IA169 Model Checking

Bounded model checking and *k*-induction

Jan Strejček

Faculty of Informatics
Masaryk University

Motivation

- BDDs represents all models of the corresponding propositional formulas
- in LTL model checking, we want to decide whether some violating run exists
- if we represent violating runs by a formula, we need to decide its satisfiability
- SAT solvers can efficiently decide it (despite NP-completeness of the problem)

Motivation

- BDDs represents all models of the corresponding propositional formulas
- in LTL model checking, we want to decide whether some violating run exists
- if we represent violating runs by a formula, we need to decide its satisfiability
- SAT solvers can efficiently decide it (despite NP-completeness of the problem)

- for satisfiable formulas, SAT solvers provide a model
- a formula φ is true iff $\neg\varphi$ is not satisfiable

Agenda and sources

agenda

- finite Kripke structures represented by formulas
- bounded model checking (BMC) for safety properties
- BMC for LTL properties
- completeness of BMC
- *k*-induction

source

- Chapter 10 of E. M. Clarke, O. Grumberg, D. Kroening, D. Peled, and H. Veith: *Model Checking*, Second Edition, MIT, 2018.

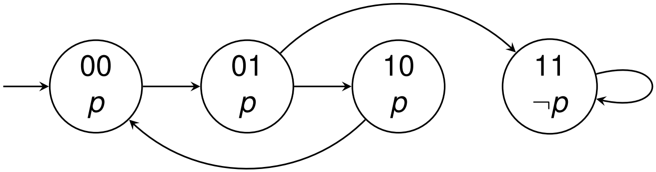
Finite Kripke structures represented by formulas

Finite Kripke structures represented by formulas

- each Kripke structure $K = (S, T, S_0, L)$ with finitely many states and a finite set of used atomic propositions can be encoded by propositional formulas
- states in S correspond to assignments $s : V \rightarrow \{0, 1\}$, where $V = \{x_1, \ldots, x_n\}$
- S_0 is identified with a formula $S_0(x_1, \ldots, x_n)$ satisfied by initial states
- transition relation $T \subseteq S \times S$ is identified with a formula $T(x_1, \ldots, x_n, x'_1, \ldots, x'_n)$
- we replace $L : S \rightarrow 2^{AP}$ with a formula $p(x_1, \ldots, x_n)$ for each relevant $p \in AP$

Finite Kripke structures represented by formulas

- each Kripke structure $K = (S, T, S_0, L)$ with finitely many states and a finite set of used atomic propositions can be encoded by propositional formulas
- states in S correspond to assignments $s : V \to \{0, 1\}$, where $V = \{x_1, \ldots, x_n\}$
- S_0 is identified with a formula $S_0(x_1, \ldots, x_n)$ satisfied by initial states
- transition relation $T \subseteq S \times S$ is identified with a formula $T(x_1, \ldots, x_n, x_1', \ldots, x_n')$
- we replace $L : S \to 2^{AP}$ with a formula $p(x_1, \ldots, x_n)$ for each relevant $p \in AP$



$$
\begin{aligned}
S_0(x_1, x_2) &= \neg x_1 \wedge \neg x_2 \\
T(x_1, x_2, x_1', x_2') &= (\neg x_1 \wedge \neg x_2 \wedge \neg x_1' \wedge x_2') \vee (\neg x_1 \wedge x_2 \wedge x_1') \vee \\
&\quad \vee (x_1 \wedge \neg x_2 \wedge \neg x_1' \wedge \neg x_2') \vee (x_1 \wedge x_2 \wedge \wedge x_1' \wedge x_2') \\
p(x_1, x_2) &= \neg x_1 \vee \neg x_2
\end{aligned}
$$

Finite Kripke structures represented by formulas

- we write $\vec{x}$ instead of $x_1, \ldots, x_n$, i.e., we use $S_0(\vec{x})$, $T(\vec{x}, \vec{x}')$ and $p(\vec{x})$
- when building formulas about more than one or two states, we will use $\vec{x}_0, \vec{x}_1, \ldots$, where $\vec{x}_i$ stands for $x_{i1}, \ldots, x_{i\,n}$
- for example, models of $T(\vec{x}_0, \vec{x}_1) \wedge T(\vec{x}_1, \vec{x}_2)$ represent paths of length 2
- recall that we assume that each state has at least one successor

Bounded model checking (BMC) for safety properties

Basic idea of bounded model checking (BMC)

- if a finite system violates a given property, it often has a short counterexample
- bounded model checking (BMC) analyzes runs up to the first k steps
- if an erroneous run is found, we know that the system violates the property; otherwise, we can increase k and try again

Basic idea of bounded model checking (BMC)

- if a finite system violates a given property, it often has a short counterexample
- bounded model checking (BMC) analyzes runs up to the first k steps
- if an erroneous run is found, we know that the system violates the property; otherwise, we can increase k and try again

- let us consider the safety property $\mathrm{G}p$
- the property is violated iff some run satisfies $\mathrm{F}\neg p$
- there is a run violating the property within the first k steps iff the following formula is satisfiable

$$S_0(\vec{x}_0) \;\wedge\; \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \;\wedge\; \bigvee_{i=0}^{k} \neg p(\vec{x}_i)$$

Basic idea of bounded model checking (BMC)

- if a finite system violates a given property, it often has a short counterexample
- bounded model checking (BMC) analyzes runs up to the first k steps
- if an erroneous run is found, we know that the system violates the property; otherwise, we can increase k and try again

- let us consider the safety property Gp
- the property is violated iff some run satisfies F$\neg p$
- there is a run violating the property within the first k steps iff the following formula is satisfiable

$$S_0(\vec{x}_0) \;\wedge\; \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \;\wedge\; \bigvee_{i=0}^{k} \neg p(\vec{x}_i)$$

- for example, for $k = 3$ the formula is

$$S_0(\vec{x}_0) \wedge T(\vec{x}_0, \vec{x}_1) \wedge T(\vec{x}_1, \vec{x}_2) \wedge T(\vec{x}_2, \vec{x}_3) \wedge \Big(\neg p(\vec{x}_0) \vee \neg p(\vec{x}_1) \vee \neg p(\vec{x}_2) \vee \neg p(\vec{x}_3)\Big)$$

BMC for safety properties

bounded model checker for safety properties

1. set k to some initial (relatively low) number
2. construct the formula

$$\psi_k \;=\; S_0(\vec{x}_0) \;\wedge\; \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \;\wedge\; \bigvee_{i=0}^{k} \neg p(\vec{x}_i)$$

3. ask a SAT solver for satisfiability of ψ_k
4. if ψ_k is satisfiable, then report $K \not\models \mathsf{G}p$ and construct a counterexample from the obtained model
5. if ψ_k is unsatisfiable, increase k and go to 2

BMC for safety properties

bounded model checker for safety properties

1. set k to some initial (relatively low) number
2. construct the formula

$$\psi_k \;=\; S_0(\vec{x}_0) \;\wedge\; \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \;\wedge\; \bigvee_{i=0}^{k} \neg p(\vec{x}_i)$$

3. ask a SAT solver for satisfiability of ψ_k
4. if ψ_k is satisfiable, then report $K \not\models \mathrm{G}p$ and construct a counterexample from the obtained model
5. if ψ_k is unsatisfiable, increase k and go to 2

- the size of ψ_k is linear in k
- the method is not complete: it never ends for correct systems

BMC for LTL properties

BMC for LTL properties

- we want to check whether a (fair) Kripke structure K satisfies an LTL formula φ
- assume that we have a generalized Büchi automaton B representing a product of K and an automaton for $\neg\varphi$
- $K \models_{(F)} \varphi$ iff $L(B) = \emptyset$
- $L(B) \neq \emptyset$ iff there exists an accepting lasso-shaped run of B of the form $\tau.\rho^\omega$
- bounded model checking looks for accepting runs $\tau.\rho^\omega$ such that $|\tau\rho| \leq k$
- if such a run exists, then $L(B) \neq \emptyset$ and thus $K \not\models_{(F)} \varphi$

BMC for LTL properties

assume that the GBA B is described by propositional formulas

- $S_0(\vec{x})$ is satisfied by initial states
- $T(\vec{x}, \vec{x}')$ represents the transiton relation (the letters on transitions are ignored as they have no influence on the existence of accepting runs)
- for each $F_l \in \mathcal{F}$, $F_l(\vec{x})$ represents the elements of accepting set F_l

BMC for LTL properties

assume that the GBA B is described by propositional formulas

- $S_0(\vec{x})$ is satisfied by initial states
- $T(\vec{x}, \vec{x}')$ represents the transiton relation (the letters on transitions are ignored as they have no influence on the existence of accepting runs)
- for each $F_l \in \mathcal{F}$, $F_l(\vec{x})$ represents the elements of accepting set F_l

- there exists an accepting run $\tau.\rho^\omega$ such that $|\tau\rho| = k$ iff the following formula is satisfiable

$$S_0(\vec{x}_0) \;\wedge\; \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \;\wedge\; \bigvee_{i=0}^{k-1} \Big(\vec{x}_i = \vec{x}_k \wedge \bigwedge_{F_l \in \mathcal{F}} \bigvee_{j=i}^{k-1} F_l(\vec{x}_j)\Big)$$

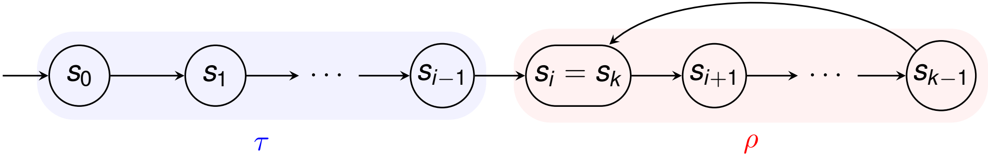
BMC for LTL properties

assume that the GBA B is described by propositional formulas

- $S_0(\vec{x})$ is satisfied by initial states
- $T(\vec{x}, \vec{x}')$ represents the transiton relation (the letters on transitions are ignored as they have no influence on the existence of accepting runs)
- for each $F_l \in \mathcal{F}$, $F_l(\vec{x})$ represents the elements of accepting set F_l

- there exists an accepting run $\tau.\rho^\omega$ such that $|\tau\rho| = k$ iff the following formula is satisfiable

$$S_0(\vec{x}_0) \wedge \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \wedge \bigvee_{i=0}^{k-1} \Big(\vec{x}_i = \vec{x}_k \wedge \bigwedge_{F_l \in \mathcal{F}} \bigvee_{j=i}^{k-1} F_l(\vec{x}_j) \Big)$$



BMC for LTL properties

- assume that there exists an accepting run $\tau.\rho^\omega$ such that $|\tau\rho| < k$
- then $\tau.\rho^\omega = \tau'.\rho'^\omega$ where $\tau'\rho'$ is the prefix of $\tau.\rho^\omega$ such that $|\tau'\rho'| = k$ and $|\rho'| = |\rho|$
- hence, there exists an accepting run $\tau.\rho^\omega$ such that $|\tau\rho| \leq k$ iff ψ_k is satisfiable

$$\psi_k = S_0(\vec{x}_0) \wedge \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \wedge \bigvee_{i=0}^{k-1} \Big(\vec{x}_i = \vec{x}_k \wedge \bigwedge_{F_l \in \mathcal{F}} \bigvee_{j=i}^{k-1} F_l(\vec{x}_j) \Big)$$

BMC for LTL properties

- assume that there exists an accepting run $\tau.\rho^\omega$ such that $|\tau\rho| < k$
- then $\tau.\rho^\omega = \tau'.\rho'^\omega$ where $\tau'\rho'$ is the prefix of $\tau.\rho^\omega$ such that $|\tau'\rho'| = k$ and $|\rho'| = |\rho|$
- hence, there exists an accepting run $\tau.\rho^\omega$ such that $|\tau\rho| \leq k$ iff ψ_k is satisfiable

$$\psi_k = S_0(\vec{x}_0) \wedge \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \wedge \bigvee_{i=0}^{k-1} \Big(\vec{x}_i = \vec{x}_k \wedge \bigwedge_{F_l \in \mathcal{F}} \bigvee_{j=i}^{k-1} F_l(\vec{x}_j)\Big)$$

bounded model checker for LTL properties

1. set k to some initial (relatively low) number
2. construct the formula ψ_k and ask a SAT solver for its satisfiability
3. if ψ_k is satisfiable, then report $K \not\models_{(F)} \varphi$ and construct a counterexample from the obtained model
4. if ψ_k is unsatisfiable, increase k and go to 2

BMC for LTL properties

- assume that there exists an accepting run $\tau.\rho^\omega$ such that $|\tau\rho| < k$
- then $\tau.\rho^\omega = \tau'.\rho'^\omega$ where $\tau'\rho'$ is the prefix of $\tau.\rho^\omega$ such that $|\tau'\rho'| = k$ and $|\rho'| = |\rho|$
- hence, there exists an accepting run $\tau.\rho^\omega$ such that $|\tau\rho| \leq k$ iff ψ_k is satisfiable

$$\psi_k = S_0(\vec{x}_0) \wedge \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \wedge \bigvee_{i=0}^{k-1} \Big(\vec{x}_i = \vec{x}_k \wedge \bigwedge_{F_l \in \mathcal{F}} \bigvee_{j=i}^{k-1} F_l(\vec{x}_j)\Big)$$

bounded model checker for LTL properties

1. set k to some initial (relatively low) number
2. construct the formula ψ_k and ask a SAT solver for its satisfiability
3. if ψ_k is satisfiable, then report $K \not\models_{(F)} \varphi$ and construct a counterexample from the obtained model
4. if ψ_k is unsatisfiable, increase k and go to 2

- the size of ψ_k (when counting all common subformulas only once) is linear in k
- the method is not complete: it never ends for correct systems

Completeness of BMC

Completeness of BMC

- is there any k such that if BMC does not find any erroneous path using k then the system has to be safe?
- we will study this question for safety property $\mathsf{G}p$

Completeness of BMC

- is there any k such that if BMC does not find any erroneous path using k then the system has to be safe?
- we will study this question for safety property Gp

the number of states

- a state satisfying $\neg p$ is reachable from initial states iff it is reachable in $|S| - 1$ steps
- if the formula ψ_k for $k = |S| - 1$ is not satisfiable, then $K \models \mathrm{G}p$
- if states are modeled by Boolean variables $x_1, \ldots, x_n$ then $|S| \leq 2^n$
- this bound is too large to be practical

Completeness of BMC

diametr of the system graph

- graph diametr d is the maximal length of all shortest paths between any two graph nodes
- a state satisfying $\neg p$ is reachable from initial states iff it is reachable in d steps
- if the formula ψ_k for $k = d$ is not satisfiable, then $K \models \mathrm{G}p$

Completeness of BMC

diametr of the system graph

- graph diametr d is the maximal length of all shortest paths between any two graph nodes
- a state satisfying $\neg p$ is reachable from initial states iff it is reachable in d steps
- if the formula ψ_k for $k = d$ is not satisfiable, then $K \models \mathrm{G}p$
- how to determine d without constructing the graph?
- asking the user is not realistic
- safe upper bounds (like $d \leq |S| - 1$) are extremely overstated

k-induction

Proof of correctness by induction

- another way to prove that $K \models \mathrm{G}p$ with SAT solvers
- we need to prove that p holds in all states reachable from the initial states

induction

1. base case: all initial states satisfy p, i.e., $S_0(\vec{x}) \wedge \neg p(\vec{x})$ is unsatisfiable
2. induction step: if a state satisfies p, then each its successor satisfies p, i.e., the following formula is unsatisfiable

$$p(\vec{x}) \wedge T(\vec{x}, \vec{x}') \wedge \neg p(\vec{x}')$$

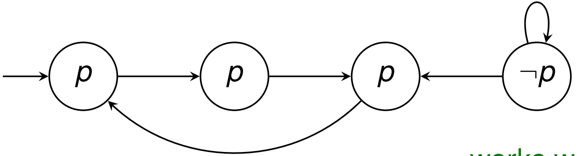
Proof of correctness by induction

- another way to prove that $K \models \mathbf{G}p$ with SAT solvers
- we need to prove that p holds in all states reachable from the initial states

induction

1. base case: all initial states satisfy p, i.e., $S_0(\vec{x}) \wedge \neg p(\vec{x})$ is unsatisfiable
2. induction step: if a state satisfies p, then each its successor satisfies p, i.e., the following formula is unsatisfiable

$$p(\vec{x}) \wedge T(\vec{x}, \vec{x}') \wedge \neg p(\vec{x}')$$



works well

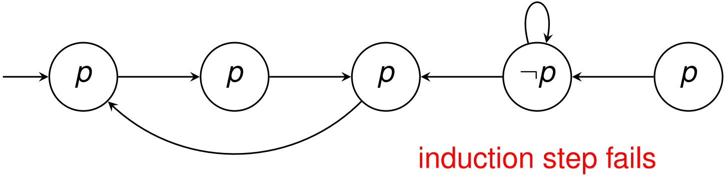
Proof of correctness by induction

- another way to prove that $K \models \mathrm{G}p$ with SAT solvers
- we need to prove that p holds in all states reachable from the initial states

induction

1. base case: all initial states satisfy p, i.e., $S_0(\vec{x}) \wedge \neg p(\vec{x})$ is unsatisfiable
2. induction step: if a state satisfies p, then each its successor satisfies p, i.e., the following formula is unsatisfiable

$$p(\vec{x}) \wedge T(\vec{x}, \vec{x}') \wedge \neg p(\vec{x}')$$



k-induction

k-induction

1. base case: each path of length k starting in an initial state does not reach any state satisfying $\neg p$, i.e., the following formula is unsatisfiable
$$S_0(\vec{x}_0) \;\wedge\; \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \;\wedge\; \bigvee_{i=0}^{k} \neg p(\vec{x}_i)$$
2. induction step: if we prolong any path of length k over states satisfying p by one step, we reach a state satisfying p, i.e., the following formula is unsatisfiable
$$\bigwedge_{i=0}^{k} \Big(p(\vec{x}_i) \wedge T(\vec{x}_i, \vec{x}_{i+1})\Big) \;\wedge\; \neg p(\vec{x}_{k+1})$$

- the base case uses the formula from BMC: if it is satisfiable then $K \not\models \mathrm{G}p$

k-induction

k-induction

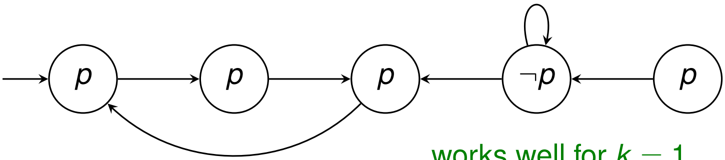
1. base case: each path of length k starting in an initial state does not reach any state satisfying $\neg p$, i.e., the following formula is unsatisfiable

$$S_0(\vec{x}_0) \;\wedge\; \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \;\wedge\; \bigvee_{i=0}^{k} \neg p(\vec{x}_i)$$

2. induction step: if we prolong any path of length k over states satisfying p by one step, we reach a state satisfying p, i.e., the following formula is unsatisfiable

$$\bigwedge_{i=0}^{k} \Big(p(\vec{x}_i) \wedge T(\vec{x}_i, \vec{x}_{i+1})\Big) \;\wedge\; \neg p(\vec{x}_{k+1})$$

- the base case uses the formula from BMC: if it is satisfiable then $K \not\models \mathrm{G}p$



works well for $k = 1$

k-induction

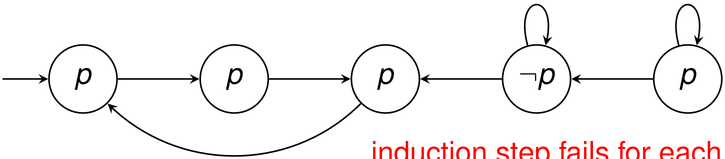
1. base case: each path of length k starting in an initial state does not reach any state satisfying $\neg p$, i.e., the following formula is unsatisfiable

$$S_0(\vec{x}_0) \wedge \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \wedge \bigvee_{i=0}^{k} \neg p(\vec{x}_i)$$

2. induction step: if we prolong any path of length k over states satisfying p by one step, we reach a state satisfying p, i.e., the following formula is unsatisfiable

$$\bigwedge_{i=0}^{k} \Big(p(\vec{x}_i) \wedge T(\vec{x}_i, \vec{x}_{i+1})\Big) \wedge \neg p(\vec{x}_{k+1})$$

- the base case uses the formula from BMC: if it is satisfiable then $K \not\models \mathrm{G}p$



induction step fails for each k

k-induction

- a state satisfying $\neg p$ is reachable iff it is reachable by an acyclic path
- hence, the induction step can consider only acyclic paths

2. induction step: if we prolong any path of length k over states satisfying p by one step such that we get an acyclic path, we reach a state satisfying p, i.e., the following formula is unsatisfiable

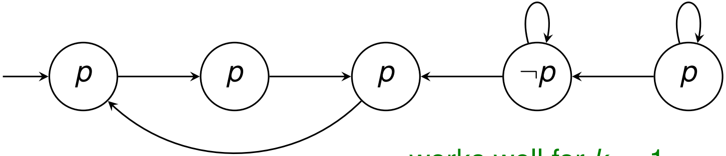
$$\bigwedge_{i=0}^{k} \Big(p(\vec{x}_i) \wedge T(\vec{x}_i, \vec{x}_{i+1}) \Big) \ \wedge \bigwedge_{0 \le i < j \le k+1} \vec{x}_i \neq \vec{x}_j \ \wedge \ \neg p(\vec{x}_{k+1})$$

k-induction

- a state satisfying $\neg p$ is reachable iff it is reachable by an acyclic path
- hence, the induction step can consider only acyclic paths

2. induction step: if we prolong any path of length k over states satisfying p by one step such that we get an acyclic path, we reach a state satisfying p, i.e., the following formula is unsatisfiable

$$\bigwedge_{i=0}^{k} \Big(p(\vec{x}_i) \wedge T(\vec{x}_i, \vec{x}_{i+1}) \Big) \ \wedge \bigwedge_{0 \le i < j \le k+1} \vec{x}_i \neq \vec{x}_j \ \wedge \ \neg p(\vec{x}_{k+1})$$



works well for $k = 1$

k-induction algorithm

k-induction algorithm for safety properties

1. set k to some initial (relatively low) number
2. construct the formulas

$$\psi_k = S_0(\vec{x}_0) \wedge \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \wedge \bigvee_{i=0}^{k} \neg p(\vec{x}_i)$$

$$\eta_k = \bigwedge_{i=0}^{k} \Big(p(\vec{x}_i) \wedge T(\vec{x}_i, \vec{x}_{i+1}) \Big) \wedge \bigwedge_{0 \leq i < j \leq k+1} \vec{x}_i \neq \vec{x}_j \wedge \neg p(\vec{x}_{k+1})$$

3. ask a SAT solver for satisfiability of ψ_k
4. if ψ_k is satisfiable, then report $K \not\models \mathsf{G}p$ and construct a counterexample from the obtained model
5. if ψ_k is unsatisfiable, ask a SAT solver for satisfiability of η_k
6. if η_k is unsatisfiable, report $K \models \mathsf{G}p$
7. if η_k is satisfiable, increase k and go to 2

k-induction algorithm for safety properties

1. set k to some initial (relatively low) number
2. construct the formulas

$$\psi_k = S_0(\vec{x}_0) \wedge \bigwedge_{i=0}^{k-1} T(\vec{x}_i, \vec{x}_{i+1}) \wedge \bigvee_{i=0}^{k} \neg p(\vec{x}_i)$$

$$\eta_k = \bigwedge_{i=0}^{k} \Big(p(\vec{x}_i) \wedge T(\vec{x}_i, \vec{x}_{i+1}) \Big) \wedge \bigwedge_{0 \leq i < j \leq k+1} \vec{x}_i \neq \vec{x}_j \wedge \neg p(\vec{x}_{k+1})$$

3. ask a SAT solver for satisfiability of ψ_k
4. if ψ_k is satisfiable, then report $K \not\models \mathrm{G}p$ and construct a counterexample from the obtained model
5. if ψ_k is unsatisfiable, ask a SAT solver for satisfiability of η_k
6. if η_k is unsatisfiable, report $K \models \mathrm{G}p$
7. if η_k is satisfiable, increase k and go to 2

- it terminates as each finite system has a bound on the length of acyclic paths

Final notes

- BMC and *k*-induction are used in practice
- tools CBMC, ESBMC, and ESBMC-kind are successful in SV-COMP
- systems can be described not only by propositional formulas, but also by predicate formulas over a suitable theory
- SMT solvers are then used instead of SAT solvers