Compositional Coinduction with Sized Types

Andreas Abel

Department of Computer Science and Engineering
Chalmers and Gothenburg University

13th International Workshop on
Coalgebraic Methods in Computer Science (CMCS'16)
Eindhoven, The Netherlands
3 April 2016

Questions

- How to reason by coinduction informally?
- How to represent coinductive definitions and proofs in a proof assistant?
- Popularity of Coq and Agda: How to do coinduction in type theory?
- What are the problems with the state-of-the-art (e.g. Coq's guardedness checker)?
- How to get compositional coinduction?

Contents

(Martin-Löf) Type Theory

- Meta-language for mathematics, logics, and computer science.
- Functional programming language based on typed λ-calculus.
- Dependent types allow natural formalizations and rich specifications.

$$\begin{array}{l} \mathrm{divide}\ :\ (n : \mathbb{N}) \to (d : \mathbb{N}) \to (p : d \not\equiv 0) \to \exists\, q\, r.\, n \equiv d \cdot q + r \\ \mathrm{divide} = \lambda\, n\, d\, p \to \ldots \end{array}$$

- Propositions-as-types:

$$\mathrm{Prop} = \mathrm{Type}$$

  - A proposition is a type (the set of its proofs).
  - An empty type denotes a false proposition.
  - To prove a proposition, construct an inhabitant of the type.

Type Theory – Computability and Decidability

- Constructive: All functions are computable.
- Excluded middle does not hold for all propositions.

$$\nvdash (A : \mathrm{Prop}) \to A + (A \to \bot)$$

- It holds for exactly the decidable propositions.

$$\mathrm{Dec}\, A = A + (A \to \bot)$$

- Sets are modeled by predicates, e.g., $\mathrm{Prime} : \mathbb{N} \to \mathrm{Prop}$.
- Decidable sets can be modeled by their characteristic functions into Bool or Dec.

Type Theory – Equality

- Built-in definitional equality $\vdash t = t' : A$ (same β normal form).
- Propositional equality $x \equiv y$ (where $x, y : A$) is the least type closed und the single introduction rule

$$\frac{\vdash x = y : A}{\vdash \mathsf{refl} : x \equiv y}$$

- Extensional only for types of finite trees, i.e., types built from $\bot$ (aka 0), $\top$ (aka 1), $\uplus$ (aka $+$), $\times$ and μ (least fixed point).
- Intensional for types involving $\to$, ν, and universes.
- For function types, we might add the axiom of function extensionality.

$$(\forall x.\ f\,x \equiv g\,x) \to f \equiv g$$

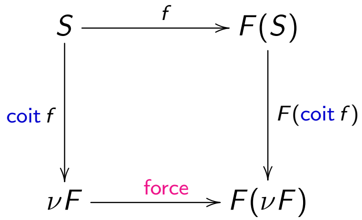
- For coinductive types, we define coinductive equality (bisimilarity).

Coinductive Definition and Reasoning

- How to reason about coinductive equality in Type Theory? Literature: bisimulations, up-to techniques.
- Can we reason with coinductive equality directly in a modular way in Type Theory?
- Can we define corecursive functions in a modular way?
- How to extend Type Theory to do this?
- What is a coinductive definition anyway?

Final Coalgebras

- (Weakly) final coalgebra.



- Coiteration = finality witness.

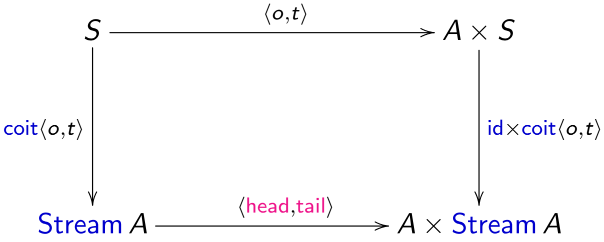
$$\mathsf{force} \circ \mathsf{coit}\, f = F\,(\mathsf{coit}\, f) \circ f$$

- Copattern matching *defines* coit by corecursion:

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

Streams as Final Coalgebra

- Output automaton is coalgebra $\langle o, t \rangle : S \to A \times S$.
- Final coalgebra = automaton unrolling = stream: $\nu S.\, A \times S$.



- Termination by induction on observation depth:

$$\begin{array}{lcl} \mathsf{head}\ (\mathsf{coit}\ \langle o, t \rangle\ s) & = & o\ s \\ \mathsf{tail}\ (\mathsf{coit}\ \langle o, t \rangle\ s) & = & \mathsf{coit}\ \langle o, t \rangle\ (t\ s) \end{array}$$

Automata as Coalgebra

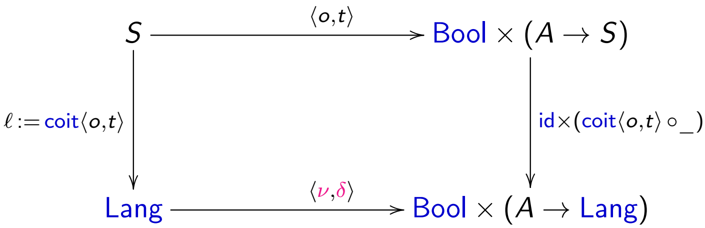
- Arbib & Manes (1986), Rutten (1998), Traytel (2016).
- Automaton structure over set of states S:

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

- 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



$$
\begin{array}{lcll}
\nu \circ \ell & = & o & \text{"nullable"} \\
\nu\ (\ell\, s) & = & o\, s & \\
\delta \circ \ell & = & (\ell \circ \_) \circ t & \text{(Brzozowski) derivative} \\
\delta\ (\ell\, s) & = & \ell \circ (t\, s) & \\
\delta\ (\ell\, s)\ a & = & \ell\,(t\, s\, a) &
\end{array}
$$

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:

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

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.
- $\_ \cong \_$ 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, l not nullable.

$$\begin{array}{lll} \multicolumn{3}{l}{\delta\,(k \cdot (l \cup m))\, a} \\ = & \boxed{\delta\, k\, a \cdot (l \cup m)} \quad \cup\ \delta\,(l \cup m)\, a & \text{by definition} \\ \cong & \boxed{(\delta\, k\, a \cdot l \cup \delta\, k\, a \cdot m)} \cup (\delta\, l\, a \cup \delta\, m\, a) & \text{by coind. hyp. (wish)} \\ \cong & (\delta\, k\, a \cdot l \cup \delta\, l\, a) \cup (\delta\, k\, a \cdot m \cup \delta\, m\, a) & \text{by union laws} \\ = & \delta\,((k \cdot l) \cup (k \cdot m))\, a & \text{by definition} \end{array}$$

- 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.

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

- Coinductive proof of dist accepted.

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

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)