A categorical framework for congruence of bisimilarity

Tom Hirschowitz and Ambroise Lafont

Topos Institute Colloquium 2023

Zooming in [1]

- Operational semantics.
    - Behavioural equivalences.
        - Bisimilarity.
            - Congruence of bisimilarity.

[1] ©Damien Pous

Operational semantics

> A set of techniques for constructing mathematical models of programming languages.

General idea:

- Programs $\in$ syntax, an inductively-generated object.
  (Think initial algebra for some endofunctor Σ.)
- Evaluation steps $\approx$ (directed) edges between programs.

$\rightsquigarrow$ Evaluation graph.

Behavioural equivalences

Goal

Correctness of program transformations.

Typically: optimisations.

Observational equivalence of programs fragments, $P \approx Q$:

Definition

Fix some basic type, e.g., the booleans `bool`.

For all valid contexts C of type `bool`, $C\{P\}$"="$C\{Q\}$, i.e.,

- one terminates iff the other does, and
- when they do, they converge to the same boolean.

Problem!

Hard to establish.

Bisimilarity

Standard idea

Find some different equivalence relation $\sim$ such that

$$P \sim Q \qquad \Longrightarrow \qquad P \approx Q,$$

and $\sim$ is easier to establish than $\approx$.

Typical choice for $\sim$: bisimilarity.

Bisimilarity, $\sim$

How to prove $P \sim Q$?

Exhibit a bisimulation $\mathscr{R}$ such that $P \mathrel{\mathscr{R}} Q$.

Definition (Bisimulation)

$$\begin{tikzcd}
P \arrow[r, dash, "\mathscr{R}"] \arrow[d, "\forall\text{step}"'] & Q \arrow[d, dashed, "\exists\text{step}"] \\
P' \arrow[r, dash, dashed, "\mathscr{R}"] & Q'
\end{tikzcd}$$

and

$$\begin{tikzcd}
P \arrow[r, dash, "\mathscr{R}"] \arrow[d, dashed, "\exists\text{step}"'] & Q \arrow[d, "\forall\text{step}"] \\
P' \arrow[r, dash, dashed, "\mathscr{R}"] & Q'
\end{tikzcd}$$

Indeed:

Bisimilarity is the largest bisimulation.

Enhanced bisimilarity

In a higher-order setting, one often restricts to

Definition (Enhanced relations)

Closed under auxiliary operations, typically capture-avoiding substitution:

$$P \mathscr{R} Q \qquad \Longrightarrow \qquad P[\sigma] \mathscr{R} Q[\sigma],$$

where σ : variables → programs.

Bisimulation + enhancement

Enhanced bisimilarity := greatest enhanced bisimulation relation.

In pure λ-calculus

Enhanced bisimilarity = Abramsky's applicative bisimilarity.

Congruence of enhanced bisimilarity

Crucial step for $P \sim Q \Longrightarrow P \approx Q$

Enhanced bisimilarity is a congruence:

$$\forall C, P \sim Q \Longrightarrow C\{P\} \sim C\{Q\}.$$

Far from obvious:

- False in Milner's π-calculus, a kernel language for concurrent programming.
- Hard in pure λ-calculus.
  - Domain-theoretic proof by Abramsky.
  - Syntactic proof by Howe → Howe's method.

The goal

A general congruence theorem for enhanced bisimilarity.

A glimpse of previous work

- Syntactic frameworks (Howe, 1996; Bernstein, 1998).
  Limited to untyped, monosorted languages.
- Categorical frameworks (Turi and Plotkin, 1997; Fiore and Staton, 2001; Staton, 2008).
  Did not cover higher-order languages until
  - a first framework with Borthelle (BHL 2020), relational, and
  - recent work by Goncharov et al. (G+ 2023, see recent Colloquium talk by Sergey), coalgebraic.

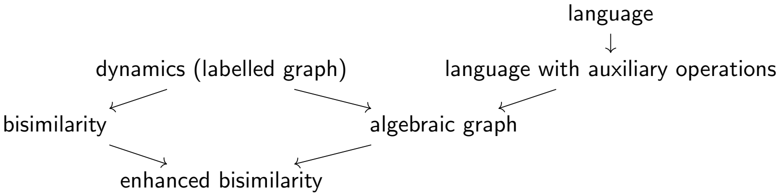
Quick comparison with G+ 2023 (more of a wild guess, really):

- In principle, coalgebra covers more “transition flavours” (e.g., probabilistic languages),
- but less “rule arities” (see below).

This work: generalises and simplifies BHL 2020.

The plan

- Introduce abstract notions of



language

language with auxiliary operations

dynamics (labelled graph)

bisimilarity

algebraic graph

enhanced bisimilarity

- Introduce signatures for generating algebraic graphs from basic data (as in initial-algebra semantics).
- Prove:

> Under suitable hypotheses, enhanced bisimilarity is a congruence in the generated algebraic graphs.

But let's start with a concrete example.

A language: λ-calculus with delimited continuations

Syntax (| means "or"):

$$\begin{aligned}\text{Values} \ni v &::= x \mid \lambda x.e \\ \text{Programs} \ni e &::= v \mid e_1\ e_2 \mid \mathcal{S}x.e \mid \langle e \rangle \\ \text{Evaluation contexts} \ni E &::= \square \mid E\ e \mid v\ E\end{aligned}$$

where x binds in e, in both $\lambda x.e$ and $\mathcal{S}x.e$.

Evaluation context application and composition:

$$\begin{aligned}\square\{e\} &= e & \square\{E'\} &= E' \\ (E\ e')\{e\} &= E\{e\}\ e' & (E\ e')\{E'\} &= E\{E'\}\ e' \\ (v\ E)\{e\} &= v\ E\{e\} & (v\ E)\{E'\} &= v\ E\{E'\}.\end{aligned}$$

Capture-avoiding substitution:

$$\begin{aligned}x[\sigma] &= \sigma(x) \\ (e_1\ e_2)[\sigma] &= e_1[\sigma]\ e_2[\sigma]\end{aligned}$$

$$\ldots$$

Transition types

Three types of transitions (between closed programs):

Silent $e \xrightarrow{\tau} e'$: e transitions to e'.

Applicative $e \xrightarrow{v} e'$: applying v to e leads to e'.

$$\boxed{(\approx \quad e\ v \xrightarrow{\tau} e'.)}$$

Preemptive $e \xrightarrow{E} e'$: e grabs the current context, say E, and then transitions to e'.

$$\boxed{(\approx \quad \langle E\{e\}\rangle \xrightarrow{\tau} e'.)}$$

Distinguished by the label type: $\{\tau\}$, value, evaluation context.

A labelled graph

Defined inductively by transition rules (≈ inductive clauses).

Example and notation

$$\frac{e_1 \xrightarrow{E\{v\ \square\}} e_2}{v\ e_1 \xrightarrow{E} e_2}$$

means

- if $e_1 \xrightarrow{E\{v\ \square\}} e_2$
- then $v\ e_1 \xrightarrow{E} e_2$.

$$\langle E\{\underline{v\ e_1}\}\rangle \xrightarrow{\tau} e_2 \qquad \text{vs} \qquad \langle E\{v\ \underline{e_1}\}\rangle \xrightarrow{\tau} e_2$$

Definition

Our triple of labelled relations is the smallest satisfying all rules.

Quite a few rules

$$\frac{e_1 \xrightarrow{v} e_2}{e_1\ v \xrightarrow{\tau} e_2} \qquad \frac{}{\lambda x.e \xrightarrow{v} e[x \mapsto v]} \qquad \frac{e_1 \xrightarrow{\tau} e_1'}{e_1\ e_2 \xrightarrow{\tau} e_1'\ e_2} \qquad \frac{e_2 \xrightarrow{\tau} e_2'}{v\ e_2 \xrightarrow{\tau} v\ e_2'}$$

$$\frac{}{\langle v \rangle \xrightarrow{\tau} v} \qquad \frac{e \xrightarrow{\tau} e'}{\langle e \rangle \xrightarrow{\tau} \langle e' \rangle} \qquad \frac{e \xrightarrow{\square} e'}{\langle e \rangle \xrightarrow{\tau} e'} \qquad \frac{e_1 \xrightarrow{E\{\square\ e_2\}} e_3}{e_1\ e_2 \xrightarrow{E} e_3}$$

$$\boxed{\frac{e_1 \xrightarrow{E\{v\ \square\}} e_2}{v\ e_1 \xrightarrow{E} e_2}} \qquad \frac{}{\mathcal{S}k.e \xrightarrow{E} \langle e[k \mapsto \lambda x.\langle E\{x\}\rangle]\rangle} \qquad \frac{}{e \xrightarrow{\tau} e}$$

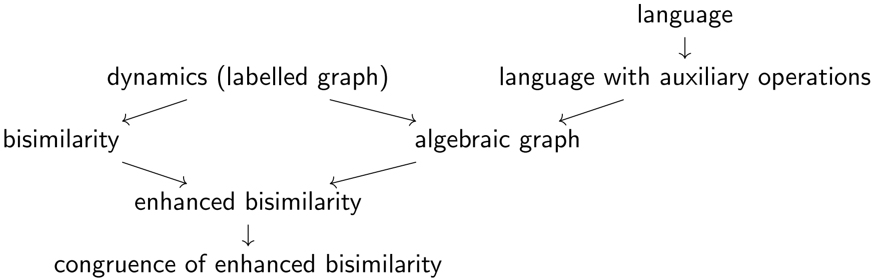
$$\frac{e_1 \xrightarrow{\tau} e_2 \xrightarrow{\alpha} e_3}{e_1 \xrightarrow{\alpha} e_3} \qquad \frac{e_1 \xrightarrow{\alpha} e_2 \xrightarrow{\tau} e_3}{e_1 \xrightarrow{\alpha} e_3}$$

A labelled graph

Example: deriving β

$$\dfrac{\dfrac{}{\lambda x.e \xrightarrow{v} e[x \mapsto v]}}{(\lambda x.e)\ v \xrightarrow{\tau} e[x \mapsto v]}$$

Recalling the plan



language

language with auxiliary operations

dynamics (labelled graph)

bisimilarity

algebraic graph

enhanced bisimilarity

congruence of enhanced bisimilarity

Transition contexts

A transition context $\mathbb{C} = (\mathbb{VT}, \mathbb{ET}, \mathbf{s}, \mathbf{t}, \mathbf{l})$ consists of:

- a category $\mathbb{VT}$ of vertex types and
- a category $\mathbb{ET}$ of edge types,

together with functors

$$\mathbf{s}, \mathbf{t} \colon \mathbb{ET} \to \mathbb{VT} \qquad \text{and} \qquad \mathbf{l} \colon \mathbb{ET} \to \mathbf{Fam}_f(\mathbb{VT}).$$

Concretely

For each edge type $\alpha \in \mathbb{ET}$, we have

- a source vertex type $\mathbf{s}(\alpha)$,
- a target vertex type $\mathbf{t}(\alpha)$, and
- a sequence $(\mathbf{l}^{\alpha}_{1}, \ldots, \mathbf{l}^{\alpha}_{n_\alpha})$ of label vertex types.

In λ-calculus with delimited continuations

We define $\mathbb{C}_\lambda$:

Vertex types Take $\mathbb{VT}_\lambda = 2 + 1$ for now, i.e.,

$$\begin{tikzcd} \mathbf{p} \arrow[r, "\iota"] & \mathbf{v} & \mathbf{c} \end{tikzcd}$$

(programs) (values) (contexts)

A presheaf $V \in \widehat{\mathbb{VT}_\lambda}$: a map $V(\mathbf{p}) \leftarrow V(\mathbf{v})$ and a set $V(\mathbf{c})$.

Edge types Take $\mathbb{ET}_\lambda = 3 \cong \{[\tau], [\mathbf{v}], [\mathbf{c}]\}$, where

$$\mathbf{s}(\alpha) = \mathbf{t}(\alpha) = \mathbf{p}$$

and names indicate corresponding labels:

$$\mathbf{l}[\tau] = () \qquad \mathbf{l}[\mathbf{v}] = (\mathbf{v}) \qquad \mathbf{l}[\mathbf{c}] = (\mathbf{c}).$$

With suitable notation:

$$[\alpha] : \mathbf{p} \xrightarrow{\alpha} \mathbf{p}, \text{ for all } \alpha \in \{\tau, \mathbf{v}, \mathbf{c}\}.$$

Graphs

We fix a transition context $\mathbb{C} = (\mathbb{VT}, \mathbb{ET}, \mathbf{s}, \mathbf{t}, \mathbf{l})$.

Definition ($\mathbb{C}$-graph)

- a vertex object $V \in \widehat{\mathbb{VT}}$,
- an edge object $E \in \widehat{\mathbb{ET}}$, and
- a border natural transformation $\partial \colon E \longrightarrow \Delta_{\mathbb{C}}(V)$,

where

$$\Delta_{\mathbb{C}}(V)(\alpha) := V(\mathbf{s}(\alpha)) \times \left(\prod_{i=1}^{n_\alpha} V(\mathbf{l}_i^\alpha) \right) \times V(\mathbf{t}(\alpha)).$$

Notation

$e \colon v \xrightarrow{\alpha(l_1,\ldots,l_{n_\alpha})} v'$ for $\partial_\alpha(e) = (v, (l_1, \ldots, l_{n_\alpha}), v')$.

In λ-calculus with delimited continuations

Remembering $\mathbb{C}_\lambda$:

[

$\mathbb{C}_\lambda$-graphs, concretely]

- A vertex object $V(\mathbf{p}) \leftarrow V(\mathbf{v}) \qquad V(\mathbf{c})$,
- an edge object $E(\tau) \qquad E[\mathbf{v}] \qquad E[\mathbf{c}]$ (just three sets),
- and border maps

$$\begin{aligned}
\partial_\tau &: E(\tau) \to V(\mathbf{p}) \times 1 \times V(\mathbf{p}) && \text{(no label)}\\
\partial_{[\mathbf{v}]} &: E[\mathbf{v}] \to V(\mathbf{p}) \times V(\mathbf{v}) \times V(\mathbf{p}) && \text{(label is a value)}\\
\partial_{[\mathbf{c}]} &: E[\mathbf{c}] \to V(\mathbf{p}) \times V(\mathbf{c}) \times V(\mathbf{p}) && \text{(label is a context).}
\end{aligned}$$

A category of $\mathbb{C}$-graphs

Trivial proposition

$\mathbb{C}$-graphs $\partial\colon E \to \Delta_{\mathbb{C}}(V)$ are the objects of the comma category $\mathbb{C}\text{-Gph} := \widehat{\mathbb{E}\mathbb{T}} \downarrow \Delta_{\mathbb{C}}$.

Isomorphic to a presheaf category by Carboni and Johnstone (1995):

- For $v \in \mathbb{V}\mathbb{T}$, we get $\mathbf{y}_v \in \mathbb{C}\text{-Gph}$: walking vertex of type v.
- For $\alpha \in \mathbb{E}\mathbb{T}$, we get $\mathbf{y}_\alpha \in \mathbb{C}\text{-Gph}$: walking edge of type α.
- Border morphisms

$$s_\alpha \colon \mathbf{y}_{\mathbf{s}(\alpha)} \to \mathbf{y}_\alpha \qquad t_\alpha \colon \mathbf{y}_{\mathbf{t}(\alpha)} \to \mathbf{y}_\alpha \qquad l_{\alpha,i} \colon \mathbf{y}_{\mathbf{l}_i^\alpha} \to \mathbf{y}_\alpha$$

(for all $i \in n_\alpha$, i omitted whin $n_\alpha = 1$).

($\mathbf{y}$ means Yoneda.)

Bisimulation and bisimilarity

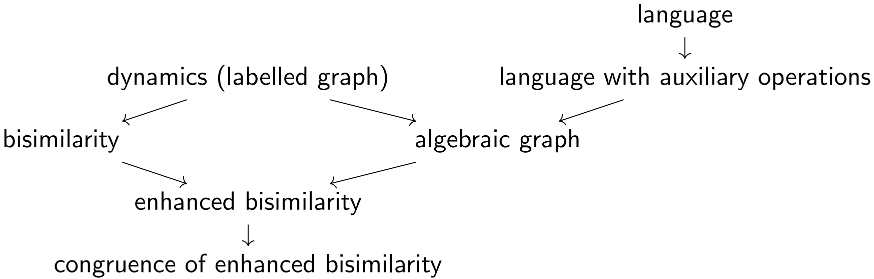
Definition

- Given $G = (V, E, \partial)$, a relation $R \hookrightarrow V^2$ is a simulation when,
  - for any transition $e \colon x \xrightarrow{\alpha(l_1,\ldots,l_{n_\alpha})} x'$ such that $x \; R \; y$,
  - there exists a transition f as in

$$\begin{tikzcd}
x \arrow[d, "e \colon \alpha(l_1,\ldots,l_{n_\alpha})"'] & R(\mathbf{s}(\alpha)) & y \arrow[d, dashed, "f \colon \alpha(l_1,\ldots,l_{n_\alpha})"] \\
x' & R(\mathbf{t}(\alpha)) & y'.
\end{tikzcd}$$

- A relation is a bisimulation when it is a simulation and so is its converse.
- Bisimilarity is the largest bisimulation relation.

Recalling the plan



language

language with auxiliary operations

dynamics (labelled graph)

bisimilarity

algebraic graph

enhanced bisimilarity

congruence of enhanced bisimilarity

Obvious categorical notion of grammar

$$\begin{array}{rcl} \text{Grammar} & = & \text{(finitary) endofunctor } F. \\ \text{Language} & = & \text{free monad } F^*. \end{array}$$

Base category for λ-calculus with delimited continuations

- Naive attempt: endofunctor on $\widehat{\mathbb{VT}_\lambda}$. No variable binding!
- Well-scoped approach: index over potential free (program) variables.
- $\rightsquigarrow$ Consider $\mathbb{VT}^+_\lambda$ such that $\widehat{\mathbb{VT}^+_\lambda} \simeq [\mathbf{Set}_f, \widehat{\mathbb{VT}_\lambda}]$ $\quad (\mathbb{VT}^+_\lambda = \mathbb{F}^{op} \times \mathbb{VT}_\lambda)$.

$\mathbb{VT}^+_\lambda$, concretely

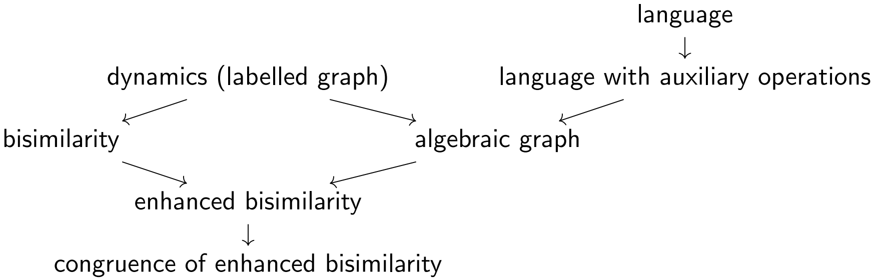
- Objects: $\mathbf{p}_n$, $\mathbf{v}_n$, $\mathbf{c}_n$. For any $V \in \widehat{\mathbb{VT}^+_\lambda}$,
  - $V(\mathbf{p}_n)$: set of **programs** with n potential free variables.
  - $V(\mathbf{v}_n)$: set of **values** with n potential free variables.
  - $V(\mathbf{c}_n)$: set of **contexts** with n potential free variables.
- Morphisms: composites of
  - renamings $\mathbf{p}_f : \mathbf{p}_n \to \mathbf{p}_m$, for $f : m \to n$ (similarly with $\mathbf{v}$, $\mathbf{c}$), and
  - $\iota_n : \mathbf{p}_n \to \mathbf{v}_n$.

Endofunctor for λ-calculus with delimited continuations

Endofunctor Σ_0 on $\widehat{\mathbb{V}\mathbb{T}^+_\lambda}$: for all $V \in \widehat{\mathbb{V}\mathbb{T}^+_\lambda}$ and $n \in \mathbb{F}$,

$$\begin{array}{rcccccccc}
\Sigma_0(V)(\mathbf{v}_n) & = & n & + & V(\mathbf{p}_{n+1}) & & & & \\
(\ v & ::= & x & | & \lambda x.e \ ) & & & & \\
\Sigma_0(V)(\mathbf{p}_n) & = & \Sigma_0(V)(\mathbf{v}_n) & + & V(\mathbf{p}_n)^2 & + & V(\mathbf{p}_{n+1}) & + & V(\mathbf{p}_n) \\
(\ e & ::= & v & | & e_1 \ e_2 & | & \mathcal{S}x.e & | & \langle e \rangle \ ) \\
\Sigma_0(V)(\mathbf{c}_n) & = & 1 & + & V(\mathbf{c}_n) \times V(\mathbf{p}_n) & + & V(\mathbf{v}_n) \times V(\mathbf{c}_n) & & \\
(\ E & ::= & \square & | & E \ e & | & v \ E \ ). & &
\end{array}$$

Recalling the plan



language

language with auxiliary operations

dynamics (labelled graph)

bisimilarity

algebraic graph

enhanced bisimilarity

congruence of enhanced bisimilarity

Need for auxiliary operations on terms

In order to abstractly account for rules like

$$\frac{e_1 \xrightarrow{E\{\nu\ \square\}} e_2}{\nu\ e_1 \xrightarrow{E} e_2}$$

need to account for context composition:

$$\begin{aligned}\square\{E'\} &= E' \\ (\nu\ E)\{E'\} &= \nu\ E\{E'\} \\ (E\ e')\{E'\} &= E\{E'\}\ e'.\end{aligned}$$

Not in the syntax, on the syntax.
Status of these auxiliary operations?

A theory of auxiliary operations

Augmented theory:

- reify auxiliary operations as part of the syntax, thus making them explicit (Abadi et al. 1990),
- mod out by recursive equations.

Calling

- Σ the endofunctor for basic syntax,
- Σ' the one for explicit auxiliary operations,

we get monad morphisms

$$\Sigma^* \longrightarrow (\Sigma + \Sigma')^* \twoheadrightarrow (\Sigma + \Sigma')^*_{/\sim}.$$

Observation

$$\Sigma^*(\varnothing) \cong (\Sigma + \Sigma')^*_{/\sim}(\varnothing)$$

Proof sketch: by the recursive equations, normal forms without explicit operations.

A theory of auxiliary operations

Admissible monad morphism

A $\alpha \colon S \to S^+$ such that $\alpha_\varnothing$ iso.

Here, slightly less general notion.

Starting point: recursive definitions have a distinguished argument.

Example: $(E, E') \mapsto E\{E'\}$

We have $\Sigma'(X) = X^2$, and the definition goes

$$\begin{aligned} \square\{E'\} &= E' \\ (\nu\ E)\{E'\} &= \nu\ E\{E'\} \\ (E\ e')\{E'\} &= E\{E'\}\ e'. \end{aligned}$$

In general we thus refine: $\Sigma'(X) = \Gamma(X, X)$.

In the example: $\Gamma(X,Y) = X \times Y$.

A theory of auxiliary operations

Definition

Enhanced syntax:

- $\Sigma \colon \widehat{\mathbb{VT}} \to \widehat{\mathbb{VT}}$ finitary,
- $\Gamma \colon \widehat{\mathbb{VT}}^2 \to \widehat{\mathbb{VT}}$ is cocontinuous-finitary, i.e.,
  - cocontinuous in its first argument and
  - finitary in its second argument,
- a distributive law $\delta \colon TS \to ST$, where
  - $S = \Sigma^*$,
  - $T = \Gamma_S^* =$ free monad on $X \mapsto \Gamma(X, S(X))^*$.

Proposition

By cocontinuity, we have $\Gamma_S(\varnothing) = \varnothing$, so $T(\varnothing) \cong \varnothing$, and hence

$$S(\varnothing) \cong S(T(\varnothing)).$$

A theory of auxiliary operations

Otherwise said, for any enhanced syntax (Σ, Γ, δ):

The initial Σ-algebra possesses a unique compatible Γ-algebra structure (which makes it an initial ST-algebra).

I.e.,

The forgetful functor $ST\text{-Alg} \to S\text{-Alg}$ creates the initial object.

Enhanced relations

Consider an ST-algebra V.

Definition

Enhanced relation: relation $R \to V^2$ in $\widehat{\mathbb{V}\mathbb{T}}$, such that $\Gamma(R, V) \subseteq R$.

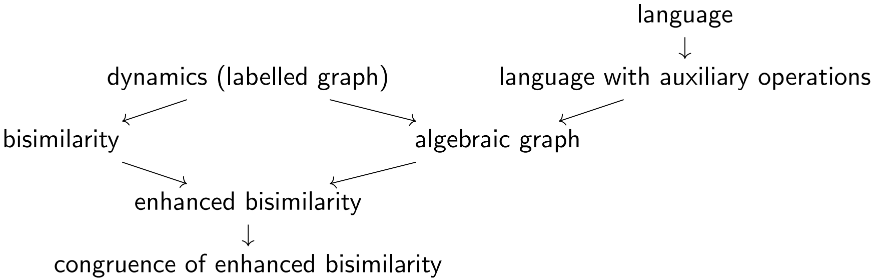
In λ-calculus with delimited continuations

- Relations $R_{\mathbf{v}}$, $R_{\mathbf{p}}$, and $R_{\mathbf{c}}$ on values, programs, and contexts.
- Such that

$$\frac{v \; R_{\mathbf{v}} \; v'}{v[\sigma] \; R_{\mathbf{v}} \; v'[\sigma]} \qquad \frac{E \; R_{\mathbf{c}} \; E'}{E\{e\} \; R_{\mathbf{c}} \; E'\{e\}} \qquad \ldots$$

Typical of applicative bisimilarity.

Recalling the plan



language

language with auxiliary operations

dynamics (labelled graph)

bisimilarity

algebraic graph

enhanced bisimilarity

congruence of enhanced bisimilarity

Algebraic graphs

Let us fix:

- a transition context $\mathbb{C} = (\mathbb{VT}, \mathbb{ET}, \mathbf{s}, \mathbf{t}, \mathbf{l})$,
- an enhanced syntax (Σ, Γ, δ) on $\widehat{\mathbb{VT}}$.

As before, let $S = \Sigma^*$ and $T = \Gamma^*_S$.

Definition

ST-graph:

- a $\mathbb{C}$-graph $\partial \colon E \to \Delta_{\mathbb{C}}(V)$,
- with ST-algebra structure on V.

Algebraic graphs

Definition

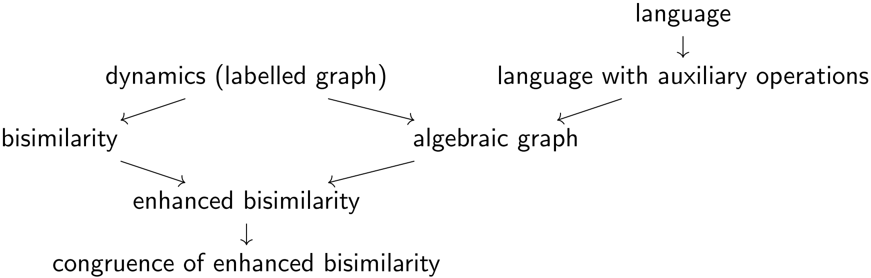
ST-graph:

- a $\mathbb{C}$-graph $\partial \colon E \to \Delta_{\mathbb{C}}(V)$,
- with ST-algebra structure on V.

A category of ST-graphs:

$$\begin{tikzcd}
ST\text{-Gph} \arrow[r, dashed, "\text{vertices}"] \arrow[d, dashed, "\text{forgetful}"'] \arrow[dr, phantom, "\lrcorner", very near start] & ST\text{-Alg} \arrow[d, "\text{forgetful}"] \\
\mathbb{C}\text{-Gph} \arrow[r, "\text{vertices}"'] & \widehat{\mathbb{V}\mathbb{T}}
\end{tikzcd}$$

Recalling the plan



language

language with auxiliary operations

dynamics (labelled graph)

bisimilarity

algebraic graph

enhanced bisimilarity

congruence of enhanced bisimilarity

Enhanced bisimilarity

Let $\partial\colon E \to \Delta_{\mathbb{C}}(V)$ be any ST-graph.

Definition

Enhanced bisimulation:

- enhanced relation on V,
- which is a bisimulation.

Enhanced bisimilarity: largest enhanced bisimulation.

Signatures

Let us fix a transition context $\mathbb{C} = (\mathbb{VT}, \mathbb{ET}, \mathbf{s}, \mathbf{t}, \mathbf{l})$.
Notions of

- Syntactic signature: omitted today, generates an enhanced syntax (Σ, Γ, δ).
- Dynamic signature: described now.

Signatures

So we assume given

- a transition context $\mathbb{C} = (\mathbb{VT}, \mathbb{ET}, \mathbf{s}, \mathbf{t}, \mathbf{l})$ and
- an enhanced syntax $\sigma = (\Sigma, \Gamma, \delta)$ on $\widehat{\mathbb{VT}}$.

Definition

Dynamic signature:

- Finitary endofunctor Σ_1 on $\mathit{ST}\text{-}\mathrm{Gph}$,
- preserving underlying ST-algebra.

$$\begin{tikzcd}
\mathit{ST}\text{-}\mathrm{Gph} \arrow[rr, "\Sigma_1"] \arrow[dr, "\text{forgetful}"'] & & \mathit{ST}\text{-}\mathrm{Gph} \arrow[dl, "\text{forgetful}"] \\
& \mathit{ST}\text{-}\mathrm{Alg} &
\end{tikzcd}$$

Example

For rule

$$\frac{e_1 \xrightarrow{E\{v\ \square\}} e_2}{v\ e_1 \xrightarrow{E} e_2}$$

we would take

$$\begin{array}{rcl}
\Sigma_1(G)[\mathbf{c}] & = & \sum_{\begin{array}{rcl} e_1, e_2 & \in & V_G(\mathbf{p}_0), \\ E & \in & V_G(\mathbf{c}_0), \\ v & \in & V_G(\mathbf{v}_0) \end{array}} \{r \in E_G[\mathbf{c}] \mid \mathbf{s}(r) = e_1 \ldots\} \\
 & = & \sum_{\begin{array}{rcl} e_1, e_2 & \in & V_G(\mathbf{p}_0), \\ E & \in & V_G(\mathbf{c}_0), \\ v & \in & V_G(\mathbf{v}_0) \end{array}} \{r \colon e_1 \xrightarrow{E\{v\ \square\}} e_2\}
\end{array}$$

$$\begin{array}{rcccccc}
\text{with } \partial(e_1, e_2, E, v, r) & = & (v\ e_1 & , & E & , & e_2) \quad . \\
 & \in & V(\mathbf{s}[\mathbf{c}]) & \times & V(\mathbf{l}_1^{[\mathbf{c}]}) & \times & V(\mathbf{t}[\mathbf{c}]) \\
 & \text{i.e.,} & V(\mathbf{p}_0) & \times & V(\mathbf{c}_0) & \times & V(\mathbf{p}_0)
\end{array}$$

Models of a dynamic signature

Definition

Vertical Σ_1-algebra G: algebra structure leaves vertices untouched.
$\rightsquigarrow$ category $\Sigma_1\text{-alg}_\mathrm{v}$.

Object of interest

Initial vertical Σ_1-algebra.

In examples, syntactic transition system given by operational semantics.

Remark

The canonical initial algebra, i.e., the colimit of

$$\varnothing \to \Sigma_1(\varnothing) \to \ldots \to \Sigma_1^n(\varnothing) \to \ldots$$

may be chosen vertical, in which case it is initial in $\Sigma_1\text{-alg}_\mathrm{v}$.

Main result

Theorem

For any

- *transition context* $\mathbb{C} = (\mathbb{VT}, \mathbb{ET}, \mathbf{s}, \mathbf{l}, \mathbf{t})$,
- *syntactic signature* $\mathbf{d}$ *generating enhanced syntax* $\sigma = (\Sigma, \Gamma, \delta)$,
- *dynamic signature* Σ_1 *satisfying a cellularity hypothesis,*

enhanced bisimilarity on the initial vertical Σ_1*-algebra is a congruence.*

A taste of cellularity

Exercise

The endofunctor for rule

$$\frac{e_1 \xrightarrow{E\{v\ \square\}} e_2}{v\ e_1 \xrightarrow{E} e_2}$$

is representable.

Remembering $\Sigma_1(G)[\mathbf{c}] = \sum_{\substack{e_1, e_2 \in V_G(\mathbf{p}_0), \\ E \in V_G(\mathbf{c}_0), \\ v \in V_G(\mathbf{v}_0)}} \{r \colon e_1 \xrightarrow{E\{v\ \square\}} e_2\}$, we have

$$\Sigma_1(G)[\mathbf{c}] \cong ST\text{-}\mathrm{Gph}(A, G),$$

for some suitable arity A.

A taste of cellularity

First, we have an adjunction $\begin{tikzcd} \mathbb{C}\text{-Gph} \arrow[r, bend left, "\mathscr{L}"] \arrow[r, phantom, "\perp"] & ST\text{-Gph} \arrow[l, bend left, "\mathscr{U}"] \end{tikzcd}$.

$\mathscr{L}(E \to \Delta_{\mathbb{C}}V) = (E \to \Delta_{\mathbb{C}}V \to \Delta_{\mathbb{C}}STV)$.

A taste of cellularity

We then take A to be the following pushout (ommitting $\mathbf{y}$ for readability),

$$\begin{tikzcd}
\mathscr{L}(\mathbf{p}_0 + \mathbf{c}_0) \arrow[r, "{\mathscr{L}[s_{[\mathbf{c}]}, l_{[\mathbf{c}]}]}"] \arrow[d, "{[e_1, E\{v\ \Box\}]}"'] & \mathscr{L}[\mathbf{c}] \arrow[d] \arrow[ddr, bend left, "r"] & \\
\mathscr{L}(\mathbf{v}_0 + \mathbf{p}_0 + \mathbf{c}_0) \arrow[r] \arrow[drr, bend right, "{[v, e_1, E]}"'] & A \arrow[ul, phantom, "\ulcorner", very near start] \arrow[dr, dashed] & \\
 & & G
\end{tikzcd}$$

where, calling v, e_1, and E the generating elements of $\mathbf{v}_0 + \mathbf{p}_0 + \mathbf{c}_0$,

$$\begin{array}{cl}
E\{v\ \Box\} \in ST(\mathbf{v}_0 + \mathbf{p}_0 + \mathbf{c}_0)(\mathbf{c}_0) & \\
\hline\hline
E\{v\ \Box\} \in \mathscr{U}\mathscr{L}(\mathbf{v}_0 + \mathbf{p}_0 + \mathbf{c}_0)(\mathbf{c}_0) & \\
\hline\hline
E\{v\ \Box\} : \mathbf{c}_0 \to \mathscr{U}\mathscr{L}(\mathbf{v}_0 + \mathbf{p}_0 + \mathbf{c}_0) & (\text{Yoneda}) \\
\hline\hline
E\{v\ \Box\} : \mathscr{L}(\mathbf{c}_0) \to \mathscr{L}(\mathbf{v}_0 + \mathbf{p}_0 + \mathbf{c}_0) & (\text{adjunction}).
\end{array}$$

Cellularity

Arrow-arity of a rule

“Characteristic” morphism: (arity of source + labels) → (dynamic arity).

For our example rule:

$$\frac{e_1 \xrightarrow{E\{v\ \square\}} e_2}{v\ e_1 \xrightarrow{E} e_2}$$

$$\begin{tikzcd} \mathscr{L}(\mathbf{p}_0+\mathbf{c}_0) \arrow[r, "{\mathscr{L}[s_{[\mathbf{c}]},l_{[\mathbf{c}]}]}"] \arrow[d, "{(e_1,E\{v\ \square\})}"'] & \mathscr{L}[\mathbf{c}] \arrow[d] \\ \mathscr{L}(\mathbf{v}_0+\mathbf{p}_0+\mathbf{c}_0) \arrow[r] & A \arrow[ul, phantom, "\ulcorner", very near start] \end{tikzcd}$$

Definition

Cellularity: all arrow-arities are cofibrations in the factorisation system generated by all $\mathscr{L}[s_{[\mathbf{c}]}, l_{[\mathbf{c}]}]$.

In the example: pushout of a generating cofibration.

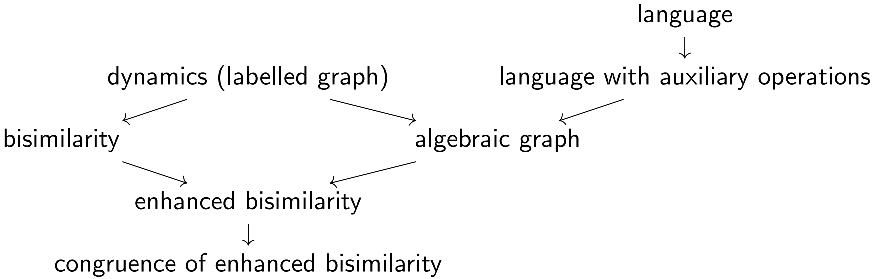
Conclusion

- Categorical framework for programming languages as (initial) algebraic graphs.
- Generic congruence result for applicative (= enhanced) bisimilarity based on cellularity.

Perspectives:

- Particularly subtle application of Howe's method by Lenglet and Schmitt (2015) still resists our abstraction efforts.
- Other variants of bisimilarity, relevant in the presence of effects.
- Apply same techniques to other areas of programming language theory (e.g., type safety).

Thanks for your attention



language

dynamics (labelled graph)

language with auxiliary operations

bisimilarity

algebraic graph

enhanced bisimilarity

congruence of enhanced bisimilarity

Any questions?