Latticed k-Induction with an Application to Probabilistic Programs

Kevin Batz, Mingshuai Chen, Benjamin Lucien Kaminski,
Joost-Pieter Katoen, Christoph Matheja, Philipp Schröer

CAV 2021

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

ETH zürich

UCL
Programming Principles, Logic and Verification Group

k-Induction [Sheeran et al. 2000]

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

1

k-Induction [Sheeran et al. 2000]

- SAT-based technique for verifying (invariant properties of) finite transition systems
- Later: Verification of infinite-state transition systems by SMT solving

1

k-Induction [Sheeran et al. 2000]

- SAT-based technique for verifying (invariant properties of) finite transition systems
- Later: Verification of infinite-state transition systems by SMT solving
- Applications: Hardware- and software model checking

1

k-Induction [Sheeran et al. 2000]

- SAT-based technique for verifying (invariant properties of) finite transition systems
- Later: Verification of infinite-state transition systems by SMT solving
- Applications: Hardware- and software model checking

" [k-induction] easily integrates with existing SAT-solvers [...]. The simplicity of applying k-induction made it the go-to technique for SMT-based infinite-state model checking."[1]

[1] [Krishnan *et al.* 2018]

k-Induction [Sheeran et al. 2000]

- SAT-based technique for verifying (invariant properties of) finite transition systems
- Later: Verification of infinite-state transition systems by SMT solving
- Applications: Hardware- and software model checking

" [k-induction] easily integrates with existing SAT-solvers [...]. The simplicity of applying k-induction made it the go-to technique for SMT-based infinite-state model checking."[1]

Is k-induction applicable to
(possibly infinite-state) probabilistic program verification?

[1] [Krishnan *et al.* 2018]

k-Induction [Sheeran et al. 2000]

- SAT-based technique for verifying (invariant properties of) finite transition systems
- Later: Verification of infinite-state transition systems by SMT solving
- Applications: Hardware- and software model checking

" [k-induction] easily integrates with existing SAT-solvers [...]. The simplicity of applying k-induction made it the go-to technique for SMT-based infinite-state model checking."[1]

Is k-induction applicable to
(possibly infinite-state) probabilistic program verification?

Yes. Enables *fully automatic* verification of non-trivial properties.

[1] [Krishnan *et al.* 2018]

For C given by

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

.

For C given by

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

the property

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

.

For C given by

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

the property

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

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

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

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

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

$$\underbrace{P(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge P(s_k)}_{\text{assuming we stay in } P \text{ for } k-1 \text{ steps,}} \wedge \underbrace{T(s_k, s_{k+1})}_{\text{after step } k,} \implies \underbrace{P(s_{k+1})}_{\text{we end up in } P \text{ again}},$$

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

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

$$\underbrace{P(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge P(s_k)}_{\text{assuming we stay in } P \text{ for } k-1 \text{ steps,}} \wedge \underbrace{T(s_k, s_{k+1})}_{\text{after step } k,} \implies \underbrace{P(s_{k+1})}_{\text{we end up in } P \text{ again}},$$

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

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

$$\underbrace{P(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge P(s_k)}_{\text{assuming we stay in } P \text{ for } k-1 \text{ steps,}} \wedge \underbrace{T(s_k, s_{k+1})}_{\text{after step } k,} \implies \underbrace{P(s_{k+1})}_{\text{we end up in } P \text{ again}},$$

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

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

$$\underbrace{P(s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge P(s_k)}_{\text{assuming we stay in } P \text{ for } k-1 \text{ steps,}} \wedge \underbrace{T(s_k, s_{k+1})}_{\text{after step } k,} \implies \underbrace{P(s_{k+1})}_{\text{we end up in } P \text{ again}},$$

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

For verifying probabilistic programs, we have to ...

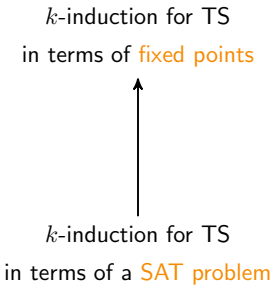
- ... leave the Boolean domain and reason about quantities
- ... reason about sets of paths rather than individual paths

Latticed k-Induction

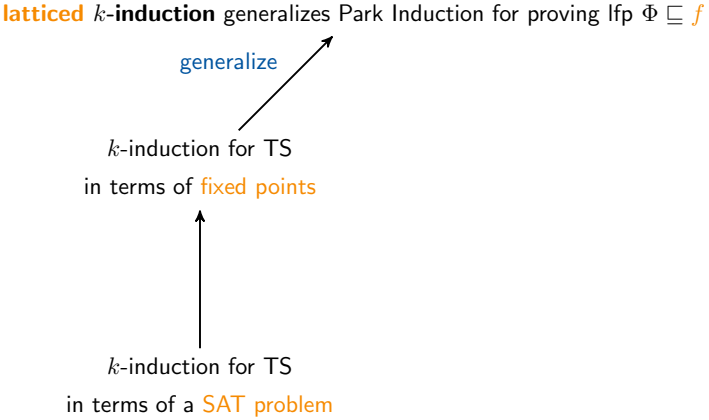
k-induction for TS

in terms of a SAT problem

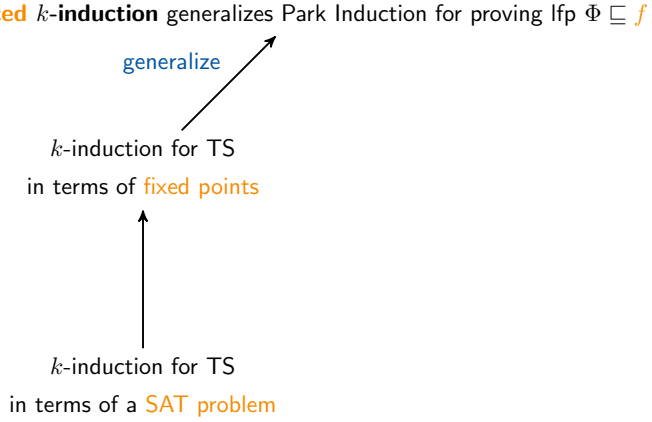
Latticed k-Induction



Latticed k-Induction

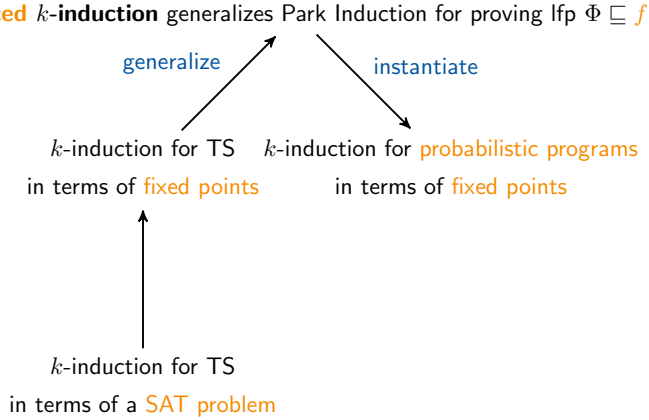


Latticed k-Induction



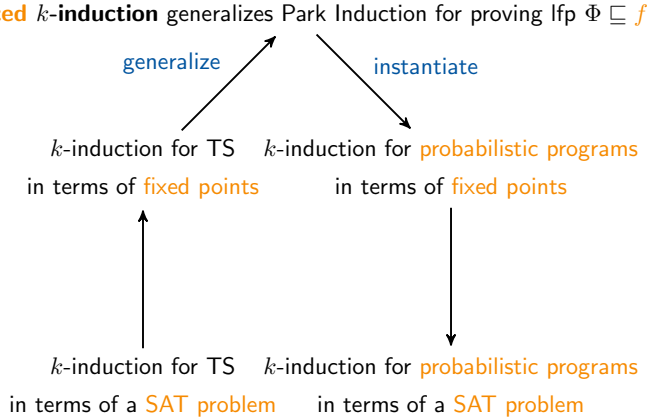
Whenever your problem boils down to verifying an upper bound on a least fixed point, latticed k-induction provides you with inductive proof rules!

Latticed k-Induction



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

Latticed k-Induction



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

Latticed k-Induction

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

Latticed k-Induction

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

Goal: Prove lfp $\Phi \sqsubseteq f$

Latticed k-Induction

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

Goal: Prove lfp $\Phi \sqsubseteq f$

Park Induction aka 1-induction:

$$\Phi(f) \sqsubseteq f \qquad \text{implies} \qquad \text{lfp } \Phi \sqsubseteq f$$

Latticed k-Induction

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

Goal: Prove lfp $\Phi \sqsubseteq f$

Park Induction aka 1-induction:

$$\Phi(f) \sqsubseteq f \qquad \text{implies} \qquad \text{lfp } \Phi \sqsubseteq f$$

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

Latticed k-Induction

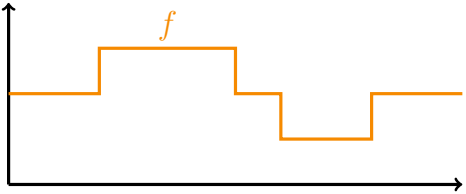
Given: Complete lattice $(E, \sqsubseteq)$, monotonic $\Phi\colon E \to E$, and $f \in E$

Goal: Prove lfp $\Phi \sqsubseteq f$

Park Induction aka 1-induction:

$$\Phi(f) \sqsubseteq f \qquad \text{implies} \qquad \text{lfp } \Phi \sqsubseteq f$$

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



Latticed k-Induction

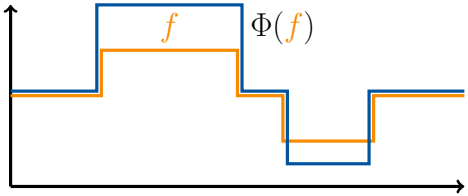
Given: Complete lattice $(E, \sqsubseteq)$, monotonic $\Phi\colon E \to E$, and $f \in E$

Goal: Prove lfp $\Phi \sqsubseteq f$

Park Induction aka 1-induction:

$$\Phi(f) \sqsubseteq f \qquad \text{implies} \qquad \text{lfp } \Phi \sqsubseteq f$$

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



Latticed k-Induction

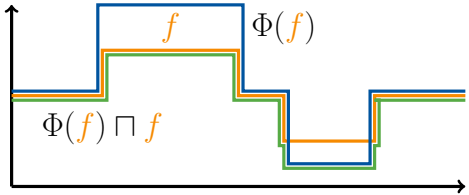
Given: Complete lattice $(E, \sqsubseteq)$, monotonic $\Phi\colon E \to E$, and $f \in E$

Goal: Prove lfp $\Phi \sqsubseteq f$

Park Induction aka 1-induction:

$$\Phi(f) \sqsubseteq f \qquad \text{implies} \qquad \text{lfp } \Phi \sqsubseteq f$$

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



Latticed k-Induction

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

Goal: Prove lfp $\Phi \sqsubseteq f$

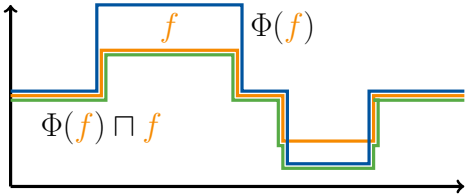
Park Induction aka 1-induction:

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

Even though 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 $\Phi\colon E \to E$, and $f \in E$

Goal: Prove lfp $\Phi \sqsubseteq f$

Park Induction aka 1-induction

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

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

2-induction:

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

3-induction:

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

Latticed k-Induction

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

Goal: Prove lfp $\Phi \sqsubseteq f$

Define $\Psi_f \colon E \to E$ by

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

Latticed k-Induction

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

Goal: Prove lfp $\Phi \sqsubseteq f$

Define $\Psi_f\colon E \to E$ by

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

Theorem (Latticed k-Induction)

For every $k \geq 1$,

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

We call such f k-inductive invariant.

Latticed k-Induction

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

Goal: Prove lfp $\Phi \sqsubseteq f$

Define $\Psi_f\colon E \to E$ by

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

Theorem (Latticed k-Induction)

For every $k \geq 1$,

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

We call such f k-inductive invariant.

k-Induction generalizes Park induction $\triangleq$ 1-induction.

Latticed k-Induction

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

Goal: Prove lfp $\Phi \sqsubseteq f$

Define $\Psi_f\colon E \to E$ by

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

Theorem (Latticed k-Induction)

For every $k \geq 1$,

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

We call such f k-inductive invariant.

k-Induction generalizes Park induction $\triangleq$ 1-induction.

Can be generalized to transfinite κ-induction (not in this talk).

Latticed k-Induction

Theorem (Park Induction from k-Induction)

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

Latticed k-Induction

Theorem (Park Induction from k-Induction)

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

Lemma

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

$$f \sqsupseteq \Psi_f(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \dots .$$

Latticed k-Induction

Theorem (Park Induction from k-Induction)

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

Lemma

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

$$f \sqsupseteq \Psi_f(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \dots .$$

Hence, if f is k-inductive invariant, then

Latticed k-Induction

Theorem (Park Induction from k-Induction)

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

Lemma

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

$$f \sqsupseteq \Psi_f(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \dots .$$

Hence, if f is k-inductive invariant, then

- $\Psi_f^{k-1}(f)$ is an inductive invariant,

Latticed k-Induction

Theorem (Park Induction from k-Induction)

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

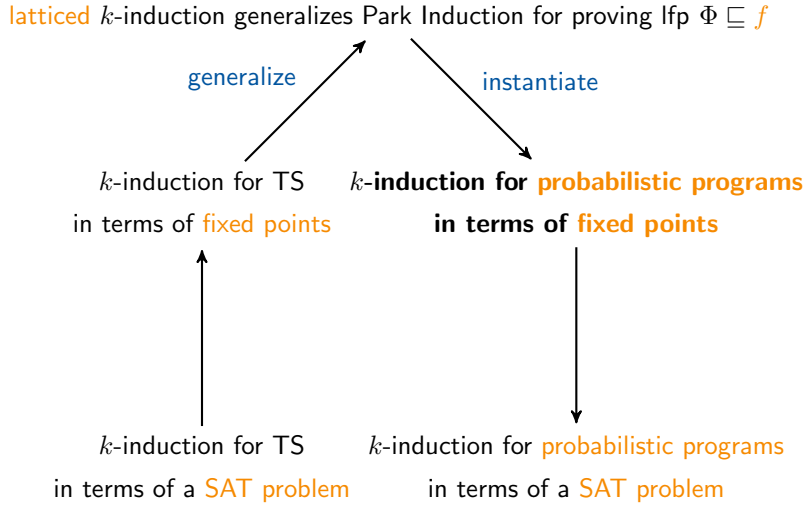
Lemma

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

$$f \sqsupseteq \Psi_f(f) \sqsupseteq \Psi_f^2(f) \sqsupseteq \Psi_f^3(f) \sqsupseteq \dots .$$

Hence, if f is k-inductive invariant, then

- $\Psi_f^{k-1}(f)$ is an inductive invariant,
- which is stronger than f.


latticed k-induction generalizes Park Induction for proving lfp $\Phi \sqsubseteq f$
generalize
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

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

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

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

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

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

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

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

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

$$\mathsf{wp}[\![C]\!] \colon \mathbb{E} \to \mathbb{E}$$

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

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

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

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

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

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

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

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

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

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]:

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

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

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

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]:

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

$$\begin{aligned}
\mathsf{wp}[\![x \coloneqq 5]\!]\,(x) &= 5 \\
\mathsf{wp}[\![\{\,\texttt{skip}\,\}\ [1/2]\ \{\,x \coloneqq x+2\,\}]\!]\,(x) &= \frac{1}{2} \cdot x + \frac{1}{2} \cdot (x+2) = x+1 \\
\mathsf{wp}[\![\{\,\texttt{skip}\,\}\ [1/2]\ \{\,x \coloneqq x+2\,\}]\!]\,([x=4]) &= \frac{1}{2} \cdot [x=4] + \frac{1}{2} \cdot [x=2]
\end{aligned}$$

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]:

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

$$\begin{aligned}
\mathsf{wp}[\![x \coloneqq 5]\!]\,(x) &= 5 \\
\mathsf{wp}[\![\{\,\texttt{skip}\,\}\ [1/2]\ \{\,x \coloneqq x+2\,\}]\!]\,(x) &= \frac{1}{2}\cdot x + \frac{1}{2}\cdot(x+2) = x+1 \\
\mathsf{wp}[\![\{\,\texttt{skip}\,\}\ [1/2]\ \{\,x \coloneqq x+2\,\}]\!]\,([x=4]) &= \frac{1}{2}\cdot[x=4] + \frac{1}{2}\cdot[x=2] \\
\mathsf{wp}[\![\texttt{while}\,(c=1)\,\{\,c \coloneqq 0\ [1/2]\ x \coloneqq x+1\,\}]\!]\,(x) &= [c=1]\cdot(x+1) + [c\neq 1]\cdot x
\end{aligned}$$

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

k-Induction for Probabilistic Programs

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

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

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

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

We have

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

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

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

We have

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

Hence, latticed k-induction applies:

Corollary

For every $k \geq 1$,

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

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

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

We have

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

Hence, latticed k-induction applies:

Corollary

For every $k \geq 1$,

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

Here

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

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

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

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

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

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

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

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

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

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

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

$$\text{Is there } k \geq 1 \text{ such that } \mathsf{wp}[\![C]\!]\,(g) \leq f \text{ is } k\text{-inductive?}$$

If $\mathsf{wp}[\![C]\!]\,(g) \not\leq f$, KIPRO2 finds via *bounded model checking* some $\sigma \in \Sigma$ with

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

For C given by

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

the property

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

is 2-inductive.

For C given by

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

the property

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

is 2-inductive. Does

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

also hold?

For C given by

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

the property

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

is 2-inductive. Does

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

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

Sampling uniformly from $\{elow, \ldots, ehigh\}$ using fair coin flips only [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
  }
}
```

Sampling uniformly from $\{elow, \ldots, ehigh\}$ using fair coin flips only [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

$$\Pr(\text{“sample fixed element”}) \leq \frac{1}{n} .$$

Table 2: Empirical results for the first benchmark set (time in seconds).

	postexpectation	variant	result	k	#formulae	formulae_t	sat_t	total_t
geo	c	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_gen	$[c = i]$	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	–	–	–	–	–

...

Conclusion

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

Conclusion

- k-Induction for transition systems in terms of fixed points
- latticed k-induction

Conclusion

- k-Induction for transition systems in terms of fixed points
- latticed k-induction
- fully automatic k-induction for probabilistic programs

Conclusion

- k-Induction for transition systems in terms of fixed points
- latticed k-induction
- fully automatic k-induction for probabilistic programs

Further topics:

- incremental SMT encoding (theory: QF_UFLIRA)

Conclusion

- k-Induction for transition systems in terms of fixed points
- latticed k-induction
- fully automatic k-induction for probabilistic programs

Further topics:

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

Conclusion

- k-Induction for transition systems in terms of fixed points
- latticed k-induction
- fully automatic k-induction for probabilistic programs

Further topics:

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

Conclusion

- k-Induction for transition systems in terms of fixed points
- latticed k-induction
- fully automatic k-induction for probabilistic programs

Further topics:

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

Conclusion

- k-Induction for transition systems in terms of fixed points
- latticed k-induction
- fully automatic k-induction for probabilistic programs

Further topics:

- incremental SMT encoding (theory: QF_UFLIRA)
- k-induction for expected run-times
- transfinite κ-induction
- (in)completeness of k-induction
- latticed bounded model checking (refute lfp $\Phi \sqsubseteq f$)

Conclusion

- k-Induction for transition systems in terms of fixed points
- latticed k-induction
- fully automatic k-induction for probabilistic programs

Further topics:

- incremental SMT encoding (theory: QF_UFLIRA)
- k-induction for expected run-times
- transfinite κ-induction
- (in)completeness of k-induction
- latticed bounded model checking (refute lfp $\Phi \sqsubseteq f$)

Thank you!