Bisimulation as path type for guarded recursive types

Niccolò Veltri
Joint work with Rasmus Ejlers Møgelberg

IT University of Copenhagen

EUTypes Meeting, Aarhus, October 9, 2018

Motivation

- In type theory, coinductive types are used to represent processes.
- In existing systems such as Coq or Agda, programming and reasoning about coinductive types is difficult:
  - Recursively defined data of coinductive type must be productive.
  - Mismatch between bisimilarity and propositional equality.
- Encoding productivity in types:
  - Sized types
  - Guarded recursion

In this talk

- We work in an extension of cubical type theory with guarded recursion.
- In this system, bisimilarity is equivalent to path equality for guarded recursive types.
- First step towards bisimilarity as path equality for coinductive types.
- Worked example of guarded labelled transition systems.
- The latter uses finite powerset defined as a higher inductive type (Frumin et al. 2018).

Cubical Type Theory (Cohen et al. 2015)

- Path types

$$\frac{\Gamma \vdash A \qquad \Gamma, i : \mathbb{I} \vdash p : A}{\Gamma \vdash \lambda i.\, p : \mathsf{Path}_A\,(p(i/0))\,(p(i/1))}$$

$$\frac{\Gamma \vdash p : \mathsf{Path}_A\, x\, y \qquad \Gamma \vdash r : \mathbb{I}}{\Gamma \vdash p\, r : A} \qquad p\,0 \equiv x \quad p\,1 \equiv y$$

- We also write $x =_A y$ or $x = y$ for $\mathsf{Path}_A\, x\, y$
- Composition operator needed to compose paths
- Univalence is derivable:

$$\mathsf{Path}_{\mathsf{U}}\, A\, B \simeq (A \simeq B)$$

- Higher inductive types (Coquand et al. 2018)

Finite power sets as HIT (Frumin et al. 2018)

$$\frac{}{\varnothing : \mathsf{P}_f A} \qquad \frac{a : A}{\{a\} : \mathsf{P}_f A} \qquad \frac{x, y : \mathsf{P}_f A}{x \cup y : \mathsf{P}_f A}$$

$$\frac{x : \mathsf{P}_f A}{\mathsf{nl}\, x : \varnothing \cup x = x} \qquad \frac{x, y, z : \mathsf{P}_f A}{\mathsf{assoc}\, x\, y\, z : (x \cup y) \cup z = x \cup (y \cup z)}$$

$$\frac{x : \mathsf{P}_f A}{\mathsf{idem}\, x : x \cup x = x} \qquad \frac{x, y : \mathsf{P}_f A}{\mathsf{com}\, x\, y : x \cup y = y \cup x}$$

set-truncation constructor

- Membership $\in : A \to \mathsf{P}_f A \to \mathrm{Prop}$ defined by induction.
- Extensional equality

$$(x =_{\mathsf{P}_f A} y) \simeq \left(\prod a : A.\, a \in x \leftrightarrow a \in y\right)$$

Guarded recursion

- Guarded fixed point operator:

$$\begin{aligned} \mathrm{fix} &: (\triangleright A \to A) \to A \\ \mathrm{fix}(f) &= f(\mathrm{next}(\mathrm{fix}(f))) \\ \mathrm{next} &: A \to \triangleright A \end{aligned}$$

- Guarded recursive types:

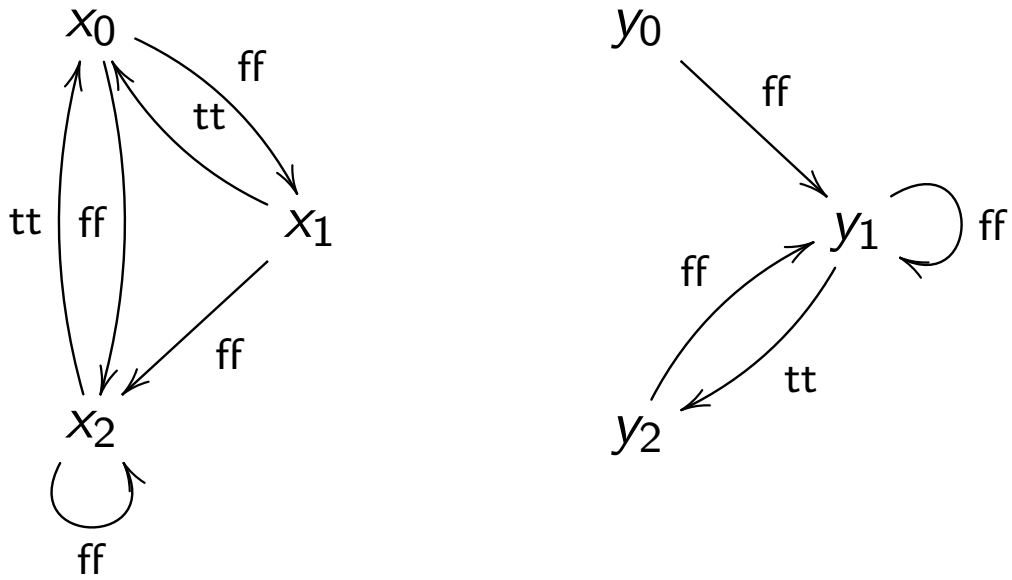
$$\mathsf{Str}^{\mathsf{g}} \simeq \mathrm{Nat} \times \triangleright \mathsf{Str}^{\mathsf{g}}$$

$$\begin{aligned} \mathsf{fold} &: \mathrm{Nat} \times \triangleright \mathsf{Str}^{\mathsf{g}} \to \mathsf{Str}^{\mathsf{g}} \\ \mathsf{unfold} &: \mathsf{Str}^{\mathsf{g}} \to \mathrm{Nat} \times \triangleright \mathsf{Str}^{\mathsf{g}} \end{aligned}$$

- $\mathsf{Str}^{\mathsf{g}}$ is *not* the usual coinductive type of streams, but it can be used to encode it (Atkey & McBride 2013).

Guarded labelled transition systems (GLTS)

- Coalgebras $c : X \to \mathsf{P}_f(A \times \triangleright X)$
- We write $x \xrightarrow{a}_c x'$ for $(a, x') \in c\, x$
- Example

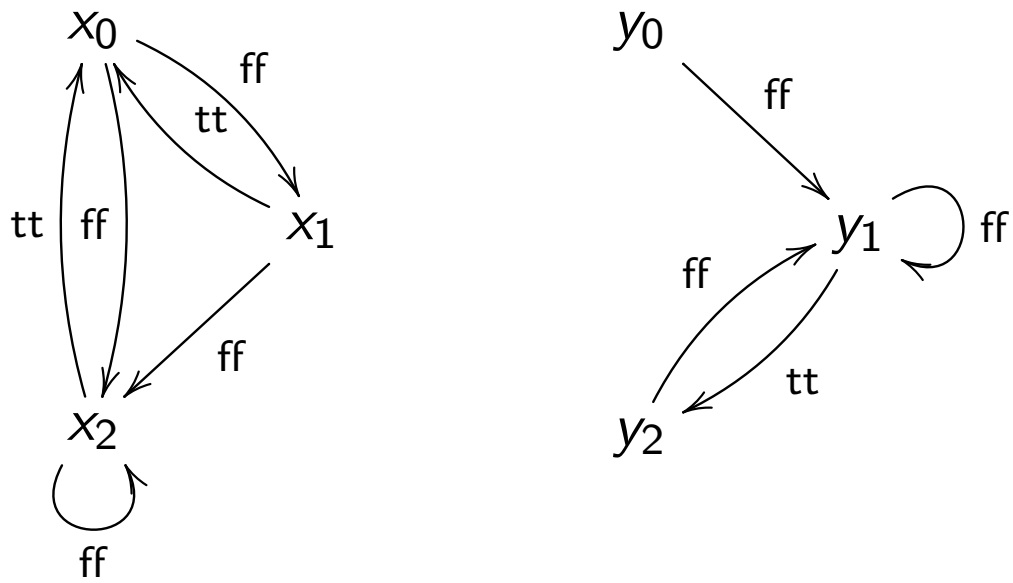


- $X \equiv_{\text{def}} \{x_0, x_1, x_2, y_0, y_1, y_2\}$

$$c\, x_0 \equiv_{\text{def}} \{\mathsf{ff}, \text{next}\, x_1\} \cup \{\mathsf{ff}, \text{next}\, x_2\}$$

$$\ldots$$

The GLTS of processes



- Guarded recursive type $\mathrm{Proc}^{\mathrm{g}} \simeq \mathrm{P}_f(A \times \triangleright \mathrm{Proc}^{\mathrm{g}})$
- Define $px_0, px_1, px_2, py_0, py_1, py_2 : \mathrm{Proc}^{\mathrm{g}}$ by guarded recursion

$$px_0 = \mathsf{fold}(\{\mathsf{ff}, \mathrm{next}\, px_1\} \cup \{\mathsf{ff}, \mathrm{next}\, px_2\})$$

$$\ldots$$

- Formally fixed point of map $\triangleright(\mathrm{Proc}^{\mathrm{g}6}) \to \mathrm{Proc}^{\mathrm{g}6}$

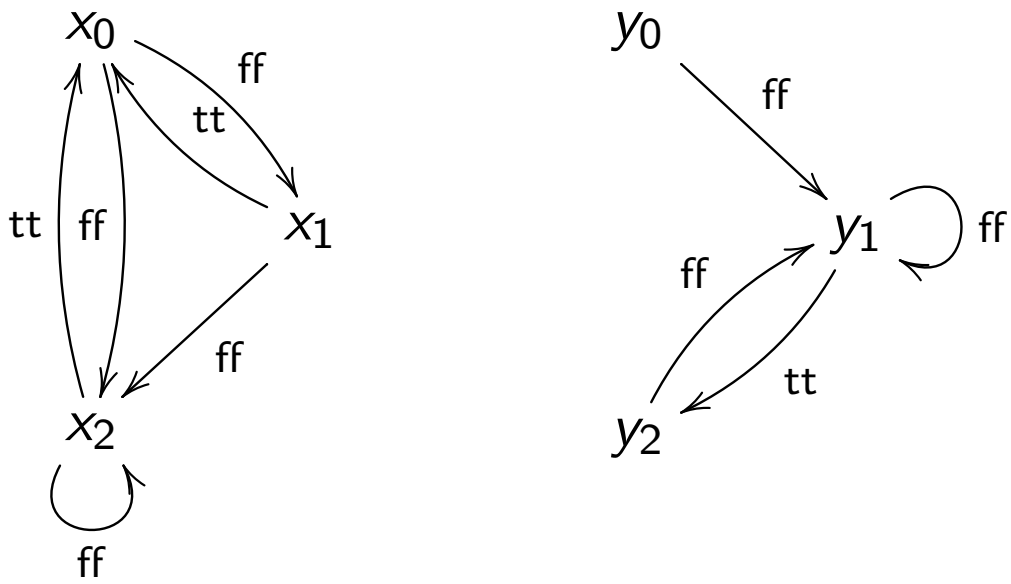
The GLTS of processes

- $\mathrm{Proc}^{\mathrm{g}} \simeq \mathrm{P}_f(A \times \triangleright \mathrm{Proc}^{\mathrm{g}})$ final coalgebra

$$\begin{array}{ccc} X & \xrightarrow{c} & \mathrm{P}_f(A \times \triangleright X) \\ {\scriptstyle [\![-]\!]}\downarrow & & \downarrow{\scriptstyle \mathrm{P}_f(A \times \triangleright [\![-]\!])} \\ \mathrm{Proc}^{\mathrm{g}} & \xrightarrow{\text{unfold}} & \mathrm{P}_f(A \times \triangleright \mathrm{Proc}^{\mathrm{g}}) \end{array}$$

- **Lemma.** $[\![x_i]\!] = px_i$, $[\![y_i]\!] = py_i$ for $i = 0, 1, 2$.
- **Proof.** By guarded recursion

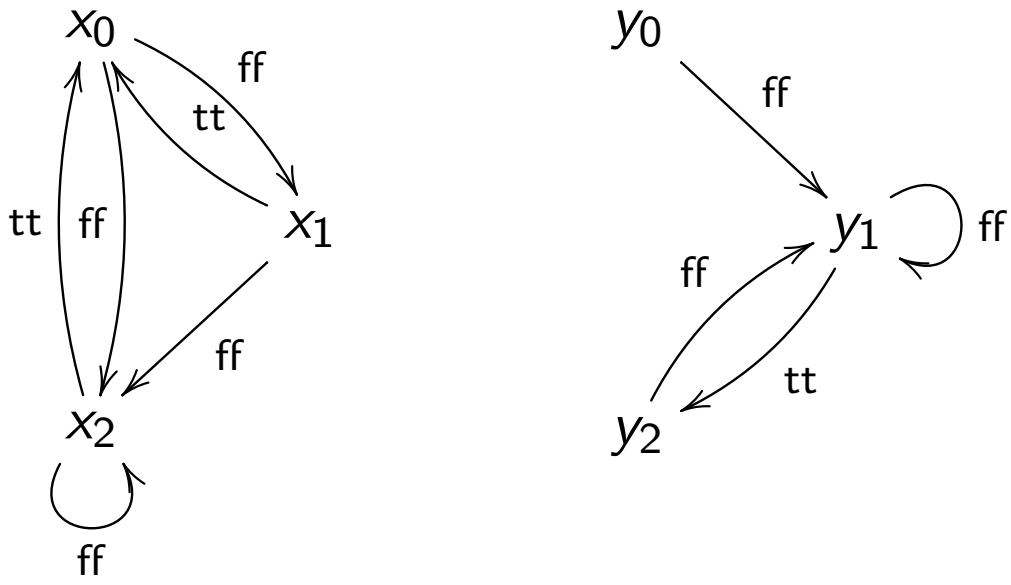
Bisimilarity example



- **Proposition.** $px_0 = py_0$.
- **Lemma.** $px_0 = py_0 \times px_1 = py_1 \times px_0 = py_2 \times px_2 = py_1$.
  - In the proof we need extensionality for the later modality:

$$\mathsf{Path}_{\triangleright A}\,(\mathrm{next}\,x)\,(\mathrm{next}\,y) \simeq \triangleright(\mathsf{Path}_A\,x\,y)$$

Bisimilarity example, proof



- Suppose
  $\triangleright((px_0 = py_0) \times (px_1 = py_1) \times (px_0 = py_2) \times (px_2 = py_1))$.

$$\begin{aligned} px_0 &= \mathsf{fold}(\{\mathsf{ff}, \mathrm{next}\, px_1\} \cup \{\mathsf{ff}, \mathrm{next}\, px_2\}) \\ &= \mathsf{fold}(\{\mathsf{ff}, \mathrm{next}\, py_1\} \cup \{\mathsf{ff}, \mathrm{next}\, py_1\}) \\ &= \mathsf{fold}(\{\mathsf{ff}, \mathrm{next}\, py_1\}) \\ &= py_0 \end{aligned}$$

- Other cases similar.

Bisimulation

- Let $c : X \to \mathsf{P}_f(A \times \triangleright X)$
- Let $R : X \to X \to \mathsf{U}$
- Simulation if $R(x, y)$ and $x \xrightarrow{a}_c x'$ implies

$$\exists\, y' : \triangleright X.\, (y \xrightarrow{a}_c y') \times \triangleright R\,??$$

- R is a relation on X but $x', y' : \triangleright X$.
- Solutions:
  - Delayed substitutions, used in Guarded Cubical Type Theory (Birkedal et al. 2016)
  - Ticks from Clocked Type Theory (Bahr et al. 2017)

Ticked Cubical Type Theory

- Extension of Cubical Type Theory with rules

$$\frac{\Gamma \vdash}{\Gamma, \alpha : \mathrm{tick} \vdash} \qquad \frac{\Gamma, \alpha : \mathrm{tick} \vdash A \text{ type}}{\Gamma \vdash \triangleright\,(\alpha : \mathrm{tick}).A \text{ type}}$$

$$\frac{\Gamma, \alpha : \mathrm{tick} \vdash t : A}{\Gamma \vdash \lambda(\alpha : \mathrm{tick}).t : \triangleright\,(\alpha : \mathrm{tick}).A} \qquad \frac{\Gamma \vdash t : \triangleright\,(\alpha : \mathrm{tick}).A}{\Gamma, \beta : \mathrm{tick}, \Gamma' \vdash t\,[\beta] : A(\alpha/\beta)}$$

- Write $\triangleright\, A$ for $\triangleright\,(\alpha : \mathrm{tick}).A$ when α not free in A.
- Example: The function next:

$$\lambda(x : A).\lambda(\alpha : \mathrm{tick}).x \;:\; A \to \triangleright\, A$$

Ticked Cubical Type Theory

- Additional rules:

$$\frac{\Gamma, \alpha : \mathrm{tick}, i : \mathbb{I} \vdash A}{\Gamma, i : \mathbb{I}, \alpha : \mathrm{tick} \vdash A} \qquad \frac{\Gamma, \alpha : \mathrm{tick}, i : \mathbb{I} \vdash t : A}{\Gamma, i : \mathbb{I}, \alpha : \mathrm{tick} \vdash t : A}$$

$$\frac{\Gamma, \alpha : \mathrm{tick}, \varphi \vdash A}{\Gamma, \varphi, \alpha : \mathrm{tick} \vdash A} \qquad \frac{\Gamma, \alpha : \mathrm{tick}, \varphi \vdash t : A}{\Gamma, \varphi, \alpha : \mathrm{tick} \vdash t : A}$$

- Consequences:
  - Extensionality principle for the later modality

$$\mathsf{Path}_{\triangleright\,(\alpha:\mathrm{tick}).A}\, x\, y \simeq \triangleright\,(\alpha : \mathrm{tick}).(\mathsf{Path}_A\,(x\,[\alpha])\,(y\,[\alpha]))$$

  - Composition operation for $\triangleright$

Fixed points and guarded recursive types

- Fixed point equality:

$$\frac{\Gamma, x : \triangleright A \vdash t : A}{\Gamma \vdash \mathsf{pfix}\, x.t : \mathsf{Path}_A(\mathrm{fix}\, x.t)(t(x/\mathrm{next}(\mathrm{fix}\, x.t)))}$$

- Processes

$$\mathsf{Proc}^{\mathsf{g}} \equiv \mathrm{fix}\, X.\mathsf{P}_f(A \times \triangleright(\alpha : \mathrm{tick}).X[\alpha]) : \mathsf{U}$$

- Then pfix proves

$$\mathsf{Path}_{\mathsf{U}}(\mathsf{Proc}^{\mathsf{g}})(\mathsf{P}_f(A \times \triangleright \mathsf{Proc}^{\mathsf{g}}))$$

- Terms

$$\mathsf{fold} : \mathsf{P}_f(A \times \triangleright \mathsf{Proc}^{\mathsf{g}}) \to \mathsf{Proc}^{\mathsf{g}}$$
$$\mathsf{unfold} : \mathsf{Proc}^{\mathsf{g}} \to \mathsf{P}_f(A \times \triangleright \mathsf{Proc}^{\mathsf{g}})$$

Back to bisimulation

- Let $c : X \to \mathsf{P}_f(A \times \triangleright X)$
- Let $R : X \to X \to \mathsf{U}$
- Simulation if $R(x, y)$ and $x \xrightarrow{a}_c x'$ implies

$$\exists\, y' : \triangleright X.\, (y \xrightarrow{a}_c y') \times \triangleright (\alpha : \mathrm{tick}).\, R\,(x'\,[\alpha])\,(y'\,[\alpha]))$$

- R is a bisimulation if R and R^{op} are both simulations
- Define by guarded recursion $\sim_c$ bisimilarity, the greatest bisimulation on c.
- **Theorem.** For all $p, q : \mathrm{Proc}^{\mathrm{g}}$, the types $p \sim_{\mathsf{unfold}} q$ and $p = q$ are equivalent.

Coalgebraic bisimilarity

- Let $F : \mathsf{U} \to \mathsf{U}$ functor, let $\nu F^{\mathrm{g}} \simeq F(\triangleright \nu F^{\mathrm{g}})$.
- Using relation lifting, we define an abstract notion of bisimulation for a coalgebra $c : X \to F(\triangleright X)$.
- Greatest bisimulation $\sim_c$ can be defined by guarded recursion.
- **Theorem.** For all $x, y : \nu F^{\mathrm{g}}$, the types $x \sim_{\mathsf{unfold}} y$ and $\mathsf{Path}_{\nu F^{\mathrm{g}}}\, x\, y$ are equivalent.

Model

- In the category of cubical trees

$$\mathbf{Set}^{(\mathcal{C}\times\omega)^{\mathrm{op}}}$$

- Previously used to model Guarded Cubical Type Theory
- Extend with ticks
- Extend with HITs using (Coquand et al 2018)

Summary

- In Ticked Cubical Type Theory, bisimilarity is equivalent to path equality for guarded recursive types
- Simple equational reasoning for guarded labelled transition systems
- Future work:
  - More examples: CCS, up-to techniques
  - Extend with $\forall\kappa$ to encode coinductive types
  - Challenge: encode $\nu X.\mathsf{P}_f(A \times X)$
  - Requires: $\forall\kappa.\mathsf{P}_f(-) \simeq \mathsf{P}_f(\forall\kappa.(-))$