Compositional Coinduction in Agda

Andreas Abel

Department of Computer Science and Engineering
Chalmers and Gothenburg University

IFIP WG 1.3 Annual meeting
Ostseebad Binz
9-12 January 2017

Agda

- Implementation of intensional Martin-Löf Type Theory
- Ongoing at Chalmers since 1990s
- Agda 2 developed since 2005
- Dependently-typed functional programming language
- Curry-Howard: Propositions-as-types
- Interactive proof assistant

Copattern matching

- And infinite object is defined by its observations:
  - A function is defined by application.
  - A stream is defined by its projections head and tail.
- Extend pattern matching notation by projections.

$$\begin{array}{lcl} \mathsf{head}\,(\mathsf{mapStream}\,\mathsf{f}\,\mathsf{s}) & = & \ldots \\ \mathsf{tail}\,(\mathsf{mapStream}\,\mathsf{f}\,\mathsf{s}) & = & \ldots \end{array}$$

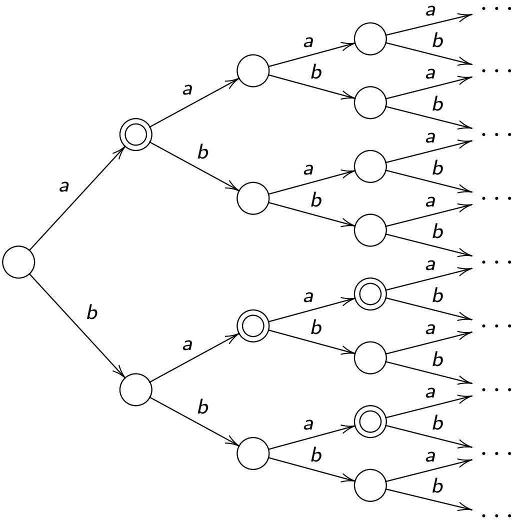
- Added to Agda (implementation started 2012)

Formal Language Example: Even binary numbers

- Even binary numbers, no leading zeros.
- Alphabet A with $0 = a$ and $1 = b$.
- $E_0 = \{a, ba, baa, bba, baaa, baba, \ldots\}$.
- Dictionary/trie/language:

$$\mathsf{Lang} \cong \mathsf{Bool} \times (A \to \mathsf{Lang})$$

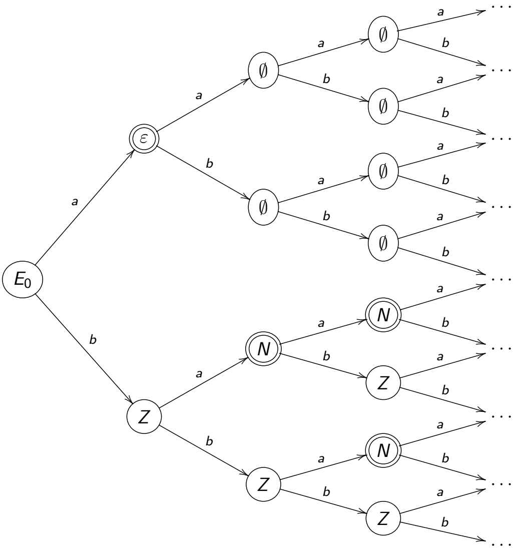
Trie of E_0



Regular Languages

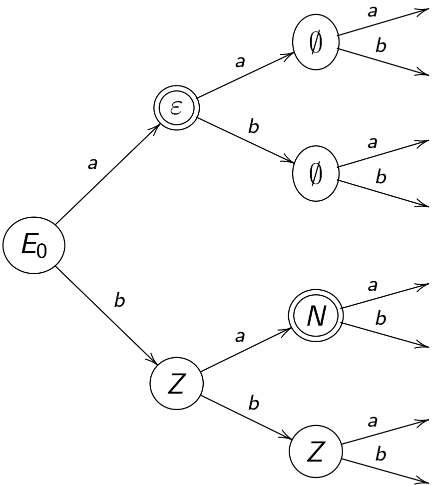
- A trie is regular if it has only finitely many different subtrees.
- Subtrees of E_0:

E_0	$=$	$a + b(a+b)^*a$	even
Z	$=$	$(a+b)^*a$	ending in a
N	$=$	$\varepsilon + (a+b)^*a$	not ending in b
ε			empty string
$\emptyset$			nothing (empty language)

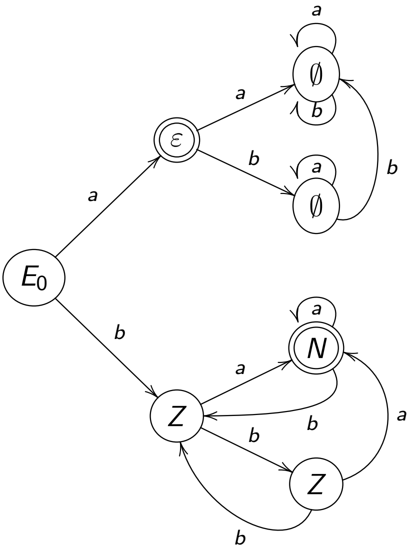


E_0
a
b
ε
Z
$\emptyset$
N
a
b
...

Cutting duplications at depth 3



Bending branches $\Longrightarrow$ finite automaton



Automata, Formally

- Automaton:
  1. state set S.
  2. acceptance function $\nu : S \to \mathsf{Bool}$
  3. transition function $\delta : S \to A \to S$.

s	νs	$\delta\, s\, a$	$\delta\, s\, b$
E_0	✗	ε	Z
ε	✓	$\emptyset$	$\emptyset$
$\emptyset$	✗	$\emptyset$	$\emptyset$
Z	✗	N	Z
N	✓	N	Z

- Language automaton
  1. State = language ℓ accepted when starting from that state.
  2. $\nu\, \ell$: Language ℓ is nullable (accepts the empty word)?
  3. $\delta\, \ell\, a = \{w \mid aw \in \ell\}$: Brzozowski derivative.

Differential equations

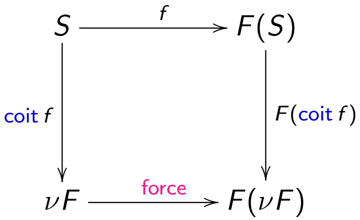
- Language E_0 and friends can be specified by differential equations:
- ν gives the initial value.

$$
\begin{array}{lcl}
\nu\,\emptyset & = & \mathsf{false} \\
\delta\,\emptyset\,x & = & \emptyset \\
\\
\nu\,\varepsilon & = & \mathsf{true} \\
\delta\,\varepsilon\,x & = & \emptyset \\
\\
\nu\,E_0 & = & \mathsf{false} \\
\delta\,E_0\,a & = & \varepsilon \\
\delta\,E_0\,b & = & Z
\end{array}
\qquad
\begin{array}{lcl}
\nu\,N & = & \mathsf{true} \\
\delta\,N\,a & = & N \\
\delta\,N\,b & = & Z \\
\\
\nu\,Z & = & \mathsf{false} \\
\delta\,Z\,a & = & N \\
\delta\,Z\,b & = & Z
\end{array}
$$

- For these simple forms, solutions exist always.
  What is the general story?

Final Coalgebras

- (Weakly) final coalgebra.



- Coiteration = finality witness.

$$\text{force} \circ \text{coit}\, f = F\,(\text{coit}\, f) \circ f$$

- Copattern matching defines coit by corecursion:

$$\text{force}\,(\text{coit}\, f\; s) = F\,(\text{coit}\, f)\,(f\; s)$$

Automata as Coalgebra

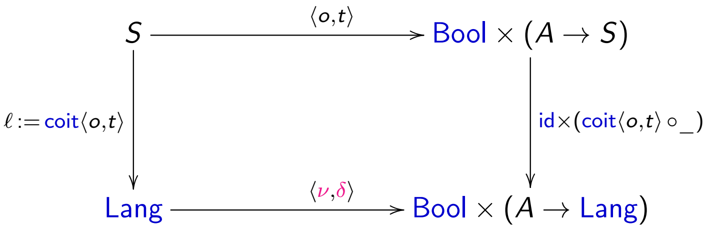
- Arbib & Manes (1986), Rutten (1998), Traytel (2016).
- Automaton structure over set of states S:

$$\begin{array}{lcll} o & : & S \to \mathsf{Bool} & \text{“output”: acceptance} \\ t & : & S \to (A \to S) & \text{transition} \end{array}$$

- Automaton is coalgebra with $F(S) = \mathsf{Bool} \times (A \to S)$.

$$\langle o, t \rangle \quad : \quad S \longrightarrow \mathsf{Bool} \times (A \to S)$$

Formal Languages as Final Coalgebra



$\nu \circ \ell$	$=$	o	"nullable"
$\nu\ (\ell\, s)$	$=$	$o\, s$	
$\delta \circ \ell$	$=$	$(\ell \circ \_) \circ t$	(Brzozowski) derivative
$\delta\ (\ell\, s)$	$=$	$\ell \circ (t\, s)$	
$\delta\ (\ell\, s)\ a$	$=$	$\ell\, (t\, s\, a)$	

Languages – Rule-Based

- Coinductive tries Lang defined via observations/projections ν and δ:
- Lang is the greatest type consistent with these rules:

$$\frac{l : \mathsf{Lang}}{\nu\, l : \mathsf{Bool}} \qquad \frac{l : \mathsf{Lang} \qquad a : A}{\delta\, l\, a : \mathsf{Lang}}$$

- Empty language $\emptyset : \mathsf{Lang}$.
- Language of the empty word $\varepsilon : \mathsf{Lang}$ defined by copattern matching:

$$\begin{array}{lcll} \nu\,\varepsilon & = & \mathsf{true} & : \mathsf{Bool} \\ \delta\,\varepsilon\,a & = & \emptyset & : \mathsf{Lang} \end{array}$$

Corecursion

- Empty language $\emptyset : \mathsf{Lang}$ defined by corecursion:

$$\begin{array}{lcl} \nu\,\emptyset & = & \mathsf{false} \\ \delta\,\emptyset\,a & = & \emptyset \end{array}$$

- Language union $k \cup l$ is pointwise disjunction:

$$\begin{array}{lcl} \nu\,(k \cup l) & = & \nu\,k \vee \nu\,l \\ \delta\,(k \cup l)\,a & = & \delta\,k\,a \cup \delta\,l\,a \end{array}$$

- Language composition $k \cdot l$ à la Brzozowski:

$$\begin{array}{lcl} \nu\,(k \cdot l) & = & \nu\,k \wedge \nu\,l \\ \delta\,(k \cdot l)\,a & = & \begin{cases} (\delta\,k\,a \cdot l) \cup \delta\,l\,a & \text{if } \nu\,k \\ (\delta\,k\,a \cdot l) & \text{otherwise} \end{cases} \end{array}$$

- Not accepted because $\cup$ is not a constructor.

Bisimilarity

- Equality of infinite tries is defined coinductively.
- _≅_ is the greatest relation consistent with

$$\frac{l \cong k}{\nu\, l \equiv \nu\, k}\ \cong\nu \qquad \frac{l \cong k \qquad a : A}{\delta\, l\, a \cong \delta\, k\, a}\ \cong\delta$$

- Equivalence relation via provable ≅refl, ≅sym, and ≅trans.

$$\begin{array}{lcll} \cong\text{trans} & : & (p : l \cong k) \to (q : k \cong m) \to l \cong m & \\ \cong\nu\,(\cong\text{trans}\, p\, q) & = & \equiv\text{trans}\,(\cong\nu\, p)\ \ (\cong\nu\, q) & : \ \nu\, l \equiv \nu\, k \\ \cong\delta\,(\cong\text{trans}\, p\, q)\, a & = & \cong\text{trans}\,(\cong\delta\, p\, a)\,(\cong\delta\, q\, a) & : \ \delta\, l\, a \cong \delta\, m\, a \end{array}$$

- Congruence for language constructions.

$$\frac{k \cong k' \qquad l \cong l'}{(k \cup k') \cong (l \cup l')}\ \cong\cup$$

Proving bisimilarity

- Composition distributes over union.

$$\mathsf{dist} \;:\; \forall\, k\, l\, m.\;\; k \cdot (l \cup m) \cong (k \cdot l) \cup (k \cdot m)$$

- Proof. Observation $\delta\ \_\ a$, case k nullable.

$$\begin{array}{lllr}
 & \delta\,(k \cdot (l \cup m))\,a & & \\
= & \boxed{\delta\,k\,a \cdot (l \cup m)} & \cup\ \delta\,(l \cup m)\,a & \text{by definition} \\
\cong & \boxed{(\delta\,k\,a \cdot l \cup \delta\,k\,a \cdot m)} & \cup\ (\delta\,l\,a \cup \delta\,m\,a) & \text{by coind. hyp. (wish)} \\
\cong & (\delta\,k\,a \cdot l \cup \delta\,l\,a) \cup (\delta\,k\,a \cdot m \cup \delta\,m\,a) & & \text{by union laws} \\
= & \delta\,((k \cdot l) \cup (k \cdot m))\,a & & \text{by definition}
\end{array}$$

- Formal proof attempt.

$$\cong\!\delta\ \mathsf{dist}\ a \;=\; \cong\!\mathsf{trans}\ (\cong\!\cup\ \boxed{\mathsf{dist}}\ \ldots)\ \ldots$$

- Not coiterative / guarded by constructors!

Construction of greatest fixed-points

- Iteration to greatest fixed-point.

$$\top \supseteq F(\top) \supseteq F^2(\top) \supseteq \cdots \supseteq F^\omega(\top) = \bigcap_{n<\omega} F^n(\top)$$

- Naming $\nu^i F = F^i(\top)$.

$$\begin{array}{lcl} \nu^0 \; F & = & \top \\ \nu^{n+1} \; F & = & F\,(\nu^n F) \\ \nu^\omega \; F & = & \bigcap_{n<\omega} \nu^n F \end{array}$$

- Deflationary iteration.

$$\nu^i \; F \;=\; \bigcap_{j<i} F\,(\nu^j F)$$

Sized coinductive types

- Add to syntax of type theory

Size	type of ordinals
i	ordinal variables
$\nu^i F$	sized coinductive type
Size$<i$	type of ordinals below i

- Bounded quantification $\forall j{<}i.\, A = (j : \mathsf{Size}{<}\, i) \to A$.
- Well-founded recursion on ordinals, roughly:

$$\frac{f : \forall\, i.\, (\forall\, j{<}i.\, \nu^j F) \to \nu^i F}{\mathsf{fix}\, f : \forall\, i.\, \nu^i F}$$

Sized coinductive type of languages

- $\mathsf{Lang}\, i \cong \mathsf{Bool} \times (\forall j{<}i.\; A \to \mathsf{Lang}\, j)$

$$\frac{l : \mathsf{Lang}\, i}{\nu\, l : \mathsf{Bool}} \qquad \frac{l : \mathsf{Lang}\, i \qquad j < i \qquad a : A}{\delta\, l\, \{j\}\, a : \mathsf{Lang}\, j}$$

- $\emptyset : \forall i.\, \mathsf{Lang}\, i$ by copatterns and induction on i:

$$\begin{array}{lcl@{\quad}c@{\quad}l} \nu\, (\emptyset\, \{i\}) & = & \mathsf{false} & : & \mathsf{Bool} \\ \delta\, (\emptyset\, \{i\})\, \{j\}\, a & = & \emptyset\, \{j\} & : & \mathsf{Lang}\, j \end{array}$$

- Note $j < i$.
- On right hand side, $\emptyset : \forall j{<}i.\, \mathsf{Lang}\, j$ (coinductive hypothesis).

Type-based guardedness checking

- Union preserves size/guardeness:

$$\frac{k : \mathsf{Lang}\, i \qquad l : \mathsf{Lang}\, i}{k \cup l : \mathsf{Lang}\, i}$$

$$\begin{array}{lcl} \nu\,(k \cup l) & = & \nu\, k \vee \nu\, l \\ \delta\,(k \cup l)\,\{j\}\,a & = & \delta\, k\,\{j\}\,a \cup \delta\, l\,\{j\}\,a \end{array}$$

- Composition is accepted and also guardedness-preserving:

$$\frac{k : \mathsf{Lang}\, i \qquad l : \mathsf{Lang}\, i}{k \cdot l : \mathsf{Lang}\, i}$$

$$\begin{array}{lcl} \nu\,(k \cdot l) & = & \nu\, k \wedge \nu\, l \\ \delta\,(k \cdot l)\,\{j\}\,a & = & \begin{cases} (\delta\, k\,\{j\}\,a \cdot l) \cup \delta\, l\,\{j\}\,a & \text{if } \nu\, k \\ (\delta\, k\,\{j\}\,a \cdot l) & \text{otherwise} \end{cases} \end{array}$$

Guardedness-preserving bisimilarity proofs

- Sized bisimilarity $\cong$ is greatest family of relations consistent with

$$\frac{l \cong^i k}{\nu\, l \equiv \nu\, k}\ \cong\nu \qquad \frac{l \cong^i k \qquad j < i \qquad a : A}{\delta\, l\, a \cong^j \delta\, k\, a}\ \cong\delta$$

- Equivalence and congruence rules are guardedness preserving.

$$\begin{array}{lcll}
\cong\text{trans} & : & (p : l \cong^i k) \to (q : k \cong^i m) \to l \cong^i m & \\
\cong\nu\,(\cong\text{trans}\, p\, q) & = & \equiv\text{trans}\,(\cong\nu\, p)\,(\cong\nu\, q) & : \ \nu\, l \equiv \nu\, k \\
\cong\delta\,(\cong\text{trans}\, p\, q)\, j\, a & = & \cong\text{trans}\,(\cong\delta\, p\, j\, a)\,(\cong\delta\, q\, j\, a) & : \ \delta\, l\, a \cong^j \delta\, m\, a
\end{array}$$

- Coinductive proof of dist accepted.

$$\cong\delta\ \text{dist}\ j\ a\ =\ \cong\text{trans}\ j\ (\cong\cup\ \boxed{(\text{dist}\ \ j)}\ (\cong\text{refl}\ j))\ \ldots$$

Conclusions

- Tracking guardedness in types allows
  - natural modular corecursive definition
  - natural bisimilarity proof using equation chains
- Implemented in Agda (ongoing)
- Abel et al (POPL 13): Copatterns
- Abel/Pientka (ICFP 13): Well-founded recursion with copatterns

Related work

- Hagino (1987): Coalgebraic types
- Cockett et al.: Charity
- Dmitriy Traytel (PhD TU Munich, 2015): Languages coinductively in Isabelle
- Kozen, Silva (2016): Practical coinduction
- Hughes, Pareto, Sabry (POPL 1996)
- Papers on sized types (1998–2015): e.g. Sacchini (LICS 2013)