * & \text{if } s \in \mathbb{N}', \\ (\pi_1(s(0)), \lambda i.s(i+1)) & \text{otherwise,} \end{cases}$$
$$length^B(s) = \begin{cases} s & \text{if } s \in \mathbb{N}', \\ \pi_2(s(0)) & \text{otherwise.} \end{cases}$$

Let AX be given by the $coList'$-formulas

$$is_{list}(s) \Rightarrow is_{1+entry\times list}([[x,y]split]s) \tag{1}$$
$$is_{entry\times list}(p) \Rightarrow is_{list}(\pi_2\langle p\rangle) \tag{2}$$
$$is_{list}(s) \Rightarrow [[x,y]length]s = [[[x]0,[[[x]succ,y]length]\pi_2]split]s \tag{3}$$

AX consists of inverse Horn clauses over $coList'$ that satisfy the assumptions of Restriction with a greatest invariant. Hence $gfp(\overline{AX}) = B$. Let $inv = \in^B$.

For all $s, s' \in is_{list}$,

$$length^B(s) \neq length^B(s') \text{ implies } t^B(s) \neq t^B(s') \text{ for some } t \in Obs_{coList,list}. \tag{4}$$

Proof.

Since B satisfies (3), inv satisfies the conclusion of (3) or, equivalently, the equations (1)-(3) of 1.6.

Hence $s \in is_{list}$ iff for all $n \in \mathbb{N}$,

$$length^B(s) = 0 \;\text{ implies }\; split^B(s) = \epsilon, \tag{5}$$
$$length^B(s) = n+1 \;\text{ implies }\; \exists\, e, s' : (split^B(s) = (e, s') \wedge length^B(s') = n), \tag{6}$$
$$length^B(s) = \omega \;\text{ implies }\; \exists\, e, s' : (split^B(s) = (e, s') \wedge length^B(s') = \omega). \tag{7}$$

It is easy to see that

- $Obs_{coList,list} = \{obs_n \mid n \in \mathbb{N}\}$ where $obs_0 = [0, [10]\pi_1]split$ and for all $n > 0$, $obs_n = [0, [10 \cdot obs_{n-1}]\pi_2]split$,
- for all $s \in B_{list}$ and $n \in \mathbb{N}$, $obs_n(s) \neq *$ iff $s(n)$ is defined. (8)

By (5)-(7) and the definition of B, for all $s \in is_{list}$ and $n \in \mathbb{N}$,

$$length^B(s) = n \;\Leftrightarrow\; s(n) \text{ is undefined } \wedge \forall\, i < n : s(i) \text{ is defined},$$
$$length^B(s) = \omega \;\Leftrightarrow\; \forall\, n \in \mathbb{N} : s(n) \text{ is defined},$$

and thus by (8),

$$length^B(s) = n \;\Leftrightarrow\; obs_n^B(s) = \epsilon \wedge \forall\, i < n : obs_i^B(s) \neq *, \tag{9}$$
$$length^B(s) = \omega \;\Leftrightarrow\; \forall\, n \in \mathbb{N} : obs_n^B(s) \neq *. \tag{10}$$

Let $s, s' \in B_{list}$ such that $length^B(s) \neq length^B(s')$. Then $length^B(s) = n$ or $length^B(s') = n$ for some $n \in \mathbb{N}$. W.l.o.g. suppose that the first case holds true. By (9), $obs_n^B(s) = \epsilon$. If $length^B(s') = \omega$, then (10) implies a contradiction: $obs_n^B(s) \neq * =$

$obs_n^B(s)$. Otherwise $length^B(s') = n'$ for some $n' \in \mathbb{N}$ with $n' \neq n$. Let $m = min(n, n')$. If $n < n'$, then by (9), $obs_m^B(s) = obs_n^B(s) = \epsilon \neq obs_n^B(s') = obs_m^B(s')$. Otherwise $n' < n$ and thus by (9), $obs_m^B(s') = obs_{n'}^B(s') = \epsilon \neq obs_{n'}^B(s) = obs_m^B(s)$. Hence (4) is valid for $t = obs_m$. ❏

Destructor extension: Flatten a cotree

We have shown that there is a unique interpretation in $\nu coList(X)$ of additional destructors $flatten : tree \to list$ and $flattenL : trees \to list$ such that the corresponding extension of $\nu coTree$ satisfies the equations (1)-(3) of 2.12.

Let $coTree' = coTree \cup \{flatten, flattenL\}$. By Lemma DESEXT, $coTree'$ is a conservative extension of $coTree$.

Let $C = \{flatten, flattenL\}$. $\nu coTree'$ is isomorphic to the $coTree'$-coalgebra $B =_{def} Tree_{coTree,C}(BA)$ of C-labelled $coTree$-trees over BA.

B_{tree} can be represented as the set of partial functions

$$t : \mathbb{N}^* \to X \times B_{list}$$

(see 2.3) such that $t(\epsilon)$ is defined and for all $w \in \mathbb{N}^*$ and $i \in \mathbb{N}$,

- if $t(wi)$ is defined, then $t(w)$ is defined,
- if $t(w(i+1))$ is defined, then $t(wi)$ is defined.

B_{trees} can be represented as the union of B_{list} and the set of partial functions

$$ts : \mathbb{N} \to B_{tree} \times B_{list}$$

such that $ts(0)$ is defined and for all $i \in \mathbb{N}$, if $ts(i+1)$ is defined, then $ts(i)$ is defined. With respect to this interpretation, the destructors of $coTree'$ are interpreted as follows: For all $t \in B_{tree}$ and $ts \in B_{trees}$,

$$\begin{array}{rcl}
root^B(t) & = & \pi_1(t(\epsilon)), \\
subtrees^B(t) & = & \lambda i.\lambda w.t(iw), \\
\mathit{flatten}^B(t) & = & \pi_2(t(\epsilon)), \\
split^B(ts) & = & \begin{cases} * & \text{if } ts \in B_{list}, \\ (\pi_1(ts(0)), \lambda i.ts(i+1)) & \text{otherwise}, \end{cases} \\
\mathit{flattenL}^B(ts) & = & \begin{cases} ts & \text{if } ts \in B_{list}, \\ \pi_2(ts(0)) & \text{otherwise.} \end{cases}
\end{array}$$

Let AX be given by the $coTree'$-formulas

$$\begin{aligned}
is_{tree}(t) &\Rightarrow is_{trees}(subtrees\langle t\rangle) & (1)\\
is_{trees}(ts) &\Rightarrow is_{1+tree\times trees}([[y,z]split]ts) & (2)\\
is_{tree\times trees}(p) &\Rightarrow is_{tree}(\pi_1\langle p\rangle)\ \wedge is_{trees}(\pi_2\langle p\rangle) & (3)\\
is_{tree}(t) &\Rightarrow \exists\ p : ([[y,z]split]flatten\langle t\rangle = [z]p \wedge \pi_1\langle p\rangle = root\langle t\rangle\ \wedge \\
&\qquad \pi_2\langle p\rangle = flattenL\langle subtrees\langle t\rangle\rangle) & (4)\\
is_{trees}(ts) &\Rightarrow \exists\ p,q : ([[y,z]split]ts = [y]p \wedge [[y,z]split]flattenL\langle ts\rangle = [y]q) \vee \\
&\quad\ \exists\ p,q : ([[y,z]split]ts = [z]p \wedge [[y,z]split]flattenL\langle ts\rangle = [z]q\ \wedge \\
&\qquad\quad \pi_1\langle q\rangle = root\langle \pi_1\langle p\rangle\rangle\ \wedge \\
&\qquad\quad \pi_2\langle q\rangle = flattenL\langle conc\langle subtrees\langle\pi_1\langle p\rangle\rangle,\pi_2\langle p\rangle\rangle\rangle) & (5)
\end{aligned}$$

AX consists of inverse Horn clauses over $coTree'$ that satisfy the assumptions of Restriction with a greatest invariant. Hence $gfp(\overline{AX}) = B$. Let $inv = \ \in^B$.

For all $t,t' \in is_{tree}$,
$flatten^B(t) \neq flatten^B(t')$ implies $u^B(t) \neq u^B(t')$ for some $u \in Obs_{coTree,tree}$. (6)
For all $ts,ts' \in is_{trees}$,
$flattenL^B(ts) \neq flattenL^B(ts')$ implies $u^B(ts) \neq u^B(ts')$ for some $u \in Obs_{coTree,trees}$. (7)

Proof.
Since B satisfies (4) and (5), inv satisfies the conclusions of (4) and (5) or, equivalently,

the equations (1)-(3) of 2.12. Hence $t \in is_{tree}$ iff

$$flatten^B(t) = (root^B(t), flattenL^B(subtrees^B(t))), \qquad (8)$$

and $ts \in is_{trees}$ iff for all $u \in B_{tree}$ and $us \in B_{trees}$,

$$split^B(ts) = \epsilon \text{ implies } split^B(flattenL^B(ts)) = \epsilon, \qquad (9)$$

$$split^B(ts) = (u, us) \text{ implies } flattenL^B(ts) = (root^B(u), flattenL^B(conc^B(subtrees^B(u), us))). \qquad (10)$$

It is easy to see that

- $Obs_{coTree,tree} = \{obs_w \mid w \in \mathbb{N}^*\}$ where $obs_\epsilon = \{[0]root\}$ and for all $w \in \mathbb{N}^+$, $obs_w = [0 \cdot obsL_w]subtrees$,
- $Obs_{coTree,trees} = \{obsL_w \mid w \in \mathbb{N}^+\}$ where for all $i > 0$ and $w \in \mathbb{N}^*$, $obsL_{0w} = [0, [10 \cdot obs^B_w]\pi_1]split$ and $obsL_{iw} = [0, [10 \cdot obsL_{(i-1)w}]\pi_2]split$,
- for all $t \in B_{tree}$ and $w \in \mathbb{N}^*$, $obs^B_w(t) = t(w)$ if $t(w)$ is defined, and $obs^B_w(t) = \epsilon$ otherwise, (11)
- for all $ts \in B_{trees}$, $i \in \mathbb{N}$ and $w \in \mathbb{N}^+$, $obsL_{iw}(ts) = ts(i)(w)$ if $ts(i)(w)$ is defined, and $obsL_{iw}(ts) = \epsilon$ otherwise. (12)

By (8)-(10) and the definition of B, for all $t \in is_{tree}$, $ts \in is_{trees}$ and $s \in B_{list}$,

$$flatten^B(t) = s \;\Leftrightarrow\; \forall\, n \in domain(s) : t(leafPos(t)(n)) = s(n),$$
$$flattenL^B(ts) = s \;\Leftrightarrow\; \forall\, n \in domain(s) : ts(i)(w) = s(n) \text{ where } leafPosL(ts)(n) = iw,$$

and thus by (11) and (12),

$$flatten^B(t) = s \;\Leftrightarrow\; \forall\, n \in domain(s) : obs^B_{leafPos(t)(n)}(t) = s(n), \tag{13}$$

$$flattenL^B(ts) = s \;\Leftrightarrow\; \forall\, n \in domain(s) : obsL^B_{leafPosL(ts)(n)}(ts) = s(n), \tag{14}$$

where $leafPos(t)(n)$ and $leafPosL(ts)(n)$ are the positions of the n-th leaf of t and ts, respectively.

Haskell code for $leafPos : B_{tree} \to \mathbb{N} \to \mathbb{N}^*$ and $leafPosL : B_{trees} \to \mathbb{N} \to \mathbb{N}^+$:

```
leafPos  = (!!) . leafPoss
leafPosL = (!!) . leafPossL

leafPoss :: B_tree -> [[Int]]
leafPoss t = if null ts then [[]] else leafPossL ts
             where ts = subtrees t

leafPossL :: B_trees -> [[Int]]
leafPossL ts = if null ts then [] else concatMap g [0..length ts-1]
               where g i = map (i:) $ leafPoss $ ts!!i
```

Let $t, t' \in B_{tree}$ and $s, s' \in B_{list}$ such that $flatten^B(t) = s \neq s' = flatten^B(t')$. Let $domain(t) \neq domain(t')$. Then there is $w \in \mathbb{N}^*$ such that $t(w)$ is defined and $t'(w)$ is undefined. Hence by (11), $obs^B_w(t) = t(w)$ and $obs^B_w(t') = \epsilon$, and thus (6) is valid for $u = obs_w$. Let $domain(t) = domain(t')$. Then $domain(s) = domain(s')$ and there is

$n \in domain(s)$ such that $s(n) \neq s'(n)$ and for all $i < n$, $s(i) = s'(i)$. By (13),

$$obs^B_{leafPos(t)(n)}(t) = s(n) \neq s'(n) = obs^B_{leafPos(t')(n)}(t') = obs^B_{leafPos(t)(n)}(t').$$

Hence (6) is valid for $u = obs_{leafPos(t)(n)}$.

Let $ts, ts' \in B_{trees}$ and $s, s' \in B_{list}$ such that $flattenL^B(ts) = s \neq s' = flattenL^B(ts')$. Let $domain(ts) \neq domain(ts')$ or $domain(ts(i)) \neq domain(ts'(i))$ for some $i \in domain(ts) = domain(ts')$. Then there are $i \in \mathbb{N}$ and $w \in \mathbb{N}^*$ such that $ts(i)(w)$ is defined and $ts'(i)(w)$ is undefined. Hence by (12), $obsL^B_{iw}(ts) = ts(i)(w)$ and $obsL^B_{iw}(ts') = \epsilon$, and thus (7) is valid for $t = obs_{iw}$. Let $domain(ts) = domain(ts')$ and for all $i \in domain(ts)$, $domain(ts(i)) = domain(ts'(i))$. Then $domain(s) = domain(s')$ and there is $n \in domain(s)$ such that $s(n) \neq s'(n)$. By (14),

$$obs^B_{leafPosL(ts)(n)}(ts) = s(n) \neq s'(n) = obs^B_{leafPosL(ts')(n)}(ts') = obs^B_{leafPos(ts)(n)}(ts').$$

Hence (7) is valid for $u = obs_{leafPosL(ts)(n)}$. ❏

Let $\in^A = \nu coTree$. Then A satisfies AX. Hence $A \in Alg_{coTree',AX}$ and thus by Lemma DESEXT, (6) and (7) imply $\in^B|_{coTree} \cong \nu coTree$.

Destructor extension: Subtree of a cobintree

Let $C = \{subtree\}$. $\nu coBintree'$ is isomorphic to the $coBintree'$-coalgebra

$$B =_{def} Tree_{coBintree,C}(BA)$$

of C-labelled $coBintree$-trees over BA.

Let $Z = Btree(X)^\infty \to Btree(X)^\infty$. B_{btree} can be represented as the set of partial functions

$$t : 2^* \to X \times Z$$

such that for all $w \in 2^*$ and $b \in 2$, if $t(wb)$ is defined, then $t(w)$ is defined.

With respect to this interpretation, the destructors of $coBintree'$ are interpreted as follows: For all $t \in B_{tree}$,

$$\begin{aligned} fork^B(t) &= \begin{cases} * & \text{if } t = \Omega, \\ (\lambda w.t(0w), \pi_1(t(\epsilon)), \lambda w.t(1w)) & otherwise, \end{cases} \\ subtree^B(t) &= \pi_2(t(\epsilon)). \end{aligned}$$

Let AX be given by the $coBintree'$-formulas

$$\begin{aligned}
is_{btree}(t) &\Rightarrow is_{1+btree\times entry\times btree}(fork\langle t\rangle) \ \wedge is_{btree^{blist}}(subtree\langle t\rangle) & (1)\\
is_{btree\times entry\times btree}(p) &\Rightarrow is_{btree}(\pi_1\langle p\rangle) \ \wedge is_{btree}(\pi_3\langle p\rangle) & (2)\\
is_{btree^{blist}}(f) &\Rightarrow is_{btree}(\$w\langle f\rangle) & (3)\\
is_{btree}(t) &\Rightarrow \exists\ p,q : ([[x,y]fork]t = [x]p \wedge \$\epsilon\langle subtree\langle t\rangle\rangle = t) \vee \\
&\quad\ \exists\ p,q : ([[x,y]fork]t = [y]p \wedge \\
&\qquad\qquad \$0w\langle subtree\langle t\rangle\rangle = \$w\langle subtree\langle \pi_1\langle p\rangle\rangle\rangle \wedge \\
&\qquad\qquad \$1w\langle subtree\langle t\rangle\rangle = \$w\langle subtree\langle \pi_3\langle p\rangle\rangle\rangle) & (4)
\end{aligned}$$

for all $w \in 2^*$.

AX consists of inverse Horn clauses over $coBintree'$ that satisfy the assumptions of Restriction with a greatest invariant. Hence $gfp(\overline{AX}) = B$. Let $inv = \in^B$.

For all $t, t' \in is_{btree}$,
$subtree^B(t) \neq subtree^B(t')$ implies $u^B(t) \neq u^B(t')$ for some $u \in Obs_{coBintree,btree}$. (5)

Proof.

Since B satisfies (4), inv satisfies the conclusion of (4) or, equivalently, the definition of subtree given in example 18 ****. Hence $t \in is_{btree}$ iff for all $w \in 2^*$,

$$\begin{aligned}
&subtree^B(t)(\epsilon) = t, & (6)\\
&fork^B(t) = (u,e,u') \ \text{ implies } \ subtree^B(t)(0\!:\!w) = subtree^B(u)(w), & (7)\\
&fork^B(t) = (u,e,u') \ \text{ implies } \ subtree^B(t)(1\!:\!w) = subtree^B(u')(w). & (8)
\end{aligned}$$

It is easy to see that

- $Obs_{coBintree,btree} = \{obs_w \mid w \in 2^+\}$ where $obs_\epsilon = [0,[10]\pi_2]fork$ and for all $w \in \mathbb{N}^+$, $obs_{0w} = [0,[10 \cdot obs_w]\pi_1]fork$ and $obs_{1w} = [0,[10 \cdot obs_w]\pi_3]fork$,
- for all $t \in B_{tree}$ and $w \in \mathbb{N}^*$, $obs_w^B(t) = t(w)$ if $t(w)$ is defined, and $obs_w(t) = \epsilon$ otherwise. (9)

By (6)-(8) and the definition of B, for all $t \in is_{btree}$ and $v \in 2^*$,

$$subtree^B(t)(v) = \lambda w.t(vw),$$

and thus by (9),

$$subtree^B(t)(v) = \lambda w.obs_{vw}(t). \tag{10}$$

Let $t, t' \in B_{btree}$ and $w \in 2^*$ such that $subtree^B(t) \neq subtree^B(t')$. Then there are $v, w \in 2^*$ such that $subtree^B(t)(v)(w) \neq subtree^B(t)(v)(w)$. Hence by (10), $\lambda w.obs_{vw}(t) \neq \lambda w.obs_{vw}(t)$, and thus (5) is valid for $u = obs_{vw}^B$. ❏

Coiterative equations

Let $\Sigma = (S, \mathcal{I}, D)$ be a destructive signature and V be a finite S-sorted set. An S-sorted function

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

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

Let $\mathcal{A}$ be a Σ-algebra with carrier A, A^V be the set of S-sorted functions from V to A und $B = \bigcup BT$.

$g \in A^V$ **solves E in $\mathcal{A}$** if for all $x \in V$ $id_A^\#(g(x)) = [g, id_B] \circ E(x)$ (see State unfolding).

E turns $T_\Sigma(V)$ into a Σ-algebra: Let $s \in S$, I be a nonempty set and $(e_i)_{i \in I} \in \mathcal{T}_p(S, \mathcal{I})^I$.

- For all $d : s \to e \in D$ and $x \in V_s$, $d^{T_\Sigma(V)}(x) = E(x)(d)$.
- For all $d : s \to e \in D$ and $t_{d'} \in T_\Sigma(V)_{e'}$, $d' : s \to e' \in D$,

$$s^{T_\Sigma(V)}(\epsilon\{d' \to t_{d'} \mid d' : s \to e' \in D\}) =_{def} t_d.$$

- For all $t_i \in T_\Sigma(V)_{e_i}$, $i \in I$, and $k \in I$, $\pi_k(tup\{i \to t_i \mid i \in I\}) =_{def} t_k$.
- For all $i \in I$ and $t \in T_\Sigma(V)_{e_i}$, $\iota_i(t) =_{def} i\{sel \to t\}$.

Let $g = V \stackrel{inc_V}{\to} T_\Sigma(V) \stackrel{unfold^{T_\Sigma(V)}}{\longrightarrow} DT_\Sigma$.

(1) g solves E in DT_Σ.

(2) Any $g : V \to DT_\Sigma$ solves E in DT_Σ iff ????.

Theorem COSOL E has a unique solution in DT_Σ.

Proof. By (1), E has a solution in DT_Σ. Suppose that $g, h : V \to DT_\Sigma$ solve E in DT_Σ. By (2), ???. Since DT_Σ is final in Alg_Σ, ???. ❑

Terms as functions

Let $\Sigma = (S, \mathcal{I}, F)$ be a signature, V be an S-sorted set of variables,

$$\begin{array}{rcl} CON &=& \{c : 1 \to X \mid c \in X \in Set_{\neq\emptyset}\}, \\ VAR &=& \{x : 1 \to e \mid x \in V_e,\ e \in \mathcal{T}_p(S,\mathcal{I})\}, \\ ID &=& \{id : e \to e \mid e \in \mathcal{T}_p(S,\mathcal{I})\}, \\ APP &=& \{\$x : e^X \to e \mid e \in \mathcal{T}_p(S,\mathcal{I}),\ x \in X \in Set_{\neq\emptyset}\}, \\ ISO &=& \{\underline{\tau} : e \to e' \mid \tau : F_e \to F_{e'} \text{ is a natural equivalence},\ e \neq e'\}. \end{array}$$

The set $Op_\Sigma(V)$ of (derived) $\Sigma(V)$-**operations** is defined inductively as follows:

- $F \cup CON \cup VAR \cup ID \cup ISO \cup INJ \cup PRJ \cup APP \subseteq Op_\Sigma(V)$.
- For all $t : e \to e', u : e' \to e'' \in Op_\Sigma(V)$, $red(u \circ t) : e \to e'' \in Op_\Sigma(V)$ where $red(u \circ t)$ is reduced with respect to the following rewrite rules:

$$\begin{array}{rcll} \pi_i \circ \langle t_1, \dots, t_n \rangle &\to& t_i, & 1 \le i \le n, \\ [t_1, \dots, t_n] \circ \iota_i &\to& t_i, & 1 \le i \le n, \\ \langle u_1, \dots, u_n \rangle \circ t &\to& \langle u_1 \circ t, \dots, u_n \circ t \rangle, & \\ u \circ [t_1, \dots, t_n] &\to& [u \circ t_1, \dots, u \circ t_n], & \\ (\$t) \circ \lambda x.u &\to& u[t/x] & \end{array}$$

  where $t[z/x]$ denotes the Σ-operation obtained from t by replacing all occurrences of x with z.

- For all $n > 1$ and $t_1 : e \to e_1, \dots, t_n : e \to e_n \in Op_\Sigma(V)$, $\langle t_1, \dots, t_n \rangle : e \to e_1 \times \dots \times e_n \in Op_\Sigma(V)$.
- For all $n > 1$ and $t_1 : e_1 \to e, \dots, t_n : e_n \to e \in Op_\Sigma(V)$, $[t_1, \dots, t_n] : e_1 + \dots + e_n \to e \in Op_\Sigma(V)$.
- For all $c \in \{word, set, bag\}$ and $t : e \to e' \in Op_\Sigma(V)$, $c(t) : c(e) \to c(e') \in Op_\Sigma(V)$.
- For all $t : e \to e' \in Op_\Sigma(V)$, and $X \in \mathcal{T}_p(S, \mathcal{I})$, $t^X : e^X \to (e')^X \in Op_\Sigma(V)$.
- For all $t : e \to e' \in Op_\Sigma(V)$, $X \in \mathcal{T}_p(S, \mathcal{I})$ and $x \in X$, $\lambda x.t : e \to (e')^X \in Op_\Sigma(V)$.

Moreover, for all $n > 1$ and $t_1 : e \to e_1, \dots, t_n : e \to e_n \in Op_\Sigma(V)$,

$$\begin{aligned} t_1 \times \dots \times t_n &=_{def} \langle t_1 \circ \pi_1, \dots, t_n \circ \pi_n \rangle, \\ t_1 + \dots + t_n &=_{def} [\iota_1 \circ t_1, \dots, \iota_n \circ t_n], \end{aligned}$$

and for all $p : e \to 2,\ t, u : e \to e' \in Op_\Sigma(V)$,

$$if\ p\ then\ t\ else\ u \quad =_{def} \quad e \xrightarrow{\langle id_e, p \rangle} e \times 2 \xrightarrow{\tau} e + e \xrightarrow{[t,u]} e'$$

where $\tau : F_{e \times 2} \to F_{e+e}$ is the natural equivalence defined as follows: For all $A \in Set^S$,

$$\begin{aligned} \tau_A : A_e \times 2 &\to A_e + A_e \\ (a, b) &\mapsto \begin{cases} (a, 1) & \text{if } b = 1 \\ (a, 2) & \text{if } b = 0. \end{cases} \end{aligned}$$

For all $t : e \to e' \in Op_\Sigma(V)$, $src(t) = e$ is the **domain** and $trg(t) = e'$ the **range** of t.

??? The adjective “implicit” is due to [134, 79] where it is also associated with operations that are not part of the underlying signature.

Given Σ-operations t and u, u is a **suboperation** of t if $t = u$ or there are $n > 1$ and Σ-operations $t_1, \ldots, t_n, u_1, \ldots, u_n$ such that

- $t = t_1 \circ t_2$ and u is a suboperation of t_2 or
- $t = t_1 \circ t_2$, $u = u_1 \circ t_2$ and u_1 is a suboperation of t_1 or
- $t = \langle t_1, \ldots, t_n \rangle$ and there is $1 \leq i \leq n$ such that u is a suboperation of t_i or
- $t = [t_1, \ldots, t_n]$, $u = [u_1, \ldots, u_n]$ and for all $1 \leq i \leq n$, u_i is a suboperation of t_i.

A Σ-algebra A interprets each ground Σ-operation $t : e \to e'$ as a function $t^A : A_e \to A_{e'}$

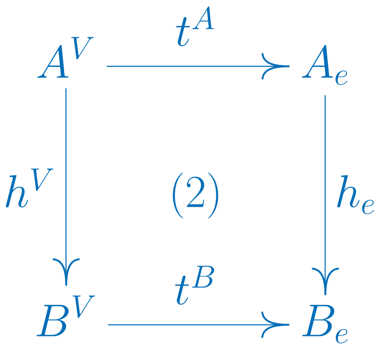
inductively on the structure of t: Let $X \in BS$, $n > 1$ and $e, e', e_1, \dots, e_n \in \mathcal{T}_p(S, \mathcal{I})$.

$$\begin{array}{l}
\forall\ a \in A_e : id^A(a) = a, \\
\forall\ \underline{\tau} \in ISO : e \to e' : \underline{\tau}^A = \tau_A, \\
\forall\ 1 \leq i \leq n,\ a \in A_i : \iota_i(a) = (a, i), \\
\forall\ (a_1, \dots, a_n) \in A_{e_1 \times \dots \times e_n} : \pi_i(a_1, \dots, a_n) = a_i, \\
\forall\ X \in BS,\ x \in X,\ f \in A_{e^X} : (\$x)^A = f(x), \\
\forall\ X \in BS,\ c \in X : c^A(\epsilon) = c, \\
\forall\ t : e \to e', u : e' \to e'' \in Op_\Sigma : (u \circ t)^A = u^A \circ t^A, \\
\forall\ t_1 : e \to e_1, \dots, t_n : e \to e_n \in Op_\Sigma : \langle t_1, \dots, t_n \rangle^A(a) = (t_1^A(a), \dots, t_n^A(a)), \\
\forall\ t_1 : e_1 \to e, \dots, t_n : e_n \to e \in Op_\Sigma,\ (b, i) \in A_{e_1 + \dots + e_n} : [t_1, \dots, t_n]^A(b, i) = t_i^A(b), \\
\forall\ t : e \to e' \in Op_\Sigma,\ a_1, \dots, a_n \in A_e : word(t)^A(a_1, \dots, a_n) = (t^A(a_1), \dots, t^A(a_n)), \\
\forall\ t : e \to e' \in Op_\Sigma,\ f \in \mathcal{P}_\omega(A_e) : set(t)^A(f) = \mathcal{P}_\omega(t^A)(f), \\
\forall\ t : e \to e' \in Op_\Sigma,\ f \in \mathcal{B}_\omega(A_e) : bag(t)^A(f) = \mathcal{B}_\omega(t^A)(f), \\
\forall\ t : e \to e' \in Op_\Sigma,\ f \in A_e^X : (t^X)^A(f) = t^A \circ f, \\
\forall\ t : e \to e' \in Op_\Sigma,\ a \in A_e,\ X \in BS,\ x, z \in X : (\lambda x.t)^A(a)(z) = t[z/x]^A(a).
\end{array}$$

Lemma TERMNAT

For all $e \in \mathcal{T}_p(S, \mathcal{I})$ and $t \in T_\Sigma(V)_e$ we define $t^A : A^V \to A_e$ by $t^A(g) = g^*(t)$ for all

$g \in A^V$. For all Σ-homomorphisms $h : A \to B$, the following diagram commutes:



Hence $\overline{t} : \_^V \to F_e U_S$ with $\overline{t}_A =_{def} t^A$ for all $A \in Alg_\Sigma$ is a natural transformation where U_S is the forgetful functor from Alg_Σ to Set^S.

Proof.

The commutativity of (2) is equivalent to (1): For all $e \in \mathcal{T}_p(S, \mathcal{I})$ and $t \in T_\Sigma(V)_e$,

$$(h \circ g)^*(t) = t^B(h \circ g) = t^B(h^V(g)) \overset{(2)}{=} h_e(t^A(g)) = h_e(g^*(t)). \qquad ❏$$

Derived Σ-operations and $\lambda\Sigma$-terms

Let $\Sigma = (S, \mathcal{I}, F)$ be a signature and V be a $\mathcal{T}(S, \mathcal{I})$-sorted set of variables.

The $\mathcal{T}_p(S,\mathcal{I})^2$-sorted class der_Σ of **derived Σ-operations** is defined inductively as follows:

- $F \subseteq der_\Sigma$. (Σ-operations)
- $Mor(Set_{\neq\emptyset}) \subseteq der_\Sigma$. (functions between nonempty sets)
- For all $e \in \mathcal{T}_p(S,\mathcal{I})$, $id_e : e \to e \in der_\Sigma$. (identities)
- For all $e \in \mathcal{T}_p(S,\mathcal{I})$, $sink_e : e \to 1 \in der_\Sigma$. (sinks)
- For all nonempty sets B and $b \in B$, $b : 1 \to B \in der_\Sigma$. (base constants)
- For all $e, e' \in \mathcal{T}_p(S,\mathcal{I})$ and $t : e \to e' \in cl\lambda T_\Sigma(V)$, $t : e \to e' \in der_\Sigma$. ($\lambda\Sigma$-term; see below)
- For all $f : e \to e', g : e' \to e'' \in der_\Sigma$, $g \circ f : e \to e'' \in der_\Sigma$. (composition)
- For all $f = (f_s : e_s \to e'_s)_{s\in S} \in der_\Sigma^S$ and $e \in \mathcal{T}_p(S,\mathcal{I})$,
  $f_e : e[e_s/s \mid s \in S] \to e[e'_s/s \mid s \in S] \in der_\Sigma$. ($\mathcal{T}_p(S,\mathcal{I})$-congruence)
- For all $I \in BT$, $i \in I$, $\pi_i : \prod_{i\in I} e_i \to e_i \in der_\Sigma$. (projection)
- For all $I \in BT$, $i \in I$, $\iota_i : e_i \to \coprod_{i\in I} e_i \in der_\Sigma$. (injection)
- For all nonempty sets I, $(c_i : e_i \to e)_{i\in I} \in der_\Sigma^I$ and all $(f_i : e_i \to e')_{i\in I} \in der_\Sigma^I$,
  $case\{c_i.f_i\}_{i\in I} : e \to e' \in der_\Sigma$. (case distinction)
- For all nonempty sets I, $(d_i : e \to e_i)_{i\in I} \in der_\Sigma^I$ and all $(f_i : e' \to e_i)_{i\in I} \in der_\Sigma^I$,
  $obj\{d_i.f_i\}_{i\in I} : e' \to e \in der_\Sigma$. (object definition)

For all $f : e \to e' \in der_\Sigma$, $src(f) = e$ and $trg(f) = e'$ is called the **domain** resp. **range**

of f.

Case distinctions and object definitions provide functional versions of the `case`- resp. `merge`-statements of [60]. The following operators are derived from the preceding ones:

- For all $(f_i : e \to e_i)_{i\in I} \in der_\Sigma^I$,
$$\langle f_i\rangle_{i\in I} =_{def} obj\{\pi_i.f_i\}_{i\in I} : e \to \prod_{i\in I} e_i. \qquad \text{(product extension)}$$
- For all $(f_i : e_i \to e)_{i\in I} \in der_\Sigma^I$,
$$[f_i]_{i\in I} =_{def} case\{\iota_i.f_i\}_{i\in I} : \coprod_{i\in I} e_i \to e. \qquad \text{(sum extension)}$$
- For all $(f_i : e_i \to e_i')_{i\in I} \in der_\Sigma^I$,
$$\prod_{i\in I} f_i =_{def} \langle f_i \circ \pi_i\rangle_{i\in I} : \prod_{i\in I} e_i \to \prod_{i\in I} e_i', \qquad \text{(product)}$$
$$\coprod_{i\in I} f_i =_{def} [\iota_i \circ f_i]_{i\in I} : \prod_{i\in I} e_i \to \prod_{i\in I} e_i'. \qquad \text{(sum)}$$

Every S-sorted set A defines a category with $\mathcal{T}_p(S,\mathcal{I})$ as the set of objects and the functions from A_e to $A_{e'}$ as the morphisms from e of e'.

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

- $V \subseteq \lambda T_\Sigma(V)$. (variables)
- For all $f : 1 \to e \in der_\Sigma$, $f : e \in \lambda T_\Sigma(V)$. (derived constant)
- For all $e \in \mathcal{T}_p(S,\mathcal{I}) \setminus \{1\}$ and $f : e \to e' \in der_\Sigma$,
$$f : e \to e' \in \lambda T_\Sigma(V). \qquad \text{(derived operation)}$$
- For all $x : e \in V$ and $t : e' \in \lambda T_\Sigma(V)$, $\lambda x.t : e \to e' \in \lambda T_\Sigma(V)$. ($\lambda$-abstraction)

- For some $(c_i : e_i \to e)_{i\in I} \in der_\Sigma^I$, all $(x_i : e_i)_{i\in I} \in V^I$ and all $(t_i : e')_{i\in I} \in \lambda T_\Sigma(V)^I$,
  $\lambda\{c_i(x_i).t_i\}_{i\in I} : e \to e' \in \lambda T_\Sigma(V).$ (case-based λ-abstraction)
- For all $e, e' \in \mathcal{T}(S)$ and $t : e \to e',\ u : e \in \lambda T_\Sigma(V)$,
  $t(u) : e' \in \lambda T_\Sigma(V).$ (term application)

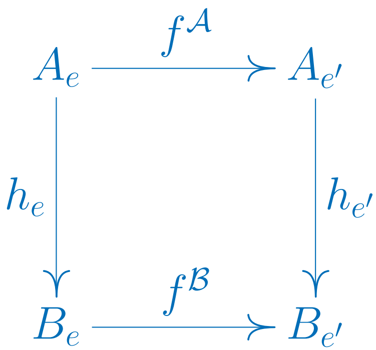
$cl\lambda T_\Sigma(V)$ denotes the set of **closed** $\lambda\Sigma$-terms over V (all variables are bound by λ).

Lemma OPNT

$\mathcal{A} = (A, Op)$ and $\mathcal{B} = (B, Op')$ be Σ-algebras and $f : e \to e' \in der_\Sigma$ such that $f^\mathcal{A}$ and $f^\mathcal{B}$ be defined. For all Σ-homomorphisms $h : \mathcal{A} \to \mathcal{B}$,

$$h_{e'} \circ f^\mathcal{A} = f^\mathcal{B} \circ h_e,$$

i.e., the following diagram commutes:



In other words, f is a natural transformation from $F_e U_S$ to $F_{e'} U_S$ where U_S denotes the forgetful functor from Alg_Σ to Set^S.

Proof. Induction on the structure of f.

The following diagrams (1), (2) and (3) commute: Let $f = case\{c_i.f_i\}_{i \in I}$.

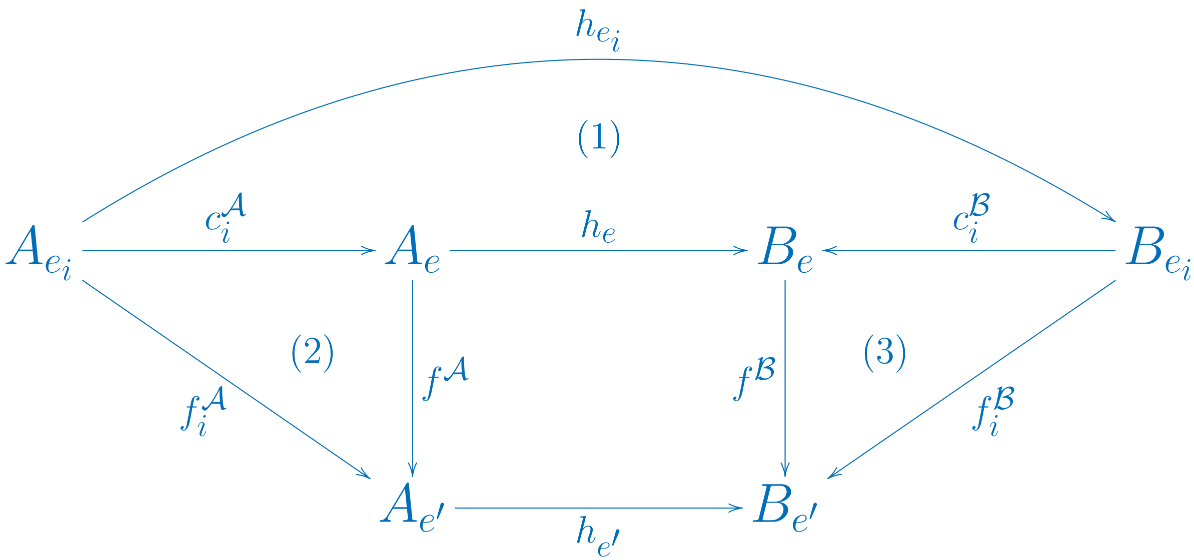


Diagram chasing leads to

$$(case\{c_i.f_i\}_{i \in I})^{\mathcal{B}} \circ h_e \circ c_i^{\mathcal{A}} = h_{e'} \circ (case\{c_i.f_i\}_{i \in I})^{\mathcal{A}} \circ c_i^{\mathcal{A}}$$

for all $i \in I$ and thus to $(case\{c_i.f_i\}_{i \in I})^{\mathcal{B}} \circ h_e = h_{e'} \circ (case\{c_i.f_i\}_{i \in I})^{\mathcal{A}}$. ❏

A **valuation** of a $\mathcal{T}(S)$-sorted set V of variables in A is a $\mathcal{T}(S)$-sorted function $g : V \to A$ such that for all $x : e \in V$, $g(x) \in A_e$. A^V denotes the set of valuations of V in A.

The **extension** $g^* : \lambda T_\Sigma(V) \to A$ **of** $g \in A^V$ **to** $\lambda T_\Sigma(V)$ is defined inductively as follows:

- For all $x \in V$, $g^*(x) = g(x)$.
- For all $f : 1 \to e \in der_\Sigma$, $g^*(f) = f^{\mathcal{A}}(*)$.
- For all $e \in \mathcal{T}_p(S, \mathcal{I}) \setminus \{1\}$, $f : e \to e' \in der_\Sigma$, $g^*(f) = f^{\mathcal{A}}$.
- For all $x : e \in V$, $t : e' \in \lambda T_\Sigma(V)$ and $a \in A_e$, $g^*(\lambda x.t)(a) = g[a/x]^*(t)$.
- For all A-constructor tuples $(c_i : e_i \to e)_{i \in I}$, $(x_i : e_i)_{i \in I} \in V^I$, $(t_i : e')_{i \in I} \in \lambda T_\Sigma(V)^I$ and $(a_i)_{i \in I} \in \times_{i \in I} A_{e_i}$, $g^*(\lambda\{c_i(x_i).t_i\}_{i \in I})(c_i(a_i)) = g[a_i/x_i]^*(t_i)$ is well-defined.
- For all $e, e' \in \mathcal{T}(S)$ and $t : e \to e'$, $u : e \in \lambda T_\Sigma(V)$, $g^*(t(u)) = g^*(t)(g^*(u))$.

For all $t \in cl\lambda T_\Sigma(V)$, $t^{\mathcal{A}} =_{def} g^*(t)$ where g is *any* valuation of V in A: Since t does not contain variables, $g^*(t) = g'^*(t)$ for all $g, g' \in A^V$.

Various notions of terms

1. Finite terms with collections

Let $\Sigma = (BA, S, F, P)$ be a constructive signature, $BA = (BS, BF, BP)$ and V be an $\mathcal{T}(S)$-sorted set whose elements are called **variables**.

$T_\Sigma(V)$ denotes the least $\mathcal{T}(S)$-sorted set such that the following conditions hold true:

- For all $B \in BS$, $T_\Sigma(V)_B = B \cup V_B$.
- For all $e \in \mathcal{T}(S) \setminus BS$, $V_e \subseteq T_\Sigma(V)_e$.
- For all $f : e_1 \times \cdots \times e_n \to s \in F$ and $t_i \in T_\Sigma(V)_{e_i}$, $1 \leq i \leq n$, $f(t_1, \ldots, t_n) \in T_\Sigma(V)_s$.
- For all $c \in Coll$, $s \in S$ and $t \in T_\Sigma(V)_s^*$, $c(t) \in T_\Sigma(V)_{c(s)}$.

Hence $T_\Sigma(V)$ consists of those Σ-terms in the sense of Signatures, which denote objects composed of constructors, and not any terms needed for building up Predicates.

Let $\sim$ be the least $\mathcal{T}(S)$-sorted equivalence relation on $T_\Sigma(V)$ such that

- for all $B \in BS$, $\sim_B = \Delta^2_{B \cup V_B}$,
- for all $s \in S$, $\Delta^2_{V_s}$,

- for all $f : e_1 \times \cdots \times e_n \to s \in F$ and $t_i, t'_i \in T_\Sigma(V)_{e_i}$, $1 \leq i \leq n$,

$$t_1 \sim_{e_1} t'_1 \wedge \cdots \wedge t_n \sim_{e_n} t'_n \quad \Rightarrow \quad f(t_1, \ldots, t_n) \sim_s f(t'_1, \ldots, t'_n),$$

- for all $s \in S$, $n > 1$, $f : [n] \stackrel{\sim}{\to} [n]$ and $t_1, \ldots, t_n, t'_1, \ldots, t'_n \in T_\Sigma(V)_s$,

$$t_1 \sim_s t'_1 \wedge \cdots \wedge t_n \sim_s t'_n \quad \Rightarrow \quad bag(t_{f(1)}, \ldots, t_{f(n)}) \sim_{bag(s)} bag(t'_1, \ldots, t'_n),$$

- for all $s \in S$, $m, n > 0$ and $t_1, \ldots, t_m, t'_1, \ldots, t'_n \in T_\Sigma(V)_s$,

$$\begin{array}{r} \forall\, i \in [m]\ \exists\, j \in [n] : t_i \sim_s t'_j \ \wedge\ \forall\, j \in [n]\ \exists\, i \in [m] : t_i \sim_s t'_j \\ \Rightarrow \quad set(t_1, \ldots, t_m) \sim_{set(s)} set(t'_1, \ldots, t'_n). \end{array}$$

Consequently, for all $B \in BS$,

$$(T_\Sigma(V)/{\sim})_B = T_\Sigma(V)_B/{\sim} \ = (B \cup V_B)/{\sim} \ = B \cup V_B = F_B(T_\Sigma(V)/{\sim}),$$

and for all $c \in Coll$ and $s \in S$,

$$(T_\Sigma(V)/{\sim})_{c(s)} = T_\Sigma(V)_{c(s)}/{\sim} \ \cong F_{c(s)}(T_\Sigma(V)/{\sim}).$$

Hence the S-sorted set $T_\Sigma(V)/{\sim}$ as well as S-sorted functions from $T_\Sigma(V)/{\sim}$ are lifted to an $\mathcal{T}(S)$-sorted set resp. $\mathcal{T}(S)$-sorted functions in the same way Σ-algebras resp. Σ-homomorphisms are lifted.

If Σ does not contain collection types, then ${\sim} = \Delta^2_{T_\Sigma(V)}$ and thus $T_\Sigma(V)/{\sim} = T_\Sigma(V)$.

For ease of notation, we identify $T_\Sigma(V)$ with $T_\Sigma(V)/{\sim}$ and thus each element of $T_\Sigma(V)$ with its equivalence class w.r.t. $\sim$.

The elements of $T_\Sigma(V)$ are called **Σ-terms over V**.

The elements of $T_\Sigma = T_\Sigma(\lambda s.\emptyset)$ are called **ground Σ-terms**.

$T_\Sigma(V)$ is extended to a Σ-algebra as follows:

For all $f : e \to e' \in F$ and $t \in T_\Sigma(V)_e$, $f^{T_\Sigma(V)}(t) =_{def} f(t)$.

$T_\Sigma(V)$ is called the **free Σ-algebra over V**.

2. Infinite terms with collections

Let $\Sigma = (S, \mathcal{I}, F)$ be a constructive signature and $\mathbb{N}_{>0}$ be the set of positive natural numbers.

CT_Σ denotes the greatest $\mathcal{T}(S)$-sorted set of prefix closed partial functions

$$t : \mathbb{N}_{>0}^* \multimap F \cup \bigcup BS \cup Coll$$

such that

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

- for all $c \in Coll$, $s \in S$ 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 $X \in BS$, $CT_{\Sigma,X} = (1 \to X)$.

Let $\sim$ be the greatest $\mathcal{T}(S)$-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 $\lambda w.t(iw) \sim_s \lambda w.t'(iw)$ for all $1 \leq i \leq n_t$,
- for all $s \in S \cup BS$ and $t \sim_{bag(s)} t'$, $n_t = n_{t'}$ and there is $f : [n_t] \xrightarrow{\sim} [n_t]$ with $\lambda w.t(iw) \sim_s \lambda w.t'(f(i)w)$ for all $1 \leq i \leq n_t$,
- for all $s \in S \cup BS$ and $t \sim_{set(s)} t'$ there are $f : [n_t] \to [n_{t'}]$ and $g : [n_{t'}] \to [n_t]$ with $\lambda w.t(iw) \sim_s \lambda w.t'(f(i)w)$ and $\lambda w.t(g(j)w) \sim_s \lambda w.t'(jw)$ for all $1 \leq i \leq n_t$ and $1 \leq j \leq n_{t'}$,
- for all $X \in BS$, $\sim_X = \Delta^2_{1 \to X}$.

Of course, $\sim\ = \Delta^2_{CT_\Sigma}$ and thus $CT_\Sigma/{\sim}\ = CT_\Sigma$ whenever Σ does not include collection types.

T_Σ and $T_\Sigma/{\sim}$ denote the S-sorted sets of **finite** (collection) Σ-trees.

CT_Σ is a Σ-algebra: For all $f : e \to s \in F$, $(t_1,\dots,t_n) \in CT_{\Sigma,e}$, $i > 0$ and $w \in \mathbb{N}^*_{>0}$,

$$f^{CT_\Sigma}(t_1,\dots,t_n)(\epsilon) =_{def} f \quad \text{and} \quad f^{CT_\Sigma}(t_1,\dots,t_n)(iw) =_{def} \begin{cases} t_i(w) & \text{if } 1 \le i \le n, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

3. λ-terms

The S-sorted set $T_\Sigma(V)$ of **Σ-terms over** V is defined inductively as follows:

- For all $e \in \mathcal{T}_p(S,\mathcal{I})$, $V_e \subseteq T_\Sigma(V)_e$.
- For all $X \in BS$, $X \subseteq T_\Sigma(V)_X$.
- **tupling**: For all $n > 1$, $e_1,\dots,e_n \in \mathcal{T}_p(S,\mathcal{I})$ and $t_i \in T_\Sigma(V)_{e_i}$, $1 \le i \le n$, $(t_1,\dots,t_n) \in T_\Sigma(V)_{e_1\times\dots\times e_n}$.
- For all $f : e \to e' \in BF \cup F \cup INJ \cup PRJ \cup ITE$ and $t \in T_\Sigma(V)_e$, $f(t) \in T_\Sigma(V)_{e'}$.
- **λ-abstraction**: Let $e, e' \in \mathcal{T}_p(S,\mathcal{I})$ and $\{c_1 : e_1 \to e,\dots,c_n : e_n \to e\}$ be a constructor set. For all $x_i \in V_{e_i}$ and $t_i \in T_\Sigma(V)_{e'}$, $1 \le i \le n$,

$$\lambda c_1(x_1).t_1|\dots|\lambda c_n(x_n).t_n) \in T_\Sigma(V)_{e\to e'}.$$

- **term application**: For all $e, e' \in \mathcal{T}_p(S,\mathcal{I})$, $t \in T_\Sigma(V)_{e\to e'}$ and $u \in T_\Sigma(V)_e$, $t(u) \in T_\Sigma(V)_{e'}$.
- **collection**: For all $c \in \{bag, set\}$, $e \in \mathcal{T}_p(S,\mathcal{I})$ and $t \in T_\Sigma(V)^*_e$, $c(t) \in T_\Sigma(V)_{c(e)}$.

$\lambda(id(x).t)$ is also written as $\lambda(x.t)$.

The **extension** $g^*: T_\Sigma(V) \multimap\!\!\to A$ **of** g **to** $T_\Sigma(V)$ is defined inductively as follows:

- For all $x \in V$, $g^*(x) = g(x)$.
- For all $x \in \bigcup BS$, $g^*(x) = x$.
- For all $n > 1$ 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 $f : e \to e' \in F \cup BF$ and $t \in T_\Sigma(V)_e$, $g^*(f(t)) = f^A(g^*(t))$.
- Let $e, e' \in \mathcal{T}_p(S, \mathcal{I})$ and $\{c_1 : e_1 \to e, \dots, c_n : e_n \to e\}$ be a constructor set. For all $x_i \in V_{e_i}$, $t_i \in T_\Sigma(V)_{e'}$ and $a_i \in A_{e_i}$, $1 \leq i \leq n$,

  $$g^*(\lambda c_1(x_1).t_1 | \dots | \lambda c_n(x_n).t_n)(f_i^A(a_i)) = g[a_i/x_i]^*(t_i).$$

  Note that, if $e_i \in BS$, then $t\{a_i/x_i\}$ is a term and thus $g[a_i/x_i]^*(t_i) = g^*(t\{a/x\})$.
- For all $e, e' \in \mathcal{T}_p(S, \mathcal{I})$, $t \in T_\Sigma(V)_{e \to e'}$ and $u \in T_\Sigma(V)_e$, $g^*(t(u)) = g^*(t)(g^*(u))$.
- For all $c \in \mathit{Coll}$, $s \in S$ and $t_1, \dots, t_n \in T_\Sigma(V)$,

  $$g^*(c(t_1, \dots, t_n)) = [(g^*(t_1), \dots, g^*(t_n))]_{=_c}.$$

Let Σ be constructive and V be an $\mathcal{T}(S)$-sorted set of variables.

Then the set of Σ-terms over V that consist of symbols of $V \cup F \cup \{(,)\} \cup \bigcup BS$ forms a Σ-algebra is also denoted by $T_\Sigma(V)$ (see Term functors).

Moreover, for all $\mathcal{T}(S)$-sorted functions $g : V \to A$, the restriction of g^* to the algebra

$T_\Sigma(V)$ forms the unique Σ-homomorphism from $T_\Sigma(V)$ to A such that

$$g^* \circ inc_V = g.$$

The uniqueness implies

$$(h \circ g)^* = h \circ g^* \tag{1}$$

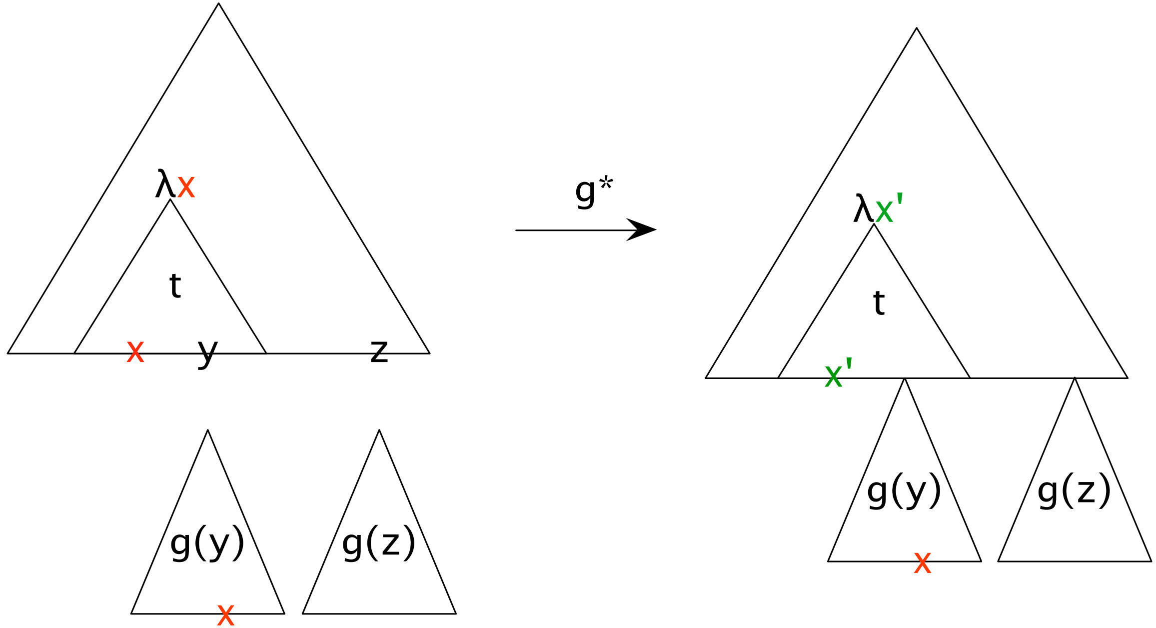
for all Σ-homomorphisms $h : A \to B$.

Substitution in λ-terms

Calculi for proving Σ-formulas often involve substitutions, and their correctness depends on the validity of (1) for $A = T_\Sigma(V)$—not only for the terms of the *algebra* $T_\Sigma(V)$, but also for, e.g., λ-abstractions. The above definition of $g^*(\lambda x.t)$, however, would not work.

Instead, for all $\mathcal{T}(S)$-sorted functions $g : V \to T_\Sigma(V)$, $g^*(\lambda x.t)$ must be redefined as follows in order both to prevent x from being substituted and to perform variable renaming if necessary:

$$g^*(\lambda x.t) \;=\; \begin{cases} \lambda x' g[x'/x]^*(t) & \text{if } x \in V_{t,g,x} =_{def} \bigcup var(g(free(t) \setminus \{x\})), \\ \lambda x.g[x/x]^*(t) & \text{otherwise.} \end{cases}$$



Lemma SUBST2

Let $\lambda T_\Sigma(V)$ be the set of Σ-terms over V that consist of symbols of $V \cup F \cup \{(,), \lambda\} \cup \bigcup BS$ and A be a Σ-algebra.

For all $\mathcal{T}(S)$-sorted functions $g : V \to \lambda T_\Sigma(V)$ and $h : V \to A$,

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

Proof. Proceeds similarly to [120], Lemma 8.3. ❑

Coterms with collections

Let $\Sigma = (BA, S, F, P)$ be a destructive signature, $BA = (BS, BF, BP)$, V be an $\mathcal{T}(S)$-sorted set whose elements are called "colors" and

$$Lab_\Sigma = \{(d,x,i,j) \mid d : s \to (\coprod_{i \in I}(e_{i1} \times \cdots \times e_{in_i}))^X \in F,\ x \in X,\ i \in I,\ 1 \leq j \leq n_i\} \cup \mathbb{N}.$$

$coT_\Sigma(V)$ denotes the greatest $\mathcal{T}(S)$-sorted set of prefix closed partial functions

$$t : Lab_\Sigma^* \multimap V \cup \bigcup BS \cup Coll$$

such that the following conditions hold true:

- For all $B \in BS$, $coT_\Sigma(V)_B = B \cup V_B$,
  here regarded as the set $1 \to B \cup V_B$ of "nullary" functions.
- For all $e \in S$ and $t \in coT_\Sigma(V)_e$, $t(\epsilon) \in V_s$,
  $$def(t) \cap Lab_\Sigma = \{(d,x,i,j) \in Lab_\Sigma \mid src(d) = s\}$$
  and for all $(d,x,i,j), (d,x,k,j') \in def(t) \cap Lab_\Sigma$,
  $i = k$ and $\lambda w.t((d,x,i,j)w) \in coT_\Sigma(V)_{e_{ij}}$.
- For all $c \in Coll$, $s \in S$ and $t \in coT_\Sigma(V)_{c(s)}$, $t(\epsilon) \in \{c\} \cup V_{c(s)}$ and there is $n \in \mathbb{N}$ such that $def(t) \cap Lab_\Sigma = [n]$ and for all $1 \leq i \leq n$, $\lambda w.t(iw) \in coT_\Sigma(V)_s$.

$Path_\Sigma$ is the least $\mathcal{T}(S)^2$-sorted subset of Lab_Σ^* such that

- for all $e \in \mathcal{T}(S)$, $\epsilon \in Path_{\Sigma,e,e}$,

- for all $d : s \to (\coprod_{i\in I}(e_{i1} \times \cdots \times e_{in_i}))^X \in F$, $x \in X$, $i \in I$, $1 \le j \le n_i$, $e \in \mathcal{T}(S)$ and $w \in Path_{\Sigma,e_j,e}$, $(d,x,i,j)w \in Path_{\Sigma,s,e}$,
- for all $e, e_1, \dots, e_n \in \mathcal{T}(S)$, $\bigwedge_{i=1}^n w_i \in Path_{\Sigma,e,e_i}$ implies $Path_{\Sigma,e,e_1\times\cdots\times e_n}$,
- for all $s \in S$, $c \in Coll$, $n \in \mathbb{N}$, $s \in S$, $e \in \mathcal{T}(S)$ and $w \in Path_{\Sigma,s,e}$, $nw \in Path_{\Sigma,c(s),e}$.

The above conditions imply that every $t \in coT_\Sigma(V)_e$ can be written as a sum of partial functions

$$\begin{array}{rcl} \coprod_{s\in S} t_s & : & Path_{\Sigma,e,s} \multimap V_s \\ +\coprod_{B\in BS} t_B & : & Path_{\Sigma,e,B} \multimap B \cup V_B \\ +\coprod_{c\in Coll,s\in S} t_{c,s} & : & Path_{\Sigma,e,c(s)} \multimap \{c\} \cup V_{c(s)}. \end{array}$$

For all $t \in coT_\Sigma(V)$, $def_1(t) =_{def} def(t) \cap Lab_\Sigma$.

Let $\sim$ be the greatest $\mathcal{T}(S)$-sorted equivalence relation on $coT_\Sigma(V)$ such that

- for all $B \in BS$, $\sim_B = \Delta^2_{B\cup V_B}$,
- for all $s \in S$ and $t \sim_s t'$, $t = t' \in V_s$ or for all $d \in def_1(t)$, $\lambda w.t(dw) \sim \lambda w.t'(dw)$,
- for all $s \in S$ and $t \sim_{bag(s)} t'$, $def_1(t) = def_1(t')$ and

$$\exists\ f : [n] \xrightarrow{\sim} [n] : \forall\ i \in def_1(t) : \lambda w.t(iw) \sim_s \lambda w.t'(f(i)w).$$

- for all $s\in S$ and $t\sim_{set(s)} t'$,

$$\forall\, i\in def_1(t)\ \exists\, j\in def_1(t') : \lambda w.t(iw)\sim_s \lambda w.t'(jw),$$
$$\forall\, j\in def_1(t')\ \exists\, i\in def_1(t) : \lambda w.t(iw)\sim_s \lambda w.t'(jw).$$

Consequently, for all $B\in BS$,

$$(coT_\Sigma(V)/{\sim})_B = coT_\Sigma(V)_B/{\sim}\ = (B\cup V_B)/{\sim}\ = B\cup V_B = F_B(coT_\Sigma(V)/{\sim}),$$

and for all $c\in Coll$ and $s\in S$,

$$(coT_\Sigma(V)/{\sim})_{c(s)} = coT_\Sigma(V)_{c(s)}/{\sim}\ \cong F_{c(s)}(coT_\Sigma(V)/{\sim})$$

(see Sorted sets, functions and relations).

Hence the S-sorted set $coT_\Sigma(V)/\sim$ as well as S-sorted functions from $coT_\Sigma(V)/\sim$ are lifted to an $\mathcal{T}(S)$-sorted set resp. $\mathcal{T}(S)$-sorted functions in the same way Σ-algebras resp. Σ-homomorphisms are lifted.

If Σ does not contain collection types, then $\sim=\Delta^2_{coT_\Sigma(V)}$ and thus $coT_\Sigma(V)/{\sim}=coT_\Sigma(V)$.

For ease of notation, we identify $coT_\Sigma(V)$ with $coT_\Sigma(V)/\sim$ and thus each element of $coT_\Sigma(V)$ with its equivalence class w.r.t. $\sim$.

The elements of $coT_\Sigma(V)$ are called **Σ-coterms over V**.

The elements of $coT_\Sigma = coT_\Sigma(\lambda e.1)$ are called **ground Σ-coterms**.

If for all $(d,x,i,j)\in Lab_\Sigma$, x, i or j depends on the other components of (d,x,i,j), then

x, i or j, respectively, is omitted.

$coT_\Sigma(V)$ is extended to a Σ-algebra as follows:

For all $d : s \to (\coprod_{i\in I}(e_{i1} \times \cdots \times e_{in_i}))^X \in F$, $t \in coT_\Sigma(V)_s$ and $x \in X$,

$$d^{coT_\Sigma(V)}(t)(x) = ((\lambda w.t((d,x,i,1)w), \ldots, \lambda w.t((d,x,i,n_i)w)), i)$$

where i is unique with $(d,x,i,1), \ldots, (d,x,i,n_i) \in def(t)$.

$coT_\Sigma(V)$ is called the **cofree Σ-algebra over** V.

Terms with product and sum extensions

Let $w \in \mathbb{N}^*$.

- For all $x \in X_s$,

$$x(w) =_{def} \begin{cases} x & \text{if } w = \epsilon, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

- For all $f : s_1 \dots s_n \to s \in F$ and $t_i \in T_\Sigma(X)_{s_i}$, $1 \leq i \leq n$,

$$f\langle t_1, \dots, t_n\rangle(w) =_{def} \begin{cases} f & \text{if } w = \epsilon, \\ t_{i+1}(v) & \text{if } w = iv \text{ for some } i \in \mathbb{N},\ v \in \mathbb{N}^*, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

- For all $f : s \to s_1 \dots s_n \in F$ and $t_i \in coT_\Sigma(X)_{s_i}$, $1 \leq i \leq n$,

$$[t_1, \dots, t_n]f(w) =_{def} \begin{cases} f & \text{if } w = \epsilon, \\ t_{i+1}(v) & \text{if } w = iv \text{ for some } i \in \mathbb{N},\ v \in \mathbb{N}^*, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

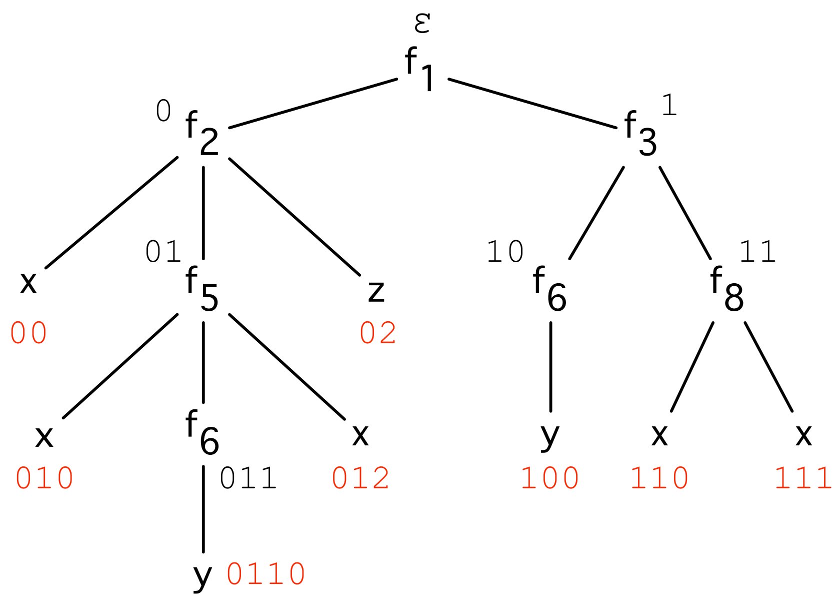
Given a coterm t and $w \in \mathbb{N}^*$, $path(t, w)$ returns the sequence of symbols on the path from the root to node w of t: For all $x \in X$, $[t_1, \dots, t_n]f \in coT_\Sigma(X)$, $i \in \mathbb{N}$ and $w \in \mathbb{N}^*$,

$$path(x, w) =_{def} \begin{cases} x & \text{if } w = \epsilon, \\ \text{undefined} & \text{otherwise,} \end{cases}$$

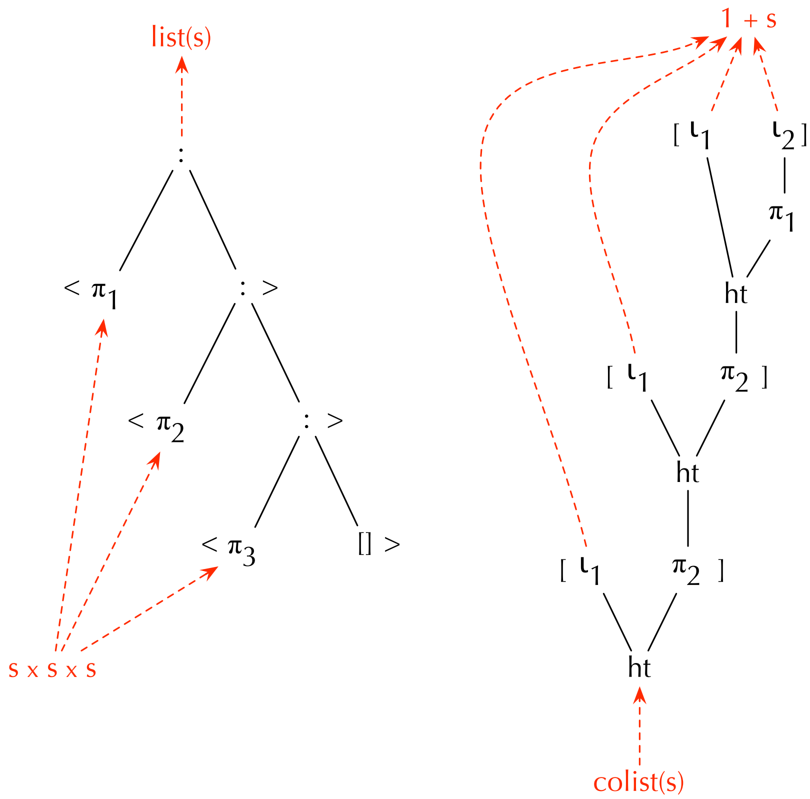
$$path([t_1, \dots, t_n]f, iw) =_{def} \begin{cases} f\ path(t_{i+1}, w) & \text{if } 0 \le i < n, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

A (co)term t over $\mathbb{N}^*$ such that all operations of t belong to $F \setminus BF$ and for all $x \in var(t) \cup cov(t)$, $sort(x) \in BS$ and $t(x) = x$, is called a **Σ-generator** resp. **Σ-observer**.

Given $w \in \mathbb{N}^*$ and a (co)term t, $w \cdot t$ denotes the (co)term obtained from t by replacing each (co)variable v of t with wv.

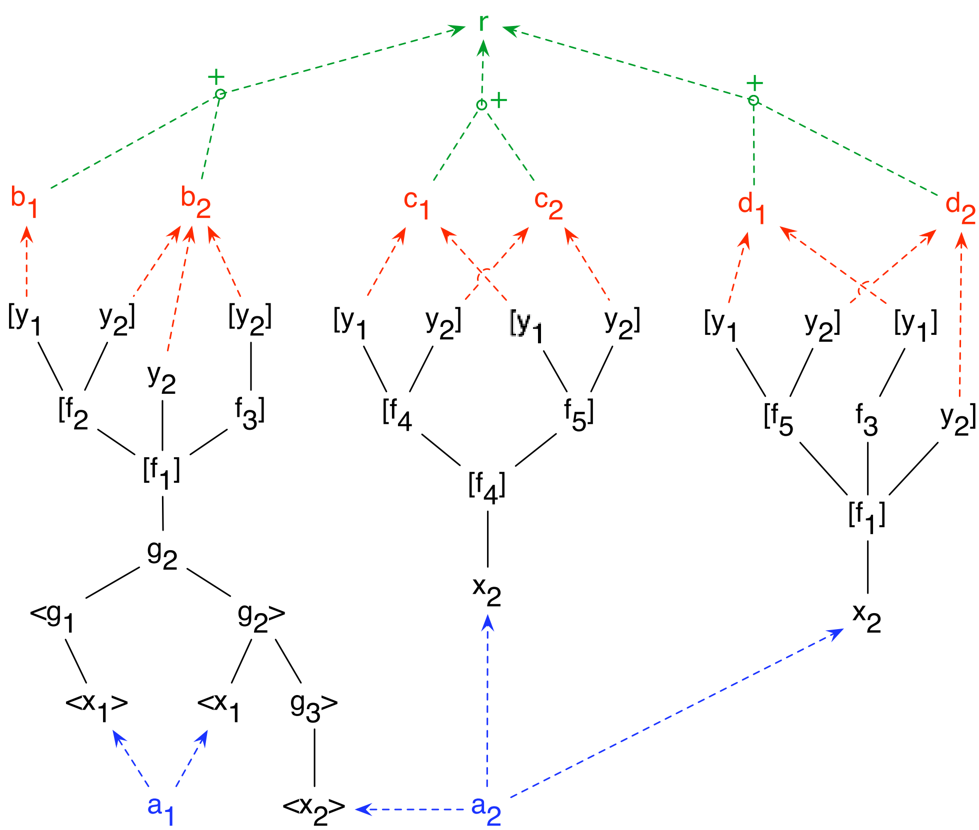


The tree representing the term $f_1\langle f_2\langle x, f_5\langle x, f_6\langle y\rangle, x\rangle, z\rangle, f_3\langle f_6\langle y\rangle, f_8\langle x, x\rangle\rangle\rangle$
or the coterm $[[[x, [x, [y]f_6, x]f_5, z]f_2, [[y]f_6, [x, x]f_8]f_3]f_1$



The term $: \langle x : \langle y : \langle x, []\rangle\rangle\rangle$ generates lists of length 3 from two elements.

If applied to a list with at least three elements, the coterm $[x, [[x, [[x, [y]\pi_1]ht]\pi_2]ht]\pi_2]ht$ returns the third element at exit y. If the list has fewer elements, the coterm returns this fact by taking exit x. The underlying signatures are given later.



The data flow induced by the formula $r(t_1, t_2, t_3)$ where
$t_1 = [[[y_1, y_2]f_2, y_2, [y_2]f_3]f_1]g_2\langle g_1\langle x_1\rangle, g_2\langle x_1, g_3\langle x_2\rangle\rangle\rangle$,
$t_2 = [[[y_1, y_2]f_4, [y_1, y_2]f_5]f_4]x_2$ and $t_3 = [[[y_1, y_2]f_5, [y_1]f_3, y_2]f_1]x_2$.

$$r(t_1, t_2, t_3)^A = \{h \in A^X \mid (t_1^A(h), t_2^A(h), t_3^A(h)) \in r^A\}$$

For all $s \in S$,

$$Beh_{0,s} =_{def} \prod_{t \in Obs_{\Sigma,s}} (BT \times cov(t)).$$

Intuitively, an element of $Beh_{0,s}$ is a tuple of possible results of applying s-observers to any s-element of a Σ-algebra. The result of applying observer t is a pair (a, x) that consists of an "output" value $a \in BA$ and a covariable x of t representing the "exit" where a is returned.

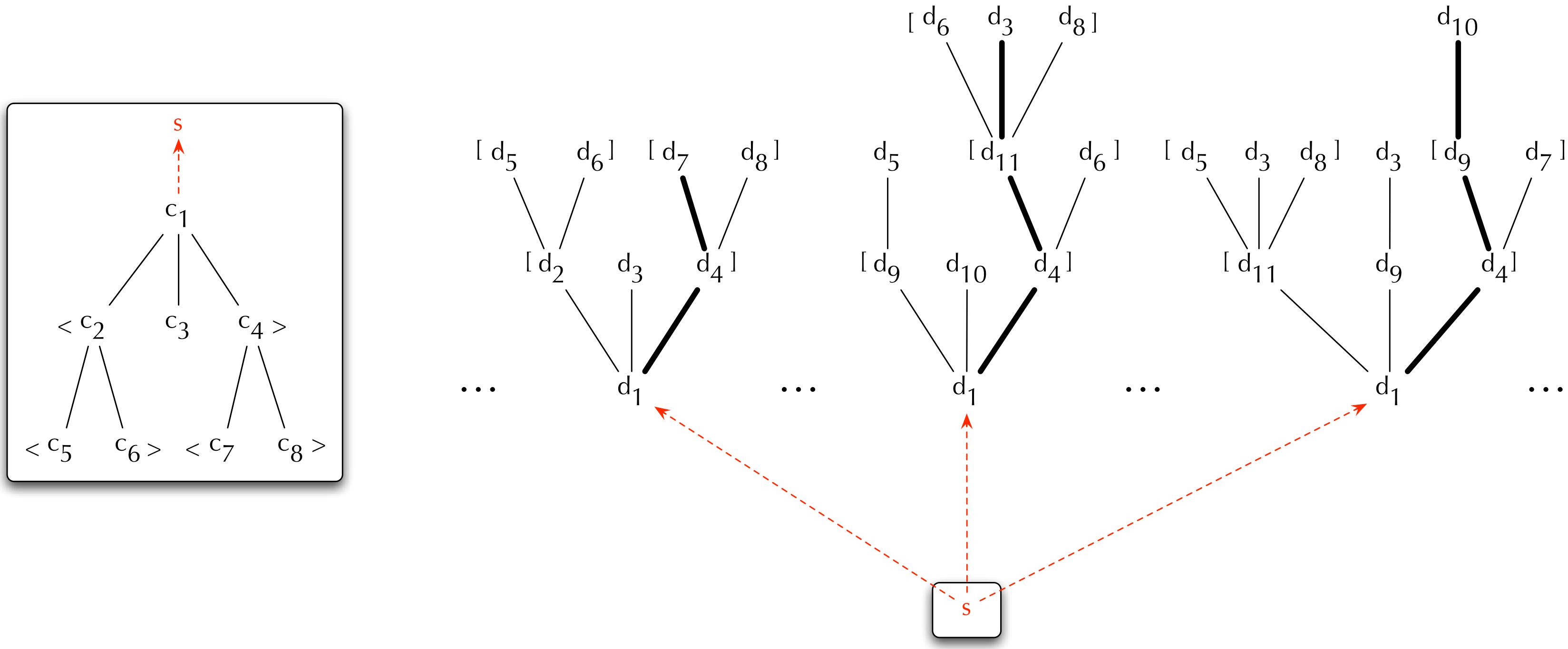
$b \in Beh_{0,s}$ is called a **Σ-behavior** if for all $t, u \in Obs_{\Sigma,s}$, $n \in \mathbb{N}$ and $w \in \mathbb{N}^n$,

$$path(t,w) = path(u,w) \text{ implies } take(n+1)(\pi_2(b_t)) = take(n+1)(\pi_2(b_u)). \quad (1)$$

By (1), the "runs" of two observers t and u on b "take the same direction" as long as both observers apply the same destructors. In particular, if they start with the same destructor f, they take the same exit of f, formally: for all $b \in Beh_\Sigma(BA)_s$ and $t, u \in Obs_{\Sigma,s}$, $t(\epsilon) = u(\epsilon)$ implies $head(\pi_2(b_t)) = head(\pi_2(b_u))$. Hence

for all $f : s \to s_1 \dots s_n \in F$ and $b \in Beh_{\Sigma,s}$ there is $1 \leq i_{f,b} \leq n$ such that
for all $t \in Obs_{\Sigma,s}$, $t(\epsilon) = f$ implies $head(\pi_2(b_t)) = i_{f,b}$. (2)

An element of $\mu\Sigma \cong T_\Sigma$ (left) resp. $\nu\Sigma_{BA} \cong Beh_\Sigma$ (right):



- For all $s \in S$, $\nu\Sigma_s = Beh_{\Sigma,s}$.
- For all $f : s \to s_1 \dots s_n \in F \setminus BF$ and $(b_t)_{t \in Obs_{\Sigma,s}} \in Beh_{\Sigma,s}$,

$$f^{\nu\Sigma}(b) = (((\langle \pi_1, tail \circ \pi_2 \rangle (b_{[t_1,\dots,t_n]f}))_{t_i \in Obs_{\Sigma,s_i}}, i)$$

where $i = i_{f,b}$ and for all $k \neq i$, $t_k \in Obs_{\Sigma,s_k}$. Note that $head(\pi_1(b_{[t_1,\dots,t_n]f})) = i$.

For all Σ-algebras A, the unique Σ-morphism $unfold^A : A \to \nu\Sigma$ is defined as follows:

For all $s \in S$ and $a \in A_s$,

$$unfold_s^A(a) \;=\; (t^A(a))_{t \in Obs_{\Sigma,s}}.$$

Recursive equations

Let $C\Sigma = (S, \mathcal{I}, C)$ be a constructive signature, $D\Sigma = (S, \mathcal{I}, D)$ be a destructive signature, $\Sigma = C\Sigma \cup D\Sigma$ and $\Psi = (C\Sigma, D\Sigma)$. A set

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

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

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

Lemma AUX Sei $s \in S$. Sei $\{c_1 : e_1 \to s, \dots, c_n : e_n \to s\} = \{c : e \to s' \in C \mid s' = s\}$. Die Summenextension $[c_1^A, \dots, c_n^A]$ ist bijektiv. Es gibt also eine Funktion

$$d_s^A : A_s \to A_{e_1} + \dots + A_{e_n}$$

mit $[c_1^A, \dots, c_n^A] \circ d_s^A = id_{A_s}$ und $d_s^A \circ [c_1^A, \dots, c_n^A] = id_{A_{e_1}+\dots+A_{e_n}}$. ❏

Theorem INDSOL If C_2 is empty, then E has a unique inductive solution in the initial $C\Sigma$-algebra.

Proof . Let $\mathcal{A} = (A, Op)$ be initial in Alg_Σ. By Lemma INI (1) and (3), $\mathcal{A}$ satisfies the induction principle. Suppose that the $\mathcal{T}_p(S, \mathcal{I})$-sorted set B with

$$B_e = \{a \in A_e \mid \forall\ f : e \to e' \in V : \pi_f(lfp(E))(a) \neq \bot\}$$

is a Σ-invariant of $\mathcal{A}$. (1)

Then a solution g of E in $\mathcal{A}$ is defined as follows: For all $f : e \to e' \in V$ and $a \in A_e$, $g(f)(a) = \pi_f(lfp(E))(a)$.

Zunächst wird die Existenz einer induktiven Lösung von E in A durch Induktion bewiesen. Wir zeigen, dass B mit

$$B_s = \{a \in A_s \mid \text{für alle } d : s \to e \in D \text{ ist } d^A(a) \text{ definiert}\}, \qquad s \in S,$$

eine Unteralgebra von A ist.

Let $a \in A_s$. By Lemma AUX there are $1 \leq i \leq n$ and $b \in A_{e_i}$ with $d_s^A(a) = (b, i)$ and $c_i^A(b) = a$.

Sei $d : s \to e' \in D$ und $b = (a_1, \ldots, a_{n_{c_i}}) \in B_{e_i}$, d.h. für alle $1 \leq j \leq n_{c_i}$ ist $d^A(a_j)$ definiert. Dann ist auch $a' = val[a_1/x_1, \ldots, a_{n_{c_i}}/x_{n_{c_i}}]^*(t_{d,c_i})$ definiert, wobei val eine beliebige Variablenbelegung ist.

Um d^A an der Stelle a durch a' zu definieren, bleibt zu zeigen, dass die Darstellung von a als Applikation $c_i^A(b)$ eines Konstruktors eindeutig ist.

Sei $1 \leq j \leq n$ und $b' \in A_{e_j}$ mit $a = c_j^A(b')$. Dann ist

$$(b, i) = d_s^A(a) = d_s^A(c_j^A(b')) = d_s^A([c_1^A, \ldots, c_n^A](b', j)) \stackrel{(4)}{=} (id_{A_{e_1} + \cdots + A_{e_n}})(b', j) = (b', j),$$

also $b = b'$ und $c_i = c_j$.

Folglich liefert $d^A(c_i^A(b)) = a'$ eine eindeutige Definition von d^A an der Stelle a. Damit gilt (2) und wir schließen aus (1), dass $d^A(a)$ für alle $a \in A_s$ (eindeutig) definiert ist.

Auch die Eindeutigkeit der induktiven Lösung von E in A lässt sich durch Induktion zeigen: Seien A_1, A_2 zwei Lösungen von E in A.

Wir zeigen, dass B mit

$$B_s \ = \ \{a \in A_s \mid \text{für alle } d : s \to e \in D,\ d^{A_1}(a) = d^{A_2}(a)\}$$

eine Unteralgebra von A ist.

Sei $c : e \to s \in C$, $d : s \to e' \in D$ und $a = (a_1, \dots, a_{n_c}) \in B_e$, d.h. für alle $1 \leq i \leq n_c$ ist $d^{A_1}(a_i) = d^{A_2}(a_i)$. Daraus folgt für $val_1 : V \to A_1$ und $val_2 : V \to A_2$ mit $val_1(x_i) = val_2(x_i) = a_i$:

$$d^{A_1}(c^A(a)) \ \stackrel{A_1 \text{ löst } E}{=} \ val_1^*(t_{d,c}) = val_2^*(t_{d,c}) \ \stackrel{A_2 \text{ löst } E}{=} \ d^{A_2}(c^A(a)).$$

Damit gilt (2) und wir schließen aus (1), dass d^{A_1} mit d^{A_2} übereinstimmt. ❏

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.

Let $\mathcal{A} = (A, Op)$ be a $D\Sigma$-algebra. A set

$$E \ = \ \{c = obj\{d.t_{c,d} \mid d \in D,\ src(d) = trg(c)\} \mid c \in C\}$$

of Σ-equations over C **defines C coinductively on $\mathcal{A}$** if E has a unique solution in $\mathcal{A}$.

Theorem COINDSOL (old version of Theorem RECFUN2/3)

Sei E ein rekursives Ψ-Gleichungssystem. E has a unique coinductive solution in the final $D\Sigma$-algebra A. Moreover, the initial $C\Sigma$-algebra $T_{C\Sigma}$ is a $D\Sigma$-algebra that satisfies E and $unfold^{T_{C\Sigma}} = fold^{\mathcal{A}}$. Die unten zur Σ-algebra T_E erweiterte Termalgebra $T_{C\Sigma}$ erfüllt E und es gilt

$$unfold^{T_E} = fold^{\mathcal{A}}.$$

Beweis. Sei V eine S-sortige Variablenmenge, die die Trägermengen von A enthält. Wir erweitern die Menge $T_{C\Sigma}(A)$ der $C\Sigma$-Terme über A wie folgt zur Σ-algebra $T_E(A)$: Für alle Operationen $f : e \to e'$ von Σ und $t \in T_{C\Sigma}(A)_e$,

$$f^{T_E(A)}(t) \;=\; eval(ft),$$

wobei $eval : T_\Sigma(V) \to T_{C\Sigma}(V)$ wie unten definiert ist. Die für eine induktive Definition erforderliche wohlfundierte Ordnung $\gg$ auf den Argumenttermen von *eval* lautet wie folgt: Für alle $t, t' \in T_\Sigma(V)$,

$$t \gg t' \;\Leftrightarrow_{def}\; (dep_{C_2}(t), dep_D(t), size(t)) >_{lex} (dep_{C_2}(t'), dep_D(t'), size(t')).$$

$>_{lex} \subseteq \mathbb{N}^3 \times \mathbb{N}^3$ bezeichnet die lexikographische Erweiterung von $> \subseteq \mathbb{N} \times \mathbb{N}$ auf Zahlentripel.

Sei $G \subseteq F$. $size(t)$ bezeichnet die Anzahl der Symbole von t, $dep_G(t)$ die maximale Anzahl von G-Symbolen auf einem Pfad von t. .

Die induktive Definition von *eval* lautet wie folgt:

- Für alle $x \in V$, $eval(x) = x$.
- Für alle $f : X \to Y \in BF \cup BF'$ und $x \in X$, $eval(fx) = f(x)$.
- Für alle $f : s \to e \in D$ und $a \in A_s$, $eval(f(a)) = f^{\mathcal{A}}(a)$. (1)
- Für alle $x \in V$ und $t \in T_\Sigma(V)$, $eval(\lambda x.t) = \lambda x.eval(t)$.
- Für alle $c : e_1 \times \dots \times e_n \to s \in C$ und $t_i \in T_\Sigma(V)_{e_i}$, $1 \leq i \leq n$,
  $eval(c(t_1, \dots, t_n)) = c(eval(t_1), \dots, eval(t_n))$. (2)
- Für alle $t, u \in T_\Sigma(V)$, $eval(t(u)) = eval(t)(eval(u))$.
- Für alle $t, u, v \in T_\Sigma(V)$, $eval(ite(t, u, v)) = ite(eval(t), eval(u), eval(v))$.
- Für alle $d : s \to e' \in D$, $c : e \to s \in C$ und $(t_1, \dots, t_n) \in T_\Sigma(V)_e$,
  $eval(dc(t_1, \dots, t_n)) = u\{t_1/x_1, \dots, t_n/x_n, eval(u_1\sigma)/z_1, \dots, eval(u_k\sigma)/z_k\}$, (3)
  wobei $u \in T_{C\Sigma}(V)$, $\{z_1, \dots, z_k\} = var(u) \setminus \{x_1, \dots, x_n\}$, $u_1, \dots, u_k \in T_\Sigma(V)$ aus Destruktoren und Variablen bestehen, $\sigma = \{t_1/x_1, \dots, t_n/x_n\}$ und

  $$t_{d,c} = u\{u_1/z_1, \dots, u_k/z_k\}.$$

- Für alle $d : s \to e \in D$, $d' : s' \to s \in D$ und $u \in T_\Sigma(V)_{s'}$,
  $eval(dd'u) = eval(d\ eval(d'u))$. (4)

(5) Für alle $t \in T_\Sigma(V)$ ist $eval(t)$ definiert und $dep_{C_2}(eval(t)) \leq dep_{C_2}(t)$.

Beweis von (5) durch Induktion über t entlang $\gg$.

Fall (1): Es gibt $f : e \to e' \in BF \cup BF' \cup D$ und $a \in A_e$ mit $t = fa$. Dann ist $eval(t) = f(a)$ bzw. $eval(t) = f^{\mathcal{A}}(a)$ und $dep_{C_2}(eval(t)) = 0 = dep_{C_2}(t)$.

Fall (2): Es gibt $c : e \to s \in C \cup \{\lambda x.\_ \mid x \in V\} \cup \{\_(\_), ite\}$ und $(t_1, \dots, t_n) \in T_\Sigma(v)_e$ mit $t = c(t_1, \dots, t_n)$. Sei $1 \leq i \leq n$.

Ist $c \in C_2$, dann gilt $dep_{C_2}(t_i) < dep_{C_2}(t)$. Ist $c \notin C_2$, dann gilt $dep_G(eval(t_i)) \leq dep_G(t)$ für $G \in \{C_2, D\}$, aber $size(t_i) < size(t)$. Demnach gilt $t \gg t_i$ in beiden Unterfällen. Also ist nach Induktionsvoraussetzung $eval(t_i)$ definiert und $dep_{C_2}(eval(t_i)) \leq dep_{C_2}(t_i)$.

Daraus folgt, dass auch $eval(t) = c(eval(t_1), \dots, eval(t_n))$ definiert ist und

$$dep_{C_2}(eval(t)) = max\{dep_{C_2}(eval(t_i)) \mid 1 \leq i \leq n\} \leq max\{dep_{C_2}(t_i) \mid 1 \leq i \leq n\}$$
$$= dep_{C_2}(t)$$

im Fall $c \in C_1$ bzw.

$$dep_{C_2}(eval(t)) = 1 + max\{dep_{C_2}(eval(t_i)) \mid 1 \leq i \leq n\}$$
$$\leq 1 + max\{dep_{C_2}(t_i) \mid 1 \leq i \leq n\} = dep_{C_2}(t)$$

im Fall $c \in C_2$.

Fall (3): Es gibt $d : s \to e' \in D$, $c : e \to s \in C$ und $(t_1, \dots, t_n) \in T_\Sigma(V)_e$ mit $t = dc(t_1, \dots, t_n)$. Seien $k, u, \sigma, z_1, \dots, z_k, u_1, \dots, u_k$ wie oben und $1 \leq i \leq k$. Ist $c \in C_1$, dann ist $u_i\sigma$ ein echter Teilterm von t und damit $dep_G(u_i\sigma) \leq dep_G(t)$ für $G \in \{C_2, D\}$, aber $size(u_i\sigma) < size(t)$. Ist $c \in C_2$, dann gilt $dep_{C_2}(u_i\sigma) < dep_{C_2}(t)$.

Folglich gilt $t \gg u_i\sigma$ in beiden Unterfällen. Also ist nach Induktionsvoraussetzung $eval(u_i\sigma)$ definiert und $dep_{C_2}(eval(u_i\sigma)) \leq dep_{C_2}(u_i\sigma)$.

Demnach ist auch

$$eval(t) = u\{t_1/x_1, \ldots, t_n/x_n, eval(u_1\sigma)/z_1, \ldots, eval(u_k\sigma)/z_k\}$$

definiert und $dep_{C_2}(eval(t)) \leq dep_{C_2}(t)$, weil jeder Pfad von u im Fall $c \in C_1$ kein C_2-Symbol und im Fall $c \in C_2$ höchstens eins enthält.

Fall (4): Es gibt $d : s \to e \in D$, $d' : s' \to s \in D$ und $u \in T_\Sigma(V)_{s'}$ mit $t = dd'u$. Dann gilt $dep_{C_2}(d'u) \leq dep_{C_2}(t)$, aber $dep_D(d'u) \leq dep_D(t)$, also $t \gg d'u$. Damit ist nach Induktionsvoraussetzung $eval(d'u)$ definiert und $dep_{C_2}(eval(d'u)) \leq dep_{C_2}(d'u)$, also auch $dep_{C_2}(d\ eval(d'u)) = dep_{C_2}(eval(d'u)) \leq dep_{C_2}(t)$. Wegen $eval(d'u) \in T_{C\Sigma}(V)$ ist jedoch $dep_D(d\ eval(d'u)) < dep_D(t)$, so dass nach Induktionsvoraussetzung auch $eval(t) = eval(d\ eval(d'u))$ definiert ist und $dep_{C_2}(eval(d\ eval(d'u))) \leq dep_{C_2}(d\ eval(d'u))$. Daraus folgt schließlich $dep_{C_2}(eval(t)) \leq dep_{C_2}(d\ eval(d'u)) \leq dep_{C_2}(t)$. ❑

Wie man ebenfalls durch Induktion über t entlang $\gg$ zeigen kann, kommen alle Variablen von $eval(t)$ in t vor. Daraus folgt $f^{T_E(A)}(t) = eval(ft) \in T_{C\Sigma}(A)$ und $f^{T_E(A)}(u) = eval(fu) \in T_{C\Sigma}$ für alle Operationen $f : e \to e'$ von Σ, $t \in T_{C\Sigma}(A)_e$ und $u \in T_{C\Sigma,e}$. Folglich ist die Einschränkung T_E von $T_E(A)$ auf die Menge $T_{C\Sigma}$ der $C\Sigma$-Grundterme eine $D\Sigma$-Unteralgebra von $T_E(A)$.

(6) Für alle $c : e \to s \in C$ und $(t_1,\dots,t_n) \in T_{C\Sigma}(A)_e$, $c^{T_E(A)}(t_1,\dots,t_n) = c(t_1,\dots,t_n)$.

Beweis von (6). $eval(t) = t$ für alle $t \in T_{C\Sigma}(A)$ erhält man durch Induktion über die Größe von t. Daraus folgt

$$c^{T_E(A)}(t_1,\dots,t_n) \stackrel{Def.\ c^{T_E(A)}}{=} eval(c(t_1,\dots,t_n)) \stackrel{(2)}{=} c(eval(t_1),\dots,eval(t_n))$$
$$\stackrel{eval(t_i)=t_i}{=} c(t_1,\dots,t_n).$$

(7) Für alle $g : V \to T_{C\Sigma}(A)$ und Σ-Terme t, die aus Destruktoren und einer Variable x bestehen, gilt $eval(t\sigma) = g^*(t)$, wobei $\sigma = \{g(x)/x\}$.

Beweis durch Induktion über die Anzahl der Destruktoren von t.

Sei $d_1,\dots,d_n \in D$ und $t = d_1 \dots d_n x$. Ist $n = 0$, dann gilt

$$eval(t\sigma) = eval(x\sigma) = eval(g(x)) \stackrel{g(x)\in T_{C\Sigma}(A)}{=} g(x) = g^*(x) = g^*(t).$$

Andernfalls ist

$$eval(t\sigma) = eval(d_1 \dots d_n x\sigma) \stackrel{(4)}{=} eval(d_1\ eval(d_2 \dots d_n x\sigma))$$
$$\stackrel{ind.\ hyp.}{=} eval(d_1\ g^*(d_2 \dots d_n x)) \stackrel{Def.\ d_1^{T_E(A)}}{=} d_1^{T_E(A)}(g^*(d_2 \dots d_n x))$$
$$\stackrel{Def.\ g^*}{=} g^*(d_1 \dots d_n x) = g^*(t).$$ ❑

(8) $T_E(A)$ erfüllt E.

Beweis.

Für alle $c : s_1 \times \cdots \times s_n \to s \in C$, $d : s \to e \in D$ und $\sigma = g : V \to T_{C\Sigma}(A)$,

$$\begin{aligned} g^*(dc(x_1,\dots,x_n)) &\overset{\text{Def. } g^*}{=} d^{T_E(A)}(c^{T_E(A)}(g(x_1),\dots,g(x_n))) \\ &\overset{\text{Def. } d^{T_E(A)}}{=} eval(dc^{T_E(A)}(g(x_1),\dots,g(x_n))) \overset{(6)}{=} eval(dc(g(x_1),\dots,g(x_n))) \\ &\overset{(3)}{=} u\{x_1\sigma/x_1,\dots,x_n\sigma/x_n, eval(u_1\sigma)/z_1,\dots,eval(u_k\sigma)/z_k\} \\ &\overset{(7)}{=} u\{x_1\sigma/x_1,\dots,x_n\sigma/x_n, g^*(u_1)/z_1,\dots,g^*(u_k)/z_k\} \overset{u\in T_{C\Sigma}(V)}{=} g^*(t_{d,c}). \quad ❏ \end{aligned}$$

Da $T_E(A)$ eine $D\Sigma$-Algebra und A die finale $D\Sigma$-Algebra ist, gibt es den eindeutigen $D\Sigma$-Homomorphismus $unfold^{T_E(A)} : T_E(A) \to A$.

(9) Für alle $a \in A$, $unfold^{T_E(A)}(a) = a$.

Beweis.

Für alle $d : s \to e \in D$ gilt $d^{\mathcal{A}}(a) = d^{T_E(A)}(a)$. Folglich sind die Inklusion $inc_A : A \to T_{C\Sigma}(A)$ und daher auch die Komposition

$$unfold^{T_E(A)} \circ inc_A : A \to A$$

$D\Sigma$-homomorph. Also stimmt diese wegen der Finalität von A mit der Identität auf A überein. ❏

A lässt sich zur $C\Sigma$-Algebra erweitern: Für alle $c : e \to s \in C$ und $a \in A_e$,

$$c^{\mathcal{A}}(a) =_{def} unfold^{T_E(A)}(c(a)). \qquad (10)$$

Für alle $c : s_1 \times \cdots \times s_n \to s \in C$, $d : s \to e \in D$ und $g : V \to A$,

$$\begin{aligned}
& g^*(dc(x_1,\dots,x_n)) = d^{\mathcal{A}}(c^{\mathcal{A}}(g(x_1),\dots,g(x_n))) \\
& \stackrel{(10)}{=} d^{\mathcal{A}}(unfold^{T_E(A)}(c(g(x_1),\dots,g(x_n)))) \\
& \stackrel{unfold^{T_E(A)} D\Sigma-homomorph}{=} unfold^{T_E(A)}(d^{T_E(A)}(c(g(x_1),\dots,g(x_n)))) \\
& \stackrel{(6)}{=} unfold^{T_E(A)}(d^{T_E(A)}(c^{T_E(A)}(g(x_1),\dots,g(x_n)))) =???unfold^{T_E(A)}(g^*(dc(x_1,\dots,x_n))) \\
& \stackrel{(8)}{=} unfold^{T_E(A)}(g^*(t_{d,c})) \stackrel{(9)}{=} g^*(t_{d,c}).
\end{aligned}$$

Also gibt es eine coinduktive Lösung von E in A.

(11) Die größte $D\Sigma$-Kongruenz R ist eine $C\Sigma$-Kongruenz.

Beweis. Sei R^C der C-Abschluss von R (s.o.). Ist R^C eine $D\Sigma$-Kongruenz, dann ist R^C in R enthalten, weil R die größte $D\Sigma$-Kongruenz ist. Andererseits ist R in R^C enthalten. Also stimmt R mit R^C überein, ist also wie R^C eine $C\Sigma$-Kongruenz. Demnach bleibt zu zeigen, dass R^C eine $D\Sigma$-Kongruenz ist.

Sei also $d : s \to e \in D$ und $(t,u) \in R^C_s$.

Gehört (t,u) zu R, dann gilt das auch für $(d^{T_E(A)}(t), d^{T_E(A)}(u))$, weil R eine $D\Sigma$-Kongruenz ist. Wegen $R \subseteq R^C$ folgt $(d^{T_E(A)}(t), d^{T_E(A)}(u)) \in R^C_e$.

Andernfalls gibt es $c : s_1 \times \cdots \times s_n \to s \in C$ und $t_1, \dots, t_n, u_1, \dots, u_n \in T_{C\Sigma}(A)$ mit $t = c(t_1, \dots, t_n)$, $u = c(u_1, \dots, u_n)$ und $(t_i, u_i) \in R^C$ für alle $1 \le i \le n$.

Nach Induktionsvoraussetzung gilt $(d'^{T_E(A)}(t_i), d'^{T_E(A)}(u_i)) \in R^C_{e'}$ für alle $1 \le i \le n$ und $d' : s_i \to e' \in D$. Seien g, g' Belegungen von V in $T_{C\Sigma}(A)$ mit $g(x_i) = t_i$ und $g'(x_i) = u_i$ für alle $1 \le i \le n$.

Wegen

$$d^{T_E(A)}(t) = d^{T_E(A)}(c(t_1, \dots, t_n)) \overset{(6)}{=} d^{T_E(A)}(c^{T_E(A)}(t_1, \dots, t_n)) \overset{(8)}{=} g^*(t_{d,c}),$$
$$d^{T_E(A)}(u) = d^{T_E(A)}(c(u_1, \dots, u_n)) \overset{(6)}{=} d^{T_E(A)}(c^{T_E(A)}(u_1, \dots, u_n)) \overset{(8)}{=} g'^*(t_{d,c})$$

und weil R^C eine $C\Sigma$-Kongruenz auf $T_{C\Sigma}(A)$ ist, folgt $(d^{T_E(A)}(t), d^{T_E(A)}(u)) \in R^C_e$ aus $(g(x_i), g'(x_i)) \in R^C$ für alle $1 \le i \le n$. Also ist R^C eine $D\Sigma$-Kongruenz. ❏

(11) liefert folgende $C\Sigma$-Algebra B mit den Trägermengen von A:

Für alle $c : e \to s \in C$ und $t \in T_{C\Sigma}(A)_e$,

$$c^B(unfold^{T_E(A)}(t)) =_{def} unfold^{T_E(A)}(c(t)). \tag{12}$$

c^B ist wohldefiniert: Sei $t, u \in T_{C\Sigma}(A)_e$ mit $unfold^{T_E(A)}(t) = unfold^{T_E(A)}(u)$. Da A final ist, stimmt R nach Satz 3.4 (3) mit dem Kern von $unfold^{T_E(A)}$ überein.

Also impliziert (11), dass der Kern von $unfold^{T_E(A)}$ eine $C\Sigma$-Kongruenz auf $T_{C\Sigma}(A)$ ist. Daraus folgt

$$c^B(unfold^{T_E(A)}(t)) \stackrel{(12)}{=} unfold^{T_E(A)}(c(t)) \stackrel{(6)}{=} unfold^{T_E(A)}(c^{T_E(A)}(t))$$
$$= unfold^{T_E(A)}(c^{T_E(A)}(u)) \stackrel{(6)}{=} unfold^{T_E(A)}(c(u)) \stackrel{(12)}{=} c^B(unfold^{T_E(A)}(u)).$$

(13) c^B stimmt mit $c^{\mathcal{A}}$ überein: Für alle $a \in A$,

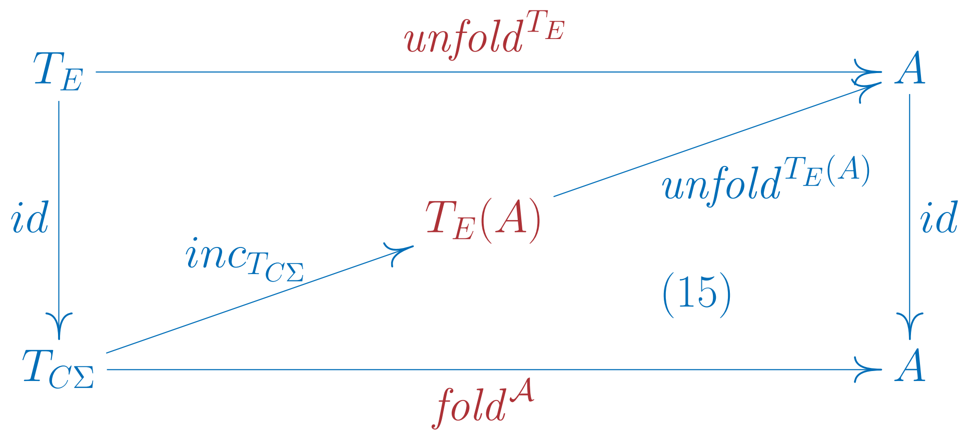
$$c^B(a) \stackrel{(9)}{=} c^B(unfold^{T_E(A)}(a)) \stackrel{(12)}{=} unfold^{T_E(A)}(c(a)) \stackrel{(10)}{=} c^{\mathcal{A}}(a).$$

(14) $unfold^{T_E(A)}$ ist $C\Sigma$-homomorph: Für alle $c : e \to s \in C$ und $t \in T_{C\Sigma}(A)_e$,

$$unfold^{T_E(A)}(c^{T_E(A)}(t)) \stackrel{(6)}{=} unfold^{T_E(A)}(c(t)) \stackrel{(12)}{=} c^B(unfold^{T_E(A)}(t)) \stackrel{(13)}{=} c^{\mathcal{A}}(unfold^{T_E(A)}(t)).$$

Sei T_E die $D\Sigma$-Unteralgebra von $T_E(A)$ mit Trägermenge $T_{C\Sigma}$.

(15) $unfold^{T_E} = fold^{\mathcal{A}}$: Da A eine finale $D\Sigma$-Algebra ist, existiert genau ein $D\Sigma$-Homomorphismus von T_E nach A. Folglich stimmt $unfold^{T_E}$ mit der Einschränkung von $unfold^{T_E(A)}$ auf $T_{C\Sigma}$ überein. Da $inc_{T_{C\Sigma}} : T_{C\Sigma} \to T_E(A)$ wegen (6) und $unfold^{T_E(A)}$ wegen (14) $C\Sigma$-homomorph ist, folgt (15) aus der Initialität von $T_{C\Sigma}$.



Es bleibt zu zeigen, dass je zwei coinduktive Lösungen A_1, A_2 von E in A miteinander übereinstimmen.

Sei Q die kleinste S-sortige Relation auf $A_1 \times A_2$, die die Diagonale von A enthält und für alle $c : e \to s \in C$ und $a, b \in A_e$ die folgende Implikation erfüllt:

$$(a, b) \in Q \quad \Rightarrow \quad (c^{A_1}(a), c^{A_2}(b)) \in Q. \tag{16}$$

(17) Q ist eine $D\Sigma$-Kongruenz.

Beweis. Sei $d : s \to e \in D$ und $(a, b) \in Q_s$. Gehört (a, b) zu Δ_A^2, dann gilt $a = b$, also $d^{\mathcal{A}}(a) = d^{\mathcal{A}}(b)$. Daraus folgt $(d^{\mathcal{A}}(a), d^{\mathcal{A}}(b)) \in Q_e$, weil Q die Diagonale von A enthält.

Andernfalls gibt es $c : s_1 \times \cdots \times s_n \to s \in C$ und $a_1, \ldots, a_n, b_1, \ldots, b_n \in A$ mit $a = c^{A_1}(a_1, \ldots, a_n)$, $b = c^{A_2}(b_1, \ldots, b_n)$ und $(a_i, b_i) \in Q$ für alle $1 \leq i \leq n$. Nach Induktionsvoraussetzung gilt $(d'^{\mathcal{A}}(a_i), d'^{\mathcal{A}}(b_i)) \in Q_{e'}$ für alle $1 \leq i \leq n$ und
$d' : s_i \to e' \in D$. Seien $g, g' : V \to A$ Belegungen mit $g(x_i) = a_i$ und $g'(x_i) = b_i$ für alle $1 \leq i \leq n$. Wegen

$$d^{\mathcal{A}}(a) = d^{\mathcal{A}}(c^{A_1}(a_1, \ldots, a_n)) = g^*(t_{d,c}), \quad d^{\mathcal{A}}(b) = d^{\mathcal{A}}(c^{A_2}(b_1, \ldots, b_n) = g'^*(t_{d,c})$$

und weil Q ein $C\Sigma$-Kongruenz ist, folgt $(d^{\mathcal{A}}(a), d^{\mathcal{A}}(b)) \in Q_e$ aus $(g(x_i), g'(x_i)) \in Q$ für alle $1 \leq i \leq n$. ❏

Wegen der Finalität von A ist nach Satz 3.4 (3) die Diagonale von A die einzige $D\Sigma$-Kongruenz auf A. Also impliziert (17), dass Q mit Δ_A^2 übereinstimmt. Sei $c : e \to s \in C$ und $a \in A_e$. Aus $(a, a) \in Q$ und (16) folgt $(c^{A_1}(a), c^{A_2}(a)) \in Q_e$, also $c^{A_1}(a) = c^{A_2}(a)$ wegen $Q = \Delta_A^2$. Demnach gilt $A_1 = A_2$. ❏

XPath and CTL on trees

Let *Label* be a set of node labels. A document tree can then be represented as an ordered labelled tree over $(\mathbb{N}, Label)$ with respect to the usual partial order $\leq$ on $\mathbb{N}$ (see chapter 3).

In this representation, links in the document are dereferenced, i.e., replaced by the documents they point to. Backreferences lead to non-wellfounded trees. As an example, take the abstract syntax of the XML grammar XMLstore of [119], Beispiel 4.7, where *Label* is the set of constructors of the abstract syntax of XMLstore.

A context-free grammar $G = (S, BS, R)$ (see [119]) that provides the concrete syntax of XPath (see, e.g., [27]), relation constructors of relational algebra ($+$, $\wedge$, $\neg$, $join$, $\times$ and div) and the unary logical operators of CTL (computation tree logic) reads as follows:

$$\begin{aligned} S &= \{nodeRel_0, nodeRel_1, nodeRel_2, nodeSet_0, nodeSet_1, nodeSet_2\}, \\ BS &= \{Label, \mathcal{P}(Label), (,), [,], +, \times, /, >>, \exists, \Rightarrow, clos, inv, self, \dots, equiv, \vee, \wedge, \neg, \\ &\quad\; EX, \dots, AG\} \end{aligned}$$

and R consists of the following rules that respect the precedence of multiplicative operators $(/, \wedge)$ over additive operators $(+, \vee)$:

$$
\begin{array}{rcl}
nodeRel_0 & \rightarrow & nodeRel_0 + nodeRel_1 \\
nodeRel_1 & \rightarrow & nodeRel_1 / nodeRel_2 \;\mid\; nodeRel_1 \wedge nodeRel_2 \\
nodeRel_2 & \rightarrow & clos(nodeRel_2) \;\mid\; inv(nodeRel_2) \;\mid\; \neg(nodeRel_2) \;\mid \\
 & & nodeRel_2 >> nodeSet_0 \;\mid\; join(nodeSet_0, nodeRel_2, nodeSet_0) \;\mid \\
 & & nodeSet_0 \times nodeSet_0 \;\mid\; self \;\mid\; child \;\mid\; parent \;\mid\; next \;\mid\; prev \;\mid \\
 & & descendant \;\mid\; ancestor \;\mid\; folSib \;\mid\; preSib \;\mid \\
 & & following \;\mid\; preceding \;\mid\; equal \;\mid\; equiv \;\mid\; (nodeRel_0) \\
nodeSet_0 & \rightarrow & nodeSet_0 \vee nodeSet_1 \\
nodeSet_1 & \rightarrow & nodeSet_1 \wedge nodeSet_2 \\
nodeSet_2 & \rightarrow & Label \;\mid\; \mathcal{P}(Label) \;\mid\; op(nodelRel_0, nodeSet_0) \;\mid\; op'(nodeSet_0) \;\mid \\
 & & (nodeSet_0) \\
op & \rightarrow & \exists \;\mid\; \forall \;\mid\; div \\
op' & \rightarrow & \neg \;\mid\; EX \;\mid\; AX \;\mid\; EF \;\mid\; AF \;\mid\; EG \;\mid\; AG
\end{array}
$$

An abstract syntax of G (see [119]) is given by the signature XCTL$=(S,\mathcal{I},F)$ where F consists of the following flat constructors:

$+, /, \wedge : nodeRel \times nodeRel \to nodeRel$	(union, composition and intersection)
$closure, inv, \neg : nodeRel \to nodeRel$	(transitive closure, inverse and complement)
$>>: nodeRel \times nodeSet \to nodeRel$	(target restriction)
$join : nodeSet \times nodeRel \times nodeSet \to nodeRel$	(join)
$\times : nodeSet \times nodeSet \to nodeRel$	(Cartesian product)
$self, child, \ldots, equiv : 1 \to nodeRel$	(axes and equivalences)
$atom : Label \to nodeSet$	(label predicate)
$atom' : \mathcal{P}(Label) \to nodeSet$	(label predicate)
$true, false : 1 \to nodeSet$	(all and nothing)
$\vee, \wedge : nodeSet \times nodeSet \to nodeSet$	(disjunction and conjunction)
$\exists, \forall, div : nodeRel \times nodeSet \to nodeSet$	(source restrictions)
$\neg, EX, \ldots, AG : nodeSet \to nodeSet$	(unary set operators)

Node set constructors are called *filters* or *qualifiers* in the XPath literature.

Examples

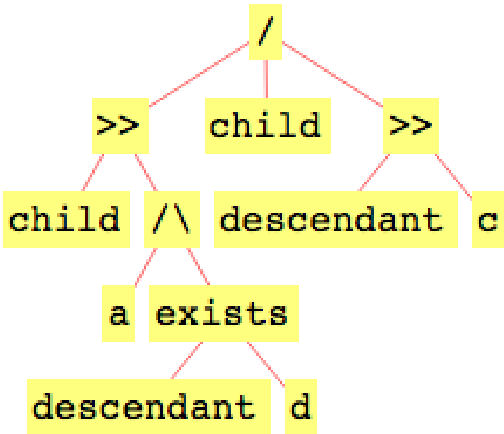
Let $a, c, d, 22, 66 \in Label$.

$$EX(true) \vee d \tag{1}$$

$$EG((\leq 22) \vee (= 66)) \tag{2}$$

$$(child >> (a \wedge \exists(descendant, d))) \; / \; child \; / \; (descendant >> c) \tag{3}$$

The third term has been taken from [147], section 2.2, where the expression is phrased in *CoreXPath*. Its syntax tree reads as follows:



Semantics of XCTL

Let $t \in ltr(\mathbb{N}, Label)$. Folding a ground XCTL-term in the following XCTL-algebra $\mathcal{A}(t)$ with carrier A yields a node relation or node set, respectively:

$$\begin{aligned} A_{nodeSet} &= \mathcal{P}(def(t)), \\ A_{nodeRel} &= \mathcal{P}(def(t))^{def(t)}. \end{aligned}$$

In the following interpretation of F, we use three ω-bi-CPOs, which are derived from t:

- $A_{nodeSet}$ and $\mathcal{C} \Leftrightarrow_{def} \mathcal{P}(def(t)^2)$, both equipped with least and greatest elements, suprema and infima defined for powerset CPOs as usually (see chapter 3),
- $A_{nodeRel}$, equipped with least and greatest elements, suprema and infima defined for CPOs consisting of functions into a CPO (here: $A_{nodeSet}$) as usually (see chapter 3).

The arrows of XCTL are then interpreted in $\mathcal{A}(t)$ as follows:

For all $R, R' \in A_{nodeRel}$, $S, S' \in A_{nodeSet}$, $a \in Label$, $p \subseteq Label$, $w \in def(t)$ and $n \in \mathbb{N}$ such that $wn \in def(t)$,

$$\begin{aligned} (R + R')(w) &= R(w) \cup R'(w), \\ (R/R')(w) &= \bigcup\{R'(v) \mid v \in R(w)\}, \end{aligned}$$

$$
\begin{aligned}
closure(R) &= \Phi^{\infty} \quad \text{(see chapter 3)} \quad \text{where} \\
&\quad \Phi : A_{nodeRel} \;\rightarrow\; A_{nodeRel} \\
&\qquad\; R' \;\mapsto\; R + R/R', \\
inv(R)(w) &= \{v \in def(t) \mid w \in R(v)\}, \\
(\neg R)(w) &= def(t) \setminus R(w), \\
(R \wedge R')(w) &= R(w) \cap R'(w), \\
(R >> S)(w) &= R(w) \cap S, \\
join(S, R, S') &= R \cap (S \times S'), \\
self(w) &= \{w\}, \\
child(w) &= \{wn \mid n \in \mathbb{N}\} \cap def(t), \\
parent(\epsilon) &= \emptyset, \\
parent(wn) &= \{w\}, \\
next(\epsilon) &= \emptyset, \\
next(wn) &= \{w(n+1)\} \cap def(t), \\
prev(\epsilon) &= \emptyset, \\
prev(wn) &= \{w(n-1)\} \cap def(t),
\end{aligned}
$$

$$
\begin{aligned}
descendant &= closure(child),\\
ancestor &= closure(parent) \;=\; inv(descendant),\\
folSib &= closure(next),\\
preSib &= closure(prev) \;=\; inv(folSib),\\
following &= (self + ancestor)/folSib/(self + descendant),\\
preceding &= (self + ancestor)/preSib/(self + descendant) \;=\; inv(following),\\
equal &= \Phi_\infty \quad \text{where} \quad \Phi : \mathcal{C} \to \mathcal{C}\\
&\qquad \sim \;\mapsto\; \{(v,w) \in def(t)^2 \mid t(v) = t(w),\\
&\qquad\qquad \forall\, n \in \mathbb{N} : vn \in def(t) \Leftrightarrow wn \in def(t),\\
&\qquad\qquad \forall\, n \in \mathbb{N} : vn \in def(t) \Rightarrow vn \sim wn\},\\
equiv &= \Phi_\infty \quad \text{where} \quad \Phi : \mathcal{C} \to \mathcal{C}\\
&\qquad \sim \;\mapsto\; \{(v,w) \in def(t)^2 \mid t(v) = t(w),\\
&\qquad\qquad \forall\, b \in child(v)\ \exists\, c \in child(w) : b \sim c,\\
&\qquad\qquad \forall\, c \in child(w)\ \exists\, b \in child(v) : b \sim c\},\\
atom(a) &= \{w \in def(t) \mid t(w) = a\},\\
atom'(p) &= \{w \in def(t) \mid t(w) \in p\},
\end{aligned}
$$

$$
\begin{aligned}
\mathit{true} &= \mathit{def}(t), \\
\mathit{false} &= \emptyset, \\
\neg S &= \mathit{def}(t) \setminus S, \\
S \vee S' &= S \cup S', \\
S \wedge S' &= S \cap S', \\
\exists(R, S) &= \{w \in \mathit{def}(t) \mid R(w) \cap S \neq \emptyset\}, \\
\forall(R, S) &= \{w \in \mathit{def}(t) \mid R(w) \subseteq S\}, \\
\mathit{div}(R, S) &= \{w \in \mathit{def}(t) \mid S \subseteq R(w)\}, \\
EX(S) &= \exists(\mathit{child}, S), \\
AX(S) &= \forall(\mathit{child}, S) \;=\; \neg EX(\neg S), \\
EF(S) &= \exists(\mathit{self} + \mathit{descendant}, S), \\
AF(S) &= \Phi^{\infty} \quad \text{where} \\
&\quad \Phi : A_{nodeSet} \to A_{nodeSet} \\
&\quad\quad S' \mapsto S \vee (AX(S') \wedge EX(\mathit{true})),
\end{aligned}
$$

$$
\begin{aligned}
EG(S) \;&=\; \Phi_\infty \;=\; \neg AF(\neg S) \quad \text{where} \\
&\qquad \Phi : A_{nodeSet} \;\rightarrow\; A_{nodeSet} \\
&\qquad\qquad\qquad S' \;\mapsto\; S \wedge (EX(S') \vee AX(\mathit{false})), \\
AG(S) \;&=\; \forall(\mathit{self} + \mathit{descendant}, S) \;=\; \neg EF(\neg S).
\end{aligned}
$$

For a complete Haskell implementation of XCTL and its semantics, see the section on tree logics in *Painter.hs*, the main program of Painter.tgz.

Painter.hs also provides two functions *drawNodeSet* und *drawNodeRel* for the graphical representation of node sets or relations that result from evaluating XCTL-terms (see *Painter.pdf*).

Let $t \in ltr(\mathbb{N}, Label)$.

Subsumption For all $\varphi, \psi \in T_{XCTL}$ and

$$\varphi \sqsubseteq_t \psi \;\Leftrightarrow_{def}\; fold^{\mathcal{A}(t)}(\varphi) \subseteq fold^{\mathcal{A}(t)}(\psi).$$

Propositions

For all $\varphi, \psi \in T_{XCTL}$, $\varphi \sqsubseteq_t \psi \Leftrightarrow fold^{\mathcal{A}(t)}(\varphi \wedge \neg\psi) = \emptyset$.

For all $\varphi \in T_{XCTL,nodeRel}$ and $\psi \in T_{XCTL,nodeSet}$, $fold^{\mathcal{A}(t)}(div(\varphi, \psi) \times \psi) = fold^{\mathcal{A}(t)}(\varphi)$.

A variant of XCTL captures description logics: The respective *domain of individuals* replaces $def(t)$. *Concepts* and *rôles* interpret the sorts $nodeSet$ and $nodeRel$, respectively. *Atomic concepts* and *atomic rôles* replace the above $nodeSet$- and $nodeRel$-constants, respectively. $\exists(R, S)$ and $\forall(R, S)$ are written as $\exists R.S$ and $\forall R.S$, respectively.

Predicates

Let $\mathcal{A}, \mathcal{B}$ be Σ-structures. Then a Σ-homomorphism $h : \mathcal{A} \to \mathcal{B}$ is P**-compatible** if for all $p : \mathcal{P}(e) \in P$ and $a \in p^{\mathcal{A}}$, $h(a) \in p^{\mathcal{B}}$.

Proposition NEGFREE

For all negation-free $\varphi \in Fo_{\Sigma}(V)$ and Σ-homomorphisms $h : \mathcal{A} \to \mathcal{B}$ and $g \in \varphi^{\mathcal{A}}$, $h \circ g \in \varphi^{\mathcal{B}}$.

Lemma MUPRED

Let $SP = (\Sigma, P, AX)$ be a Horn specification and $\mathcal{A}$ be a Σ'-algebra with carrier A that satisfies AX.

Every Σ-homomorphism $h : \mathcal{C} \to \mathcal{A}|_{\Sigma}$ is a Σ'-homomorphism from $lfp(\Phi)$ to $\mathcal{A}$.

In particular, if $\mathcal{C}$ is initial in Alg_{Σ}, then $lfp(\Phi)$ is initial in $Alg_{\Sigma',AX}$.

Proof. It is sufficient to show that for all $p \in P$,

$$h(p^{lfp(\Phi)}) \subseteq p^{\mathcal{A}},$$

or, equivalently, by Theorem CONSTEP (i) and Theorem kleene0 (1), for all $i \in \mathbb{N}$,

$$h(p^{\Phi^i(\bot)}) \subseteq p^{\mathcal{A}}. \quad (9)$$

Case 1: $i = 0$. Since $p^\bot = \emptyset$, (9) holds true trivially.

Case 2: Let $i > 0$ and $c \in p^{\Phi^i(\bot)}$. Then $c = t^{\mathcal{C}}(g)$ for some $\varphi \Rightarrow p(t) \in AX$ and $g \in \varphi^{\Phi^{i-1}(\bot)}$.

By induction hypothesis, h is a Σ'-homomorphism from $\Phi^{i-1}(\bot)$ to $\mathcal{A}$. Hence by Proposition NEGFREE, $h \circ g \in \varphi^{\mathcal{A}}$. Since $\mathcal{A}$ satisfies $p(t) \Leftarrow \varphi$, we conclude $h \circ g \in p(t)^{\mathcal{A}} = \{f \in A^V \mid f^*(t) \in p^{\mathcal{A}}\}$ and thus

$$h(c) = h(t^{\mathcal{C}}(g)) = (h \circ g)^*(t) \in p^{\mathcal{A}}.$$

Hence again, (9) holds true. ❑

Lemma NUPRED

Let $SP = (\Sigma, P, AX)$ be a co-Horn specification and $\mathcal{A}$ be a Σ'-algebra with carrier A that satisfies AX.

Every Σ-homomorphism $h : \mathcal{A}|_\Sigma \to \mathcal{C}$ is a Σ'-homomorphism from $\mathcal{A}$ to $gfp(\Phi)$.

In particular, if $\mathcal{C}$ is final in Alg_{Σ}, then $gfp(\Phi)$ is final in $Alg_{\Sigma',AX}$.

Proof. It remains to show that for all $p : e \in P$,

$$h(p^{\mathcal{A}}) \subseteq p^{gfp(\Phi)},$$

or, equivalently, by Theorem CONSTEP (ii) and Theorem kleene0 (2), for all $i \in \mathbb{N}$,

$$h(p^{\mathcal{A}}) \subseteq p^{\Phi^i(\top)}. \tag{10}$$

Let $e = s_1 \times \cdots \times s_n$.

Case 1: $i = 0$. Since $p^{\top} = C_e$, (10) holds true trivially.

Case 2: Let $i > 0$ and $c = (c_1, \dots, c_n) \in C_e \setminus p^{\Phi^i(\top)}$.

Then there are $t = (t_1, \dots, t_n) \in T_{\Sigma}(X)^n$, $ax = (p(t) \Rightarrow \varphi) \in AX$ and

$$g \in C^V \setminus \varphi^{\Phi^{i-1}(\top)} \tag{11}$$

such that $c = t^{\mathcal{C}}(g)$. Let $X = \{x_1, \dots, x_n\}$ be a set of pairwise different variables disjoint from $var(ax)$. Let $\{z_1, \dots, z_m\} = var(ax)$ and $\psi = (\forall z_1 \dots \forall z_m(\varphi \vee \bigvee_{k=1}^n x_k \neq t_k))$.

Obviously, ax is equivalent to the Σ'-formula $ax' = (p(x_1, \dots, x_n) \Rightarrow \psi)$.

W.l.o.g. $g(x_k) = g^*(t_k)$ for all $1 \le k \le n$. It remains to show $c \notin h(p^A)$.

Hence assume that there is $a = (a_1, \ldots, a_n) \in p^A$ with $c = h(a)$. Let $f \in A^V$ be such that $f(x_k) = a_k$ for all $1 \leq k \leq n$. Then for all $1 \leq k \leq n$,

$$h(f(x_k)) = h(a_k) = c_k = g^*(t_k) = g(x_k). \tag{12}$$

$f(x_1, \ldots, x_n) = a \in p^A$ implies $f \in p(x_1, \ldots, x_n)^A$ and thus $f \in \psi^A$ because $A \models ax$ implies $A \models ax'$.

By induction hypothesis, h is a Σ'-homomorphism from A to $\Phi^{i-1}(\top)$. Hence by Proposition NEGFREE, $h \circ f \in \psi^{\Phi^{i-1}(\top)}$ and thus

$$h \circ f \in \psi^{\Phi^{i-1}(\top)}. \tag{13}$$

Since all variables of $var(\psi) \setminus X$ are universally quantified, (12) and (13) imply

$$g \in (\varphi \vee \bigvee_{k=1}^{n} x_k \neq t_k)^{\Phi^{i-1}(\top)}$$

and thus $g \in \varphi^{\Phi^{i-1}(\top)}$ because $g(x_k) = g^*(t_k)$ for all $1 \leq k \leq n$. $g \in \varphi^{\Phi^{i-1}(\top)}$ contradicts (11). Hence $c \notin h(p^A)$. ❑

Bibliography

[1] A. Abel, B. Pientka, D. Thibodeau, A. Setzer, *Copatterns: Programming Infinite Structures by Observations*, Proc. ACM POPL (2013) 27-38

[2] P. Aczel, *An Introduction to Inductive Definitions*, in: J. Barwise, ed., Handbook of Mathematical Logic, North-Holland (1977) 739-782

[3] P. Aczel, J. Adamek, J. Velebil, *A Coalgebraic View of Infinite Trees and Iteration*, Proc. Coalgebraic Methods in Computer Science, Elsevier ENTCS 44 (2001) 1-26

[4] J. Adamek, *Free algebras and automata realizations in the language of categories*, Commentat. Math Univers. Carolinae 15 (1974) 589-602

[5] J. Adamek, *Final coalgebras are ideal completions of initial algebras*, Journal of Logic and Computation 12 (2002) 217-242

[6] J. Adamek, *Introduction to Coalgebra*, Theory and Applications of Categories 14 (2005) 157-199

[7] J. Adamek, *A Logic of Coequations*, Proc. CSL 2005, Springer LNCS 3634 (2005) 70-86

[8] J. Adamek, M. Haddadi, S. Milius, *Corecursive Algebras, Corecursive Monads and Bloom Monads*, Logical Methods in Computer Science 10 (2014) 1-51

[9] J. Adamek, D. Lücke, S. Milius, *Recursive Coalgebras of Finitary Functors*, Theor. Inform. and Appl. 41 (2007) 447–462

[10] J. Adamek, S. Milius, L.S. Moss, *Initial algebras and terminal coalgebras: a survey*, draft of Feb. 7, 2011, TU Braunschweig

[11] J. Adamek, H.–E. Porst, *On varieties and covarieties in a category*, Math. Structures in Computer Science 13 (2003) 201-232

[12] J. Adamek, H.–E. Porst, *On Tree Coalgebras and Coalgebra Presentations*, Theoretical Computer Science 311 (2004) 257-283

[13] Th. Altenkirch, *Naïve Type Theory*, in: Reflections on the Foundations of Mathematics, Springer (2019) 101-136

[14] M.A. Arbib, *Free dynamics and algebraic semantics*, Proc. Fundamentals of Computation Theory, Springer LNCS 56 (1977) 212-227

[15] M.A. Arbib, E.G. Manes, *Arrows, Structures, and Functors*, Academic Press 1975

[16] M.A. Arbib, E.G. Manes, *Parametrized Data Types Do Not Need Highly Constrained Parameters*, Information and Control 52 (1982) 139-158

[17] E. Astesiano, H.-J. Kreowski, B. Krieg-Brückner, eds., *Algebraic Foundations of Systems Specification*, IFIP State-of-the-Art Report, Springer 1999

[18] R. Backhouse, R. Crole, J. Gibbons, eds., *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction*, Tutorial, Springer LNCS 2297 (2002)

[19] A. Ballester-Bolinches, E. Cosme-Llópez, J. Rutten, *The dual equivalence of equations and coequations for automata*, Information and Computation 244 (2015) 49–75

[20] M. Barr, *Terminal coalgebras in well-founded set theory*, Theoretical Computer Science 114 (1993) 299-315

[21] M. Barr, *Terminal coalgebras for endofunctors on sets*, ftp://ftp.math.mcgill.ca/pub/barr/pdffiles/trmclg.pdf, McGill University, Montreal 1999

[22] M. Barr, *Coequalizers and Free Triples*, Math. Zeitschrift 116 (1970) 307-322

[23] M. Barr, Ch. Wells, *Category Theory*, Lecture Notes for ESSLLI, 1999

[24] M. Barr, Ch. Wells, *Category Theory for Computing Science*, Reprints in Theory and Applications of Categories 22, 2012.

[25] F. Bartels, *Generalised Coinduction*, Proc. CMCS 2001, Elsevier ENTCS 44 (2001)

[26] A. Bauer, *What is algebraic about algebraic effects and handlers?*, submitted

[27] M. Benedikt, C. Koch, *XPath Leashed*, ACM Computing Surveys 41 (2009) 3:1-3:54

[28] R. Bird, O. de Moor, *Algebra of Programming*, Prentice Hall 1997

[29] S.L. Bloom, E.G. Wagner, Many-sorted theories and their algebras with some applications to data types, in: Maurive Nivat, John C. Reynolds: Algebraic Methods in Semantics, Cambridge University Press (1985) 133-168

[30] L.S. Bobrow, M.A. Arbib, *Discrete Mathematics: Applied Algebra for Computer and information Science*, W.B. Saunders Company 1974

[31] F. Bonchi, M. Bonsangue, M. Boreale, J. Rutten, A. Silva, *A coalgebraic perspective on linear weighted automata*, Information and Computation 211 (2012) 77–105

[32] M. Bonsangue, J. Rutten, A. Silva, *An Algebra for Kripke Polynomial Coalgebras*, Proc. 24th LICS (2009) 49-58

[33] M. Brandenburg, *Einführung in die Kategorientheorie*, Springer 2016

[34] J.A. Brzozowski, *Derivatives of regular expressions*, Journal ACM 11 (1964) 481–494

[35] D. Cancila, F. Honsell, M. Lenisa, *Generalized Coiteration Schemata*, Elsevier ENTCS 82 (2003)

[36] V. Capretta, T. Uustalu, V. Vene, *Recursive coalgebras from comonads*, Information and Computation 204 (2006) 437-468

[37] V. Capretta, T. Uustalu, V. Vene, *Corecursive algebras: A study of general structured corecursion*, Springer LNCS 5902 (2009) 84–100

[38] R. Cockett, T. Fukushima, *About Charity*, Yellow series report 92/480/18, Dept. of Comput. Sci., Univ. of Calgary (1992)

[39] J.R.B. Cockett, D. Spencer, *Strong categorical datatypes II: A term logic for categorical programming*, Theoretical Computer Science 139 (1995) 69-113

[40] H. Comon et al., *Tree Automata: Techniques and Applications*, Inria 2008

[41] C. Cîrstea, *A coalgebraic equational approach to specifying observational structures*, Theoretical Computer Science 280 (2002) 35–68

[42] J. Cristau, C. Löding, W. Thomas, *Deterministic automata on unranked trees*, Proc. 15th FCT, Springer LNCS 3623 (2005) 68–79

[43] A. Cunha, *Recursion Patterns as Hylomorphisms*, Technical Report DI-PURe-03.11.01, Department of Informatics, University of Minho, Portugal 2003

[44] F. Drewes, ed., *Tree Automata*, Course notes, Umeå University, Sweden 2009

[45] M. Droste, P. Gastin, *Weighted automata and weighted logics*, Theoretical Computer Science 380 (2007) 69–86

[46] A. Dudenhefner, *Untersuchung und Implementierung des coinduktiven Stromkalküls*, Bachelor thesis, TU Dortmund 2011

[47] H. Ehrig, B. Mahr, F. Cornelius, M. Große-Rhode, P. Zeitz, *Mathematisch-strukturelle Grundlagen der Informatik*, Springer 2001

[48] M. Erwig, *Categorical Programming with Abstract Data Types*, Proc. AMAST'98, Springer LNCS 1548, 406-421

[49] B. Fong, D.I. Spivak, *Seven Sketches in Compositionality: An Invitation to Applied Category Theory*, https://math.mit.edu/ dspivak/teaching/sp18/7Sketches.pdf

[50] M.M. Fokkinga, E. Meijer, *Program Calculation Properties of Continuous Algebras*, CWI Report CS-R9104, Amsterdam 1991

[51] J. Gibbons, G. Hutton, Th. Altenkirch, *When is a function a fold or an unfold?*, Elsevier ENTCS 44 (2001) 146-160

[52] J.A. Goguen, R. Burstall, *Institutions: Abstract Model Theory for Specification and Programming*, J. ACM 39 (1992) 95-146

[53] J.A. Goguen, J.W. Thatcher, E.G. Wagner, J.B. Wright, *Initial Algebra Semantics and Continuous Algebras*, J. ACM 24 (1977) 68-95

[54] R. Goldblatt, *A Calculus of Terms for Coalgebras of Polynomial Functors*, Elsevier ENTCS 44 (2001) 161-184

[55] H.P. Gumm, T. Schröder, *Coalgebras of bounded type*, Math. Structures in Computer Science 12 (2002) 565-578

[56] H.P. Gumm, *State Based Systems are Coalgebras*, Cubo - Matematica Educacional 5 (2003) 239-262

[57] H.P. Gumm, *Equational and implicational classes of coalgebras*, Theoretical Computer Science 260 (2001) 57-69

[58] H.P. Gumm, *Universelle Coalgebra*, in: Th. Ihringer, *Allgemeine Algebra*, Heldermann Verlag 2003

[59] G. Gupta et al., *Infinite Computation, Co-induction and Computational Logic*, Proc. CALCO 2011, Springer LNCS 6859 (2011) 40-54

[60] T. Hagino, *Codatatypes in ML*, J. Symbolic Computation 8 (1989) 629-650

[61] H.H. Hansen, C. Kupke, J. Rutten, *Stream Differential Equations: Specification Formats and Solution Methods*, 2016

[62] H.H. Hansen, J. Rutten, *Symbolic Synthesis of Mealy Machines from Arithmetic Bitstream Functions*, Scientific Annals of Computer Science (2010) 97-130

[63] I. Hasuo, B. Jacobs, A. Sokolova, *Generic Trace Theory*, Proc. CMCS 2006, Elsevier ENTCS 164, 47-65

[64] J.G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, R. Paige, Th. Rauhe, A. Sandholm, *Mona: Monadic second-order logic in practice*, Proc. TACAS 1995, Springer LNCS 1019, 89-110

[65] R. Hinze, *Adjoint Folds and Unfolds*, Proc. Mathematics of Program Construction 2010, Springer LNCS 6120, 195–228 *link*

[66] R. Hinze, *Adjoint Folds and Unfolds—An extended study*, Science of Computer Programming 78 (2013) 2108-2159

[67] R. Hinze, *Reasoning about codata*, Third Central European Functional Programming School, Springer LNCS 6299 (2010) 42-93

[68] R. Hinze, *Functional Pearl: Streams and Unique Fixed Points*, Proc. 13th ICFP (2008) 189-200

[69] R. Hinze, D.W.H. James, *Proving the Unique-Fixed Point Principle Correct*, Proc. 16th ICFP (2011) 359-371

[70] R. Hinze, N. Wu, J. Gibbons, *Conjugate Hylomorphisms*, Proc. ACM POPL (2015) 527-538

[71] G. Hutton, *Fold and unfold for program semantics*, Proc. 3rd ICFP (1998) 280-288

[72] B. Jacobs, *Invariants, Bisimulations and the Correctness of Coalgebraic Refinements*, Proc. Algebraic Methodology and Software Technology, Springer LNCS 1349 (1997) 276-291

[73] B. Jacobs, *Introduction to Coalgebra*, Cambridge University Press 2017

[74] B. Jacobs, *Exercises in Coalgebraic Specification*, Springer LNCS 2297 (2002) 237-280

[75] B. Jacobs, *A Bialgebraic Review of Deterministic Automata, Regular Expressions and Languages*, in: K. Futatsugi et al. (eds.), Goguen Festschrift, Springer LNCS 4060 (2006) 375–404

[76] B. Jacobs, *Trace Semantics for Coalgebras*, CMCS 2004

[77] B. Jacobs, J. Rutten, *A Tutorial on (Co)Algebras and (Co)Induction*, EATCS Bulletin 62 (1997) 222-259

[78] B. Jacobs, J. Rutten, *An introduction to (co)algebras and (co)induction*, in: D. Sangiorgi, J. Rutten (eds), Advanced topics in bisimulation and coinduction, Cambridge Univ. Press (2012) 38-99

[79] G. Jarzembski, *A new proof of Reiterman's theorem*, Cahiers de topologie et géométrie différentielle catégoriques 35 (1994) 239-247

[80] S. Kamin, *Final data types an their specification extension*, ACM Trans. on Prog. Lang. and Systems Comp. Syst. Sci. 5 (1983) 97-123

[81] S.C. Kleene, *Introduction to Metamathematics*, Van Nostrand 1952

[82] B. Klin, *Structural Operational Semantics for Weighted Transition Systems*, Mosses Festschrift, Springer LNCS 5700 (2009) 121–139

[83] B. Klin, *Bialgebras for structural operational semantics: An introduction*, Theoretical Computer Science 412 (2011) 5043-5069

[84] D. Kozen, *Realization of Coinductive Types*, Proc. Math. Foundations of Prog. Lang. Semantics 27, Carnegie Mellon University, Pittsburgh 2011

[85] C. Kupke, M. Niqui, J. Rutten, *Stream Differential Equations: concrete formats for coinductive definitions*, 2011

[86] C. Kupke, Y. Venema, *Coalgebraic automata theory: basic results*, Logical Methods in Computer Science 4 (2008) 1–43

[87] A. Kurz, *Specifying coalgebras with modal logic*, Theoretical Computer Science 260 (2001) 119–138

[88] A. Kurz, J. Velebil, *Relation lifting, a survey*, Journal of Logical and Algebraic Methods in Programming 85 (2016) 475–499

[89] J. Lambek, *A fixpoint theorem for complete categories*, Math. Zeitschrift 103 (1968) 151-161

[90] J.-L. Lassez, V.L. Nguyen, E.A. Sonenberg, *Fixed Point Theorems and Semantics: A Folk Tale*, Information Processing Letters 14 (1982) 112-116

[91] F.W. Lawvere, *Diagonal arguments in cartesian closed categories*, Reprints in Theory and Applications of Categories 15 (2006) 1–13

[92] D.J. Lehmann, M.B. Smyth, *Algebraic Specification of Data Types: A Synthetic Approach*, Math. Systems Theory 14 (1981) 97-139

[93] Z. Manna, *Mathematical Theory of Computation*, McGraw-Hill 1974

[94] E.G. Manes, M.A. Arbib *Algebraic Approaches to Program Semantics*, Springer 1986

[95] G. Markowsky, *Chain-complete posets and directed sets with applications*, Algebra Universalis 6 (1976) 53-68

[96] E. Meijer, M. Fokkinga, and R. Paterson, *Functional programming with bananas, lenses, envelopes and barbed wire*, Proc. FPCA '91, Springer LNCS 523 (1991) 124-144

[97] E. Meijer, G. Hutton, *Bananas in Space: Extending Fold and Unfold to Exponential Types*, Proc. FPCA '95, ACM Publications (1995) 324-333

[98] B. Milewski, *Category Theory for Programmers*, https://bartoszmilewski.com/2014/10/28/category-theory-for-programmers-the-preface

[99] R. Milner, *Communication and Concurrency*, Prentice-Hall 1989

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

[101] T. Mossakowski, L. Schröder, M. Roggenbach, H. Reichel, *Algebraic-coalgebraic specification in CoCASL*, J. Logic and Algebraic Programming 67 (2006) 146-197

[102] D. Orchard, *Should I use a Monad or a Comonad?*, submitted to MSFP 2012

[103] P. Padawitz, *Church-Rosser-Eigenschaften von Graphgrammatiken und Anwendungen auf die Semantik von LISP*, Diplomarbeit, TU Berlin 1978

[104] P. Padawitz, Computing in Horn Clause Theories, EATCS Monographs on Theoretical Computer Science 16, Springer-Verlag, 1988 (free copies are available from the author)

[105] P. Padawitz, Deduction and Declarative Programming, Cambridge Tracts in Theoretical Computer Science 28, Cambridge University Press, 1992

[106] P. Padawitz, Swinging Types = Functions + Relations + Transition Systems, Theoretical Computer Science 243 (2000) 93-165

[107] P. Padawitz, *Expander2: program verification between interaction and automation*, slides for [114], WFLP 2006

[108] P. Padawitz, *Expander2: Two inductive proofs*, Video, TU Dortmund 2017

[109] P. Padawitz, *Formale Methoden des Systementwurfs*, TU Dortmund 2007

[110] P. Padawitz, *Swinging Data Types*, TU Dortmund 2009

[111] P. Padawitz, *Dialgebraic Specification and Modeling*, TU Dortmund 2010

[112] P. Padawitz, *Algebraic Model Checking*, in: F. Drewes, A. Habel, B. Hoffmann, D. Plump, eds., Manipulation of Graphs, Algebras and Pictures, Electronic Communications of the EASST Vol. 26 (2010)

[113] P. Padawitz, *From grammars and automata to algebras and coalgebras*, Proc. CAI 2011, Springer LNCS 6742 (2011) 21-43

[114] P. Padawitz, *Expander2 as a Prover and Rewriter*, TU Dortmund 2012

[115] P. Padawitz, *From fixpoint to predicate co/induction and its use in standard models*, TU Dortmund 2014

[116] P. Padawitz, *(Co)Algebraic Specification with Base Sets, Recursive and Iterative Equations*, IFIP WG 1.3 Meeting 2014

[117] P. Padawitz, *Modeling and reasoning with I-polynomial data types*, IFIP WG 1.3 Meeting + CMS 2016

[118] P. Padawitz, *Modellieren und Implementieren in Haskell*, TU Dortmund 2017

[119] P. Padawitz, *Übersetzerbau (Algebraic Compiler Construction*, TU Dortmund 2016

[120] P. Padawitz, *Logik für Informatiker (Logic for Computer Scientists)*, TU Dortmund 2017

[121] P. Padawitz, *From Modal Logic to (Co-)Algebraic Reasoning*, TU Dortmund 2017

[122] P.K. Pandya, *Monadic Second-Order Logic - Automata: Theory and Practice*, course notes, TIFR, Mumbai, India 2005

[123] D. Pattinson, *An Introduction to the Theory of Coalgebras*, Course notes for the North American Summer School in Logic, Language and Information (NASSLLI), LMU München, Germany 2003

[124] D. Pattinson, L. Schröder, *Program Equivalence is Coinductive*, Proc. 21st LICS (2016), 337-346

[125] D. Pavlovic and M. Escardó, *Calculus in coinductive form*, Proc. 13th LICS (1998), 408-417

[126] S. Phillips, W.H. Wilson, G.S. Halford, *What Do Transitive Inference and Class Inclusion Have in Common? Categorical (Co)Products and Cognitive Development*, PLoS Computational Biology 5 (2009)

[127] S. Phillips, W.H. Wilson, *Categorial Compositionality: A Category Theory Explanation for the Systematicity of Human Cognition*, PLoS Computational Biology 6 (2010)

[128] B. Pierce, *Basic Category Theory for Computer Scientists*, MIT Press 1991

[129] G.D. Plotkin, J. Power, *Tensors of comodels and models for operational semantics*, Elsevier ENTCS 218 (2008) 295–311

[130] C. Pulte, *Natürliche Transformationen*, Lecture in the Proseminar *Kategorientheoretische Grundlagen*, TU Dortmund 2012

[131] H. Reichel, *An Approach to Object Semantics based on Terminal Coalgebras*, Mathematic Structures in Computer Science 5 (1995) 129-152

[132] H. Reichel, *Dialgebraic Logics*, Elsevier ENTCS 11 (1998) 1-9

[133] H. Reichel, *An Algebraic Approach to Regular Sets*, in: K. Futatsugi et al., Goguen Festschrift, Springer LNCS 4060 (2006) 449-458

[134] J. Reiterman, *The Birkhoff theorem for finite algebras*, Algebra Universalis 14 (1982) 1-10

[135] J. Rothe, H. Tews, B. Jacobs, *The Coalgebraic Class Specification Language CCSL*, Journal of Universal Computer Science 7 (2001) 175-193

[136] W.C. Rounds, *Mappings and Grammars on Trees*, Mathematical Systems Theory 4 (1970) 256-287

[137] J. Rutten, *Processes as terms: non-wellfounded models for bisimulation*, Math. Struct. in Comp. Science 15 (1992) 257-275

[138] J. Rutten, *Universal coalgebra: a theory of systems*, Theoretical Computer Science 249 (2000) 3-80

[139] J. Rutten, *Automata and coinduction (an exercise in coalgebra)*, Proc. CONCUR '98, Springer LNCS 1466 (1998) 194–218

[140] J. Rutten, *Automata, Power Series, and Coinduction: Taking Input Derivatives Seriously*, Proc. ICALP '99, Springer LNCS 1644 (1998) 645-654

[141] J. Rutten, *Behavioral differential equations: a coinductive calculus of streams, automata, and power series*, Theoretical Computer Science 308 (2003) 1-53

[142] J. Rutten, *On Streams and Coinduction*, in: *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, CRM Monograph Series 23 (2004) 1-92

[143] J. Rutten, *A coinductive calculus of streams*, Math. Struct. in Comp. Science 15 (2005) 93-147

[144] J. Salamanca, M. Bonsangue, J. Rutten, *Equations and Coequations for Weighted Automata*, Proc. MFCS 2015, Springer LNCS 9234 (2015) 444–456

[145] D. Sannella, A. Tarlecki, *Foundations of Algebraic Specification and Formal Software Development*, Springer 2012

[146] D. Schwencke, *Coequational logic for accessible functors*, Information and Computation 208 (2010) 1469–1489

[147] T. Schwentick, *XPath query containment*, SIGMOD Record 33 (2004) 101–109

[148] K. Sen, G. Rosu, *Generating Optimal Monitors for Extended Regular Expressions*, Proc. Runtime Verification 2003, Elsevier ENTCS 89 (2003) 226-245

[149] L. Simon, *Coinductive Logic Programming*, Ph.D. thesis, University of Texas at Dallas (2006)

[150] A. Silva, J. Rutten, *A coinductive calculus of binary trees*, Information and Computation 208 (2010) 578–593

[151] A. Silva, F. Bonchi, M. Bonsangue, J. Rutten, *Quantitative Kleene coalgebras*, Information and Computation 209 (2011) 822-849

[152] J. Soto-Andrade, F.J. Varela, *Self-Reference and Fixed Points: A Discussion and an Extension of Lawvere's Theorem*, Acta Applicandae Mathematicae 2 (1984) 1-19

[153] D.I. Spivak, T. Giesa, E. Wood, M.J. Buehler, *Category Theoretic Analysis of Hierarchical Protein Materials and Social Networks*, www.plosone.org 2011

[154] D.I. Spivak, R.E. Kent, *Ologs: A Categorical Framework for Knowledge Representation*, www.plosone.org 2012

[155] D.I. Spivak, Category Theory for the Sciences, MIT Press 2014

[156] A. Tarski, *A lattice-theoretical fixpoint theorem and its applications*, Pacific J. Math. 5 (1955), 285-309

[157] W. Thomas, *Languages, Automata, and Logic*, in: Handbook of Formal Languages, Vol. 3: Beyond Words, Springer (1997) 389-456

[158] W. Thomas, *Applied Automata Theory*, Course Notes, RWTH Aachen (2005)

[159] D. Turi, G. Plotkin, *Towards a Mathematical Operational Semantics*, Proc. 12th LICS (1997) 280-291

[160] 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

[161] J.W. Thatcher, J.B. Wright, *Generalized Finite Automata Theory with an Application to a Decision Problem of Second-Order Logic*, Theory of Computing Systems 2 (1968) 57-81

[162] T. Uustalu, V. Vene, *Primitive (Co)Recursion and Course-of-Value (Co)Iteration*, INFORMATICA 10 (1999) 5-26

[163] Ph. Wadler, *Theorems for free!*, Proc. FPLCA '89, ACM Press (1989) 347-359

[164] E.G. Wagner, J.B. Wright, J.A. Goguen, J.W. Thatcher, *Some Fundamentals of Order-Algebraic Semantics*, IBM Research Report 6020 (1976)

[165] E.G. Wagner, J.W. Thatcher, J.B. Wright, *Free continuous theories*, IBM Research Report 6906 (1977)

[166] E.G. Wagner, S.L. Bloom, J.W. Thatcher, *Why algebraic theories?*, in: Maurive Nivat, John C. Reynolds: Algebraic Methods in Semantics, Cambridge University Press (1985) 607-634

[167] E.G. Wagner, *Algebraic semantics*, in: Handbook of Logic in Computer Science 3: Semantic Structures, Clarendon Press (1994) 323-393

[168] M. Wand, *Final algebra semantics and data type extension*, J. Comp. Syst. Sci. 19 (1979) 27-44

[169] R.F.C. Walters, *Categories and Computer Science*, Cambridge University Press 1992

[170] J. Winter, M.M. Bonsague, J. Rutten, *Context-Free Languages, Coalgebraically*, Proc. CALCO 2011

[171] M. Wirsing, *Structured Algebraic Specifications: A Kernel Language*, ?Theoretical Computer Science 42 (1986) 123-249

[172] M. Wirsing, *Algebraic Specification*, in: J. van Leeuwen, ed., Handbook of Theoretical Computer Science, Elsevier (1990) 675-788

[173] N.S. Yanofsky, *A Universal Approach to Self-Referential Paradoxes, Incompleteness and Fixed Points*, The Bulletin of Symbolic Logic 9 (2003) 362-386