A generic congruence theorem for enhanced bisimilarity

Tom Hirschowitz (CNRS and Savoie Mont Blanc Univ.) and Ambroise Lafont (Univ. Cambridge)

Behavioural equivalences

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.

Notions of behavioural equivalence

- Determine when one program fragment may replace another.
- E.g., compilation, optimisations.

Typical choice:

Observational equivalence $P \approx Q$

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

Standard idea: find easier-to-establish $\sim$ such that $\quad P \sim Q \quad \Longrightarrow \quad P \approx Q.$

(Enhanced) bisimilarity

Typical choice for $\sim$: bisimilarity

How to prove $P \sim Q$?

Exhibit a bisimulation $\mathscr{R}$ such that $P \mathrel{\mathscr{R}} Q$ (which may be a small/tractable relation).

Bisimulation $\mathscr{R}$

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

Bisimilarity

Largest bisimulation.

In a higher-order setting (e.g., functional programming): enhanced bisimilarity

Enhanced relations

Closed under external operations, typically capture-avoiding substitution:

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

where σ: variables $\to$ programs.

Enhanced bisimilarity

Largest enhanced bisimulation relation.

In λ-calculus: Abramsky's applicative bisimilarity.

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

(Enhanced) bisimilarity is a congruence: $\quad \forall C, P \sim Q \Longrightarrow C\{P\} \sim C\{Q\}$.

Often proved using Howe's method (not explained here).

This work

A generic congruence theorem for enhanced bisimilarity, covering many existing adaptations of Howe's method.

Transition contexts

$\mathbb{C} = (\mathbb{VT}, \mathbb{ET}, \mathbf{s}, \mathbf{t}, \mathbf{l})$ where

- $\mathbb{VT}$ category of vertex types,
- $\mathbb{ET}$ category of edge types,
- $\mathbf{s}, \mathbf{t} \colon \mathbb{ET} \to \mathbb{VT}$ source and target functors,
- $\mathbf{l} \colon \mathbb{ET} \to \mathbf{Fam}_f(\mathbb{VT})$ label functor.

Example

$\mathbb{C}_0$: $\mathbb{VT} = \mathbb{ET} = 1$, $\mathbf{s} = \mathbf{t} = !$, $\mathbf{l}(\alpha) =$ empty family.

Labelled graphs as a comma cat

$\mathbb{C}$-graph: $G = (V, E, \partial)$ where

- $V \in \widehat{\mathbb{VT}}$ vertex object,
- $E \in \widehat{\mathbb{ET}}$ edge object,
- $\partial \colon E \to \Delta_{\mathbb{C}}(V)$ border morphism,

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

Notation

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

Example

For $\mathbb{C}_0$, we get usual graphs (a.k.a. quivers).

Form a category $\mathbb{C}\text{-Gph} := \widehat{\mathbb{ET}} \downarrow \Delta_{\mathbb{C}}$

$\cong$ presheaf cat (Carboni and Johnstone, 1995).

Examples

- $\mathbf{y}_v$ walking vertex of type v, for any $v \in \mathbb{VT}$.
- $\mathbf{y}_\alpha$ walking edge of type α, for any $\alpha \in \mathbb{ET}$.

Languages as initial algebras

Initial-algebra semantics:

Grammar $=$ (finitary) endofunctor Σ on $\widehat{\mathbb{VT}}$.

Language $=$ free monad $S := \Sigma^*$.

Real object of interest: $S(\emptyset)$.

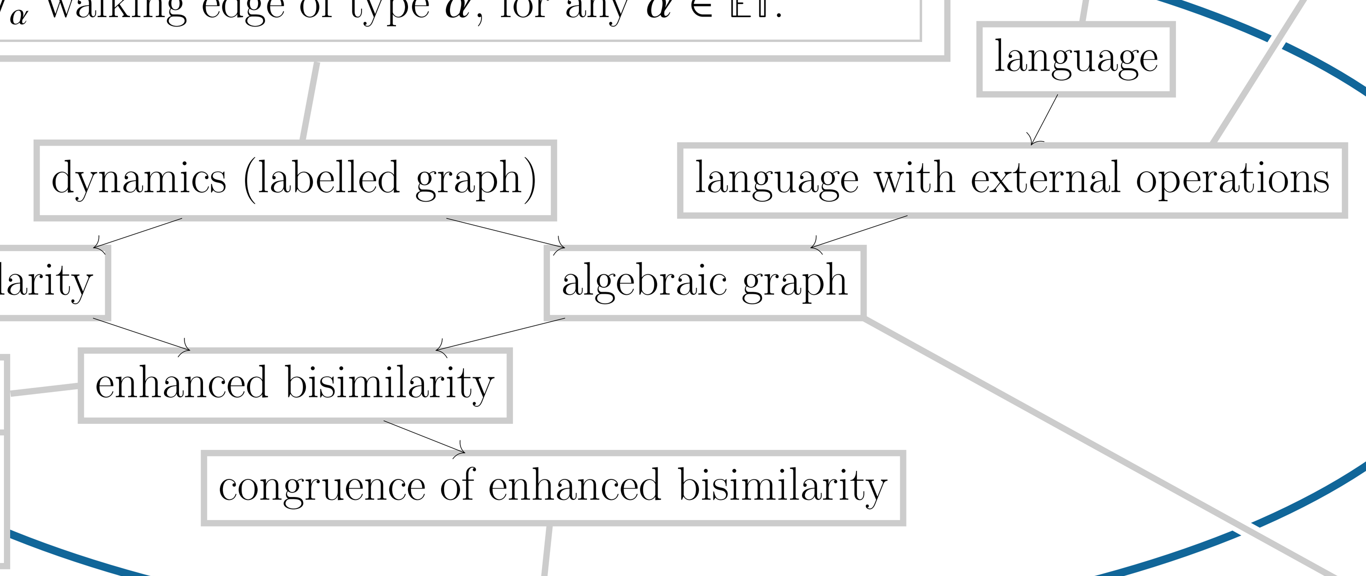
Initiality $\approx$ recursion principle.

Bisimilarity

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

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

  and conversely.
- Bisimilarity := largest bisimulation.



External operations via distributive laws

Motivation

- Usual (β) rule: $(\lambda x.e)\ v \to e[v]$.
- In the syntax: λ and application.
- On the syntax: substitution.

A theory of external operations

- Arity of external ops: $\Gamma \colon \widehat{\mathbb{VT}}^2 \to \widehat{\mathbb{VT}}$.
  (first argument = recursive argument).
- Let $T := (V \mapsto \Gamma(V, S(V)))^*$.
- Recursive equations given as distributive law $\delta \colon TS \to ST$.
- We impose Γ cocontinuous in first argument $\quad \Longrightarrow \quad T(\emptyset) \cong \emptyset \quad \Longrightarrow$

$S(\emptyset)$ is a T-algebra via
$T(S(\emptyset)) \xrightarrow{\delta_\emptyset} S(T(\emptyset)) \cong S(\emptyset)$.

Enhanced bisimilarity

Enhanced relation: $\mathscr{R}$ such that $\Gamma(\mathscr{R}, \mathscr{R}) \subseteq \mathscr{R}$.

Enhanced bisimilarity $\sim^\Gamma$: largest enhanced bisimulation.

Congruence of enhanced bisimilarity from factorisation system

Specifying dynamics

- Endofunctor Σ_1 on ST-Gph.
- Preserving vertex object.
- Operational semantics = initial (vertical) Σ_1-algebra.

Example: big-step β

$$\frac{e_1 \xrightarrow{\tau} e_3 \qquad e_3[e_2] \xrightarrow{\tau} e_4}{e_1\ e_2 \xrightarrow{\tau} e_4}$$

Arrow arities

If Σ_1 suitably familial: arrow arity of a rule.

Example, using left adjoint $\mathscr{L} \colon \mathbb{C}\text{-Gph} \to ST\text{-Gph}$

$$\begin{tikzcd} & \mathscr{L}(\mathbf{y}_0) \arrow[r, "\mathscr{L}(s)"] \arrow[d, "\text{``}e_3[e_2]\text{''}"'] & \mathscr{L}(\mathbf{y}_\tau) \arrow[d] \arrow[ddr, bend left, "\text{premise 2}"] & \\ \mathscr{L}(\mathbf{y}_0 + \mathbf{y}_0) \arrow[r, "\mathscr{L}(s+\mathbf{y}_0)"'] \arrow[drrr, bend right, "{[e_1,e_2]}"'] & \mathscr{L}(\mathbf{y}_\tau + \mathbf{y}_0) \arrow[r, "\mathscr{L}(s)"'] \arrow[drr, bend right, "{[\text{premise } 1, e_2]}"] & A \arrow[dr, dashed] & \\ & & & G \end{tikzcd}$$

Arrow arity = green composite.

Main result

Cellularity

Arrow arities are cofibrations in the factorisation system generated by [source,labels] morphisms.

Theorem

Cellularity $\Longrightarrow$ enhanced bisimilarity is a congruence, i.e., $S(\sim^\Gamma) \subseteq \sim^\Gamma$.

Proof: Howe's method (abstract).

Algebraic graphs

ST-graph:

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

$\rightsquigarrow$ category ST-Gph.

Key

Definition or reference	Emphasis
Structure or emphasis	Emphasis
Main contributions	

Conclusion

- Categorical framework for programming languages as (initial) algebraic graphs.
- Generic congruence result for applicative (= enhanced) bisimilarity.
- External operations via distributive laws.
- Congruence from factorisation system.

Perspectives

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