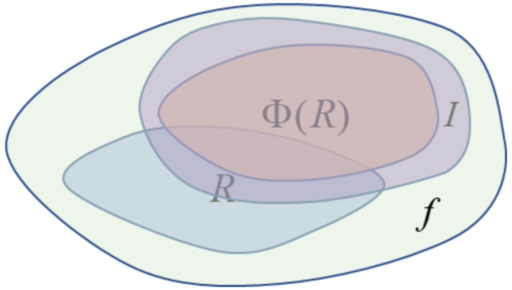
Latticed Craig Interpolation with an Application to Probabilistic Verification

Mingqi Yang, **Kevin Batz, Mingshuai Chen, Joost-Pieter Katoen, Zhiang Wu, and Jianwei Yin**

浙江大學 ZHEJIANG UNIVERSITY

RWTH AACHEN UNIVERSITY



Craig interpolation

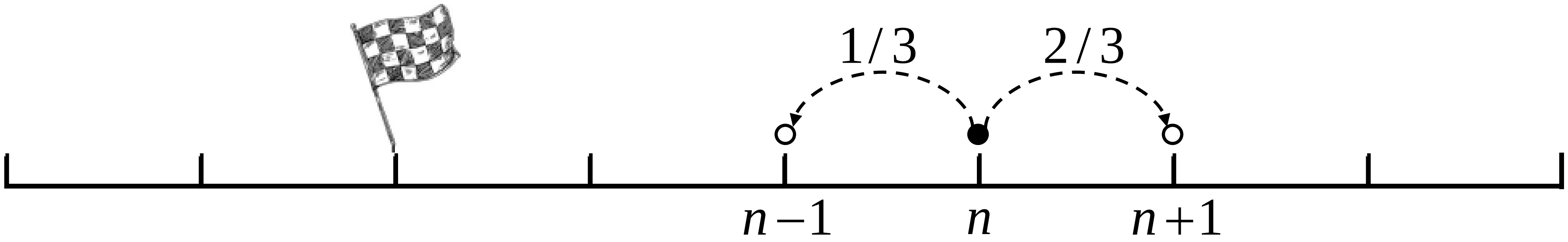
lattice structure

$$\texttt{while}\,(x \neq 0 \vee y \neq 0)$$
$$\{$$
$$x := x - 1 \quad 1/4$$
$$x := x + 1 \quad 1/4$$
$$y := y - 1 \quad 1/4$$
$$y := y + 1 \quad 1/4$$
$$\}$$

probabilistic programs

Probabilistic Programs

$$\mathit{while}\ (n > 0)\ \{\ n := n - 1\ [1/3]\ n := n + 1\ \}$$



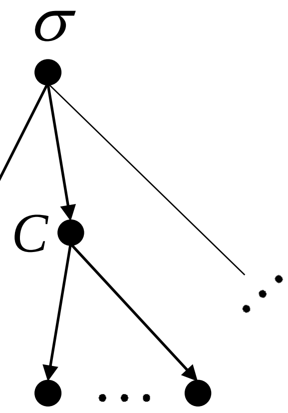
"The crux of probabilistic programming is to treat normal-looking programs as if they were probability distributions."

— Michael Hicks, The PL Enthusiast

ZHEJIANG UNIVERSITY

Quantitative Reasoning about Probabilistic Loops

Expected value of g (post expectation) evaluated in fanal states reached after executing C on σ



$$\boxed{\mathrm{wp}\ \mathrm{C}\ (g)(\sigma)} = \mathrm{Exp}\ [\ g(\tau_1)\quad g(\tau_2)\quad \cdots \quad g(\tau_3)\ ]$$

$$\mathrm{wp}[\![n := 5]\!](n) = 5$$

$$\mathrm{wp}[\![n := n-1\ [1/3]\ n := n+1]\!](n) = 1/3 \bullet (n-1) + 2/3 \bullet (n+1)$$

$$\mathrm{wp}[\![\ \mathit{while}(n > 0)\{n := n-1\ [1/3]\ n := n+1\}\ ]\!](n) = 1/3 \bullet (n-1) + 2/3 \bullet (n+1)$$

$$\mathrm{wp}\ \ \mathit{while}(\varphi)\{\mathrm{C}\}\ \ (g) = \mathrm{lfp}\Phi$$

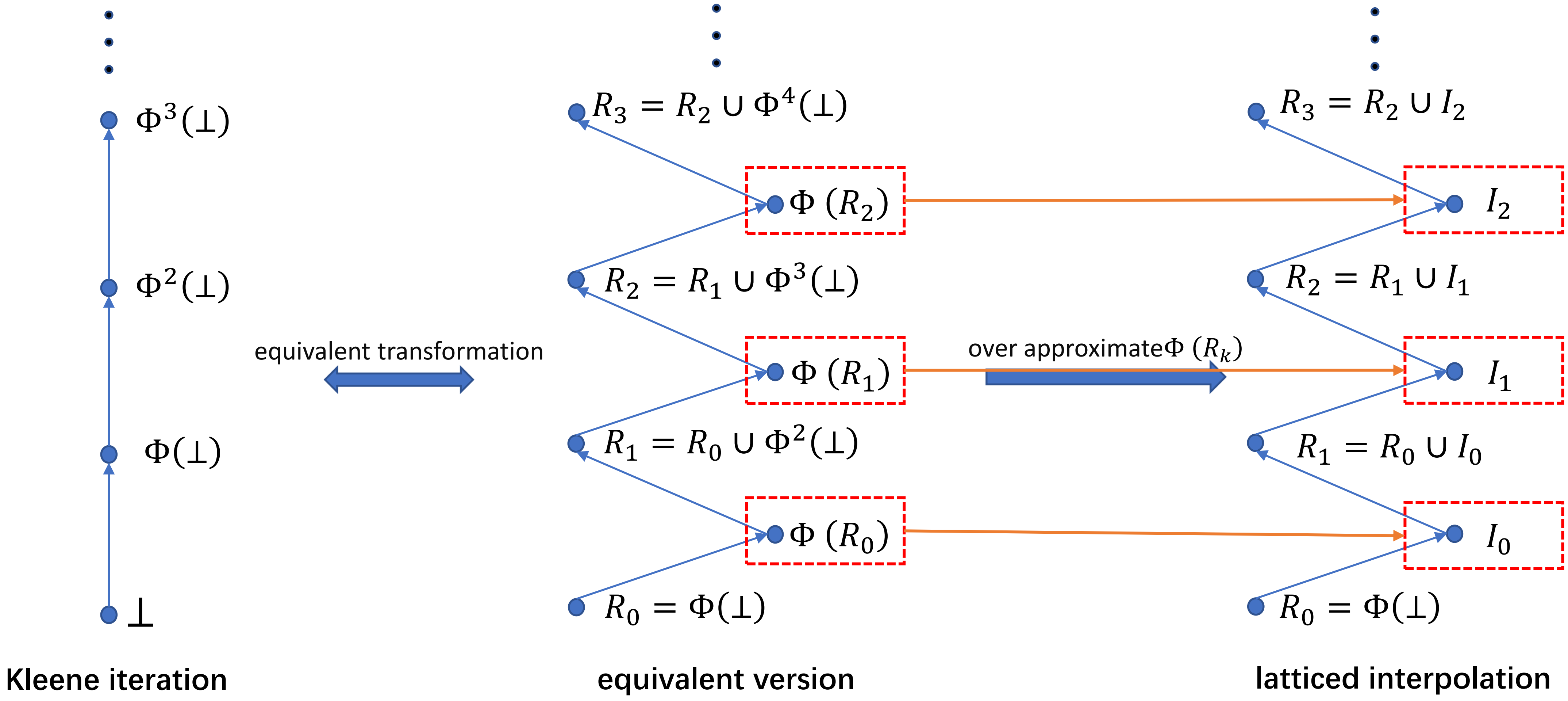
Bounding the Least Fixed Point

$$lfp\Phi \subseteq \boxed{\Phi(u) \subseteq u} \subseteq f$$

inductive invariant

candidate upper bound

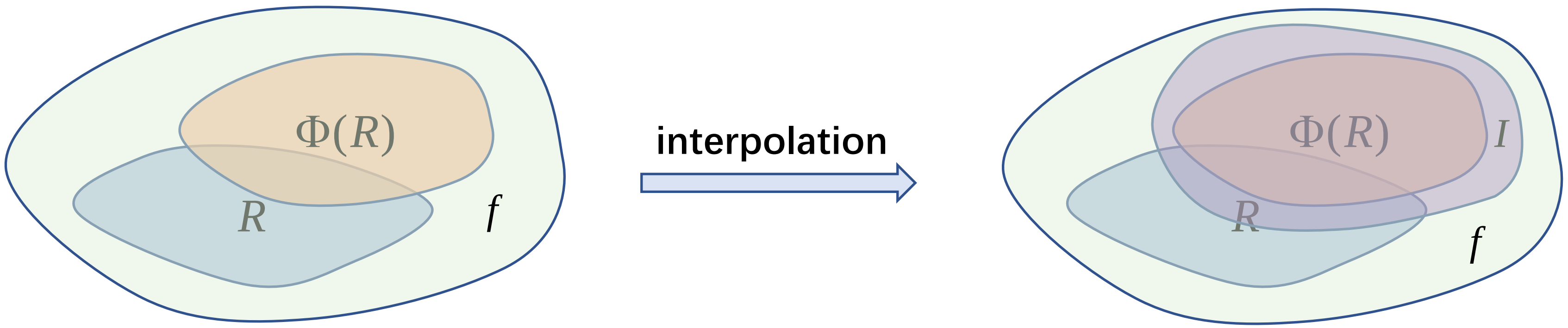
Core Idea


$\Phi^3(\perp)$
$\Phi^2(\perp)$
$\Phi(\perp)$
$\perp$
equivalent transformation
$R_3 = R_2 \cup \Phi^4(\perp)$
$\Phi\ (R_2)$
$R_2 = R_1 \cup \Phi^3(\perp)$
$\Phi\ (R_1)$
$R_1 = R_0 \cup \Phi^2(\perp)$
$\Phi\ (R_0)$
$R_0 = \Phi(\perp)$
over approximate$\Phi\ (R_k)$
$R_3 = R_2 \cup I_2$
I_2
$R_2 = R_1 \cup I_1$
I_1
$R_1 = R_0 \cup I_0$
I_0
$R_0 = \Phi(\perp)$
Kleene iteration
equivalent version
latticed interpolation

Latticed Craig Interpolation

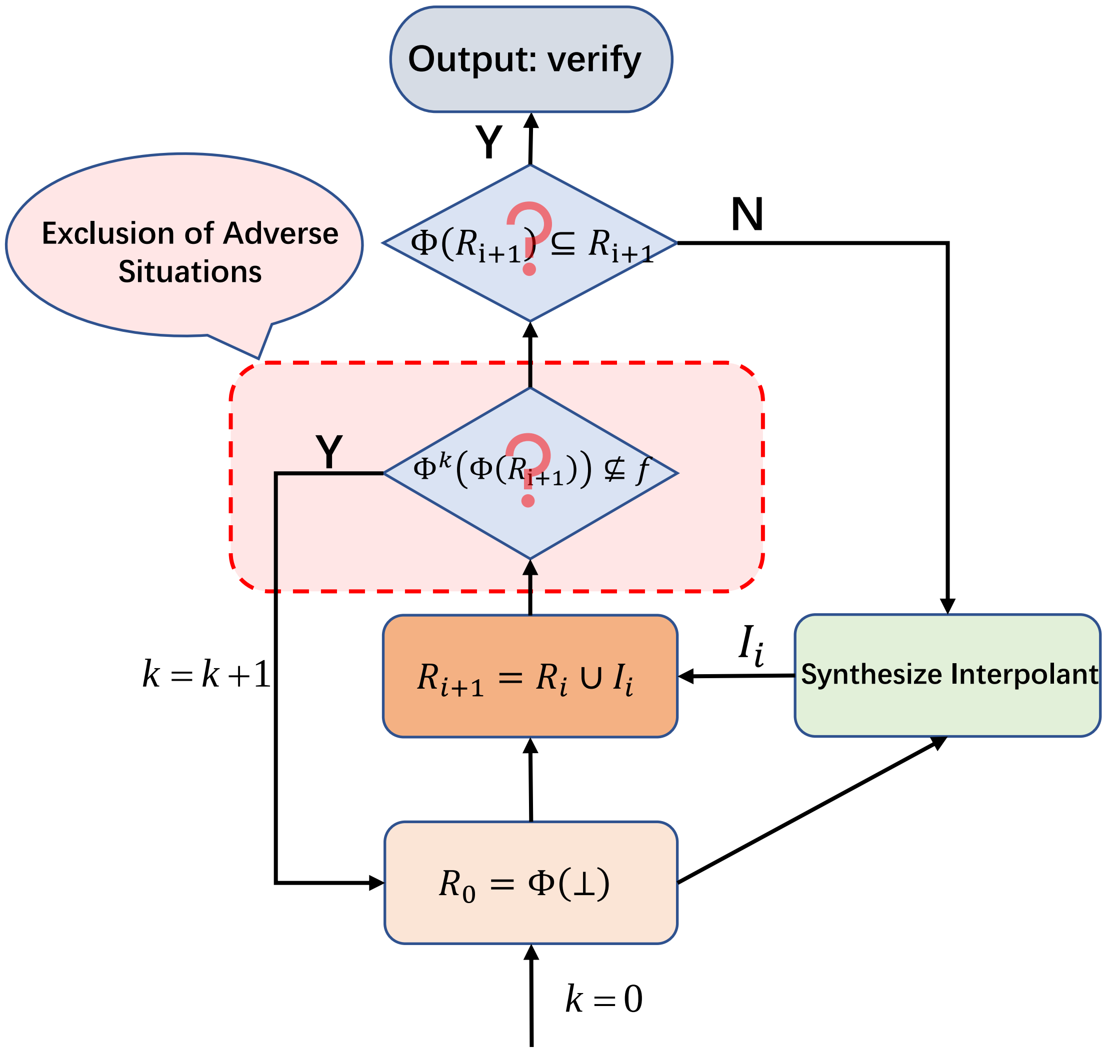
Definition (Latticed Craig Interpolant)

$$\Phi(R) \subseteq I \ \ and \ \ \forall m \in \{0, \cdots, k\} : \Phi^m(I) \subseteq f$$

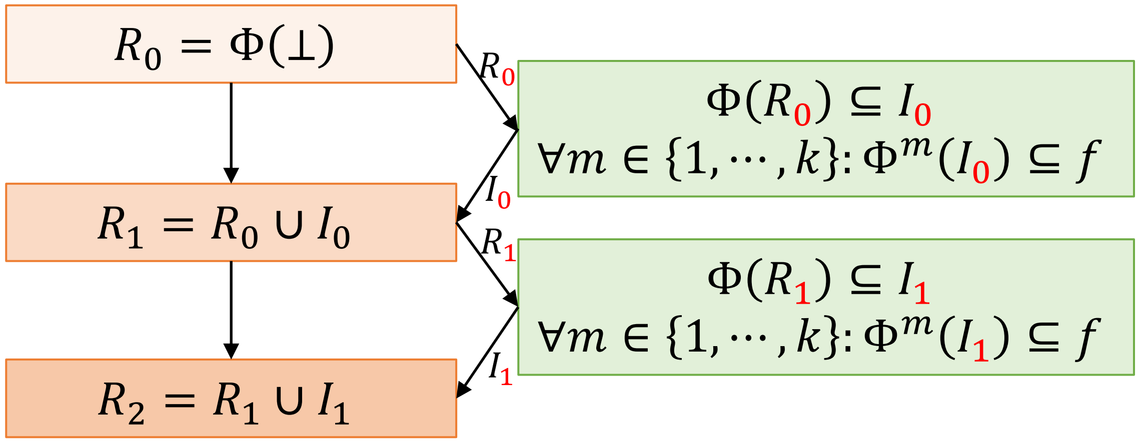


- The parameter k controls the quality of the generated interpolant (the larger, the better).

Latticed Craig Interpolation



- Interpolant accumulation



Example

```
nat c;
nat f;
while (f = 1) {
    {
        f := 0;
    } [0.5] {
        c := c + 1;
    }
}
```

$$post\ expectation = c$$
$$f = c + 1$$

```
([((not (((not (f = 1) & not (f = 1)) & (not (f = 1) & not (f = 1))) & (c <= c)) & ((((f = 1) & (f = 1)) & ((f = 1) & (f = 1))) & (((0 * 0.
5) + (0 * (1.0 - 0.5))) <= (((643/648 * c) + (11/48 * f)) + 43/72)))) & ((f = 1) & (f = 1))) & ((((643/648 * c) + (11/48 * f)) + 43/72) <=
((c + (3307/5184 * f)) + 1877/5184))] * ((c + (3307/5184 * f)) + 1877/5184)) + ([(((((not (f = 1) & not (f = 1)) & (not (f = 1) & not (f =
1))) & (c <= c)) & not ((((f = 1) & (f = 1)) & ((f = 1) & (f = 1))) & (((0 * 0.5) + (0 * (1.0 - 0.5))) <= (((643/648 * c) + (11/48 * f)) + 4
3/72)))) & (not (f = 1) & not (f = 1))) & (c <= c)] * c)
```

$R_2 = R_1 \cup I_1$ (inductive invariant)

$$I_1 = ([f=1] * (((1 * c) + \frac{3307}{5184} * f)) + \frac{1877}{5184})) + ([\neg(f=1)] * c)$$

$R_1 = R_0 \cup I_0$

$$I_0 = ([f=1] * (((\frac{643}{648} * c) + \frac{11}{48} * f)) + \frac{43}{72})) + ([\neg(f=1)] * c)$$

$$R_0 = \Phi(\bot)$$
$$= ([f=1] * 0) + ([\neg(f=1)] * c)$$

ZHEJIANG UNIVERSITY FICTION

Latticed Craig Interpolation

- **Soundness**: If Algorithm returns ⟨verify⟩, then R is an inductive invariant and $lfp\Phi \subseteq f$.

- **Completeness**: The algorithm terminates for any *finite* lattice and *distributive* operator.

- **Interpolant synthesis**: We use *template-based method* and *counterexample-guided synthesis* [Batz et al., TACAS 2023] to synthesize interpolants.

Summary

- **Quantitative Craig interpolants** by extending predicates to expectations.
- **Latticed Craig interpolation** by exploiting quantitative interpolants over complete lattices.
- **Soundness and Completeness**: Our latticed interpolation procedure is sound and complete (under some identified sufficient conditions).
- **Synthesizing quantitative interpolants:** A (semi-)automated synthesis approach by employing a counterexample-guided inductive synthesis framework.

Appendix

Distributivity:

Let $(E, \sqsubseteq)$ be a complete lattice. An operator $\Phi : E \rightarrow E$ is called distributive w.r.t. $\sqcup$ iff

$$\forall h1, h2 \in E : \Phi(h1 \sqcup h2) = \Phi(h1) \sqcup \Phi(h2) \,.$$