#Slides

Foxtrot

Contextual Refinement of Higher-Order Concurrent Probabilistic Programs

A Logsem talk by Heili

Joint work with Alejandro, Joe, Lars

WARNING:

Increasingly worse handwriting incoming

- Agreed to step in last Tuesday
- Got sick last week
- I just have bad handwriting in general
- I am tired of using Latex atm
- Studies have shown that people pay more attention to slides with ugly font (Oppenheimer, Jan 2011)

- Concurrency is important
  - e.g. fork, $e_1 \| e_2$

- Concurrency is important
  - e.g. fork, $e_1 \| e_2$
- Probability is also important
  - e.g. Rand N steps to a number from 0 to N (inclusive) uniformly

$$\text{Rand } 1 \xrightarrow{\frac{1}{2}} 0 \qquad \text{Rand } 1 \xrightarrow{\frac{1}{2}} 1$$

- Concurrency is important
  - e.g. fork, $e_1 \| e_2$
- Probability is also important
  - e.g. Rand N steps to a number from 0 to N (inclusive) uniformly
- Reasoning about <u>BOTH</u> concurrency and probability is hard
  - Varacca & Winskel showed nondeterminism + prob do not compose nicely

Consider

$$fImpl \triangleq \lambda_.\ \mathrm{let}\ (x, y) = (\mathrm{rand}\ 1 \mid\mid\mid \mathrm{rand}\ 1)\ \mathrm{in}$$
$$2 \cdot x + y$$

Consider

$$fImpl \triangleq \lambda_.\ \text{let } (x, y) = (rand\ 1 \mid\mid\mid rand\ 1) \text{ in } 2 \cdot x + y$$

This behaves the same as

$$fSpec \triangleq \lambda_.\ Rand\ 3$$

Consider

$$fImpl \triangleq \lambda_.\ \text{let } (x, y) = (rand\ 1 \mid\mid\mid rand\ 1) \text{ in } 2 \cdot x + y$$

This behaves the same as

$$fSpec \triangleq \lambda_.\ Rand\ 3$$

AIM: Prove that $fImpl \cong_{ctx} fSpec$

($fImpl$ is contextually equivalent to $fSpec$)

- $e_1 \cong_{CE_*} e_2$ informally means that $\forall$ well-typed contexts C of ground type, the behavior of $C[e_1] = C[e_2]$

- $e_1 \cong_{ctx} e_2$ informally means that $\forall$ well-typed contexts C of ground type, the behavior of $C[e_1] = C[e_2]$

- $e_1 \cong_{ctx} e_2$ defined as $\underbrace{e_1 \preceq_{ctx} e_2}_{e_1 \text{ contextually refines } e_2} \wedge e_2 \preceq_{ctx} e_1$

- $e_1 \cong_{ctx} e_2$ informally means that $\forall$ well-typed contexts C of ground type, the behavior of $C[e_1] = C[e_2]$

- $e_1 \cong_{ctx} e_2$ defined as $\underbrace{e_1 \preceq_{ctx} e_2}_{e_1 \text{ contextually refines } e_2} \wedge e_2 \preceq_{ctx} e_1$

- $e_1 \preceq_{ctx} e_2$ informally means $\forall$ well-typed contexts C of ground type, the behavior of $C[e_1] \subseteq C[e_2]$

- $e_1 \cong_{ctx} e_2$ informally means that $\forall$ well-typed contexts C of ground type, the behavior of $C[e_1] = C[e_2]$

- $e_1 \cong_{ctx} e_2$ defined as $\underbrace{e_1 \preceq_{ctx} e_2}_{e_1 \text{ contextually refines } e_2} \wedge e_2 \preceq_{ctx} e_1$

- $e_1 \preceq_{ctx} e_2$ informally means $\forall$ well-typed contexts C of ground type, the behavior of $C[e_1] \subseteq C[e_2]$

  Usually hard to prove!
  - Quantifies over all contexts C
  - Context C might use probability + concurrency

Prior Work on Proving Contextual Refinements

- Various work for concurrency, e.g. CaReSL, ReLoC
- Various techniques for probability
  - Bizjak and Birkedal used Logical Relations
  - Extended by Ward et al. to continuous distributions
  - Ehrhard et al. used denotational model
  - Bisimulation techniques for prob π-calculus (Crubillé, Dal Lago, Sabel, ...)
- Aguirre & Birkedal $\longrightarrow$ Prob + countable nondeterminism
- Clutch by Gregersen et al. $\longrightarrow$ Asynchronous couplings
- Approxis by Haselwarter et al. $\longrightarrow$ CR by error approximation

Prior Work on Proving Contextual Refinements

- Various work for concurrency, e.g. CaReSL, ReLoC
- Various techniques for probability
  - Bizjak and Birkedal used Logical Relations
  - Extended by Ward et al. to continuous distributions
  - Ehrhard et al. used denotational model
  - Bisimulation techniques for prob π-calculus (Crubillé, Dal Lago, Sabel, ...)
- Aguirre & Birkedal $\longrightarrow$ Prob + countable nondeterminism
- Clutch by Gregersen et al. $\longrightarrow$ Asynchronous couplings
- Approx is by Haselwarter et al. $\longrightarrow$ CR by error approximation

AFAIK, no work on both concurrency + prob

Foxtrot is

- A higher-order Separation Logic for proving CR of higher-order concurrent probabilistic programs with higher-order local state

Foxtrot is

- A higher-order Separation Logic for proving CR of higher-order concurrent probabilistic programs with higher-order local state
- Supports reasoning principles in standard CSL, e.g. invariants, ghost states

Foxtrot is

- A higher-order Separation Logic for proving CR of higher-order concurrent probabilistic programs with higher-order local state
- Supports reasoning principles in standard CSL, e.g. invariants, ghost states
- Supports advanced probabilistic reasoning principles, e.g. presampling tapes, induction by error amplification

Foxtrot is

- A higher-order Separation Logic for proving CR of higher-order concurrent probabilistic programs with higher-order local state
- Supports reasoning principles in standard CSL, e.g. invariants, ghost states
- Supports advanced probabilistic reasoning principles, e.g. presampling tapes, induction by error amplification
- Mechanized in Rocq + Iris

Foxtrot is

- A higher-order Separation Logic for proving CR of higher-order concurrent probabilistic programs with higher-order local state
- Supports reasoning principles in standard CSL, e.g. invariants, ghost states
- Supports advanced probabilistic reasoning principles, e.g. presampling tapes, induction by error amplification
- Mechanized in Rocq + Iris

* Neither a fox or a trot (actually a kind of dance)

Key idea: A logical concurrent probabilistic refinement judgement

$$\models e \preceq e' : \tau$$

Key idea: A logical concurrent probabilistic refinement judgement

$$\models e \preceq e' : \tau$$

Soundness: $\models e \preceq e' : \tau \rightarrow \vdash e \leq_{ctx} e' : \tau$

Fundamental lemma: $\vdash e : \tau \rightarrow \models e \preceq e' : \tau$

Key idea: A logical concurrent probabilistic refinement judgement

$$\models e \preceq e' : \tau$$

Soundness: $\models e \preceq e' : \tau \rightarrow \vdash e \preceq_{ctx} e' : \tau$

Fundamental lemma: $\vdash e : \tau \rightarrow \models e \preceq e' : \tau$

- $\models e \preceq e' : \tau$ defined as

$$\forall K, j. \underbrace{\{ j \Mapsto K[e'] \}\ e\ \{ v. \exists v'. j \Mapsto K[v'] * [\![\tau]\!](v, v') \}}_{\text{All the magic is in the wp def!}}$$

Logical Relations in standard way! (Timany et al.)

Standard rules in Separation Logic!

$$\frac{\{P\}\, e\, \{v.\ Q\} \qquad \forall v.\ \{Q\}\, k[v]\, \{R\}}{\{P\}\, e\, \{R\}}$$

$$\frac{\{P\}\, e\, \{Q\}}{\{P * R\}\, e\, \{Q * R\}}$$

$$\frac{\{T\}\, e\, \{T\}}{\{T\}\ \mathrm{fork}\ e\ \{v.\ v = ()\}}$$

$$\frac{}{\{l \mapsto v\}\ !\, l\ \{x.\ x = v * l \mapsto v\}}$$

Standard rules in Separation Logic!

$$\frac{\{P\}\, e\, \{v.\ Q\} \qquad \forall v.\ \{Q\}\, k[v]\, \{R\}}{\{P\}\, e\, \{R\}}$$

$$\frac{\{P\}\, e\, \{Q\}}{\{P * R\}\, e\, \{Q * R\}}$$

$$\frac{\{T\}\, e\, \{T\}}{\{T\}\, \mathrm{fork}\ e\, \{v.\ v = ()\}}$$

$$\frac{}{\{l \mapsto v\}\, !l\, \{x.\ x = v * l \mapsto v\}}$$

Spec-side rules defined with probabilistic update modality for simplicity

$$\frac{\{P * Q\}\, e\, \{R\}}{\{\Rightarrow P * Q\}\, e\, \{R\}}$$

$$\frac{l \mapsto_s v \qquad j \Mapsto K[!l]}{\Rightarrow (l \mapsto_s v * j \Mapsto K[v])}$$

$$\frac{j \Mapsto K[\mathrm{fork}\ e]}{\Rightarrow (j \Mapsto K[()] * \exists j'.\ j' \Mapsto e)}$$

Standard Rules in CSL

$$\frac{\{P_1\}\, e_1\, \{v_1.\, Q_1 v_1\} \qquad \{P_2\}\, e_2\, \{v_2.\, Q_2 v_2\}}{\{P_1 * P_2\}\; e_1 \,|||\, e_2 \;\; \{(v_1, v_2).\; Q_1 v_1 * Q_2 v_2\}}$$

$$\frac{j \Mapsto K[e_1 \,|||\, e_2]}{\Rrightarrow \exists j_1\, j_2\, K_1\, K_2.\; \left(\begin{array}{l} j_1 \Mapsto K_1[e_1] * j_2 \Mapsto K_2[e_2] * \\ \forall v_1\, v_2.\, (j_1 \Mapsto K_1[v_1] * j_2 \Mapsto K_2[v_2]) \mathrel{-\!\!*} \\ \qquad \Rrightarrow j \Mapsto K[(v_1, v_2)] \end{array} \right)}$$

Invariants and Ghost Resources

$$\frac{\{\boxed{P}^{\iota} * Q\}\, e\, \{R\}}{\{P * Q\}\, e\, \{R\}}$$

$$\frac{e \text{ atomic} \quad \{I * P\}\, e\, \{I * Q\}_{\mathcal{E}}}{\{\boxed{I}^{\iota} * P\}\, e\, \{Q\}_{\mathcal{E} \uplus \{\iota\}}}$$

$$\frac{a \in V}{\Rightarrow \exists \gamma.\ \boxed{a}^{\gamma}}$$

$$\frac{a \rightsquigarrow b}{\boxed{a}^{\gamma} \vdash \Rightarrow \boxed{b}^{\gamma}}$$

Coupling Rule

Sequential coupling-based logics support:

$$\frac{f \text{ bijective} \qquad \forall n.\, n \leq N \rightarrow n \leq f(n) : \Phi}{\text{Rand } N \preceq \text{Rand } N : \Phi}$$

Coupling Rule

Sequential coupling-based logics support:

$$\frac{f \text{ bijective} \qquad \forall n.\ n \leq N \rightarrow n \leq f(n) : \Phi}{\text{Rand } N \preceq \text{Rand } N : \Phi}$$

We extend that to the concurrent case

$$\frac{f \text{ bijective} \qquad \forall n.\ n \leq N \rightarrow \{j \mapsto K[f(n)]\}\ n\ \{\Phi\}}{\{j \mapsto K[\text{rand } N]\}\ \text{rand } N\ \{\Phi\}}$$

Slogan 1:

Invariants + ghost resources + coupling rule

= (almost) all you need!

$$
\begin{array}{l}
\text{let } (y, r) = (\text{ref } 0, \text{ref } 0) \text{ in} \\
\text{let } x_1 = !y \text{ in} \\
\text{let } x_2 = \text{Rand } 1 \text{ in} \\
r \leftarrow ((x_1 + x_2) \bmod 2) \\
\\
!r
\end{array}
\quad \Bigg\| \quad y \leftarrow 1
$$

$$
\begin{array}{l}
\text{let } (y, r) = (\text{ref } 0, \text{ref } 0) \text{ in} \\
\text{let } x_1 = !y \text{ in} \\
\text{let } x_2 = \text{Rand } 1 \text{ in} \\
r \leftarrow ((x_1 + x_2) \bmod 2) \\
\\
!r
\end{array}
\quad \Bigg\| \quad y \leftarrow 1
$$

This is Ctx-equivalent to Rand 1 !

$$
\begin{array}{l}
\text{let } (y, r) = (\text{ref } 0, \text{ref } 0) \text{ in} \\
\text{let } x_1 = !y \text{ in} \\
\text{let } x_2 = \text{Rand } 1 \text{ in} \\
r \leftarrow ((x_1 + x_2) \bmod 2) \\
\\
!r
\end{array}
\quad \Bigg\| \quad y \leftarrow 1
\quad \leq \quad \text{Rand } 1
$$

- Define invariant $I \triangleq y \mapsto 0 \vee y \mapsto 1$

$$
\begin{array}{l}
\text{let } (y, r) = (\text{ref } 0, \text{ref } 0) \text{ in} \\
\text{let } x_1 = !y \text{ in} \\
\text{let } x_2 = \text{Rand } 1 \text{ in} \\
r \leftarrow ((x_1 + x_2) \bmod 2) \\
\\
\quad\quad\quad

$$\text{Rand } 1 \quad \leq \quad \begin{array}{l} \text{let } (y, r) = (\text{ref } 0, \text{ref } 0) \text{ in} \\ \text{let } x_1 = !y \text{ in} \\ \text{let } x_2 = \text{Rand } 1 \text{ in} \\ r \leftarrow ((x_1 + x_2) \bmod 2) \\ !r \end{array} \quad \Bigg\| \quad y \leftarrow 1$$

- This direction is even easier!

$$\mathsf{Rand}\ 1 \quad \preceq \quad \begin{array}{l} \mathsf{let}\ (y, r) = (\mathsf{ref}\ 0, \mathsf{ref}\ 0)\ \mathsf{in} \\ \mathsf{let}\ x_1 = !y\ \mathsf{in} \\ \mathsf{let}\ x_2 = \mathsf{Rand}\ 1\ \mathsf{in} \\ r \leftarrow ((x_1 + x_2) \bmod 2) \end{array} \quad \Bigg\|\Bigg\| \quad y \leftarrow 1$$

$$!r$$

- This direction is even easier!
- We can choose how to execute the RHS
- Read 0 from y and couple with

$$f \triangleq \lambda x.x$$

What about our first example?

let $(x, y) = (\text{Rand } 1) \,|||\, (\text{Rand } 1)$ in $\cong$ Rand 3

$2 \cdot x + y$

Let us consider the left refines right direction.

What about our first example?

let $(x, y) = (\text{Rand } 1) \,|||\, (\text{Rand } 1)$ in $\cong$ Rand 3

$2 \cdot x + y$

Let us consider the left refines right direction.

- How to split the spec resource between the two threads?
- How to couple two Rand 1 with a Rand 3?

Presampling Tapes

$$\frac{}{\{\top\}\ \text{tape}\ N\ \{r.\ r \hookrightarrow (N, [\,])\}}$$

$$\frac{}{\{r \hookrightarrow (N, n \cdot \vec{n})\}\ \text{rand}\ r\ N\ \{x.\ x = n * r \hookrightarrow (N, \vec{n})\}}$$

We can show programs with tapes behave the same as those without (proof + rules omitted)

$$\text{let } (x, y) = (\text{Rand } 1) \,|||\, (\text{Rand } 1) \text{ in } 2 \cdot x + y$$

$$\cong_{ctx}$$

$$\text{let } (r_1, r_2) = (\text{tape } 1, \text{tape } 1) \text{ in } \text{let } (x, y) = (\text{Rand } r_1\ 1) \,|||\, (\text{Rand } r_2\ 1) \text{ in } 2 \cdot x + y$$

$$\begin{array}{l} \text{let } (r_1, r_2) = (\text{tape } 1, \text{tape } 1) \text{ in} \\ \text{let } (x, y) = (\text{Rand } r_1\ 1) \parallel\parallel (\text{Rand } r_2\ 1) \text{ in} \\ \quad 2 \cdot x + y \end{array} \leq \text{Rand } 3$$

- We allocate tape resources $r_1 \hookrightarrow (1, [\,]) * r_2 \hookrightarrow (1, [\,])$

$$\begin{array}{l} \text{let } (r_1, r_2) = (\text{tape } 1, \text{tape } 1) \text{ in} \\ \text{let } (x,y) = (\text{Rand } r_1\ 1) \,|||\, (\text{Rand } r_2\ 1) \text{ in} \\ \quad 2 \cdot x + y \end{array} \quad \leq \quad \text{Rand } 3$$

- We allocate tape resources $r_1 \hookrightarrow (1, [\,]) * r_2 \hookrightarrow (1, [\,])$
- We apply following rule: $(f : \{0,1\} \times \{0,1\} \to \{0,1,2,3\})$

$$\frac{j \Rightarrow (K[\text{rand } 3]) \qquad r_1 \hookrightarrow (1, [\,]) \qquad r_2 \hookrightarrow (1, [\,]) \qquad f \text{ bijective}}{\Rightarrow \exists\, n \leq 1 .\, m \leq 1 .\; j \Rightarrow (K[f(n,m)]) * r_1 \hookrightarrow (1, [n]) * r_2 \hookrightarrow (1, [m])}$$

What about the right refines left direction?

Can we use the same trick with tapes?

What about the right refines left direction?

Can we use the same trick with tapes?

It turns out that presampling tapes on the right is UNSOUND!!

(though sound in the sequential case)

$$\text{Rand } 3 \preceq \text{let } (x,y) = (\text{Rand } 1) \,|||\, (\text{Rand } 1) \text{ in } 2 \cdot x + y$$

- After the parallel composition rule, we have

$$j_1 \Mapsto K_1(\text{Rand } 1) * j_2 \Mapsto K_2(\text{Rand } 1) * \ldots$$

$$\text{Rand } 3 \preceq \text{let } (x,y) = (\text{Rand } 1) \,|||\, (\text{Rand } 1) \text{ in } 2 \cdot x + y$$

- After the parallel composition rule, we have

$$j_1 \Mapsto K_1(\text{Rand } 1) * j_2 \Mapsto K_2(\text{Rand } 1) * \ldots$$

- We can apply: $(f : \{0,1\} \times \{0,1\} \to \{0,1,2,3\})$

$$\frac{f \text{ bijective} \qquad \forall n \leq 1.\, m \leq 1.\, \left\{ \begin{array}{l} j_1 \Mapsto K_1[n] \\ * \\ j_2 \Mapsto K_2[m] \end{array} \right\} f(n,m) \left\{ \Phi \right\}}{\left\{ \begin{array}{l} j_1 \Mapsto K_1[\text{Rand } 1] * \\ j_2 \Mapsto K_2[\text{Rand } 1] \end{array} \right\} \quad \text{rand } 3 \quad \left\{ \Phi \right\}}$$

Slogan 2:

Tape the left;

parallelize the right.

Exercise: Prove

$$\begin{array}{l} \text{let } x = \text{Rand } 1 \text{ in} \\ \quad \text{let } y = \text{Rand } 1 \text{ in} \\ \quad 2 \cdot x + y \end{array} \quad \simeq_{ctx} \quad \text{Rand } 3$$

Conclusion

- Logic for proving contextual refinement of concurrent probabilistic programs
- Inherits many rules from previous logics
  - This is a strong feature! (Not trivial to prove)
- More examples
  - Rejection samplers
  - Adversarial von Neumann Coin
  - Sodium sampling function
  - Equations of algebraic theory of $\preceq_{ctx}$