Saturated Semantics for Coalgebraic Logic Programming

Filippo Bonchi, **Fabio Zanasi**

École Normale Supérieure de Lyon, France

CALCO 2013

Coalgebraic Logic Programming (the ground case)

[E. Komendantskaya, G. McCusker & J. Power 2010]

$$
\begin{array}{rcl}
\mathtt{r(b,c)} & \leftarrow & \mathtt{q(a),q(b),q(c)} \\
\mathtt{r(b,b)} & \leftarrow & \mathtt{r(b,a),r(b,c)} \\
\mathtt{r(b,b)} & \leftarrow & \mathtt{q(c)} \\
\mathtt{q(c)} & \leftarrow &
\end{array}
$$

Coalgebraic Logic Programming (the ground case)

[E. KOMENDANTSKAYA, G. MCCUSKER & J. POWER 2010]

```
r(b,c)  ←  q(a),q(b),q(c)
r(b,b)  ←  r(b,a),r(b,c)
r(b,b)  ←  q(c)
  q(c)  ←
```

Coalgebra in Set

$$p \quad : \quad At \to \mathcal{P}_f\mathcal{P}_f(At)$$

Coalgebraic Logic Programming (the ground case)

[E. KOMENDANTSKAYA, G. MCCUSKER & J. POWER 2010]

$$p(\texttt{r(b,b)}) = \{\{\texttt{r(b,a)}, \texttt{r(b,c)}\}, \{\texttt{q(c)}\}\}$$

$$\begin{array}{rcl}
\texttt{r(b,c)} & \leftarrow & \texttt{q(a)}, \texttt{q(b)}, \texttt{q(c)} \\
\texttt{r(b,b)} & \leftarrow & \texttt{r(b,a)}, \texttt{r(b,c)} \\
\texttt{r(b,b)} & \leftarrow & \texttt{q(c)} \\
\texttt{q(c)} & \leftarrow &
\end{array}$$

Coalgebra in **Set**

$$p \quad : \quad At \to \mathcal{P}_f \mathcal{P}_f (At)$$

Coalgebraic Logic Programming (the ground case)

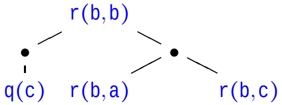
[E. KOMENDANTSKAYA, G. MCCUSKER & J. POWER 2010]

$$p(\mathtt{r(b,b)}) = \{\{\mathtt{r(b,a)}, \mathtt{r(b,c)}\}, \{\mathtt{q(c)}\}\}$$

$$\begin{array}{rcl}
\mathtt{r(b,c)} & \leftarrow & \mathtt{q(a)}, \mathtt{q(b)}, \mathtt{q(c)} \\
\mathtt{r(b,b)} & \leftarrow & \mathtt{r(b,a)}, \mathtt{r(b,c)} \\
\mathtt{r(b,b)} & \leftarrow & \mathtt{q(c)} \\
\mathtt{q(c)} & \leftarrow &
\end{array}$$

Coalgebra in **Set**

$$p \quad : \quad At \to \mathcal{P}_f \mathcal{P}_f (At)$$



Coalgebraic Logic Programming (the ground case)

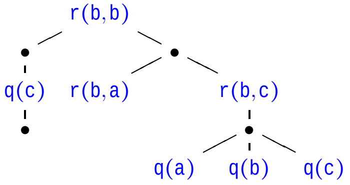
[E. KOMENDANTSKAYA, G. MCCUSKER & J. POWER 2010]

$$p(\texttt{r(b,b)}) = \{\{\texttt{r(b,a)}, \texttt{r(b,c)}\}, \{\texttt{q(c)}\}\}$$

$$
\begin{array}{rcl}
\texttt{r(b,c)} & \leftarrow & \texttt{q(a)}, \texttt{q(b)}, \texttt{q(c)} \\
\texttt{r(b,b)} & \leftarrow & \texttt{r(b,a)}, \texttt{r(b,c)} \\
\texttt{r(b,b)} & \leftarrow & \texttt{q(c)} \\
\texttt{q(c)} & \leftarrow &
\end{array}
$$

Coalgebra in **Set**

$$p \quad : \quad At \to \mathcal{P}_f\mathcal{P}_f(At)$$

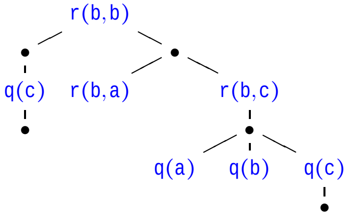


Coalgebraic Logic Programming (the ground case)

[E. KOMENDANTSKAYA, G. MCCUSKER & J. POWER 2010]

$$p(\mathtt{r(b,b)}) = \{\{\mathtt{r(b,a)}, \mathtt{r(b,c)}\}, \{\mathtt{q(c)}\}\}$$

$$\begin{array}{rcl} \mathtt{r(b,c)} & \leftarrow & \mathtt{q(a)}, \mathtt{q(b)}, \mathtt{q(c)} \\ \mathtt{r(b,b)} & \leftarrow & \mathtt{r(b,a)}, \mathtt{r(b,c)} \\ \mathtt{r(b,b)} & \leftarrow & \mathtt{q(c)} \\ \mathtt{q(c)} & \leftarrow & \end{array}$$



Coalgebra in **Set**

$$p \quad : \quad At \rightarrow \mathcal{P}_f \mathcal{P}_f(At)$$

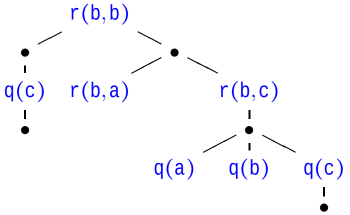
Coalgebraic Logic Programming (the ground case)

[E. KOMENDANTSKAYA, G. MCCUSKER & J. POWER 2010]

$$p(\mathtt{r(b,b)}) = \{\{\mathtt{r(b,a)}, \mathtt{r(b,c)}\}, \{\mathtt{q(c)}\}\}$$

$$\begin{array}{rcl} \mathtt{r(b,c)} & \leftarrow & \mathtt{q(a)}, \mathtt{q(b)}, \mathtt{q(c)} \\ \mathtt{r(b,b)} & \leftarrow & \mathtt{r(b,a)}, \mathtt{r(b,c)} \\ \mathtt{r(b,b)} & \leftarrow & \mathtt{q(c)} \\ \mathtt{q(c)} & \leftarrow & \end{array}$$

$$[\![\mathtt{r(b,b)}]\!]_p =$$



Coalgebra in **Set**

$$\begin{array}{rcl} p & : & At \to \mathcal{P}_f\mathcal{P}_f(At) \\ [\![-]\!]_p & : & At \to \mathcal{C}(\mathcal{P}_f\mathcal{P}_f)(At) \end{array}$$

Coalgebraic Logic Programming (the general case)

[E. KOMENDANTSKAYA & J. POWER, CALCO 2011]

$$
\begin{aligned}
\mathtt{List}(\mathtt{c}(\mathtt{x}_1,\mathtt{x}_2)) &\leftarrow \mathtt{Nat}(\mathtt{x}_1), \mathtt{List}(\mathtt{x}_2) \\
\mathtt{List}(\mathtt{nil}) &\leftarrow \\
\mathtt{Nat}(\mathtt{succ}(\mathtt{x}_1)) &\leftarrow \mathtt{Nat}(\mathtt{x}_1) \\
\mathtt{Nat}(\mathtt{zero}) &\leftarrow
\end{aligned}
$$

Coalgebraic Logic Programming (the general case)

[E. KOMENDANTSKAYA & J. POWER, CALCO 2011]

$$\begin{aligned}
\texttt{List}(\texttt{c}(\texttt{x}_1,\texttt{x}_2)) &\leftarrow \texttt{Nat}(\texttt{x}_1),\texttt{List}(\texttt{x}_2)\\
\texttt{List}(\texttt{nil}) &\leftarrow\\
\texttt{Nat}(\texttt{succ}(\texttt{x}_1)) &\leftarrow \texttt{Nat}(\texttt{x}_1)\\
\texttt{Nat}(\texttt{zero}) &\leftarrow
\end{aligned}$$

$\boxed{\mathbf{L}_\Sigma}$ free Lawvere Theory on Σ

$\boxed{\textbf{objects}}$ natural numbers $(n \approx < x_1, \ldots, x_n >)$.

$\boxed{\textbf{arrow } \theta : n \to m}$ a substitution $[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$, where $t_1, \ldots, t_n$ are Σ-terms on variables $x_1, \ldots, x_m$.

Coalgebraic Logic Programming (the general case)

[E. Komendantskaya & J. Power, CALCO 2011]

$\mathtt{List}(\mathtt{c}(\mathtt{x}_1,\mathtt{x}_2)) \leftarrow \mathtt{Nat}(\mathtt{x}_1), \mathtt{List}(\mathtt{x}_2)$

$\mathtt{List}(\mathtt{nil}) \leftarrow$

$\mathtt{Nat}(\mathtt{succ}(\mathtt{x}_1)) \leftarrow \mathtt{Nat}(\mathtt{x}_1)$

$\mathtt{Nat}(\mathtt{zero}) \leftarrow$

$\boxed{\mathbf{L}_\Sigma}$ free Lawvere Theory on Σ

objects natural numbers $(n \approx < x_1, \ldots, x_n >)$.

arrow $\theta : n \to m$ a substitution $[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$, where $t_1, \ldots, t_n$ are Σ-terms on variables $x_1, \ldots, x_m$.

$\boxed{At : \mathbf{L}_\Sigma \to \mathbf{Set}}$ space of $\mathbf{L}_\Sigma$-typed atoms

$\boxed{At(n)}$ the set of atoms on variables $x_1, \ldots, x_n$.

$\boxed{At(\theta : n \to m)}$ a function mapping A into $\theta(A)$.

Coalgebraic Logic Programming (the general case)

[E. KOMENDANTSKAYA & J. POWER, CALCO 2011]

$\text{List}(\text{c}(x_1,x_2)) \leftarrow \text{Nat}(x_1), \text{List}(x_2)$
$\text{List}(\text{nil}) \leftarrow$
$\text{Nat}(\text{succ}(x_1)) \leftarrow \text{Nat}(x_1)$
$\text{Nat}(\text{zero}) \leftarrow$

$\boxed{\mathbf{L}_\Sigma}$ free Lawvere Theory on Σ

objects natural numbers $(n \approx < x_1, \ldots, x_n >)$.

arrow $\theta{:}n \to m$ a substitution $[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$, where $t_1, \ldots, t_n$ are Σ-terms on variables $x_1, \ldots, x_m$.

Coalgebra in $\mathbf{Set}^{\mathbf{L}_\Sigma}$

$$\begin{array}{rcl} p & : & At \to \widetilde{\mathcal{P}_f}\widetilde{\mathcal{P}_f}(At) \\ [\![-]\!]_p & : & At \to \mathcal{C}(\widetilde{\mathcal{P}_f}\widetilde{\mathcal{P}_f})(At) \end{array}$$

$\boxed{At{:}\mathbf{L}_\Sigma \to \mathbf{Set}}$ space of $\mathbf{L}_\Sigma$-typed atoms

$\boxed{At(n)}$ the set of atoms on variables $x_1, \ldots, x_n$.

$\boxed{At(\theta{:}n \to m)}$ a function mapping A into $\theta(A)$.

Definition of p

[E. KOMENDANTSKAYA & J. POWER, CALCO 2011]

Term-Matching

$$\text{Atom } A \rightsquigarrow A = \tau(H) \overset{\exists\tau}{\leftsquigarrow} \text{Clause } \boxed{H \leftarrow B_1, \ldots, B_k}$$

$$\Downarrow$$

$$\{\tau(B_1), \ldots, \tau(B_k)\} \in p_n(A)$$

Definition of p

[E. KOMENDANTSKAYA & J. POWER, CALCO 2011]

Term-Matching

$$
\begin{tikzcd}
\text{Atom } A \arrow[r, squiggly] & A = \tau(H) \arrow[d, Rightarrow] & \text{Clause } \boxed{H \leftarrow B_1, \ldots, B_k} \arrow[l, squiggly, "\exists\tau"'] \\
 & \{\tau(B_1), \ldots, \tau(B_k)\} \in p_n(A) &
\end{tikzcd}
$$

The problem

$p : At \to \widetilde{\mathcal{P}_f}\widetilde{\mathcal{P}_f}(At)$ is **not** a natural transformation.

$$
\begin{tikzcd}
\texttt{List}(\texttt{x}_1) \arrow[r, mapsto, "p_1"] \arrow[d, mapsto, "At([x_1 \mapsto nil])"'] & \varnothing \arrow[d, mapsto, "\widetilde{\mathcal{P}_f}\widetilde{\mathcal{P}_f}(At)([x_1 \mapsto nil])"] \\
\texttt{List}(\texttt{nil}) \arrow[r, mapsto, "p_0"'] & {}
\end{tikzcd}
$$

Definition of p

[E. Komendantskaya & J. Power, CALCO 2011]

Term-Matching

$$\begin{tikzcd}
\text{Atom } A \arrow[r, squiggly] & A = \tau(H) \arrow[d, Rightarrow] & \text{Clause } \boxed{H \leftarrow B_1, \ldots, B_k} \arrow[l, squiggly, "\exists\tau"'] \\
 & \{\tau(B_1), \ldots, \tau(B_k)\} \in p_n(A) &
\end{tikzcd}$$

The problem

$p{:}At \to \widetilde{\mathcal{P}_f}\widetilde{\mathcal{P}_f}(At)$ is **not** a natural transformation.

$$\begin{tikzcd}
\texttt{List}(\texttt{x}_1) \arrow[r, mapsto, "p_1"] \arrow[d, mapsto, "At([x_1 \mapsto nil])"'] & \varnothing \arrow[d, mapsto, "\widetilde{\mathcal{P}_f}\widetilde{\mathcal{P}_f}(At)([x_1 \mapsto nil])"] \\
\texttt{List}(\texttt{nil}) \arrow[r, mapsto, "p_0"'] & \{\varnothing\}
\end{tikzcd}$$

Definition of p

[E. KOMENDANTSKAYA & J. POWER, CALCO 2011]

Term-Matching

$$\begin{tikzcd} \text{Atom } A \arrow[r, squiggly] & A = \tau(H) \arrow[d, Rightarrow] & \text{Clause } \boxed{H \leftarrow B_1, \ldots, B_k} \arrow[l, squiggly, "\exists\tau"'] \\ & \{\tau(B_1), \ldots, \tau(B_k)\} \in p_n(A) & \end{tikzcd}$$

The problem

$p : At \to \widetilde{\mathcal{P}_f}\widetilde{\mathcal{P}_f}(At)$ is **not** a natural transformation.

$$\begin{tikzcd} \texttt{List}(\texttt{x}_1) \arrow[r, mapsto, "p_1"] \arrow[d, mapsto, "At([x_1 \mapsto nil])"'] & \emptyset \arrow[d, mapsto, "\widetilde{\mathcal{P}_f}\widetilde{\mathcal{P}_f}(At)([x_1 \mapsto nil])"] \\ \texttt{List}(\texttt{nil}) \arrow[r, mapsto, "p_0"'] & \emptyset \end{tikzcd}$$

Definition of p

[E. KOMENDANTSKAYA & J. POWER, CALCO 2011]

Term-Matching

$$\text{Atom } A \rightsquigarrow A = \tau(H) \overset{\exists\tau}{\leftsquigarrow} \text{Clause } \boxed{H \leftarrow B_1, \ldots, B_k}$$

$$\Downarrow$$

$$\{\tau(B_1), \ldots, \tau(B_k)\} \in p_n(A)$$

The problem

$p: At \to \widetilde{\mathcal{P}_f}\widetilde{\mathcal{P}_f}(At)$ is **not** a natural transformation.

$$\begin{tikzcd}
\texttt{List}(\texttt{x}_1) \arrow[r, mapsto, "p_1"] \arrow[d, mapsto, "At([x_1 \mapsto nil])"'] & \varnothing \arrow[d, mapsto, "\widetilde{\mathcal{P}_f}\widetilde{\mathcal{P}_f}(At)([x_1 \mapsto nil])"] \\
\texttt{List}(\texttt{nil}) \arrow[r, mapsto, "p_0"'] &
\end{tikzcd}$$

The final semantics is **not compositional**:

$$[[\theta(A)]]_p \neq \overline{\theta}([[A]]_p).$$

Definition of p

[E. KOMENDANTSKAYA & J. POWER, CALCO 2011]

Term-Matching

$$\begin{tikzcd}
\text{Atom } A \arrow[r, squiggly] & A = \tau(H) \arrow[d, Rightarrow] & \text{Clause } \boxed{H \leftarrow B_1, \ldots, B_k} \arrow[l, squiggly, "\exists\tau"'] \\
& \{\tau(B_1), \ldots, \tau(B_k)\} \in p_n(A) &
\end{tikzcd}$$

Cfr.

$$\begin{tikzcd}
t = \overline{a} < x > | b(y) \arrow[r, mapsto] \arrow[d, mapsto, "{[b \mapsto a]}"'] \arrow[dr, phantom, "\neq"] & \alpha(t) \arrow[d, mapsto, "{\alpha([b \mapsto a])}"] \\
t' = \overline{a} < x > | a(y) \arrow[r, mapsto] & \alpha(t')
\end{tikzcd}$$

The problem

$p : At \to \widetilde{\mathcal{P}_f}\widetilde{\mathcal{P}_f}(At)$ is **not** a natural transforma

$$\begin{tikzcd}
\texttt{List}(\texttt{x}_1) \arrow[r, mapsto, "p_1"] \arrow[d, mapsto, "{At([x_1 \mapsto nil])}"'] & \varnothing \arrow[d, mapsto, "{\widetilde{\mathcal{P}_f}\widetilde{\mathcal{P}_f}(At)([x_1 \mapsto nil])}"] \\
\texttt{List}(\texttt{nil}) \arrow[r, mapsto, "p_0"'] & {}
\end{tikzcd}$$

The final semantics is **not compositional**:

$$[[\theta(A)]]_p \neq \overline{\theta}([[A]]_p).$$

The Saturated Approach

Saturated Semantics

$$p \quad \mapsto \quad p^\sharp$$

The Saturated Approach

Saturated Semantics

$$p \quad \mapsto \quad p^\sharp$$

$$\begin{tikzcd} At \in \mathbf{Set}^{\mathbf{L}_\Sigma} \arrow[rr, bend left, "\mathcal{U}"] & & \mathbf{Set}^{|\mathbf{L}_\Sigma|} \end{tikzcd}$$

The Saturated Approach

Saturated Semantics

$$p \quad \mapsto \quad p^\sharp$$

$$\begin{tikzcd} At \in \mathbf{Set}^{\mathbf{L}_\Sigma} \arrow[rr, bend left, "\mathcal{U}"] & & \mathbf{Set}^{|\mathbf{L}_\Sigma|} \arrow[loop right, "\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f"] \end{tikzcd}$$

$$\begin{tikzcd} & \mathcal{U}(At) \arrow[d, "p"] \\ \boxed{\mathbf{Set}^{|\mathbf{L}_\Sigma|}} & \hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At) \end{tikzcd}$$

The Saturated Approach

Saturated Semantics

$$p \quad \mapsto \quad p^{\sharp}$$

$$\begin{tikzcd} At \in \mathbf{Set}^{\mathbf{L}_\Sigma} \arrow[rr, bend left, "\mathcal{U}"] \arrow[rr, phantom, "\perp"] & & \mathbf{Set}^{|\mathbf{L}_\Sigma|} \arrow[ll, bend left, "\mathcal{K}"] \arrow[loop right, "\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f"] \end{tikzcd}$$

$$\boxed{\mathbf{Set}^{|\mathbf{L}_\Sigma|}} \qquad \begin{tikzcd} \mathcal{U}(At) \arrow[d, "p"] \\ \hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At) \end{tikzcd}$$

The Saturated Approach

Saturated Semantics

$$p \quad \mapsto \quad p^\sharp$$

$$\begin{tikzcd} At \in \mathbf{Set}^{\mathbf{L}_\Sigma} \arrow[rr, bend left, "\mathcal{U}"] \arrow[rr, phantom, "\perp"] & & \mathbf{Set}^{|\mathbf{L}_\Sigma|} \arrow[ll, bend left, "\mathcal{K}"] \arrow[loop right, "\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f"] \end{tikzcd}$$

$$\begin{tikzcd} At \arrow[r, "\eta_{At}"] \arrow[dr, "p^\sharp"'] & \mathcal{K}\mathcal{U}(At) \arrow[d, "K(p)"] \\ & \mathcal{K}\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At) \end{tikzcd}$$

$\boxed{\mathbf{Set}^{\mathbf{L}_\Sigma}}$ | $\boxed{\mathbf{Set}^{|\mathbf{L}_\Sigma|}}$

$$\begin{tikzcd} \mathcal{U}(At) \arrow[d, "p"] \\ \hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At) \end{tikzcd}$$

The Saturated Approach

Saturated Semantics

$$p \quad \mapsto \quad p^\sharp$$

Coalgebra in $\mathbf{Set}^{\mathbf{L}_\Sigma}$

$$\begin{aligned} p^\sharp &: At \to \mathcal{K}\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At) \\ [[-]]_{p^\sharp} &: At \to \mathcal{C}(\mathcal{K}\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U})(At) \end{aligned}$$

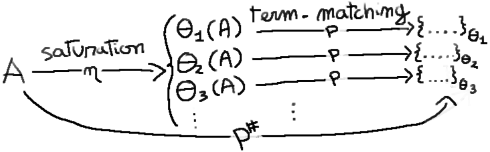
$$\begin{tikzcd} At \arrow[r, "\eta_{At}"] \arrow[dr, "p^\sharp"'] & \mathcal{K}\mathcal{U}(At) \arrow[d, "K(p)"] \\ & \mathcal{K}\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At) \end{tikzcd}$$

$\boxed{\mathbf{Set}^{\mathbf{L}_\Sigma}}$ | $\boxed{\mathbf{Set}^{|\mathbf{L}_\Sigma|}}$

$$\begin{tikzcd} \mathcal{U}(At) \arrow[d, "p"] \\ \hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At) \end{tikzcd}$$

The Saturated Approach

Saturated Semantics



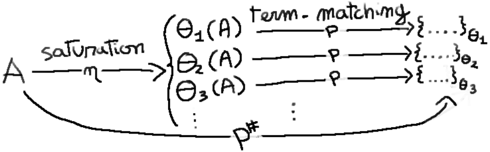
Coalgebra in $\mathbf{Set}^{\mathbf{L}_\Sigma}$

$$
\begin{aligned}
p^\sharp &: At \to \mathcal{K}\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At)\\
[[-]]_{p^\sharp} &: At \to \mathcal{C}(\mathcal{K}\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U})(At)
\end{aligned}
$$

$$
\begin{tikzcd}
At \arrow[r, "\eta_{At}"] \arrow[dr, "p^\sharp"'] & \mathcal{K}\mathcal{U}(At) \arrow[d, "K(p)"] \\
& \mathcal{K}\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At)
\end{tikzcd}
\qquad
\boxed{\mathbf{Set}^{\mathbf{L}_\Sigma}} \;\Big|\; \boxed{\mathbf{Set}^{|\mathbf{L}_\Sigma|}}
\qquad
\begin{tikzcd}
\mathcal{U}(At) \arrow[d, "p"] \\
\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At)
\end{tikzcd}
$$

The Saturated Approach

Saturated Semantics



p ~ term-matching
$p^\sharp$ ~ unification

Coalgebra in $\mathbf{Set}^{\mathbf{L}_\Sigma}$

$$
\begin{aligned}
p^\sharp \quad &: \quad At \to \mathcal{K}\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At) \\
[[-]]_{p^\sharp} \quad &: \quad At \to \mathcal{C}(\mathcal{K}\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U})(At)
\end{aligned}
$$

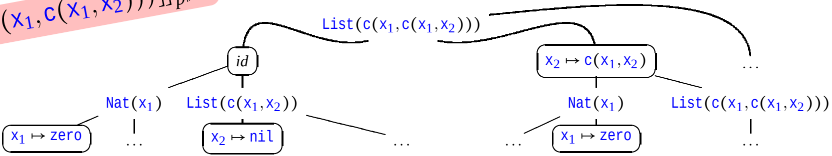
$$
\begin{tikzcd}
At \arrow[r, "\eta_{At}"] \arrow[dr, "p^\sharp"'] & \mathcal{K}\mathcal{U}(At) \arrow[d, "K(p)"] \\
& \mathcal{K}\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At)
\end{tikzcd}
$$

$\boxed{\mathbf{Set}^{\mathbf{L}_\Sigma}}$ | $\boxed{\mathbf{Set}^{|\mathbf{L}_\Sigma|}}$

$$
\begin{tikzcd}
\mathcal{U}(At) \arrow[d, "p"] \\
\hat{\mathcal{P}}_f\hat{\mathcal{P}}_f\mathcal{U}(At)
\end{tikzcd}
$$

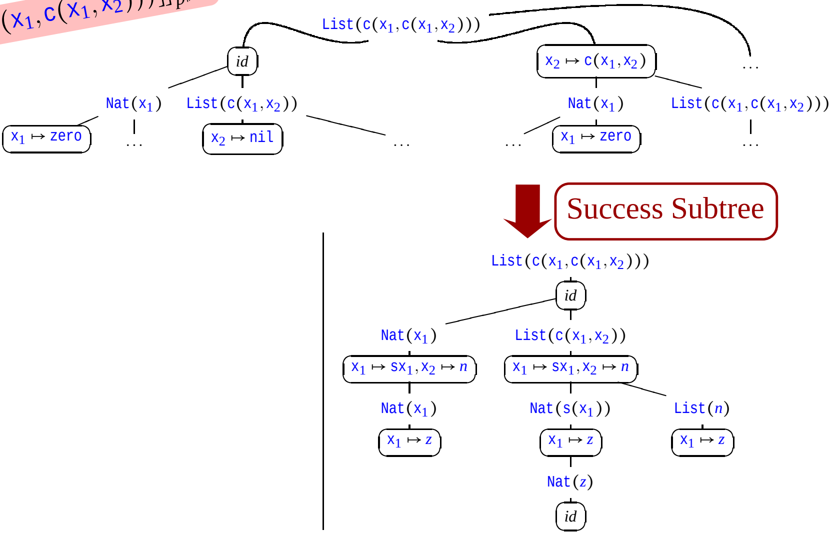
Saturated Derivation Trees

$[\![\mathtt{List}(\mathtt{c}(\mathtt{x}_1, \mathtt{c}(\mathtt{x}_1, \mathtt{x}_2)))]\!]_{p^\sharp} =$

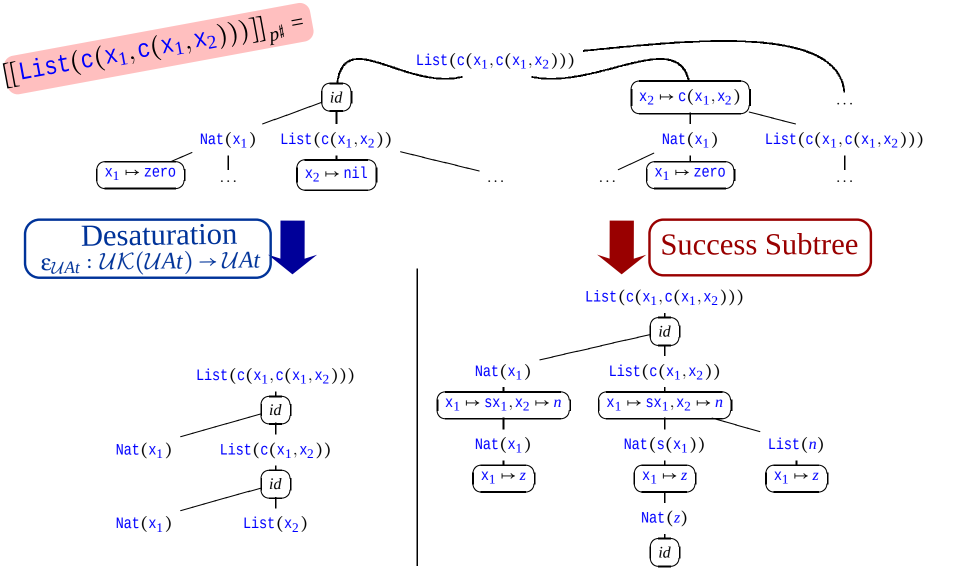


Saturated Derivation Trees

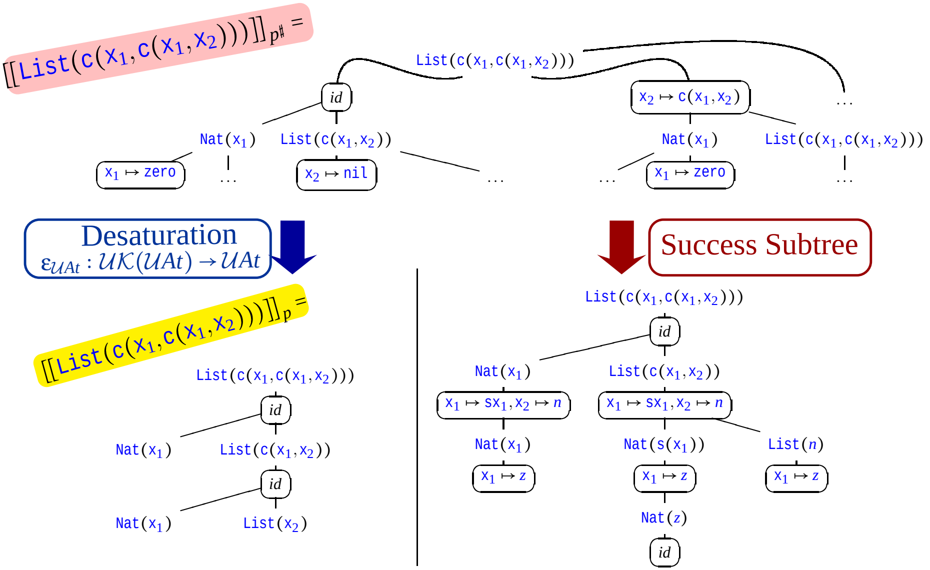
$[\![\texttt{List}(\texttt{c}(\texttt{x}_1,\texttt{c}(\texttt{x}_1,\texttt{x}_2)))]\!]_{p^\sharp} =$



Saturated Derivation Trees



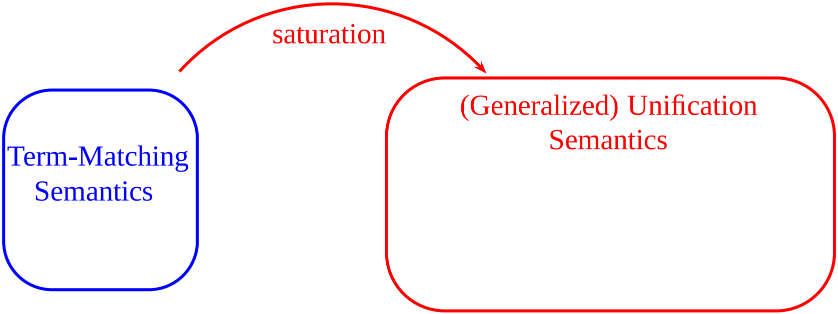
Saturated Derivation Trees



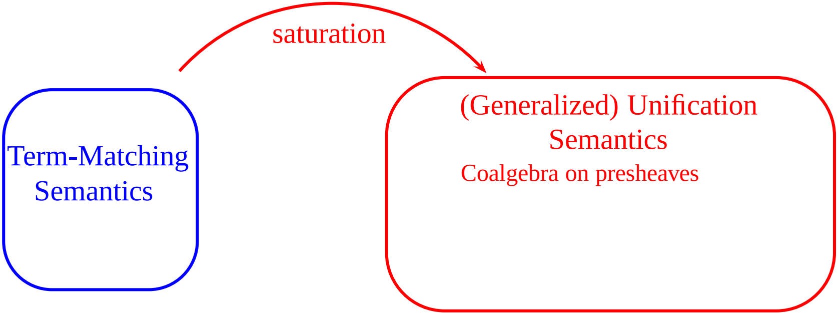
Summary

Term-Matching
Semantics

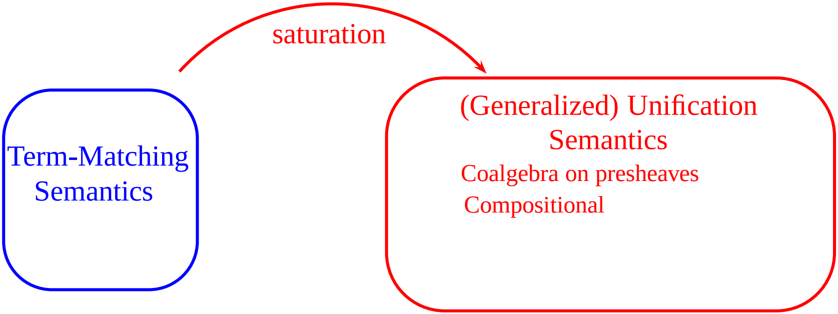
Summary


saturation
Term-Matching Semantics
(Generalized) Unification Semantics

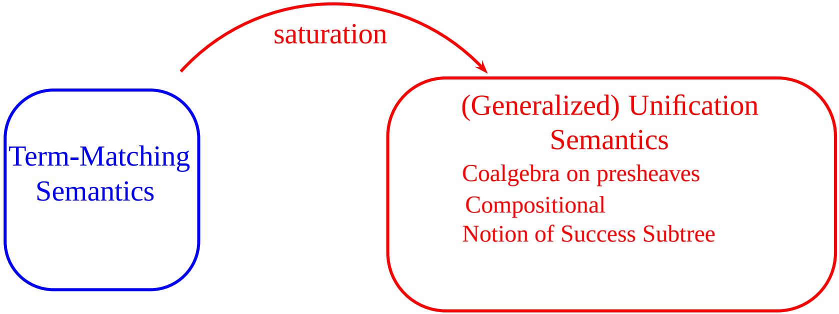
Summary


saturation
Term-Matching Semantics
(Generalized) Unification Semantics
Coalgebra on presheaves

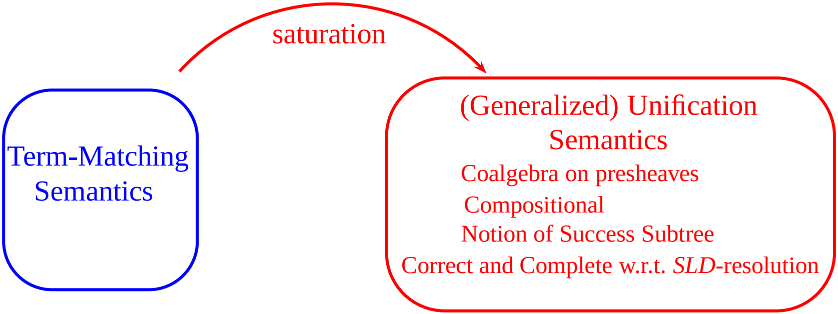
Summary


saturation
Term-Matching Semantics
(Generalized) Unification Semantics
Coalgebra on presheaves
Compositional

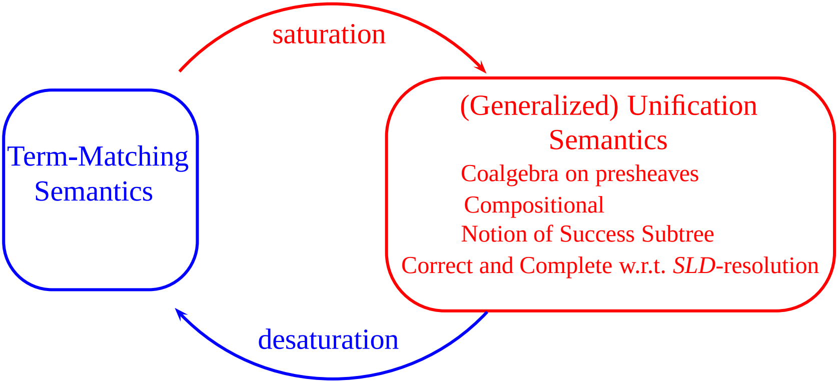
Summary


saturation
Term-Matching Semantics
(Generalized) Unification Semantics
Coalgebra on presheaves
Compositional
Notion of Success Subtree

Summary


Term-Matching Semantics
saturation
(Generalized) Unification Semantics
Coalgebra on presheaves
Compositional
Notion of Success Subtree
Correct and Complete w.r.t. *SLD*-resolution

Summary


saturation
(Generalized) Unification
Semantics
Coalgebra on presheaves
Compositional
Notion of Success Subtree
Correct and Complete w.r.t. *SLD*-resolution
Term-Matching
Semantics
desaturation