Latticed *k*-Induction

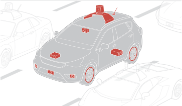
with an Application to Probabilistic Programs

Mingshuai Chen

—Joint work with K. Batz, B. L. Kaminski, J.-P. Katoen, C. Matheja, P. Schröer—

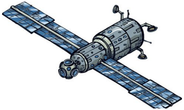
Cyber-Physical Systems (CPS)

An open, interconnected form of embedded systems that integrates capabilities of *computing*, *communication*, and *control*, among which many are safety-critical.



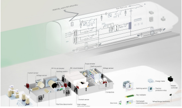
©TheOneBrief

automobiles



©clipartlogo

spacecrafts

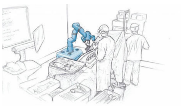


©Sécheron

high-speed rail

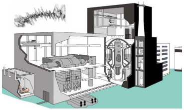
©VectorStock

nuclear reactors



©Scoop.it

robot surgeon

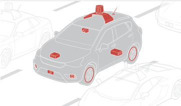


©KIT

robust control

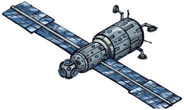
Cyber-Physical Systems (CPS)

An open, interconnected form of embedded systems that integrates capabilities of *computing*, *communication*, and *control*, among which many are safety-critical.



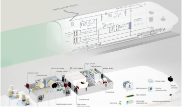
©TheOneBrief

automobiles



©clipartlogo

spacecrafts

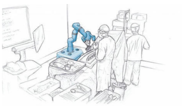


©Sécheron

high-speed rail

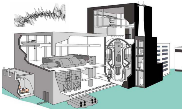
©VectorStock

nuclear reactors



©Scoop.it

robot surgeon



©KIT

robust control

"How can we provide people with CPS they can bet their lives on?"

— Jeannette Wing

Formal Methods

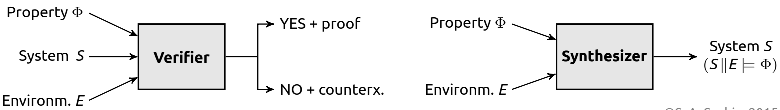
"By a formal method we shall understand a method whose techniques and tools can be explained in mathematics."

[D. Bjørner & K. Havelund, FM '14]

Formal Methods

"By a formal method we shall understand a method whose techniques and tools can be explained in mathematics."

[D. Bjørner & K. Havelund, FM'14]



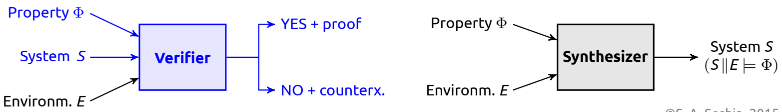
©S. A. Seshia, 2015

Develop mathematically rigorous techniques for designing safety-critical CPS while *pushing the limits of automation* as far as possible.

Formal Methods

"By a formal method we shall understand a method whose techniques and tools can be explained in mathematics."

[D. Bjørner & K. Havelund, FM'14]



©S. A. Seshia, 2015

Develop mathematically rigorous techniques for designing safety-critical CPS while *pushing the limits of automation* as far as possible.

k-Induction [Sheeran et al., FMCAD '00]

- SAT-based technique for verifying invariant properties of finite transition systems.

k-Induction [Sheeran et al., FMCAD '00]

- SAT-based technique for verifying invariant properties of finite transition systems.
- Later : verification of infinite-state transition systems via SMT solving.

k-Induction [Sheeran et al., FMCAD '00]

- SAT-based technique for verifying invariant properties of finite transition systems.
- Later : verification of infinite-state transition systems via SMT solving.
- Applications : hardware- and software model checking.

k-Induction [Sheeran et al., FMCAD '00]

- SAT-based technique for verifying invariant properties of finite transition systems.
- Later : verification of infinite-state transition systems via SMT solving.
- Applications : hardware- and software model checking.

> *"The simplicity of applying k-induction made it the go-to technique for SMT-based infinite-state model checking."*
>
> [Krishnan et al., CAV '19]

k-Induction [Sheeran et al., FMCAD '00]

- SAT-based technique for verifying invariant properties of finite transition systems.
- Later : verification of infinite-state transition systems via SMT solving.
- Applications : hardware- and software model checking.

> *"The simplicity of applying k-induction made it the go-to technique for SMT-based infinite-state model checking."*
>
> [Krishnan et al., CAV '19]

Is k-induction applicable to verifying infinite-state probabilistic programs?

k-Induction [Sheeran et al., FMCAD '00]

- SAT-based technique for verifying invariant properties of finite transition systems.
- Later : verification of infinite-state transition systems via SMT solving.
- Applications : hardware- and software model checking.

> *"The simplicity of applying k-induction made it the go-to technique for SMT-based infinite-state model checking."*
>
> [Krishnan et al., CAV '19]

Is k-induction applicable to verifying infinite-state probabilistic programs?

$\Downarrow$ **Latticed k-Induction**

Yes. It enables fully automatic verification of non-trivial properties.

k-Induction for Probabilistic Programs — Intuition

For a probabilistic loop C:

$$\texttt{while}\,(\,c = 1\,)\,\{\,c \coloneqq 0\,[\,1/2\,]\,x \coloneqq x + 1\,\}\,,$$

k-Induction for Probabilistic Programs — Intuition

For a probabilistic loop C:

$$\texttt{while}\,(\,c = 1\,)\,\{\,c \coloneqq 0\,[\,1/2\,]\,x \coloneqq x + 1\,\}\,,$$

the property

$$\forall \text{ initial state } \sigma: \quad \mathrm{wp}[\![C]\!]\,(x)\,(\sigma) \;\leq\; \sigma(x) + 1$$

k-Induction for Probabilistic Programs — Intuition

For a probabilistic loop C:

$$\texttt{while}\,(\,c = 1\,)\,\{\,c := 0\,[\,1/2\,]\,x := x + 1\,\}\,,$$

the property

$$\forall\ \text{initial state}\ \sigma: \quad \mathrm{wp}[\![C]\!]\,(x)\,(\sigma)\ \leq\ \sigma(x) + 1$$

is not inductive but 2-inductive.

k-Induction for Transition Systems

Given : TS $= (S, I, T)$, invariant property $P \subseteq S$.

Goal : Prove that P covers all *reachable states* of TS .

k-Induction for Transition Systems

Given : TS $= (S, I, T)$, invariant property $P \subseteq S$.

Goal : Prove that P covers all *reachable states* of TS .

Let $k \geq 1$. If the following two formulae are valid

$$\underbrace{I(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge T(s_{k-1}, s_k)}_{\text{all states reachable within } k-1 \text{ steps}} \implies \underbrace{P(s_1) \wedge \ldots \wedge P(s_k)}_{\text{are } P\text{-states}} \qquad \text{[base case]}$$

k-Induction for Transition Systems

Given : TS $= (S, I, T)$, invariant property $P \subseteq S$.

Goal : Prove that P covers all *reachable states* of TS .

Let $k \geq 1$. If the following two formulae are valid

$$\underbrace{I(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge T(s_{k-1}, s_k)}_{\text{all states reachable within } k-1 \text{ steps}} \implies \underbrace{P(s_1) \wedge \ldots \wedge P(s_k)}_{\text{are } P\text{-states}} \qquad \text{[base case]}$$

$$\underbrace{P(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge P(s_k)}_{\text{assume staying in } P \text{ for } k-1 \text{ steps}} \wedge \underbrace{T(s_k, s_{k+1})}_{\text{after step } k} \implies \underbrace{P(s_{k+1})}_{\text{end up in } P \text{ again}} \qquad \text{[ind. step]}$$

k-Induction for Transition Systems

Given : TS $= (S, I, T)$, invariant property $P \subseteq S$.

Goal : Prove that P covers all *reachable states* of TS .

Let $k \geq 1$. If the following two formulae are valid

$$\underbrace{I(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge T(s_{k-1}, s_k)}_{\text{all states reachable within } k-1 \text{ steps}} \implies \underbrace{P(s_1) \wedge \ldots \wedge P(s_k)}_{\text{are } P\text{-states}} \qquad \text{[base case]}$$

$$\underbrace{P(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge P(s_k)}_{\text{assume staying in } P \text{ for } k-1 \text{ steps}} \wedge \underbrace{T(s_k, s_{k+1})}_{\text{after step } k} \implies \underbrace{P(s_{k+1})}_{\text{end up in } P \text{ again}} \qquad \text{[ind. step]}$$

then P is a k-inductive invariant covering all reachable states of TS.

k-Induction for Transition Systems

Given : TS $= (S, I, T)$, invariant property $P \subseteq S$.

Goal : Prove that P covers all *reachable states* of TS .

Let $k \geq 1$. If the following two formulae are valid

$$\underbrace{I(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge T(s_{k-1}, s_k)}_{\text{all states reachable within } k-1 \text{ steps}} \implies \underbrace{P(s_1) \wedge \ldots \wedge P(s_k)}_{\text{are } P\text{-states}} \qquad \text{[base case]}$$

$$\underbrace{P(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge P(s_k)}_{\text{assume staying in } P \text{ for } k-1 \text{ steps}} \wedge \underbrace{T(s_k, s_{k+1})}_{\text{after step } k} \implies \underbrace{P(s_{k+1})}_{\text{end up in } P \text{ again}} \qquad \text{[ind. step]}$$

then P is a k-inductive invariant covering all reachable states of TS.

For verifying *probabilistic programs*, we have to

- leave the Boolean domain and reason about quantities;

k-Induction for Transition Systems

Given : TS $= (S, I, T)$, invariant property $P \subseteq S$.

Goal : Prove that P covers all *reachable states* of TS .

Let $k \geq 1$. If the following two formulae are valid

$$\underbrace{I(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge T(s_{k-1}, s_k)}_{\text{all states reachable within } k-1 \text{ steps}} \implies \underbrace{P(s_1) \wedge \ldots \wedge P(s_k)}_{\text{are } P\text{-states}} \qquad \text{[base case]}$$

$$\underbrace{P(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge P(s_k)}_{\text{assume staying in } P \text{ for } k-1 \text{ steps}} \wedge \underbrace{T(s_k, s_{k+1})}_{\text{after step } k} \implies \underbrace{P(s_{k+1})}_{\text{end up in } P \text{ again}} \qquad \text{[ind. step]}$$

then P is a k-inductive invariant covering all reachable states of TS.

For verifying *probabilistic programs*, we have to

- leave the Boolean domain and reason about quantities ;
- reason about sets of paths rather than individual paths.

Idea Sketch

k-induction for TS
in terms of a **SAT problem**

Idea Sketch

k-induction for TS
in terms of **fixed points**

↑

k-induction for TS
in terms of a **SAT problem**

Idea Sketch

Latticed k-induction for proving $\mathrm{lfp}\ \Phi \sqsubseteq f$

$\uparrow$ abstract

k-induction for TS
in terms of **fixed points**

$\uparrow$

k-induction for TS
in terms of a **SAT problem**

Idea Sketch

Latticed *k*-induction for proving $\mathrm{lfp}\ \Phi \sqsubseteq f$

abstract

k-induction for TS
in terms of **fixed points**

k-induction for TS
in terms of a **SAT problem**

Whenever your problem boils down to verifying an upper bound on a lfp ,
latticed *k*-induction provides you with **inductive proof rules**!

Idea Sketch

Latticed *k*-induction for proving $\mathrm{lfp}\ \Phi \sqsubseteq f$

abstract

instantiate

k-induction for TS
in terms of **fixed points**

k-induction for **probabilistic programs**
in terms of **fixed points**

k-induction for TS
in terms of a **SAT problem**

Whenever your problem boils down to verifying an upper bound on a lfp ,
latticed *k*-induction provides you with **inductive proof rules**!

Idea Sketch

Latticed *k*-induction for proving lfp $\Phi \sqsubseteq f$

abstract

instantiate

k-induction for TS
in terms of **fixed points**

k-induction for **probabilistic programs**
in terms of **fixed points**

k-induction for TS
in terms of a **SAT problem**

k-induction for **probabilistic programs**
in terms of a **SAT problem**

Whenever your problem boils down to verifying an upper bound on a lfp ,
latticed *k*-induction provides you with **inductive proof rules**!

Idea Sketch

Latticed *k*-induction for proving $\text{lfp}\ \Phi \sqsubseteq f$

abstract

instantiate

k-induction for TS
in terms of **fixed points**

k-induction for **probabilistic programs**
in terms of **fixed points**

k-induction for TS
in terms of a **SAT problem**

k-induction for **probabilistic programs**
in terms of a **SAT problem**

Whenever your problem boils down to verifying an upper bound on a lfp ,
latticed *k*-induction provides you with **inductive proof rules**!

Latticed k-Induction

Given : Complete lattice $(E, \sqsubseteq)$, monotonic operator $\Phi\colon\ E \to E$, and candidate $f \in E$.

Goal : Prove $\text{lfp}\ \Phi \sqsubseteq f$.

Latticed k-Induction

Given : Complete lattice $(E, \sqsubseteq)$, monotonic operator $\Phi\colon\ E \to E$, and candidate $f \in E$.
Goal : Prove $\mathsf{lfp}\ \Phi \sqsubseteq f$.

Park induction (aka 1-induction) :

$$\Phi(f) \sqsubseteq f \quad \text{implies} \quad \mathsf{lfp}\ \Phi \sqsubseteq f.$$

Latticed k-Induction

Given : Complete lattice $(E, \sqsubseteq)$, monotonic operator $\Phi \colon E \to E$, and candidate $f \in E$.
Goal : Prove $\mathsf{lfp}\, \Phi \sqsubseteq f$.

Park induction (aka 1-induction) :

$$\Phi(f) \sqsubseteq f \quad \text{implies} \quad \mathsf{lfp}\, \Phi \sqsubseteq f.$$

Even though $\mathsf{lfp}\, \Phi \sqsubseteq f$ we might have $\Phi(f) \not\sqsubseteq f$!

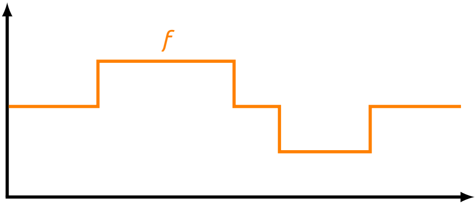
Latticed k-Induction

Given : Complete lattice $(E, \sqsubseteq)$, monotonic operator $\Phi\colon E \to E$, and candidate $f \in E$.
Goal : Prove $\mathsf{lfp}\, \Phi \sqsubseteq f$.

Park induction (aka 1-induction) :

$$\Phi(f) \sqsubseteq f \quad \text{implies} \quad \mathsf{lfp}\,\Phi \sqsubseteq f.$$

Even though $\mathsf{lfp}\,\Phi \sqsubseteq f$ we might have $\Phi(f) \not\sqsubseteq f$!



Latticed k-Induction

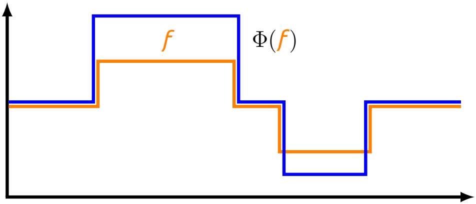
Given : Complete lattice $(E, \sqsubseteq)$, monotonic operator $\Phi\colon E \to E$, and candidate $f \in E$.

Goal : Prove $\mathsf{lfp}\,\Phi \sqsubseteq f$.

Park induction (aka 1-induction) :

$$\Phi(f) \sqsubseteq f \quad \text{implies} \quad \mathsf{lfp}\,\Phi \sqsubseteq f.$$

Even though $\mathsf{lfp}\,\Phi \sqsubseteq f$ we might have $\Phi(f) \not\sqsubseteq f$!



Latticed k-Induction

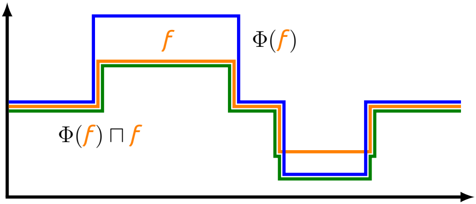
Given : Complete lattice $(E, \sqsubseteq)$, monotonic operator $\Phi\colon E \to E$, and candidate $f \in E$.

Goal : Prove $\mathsf{lfp}\ \Phi \sqsubseteq f$.

Park induction (aka 1-induction) :

$$\Phi(f) \sqsubseteq f \quad \text{implies} \quad \mathsf{lfp}\ \Phi \sqsubseteq f.$$

Even though $\mathsf{lfp}\ \Phi \sqsubseteq f$ we might have $\Phi(f) \not\sqsubseteq f$!



Latticed k-Induction

Given : Complete lattice $(E, \sqsubseteq)$, monotonic operator $\Phi\colon E \to E$, and candidate $f \in E$.

Goal : Prove $\text{lfp}\,\Phi \sqsubseteq f$.

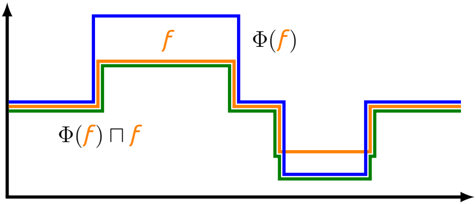
Park induction (aka 1-induction) :

$$\Phi(f) \sqsubseteq f \quad \text{implies} \quad \text{lfp}\,\Phi \sqsubseteq f.$$

Even though $\text{lfp}\,\Phi \sqsubseteq f$ we might have $\Phi(f) \not\sqsubseteq f$!

2-induction :

$$\Phi(\Phi(f) \sqcap f) \sqsubseteq f \quad \text{implies} \quad \text{lfp}\,\Phi \sqsubseteq f.$$



Latticed k-Induction

Given : Complete lattice $(E, \sqsubseteq)$, monotonic operator $\Phi \colon E \to E$, and candidate $f \in E$.
Goal : Prove $\mathrm{lfp}\,\Phi \sqsubseteq f$.

Park induction (aka 1-induction) :

$$\Phi(f) \sqsubseteq f \quad \text{implies} \quad \mathrm{lfp}\,\Phi \sqsubseteq f.$$

Even though $\mathrm{lfp}\,\Phi \sqsubseteq f$ we might have $\Phi(f) \not\sqsubseteq f$!

2-induction :

$$\Phi(\Phi(f) \sqcap f) \sqsubseteq f \quad \text{implies} \quad \mathrm{lfp}\,\Phi \sqsubseteq f.$$

3-induction :

$$\Phi(\Phi(\Phi(f) \sqcap f) \sqcap f) \sqsubseteq f \quad \text{implies} \quad \mathrm{lfp}\,\Phi \sqsubseteq f.$$

Latticed k-Induction

Given : Complete lattice $(E, \sqsubseteq)$, monotonic operator $\Phi\colon E \to E$, and candidate $f \in E$.
Goal : Prove $\mathsf{lfp}\, \Phi \sqsubseteq f$.

Define k-induction operator $\Psi_f\colon E \to E$ by

$$\Psi_f(g) = \Phi(g) \sqcap f .$$

Latticed k-Induction

Given : Complete lattice $(E, \sqsubseteq)$, monotonic operator $\Phi\colon E \to E$, and candidate $f \in E$.
Goal : Prove $\mathsf{lfp}\, \Phi \sqsubseteq f$.

Define k-induction operator $\Psi_f\colon E \to E$ by

$$\Psi_f(g) = \Phi(g) \sqcap f.$$

Theorem (Latticed k-induction)

For every $k \geq 1$,

$$\Phi\left(\Psi_f^{k-1}(f)\right) \sqsubseteq f \quad \text{implies} \quad \mathsf{lfp}\, \Phi \sqsubseteq f.$$

We call such f k-inductive invariant.

Latticed k-Induction

Given : Complete lattice $(E, \sqsubseteq)$, monotonic operator $\Phi\colon E \to E$, and candidate $f \in E$.
Goal : Prove $\operatorname{lfp} \Phi \sqsubseteq f$.

Define k-induction operator $\Psi_f\colon E \to E$ by

$$\Psi_f(g) = \Phi(g) \sqcap f .$$

Theorem (Latticed k-induction)

For every $k \geq 1$,

$$\Phi\left(\Psi_f^{k-1}(f)\right) \sqsubseteq f \quad \text{implies} \quad \operatorname{lfp} \Phi \sqsubseteq f .$$

We call such f k-inductive invariant.

k-induction generalizes Park induction $\widehat{=}$ 1-induction.

Latticed k-Induction

Given : Complete lattice $(E, \sqsubseteq)$, monotonic operator $\Phi\colon E \to E$, and candidate $f \in E$.
Goal : Prove lfp $\Phi \sqsubseteq f$.

Define k-induction operator $\Psi_f\colon E \to E$ by

$$\Psi_f(g) = \Phi(g) \sqcap f\,.$$

Theorem (Latticed k-induction)

For every $k \geq 1$,

$$\Phi\left(\Psi_f^{k-1}(f)\right) \sqsubseteq f \quad \text{implies} \quad \operatorname{lfp} \Phi \sqsubseteq f\,.$$

We call such f k-inductive invariant.

k-induction generalizes Park induction $\widehat{=}$ 1-induction.
Can be further generalized to *transfinite κ-induction* (not in this talk).

Key Insights of Soundness

Lemma (Descending chain)

Iterating Ψ_f on f yields a descending chain, i.e.,

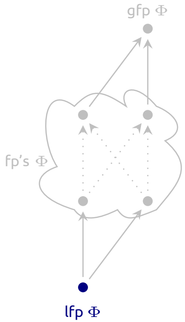
$$f \sqsupseteq \Psi_f^1(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \ldots .$$

Key Insights of Soundness

Lemma (Descending chain)

Iterating Ψ_f on f yields a descending chain, i.e.,

$$f \sqsupseteq \Psi_f^1(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \ldots .$$

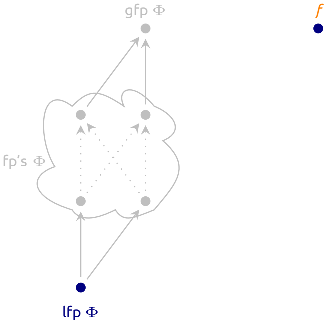


Key Insights of Soundness

Lemma (Descending chain)

Iterating Ψ_f on f yields a descending chain, i.e.,

$$f \sqsupseteq \Psi_f^1(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \ldots .$$

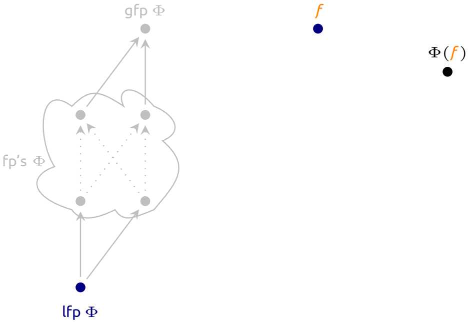


Key Insights of Soundness

Lemma (Descending chain)

Iterating Ψ_f on f yields a descending chain, i.e.,

$$f \sqsupseteq \Psi_f^1(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \ldots .$$

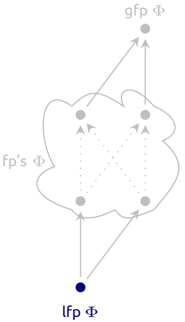


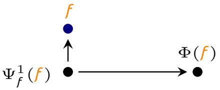
Key Insights of Soundness

Lemma (Descending chain)

Iterating Ψ_f on f yields a descending chain, i.e.,

$$f \sqsupseteq \Psi_f^1(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \ldots .$$



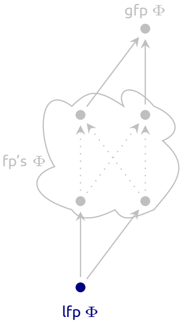


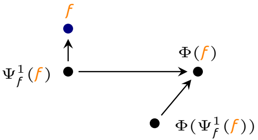
Key Insights of Soundness

Lemma (Descending chain)

Iterating Ψ_f on f yields a descending chain, i.e.,

$$f \sqsupseteq \Psi_f^1(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \ldots .$$



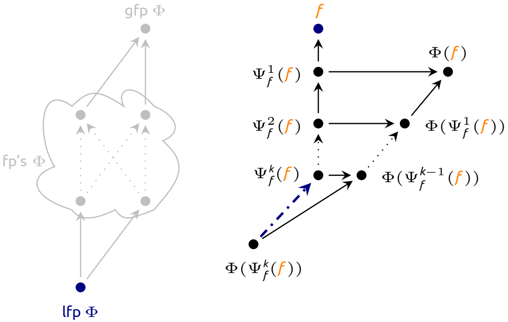


Key Insights of Soundness

Lemma (Descending chain)

Iterating Ψ_f on f yields a descending chain, i.e.,

$$f \sqsupseteq \Psi_f^1(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \ldots .$$

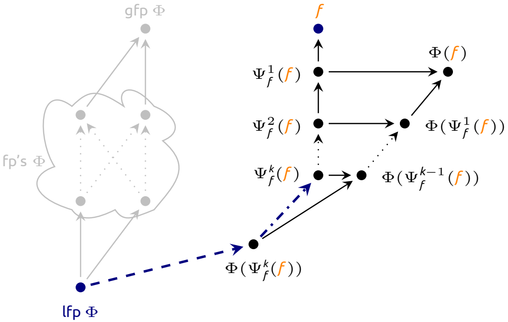


Key Insights of Soundness

Lemma (Descending chain)

Iterating Ψ_f on f yields a descending chain, i.e.,

$$f \sqsupseteq \Psi_f^1(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \ldots .$$

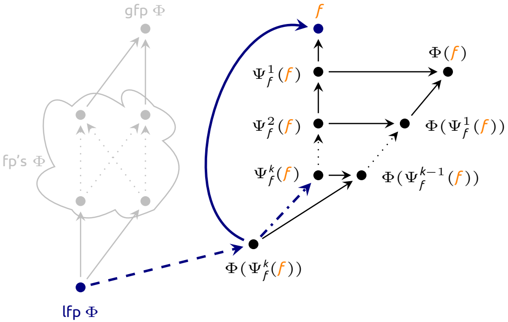


Key Insights of Soundness

Lemma (Descending chain)

Iterating Ψ_f on f yields a descending chain, i.e.,

$$f \sqsupseteq \Psi_f^1(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \ldots .$$



Key Insights of Soundness

Lemma (Descending chain)

Iterating Ψ_f on f yields a descending chain, i.e.,

$$f \sqsupseteq \Psi_f^1(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \ldots .$$

Theorem (Park induction from k-induction)

$$\underbrace{\Phi\left(\Psi_f^{k-1}(f)\right) \sqsubseteq f}_{f \text{ is k-inductive invariant}} \quad \textit{iff} \quad \underbrace{\Phi\left(\Psi_f^{k-1}(f)\right) \sqsubseteq \Psi_f^{k-1}(f)}_{\Psi_f^{k-1}(f) \text{ is inductive invariant}}$$

Key Insights of Soundness

Lemma (Descending chain)

Iterating Ψ_f on f yields a descending chain, i.e.,

$$f \sqsupseteq \Psi_f^1(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \ldots .$$

Theorem (Park induction from k-induction)

$$\underbrace{\Phi\left(\Psi_f^{k-1}(f)\right) \sqsubseteq f}_{f \text{ is } k\text{-inductive invariant}} \quad \textit{iff} \quad \underbrace{\Phi\left(\Psi_f^{k-1}(f)\right) \sqsubseteq \Psi_f^{k-1}(f)}_{\Psi_f^{k-1}(f) \text{ is inductive invariant}}$$

f is a k-inductive invariant $\iff$ $\Psi_f^{k-1}(f)$ is an inductive invariant *stronger than* f.

Idea Sketch

Latticed k-induction for proving $\mathrm{lfp}\ \Phi \sqsubseteq f$

abstract

instantiate

k-induction for TS
in terms of **fixed points**

k-induction for **probabilistic programs**
in terms of **fixed points**

k-induction for TS
in terms of a **SAT problem**

k-induction for **probabilistic programs**
in terms of a **SAT problem**

Weakest Preexpectation Transformer

Consider the complete lattice $(\mathbb{E}, \leq)$ of *expectations* :

Weakest Preexpectation Transformer

Consider the complete lattice $(\mathbb{E}, \leq)$ of *expectations* :

$$\mathbb{E} = \{f \mid f \colon \Sigma \to \mathbb{R}_{\geq 0}^{\infty}\}$$

Weakest Preexpectation Transformer

Consider the complete lattice $(\mathbb{E}, \leq)$ of *expectations* :

$$\mathbb{E} = \{f \mid f\colon \Sigma \to \mathbb{R}_{\geq 0}^{\infty}\} \qquad \text{with} \qquad f \leq g \quad \text{iff} \quad \forall \sigma \in \Sigma\colon f(\sigma) \leq g(\sigma)\,.$$

Weakest Preexpectation Transformer

Consider the complete lattice $(\mathbb{E}, \leq)$ of *expectations* :

$$\mathbb{E} = \{f \mid f\colon \Sigma \to \mathbb{R}_{\geq 0}^{\infty}\} \qquad \text{with} \qquad f \leq g \quad \text{iff} \quad \forall \sigma \in \Sigma\colon f(\sigma) \leq g(\sigma)\,.$$

Weakest preexpectation transformer [Kozen '83, McIver '99, McIver & Morgan '05] :

$$\mathrm{wp}[\![C]\!] : \mathbb{E} \to \mathbb{E}\,,$$

Weakest Preexpectation Transformer

Consider the complete lattice $(\mathbb{E}, \leq)$ of *expectations*:

$$\mathbb{E} = \{f \mid f\colon \Sigma \to \mathbb{R}_{\geq 0}^{\infty}\} \qquad \text{with} \qquad f \leq g \quad \text{iff} \quad \forall \sigma \in \Sigma\colon f(\sigma) \leq g(\sigma)\,.$$

Weakest preexpectation transformer [Kozen '83, McIver '99, McIver & Morgan '05]:

$$\mathsf{wp}[\![C]\!]\colon \mathbb{E} \to \mathbb{E}\,, \qquad \mathsf{wp}[\![C]\!]\,(g)\,(\sigma) \;\widehat{=}\; \text{expected value of } g \text{ evaluated in final states reached after executing } C \text{ on } \sigma\,.$$

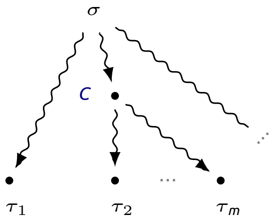
Weakest Preexpectation Transformer

Consider the complete lattice $(\mathbb{E}, \leq)$ of *expectations* :

$$\mathbb{E} = \{f \mid f\colon \Sigma \to \mathbb{R}_{\geq 0}^{\infty}\} \qquad \text{with} \qquad f \leq g \quad \text{iff} \quad \forall \sigma \in \Sigma\colon f(\sigma) \leq g(\sigma)\,.$$

Weakest preexpectation transformer [Kozen '83, McIver '99, McIver & Morgan '05] :

$$\mathsf{wp}[\![C]\!]\colon \mathbb{E} \to \mathbb{E}\,, \qquad \mathsf{wp}[\![C]\!]\,(g)\,(\sigma) \;\widehat{=}\; \text{expected value of } g \text{ evaluated in final states reached after executing } C \text{ on } \sigma\,.$$



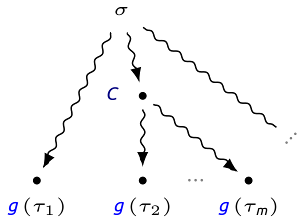
Weakest Preexpectation Transformer

Consider the complete lattice $(\mathbb{E}, \leq)$ of *expectations*:

$$\mathbb{E} = \{f \mid f\colon \Sigma \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{with} \qquad f \leq g \quad \text{iff} \quad \forall \sigma \in \Sigma\colon f(\sigma) \leq g(\sigma)\,.$$

Weakest preexpectation transformer [Kozen '83, McIver '99, McIver & Morgan '05]:

$$\mathrm{wp}[\![C]\!]\colon \mathbb{E} \to \mathbb{E}\,, \qquad \mathrm{wp}[\![C]\!]\,(g)\,(\sigma) \;\widehat{=}\;$$ expected value of g evaluated in final states reached after executing C on σ.



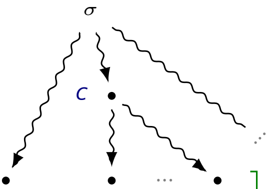
Weakest Preexpectation Transformer

Consider the complete lattice $(\mathbb{E}, \leq)$ of *expectations* :

$$\mathbb{E} = \{f \mid f\colon \Sigma \to \mathbb{R}_{\geq 0}^{\infty}\} \qquad \text{with} \qquad f \leq g \quad \text{iff} \quad \forall \sigma \in \Sigma\colon f(\sigma) \leq g(\sigma)\,.$$

Weakest preexpectation transformer [Kozen '83, McIver '99, McIver & Morgan '05] :

$$\mathrm{wp}[\![C]\!]\colon \mathbb{E} \to \mathbb{E}\,, \qquad \mathrm{wp}[\![C]\!]\,(g)\,(\sigma) \;\widehat{=}\; \text{expected value of } g \text{ evaluated in final states reached after executing } C \text{ on } \sigma\,.$$



$$\mathrm{wp}[\![C]\!](g)(\sigma) \;\widehat{=}\; \mathrm{Exp}\left[\; g(\tau_1) \qquad g(\tau_2) \quad \cdots \quad g(\tau_m) \;\right]$$

Weakest Preexpectation Transformer

Consider the complete lattice $(\mathbb{E}, \leq)$ of *expectations*:

$$\mathbb{E} = \{f \mid f\colon \Sigma \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{with} \qquad f \leq g \quad \text{iff} \quad \forall \sigma \in \Sigma\colon f(\sigma) \leq g(\sigma)\,.$$

Weakest preexpectation transformer [Kozen '83, McIver '99, McIver & Morgan '05]:

$$\mathsf{wp}[\![C]\!]\colon \mathbb{E} \to \mathbb{E}\,, \qquad \mathsf{wp}[\![C]\!]\,(g)\,(\sigma) \;\widehat{=}\; \text{expected value of } g \text{ evaluated in final states reached after executing } C \text{ on } \sigma\,.$$

$$\mathsf{wp}[\![x := 5]\!]\,(x) \;=\; 5$$

Weakest Preexpectation Transformer

Consider the complete lattice $(\mathbb{E}, \leq)$ of *expectations* :

$$\mathbb{E} = \{f \mid f\colon \Sigma \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{with} \qquad f \leq g \quad \text{iff} \quad \forall \sigma \in \Sigma\colon f(\sigma) \leq g(\sigma)\,.$$

Weakest preexpectation transformer [Kozen '83, McIver '99, McIver & Morgan '05] :

$$\mathsf{wp}[\![C]\!]\colon \mathbb{E} \to \mathbb{E}\,, \qquad \mathsf{wp}[\![C]\!]\,(g)\,(\sigma) \;\widehat{=}\; \text{expected value of } g \text{ evaluated in final states reached after executing } C \text{ on } \sigma\,.$$

$$\mathsf{wp}[\![x := 5]\!]\,(x) \;=\; 5$$

$$\mathsf{wp}[\![\{\,\mathtt{skip}\,\}\;[\,1/2\,]\;\{\,x := x + 2\,\}]\!]\,(x) \;=\; \frac{1}{2}\cdot x + \frac{1}{2}\cdot (x+2) \;=\; x + 1$$

Weakest Preexpectation Transformer

Consider the complete lattice $(\mathbb{E}, \leq)$ of *expectations* :

$$\mathbb{E} = \{f \mid f \colon \Sigma \to \mathbb{R}_{\geq 0}^{\infty}\} \qquad \text{with} \qquad f \leq g \quad \text{iff} \quad \forall \sigma \in \Sigma \colon f(\sigma) \leq g(\sigma)\,.$$

Weakest preexpectation transformer [Kozen '83, McIver '99, McIver & Morgan '05] :

$$\mathsf{wp}[\![C]\!] \colon \mathbb{E} \to \mathbb{E}\,, \qquad \mathsf{wp}[\![C]\!]\,(g)\,(\sigma) \;\widehat{=}\; \text{expected value of } g \text{ evaluated in final states reached after executing } C \text{ on } \sigma\,.$$

$$\mathsf{wp}[\![x := 5]\!]\,(x) \;=\; 5$$

$$\mathsf{wp}[\![\{\,\mathtt{skip}\,\}\;[\,1/2\,]\;\{\,x := x + 2\,\}]\!]\,(x) \;=\; \frac{1}{2}\cdot x + \frac{1}{2}\cdot (x+2) \;=\; x+1$$

$$\mathsf{wp}[\![\{\,\mathtt{skip}\,\}\;[\,1/2\,]\;\{\,x := x + 2\,\}]\!]\,([x = 4]) \;=\; \frac{1}{2}\cdot [x=4] + \frac{1}{2}\cdot [x=2]$$

Weakest Preexpectation Transformer

Consider the complete lattice $(\mathbb{E}, \leq)$ of *expectations* :

$$\mathbb{E} = \{f \mid f\colon \Sigma \to \mathbb{R}^{\infty}_{\geq 0}\} \qquad \text{with} \qquad f \leq g \quad \text{iff} \quad \forall \sigma \in \Sigma\colon f(\sigma) \leq g(\sigma)\,.$$

Weakest preexpectation transformer [Kozen '83, McIver '99, McIver & Morgan '05] :

$$\mathsf{wp}[\![C]\!]\colon \mathbb{E} \to \mathbb{E}\,, \qquad \mathsf{wp}[\![C]\!]\,(g)\,(\sigma) \;\widehat{=}\; \text{expected value of } g \text{ evaluated in final states reached after executing } C \text{ on } \sigma\,.$$

$$\mathsf{wp}[\![x := 5]\!]\,(x) \;=\; 5$$

$$\mathsf{wp}[\![\{\,\texttt{skip}\,\}\;[\,1/2\,]\;\{\,x := x+2\,\}]\!]\,(x) \;=\; \frac{1}{2}\cdot x + \frac{1}{2}\cdot(x+2) \;=\; x+1$$

$$\mathsf{wp}[\![\{\,\texttt{skip}\,\}\;[\,1/2\,]\;\{\,x := x+2\,\}]\!]\,([x=4]) \;=\; \frac{1}{2}\cdot[x=4] + \frac{1}{2}\cdot[x=2]$$

$$\mathsf{wp}[\![\texttt{while}\,(\,c=1\,)\,\{\,c := 0\;[1/2]\;x := x+1\,\}]\!]\,(x) \;=\; [c=1]\cdot(x+1) + [c\neq 1]\cdot x$$

k-Induction for Probabilistic Programs

Given : Loop $C = \texttt{while}\,(\,\varphi\,)\,\{\,C'\,\}$, postexpectation $g \in \mathbb{E}$ and candidate $f \in \mathbb{E}$.

Goal : Prove $\mathrm{wp}[\![C]\!]\,(g) \leq f$.

k-Induction for Probabilistic Programs

Given : Loop $C = \texttt{while}\,(\,\varphi\,)\,\{\,C'\,\}$, postexpectation $g \in \mathbb{E}$ and candidate $f \in \mathbb{E}$.
Goal : Prove $\mathrm{wp}[\![C]\!]\,(g) \leq f$.

We have

$$\mathrm{wp}[\![C]\!]\,(g) \;=\; \mathbf{lfp}\,\Phi \quad \text{with} \quad \Phi\colon \mathbb{E} \to \mathbb{E} \text{ monotonic}\,.$$

k-Induction for Probabilistic Programs

Given : Loop $C = \texttt{while}\,(\varphi)\,\{\,C'\,\}$, postexpectation $g \in \mathbb{E}$ and candidate $f \in \mathbb{E}$.
Goal : Prove $\mathrm{wp}[\![C]\!]\,(g) \leq f$.

We have

$$\mathrm{wp}[\![C]\!]\,(g) \;=\; \textbf{lfp}\;\Phi \quad \text{with} \quad \Phi\colon \mathbb{E} \to \mathbb{E} \text{ monotonic}\,.$$

Hence, latticed k-induction applies :

Corollary

For every $k \geq 1$,

$$\Phi\left(\Psi_f^{k-1}(f)\right) \leq f \quad \textit{implies} \quad \mathrm{wp}[\![C]\!]\,(g) \leq f\,.$$

k-Induction for Probabilistic Programs

Given : Loop $C = \texttt{while}\,(\,\varphi\,)\,\{\,C'\,\}$, postexpectation $g \in \mathbb{E}$ and candidate $f \in \mathbb{E}$.
Goal : Prove $\mathrm{wp}[\![C]\!]\,(g) \leq f$.

We have

$$\mathrm{wp}[\![C]\!]\,(g) \;=\; \textbf{lfp}\,\boldsymbol{\Phi} \quad \text{with} \quad \boldsymbol{\Phi}\colon \mathbb{E} \to \mathbb{E} \text{ monotonic} .$$

Hence, latticed k-induction applies :

Corollary

For every $k \geq 1$,

$$\Phi\left(\Psi_f^{k-1}(f)\right) \leq f \quad \textit{implies} \quad \mathrm{wp}[\![C]\!]\,(g) \leq f .$$

Here

$$\Psi_f(h) = \Phi\,(h) \sqcap f \qquad \text{where for } h, h' \in \mathbb{E}, \quad h \sqcap h' = \lambda\sigma\text{.}\ \min\{h\,(\sigma), h'\,(\sigma)\} .$$

Tool Support

kipro2 : ***k*-Induction for PRObabilistic PROgrams**

https://github.com/moves-rwth/kipro2

Tool Support

kipro2 : *k*-Induction for PRObabilistic PROgrams

https://github.com/moves-rwth/kipro2

functional · Artifact · CAV · AEC · Evaluated · available · reusable

For *linear* $C = \texttt{while}\,(\varphi)\,\{C'\}$ and *piecewise linear* f, g, kipro2 *semi-decides* by SMT :

Is there $k \geq 1$ s.t. $\mathrm{wp}[\![C]\!]\,(g) \leq f$ is k-inductive?

Tool Support

kipro2 : ***k*****-Induction for PRObabilistic PROgrams**

https://github.com/moves-rwth/kipro2

For *linear* $C = \texttt{while}\,(\varphi)\,\{C'\}$ and *piecewise linear* f, g, kipro2 *semi-decides* by SMT :

$$\text{Is there } k \geq 1 \text{ s.t. } \mathrm{wp}[\![C]\!]\,(g) \leq f \text{ is } k\text{-inductive?}$$

If $\mathrm{wp}[\![C]\!]\,(g) \not\leq f$, kipro2 finds via *bounded model checking* some $\sigma \in \Sigma$ with

$$\mathrm{wp}[\![C]\!]\,(g)\,(\sigma) > f(\sigma)\,.$$

Example — Geometric Distribution

For C given by

$$\texttt{while}\,(\,c = 1\,)\,\{\,c := 0\;[1/2]\;x := x + 1\,\}\,,$$

the property

$$\forall \text{ initial state } \sigma: \quad \mathrm{wp}[\![C]\!]\,(x)\,(\sigma) \;\leq\; \sigma(x) + 1$$

is 2-inductive.

Example — Geometric Distribution

For C given by

$$\texttt{while}\,(\,c = 1\,)\,\{\,c := 0\ [1/2]\ x := x + 1\,\}\,,$$

the property

$$\forall \text{ initial state } \sigma: \quad \mathsf{wp}[\![C]\!]\,(x)\,(\sigma) \;\leq\; \sigma(x) + 1$$

is 2-inductive. Does

$$\forall \text{ initial state } \sigma: \quad \mathsf{wp}[\![C]\!]\,(x)\,(\sigma) \;\leq\; \sigma(x) + 0.99$$

also hold ?

Example — Geometric Distribution

For C given by

$$\texttt{while}\,(\,c = 1\,)\,\{\,c := 0\;[1/2]\;x := x + 1\,\}\,,$$

the property

$$\forall\,\text{initial state}\,\sigma: \quad \mathsf{wp}[\![C]\!]\,(x)\,(\sigma)\;\leq\;\sigma(x) + 1$$

is 2-inductive. Does

$$\forall\,\text{initial state}\,\sigma: \quad \mathsf{wp}[\![C]\!]\,(x)\,(\sigma)\;\leq\;\sigma(x) + 0.99$$

also hold? No; counterexample by BMC : $\sigma(c) = 1, \sigma(x) = 6$.

Example — Uniform Sampling by Fair-Coin Flips [Lumbroso, 2013]

```
while(running = 0){

    v := 2*v;
    {c := 2*c+1}[0.5]{c := 2*c};
    if(not (v<n)){
        if((not (n=c)) & (not (n<c))){ #terminate
            running := 1
        }{
            v := v-n;
            c := c-n;
        }
    }{
        skip
    }

    #on termination, determine correct index
    if((not (running = 0))){
        c := elow + c;
    }{
        skip
    }
}
```

Example — Uniform Sampling by Fair-Coin Flips [Lumbroso, 2013]

```
while(running = 0){

    v := 2*v;
    {c := 2*c+1}[0.5]{c := 2*c};
    if(not (v<n)){
        if((not (n=c)) & (not (n<c))){ #terminate
            running := 1
        }{
            v := v-n;
            c := c-n;
        }
    }{
        skip
    }

    #on termination, determine correct index
    if((not (running = 0))){
        c := elow + c;
    }{
        skip
    }
}
```

For *arbitrary* array of fixed size $n = \{2, 3, 4, 5\}$, we verify

Pr(“sample fixed element”) $\leq 1/n$.

Empirical Results (Partial)

	postexpectation	variant	result	k	#formulae	formulae_t	sat_t	total_t
geo	x	1	ind	2	18	0.01	0.00	0.08
		2	ref	11	103	0.04	0.01	0.09
		3	ref	46	1223	0.39	0.04	0.48
unif_sam	$[c = i]$	1	ind	2	267	0.27	0.02	0.56
		2	ind	3	1402	1.45	0.10	1.81
		3	ind	3	1402	1.48	0.11	1.86
		4	ind	5	40568	47.31	15.70	63.28
		5	TO	–	–	–	–	–

$\vdots$

Empirical Results (Partial)

	postexpectation	variant	result	k	#formulae	formulae_t	sat_t	total_t
geo	x	1	ind	2	18	0.01	0.00	0.08
		2	ref	11	103	0.04	0.01	0.09
		3	ref	46	1223	0.39	0.04	0.48
unif_sam	$[c = i]$	1	ind	2	267	0.27	0.02	0.56
		2	ind	3	1402	1.45	0.10	1.81
		3	ind	3	1402	1.48	0.11	1.86
		4	ind	5	40568	47.31	15.70	63.28
		5	TO	–	–	–	–	–

⋮

Empirical Results (Partial)

	postexpectation	variant	result	k	#formulae	formulae_t	sat_t	total_t
geo	x	1	ind	2	18	0.01	0.00	0.08
		2	ref	11	103	0.04	0.01	0.09
		3	ref	46	1223	0.39	0.04	0.48
unif_sam	$[c = i]$	1	ind	2	267	0.27	0.02	0.56
		2	ind	3	1402	1.45	0.10	1.81
		3	ind	3	1402	1.48	0.11	1.86
		4	ind	5	40568	47.31	15.70	63.28
		5	TO	–	–	–	–	–

⋮

Summary

- k-Induction for transition systems in terms of fixed points;

⇒ K. Batz, M. Chen, B. L. Kaminski, J.-P. Katoen, C. Matheja, P. Schröer : *Latticed k-Induction with an Application to Probabilistic Programs*. CAV'21.

Summary

- k-Induction for transition systems in terms of fixed points;
- latticed k-induction for *verifying* $\text{lfp}\ \Phi \sqsubseteq f$;

⇒ K. Batz, M. Chen, B. L. Kaminski, J.-P. Katoen, C. Matheja, P. Schröer : *Latticed k-Induction with an Application to Probabilistic Programs*. CAV '21.

Summary

- k-Induction for transition systems in terms of fixed points;
- latticed k-induction for *verifying* $\text{lfp}\ \Phi \sqsubseteq f$;
- fully automatic verification of infinite-state probabilistic programs.

⇒ K. Batz, M. Chen, B. L. Kaminski, J.-P. Katoen, C. Matheja, P. Schröer : *Latticed k-Induction with an Application to Probabilistic Programs*. CAV '21.

Summary

- k-Induction for transition systems in terms of fixed points;
- latticed k-induction for *verifying* $\mathrm{lfp}\ \Phi \sqsubseteq f$;
- fully automatic verification of infinite-state probabilistic programs.

More in the paper :

- incremental SMT encoding (theory : QF_UFLIRA);

⇒ K. Batz, M. Chen, B. L. Kaminski, J.-P. Katoen, C. Matheja, P. Schröer : *Latticed k-Induction with an Application to Probabilistic Programs*. CAV '21.

Summary

- k-Induction for transition systems in terms of fixed points;
- latticed k-induction for *verifying* $\mathsf{lfp}\ \Phi \sqsubseteq f$;
- fully automatic verification of infinite-state probabilistic programs.

More in the paper:

- incremental SMT encoding (theory: QF_UFLIRA);
- k-induction for expected run-times;

⇒ K. Batz, M. Chen, B. L. Kaminski, J.-P. Katoen, C. Matheja, P. Schröer: *Latticed k-Induction with an Application to Probabilistic Programs*. CAV'21.

Summary

- k-Induction for transition systems in terms of fixed points ;
- latticed k-induction for *verifying* lfp $\Phi \sqsubseteq f$;
- fully automatic verification of infinite-state probabilistic programs.

More in the paper :

- incremental SMT encoding (theory : QF_UFLIRA) ;
- k-induction for expected run-times ;
- transfinite κ-induction ;

⇒ K. Batz, M. Chen, B. L. Kaminski, J.-P. Katoen, C. Matheja, P. Schröer : *Latticed k-Induction with an Application to Probabilistic Programs*. CAV '21.

Summary

- k-Induction for transition systems in terms of fixed points ;
- latticed k-induction for *verifying* lfp $\Phi \sqsubseteq f$;
- fully automatic verification of infinite-state probabilistic programs.

More in the paper :

- incremental SMT encoding (theory : QF_UFLIRA) ;
- k-induction for expected run-times ;
- transfinite κ-induction ;
- (in)completeness of latticed k-induction ;

⇒ K. Batz, M. Chen, B. L. Kaminski, J.-P. Katoen, C. Matheja, P. Schröer : *Latticed k-Induction with an Application to Probabilistic Programs*. CAV '21.

Summary

- k-Induction for transition systems in terms of fixed points;
- latticed k-induction for *verifying* $\mathrm{lfp}\ \Phi \sqsubseteq f$;
- fully automatic verification of infinite-state probabilistic programs.

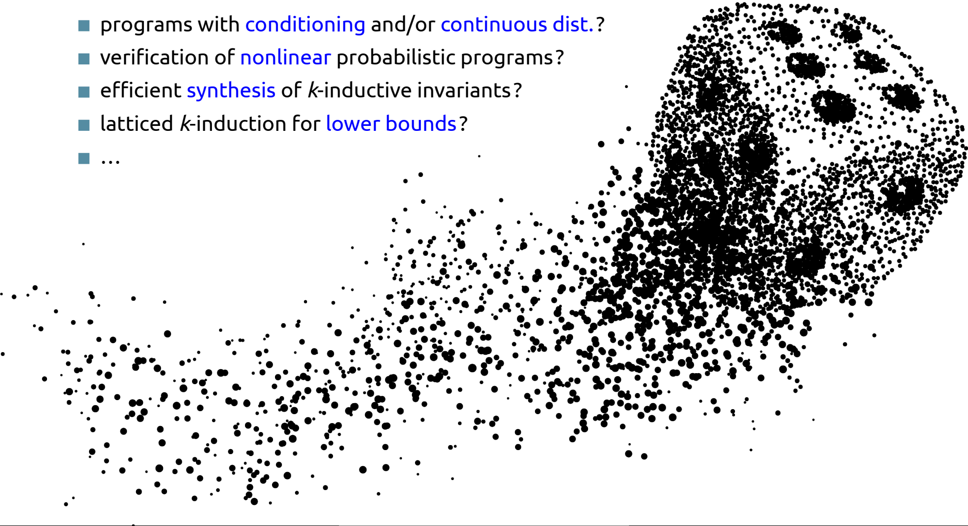
More in the paper :

- incremental SMT encoding (theory : QF_UFLIRA);
- k-induction for expected run-times;
- transfinite κ-induction;
- (in)completeness of latticed k-induction;
- latticed bounded model checking for *refuting* $\mathrm{lfp}\ \Phi \sqsubseteq f$.

⇛ K. Batz, M. Chen, B. L. Kaminski, J.-P. Katoen, C. Matheja, P. Schröer : *Latticed k-Induction with an Application to Probabilistic Programs*. CAV '21.

Future Directions

- programs with conditioning and/or continuous dist.?
- verification of nonlinear probabilistic programs?
- efficient synthesis of *k*-inductive invariants?
- latticed *k*-induction for lower bounds?
- ...



In Combination with Latticed BMC

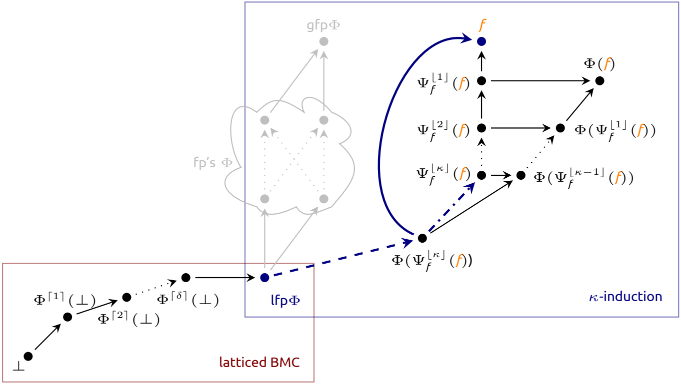


Figure – κ-induction and latticed BMC in case that lfp $\Phi \sqsubseteq f$.