Outcome Separation Logic

Local Reasoning for Correctness and Incorrectness with Computational Effects

Noam Zilberstein, Cornell University
(Based on joint work with Angelina Saliling, Alexandra Silva, and Derek Dreyer)

Iris Workshop
Saarbrücken, Germany. May 23, 2023

Overview

- Since Hoare Logic was introduced, new programming paradigms have arisen:
  - ***Computational effects:*** nondeterminism, randomization, quantum computation, concurrency, . . .
  - ***Incorrectness Reasoning***

Overview

- Since Hoare Logic was introduced, new programming paradigms have arisen:
  - ***Computational effects:*** nondeterminism, randomization, quantum computation, concurrency, ...
  - ***Incorrectness Reasoning***
- Can a single logical foundation capture it all?

Overview

- Since Hoare Logic was introduced, new programming paradigms have arisen:
  - ***Computational effects:*** nondeterminism, randomization, quantum computation, concurrency, . . .
  - ***Incorrectness Reasoning***
- Can a single logical foundation capture it all?
- What does a Separation Logic based on this look like?

Computational Effects and Incorrectness

Nondeterministic Bugs

- Malloc is nondeterministic, it might return null.

$$\begin{array}{l}\{\mathrm{ok} : \mathrm{emp}\} \\ \quad x := \mathrm{malloc}() \mathbin{;} \\ \quad [x] \leftarrow 1 \\ \{(\mathrm{ok} : x \mapsto 1) \vee (\mathrm{er} : x = \mathrm{null})\}\end{array}$$

- Does this specification characterize the bug?

Nondeterministic Bugs

- Malloc is nondeterministic, it might return null.

$$
\begin{array}{l}
\{\mathrm{ok} : \mathrm{emp}\} \\
\quad x := \mathrm{malloc}() \mathbin{;} \\
\quad [x] \leftarrow 1 \\
\{(\mathrm{ok} : x \mapsto 1) \lor (\mathrm{er} : x = \mathrm{null})\}
\end{array}
$$

- Does this specification characterize the bug?
- ***No!*** We don't know if the crash state is reachable

Incorrectness Logic

- Incorrectness Logic Semantics

$$\models [P]\ C\ [Q] \qquad \text{iff} \qquad \forall \tau \models Q.\quad \exists \sigma \models P.\quad \tau \in [\![C]\!](\sigma)$$

- Every state satisfying the post is *reachable* from some state satisfying the pre

Incorrectness Logic

- Incorrectness Logic Semantics

$$\vDash [P]\ C\ [Q] \qquad \text{iff} \qquad \forall \tau \vDash Q.\quad \exists \sigma \vDash P.\quad \tau \in [\![C]\!]\,(\sigma)$$

- Every state satisfying the post is *reachable* from some state satisfying the pre
- Does this spec characterize the bug?

$$\begin{array}{l} [\mathrm{ok} : \mathrm{emp}] \\ \quad x := \mathrm{malloc}()\ ; \\ \quad [x] \leftarrow 1 \\ [(\mathrm{ok} : x \mapsto 1) \vee (\mathrm{er} : x = \mathrm{null})] \end{array}$$

Dropping Disjuncts

- The following (more concise) spec is also valid

$$\begin{array}{l} [\mathrm{ok} : \mathrm{emp}] \\ \quad x := \mathrm{malloc}() \fatsemi \\ \quad [x] \leftarrow 1 \\ [(\mathrm{ok} : x \mapsto 1) \vee (\mathrm{er} : x = \mathrm{null})] \end{array} \implies \begin{array}{l} [\mathrm{ok} : \mathrm{emp}] \\ \quad x := \mathrm{malloc}() \fatsemi \\ \quad [x] \leftarrow 1 \\ [\mathrm{er} : x = \mathrm{null}] \end{array}$$

- Static analyzers do not have to traverse all program paths

Program Logics—Comparison

- Incorrectness Logic:
  - Specify *true* bugs
  - Drop program paths for efficiency
  - ...but specialized to nondeterminism
- Hoare Logic:
  - Can be used for correctness
  - Supports many types of effects (*e.g.*, probabilistic programs)
  - ...but cannot identify bugs

Program Logics—Comparison

- Incorrectness Logic:
  - Specify *true* bugs
  - Drop program paths for efficiency
  - …but specialized to nondeterminism
- Hoare Logic:
  - Can be used for correctness
  - Supports many types of effects (*e.g.*, probabilistic programs)
  - …but cannot identify bugs
- Can we get the best of both worlds?

Outcome Logic

- Similar to Hoare Logic, but pre/post refer to *collections* of states

$$\vDash \langle \varphi \rangle \; C \; \langle \psi \rangle \qquad \text{iff} \qquad \forall m. \quad m \vDash \varphi \implies [\![C]\!]\, m \vDash \psi$$

- What is m?

$$\begin{array}{ll} \text{Nontermination:} & m \in \mathsf{State} + \{\bot\} \\ \text{Nondeterminism:} & m \in \mathsf{Set}(\mathsf{State}) \\ \text{Randomization:} & m \in \mathsf{Dist}(\mathsf{State}) \\ \text{Exceptions:} & m \in \mathsf{State} + E \\ & \vdots \end{array}$$

Outcome Assertions

- Syntax
$$\varphi ::= \top \mid \top^{\oplus} \mid \varphi \vee \psi \mid \varphi \oplus \psi \mid \epsilon : P \qquad\qquad \epsilon ::= \mathrm{ok} \mid \mathrm{er}$$
- Semantics (Nondeterministic Interpretation):

$$\begin{array}{lll}
S \vDash \top^{\oplus} & \text{iff} & S = \emptyset \\
S \vDash \varphi \oplus \psi & \text{iff} & \exists S_1, S_2.\quad S = S_1 \cup S_2 \quad \text{and} \quad S_1 \vDash \varphi \quad \text{and} \quad S_2 \vDash \psi \\
S \vDash (\epsilon : P) & \text{iff} & S \neq \emptyset \quad \text{and} \quad \forall \sigma \in S.\ \sigma \vDash (\epsilon : P) \\
 & \vdots &
\end{array}$$

Specifying a Bug

- Does this spec characterize the bug?

$$\begin{array}{l} \langle \mathrm{ok} : \mathrm{emp} \rangle \\ \quad x := \mathrm{malloc}() \fatsemi \\ \quad [x] \leftarrow 1 \\ \langle (\mathrm{ok} : x \mapsto 1) \oplus (\mathrm{er} : x = \mathrm{null}) \rangle \end{array}$$

Specifying a Bug

- Does this spec characterize the bug?

$$\begin{array}{l} \langle \mathrm{ok} : \mathrm{emp} \rangle \\ \quad x := \mathrm{malloc}() \,⨟ \\ \quad [x] \leftarrow 1 \\ \langle (\mathrm{ok} : x \mapsto 1) \oplus (\mathrm{er} : x = \mathrm{null}) \rangle \end{array}$$

- Yes!

Dropping Outcomes

- We can also drop some of the program paths

$$
\begin{array}{l}
\langle \mathrm{ok} : \mathrm{emp} \rangle \\
\quad x := \mathrm{malloc}() \fatsemi \\
\quad [x] \leftarrow 1 \\
\langle (\mathrm{ok} : x \mapsto 1) \oplus (\mathrm{er} : x = \mathrm{null}) \rangle
\end{array}
\implies
\begin{array}{l}
\langle \mathrm{ok} : \mathrm{emp} \rangle \\
\quad x := \mathrm{malloc}() \fatsemi \\
\quad [x] \leftarrow 1 \\
\langle (\mathrm{er} : x = \mathrm{null}) \oplus \top \rangle
\end{array}
$$

Probabilistic Outcomes

- Outcomes are more general than nondeterminism
- We can reason probabilistically about unreliable network connections

$$\begin{array}{l}\langle \mathsf{ok} : \mathsf{true} \rangle \\ \quad x := \mathsf{ping}(192.0.2.1) \\ \langle (\mathsf{ok} : x = 200) \oplus_{99\%} (\mathsf{er} : x = 500) \rangle\end{array}$$

The Frame Rule

The Outcome Separating Conjunction

- A new operator $\varphi \circledast F$ is defined inductively:

$$
\begin{aligned}
\top \circledast F &\triangleq \top \\
\top^{\oplus} \circledast F &\triangleq \top^{\oplus} \\
(\varphi \vee \psi) \circledast F &\triangleq (\varphi \circledast F) \vee (\psi \circledast F) \\
(\varphi \oplus \psi) \circledast F &\triangleq (\varphi \circledast F) \oplus (\psi \circledast F) \\
(\epsilon : P) \circledast F &\triangleq \epsilon : P * F
\end{aligned}
$$

The Outcome Logic Frame Rule

- The usual Frame Rule:

$$\frac{\{P\}\ C\ \{Q\} \qquad \mathrm{mod}(C) \cap \mathrm{fv}(F) = \emptyset}{\{P * F\}\ C\ \{Q * F\}}\textsc{Frame}$$

- What we want:

$$\frac{\langle \varphi \rangle\ C\ \langle \psi \rangle \qquad \mathrm{mod}(C) \cap \mathrm{fv}(F) = \emptyset}{\langle \varphi \circledast F \rangle\ C\ \langle \psi \circledast F \rangle}\textsc{Frame}$$

Memory Allocation and the Frame Rule

- Suppose allocation is *deterministic*:

$$\{\mathrm{emp}\}\ x := \mathrm{alloc}()\ \{x = 1\}$$

- The *address* of x is 1, since that is the first address

Memory Allocation and the Frame Rule

- Suppose allocation is *deterministic*:

$$\{\mathrm{emp}\}\ x := \mathrm{alloc}()\ \{x = 1\}$$

- The *address* of x is 1, since that is the first address
- Now, we use the frame rule:

$$\dfrac{\{\mathrm{emp}\}\ x := \mathrm{alloc}()\ \{x = 1\}}{\{y \mapsto 2\}\ x := \mathrm{alloc}()\ \{y \mapsto 2 \land x = 1\}}\ \textsc{Frame}$$

- Was this inference valid?

Nondeterminism and the Frame Rule

- Idea: make memory allocation nondeterministic
- We cannot say that $x = 1$, we can only say:

$$\{\mathrm{emp}\}\ x := \mathrm{alloc}()\ \{x = 1 \vee x = 2 \vee \ldots\}$$

- After applying the frame rule, this is still true

$$\{y \mapsto 2\}\ x := \mathrm{alloc}()\ \{y \mapsto 2 \wedge (x = 1 \vee x = 2 \vee \ldots)\}$$

Nondeterminism and the Frame Rule

- Idea: make memory allocation nondeterministic
- We cannot say that $x = 1$, we can only say:

$$\{\mathrm{emp}\}\ x := \mathrm{alloc}()\ \{x = 1 \lor x = 2 \lor \ldots\}$$

- After applying the frame rule, this is still true

$$\{y \mapsto 2\}\ x := \mathrm{alloc}()\ \{y \mapsto 2 \land (x = 1 \lor x = 2 \lor \ldots)\}$$

- Problem: OL supports execution models *other than* nondeterminism

Must and May Properties

- Separation Logic only supports *must* properties
- Outcome Logic also supports *may* properties

$$
\begin{array}{l}
\langle \mathrm{ok} : \mathrm{emp} \rangle \\
\quad x := \mathrm{malloc}() \fatsemi \\
\quad [x] \leftarrow 1 \\
\langle (\mathrm{er} : x = \mathrm{null}) \oplus \top \rangle
\end{array}
$$

May Properties and Memory Allocation

- Even with nondeterministic allocation, the following is valid:

$$\langle \mathrm{ok} : \mathrm{emp} \rangle \; x := \mathrm{alloc}() \; \langle (\mathrm{ok} : x = 1) \oplus \top \rangle$$

- Using the frame rule, we get something that is not true

$$\langle \mathrm{ok} : y \mapsto 2 \rangle \; x := \mathrm{alloc}() \; \langle (\mathrm{ok} : y \mapsto 2 \land x = 1) \oplus \top \rangle$$

The Solution

- To make memory allocation *local*, assertions cannot mention heap addresses
- Lemma: Outcome Logic assertions are invariant to heap permutations

$$\forall \pi. \quad m \vDash \varphi \implies \pi(m) \vDash \varphi$$

Safe Preconditions

- Separation Logic requires preconditions to be *safe*
- If not, the following would be valid:

$$\{\mathrm{emp}\}\ [x] \leftarrow 1\ \{\mathrm{emp}\}$$

- Now, using the frame rule, we get

$$\{x \mapsto 2\}\ [x] \leftarrow 1\ \{x \mapsto 2\}$$

- Which is clearly false!

Safety in Outcome Logic

- Safe preconditions are *undesirable* in Outcome Logic
- Bugs do not require us to look at all paths

$$\langle \mathrm{ok} : x = \mathrm{null} \rangle \; ([x] \leftarrow 1) + C \; \langle (\mathrm{er} : x = \mathrm{null}) \oplus \top \rangle$$

- If the pre had to be safe, we would need to inspect C

Outcome Logic + Concurrency

Concurrent Outcomes

- Outcomes correspond to possible interleavings

$$
\begin{array}{l}
\langle \mathrm{ok} : x \mapsto -\rangle \\
\quad [x] \leftarrow 1 \parallel [x] \leftarrow 2 \\
\langle (\mathrm{ok} : x \mapsto 1) \oplus (\mathrm{ok} : x \mapsto 2)\rangle
\end{array}
$$

The Exchange Law

- From Concurrent Kleene Algebra:

$$(C_1 \parallel C_1') \fatsemi (C_2 \parallel C_2') \leq (C_1 \fatsemi C_2) \parallel (C_1' \fatsemi C_2')$$

- In Outcome Logic:

$$\frac{\langle \varphi \rangle\ (C_1 \parallel C_1') \fatsemi (C_2 \parallel C_2')\ \langle \psi \rangle}{\langle \varphi \rangle\ (C_1 \fatsemi C_2) \parallel (C_1' \fatsemi C_2')\ \langle \psi \oplus \top \rangle}\ \textsc{Exchange}$$

Concurrent Bugs

- The following program will crash in *some* interleavings

$$\begin{array}{l|l} [x] \leftarrow 1 \,;\, & \mathsf{free}(x) \,;\, \\ [x] \leftarrow 2 & \mathsf{skip} \end{array}$$

- The exchange law can partially sequentialize the program

$$\dfrac{\dfrac{\vdots}{\langle \mathsf{ok} : x \mapsto - \rangle \; ([x] \leftarrow 1 \parallel \mathsf{free}(x)) \,;\, ([x] \leftarrow 2 \parallel \mathsf{skip}) \; \langle \mathsf{er} : x \not\mapsto \rangle}}{\langle \mathsf{ok} : x \mapsto - \rangle \; ([x] \leftarrow 1 \,;\, [x] \leftarrow 2) \parallel (\mathsf{free}(x) \,;\, \mathsf{skip}) \; \langle (\mathsf{er} : x \not\mapsto) \oplus \top \rangle}\;\textsc{Exchange}$$

- Complete the proof using all the Iris machinery

Conclusion

- OL is sound for correctness and incorrectness with effects
- The OL frame rule allows effects and dropping paths
- Future work: OL + concurrency (with Iris)
- Further reading at `cs.cornell.edu/~noamz`
  - Outcome Logic: A Unified Foundation for Correctness and Incorrectness Reasoning [OOPSLA'23]
  - Outcome Separation Logic: Local Reasoning for Correctness and Incorrectness with Computational Effects [arXiv]
- A lot more cool stuff!
  - Algebraic semantics based on semirings
  - Relative completeness proof
  - OL subsumes Hoare Logic, probabilistic Hoare Logic, etc
  - Symbolic execution using bi- and tri-abduction