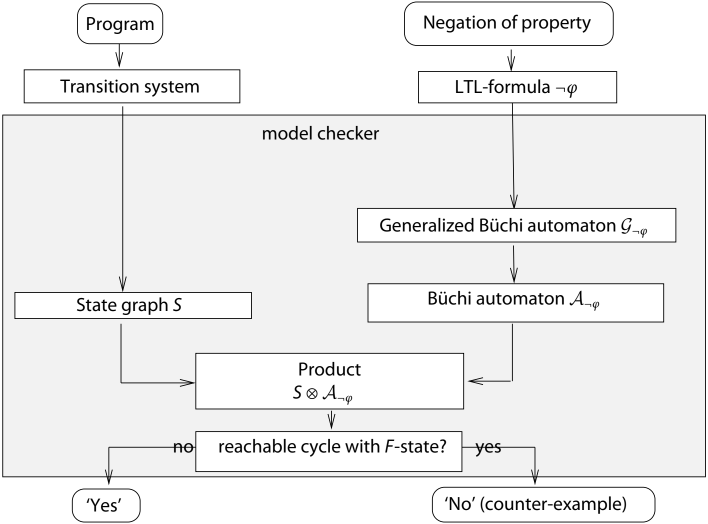
Verification

Lecture 13

Bernd Finkbeiner
Peter Faymonville
Michael Gerke

UNIVERSITÄT
DES
SAARLANDES

REVIEW: (explicit-state) LTL model checking



REVIEW: (explicit-state) CTL model checking

{compute the sets $Sat(\Phi) = \{ q \in S \mid q \models \Phi \}$}
for all $i \leq |\Phi|$ **do**
  for all $\Psi \in Sub(\Phi)$ with $|\Psi| = i$ **do**
    compute $Sat(\Psi)$ from $Sat(\Psi')$ {for maximal proper $\Psi' \in Sub(\Psi)$}
  end for
end for
return $I \subseteq Sat(\Phi)$

$$
\begin{aligned}
Sat(\text{true}) &= Q \\
Sat(a) &= \{ q \in Q \mid a \in L(q) \}, \text{ for any } a \in AP \\
Sat(\Phi \wedge \Psi) &= Sat(\Phi) \cap Sat(\Psi) \\
Sat(\neg\Phi) &= Q \setminus Sat(\Phi) \\
Sat(\text{EX}\,\Phi) &= \{ q \in Q \mid Post(q) \cap Sat(\Phi) \neq \varnothing \}
\end{aligned}
$$

REVIEW: ROBDDs

- Binary decision diagram (OBDD) is a directed graph over $\langle X, < \rangle$ with:
  - each leaf v is labeled with a boolean value $val(v) \in \{0, 1\}$
  - non-leaf v is labeled by a boolean variable $Var(v) \in X$
  - such that for each non-leaf v and vertex w:

$$w \in \{ left(v), right(v) \} \;\Rightarrow\; (Var(v) < Var(w) \;\vee\; w \text{ is a leaf})$$

- OBDD B over $\langle X, < \rangle$ is called reduced (ROBDD) iff:
  1. for each leaf v, w: $(val(v) = val(w)) \;\Rightarrow\; v = w$
  2. for each non-leaf v: $left(v) \neq right(v)$
  3. for each non-leaf v, w:

$$(Var(v) = Var(w) \;\wedge\; right(v) \cong right(w) \;\wedge\; left(v) \cong left(w)) \;\Rightarrow\; v = w$$

REVIEW: The importance of canonicity

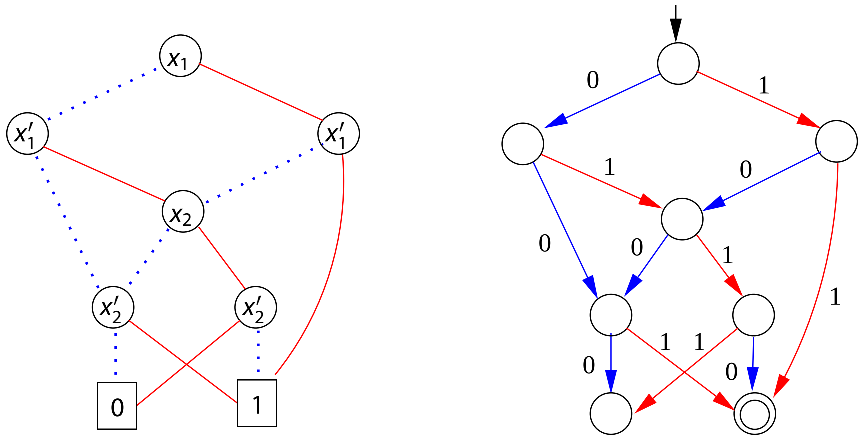
- Absence of redundant vertices
  - if f_{B} does not depend on x_i, ROBDD B does not contain an x_i vertex
- Test for equivalence: $f(x_1, \ldots, x_n) \equiv g(x_1, \ldots, x_n)$?
  - generate ROBDDs B_f and B_g, and check isomorphism
- Test for validity: $f(x_1, \ldots, x_n) = 1$?
  - generate ROBDD B_f and check whether it only consists of a 1-leaf
- Test for implication: $f(x_1, \ldots, x_n) \rightarrow g(x_1, \ldots, x_n)$?
  - generate ROBDD $\mathrm{B}_f \wedge \neg \mathrm{B}_g$ and check if it just consist of a 0-leaf
- Test for satisfiability
  - f is satisfiable if and only if B_f is not just the 0-leaf

Operations on ROBDDs

Algorithm	Output	Time complexity	Space complexity
Reduce	B' (reduced) with $f_{\mathrm{B}} = f_{\mathrm{B}'}$	$\mathcal{O}(\lvert B_f\rvert \cdot \log \lvert \mathrm{B}_f \rvert)$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$
Not	$\mathrm{B}_{\neg f}$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$
Apply	$\mathrm{B}_{f\ op\ g}$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert \cdot \lvert \mathrm{B}_g \rvert)$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert \cdot \lvert \mathrm{B}_g \rvert)$
Restrict	$\mathrm{B}_{f[x:=b]}$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$
Rename	$\mathrm{B}_{f[x:=y]}$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$
Abstract	$\mathrm{B}_{\exists x.\, f}$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert^2)$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert^2)$

operations are only efficient if f and g have compact ROBDD representations

OBDDs versus deterministic automata



each OBDD B is a deterministic automaton A_B with $f_B^{-1}(1) = L(A_B)$

Analogies between ROBDDs and deterministic automata

- For language L, a minimized automaton is unique up to isomorphism
  - for a given variable ordering $<$, and function f, an ROBDD is unique upto $\cong$
- $L = L'$? can be checked by verifying isomorphism of their automata
  - $f = f'$? for boolean functions can be checked by verifying $\mathrm{B}_f \cong \mathrm{B}_{f'}$
  - $\Rightarrow$ in both cases, efficient algorithms do exist for this
- $L \neq \varnothing$? $\equiv$ is there a reachable accept state?
  - is f satisfiable? $\equiv$ its ROBDD has a reachable leaf 1
- Union, intersection, and complementation on det. automata is efficient
  - disjunction, conjunction, and negation on ROBDDs are efficient

Symbolic CTL model checking: Computing $Sat(\Phi)$

Require: CTL-formula Φ in ENF
Ensure: ROBDD $B_{Sat(\Phi)}$

---

switch(Φ):

true	:	**return** Const(1);
false	:	**return** Const(0);
x_i	:	**return** ROBDD B_f for $f(x_1, \ldots, x_n) = x_i$;
$\neg\Psi$	:	**return** Not($bddSat(\Psi)$)
$\Phi_1 \wedge \Phi_2$	:	**return** Apply($\wedge, bddSat(\Phi_1), bddSat(\Phi_2)$)
EX Ψ	:	**return** $bddEX(\Psi)$;
E (Φ_1 U Φ_2)	:	**return** $bddEU(\Phi_1, \Phi_2)$
EG Ψ	:	**return** $bddEG(\Psi)$

end switch

Symbolic CTL model checking: The next-step operator

$$Sat(\mathsf{X}\,\Phi) \;=\; \{\, q \in Q \mid \exists q'.\,(q,q') \in E \text{ and } q' \in Sat(\Phi) \,\}$$

Require: CTL-formula Φ in ENF
Ensure: ROBDD $\mathrm{B}_{Sat(\mathsf{X}\,\Phi)}$

B := $bddSat(\Phi)$; $\{Sat(\Phi)\}$
B := Rename$(\mathrm{B}, x_1, \ldots, x_n, x'_1, \ldots, x'_n)$;
B := Apply$(\wedge, \mathrm{B}_\rho, \mathrm{B})$; $\{Pre(Sat(\Phi))\}$
return Abstract$(\mathrm{B}, x'_1, \ldots, x'_n)$

Symbolic CTL model checking: Existential until

Require: CTL-formulas Φ, Ψ in ENF
Ensure: ROBDD $B_{Sat(\exists(\Phi \, \mathsf{U} \, \Psi))}$

---

var N, P, B : *ROBDD*;
N := $bddSat(\Psi)$;
P := Const(0);
B := $bddSat(\Phi)$;
while $(\mathrm{N} \neq \mathrm{P})$ **do**
  P := N; $\{T_i\}$
  N := Rename$(\mathrm{N}, x_1, \ldots, x_n, x'_1, \ldots, x'_n)$;
  N := Apply$(\wedge, \mathrm{B}_\rho, \mathrm{N})$; $\{Pre(T_i)\}$
  N := Abstract$(\mathrm{N}, x'_1, \ldots, x'_n)$;
  N := Apply$(\wedge, \mathrm{N}, \mathrm{B})$; $\{Pre(T_i) \cap Sat(\Phi)\}$
  N := Apply$(\vee, \mathrm{P}, \mathrm{N})$; $\{T_{i+1} = T_i \cup \ldots\ldots\}$
end while
return N

Symbolic CTL model checking: Possibly always

Require: CTL-formula Φ in ENF
Ensure: ROBDD $B_{Sat(\text{EG}\,\Phi)}$

---

var N, P, B : *ROBDD*;
B := *bddSat*(Φ);
N := B;
P := Const(0);
while (N $\neq$ P) **do**
  P := N; $\{T_i\}$
  N := Rename(N, $x_1, \ldots, x_n, x'_1, \ldots, x'_n$);
  N := Apply($\wedge$, B_ρ, N); $\{Pre(T_i)\}$
  N := Abstract(N, $x'_1, \ldots, x'_n$);
  N := Apply($\wedge$, N, B); $\{Pre(T_i) \cap Sat(\Phi)\}$
  N := Apply($\wedge$, P, N); $\{T_{i+1} = T_i \cap \ldots\ldots\}$
end while
return N

REVIEW: The GNBA of LTL-formula φ

For LTL-formula φ, let $\mathcal{G}_\varphi = (Q, 2^{AP}, \delta, Q_0, \mathcal{F})$ where

- Q = all elementary sets $B \subseteq closure(\varphi)$, $Q_0 = \{ B \in Q \mid \varphi \in B \}$
- $\mathcal{F} = \left\{ \left\{ B \in Q \mid \varphi_1 \mathsf{U} \varphi_2 \notin B \text{ or } \varphi_2 \in B \right\} \mid \varphi_1 \mathsf{U} \varphi_2 \in closure(\varphi) \right\}$
- The transition relation $\delta : Q \times 2^{AP} \to 2^Q$ is given by:
  - If $A \neq B \cap AP$ then $\delta(B, A) = \varnothing$
  - $\delta(B, B \cap AP)$ is the set of all elementary sets of formulas B' satisfying:
    - (i) For every $\mathsf{X}\,\psi \in closure(\varphi)$: $\mathsf{X}\,\psi \in B \iff \psi \in B'$, and
    - (ii) For every $\varphi_1 \mathsf{U} \varphi_2 \in closure(\varphi)$:

$$\varphi_1 \mathsf{U} \varphi_2 \in B \iff \left(\varphi_2 \in B \vee (\varphi_1 \in B \wedge \varphi_1 \mathsf{U} \varphi_2 \in B') \right)$$

A symbolic representation of $S \otimes \mathcal{G}_{\neg\varphi}$

- variables $V \cup \{v_\psi \mid \psi \in el(\varphi) \setminus AP\}$, where
  - $el(p) = \{p\}$ if $p \in AP$,
  - $el(\neg\psi) = el(\psi)$,
  - $el(\psi_1 \wedge \psi_2) = el(\psi_1) \cup el(\psi_2)$,
  - $el(\mathsf{X}\,\psi) = \{\mathsf{X}\,\psi\} \cup el(\psi)$,
  - $el(\psi_1 \,\mathsf{U}\, \psi_2) = \{\psi_1 \,\mathsf{U}\, \psi_2\} \cup el(\psi_1) \cup el(\psi_1)$.
- initial condition $\theta \wedge \neg\underline{\varphi} \wedge \mathit{consistency}$, where
  - $\underline{p} = p$ if $p \in AP$,
  - $\underline{\neg\psi} = \neg\underline{\psi}$,
  - $\underline{\psi_1 \wedge \psi_2} = \underline{\psi_1} \wedge \underline{\psi_2}$,
  - $\underline{\mathsf{X}\,\psi} = v_{\mathsf{X}\,\psi}$,
  - $\underline{\psi_1 \,\mathsf{U}\, \psi_2} = v_{\psi_1 \mathsf{U} \psi_2}$,

  and $\mathit{consistency} = \bigwedge_{(\psi_1 \mathsf{U} \psi_2) \in el(\varphi)} \left(\underline{\psi_2} \to v_{\psi_1 \mathsf{U} \psi_2}\right) \wedge \left(\neg v_{\psi_1 \mathsf{U} \psi_2} \vee \underline{\psi_1} \vee \underline{\psi_2}\right)$.

A symbolic representation of $S \otimes \mathcal{G}_{\neg\varphi}$, cont'd

- transition relation ρ:

$$\mathit{consistency}' \;\wedge \bigwedge_{\mathsf{X}\,\psi \in el(\varphi)} \underline{\mathsf{X}\,\psi} \leftrightarrow \underline{\psi}'$$

$$\wedge \bigwedge_{\psi_1\,\mathsf{U}\,\psi_2 \in el(\varphi)} \underline{\psi_1\,\mathsf{U}\,\psi_2} \leftrightarrow \underline{\psi_2} \vee \left(\underline{\psi_1} \wedge \underline{\psi_1\,\mathsf{U}\,\psi_2}'\right)$$

- acceptance condition $F = \bigwedge_{\psi_1\,\mathsf{U}\,\psi_2 \in el(\varphi)} \square \lozenge F_{\psi_1\,\mathsf{U}\,\psi_2}$ where

$$F_{\psi_1\,\mathsf{U}\,\psi_2} = \neg\left(\underline{\psi_1\,\mathsf{U}\,\psi_2}\right) \vee \underline{\psi_2}.$$

Symbolic Emptiness Check

The language of $S \otimes \mathcal{G}_{\neg\varphi}$ is nonempty
iff there exists a non-empty set Z of reachable states such that

> for all states $s \in Z$ and for all $\psi_1 \,\mathsf{U}\, \psi_2 \in el(\varphi)$,
> there is a path of length ≥ 1 to a state in $Z \cap Sat(F_{\psi_1 \mathsf{U} \psi_2})$.

Symbolic Emptiness Check

The language of $S \otimes \mathcal{G}_{\neg\varphi}$ is nonempty
iff there exists a non-empty set Z of reachable states such that

for all states $s \in Z$ and for all $\psi_1 \,\mathsf{U}\, \psi_2 \in el(\varphi)$,
there is a path of length ≥ 1 to a state in $Z \cap Sat(F_{\psi_1 \,\mathsf{U}\, \psi_2})$.

1. Compute Z as the greatest fixpoint of the equation

$$Z = \bigcap_{\psi_1 \,\mathsf{U}\, \psi_2 \in el(\varphi)} Sat(\mathrm{EX}\,\mathrm{EF}\,(a_Z \wedge F_{\psi_1 \,\mathsf{U}\, \psi_2}))$$

   where a_Z is true iff a state is in Z.
2. Check if the intersection of Z and the initial states is non-empty.

Bounded Model Checking

BDD vs. SAT based approaches

BDD-based approaches

- Approach used by many "industrial-strength" model checkers
- Hundreds of state variables
- Canonical representation $\Rightarrow$ BDDs often too large
- Variable order uniform along all paths, selection of good order very difficult

SAT-based approaches

- Avoid space explosion of BDDs
- Different split orders possible on different branches
- Very efficient implementations available

Basic idea

Search for counterexamples of bounded length

There exists a counterexample of length k to the invariant AG p
iff the following formula is satisfiable:

$$f_I(\vec{v}_0) \wedge f_\to(\vec{v}_0, \vec{v}_1) \wedge f_\to(\vec{v}_1, \vec{v}_2) \wedge \ldots f_\to(\vec{v}_{k-2}, \vec{v}_{k-1}) \wedge (\neg p_0 \vee \neg p_1 \vee \ldots \vee \neg p_{k-1})$$

Example: two-bit counter

- Initial state: $f_I = (\neg l \wedge \neg r)$
- Transition: $f_{\rightarrow}(l, r, l', r') = (r' \leftrightarrow \neg r) \ \wedge \ (l' \leftrightarrow (l \leftrightarrow \neg r))$
- Property: $\mathrm{AG}\,(\neg l \vee \neg r)$

Counterexample of length 3?

$$\underbrace{\neg l_0 \wedge \neg r_0}_{f_I(\vec{v}_0)} \wedge \underbrace{r_1 \leftrightarrow \neg r_0 \ \wedge \ l_1 \leftrightarrow (l_0 \leftrightarrow \neg r_0)}_{f_{\rightarrow}(\vec{v}_0,\vec{v}_1)}$$

$$\wedge \underbrace{r_2 \leftrightarrow \neg r_1 \ \wedge \ l_2 \leftrightarrow (l_1 \leftrightarrow \neg r_1)}_{f_{\rightarrow}(\vec{v}_1,\vec{v}_2)} \wedge (\underbrace{l_0 \wedge r_0}_{\neg p_0} \vee \underbrace{l_1 \wedge r_1}_{\neg p_1} \vee \underbrace{l_2 \wedge r_2}_{\neg p_2})$$

unsatisfiable $\Rightarrow$ no counterexample

Example: two-bit counter

- Initial state: $f_I = (\neg l \wedge \neg r)$
- Transition: $f_{\rightarrow}(l, r, l', r') = (r' \leftrightarrow \neg r) \ \wedge \ (l' \leftrightarrow (l \leftrightarrow \neg r))$
- Property: $\mathrm{AG}\,(\neg l \vee \neg r)$

Counterexample of length 4?

$$\underbrace{\neg l_0 \wedge \neg r_0}_{f_I(\vec{v}_0)} \wedge \underbrace{r_1 \leftrightarrow \neg r_0 \ \wedge \ l_1 \leftrightarrow (l_0 \leftrightarrow \neg r_0)}_{f_{\rightarrow}(\vec{v}_0,\vec{v}_1)} \wedge \underbrace{r_2 \leftrightarrow \neg r_1 \ \wedge \ l_2 \leftrightarrow (l_1 \leftrightarrow \neg r_1)}_{f_{\rightarrow}(\vec{v}_1,\vec{v}_2)}$$

$$\wedge \ \underbrace{r_3 \leftrightarrow \neg r_2 \ \wedge \ l_3 \leftrightarrow (l_2 \leftrightarrow \neg r_2)}_{f_{\rightarrow}(\vec{v}_2,\vec{v}_3)} \wedge (\underbrace{l_0 \wedge r_0}_{\neg p_0} \vee \underbrace{l_1 \wedge r_1}_{\neg p_1} \vee \underbrace{l_2 \wedge r_2}_{\neg p_2} \vee \underbrace{l_3 \wedge r_3}_{\neg p_3})$$

satisfiable $\Rightarrow$ counterexample!

SAT

- Given a propositional formula ψ,
  does there exist a variable assignment
  under which ψ evaluates to true?
- NP-complete
- In practice, tremendous progress over the last years
- Most solvers use Conjunctive Normal Form (CNF)
- Arbitrary formulas can be transformed in polynomial time into satisfiability equivalent formulas in CNF

Davis-Putnam-Logemann-Loveland (DPLL) algorithm

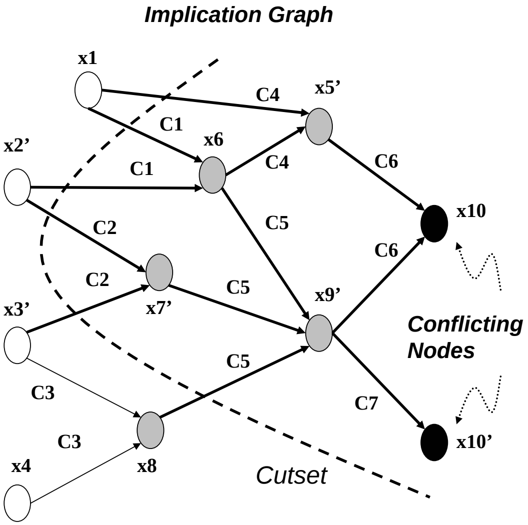
```
if preprocess() = CONFLICT then
        return UNSAT;
while TRUE do
        if not decide-next-branch() then
                return SAT;
        while deduce() = CONFLICT do
                blevel := analyze-conflict();
                if blevel=0 then
                        return UNSAT;
                backtrack(blevel);
        done;
done;
```

Conflict analysis using an implication graph

Clauses:
C1: x1'+ x2 + x6
C2: x2 + x3 + x7'
C3: x3 + x4'+ x8
C4: x1'+ x6'+ x5'
C5: x6'+ x7+ x8'+ x9'
C6: x5 + x9 + x10
C7: x9 + x10'

Conflict Clause C8:
x1'+ x2 + x3 + x8'

Due to conflict
(x10, x10')



Efficiency

- conflict learning: adding conflict clauses
- non-chronological backtracking
- heuristics for decisions
- efficient data structures
- incremental satisfiability

Bounded LTL model checking

Automata-based approach:

- Translate LTL formula $\neg\varphi$ to Büchi automaton
- Build product with transition system
- Encode all paths that start in initial state and are k steps long
- Require that path contains loop with accepting state

$$f_I(\vec{v}_0) \;\wedge\; \bigwedge_{i=0}^{k-2} f_{\rightarrow}(\vec{v}_i, \vec{v}_{i+1}) \;\wedge\; \bigvee_{i=0}^{k-1} \Big((\vec{v}_i = \vec{v}_k) \wedge \bigvee_{j=i}^{k-1} f_F(\vec{v}_j) \Big)$$

Formula size: $O(k \cdot |TS| \cdot 2^{|\varphi|})$

Fixpoint-based translation

$$\psi_{TS} \wedge \psi_{loop} \wedge [\psi]_0$$

- $\psi_{TS} = f_I(\vec{v}_0) \;\wedge\; \bigwedge_{i=0}^{k-2} f_\rightarrow(\vec{v}_i, \vec{v}_{i+1})$
- ψ_{loop}: loop constraint, ensures the existence of exactly one loop
- $[\varphi]_0$: fixpoint formula, ensures that LTL formula holds

Formula size: $O(k \cdot (|TS| + |\varphi|))$

Loop constraint

- $\psi_{loop} = AtLeastOneLoop \wedge AtMostOneLoop$
- $AtLeastOneLoop = \bigwedge_{i=0}^{k-2}(l_i \Rightarrow (\vec{v}_i = \vec{v}_{k-1}))$
- $AtMostOneLoop = \bigwedge_{i=0}^{k-2}(SmallerExists_i \Rightarrow \neg l_i$
- $\mathrm{SmallerExists}_0 = \mathit{false}$
- $\mathrm{SmallerExists}_{i+1} = \mathrm{SmallerExists}_i \vee l_i$ for $0 \leq i < k - 1$.

Fixpoint formula

Let φ be in PNF.

- $[p]_i = p_i$ for $i < k-1$
  $[p]_i = \bigvee_{j=0}^{k-2}(l_j \wedge p_j)$ for $i = k-1$
- $[\neg p]_i = \neg p_i$ for $i < k-1$
  $[\neg p]_i = \bigvee_{j=0}^{k-2}(l_j \wedge \neg p_j)$ for $i = k-1$
- $[\bigcirc \varphi']_i = [\varphi']_{i+1}$ for $i < k-2$
  $[\bigcirc \varphi']_i = \bigvee_{j=0}^{k-2}(l_j \wedge [\varphi'])$ for $i = k-2$
- $[\varphi_1 \ \mathrm{U} \ \varphi_2]_i = [\varphi_2]_i \vee ([\varphi_1]_i \wedge [\varphi_1 \ \mathrm{U} \ \varphi_2]_{i+1}$ for $i < k-1$
  $[\varphi_1 \ \mathrm{U} \ \varphi_2]_i = \bigvee_{j=0}^{k-2}(l_j \wedge \langle \varphi_1 \ \mathrm{U} \ \varphi_2 \rangle_j)$ for $i = k-1$
- $[\varphi_1 \ \mathrm{R} \ \varphi_2]_i = [\varphi_2]_i \wedge ([\varphi_1]_i \vee [\varphi_1 \ \mathrm{R} \ \varphi_2]_{i+1}$ for $i < k-1$
  $[\varphi_1 \ \mathrm{R} \ \varphi_2]_i = \bigvee_{j=0}^{k-2}(l_j \wedge \langle \varphi_1 \ \mathrm{R} \ \varphi_2 \rangle_j)$ for $i = k-1$
- $\langle \varphi_1 \ \mathrm{U} \ \varphi_2 \rangle_i = [\varphi_2]_i \vee ([\varphi_1]_i \wedge \langle \varphi_1 \ \mathrm{U} \ \varphi_2 \rangle_{i+1}$ for $i < k-1$
  $\langle \varphi_1 \ \mathrm{U} \ \varphi_2 \rangle_i = \mathit{false}$ for $i = k-1$
- $\langle \varphi_1 \ \mathrm{R} \ \varphi_2 \rangle_i = [\varphi_2]_i \wedge ([\varphi_1]_i \vee \langle \varphi_1 \ \mathrm{R} \ \varphi_2 \rangle_{i+1}$ for $i < k-1$
  $\langle \varphi_1 \ \mathrm{R} \ \varphi_2 \rangle_i = \mathit{true}$ for $i = k-1$

The Completeness Threshold

The bound k is increased incrementally until

- a counterexample is found, or
- the problem becomes intractable due to the complexity of the SAT problem
- k reaches a precomputed threshold that guarantees that there is no counterexample

$\rightarrow$ this threshold is called the completeness threshold CL.

The completeness threshold

- Computing *CL* is as hard as model checking
- Idea: Compute an overapproximation of *CL* based on the graph structure

Basic notions:

- Diameter *D*: Longest shortest path between any two reachable states
- Recurrence diameter *RD*: Longest loop-free path between any two reachable states
- Initialized diameter D^I: Longest shortest path between some initial state and some reachable state
- Initialized recurrence diameter RD^I: Longest loop-free path between some initial state and some reachable state