Unnesting of Copatterns

RTA-TLCA, 15 July 2014

Anton Setzer	Andreas Abel	Brigitte Pientka	David Thibodeau
Swansea	Gothenburg	Montreal	Montreal
UK	Sweden	Canada	Canada

Pattern and Copattern Matching

Unnesting of Copatterns/Patterns

Proof of Conservatity and Preservation of SN/WN

Natural Numbers

Syntax for Algebraic Data Types in Paper

$$\mathsf{Nat} := \mu X.\langle \mathsf{zero}\ \mathbf{1} \mid \mathsf{suc}\ X\rangle$$

Introduction Rule

(All constructors will have exactly one argument)

$$\begin{array}{lcccc} \mathsf{zero} & : & \mathbf{1} & \rightarrow & \mathsf{Nat} \\ \mathsf{suc} & : & \mathsf{Nat} & \rightarrow & \mathsf{Nat} \end{array}$$

Elimination Rule (Pattern Matching)

$$\begin{array}{lcl} \multicolumn{3}{l}{\mathsf{pred} : \mathsf{Nat} \rightarrow \mathsf{Nat}} \\ \mathsf{pred}\ (\mathsf{zero}\ x) & = & ? \\ \mathsf{pred}\ (\mathsf{suc}\ n) & = & ? \end{array}$$

Full recursion allowed (normalisation for individual terms see later)

Nested Patterns and CC-Pattern-Sets

$$\begin{array}{lcl} \text{pred} : \text{Nat} \to \text{Nat} & & \\ \text{pred}\ (\text{zero}\ x) & = & ? \\ \text{pred}\ (\text{suc}\ n) & = & ? \end{array}$$

Pattern matching on $\mathbf{1}$ (containing ()) yields the **nested pattern**

$$\begin{array}{lcl} \text{pred} : \text{Nat} \to \text{Nat} & & \\ \text{pred}\ (\text{zero}\ ()) & = & ? \\ \text{pred}\ (\text{suc}\ \ n) & = & ? \end{array}$$

which formally is the coverage complete (**cc**) **pattern set**

$$\text{pred} : \text{Nat} \to \text{Nat} \ \triangleleft \left| \begin{array}{lllll} (\cdot & \vdash & \text{pred}\ (\text{zero}\ ()) & : & \text{Nat}) \\ (n : \text{Nat} & \vdash & \text{pred}\ (\text{suc}\ n) & : & \text{Nat}) \end{array} \right.$$

Coverage Complete Rule Sets (CC-Rule Sets)

After full pattern derived we fill in the "?"

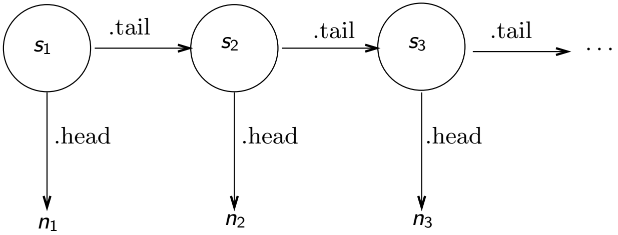
$$\begin{array}{lcl} \mathrm{pred} : \mathrm{Nat} \to \mathrm{Nat} & & \\ \mathrm{pred}\ (\mathrm{zero}\ ()) & = & \mathrm{zero}\ () \\ \mathrm{pred}\ (\mathrm{suc}\ \ n) & = & n \end{array}$$

corresponds to the coverage complete (**cc**) **rule set**

$$\mathrm{pred} : \mathrm{Nat} \to \mathrm{Nat} \ \lhd \left| \begin{array}{llllll} (\cdot & \vdash & \mathrm{pred}\ (\mathrm{zero}\ ()) & \longrightarrow & \mathrm{zero}\ () & : & \mathrm{Nat}) \\ (n : \mathrm{Nat} & \vdash & \mathrm{pred}\ (\mathrm{suc}\ n) & \longrightarrow & n & : & \mathrm{Nat}) \end{array} \right.$$

Stream

$$\text{Stream} := \nu X.\{\text{head} : \text{Nat}, \text{tail} : X\} \qquad \text{(In paper called StrN)}$$



Stream

Elimination Rule

$$\text{If } s : \text{Stream then} \quad \begin{array}{lcl} s\ .\text{head} & : & \text{Nat} \\ s\ .\text{tail} & : & \text{Stream} \end{array}$$

.head, .tail treated like application.

Introduction Rule (Copattern Matching)

$$\begin{array}{lcl} \multicolumn{3}{l}{\text{inc} : \text{Nat} \to \text{Stream}} \\ \text{inc } n\ .\text{head} & = & n \\ \text{inc } n\ .\text{tail} & = & \text{inc } (n+1) \end{array}$$

$$\text{Informally} \quad \text{inc } n = n, n+1, n+2, \ldots$$

CC-Rule/Pattern Set

$$\begin{array}{l} \mathrm{inc} : \mathrm{Nat} \to \mathrm{Stream} \\ \mathrm{inc}\ n\ .\mathrm{head} \quad = \quad n \\ \mathrm{inc}\ n\ .\mathrm{tail} \quad = \quad \mathrm{inc}\ (n+1) \end{array}$$

This corresponds to the cc-pattern-set

$$\mathrm{inc} : \mathrm{Nat} \to \mathrm{Stream} \ \lhd \left| \begin{array}{lclcl} (n : \mathrm{Nat} & \vdash & \mathrm{inc}\ n\ .\mathrm{head} & : & \mathrm{Nat}) \\ (n : \mathrm{Nat} & \vdash & \mathrm{inc}\ n\ .\mathrm{tail} & : & \mathrm{Stream}) \end{array} \right.$$

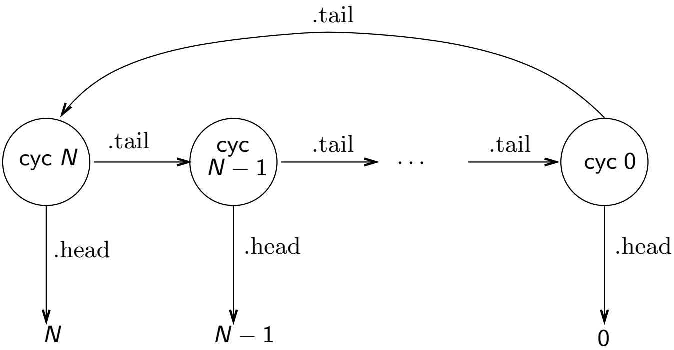
and cc-rule-set

$$\mathrm{inc} : \mathrm{Nat} \to \mathrm{Stream} \ \lhd \left| \begin{array}{lclcll} (n : \mathrm{Nat} & \vdash & \mathrm{inc}\ n\ .\mathrm{head} & \longrightarrow & n & : \mathrm{Nat}) \\ (n : \mathrm{Nat} & \vdash & \mathrm{inc}\ n\ .\mathrm{tail} & \longrightarrow & \mathrm{inc}\ (n+1) & : \mathrm{Stream}) \end{array} \right.$$

cyc n

Let N be **fixed**.
For n we define a stream which is informally given as

$$\text{cyc } n = n, n-1, n-2, \ldots, 0, N, N-1, N-2, \ldots, 0, N, N-1, \ldots$$



Development of cyc

(Paper contains **rules for deriving cc-pattern/rule-sets**.)

The **simplest pattern matching** is by itself:

$$\mathsf{cyc} : \mathsf{Nat} \to \mathsf{Stream} \ \triangleleft| \ \ (\cdot \ \vdash \ \mathsf{cyc} \ : \ \mathsf{Nat} \to \mathsf{Stream})$$

Copattern matching for functions is **application**:

$$\mathsf{cyc} : \mathsf{Nat} \to \mathsf{Stream} \ \triangleleft| \ \ (n : \mathsf{Nat} \ \vdash \ \mathsf{cyc}\ n \ : \ \mathsf{Stream})$$

Copattern matching on Stream yields:

$$\mathsf{cyc} : \mathsf{Nat} \to \mathsf{Stream} \ \triangleleft| \ \begin{array}{llll} (n : \mathsf{Nat} & \vdash & \mathsf{cyc}\ n\ .\mathsf{head} & : \ \mathsf{Nat}) \\ (n : \mathsf{Nat} & \vdash & \mathsf{cyc}\ n\ .\mathsf{tail} & : \ \mathsf{Stream}) \end{array}$$

Development of cyc (Cont.)

Pattern matching on Nat yields:

$$\mathrm{cyc} : \mathrm{Nat} \to \mathrm{Stream} \lhd \left| \begin{array}{llllll} (n : \mathrm{Nat} & \vdash & \mathrm{cyc}\ n & .\mathrm{head} & : & \mathrm{Nat}) \\ (x : \mathbf{1} & \vdash & \mathrm{cyc}\ (\mathrm{zero}\ x) & .\mathrm{tail} & : & \mathrm{Stream}) \\ (n : \mathrm{Nat} & \vdash & \mathrm{cyc}\ (\mathrm{suc}\ n) & .\mathrm{tail} & : & \mathrm{Stream}) \end{array} \right.$$

Pattern matching on $\mathbf{1}$ yields:

$$\mathrm{cyc} : \mathrm{Nat} \to \mathrm{Stream} \lhd \left| \begin{array}{llllll} (n : \mathrm{Nat} & \vdash & \mathrm{cyc}\ n & .\mathrm{head} & : & \mathrm{Nat}) \\ (\ \cdot & \vdash & \mathrm{cyc}\ (\mathrm{zero}\ ()) & .\mathrm{tail} & : & \mathrm{Stream}) \\ (n : \mathrm{Nat} & \vdash & \mathrm{cyc}\ (\mathrm{suc}\ n) & .\mathrm{tail} & : & \mathrm{Stream}) \end{array} \right.$$

By adding results we obtain a **cc-rule-set:**

$$\mathrm{cyc} : \mathrm{Nat} \to \mathrm{Stream} \lhd \left| \begin{array}{lllllll} (n : \mathrm{Nat} & \vdash & \mathrm{cyc}\ n & .\mathrm{head} & \longrightarrow & n & : \mathrm{Nat}) \\ (\ \cdot & \vdash & \mathrm{cyc}\ (\mathrm{zero}\ ()) & .\mathrm{tail} & \longrightarrow & \mathrm{cyc}\ N & : \mathrm{Stream}) \\ (n : \mathrm{Nat} & \vdash & \mathrm{cyc}\ (\mathrm{suc}\ n) & .\mathrm{tail} & \longrightarrow & \mathrm{cyc}\ n & : \mathrm{Stream}) \end{array} \right.$$

Unnesting of cyc

$$\text{cyc} : \text{Nat} \to \text{Stream} \lhd| \begin{array}{llllll} (n : \text{Nat} & \vdash & \text{cyc}\ n & .\text{head} & \longrightarrow n & : \text{Nat}) \\ (\ \cdot & \vdash & \text{cyc}\ (\text{zero}\ ()) & .\text{tail} & \longrightarrow \text{cyc}\ N & : \text{Stream}) \\ (n : \text{Nat} & \vdash & \text{cyc}\ (\text{suc}\ n) & .\text{tail} & \longrightarrow \text{cyc}\ n & : \text{Stream}) \end{array}$$

We unnest the last step (pattern matching on $\mathbf{1}$) and delegate it to a new function g_2

$$\text{cyc} : \text{Nat} \to \text{Stream} \lhd| \begin{array}{llllll} (n : \text{Nat} & \vdash & \text{cyc}\ n & .\text{head} & \longrightarrow n & : \text{Nat}) \\ (x : \mathbf{1} & \vdash & \text{cyc}\ (\text{zero}\ x) & .\text{tail} & \longrightarrow g_2\ x & : \text{Stream}) \\ (n : \text{Nat} & \vdash & \text{cyc}\ (\text{suc}\ n) & .\text{tail} & \longrightarrow \text{cyc}\ n & : \text{Stream}) \end{array}$$

$$g_2 : \mathbf{1} \to \text{Stream} \quad \lhd| \quad (\ \cdot \ \vdash \ g_2\ () \ \longrightarrow \ \text{cyc}\ N \ : \ \text{Stream})$$

Unnesting of cyc (Cont)

$$\mathrm{cyc} : \mathrm{Nat} \to \mathrm{Stream} \lhd| \begin{array}{llllll} (n : \mathrm{Nat} & \vdash & \mathrm{cyc}\ n & .\mathrm{head} \longrightarrow n & : \mathrm{Nat}) \\ (x : \mathbf{1} & \vdash & \mathrm{cyc}\ (\mathrm{zero}\ x) & .\mathrm{tail} \longrightarrow g_2\ x & : \mathrm{Stream}) \\ (n : \mathrm{Nat} & \vdash & \mathrm{cyc}\ (\mathrm{suc}\ n) & .\mathrm{tail} \longrightarrow \mathrm{cyc}\ n & : \mathrm{Stream}) \end{array}$$

$$g_2 : \mathbf{1} \to \mathrm{Stream} \lhd| \ (\cdot \vdash g_2\ () \longrightarrow \mathrm{cyc}\ N : \mathrm{Stream})$$

Now we unnest pattern matching on $x : \mathrm{Nat}$ and delegate it to a new function g_1:

$$\mathrm{cyc} : \mathrm{Nat} \to \mathrm{Stream} \lhd| \begin{array}{lllll} (n : \mathrm{Nat} & \vdash & \mathrm{cyc}\ n\ .\mathrm{head} & \longrightarrow n & : \mathrm{Nat}) \\ (n : \mathrm{Nat} & \vdash & \mathrm{cyc}\ n\ .\mathrm{tail} & \longrightarrow g_1\ n & : \mathrm{Stream}) \end{array}$$

$$g_1 : \mathrm{Nat} \to \mathrm{Stream} \lhd| \begin{array}{lllll} (x : \mathbf{1} & \vdash & g_1\ (\mathrm{zero}\ x) & \longrightarrow g_2\ x & : \mathrm{Stream}) \\ (n : \mathrm{Nat} & \vdash & g_1\ (\mathrm{suc}\ n) & \longrightarrow \mathrm{cyc}\ n & : \mathrm{Stream}) \end{array}$$

$$g_2 : \mathbf{1} \to \mathrm{Stream} \lhd| \ (\cdot \vdash g_2\ () \longrightarrow \mathrm{cyc}\ N : \mathrm{Stream})$$

Simple Pattern

- End result is simple pattern:
  - There is at most one proper pattern/copattern step which is the last one.

Programs

Definition

(a) A **program** $\mathcal{P}$ is given by constants with their types and a cc-rule-set for each constant (referring to terms in the same language).

(b) A program $\mathcal{P}'$ **extends** $\mathcal{P}$ if the it contains all the constants of $\mathcal{P}$ with the same types
(but not necessarily the same cc-rule-sets).

Conservative Extensions, Preservation of SN, WN

Definition

Let $\mathcal{P}'$ be a program extending $\mathcal{P}$.

(a) $\mathcal{P}'$ is a **conservative extension** of $\mathcal{P}$ iff

$$\forall t, t' \in \mathrm{Term}_{\mathcal{P}}. t \longrightarrow^*_{\mathcal{P}} t' \Leftrightarrow t \longrightarrow^*_{\mathcal{P}'} t'$$

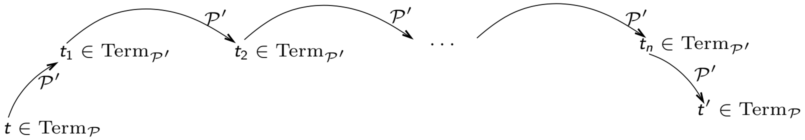
(b) $\mathcal{P}'$ **preserves** strong normalisation (**SN**) iff

$$\forall t \in \mathrm{Term}_{\mathcal{P}}. t \in \mathsf{SN}(\mathcal{P}) \Leftrightarrow t \in \mathsf{SN}(\mathcal{P}')$$

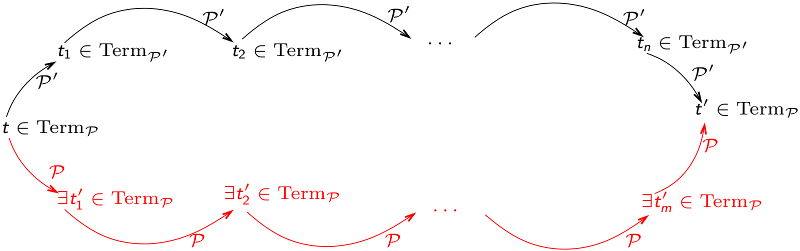
(c) $\mathcal{P}'$ **preserves** weak normalisation (**WN**) iff

$$\forall t \in \mathrm{Term}_{\mathcal{P}}. t \in \mathsf{WN}(\mathcal{P}) \Leftrightarrow t \in \mathsf{WN}(\mathcal{P}')$$

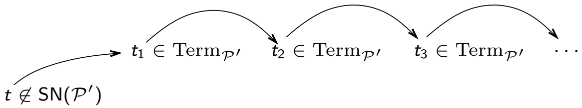
Conservative Extension



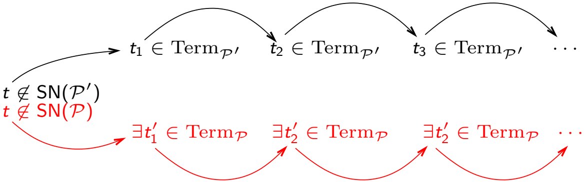
Conservative Extension



Preservation of ¬SN



Preservation of ¬SN



Main Theorem

Theorem

Let $\mathcal{P}$ be a program.
There exist an extension of $\mathcal{P}$ which is

- *conservative,*
- *preserves* SN*,*
- *preserves* WN
- *and has only simple patterns.*

Proof of Theorem General Methodology

- $\mathcal{P}'$ is obtained from $\mathcal{P}$ by replacing last step of an nested pattern until all patterns are simple.
- Show
  - reduction of one non-nested reduction step
    yields a conservative extension preserving SN/WN.
- Illustration by considering the last step in the example above.

Last Step in Example

Program $\mathcal{P}$ (g_2 omitted since unchanged):

$$\text{cyc} : \text{Nat} \to \text{Stream} \lhd| \begin{array}{llllll} (n : \text{Nat} & \vdash & \text{cyc}\ n & .\text{head} & \longrightarrow n & : \text{Nat}) \\ (x : \mathbf{1} & \vdash & \text{cyc}\ (\text{zero}\ x) & .\text{tail} & \longrightarrow g_2\ x & : \text{Stream}) \\ (n : \text{Nat} & \vdash & \text{cyc}\ (\text{suc}\ n) & .\text{tail} & \longrightarrow \text{cyc}\ n & : \text{Stream}) \end{array}$$

Program $\mathcal{P}'$:

$$\text{cyc} : \text{Nat} \to \text{Stream} \lhd| \begin{array}{llllll} (n : \text{Nat} & \vdash & \text{cyc}\ n\ .\text{head} & \longrightarrow & n & : \text{Nat}) \\ (n : \text{Nat} & \vdash & \text{cyc}\ n\ .\text{tail} & \longrightarrow & g_1\ n & : \text{Stream}) \end{array}$$

$$g_1 : \text{Nat} \to \text{Stream} \lhd| \begin{array}{llllll} (x : \mathbf{1} & \vdash & g_1\ (\text{zero}\ x) & \longrightarrow & g_2\ x & : \text{Stream}) \\ (n : \text{Nat} & \vdash & g_1\ (\text{suc}\ n) & \longrightarrow & \text{cyc}\ n & : \text{Stream}) \end{array}$$

Proof of Theorem - First easy steps

- Easy to see for $t, t' \in \mathrm{Term}_{\mathcal{P}}$

$$t \longrightarrow_{\mathcal{P}} t' \;\Rightarrow\; t \longrightarrow^{\geq 1}_{\mathcal{P}'} t'$$

  In example

$$\begin{array}{llllll} \mathsf{cyc}\ (\mathsf{zero}\ x) & .\mathsf{head} & \longrightarrow_{\mathcal{P}'} & g_1\ (\mathsf{zero}\ x) & \longrightarrow_{\mathcal{P}'} & g_2\ x \\ \mathsf{cyc}\ (\mathsf{suc}\ n) & .\mathsf{head} & \longrightarrow_{\mathcal{P}'} & g_1\ (\mathsf{suc}\ n) & \longrightarrow_{\mathcal{P}'} & \mathsf{cyc}\ n \end{array}$$

- Implies for $t, t' \in \mathrm{Term}_{\mathcal{P}}$

$$t \longrightarrow^{*}_{\mathcal{P}} t' \;\Rightarrow\; t \longrightarrow^{*}_{\mathcal{P}'} t'$$

  and

$$t \notin \mathsf{SN}(\mathcal{P}) \Rightarrow t \notin \mathsf{SN}(\mathcal{P}')$$

Back Translation and Conservativity

- Let (in above example)

$$\text{Good} = \{t \in \mathrm{Term}_{\mathcal{P}'} \mid g_1 \text{ always applied at least once } \}$$

- Let the back translation be

$$\begin{array}{l} \text{int} : \text{Good} \to \mathrm{Term}_{\mathcal{P}} \\ \text{int}(t) = \quad \text{result of replacing in } t \text{ subterms} \quad (g_1\ s) \ \text{ by } \ (\text{cyc } s\ .\mathrm{tail}) \end{array}$$

- We have

$$\begin{array}{l} \forall t \in \mathrm{Term}_{\mathcal{P}}.t \in \text{Good} \wedge \text{int}(t) = t \\ \text{Good closed under } \longrightarrow_{\mathcal{P}'} \\ \forall t, t' \in \mathrm{Term}_{\mathcal{P}'}.t \longrightarrow_{\mathcal{P}'} t' \ \Rightarrow \ \text{int}(t) \longrightarrow^*_{\mathcal{P}} \text{int}(t') \end{array}$$

- Therefore for $t, t' \in \mathrm{Term}_{\mathcal{P}}$

$$t \longrightarrow^*_{\mathcal{P}'} t' \Rightarrow t = \text{int}(t) \longrightarrow^*_{\mathcal{P}} \text{int}(t') = t'$$

Preservation of Normalisation

- We might have

$$t \longrightarrow_{\mathcal{P}'} t' \;\text{ but }\; \mathsf{int}(t) = \mathsf{int}(t')$$

- However, there are no infinitely long chains leaving $\mathsf{int}(t)$ unchanged.
- Therefore we obtain for $t \in \mathrm{Term}_{\mathcal{P}'}$

$$t \notin \mathsf{SN}(\mathcal{P}') \Rightarrow t \notin \mathsf{SN}(\mathcal{P})$$

- Preservation of WN (thanks to referee!):

$$\begin{array}{rl} t \in \mathsf{WN}(\mathcal{P}) \Rightarrow & t \longrightarrow^{*}_{\mathcal{P}} t' \in \mathsf{NF}(\mathcal{P}) \\ \Rightarrow & t \longrightarrow^{*}_{\mathcal{P}} t' \in \mathsf{SN}(\mathcal{P}) \\ \Rightarrow & t \longrightarrow^{*}_{\mathcal{P}'} t' \in \mathsf{SN}(\mathcal{P}') \\ \Rightarrow & t \in \mathsf{WN}(\mathcal{P}') \end{array}$$

Similarly in other direction (using back translation).

Conclusion

Algebras	Coalgebras
defined by introduction rules	defined by elimination rules
elimination rules given by pattern matching	introduction rules given by copattern matching

Conclusion

- Calculus for deriving nested coverage complete rule sets.
- Reduction of nested (co)patterns to simple (co)patterns.
- Proof of correctness.
  - Conservative extension,
  - preservation of SN,
  - preservation of WN.

Future Work

- Reduction to combinators (writing up phase).
- Having conservative extension, preservation of SN and of WN sounds ad hoc.
  - What is a general notion of properties to be preserved?
  - Probably all formulas expressible in a certain language to be defined.
- Use for compilaton of copatterns.
- Development of termination checker based on principles terminating programs are those reducible to primitive corecursion.