A Sound and Complete Logic for Algebraic Effects

Cristina Matache (joint work with Sam Staton)

University of Oxford

Question

Programming Language

- Higher-order functions
- Algebraic effects [Plotkin & Power]
- Recursive functions
- Continuation passing (CPS)

program → $M \vDash \phi$ ← formula

Logic

formula = program property

Contextual Equivalence

Main Theorem

=

Logical Equivalence

$\forall \phi. M \vDash \phi \iff N \vDash \phi.$

[Matache & Staton, FoSSaCS'19]

Motivation: [Simpson & Voorneveld, ESOP'18].

Question

program

Logic $M \vDash \phi$

formula = program property

Example

ϕ could be a Hoare logic assertion:

$$[S[l_0 := n]](-)[S[l_0 := n,\ l_1 := n+1]]$$

But we have higher-order functions so instead

$$\phi = \Big(() \mapsto [S[l_0 := n,\ l_1 := n+1]{\downarrow}]\Big) \mapsto [S[l_0 := n]{\downarrow}]$$

Outline

Program equivalence

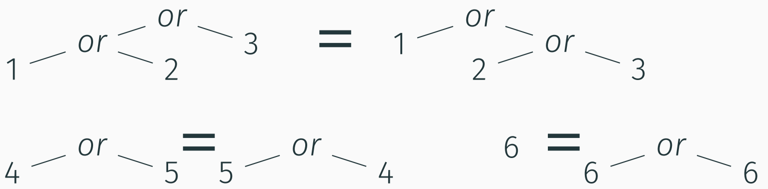
Establishes when two programs have the same behaviour.

! Higher-order functions make it hard.
Effects make it hard.

Example

$or(x, y)$ = nondeterministically choose x or y

Want:



Contextual equivalence

$M \equiv_{ctx} N$ iff M and N are *observably* the same

Let: $\mathfrak{P}$ = set of observations

$M \equiv_{ctx} N$ iff $\forall \mathcal{C}.\ \forall P \in \mathfrak{P}.$ $\mathcal{C}[M] \in P \iff \mathcal{C}[N] \in P$

Example

For untyped λ-calculus P = termination
and with nondeterminism: $\Diamond$ = may terminate,
$\Box$ = must terminate

Behavioural Logic

program

Logic $M \vDash \phi$ formula = program property

$\vDash$ describes the behaviour of programs

Example

$or(or(1,2), 3) \vDash \Diamond\{3\}$ may return 3

$\vDash \Box\{1,2,3\}$

always returns one of $\{1,2,3\}$

Behavioural Logic

program

Logic $M \vDash \phi$

formula = program property

$\vDash$ describes the behaviour of programs

Logical Equivalence

$$M \equiv_{\mathrm{log}} N \quad \text{iff} \quad \forall \phi.\, M \vDash \phi \iff N \vDash \phi.$$

Want: $(\equiv_{log}) = (\equiv_{\mathrm{ctx}})$

Continuation-Passing Style (CPS)

Type $A \to B$ becomes $A \to (B \to \mathsf{R}) \to \mathsf{R}$

R = fixed return type

Example

$$\texttt{add-cps} = \lambda(n{:}\texttt{nat}, m{:}\texttt{nat}, k{:}\texttt{nat}{\to}\mathsf{R}).\ k\ (n + m)$$

$$: (\texttt{nat}, \texttt{nat}, \texttt{nat}{\to}\mathsf{R}){\to}\mathsf{R}$$

Outline

ECPS Calculus

- Types: $A, A_i \coloneqq (A_1, \ldots, A_n)\to\mathsf{R} \mid \texttt{nat} \qquad (n \geq 0)$
- Values vs. computations

$$\frac{\Gamma, \overrightarrow{x : A} \vdash^{\mathfrak{c}} t : \mathsf{R}}{\Gamma \vdash^{\mathfrak{v}} \lambda(\overrightarrow{x{:}A}).t : (\overrightarrow{A})\to\mathsf{R}} \qquad \frac{\Gamma \vdash^{\mathfrak{v}} v : (\overrightarrow{A})\to\mathsf{R} \quad (\Gamma \vdash^{\mathfrak{v}} w_i : A_i)_i}{\Gamma \vdash^{\mathfrak{c}} v\,(\overrightarrow{w}) : \mathsf{R}}$$

ECPS Calculus

- Types: $A, A_i ::= (A_1, \ldots, A_n)\to\mathsf{R} \mid \mathtt{nat} \qquad (n \geq 0)$
- Values vs. computations

$$\frac{\Gamma, \overrightarrow{x : A} \vdash^{\mathfrak{c}} t : \mathsf{R}}{\Gamma \vdash^{\mathfrak{v}} \lambda(\overrightarrow{x{:}A}).t : (\overrightarrow{A})\to\mathsf{R}} \qquad \frac{\Gamma \vdash^{\mathfrak{v}} v : (\overrightarrow{A})\to\mathsf{R} \quad (\Gamma \vdash^{\mathfrak{v}} w_i : A_i)_i}{\Gamma \vdash^{\mathfrak{c}} v\,(\overrightarrow{w}) : \mathsf{R}}$$

- Effect operations.

$$\frac{\sigma \in \Sigma \quad (\Gamma \vdash^{\mathfrak{v}}_{\Sigma} v_i : \mathtt{nat})_i \quad (\Gamma \vdash^{\mathfrak{v}}_{\Sigma} k_j : (\mathtt{nat}, \ldots, \mathtt{nat})\to\mathsf{R})_j}{\Gamma \vdash^{\mathfrak{c}}_{\Sigma} \sigma(\overrightarrow{v_i}, \overrightarrow{k_j}) : \mathsf{R}}$$

ECPS Calculus

- Types: $A, A_i \coloneqq (A_1, \ldots, A_n) \to \mathsf{R} \mid \mathtt{nat} \qquad (n \geq 0)$
- Values vs. computations

$$\frac{\Gamma, \overrightarrow{x : A} \vdash^{\mathfrak{c}} t : \mathsf{R}}{\Gamma \vdash^{\mathfrak{v}} \lambda(\overrightarrow{x{:}A}).t : (\overrightarrow{A}) \to \mathsf{R}} \qquad \frac{\Gamma \vdash^{\mathfrak{v}} v : (\overrightarrow{A}) \to \mathsf{R} \quad (\Gamma \vdash^{\mathfrak{v}} w_i : A_i)_i}{\Gamma \vdash^{\mathfrak{c}} v\,(\overrightarrow{w}) : \mathsf{R}}$$

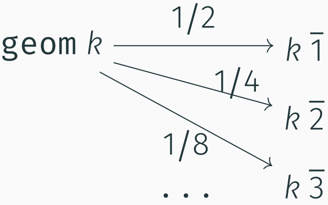
- Effect operations.

$$\frac{\sigma \in \Sigma \quad (\Gamma \vdash^{\mathfrak{v}}_{\Sigma} v_i : \mathtt{nat})_i \quad (\Gamma \vdash^{\mathfrak{v}}_{\Sigma} k_j : (\mathtt{nat}, \ldots, \mathtt{nat}) \to \mathsf{R})_j}{\Gamma \vdash^{\mathfrak{c}}_{\Sigma} \sigma(\overrightarrow{v_i}, \overrightarrow{k_j}) : \mathsf{R}}$$

- Recursion

Examples of Effect Signatures

Probability: $p\text{-}or : ((\,)\to\mathsf{R}, (\,)\to\mathsf{R})\to\mathsf{R}$ like $\oplus : \mathsf{R}\times\mathsf{R}\to\mathsf{R}$



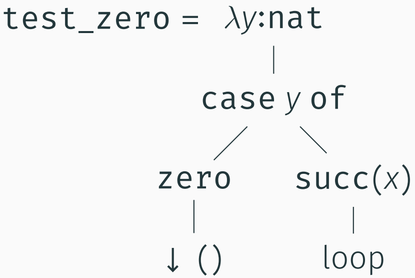
- `geom` computes the geometric distribution: it passes $\overline{n}$ to the continuation k with probability $\frac{1}{2^n}$.

$$\begin{aligned}
\texttt{geom} = \lambda k{:}\texttt{nat}\to\mathsf{R}.\\
&\big(\texttt{rec}\ f.\ \lambda(n{:}\texttt{nat}, k'{:}\texttt{nat}\to\mathsf{R}).\\
&\qquad p\text{-}or(\lambda().k'\ n,\ \lambda().f\,(\texttt{succ}(n), k'))\\
&\big)\ (\overline{1},\ k).
\end{aligned}$$

Examples of Effect Signatures

Success: $\Sigma = \{\downarrow : () \to R\}$

Abstract syntax tree



Computation tree

$[\![-]\!] : (\vdash_\Sigma R) \longrightarrow \mathit{Trees}_\Sigma$

[Plotkin and Power, FoSSaCS'01]
[Johann et al. LICS'10]

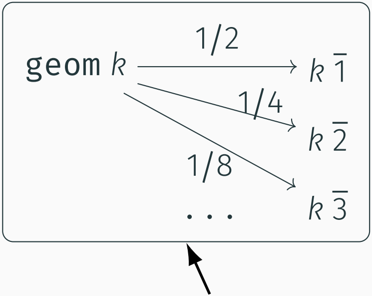
$[\![\texttt{test\_zero}\ \overline{0}]\!] = \downarrow$

$[\![\texttt{test\_zero}\ \overline{1}]\!] = \bot$

- `test_zero`: continuation that succeeds only on input $\overline{0}$.

Examples of Effect Signatures

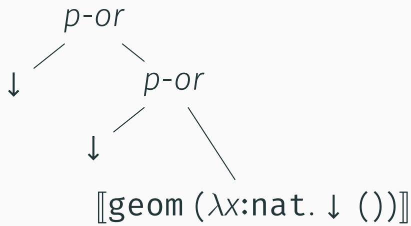
Probability: $\Sigma = \{p\text{-}or : (()\to R, ()\to R)\to R,\ \downarrow : ()\to R\}$



geometric distribution

Computation tree

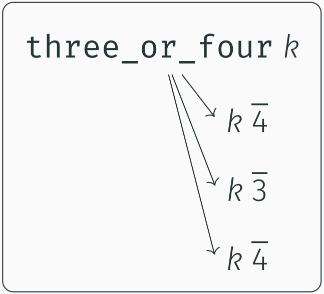
$[\![\texttt{geom}\ (\lambda x{:}\texttt{nat}.\downarrow ())]\!] =$



Examples of Effect Signatures

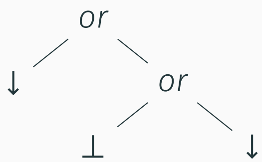
Nondeterminism: $\Sigma = \{or : (()\to R, ()\to R)\to R,\ \downarrow : ()\to R\}$

Abstract syntax tree



Computation tree

$[\![$`three_or_four` $(\lambda x{:}\texttt{nat}.$ `if` $x = 4$ `then` $\downarrow$ `else` $loop)]\!] =$



- `three_or_four` returns either $\overline{3}$ or $\overline{4}$ to continuation.

Examples of Effect Signatures

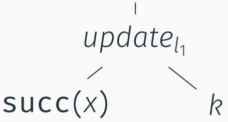
Global Store:

$$\Sigma = \{lookup_l : (\mathtt{nat}{\to}\mathrm{R}){\to}\mathrm{R},\ update_l : (\mathtt{nat}, (){\to}\mathrm{R}){\to}\mathrm{R} \mid l \in \mathbb{L}\}$$
$$\cup \{\downarrow : (){\to}\mathrm{R}\}$$

Abstract syntax tree

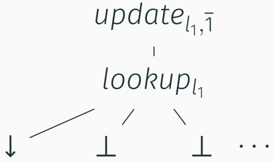
$\mathtt{suc\_update} =$

$\lambda(x{:}\mathtt{nat}, k{:}(){\to}\mathrm{R})$



Computation tree

$[\![\mathtt{suc\_update}$
$(\overline{0}, \lambda().lookup_{l_1}(\mathtt{test\_zero}))]\!] =$



- $\mathtt{suc\_update}$: write successor of the input to location l_1.

Other examples: I/O.

Observations

Observation P = Set of trees

- Σ = Success: $\Downarrow = \{\downarrow\}$
- Σ = Probability: for $q \in \mathbb{Q}$, $0 \leq q < 1$

  $\mathsf{P}_{>q}$ = {trees that succeed with probability $> q$}
- Σ = Nondeterminism:

  $\Diamond$ = {trees with at least one $\downarrow$ leaf}

  $\Box$ = {trees of finite height with only $\downarrow$ leaves}
- Σ = Global store: $S \in \mathbb{L} \longrightarrow \mathbb{N}$

  $[S\downarrow]$ = {trees that succeed when started in state S}

Observations

Observation P = Set of trees

- Σ = Success: $\Downarrow = \{\downarrow\}$

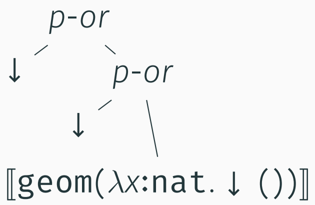
$$[\![\texttt{test\_zero}\,\overline{0}]\!] = \downarrow \in \Downarrow$$

Observations

Observation P = Set of trees

- Σ = Probability:
  $\mathsf{P}_{>q}$ = {trees that succeed with probability $> q$},
  $q \in \mathbb{Q},\ 0 \leq q < 1$

$[\![\texttt{geom}\ (\lambda x{:}\texttt{nat}.\downarrow ())]\!] =$



$\in\ \mathsf{P}_{>0.9}$

Observations

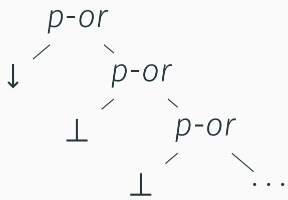
Observation P = Set of trees

- Σ = Probability:

  $\mathsf{P}_{>q}$ = {trees that succeed with probability $> q$},

  $q \in \mathbb{Q}$, $0 \le q < 1$

$[\![$`geom` $(\lambda x$:`nat`. `if` $x = 1$ `then` $\downarrow$() `else` *loop*$)]\!]$ =



$\notin \mathsf{P}_{>0.5}$

Observations

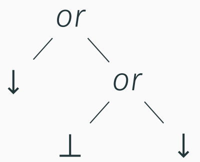
Observation P = Set of trees

- Σ = Nondeterminism:

  $\Diamond$ = {trees with at least one $\downarrow$ leaf}

  $\Box$ = {trees of finite height with only $\downarrow$ leaves}

$[\![$`three_or_four` $(\lambda x$:`nat`. `if` $x = 4$ `then` $\downarrow$ `else` *loop*$)]\!] =$



$\in \Diamond$

$\notin \Box$

Observations

Observation P = Set of trees

- Σ = Global store: $S \in \mathbb{L} \longrightarrow \mathbb{N}$
  $[S\downarrow]$ = {trees that succeed when started in state S}

$[\![$`suc_update`
$(\overline{0},\ \lambda().lookup_{l_1}($`test_zero`$))]\!]$ =

$update_{l_1,\overline{1}}$
|
$lookup_{l_1}$
/ / \
$\downarrow$ $\perp$ $\perp$ $\cdots$

$\notin [S\{l_1 := 0\}\downarrow]$

Summary

2 Programming Calculus

- Σ = signature of effect operations
- P = an observation = a set of trees
- $\mathfrak{P}$ = set of observations = set of sets of trees

Outline

Logic

- Parametrized by Σ and $\mathfrak{P}$ (set of observations).
- Value formulas

$$\phi \coloneqq \{n\} \mid (\phi_1, \ldots, \phi_n) \mapsto P \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \neg\phi$$

Each value formula has a type:

$$\frac{}{\{n\} : \mathtt{nat}}\, n \in \mathbb{N} \qquad \frac{\phi_1 : A_1 \ldots \phi_n : A_n \qquad P \in \mathfrak{P}}{(\phi_1, \ldots, \phi_n) \mapsto P : (A_1, \ldots, A_n) \to \mathrm{R}}$$

- Computation formulas = Observations from $\mathfrak{P}$

Logic

- Value formulas:

$$\frac{}{\{n\} : \mathtt{nat}}\, n \in \mathbb{N} \qquad \frac{\phi_1 : A_1 \ldots \phi_n : A_n \qquad P \in \mathfrak{P}}{(\phi_1, \ldots, \phi_n) \mapsto P : (A_1, \ldots, A_n) \to \mathsf{R}} \qquad \vee, \wedge, \neg$$

- Computation formulas: $P \in \mathfrak{P}$

Examples of formulas

$$\phi_1 = (\{1\} \vee \{2\} \vee \{3\}) \mapsto \square \qquad : \mathtt{nat} \to \mathsf{R}$$

$$\phi_2 = \neg\Big((\{1\}) \mapsto \Diamond\Big) \wedge \neg\Big((\{2\}) \mapsto \square\Big) \qquad : \mathtt{nat} \to \mathsf{R}$$

$$\phi_3 = \Big(() \mapsto [S[l_0 := n,\ l_1 := n + 1]{\downarrow}]\Big) \mapsto [S[l_0 := n]{\downarrow}]$$

$$: (() \to \mathsf{R}) \to \mathsf{R}$$

Logic

- Value formulas:

$$\frac{}{\{n\} : \mathtt{nat}}\, n \in \mathbb{N} \qquad \frac{\phi_1 : A_1 \ldots \phi_n : A_n \qquad P \in \mathfrak{P}}{(\phi_1, \ldots, \phi_n) \mapsto P : (A_1, \ldots, A_n) \to \mathsf{R}} \qquad \vee, \wedge, \neg$$

- Computation formulas: $P \in \mathfrak{P}$
- Satisfaction:

$$\begin{aligned}
v \vDash \{n\} &\iff v = \overline{n} \\
v \vDash (\phi_1, \ldots, \phi_n) \mapsto P &\iff \text{for all } w_1, \ldots, w_n \text{ such that} \\
&\qquad \forall i.\ w_i \vDash \phi_i \text{ then } v(w_1, \ldots, w_n) \vDash P \\
v \vDash \neg\phi &\iff \text{it is false that } v \vDash \phi \\
t \vDash P &\iff [\![t]\!] \in P. \\
&\ldots
\end{aligned}$$

Examples of Logical Satisfaction

given input $\overline{0}$, the function succeeds

$$\texttt{test\_zero} \vDash \overbrace{\{0\} \mapsto \Downarrow} : \texttt{nat} \to \texttt{R}$$

$\Downarrow = \{\downarrow\}$

continuation that tests whether the input is $\overline{0}$

Examples of Logical Satisfaction

geometric distribution

$$\texttt{geom} \vDash \left((\vee_{n \gneq 1}\{n\}) \mapsto \mathsf{P}_{>q} \right) \mapsto \mathsf{P}_{>q/2}$$

$$: (\texttt{nat} \rightarrow \mathsf{R}) \rightarrow \mathsf{R}$$

- given a continuation that succeeds with probability $> q$ for all inputs $n > 1$, the function succeeds with probability $> \frac{q}{2}$.

Examples of Logical Satisfaction

chooses nondeterministically between $\overline{3}$ or $\overline{4}$

$$\texttt{three\_or\_four} \models \left((\{3\} \mapsto \diamond) \mapsto \diamond\right) \wedge \left((\{4\} \mapsto \diamond) \mapsto \diamond\right) \wedge$$
$$\left((\{3\} \mapsto \square \wedge \{4\} \mapsto \square) \mapsto \square\right)$$
$$: (\texttt{nat} \to \texttt{R}) \to \texttt{R}$$

- Function may pass to the continuation only $\overline{3}$ or $\overline{4}$.

Examples of Logical Satisfaction

writes $n + 1$ to location l_1

`suc_update` $\models$

$$\bigwedge_{S \in State} \bigwedge_{n \in \mathbb{N}} \Big(\{n\}, \quad () \mapsto \big[S\{l_0 := n, l_1 := n + 1\}\!\downarrow \big] \Big) \mapsto \big[S\{l_0 := n\}\!\downarrow \big]$$

`: (nat, ()→R)→R`

- Given argument $\overline{n}$ and a continuation that succeeds when started in state $[S\{l_0 := n, l_1 := n + 1\}\!\downarrow]$, the function succeeds when started in state $[S\{l_0 := n\}\!\downarrow]$.

Outline

Main Theorem

$\mathfrak{P}$ = set of sets of trees

Logical Equivalence

$$\forall \phi.\, M \vDash \phi \iff N \vDash \phi.$$

Contextual Equivalence

$$\forall \mathcal{C}.\, \forall P \in \mathfrak{P}.\, [\![\mathcal{C}[M]]\!] \in P \iff [\![\mathcal{C}[N]]\!] \in P.$$

Main Theorem

$\mathfrak{P}$ consistent
$\mathfrak{P}$ decomposable
$P \in \mathfrak{P}$ Scott-open

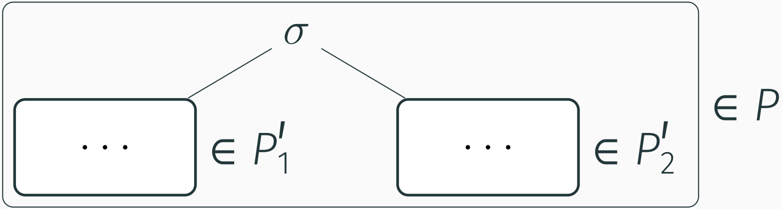
$\implies$

Logical Equivalence
=
Contextual Equivalence

Decomposability

$\mathfrak{P}$ is decomposable if for any $P \in \mathfrak{P}$, and for any $tr \in P$:

$$\forall \sigma \in \Sigma.\ \Big(tr = \sigma_{\vec{v}}(\overrightarrow{tr'}) \implies \exists \overrightarrow{P'} \in \mathfrak{P} \cup \{Trees_\Sigma\}.\ \overrightarrow{tr'} \in \overrightarrow{P'} \text{ and } \forall \overrightarrow{p'} \in \overrightarrow{P'}.\ \sigma_{\vec{v}}(\overrightarrow{p'}) \in P\Big).$$

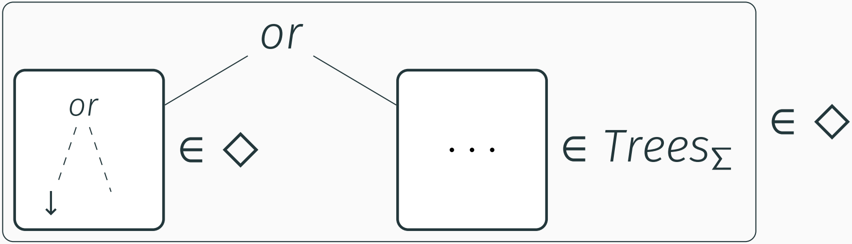


[Johann et.al. LICS'10], [Simpson & Voorneveld, ESOP'18]

Decomposability

$\mathfrak{P}$ is decomposable if for any $P \in \mathfrak{P}$, and for any $tr \in P$:

$$\forall \sigma \in \Sigma. \Big(tr = \sigma_{\vec{v}}(\overrightarrow{tr'}) \implies \exists \overrightarrow{P'} \in \mathfrak{P} \cup \{Trees_\Sigma\}. \; \overrightarrow{tr'} \in \overrightarrow{P'} \text{ and } \forall \overrightarrow{p'} \in \overrightarrow{P'}. \; \sigma_{\vec{v}}(\overrightarrow{p'}) \in P\Big).$$



Proof Sketch

1) Applicative bisimilarity compatible
by Howe's method [Howe, Inf. Comput.'96],
using Scott-openness and decomposability

Logical Equivalence	$\overset{=}{2)}$	Applicative Bisimilarity	$\overset{=}{3)}$	Contextual Equivalence

[Effectful Applicative Bisimilarity, Dal Lago et al. LICS'17]

Applicative Bisimilarity

Applicative simulation

A collection of relations $\mathcal{R}^{\mathfrak{v}}_A \subseteq (\vdash_\Sigma A)^2$ for each type A and $\mathcal{R}^{\mathfrak{c}} \subseteq (\vdash_\Sigma \mathrm{R})^2$ is an applicative $\mathfrak{P}$-*simulation* if:

- $v \, \mathcal{R}^{\mathfrak{v}}_{\mathtt{nat}} \, w \implies v = w$.
- $s \, \mathcal{R}^{\mathfrak{c}} \, t \implies \forall P \in \mathfrak{P}. \, (\llbracket s \rrbracket \in P \implies \llbracket t \rrbracket \in P)$.
- $v \, \mathcal{R}^{\mathfrak{v}}_{(\vec{A}) \to \mathrm{R}} \, u \implies \forall (\vdash_\Sigma w_i : A_i)_i. \, v(\vec{w_i}) \, \mathcal{R}^{\mathfrak{c}} \, u(\vec{w_i})$.

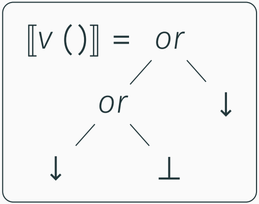
Applicative Bisimilarity

Is the *greatest* symmetric $\mathfrak{P}$-simulation.

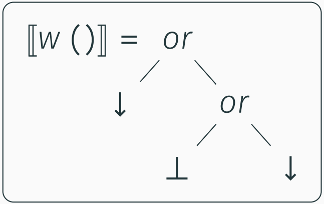
Applicative Bisimilarity — Example

$$v = \lambda().\,or\big(or(\lambda().\downarrow,\ loop),\ \lambda().\downarrow\big) \quad : ()\to\mathsf{R}$$
$$w = \lambda().\,or\big(\lambda().\downarrow,\ or(loop,\ \lambda().\downarrow)\big)$$

Prove: v bisimilar to w



$\in \Diamond$ $\notin \Box$ $\iff$



$\in \Diamond$ $\notin \Box$

- Choose $\mathcal{R}^{\mathfrak{v}}_{()\to\mathsf{R}} = \{(v, w),\ (w, v)\}$ and
  $\mathcal{R}^{\mathfrak{c}} = \{(v\,(),\ w\,()),\ (w\,(),\ v\,())\}$.
  $\mathcal{R}$ is a bisimulation $\Longrightarrow$ $\mathcal{R}$ included in bisimilarity.

Proof Sketch

1) Applicative bisimilarity compatible
by Howe's method [Howe, Inf. Comput.'96],
using Scott-openness and decomposability

Logical Equivalence $\overset{}{\underset{2)}{=}}$ Applicative Bisimilarity $\underset{3)}{=}$ Contextual Equivalence

via a simpler
equi-expressive logic,
using 1)

[Simpson and Voorneveld, ESOP'18]

$$\frac{\vdash_\Sigma w_1 : A_1 \ldots \vdash_\Sigma w_n : A_n \quad P \in \mathfrak{P}}{(w_1, \ldots, w_n) \mapsto P : (A_1, \ldots, A_n) \to \mathrm{R}}$$

Proof Sketch

1) Applicative bisimilarity compatible
by Howe's method [Howe, Inf. Comput.'96],
using Scott-openness and decomposability

Logical Equivalence $\underset{2)}{=}$ Applicative Bisimilarity $\underset{3)}{=}$ Contextual Equivalence

using consistency,
Scott-openness and 1)
N.B. $\supseteq$ interesting

Proof Sketch

Applicative simulation

A collection of relations $\mathcal{R}^{\mathfrak{v}}_A \subseteq (\vdash_\Sigma A)^2$ for each type A and $\mathcal{R}^{\mathfrak{c}} \subseteq (\vdash_\Sigma \mathsf{R})^2$ is an applicative $\mathfrak{P}$-*simulation* if:

- ...
- $s \, \mathcal{R}^{\mathfrak{c}} \, t \implies \forall P \in \mathfrak{P}. \, ([\![s]\!] \in P \implies [\![t]\!] \in P)$.
- ...

Logical Equivalence $\overset{}{\underset{2)}{=}}$ Applicative Bisimilarity $\underset{3)}{=}$ Contextual Equivalence

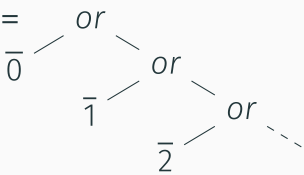
Comparison with Previous Work

In [Simpson & Voorneveld, ESOP'18]:

Logical Equivalence = Applicative Bisimilarity $\subset$ Contextual Equivalence

Example (in direct style)

$[\![\texttt{?nat}]\!] =$



$$M = \textbf{return } \lambda().\texttt{?nat}$$

$$N = \textbf{let } y \Rightarrow \texttt{?nat} \textbf{ in } \big(\textbf{return } \lambda().\texttt{min(?nat, y)}\big)$$

Comparison with Previous Work

In [Simpson & Voorneveld, ESOP'18]:

Logical Equivalence $=$ Applicative Bisimilarity $\subset$ Contextual Equivalence

Example (in direct style)

$$M = \mathbf{return}\ \lambda().\texttt{?nat}$$

$$N = \mathbf{let}\ y \Rightarrow \texttt{?nat}\ \mathbf{in}\ (\mathbf{return}\ \lambda().\texttt{min(?nat, y)})$$

$$M \models \Phi = \Diamond(() \mapsto \bigwedge_{n\in\mathbb{N}} \Diamond\{n\}) \quad \text{but} \quad N \not\models \Phi$$

$$M \not\equiv_{log} N \quad \text{but} \quad M \equiv_{ctx} N$$

$$[M]^{cps} \equiv^{cps}_{log/ctx} [N]^{cps}$$

Summary

- ECPS calculus with
  - algebraic effects
  - recursive functions
- Effects: probability, global store, I/O, nondeterminism

Main Theorem [Matache & Staton, FoSSaCS'19]

$\mathfrak{P}$ consistent $\mathfrak{P}$ decomposable $P \in \mathfrak{P}$ Scott-open	$\Longrightarrow$	Logical Equivalence = Contextual Equivalence

See also [Dal Lago et al. ICTCS/CILC'17]

Summary

- Haven't done yet:
  - local state
  - combining effects
  - game characterization of logical satisfaction

Main Theorem [Matache & Staton, FoSSaCS'19]

$\mathfrak{P}$ consistent
$\mathfrak{P}$ decomposable
$P \in \mathfrak{P}$ Scott-open

$\Longrightarrow$

Logical Equivalence
=
Contextual Equivalence