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)

Logic

Main Theorem

Contextual Equivalence

Logical Equivalence

[Matache & Staton, FoSSaCS’19]

Motivation: [Simpson & Voorneveld, ESOP’18].
Logic \[ M \models \phi \] program property

Example

\( \phi \) 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

\[
\phi = \left( () \mapsto [S[l_0 := n, l_1 := n + 1] \downarrow] \right) \mapsto [S[l_0 := n] \downarrow]
\]
Outline

1. Introduction: Program Equivalence and CPS
2. Programming Calculus
3. Logic
4. Main Theorem
Establishes when two programs have the same behaviour.

Useful for:

- checking whether a program satisfies a specification
- meaning of program = its equivalence class.
Higher-order functions make it hard.

Example

\[
\begin{align*}
\text{one} &= \lambda f. \lambda y. (f \, y) \\
\text{two} &= \lambda f. \lambda y. f \, (f \, y) \\
\text{two}' &= \lambda f. \lambda y. f \, (\text{one} \, f \, y)
\end{align*}
\]

Want: \( \text{two} \equiv \text{two}' \)
Program equivalence

Effects make it hard.

Example

\[ \text{or}(x, y) = \text{nondeterministically choose } x \text{ or } y \]

Want:

\[
\begin{align*}
\text{or} & \quad \text{or} & \quad 3 \\
1 & \quad 2 & \quad \text{or} \\
\end{align*}
\]

\[
\begin{align*}
1 & \quad \text{or} & \quad 3 \\
2 & \quad \text{or} & \quad 3 \\
\end{align*}
\]

\[
\begin{align*}
\text{or} & \quad \text{or} \\
4 & \quad 5 \\
5 & \quad 5 \\
4 & \quad 4 \\
\end{align*}
\]

\[
\begin{align*}
6 & \quad \text{or} & \quad 6 \\
6 & \quad 6 \\
6 & \quad 6 \\
\end{align*}
\]
Contextual equivalence

\[ M \equiv_{\text{ctx}} N \iff M \text{ and } N \text{ are } \text{observably} \text{ the same} \]

Let: \( \Psi = \text{set of observations} \)

\[ M \equiv_{\text{ctx}} N \iff \forall C. \forall P \in \Psi. \ C[M] \in P \iff C[N] \in P \]

Example

For untyped \( \lambda \)-calculus \( P = \) termination and with nondeterminism: \( \Diamond = \) may terminate, \( \Box = \) must terminate
Behavioural Logic

program $\models \phi$

$\models$ describes the behaviour of programs

Example

$or(or(1,2), 3) \models \diamond \{3\}$ may return 3

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

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

 Logic

 $\models \phi$

 program

 formula

 $\models$ describes the behaviour of programs

 Logical Equivalence

 $M \equiv_{\log} N$ iff $\forall \phi. M \models \phi \iff N \models \phi$.

 Want: $(\equiv_{\log}) = (\equiv_{\text{ctx}})$
Continuation-Passing Style (CPS)

Type \( A \to B \) becomes \( A \to (B \to R) \to R \)

\( R = \) fixed return type

Example

\[
\text{add-cps} = \lambda(n:\text{nat}, m:\text{nat}, k:\text{nat} \to R). \ k \ (n + m) \\
: (\text{nat, nat, nat} \to R) \to R
\]
1. **Introduction: Program Equivalence and CPS**

- Higher-order functions, effects
- Contextual vs. logical equivalence
Outline

1. Introduction: Program Equivalence and CPS
2. Programming Calculus
3. Logic
4. Main Theorem
ECPS Calculus

- Types: $A, A_i ::= (A_1, \ldots, A_n) \rightarrow R \mid \text{nat} \quad (n \geq 0)$

- Values vs. computations

\[
\frac{
\Gamma, \overrightarrow{x : A} \vdash^c \textcolor{red}{t : R} \\
\Gamma \vdash^b \lambda(x : \overrightarrow{A}). \overrightarrow{t} : (\overrightarrow{A}) \rightarrow R
}{\Gamma \vdash^b \nu : (\overrightarrow{A}) \rightarrow R}
\]

\[
\frac{
\Gamma \vdash^b \nu : (\overrightarrow{A}) \rightarrow R \\
(\Gamma \vdash^b w_i : A_i)_i
}{\Gamma \vdash^c \nu(\overrightarrow{w}) : R}
\]
ECPS Calculus

- Types: $A, A_i := (A_1, \ldots, A_n) \to R \mid \text{nat} \quad (n \geq 0)$

- Values vs. computations

\[
\frac{\Gamma, x : A \vdash^c t : R}{\Gamma \vdash^b \lambda(x: A).t : (\vec{A}) \to R}
\quad
\frac{\Gamma \vdash^b v : (\vec{A}) \to R \quad (\Gamma \vdash^b w_i : A_i)_i}{\Gamma \vdash^c v(\vec{w}) : R}
\]

- Effect operations.

\[
\sigma \in \Sigma \quad (\Gamma \vdash^b_{\Sigma} v_i : \text{nat})_i \quad (\Gamma \vdash^b_{\Sigma} k_j : (\text{nat}, \ldots, \text{nat}) \to R)_j
\]

\[
\frac{}{\Gamma \vdash^c_{\Sigma} \sigma(\vec{v}_i, \vec{k}_j) : R}
\]
ECPS Calculus

- Types: $A, A_i := (A_1, \ldots, A_n) \rightarrow \mathbb{R} \mid \text{nat} \quad (n \geq 0)$

- Values vs. computations

\[
\begin{align*}
\Gamma, \chi : A \vdash^c t : \mathbb{R} & \quad \Gamma \vdash^b v : (\vec{A}) \rightarrow \mathbb{R} & \quad (\Gamma \vdash^b w_i : A_i)_i \\
\Gamma \vdash^b \lambda(\chi : \vec{A}). t : (\vec{A}) \rightarrow \mathbb{R} & \quad \Gamma \vdash^c v(\vec{w}) : \mathbb{R}
\end{align*}
\]

- Effect operations.

\[
\begin{align*}
\sigma \in \Sigma & \quad (\Gamma \vdash^b \nu_i : \text{nat})_i & \quad (\Gamma \vdash^b k_j : (\text{nat}, \ldots, \text{nat}) \rightarrow \mathbb{R})_j \\
\Gamma \vdash^c \sigma(\vec{\nu}_i, \vec{k}_j) : \mathbb{R}
\end{align*}
\]

- Recursion
Examples of Effect Signatures

**Probability:** 
\[ p\text{-or} : ((\cdot) \to \mathbb{R}, (\cdot) \to \mathbb{R}) \to \mathbb{R} \]

like \( \oplus : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \)

\( \text{geom} k \quad 1/2 \quad k \bar{1} \)
\( \quad 1/4 \quad k \bar{2} \)
\( \quad 1/8 \quad k \bar{3} \)
\( \quad \ldots \quad k \bar{n} \)

\( \text{geom} = \lambda k: \text{nat} \to \mathbb{R}. \)

\( (\text{rec } f. \lambda (n: \text{nat}, k': \text{nat} \to \mathbb{R}). \)
\( \quad p\text{-or}(\lambda().k' n, \lambda().f (\text{succ}(n), k')) \)
\( \quad ) (\bar{1}, k). \)

\( \text{geom} \) computes the geometric distribution:
it passes \( \bar{n} \) to the continuation \( k \) with probability \( \frac{1}{2^n} \).
Examples of Effect Signatures

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

Abstract syntax tree

```latex
test\_zero = \lambda y: \text{nat} \\
| \\
| case y of \\
| | \\
| zero \\
| | \\
\downarrow () \\
| | \\
| | \\
| loop
```

Computation tree

\( \llbracket - \rrbracket : (\vdash_{\Sigma} R) \to Trees_{\Sigma} \)  

\( \llbracket \text{test\_zero} \_0 \rrbracket = \downarrow \)  
\( \llbracket \text{test\_zero} \_1 \rrbracket = \bot \)

\textbf{test\_zero}: continuation that succeeds only on input 0.
Examples of Effect Signatures

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

Computation tree

$$\llbracket \text{geom} (\lambda x : \text{nat}. \downarrow (\text{()})) \rrbracket =$$

geometric distribution
Examples of Effect Signatures

Nondeterminism: \( \Sigma = \{ \text{or} : ((\rightarrow \mathbb{R}, (\rightarrow \mathbb{R}) \rightarrow \mathbb{R}, \downarrow : (\rightarrow \mathbb{R}) \rightarrow \mathbb{R} \}\)

Abstract syntax tree

```
    three_or_four \(k\)
       \(\downarrow\)
        \(k \ 4\)
       \(\downarrow\)
        \(k \ 3\)
       \(\downarrow\)
        \(k \ 4\)
```

Computation tree

\[
\left[ \text{three_or_four} \ (\lambda x: \text{nat}. \ \text{if } x = 4 \ \text{then } \downarrow \ \text{else } \text{loop} ) \right] =
\]

1. \(\text{or} \ 4\)
2. \(\downarrow\)
3. \(\text{or} \ \downarrow\)
4. \(\downarrow\)

\(\textbf{three_or_four} \) returns either \(3\) or \(4\) to continuation.
Examples of Effect Signatures

Global Store:
\[ \Sigma = \{ \text{lookup}_l : (\text{nat} \to \mathbb{R}) \to \mathbb{R}, \text{update}_l : (\text{nat}, () \to \mathbb{R}) \to \mathbb{R} \mid l \in \mathbb{L} \} \]
\[ \cup \{ \downarrow : () \to \mathbb{R} \} \]

Abstract syntax tree

\[
\text{suc\_update} = \\
\lambda (x : \text{nat}, k : () \to \mathbb{R}) \\
\quad \text{update}_{l_1} \\
\quad \text{succ}(x) \quad \downarrow \\
\]

Computation tree

\[
\text{[[suc\_update}} \\
\quad (\bar{0}, \lambda().\text{lookup}_{l_1}(\text{test\_zero}))]] = \\
\quad \text{update}_{l_1, \bar{1}} \\
\quad \text{lookup}_{l_1} \\
\quad \downarrow \\
\quad \downarrow \\
\quad \downarrow \\
\quad \downarrow \quad \downarrow \quad \downarrow \quad \ldots \\
\]

- **suc\_update**: write successor of the input to location \( l_1 \).

Other examples: I/O.
Observation \( P = \text{Set of trees} \)

- \( \Sigma = \text{Success}: \downarrow = \{ \downarrow \} \)
- \( \Sigma = \text{Probability}: \text{for } q \in \mathbb{Q}, \ 0 \leq q < 1 \)
  \[ P_{>q} = \{ \text{trees that succeed with probability } > q \} \]
- \( \Sigma = \text{Nondeterminism}: \)
  \[ \Diamond = \{ \text{trees with at least one } \downarrow \text{ leaf} \} \]
  \[ \Box = \{ \text{trees of finite height with only } \downarrow \text{ leaves} \} \]
- \( \Sigma = \text{Global store}: S \in \mathbb{L} \rightarrow \mathbb{N} \)
  \[ [S\downarrow] = \{ \text{trees that succeed when started in state } S \} \]
Observation $P = \text{Set of trees}$

$\Sigma = \text{Success: } ↓ = \{↓\}$

$[\text{test\_zero \overline{0}}] = ↓ \in ↓$
Observation $P = \text{Set of trees}$

$\Sigma = \text{Probability:}$

$P_{>q} = \{\text{trees that succeed with probability } > q\}$,

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

$\text{geom}(\lambda x: \text{nat.} \downarrow ()) = \in P_{>0.9}$
Observation $P = \text{Set of trees}$

- $\Sigma =$ Probability:
  $P_{>q} = \{\text{trees that succeed with probability } > q\}$,
  \[q \in \mathbb{Q}, \ 0 \leq q < 1\]

\[
\begin{align*}
[\text{geom} (\lambda x: \text{nat}. \text{if } x = 1 \text{ then } \downarrow () \text{ else } \text{loop})] &= \\
\end{align*}
\]

$\notin P_{>0.5}$
Observations

Observation $P = \text{Set of trees}$

$\Sigma = \text{Nondeterminism:}$
$\diamondsuit = \{\text{trees with at least one } \downarrow \text{ leaf}\}$
$\square = \{\text{trees of finite height with only } \downarrow \text{ leaves}\}$

$$\left[ \text{three_or_four } (\lambda x: \text{nat. if } x = 4 \text{ then } \downarrow \text{ else loop}) \right] =$$

$\in \diamondsuit$
$\notin \square$
Observation \( P = \text{Set of trees} \)

- \( \Sigma = \text{Global store}: S \in \mathbb{L} \rightarrow \mathbb{N} \)
  
  \[ [S \downarrow] = \{ \text{trees that succeed when started in state } S \} \]

\[
\begin{align*}
\{ \text{suc\_update} \} & = \\{ 0, \lambda().\mathit{lookup}_{l_1}(\text{test\_zero}) \} \\
\uparrow & \quad \uparrow \\
\mathit{update}_{l_1, \bar{l}} & \quad \mathit{lookup}_{l_1} \\
\downarrow & \quad \downarrow \\
\not\in [S\{l_1 := 0\} \downarrow] 
\end{align*}
\]
Programming Calculus

- $\Sigma = \text{signature of effect operations}$
- $P = \text{an observation} = \text{a set of trees}$
- $\mathcal{P} = \text{set of observations} = \text{set of sets of trees}$
1 Introduction: Program Equivalence and CPS
2 Programming Calculus
3 Logic
4 Main Theorem
Logic

- Parametrized by $\Sigma$ and $\mathcal{P}$ (set of observations).

- Value formulas

$$\phi := \{n\} \mid (\phi_1, \ldots, \phi_n) \mapsto P \mid \phi_1 \lor \phi_2 \mid \phi_1 \land \phi_2 \mid \neg \phi$$

Each value formula has a type:

$$\frac{n \in \mathbb{N}}{\{n\} : \text{nat}} \quad \frac{\phi_1 : A_1 \ldots \phi_n : A_n \quad P \in \mathcal{P}}{(\phi_1, \ldots, \phi_n) \mapsto P : (A_1, \ldots, A_n) \mapsto \mathbb{R}}$$

- Computation formulas = Observations from $\mathcal{P}$
Logic

▶ Value formulas:

\[ n \in \mathbb{N} \quad \Rightarrow \quad \{n\} : \text{nat} \]

\[ \phi_1 : A_1 \ldots \phi_n : A_n \quad \Rightarrow \quad P : (A_1, \ldots, A_n) \rightarrow \mathbb{R} \]

▶ Computation formulas: \( P \in \mathcal{P} \)

Examples of formulas

\[ \phi_1 = (\{1\} \lor \{2\} \lor \{3\}) \mapsto \square \quad : \text{nat} \rightarrow \mathbb{R} \]

\[ \phi_2 = \neg ((\{1\}) \mapsto \lozenge) \land \neg ((\{2\}) \mapsto \square) \quad : \text{nat} \rightarrow \mathbb{R} \]

\[ \phi_3 = \left( () \mapsto [S[l_0 := n, l_1 := n + 1] \downarrow] \right) \mapsto [S[l_0 := n] \downarrow] \quad : (() \rightarrow \mathbb{R}) \rightarrow \mathbb{R} \]
Logic

- Value formulas:
  \[
  n \in \mathbb{N} \quad \phi_1 : A_1 \ldots \phi_n : A_n \quad P \in \mathcal{P} \\
  (\phi_1, \ldots, \phi_n) \iff P : (A_1, \ldots, A_n) \rightarrow \mathbb{R}
  \]

- Computation formulas: \( P \in \mathcal{P} \)

- Satisfaction:
  \[
  v \models \{n\} \iff v = \overline{n}
  \]
  \[
  v \models (\phi_1, \ldots, \phi_n) \iff P \iff \text{for all } w_1, \ldots, w_n \text{ such that } \forall i. \ w_i \models \phi_i \text{ then } v(w_1, \ldots, w_n) \models P
  \]
  \[
  v \models \neg \phi \iff \text{it is false that } v \models \phi
  \]
  \[
  t \models P \iff \llbracket t \rrbracket \in P.
  \]
  \[
  \ldots
  \]
Examples of Logical Satisfaction

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

$\text{test}_{\text{zero}} \vdash \{0\} \leftrightarrow \downarrow$

$\downarrow = \{\downarrow\}$

continuation that tests whether the input is $\bar{0}$
Examples of Logical Satisfaction

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 $\bar{3}$ or $\bar{4}$

three_or_four $\models$

\[
\left(\left(\{3\} \mapsto \lozenge\right) \mapsto \lozenge\right) \land \left(\left(\{4\} \mapsto \lozenge\right) \mapsto \lozenge\right) \land \\
\left(\left(\{3\} \mapsto \square \land \{4\} \mapsto \square\right) \mapsto \square\right)
\]

: $(\text{nat} \rightarrow \text{R}) \rightarrow \text{R}$

- Function may pass to the continuation only $\bar{3}$ or $\bar{4}$. 
Examples of Logical Satisfaction

\[ \text{suc\_update} \equiv \]
\[ \land_{S \in \text{State}} \land_{n \in \mathbb{N}} \]
\[ (\{n\}, (\cdot) \mapsto [S\{l_0 := n, l_1 := n + 1\}]) \]
\[ \mapsto [S\{l_0 := n\}] \]
\[ : (\text{nat}, (\cdot) \to \mathbb{R}) \to \mathbb{R} \]

Given argument \( \bar{n} \) and a continuation that succeeds when started in state \([S\{l_0 := n, l_1 := n + 1\}]\), the function succeeds when started in state \([S\{l_0 := n\}]\).
Outline

1. Introduction: Program Equivalence and CPS
2. Programming Calculus
3. Logic
4. Main Theorem
Main Theorem

\[ \mathcal{P} = \text{set of sets of trees} \]

**Logical Equivalence**

\[ \forall \phi. M \models \phi \iff N \models \phi. \]

**Contextual Equivalence**

\[ \forall C. \forall P \in \mathcal{P}. [C[M]] \in P \iff [C[N]] \in P. \]

**Main Theorem**

\[ \begin{align*}
\mathcal{P} &\text{ consistent} \\
\mathcal{P} &\text{ decomposable} \\
P &\in \mathcal{P} \text{ Scott-open} \\
\end{align*} \]

\[ \begin{align*}
\text{Logical Equivalence} &\iff \\
\text{Contextual Equivalence} &\iff \\
\end{align*} \]
Decomposability

\( \mathcal{P} \) is decomposable if for any \( P \in \mathcal{P} \), and for any \( tr \in P \):

\[
\forall \sigma \in \Sigma. ( tr = \sigma \overleftarrow{v}(tr') \implies \\
\exists P' \in \mathcal{P} \cup \{Trees_\Sigma\}. \\
\overrightarrow{tr'} \in \overrightarrow{P'} \text{ and } \forall \overrightarrow{p'} \in \overrightarrow{P'}. \sigma \overleftarrow{v}(\overrightarrow{p'}) \in P ).
\]

[Johann et.al. LICS’10], [Simpson & Voorneveld, ESOP’18]
\( \mathcal{P} \) is decomposable if for any \( P \in \mathcal{P} \), and for any \( tr \in P \):

\[
\forall \sigma \in \Sigma. \ (tr = \sigma \overrightarrow{\lor}(tr')) \implies \\
\exists P' \in \mathcal{P} \cup \{Trees_\Sigma\}. \\
\quad \overrightarrow{tr'} \in P' \text{ and } \forall p' \in P'. \ \sigma \overrightarrow{\lor}(p') \in P).
\]
Proof Sketch

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

Logical Equivalence \(\cong\) Applicative Bisimilarity \(\cong\) Contextual Equivalence

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

Applicative simulation

A collection of relations $R^b_A \subseteq (\vdash \Sigma A)^2$ for each type $A$ and $R^c \subseteq (\vdash \Sigma R)^2$ is an applicative $\Psi$-simulation if:

- $\forall R^b_{nat} \; w \implies v = w$.
- $s \; R^c \; t \implies \forall P \in \Psi. ([s] \in P \implies [t] \in P)$.
- $\forall R^b_{(\overrightarrow{A}) \rightarrow R} \; u \implies \forall (\vdash \Sigma w_i : A_i). \; v(\overrightarrow{w_i}) \; R^c \; u(\overrightarrow{w_i})$.

Applicative Bisimilarity

Is the greatest symmetric $\Psi$-simulation.
Applicative Bisimilarity — Example

\[ v = \lambda(). or(\lambda(). \downarrow, \text{loop}, \lambda(). \downarrow) : (\to R \]  
\[ w = \lambda(). or(\lambda(). \downarrow, or(\text{loop, } \lambda(). \downarrow)) \]

Prove: \( v \) bisimilar to \( w \)

Choose \( R^v_{(\to_R} = \{(v, w), (w, v)\} \) and \( R^c = \{(v(), w()), (w(), v())\} \).

\( R \) is a bisimulation \( \implies \) \( R \) included in bisimilarity.
Proof Sketch

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

\[ \vdash_{\Sigma} w_1 : A_1 \ldots \vdash_{\Sigma} w_n : A_n \quad P \in \mathcal{P} \]
\[ (w_1, \ldots, w_n) \mapsto P : (A_1, \ldots, A_n) \rightarrow \mathcal{R} \]

Logical Equivalence \(\equiv\) Applicative Bisimilarity \(\equiv\) Contextual Equivalence

via a simpler equi-expressive logic, using 1)

[Simpson and Voorneveld, ESOP’18]
Proof Sketch

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

\[ \\begin{align*}
\text{Logical Equivalence} & \quad \equiv \quad \text{Applicative Bisimilarity} \\
\text{2) } & \quad \equiv \quad \text{Contextual Equivalence} \\
\text{using consistency, Scott-openness and 1)} \\
\text{N.B. } & \quad \geq \text{ interesting}
\end{align*} \]
Proof Sketch

Applicative simulation

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

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

Logical Equivalence $\equiv 2)$  
Applicative Bisimilarity $\equiv 3)$  
Contextual Equivalence
Summary

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

Main Theorem [Matache & Staton, FoSSaCS’19]

\[ \varPsi \text{ consistent} \]
\[ \varPsi \text{ decomposable} \]
\[ P \in \varPsi \text{ Scott-open} \]

\[ \iff \]

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

\[ \mathcal{P} \text{ consistent} \quad \mathcal{P} \text{ decomposable} \quad P \in \mathcal{P} \text{ Scott-open} \]

\[ \Leftrightarrow \]

Logical Equivalence

\[ \equiv \]

Contextual Equivalence

[Matache & Staton, FoSSaCS’19]