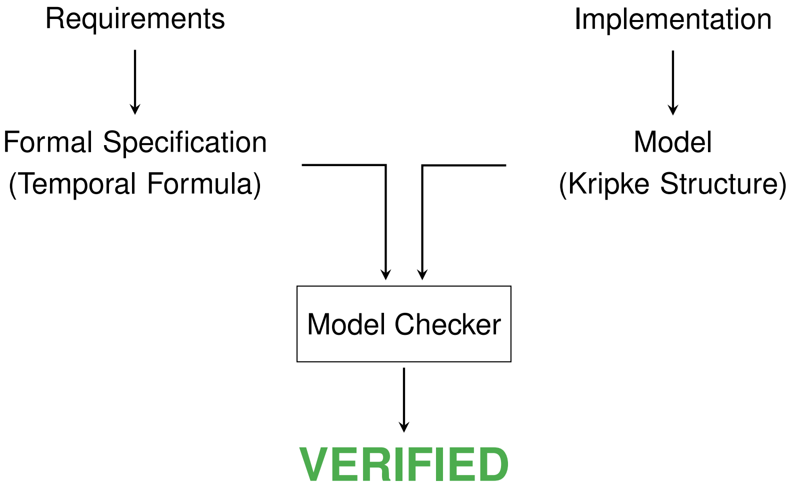
SAT-BASED BOUNDED MODEL CHECKING

Formal Models SS16

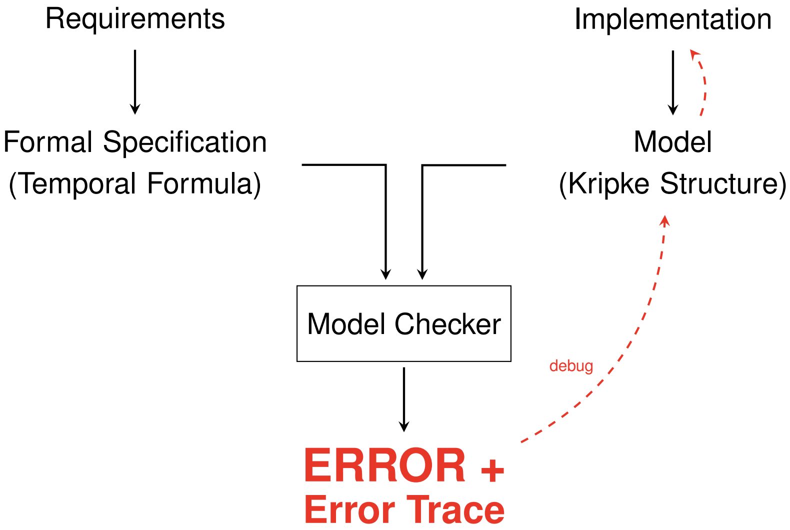
Martina Seidl
Institute for Formal Models and Verification

JOHANNES KEPLER UNIVERSITY LINZ

Model Checking



Model Checking



Types of Model Checking

General question: Given a system K and a property p, does p hold for K (i.e., for all initial states of K) ?

- Explicit state model checking
  - enumeration of the state space
  - state explosion problem
- Symbolic model checking
  - representation of model checking problem as logical formula (e.g., in propositional logic (SAT) or QBF)

Some Properties

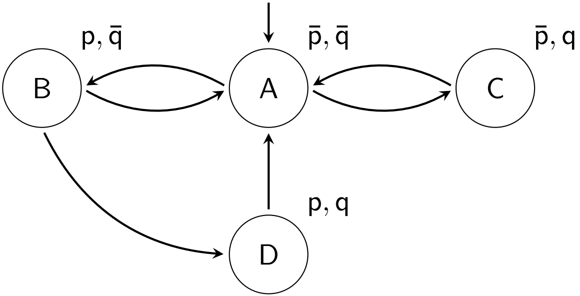
- **Reachability**: property p holds in one reachable state
- **Invariant**: property p holds in all reachable states
- **Safety**: some bad property p never holds
  "something bad will never happen"
- **Liveness**: something good will eventually happen
- **Fairness**: under certain conditions, some property holds repeatedly

Example: Mutual Exclusion

Given two processes P and Q which share a resource R.

- If R is accessed by P, then property p is true.
- If R is accessed by Q, then property q is true.

The bahavior of P and Q is modeled by this Kripke structure:



Question: Does $F(p \wedge q)$ hold?

Limboole

- SAT-solver for formulas in non-CNF
- available at `http://fmv.jku.at/limboole/`
- input format in BNF:

$$
\begin{aligned}
\langle expr \rangle &::= \langle \mathit{iff} \rangle \\
\langle \mathit{iff} \rangle &::= \langle implies \rangle \mid \langle implies \rangle \text{ “}\texttt{<->}\text{” } \langle implies \rangle \\
\langle implies \rangle &::= \langle or \rangle \mid \langle or \rangle \text{ “}\texttt{->}\text{” } \langle or \rangle \mid \langle or \rangle \text{ “}\texttt{<-}\text{” } \langle or \rangle \\
\langle or \rangle &::= \langle and \rangle \mid \langle and \rangle \text{ “}\texttt{|}\text{” } \langle and \rangle \\
\langle and \rangle &::= \langle not \rangle \mid \langle not \rangle \text{ “}\texttt{\&}\text{” } \langle not \rangle \\
\langle not \rangle &::= \langle basic \rangle \mid \text{“}\texttt{!}\text{” } \langle not \rangle \\
\langle basic \rangle &::= \langle var \rangle \mid \text{“}\texttt{(}\text{” } \langle expr \rangle \text{ “}\texttt{)}\text{”}
\end{aligned}
$$

where ’var’ is a string over letters, digits, and

`- _ . [ ] $ @`

Symbolic Encoding of Kripke Structures

Given Kripke structure $K = (S, I, T, L)$ over $\mathcal{A} = \{a_1, \ldots, a_n\}$.

1. Introduce sets $\mathcal{A}' = \{a'_1, \ldots, a'_n\}$ and $\mathcal{A}'' = \{a''_1, \ldots, a''_n\}$ for the **definition of one transition step $\mathcal{T}$ over $\mathcal{A}'$ and $\mathcal{A}''$**.
2. Associate each state $s \in S$ with two conjunctions of literals $current(s)$ and $next(s)$:[1]
   - $current(s) := (l_1 \wedge \ldots \wedge l_n)$
     such that $l_i = a'_i$ if $a_i \in L(s)$ else $l_i = \bar{a}'_i$;
   - $next(s) := (k_1 \wedge \ldots \wedge k_n)$
     such that $k_i = a''_i$ if $a_i \in L(s)$ else $k_i = \bar{a}''_i$.
3. Define prop. formula $\mathcal{T}$ over $\mathcal{A}', \mathcal{A}''$ such that $\forall s_i, s_j \in S$ $(\mathcal{T} \wedge current(s_i) \wedge next(s_j))$ **is satisfiable iff** $(s_i, s_j) \in T$.

[1] note: the mapping "state to conjunction" has to be bijective

Naive Encoding of Kripke Structures in SAT

Let $K = (S, I, T, L)$ be a Kripke structure over $\mathcal{A}$.

$\mathcal{T} := \top$

while $S \neq \emptyset$ **do**

  select $s \in S$

  $S := S \backslash \{s\}$

  $N := \bot$

  for all $(s, t) \in T$ **do**

    $N := N \vee next(t)$

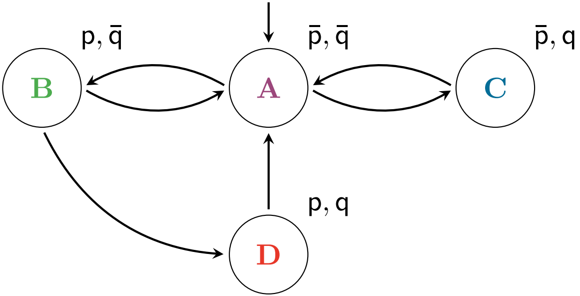
  end for

  $\mathcal{T} := \mathcal{T} \wedge (current(s) \rightarrow N)$

end while

return $\mathcal{T}$

Naive Encoding of Kripke Structures in SAT



$$
\begin{aligned}
\mathcal{T} \;:=\; & \top & \wedge \\
& (\bar{p} \wedge \bar{q}) \rightarrow (\bot \vee (\bar{p}' \wedge q') \vee (p' \wedge \bar{q}')) & \wedge \\
& (p \wedge \bar{q}) \rightarrow (\bot \vee (\bar{p}' \wedge \bar{q}') \vee (p' \wedge q')) & \wedge \\
& (\bar{p} \wedge q) \rightarrow (\bot \vee (\bar{p}' \wedge \bar{q}')) & \wedge \\
& (p \wedge q) \rightarrow (\bot \vee (\bar{p}' \wedge \bar{q}')) &
\end{aligned}
$$

Naive Encoding of Kripke Structures in SAT

Encoding in Limboole syntax:

```
((!p & !q)  -> (!p-next & q-next) | (p-next & !q-next)) &
((p & !q)   -> (!p-next & !q-next) | (p-next & q-next)) &
((!p & q)   -> (!p-next & !q-next)) &
((p & q)   -> (!p-next & !q-next))
```

```
> limboole limboole/mutual.boole -s
% SATISFIABLE formula (satisfying assignment follows)

p = 0
q = 0
p-next = 0
q-next = 1
```

Symbolic Encoding of Kripke Structures

Alternative encoding of transition function:

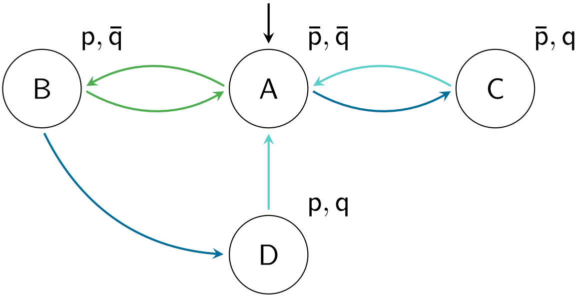
Successor states p', q':

$(p' \leftrightarrow (\bar{p} \wedge \bar{q}) \wedge q' \leftrightarrow 0)$

$\vee$

$(p' \leftrightarrow (p \wedge \bar{q}) \wedge q' \leftrightarrow \bar{q})$

p	q	p′	q′	or	p′	q′
0	0	1	0		0	1
0	1	0	0		0	0
1	0	0	0		1	1
1	1	0	0		0	0



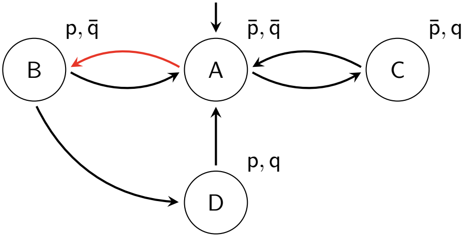
Example: One Step

Encoding in Limboole syntax:

```
(((p-next <-> (!p & !q)) & (!q-next)) |
((p-next <-> (p & !q)) & (q-next <-> !q)))

> limboole -s mutual2.boole
% SATISFIABLE formula (satisfying assignment follows)

p = 0
q = 0
p-next = 1
q-next = 0
```



Multiple Transtion Steps

- $\mathcal{T}$ over $\mathcal{A}'$ and $\mathcal{A}''$ defines one transition step
  - we also write $\mathcal{T}(s_0, s_1)$ indicating that we can go from state a s_0 to a state s_1
- $\mathcal{T}$ over $\mathcal{A}''$ and $\mathcal{A}'''$ defines one transition step
  - we also write $\mathcal{T}(s_1, s_2)$ indicating that we can go from state a s_1 to a state s_2
- $\mathcal{T}(s_0, s_1) \wedge \mathcal{T}(s_1, s_2)$ defines two transition steps from a state s_0 to a state s_1
- Example (previous slides):

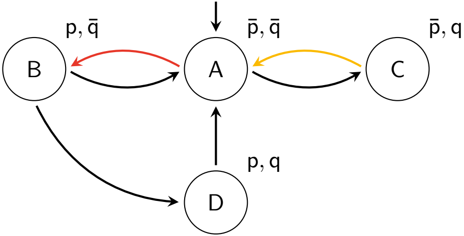
$$(((\mathsf{p}' \leftrightarrow (\bar{\mathsf{p}} \wedge \bar{\mathsf{q}})) \wedge (\mathsf{q}' \leftrightarrow 0)) \vee ((\mathsf{p}' \leftrightarrow (\mathsf{p} \wedge \bar{\mathsf{q}})) \wedge (\mathsf{q}' \leftrightarrow \bar{\mathsf{q}}))) \quad \wedge$$
$$(((\mathsf{p}'' \leftrightarrow (\bar{\mathsf{p}}' \wedge \bar{\mathsf{q}}')) \wedge (\mathsf{q}'' \leftrightarrow 0)) \vee ((\mathsf{p}'' \leftrightarrow (\mathsf{p}' \wedge \bar{\mathsf{q}}')) \wedge (\mathsf{q}'' \leftrightarrow \bar{\mathsf{q}}')))$$

Example: Two Steps

Encoding in Limboole syntax:

```
((((p-next <-> (!p & !q)) & (!q-next))  |
((p-next <-> (p & !q)) & (q-next <-> !q)))) &
((((p-next2 <-> (!p-next & !q-next)) & (!q-next2)) |
((p-next2 <-> (p-next & !q-next)) & (q-next2 <-> !q-next))))
```

```
> limboole -s mutual2-twoSteps.boole
% SATISFIABLE formula (satisfying assignment follows)
p = 0
q = 1
p-next = 0
q-next = 0
p-next2 = 1
q-next2 = 0
```

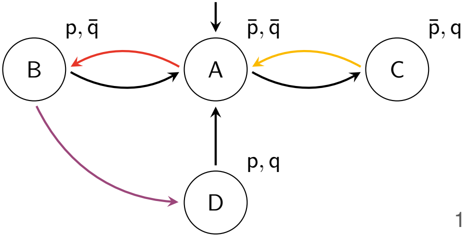


Example: Three Steps

Encoding in Limboole syntax:

```
((((p-next <-> (!p & !q)) & (!q-next))  |
((p-next <-> (p & !q)) & (q-next <-> !q)))) &
((((p-next2 <-> (!p-next & !q-next)) & (!q-next2)) |
((p-next2 <-> (p-next & !q-next)) & (q-next2 <-> !q-next)))) &
((((p-next3 <-> (!p-next2 & !q-next2)) & (!q-next3)) |
((p-next3 <-> (p-next2 & !q-next2)) & (q-next3 <-> !q-next2))))
```

```
limboole -s mutual2-threeSteps.boole
% SATISFIABLE formula (satisfying assignment follows)
p = 0
q = 1
p-next = 0
q-next = 0
p-next2 = 1
q-next2 = 0
p-next3 = 1
q-next3 = 1
```



Bounded Model Checking (Safety)

- Given a Kripke structure K. Is there a path of length k to a **bad state** s, i.e., a certain property p is violated in s?
- In other words: there is a path where $\mathsf{G}p$ does not hold in K
- Observation: if $\mathsf{G}p$ does not hold in K, there is a **finite counter-example.**
- Idea: consider paths of fixed length k
  - encode problem to propositional formula ϕ
  - pass problem to SAT solver
  - ϕ is true $\Leftrightarrow$ model of ϕ is counter-example
  - if ϕ is false, then increase k

Bounded Model Checking (Safety)

A bounded model checking (BMC) problem for Kripke structure K and safety property G*p* is encoded by

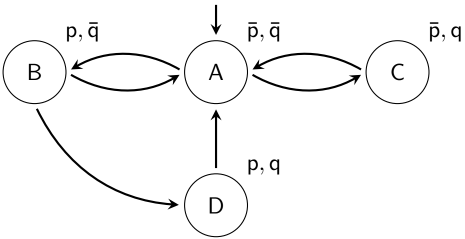
$$I(s_0) \wedge \mathcal{T}(s_0, s_1) \wedge \mathcal{T}(s_1, s_2) \wedge \ldots \wedge \mathcal{T}(s_{k-1}, s_k) \wedge B(s_k)$$

where

- $I(s_0)$ is true $\Leftrightarrow$ s_0 is an initial state
- $\mathcal{T}$ is the transition function of K
- $B(s_k)$ is true $\Leftrightarrow$ s_k is a bad state, i.e., $\neg p$ holds in s_k

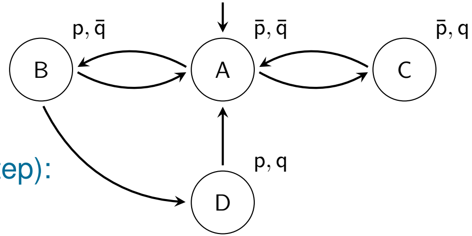
BMC Example

We want to know if $\mathsf{G}(\bar{\mathsf{p}} \vee \bar{\mathsf{q}})$ holds for Kripke structure K:



BMC Example

We want to know if $\mathsf{G}(\bar{\mathsf{p}} \vee \bar{\mathsf{q}})$ holds for Kripke structure K:



$k = 1$ (one step):

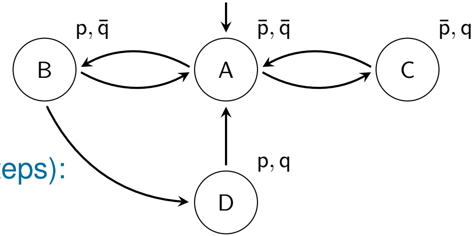
$(\bar{p} \wedge \bar{q}) \quad \wedge$

$((p' \leftrightarrow (\bar{p} \wedge \bar{q})) \wedge (q' \leftrightarrow 0)) \vee ((p' \leftrightarrow (p \wedge \bar{q})) \wedge (q' \leftrightarrow \bar{q})) \quad \wedge$

$(p' \wedge q')$

BMC Example

We want to know if $\mathsf{G}(\bar{p} \vee \bar{q})$ holds for Kripke structure K:



$k = 2$ (two steps):

$(\bar{p} \wedge \bar{q}) \quad \wedge$

$(((p' \leftrightarrow (\bar{p} \wedge \bar{q})) \wedge (q' \leftrightarrow 0)) \vee ((p' \leftrightarrow (p \wedge \bar{q})) \wedge (q' \leftrightarrow \bar{q})) \quad \wedge$

$(((p'' \leftrightarrow (\bar{p}' \wedge \bar{q}')) \wedge (q'' \leftrightarrow 0)) \vee$

$((p'' \leftrightarrow (p' \wedge \bar{q}')) \wedge (q'' \leftrightarrow \bar{q}'))) \quad \wedge$

$(p'' \wedge q'')$

Bounded Model Checking (Safety) in QBF

A bounded model checking (BMC) problem for Kripke structure K and reachability property p is encoded by

$$\exists s_0, s_1, \ldots, s_k \forall x, x'. \quad (I(s_0) \quad \wedge \quad B(s_k) \quad \wedge \quad (\bigvee_{i=0}^{k-1} (x \leftrightarrow s_i \wedge x' \leftrightarrow s_{i+1}) \rightarrow \mathcal{T}(x, x')))$$

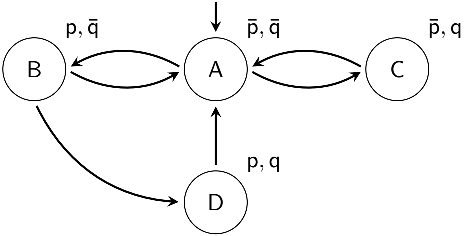
where

- $I(s_0)$ is true $\Leftrightarrow$ s_0 is an initial state
- $\mathcal{T}$ is the transition function of K
- $B(s_k)$ is true $\Leftrightarrow$ s_k is a bad state, i.e., p holds in s_k

Advantage: only one copy of transition function!

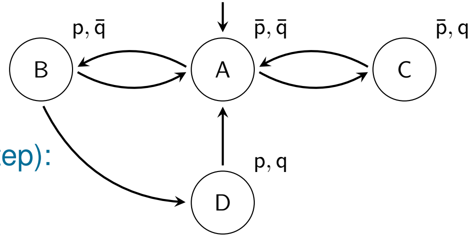
BMC Example in QBF

We want to know if $\mathsf{G}(\bar{\mathsf{p}} \vee \bar{\mathsf{q}})$ holds for Kripke structure K:



BMC Example in QBF

We want to know if $\mathsf{G}(\bar{p} \vee \bar{q})$ holds for Kripke structure K:



$k = 1$ (one step):

$\exists p, q, p', q' \forall x, y, x', y'.$

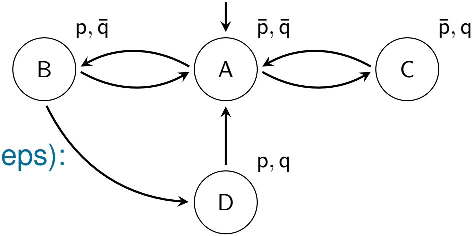
$(\bar{p} \wedge \bar{q}) \quad \wedge \quad (p' \wedge q') \quad \wedge$

$(((x \leftrightarrow p) \wedge (y \leftrightarrow q) \wedge (x' \leftrightarrow p') \wedge (y' \leftrightarrow q')) \rightarrow$

$((x' \leftrightarrow (\bar{x} \wedge \bar{y})) \wedge (y' \leftrightarrow 0)) \vee ((x' \leftrightarrow (x \wedge \bar{y})) \wedge (y' \leftrightarrow \bar{y})))$

BMC Example in QBF

We want to know if $\mathsf{G}(\bar{p} \vee \bar{q})$ holds for Kripke structure K:



$k = 2$ (two steps):

$$\exists p, q, p', q', p'', q'' \forall x, y, x', y'.$$

$$(\bar{p} \wedge \bar{q}) \quad \wedge \quad (p'' \wedge q'') \quad \wedge$$

$$((((x \leftrightarrow p) \wedge (y \leftrightarrow q) \wedge (x' \leftrightarrow p') \wedge (y' \leftrightarrow q')) \quad \vee$$

$$(((x \leftrightarrow p') \wedge (y \leftrightarrow q') \wedge (x' \leftrightarrow p'') \wedge (y' \leftrightarrow q'')))) \rightarrow$$

$$((x' \leftrightarrow (\bar{x} \wedge \bar{y})) \wedge (y' \leftrightarrow 0)) \vee ((x' \leftrightarrow (x \wedge \bar{y})) \wedge (y' \leftrightarrow \bar{y})))$$

Transformation to PCNF

We assume QBF in prenex normal form, i.e., we have a quantifier prefix Π and propositional formula ϕ.

1. for subformula ψ of ϕ' introduce new variable x_ψ
2. replace ψ by x_ψ
3. add definition $\psi \leftrightarrow x_\psi$ as clauses
4. collect all constraints in a big conjunction
5. add existential quantifier with new variables at the end of the prefix

The transformation is **satisfiability equivalent**:
the result is satisfiable iff the original formula is satisfiable

Normalform Transformation: Definitions

Negation

$$\begin{aligned} x \leftrightarrow \overline{y} \quad &\Leftrightarrow \quad (x \rightarrow \overline{y}) \wedge (\overline{y} \rightarrow x) \\ &\Leftrightarrow \quad (\overline{x} \vee \overline{y}) \wedge (y \vee x) \end{aligned}$$

Normalform Transformation: Definitions

Disjunction

$$
\begin{aligned}
x \leftrightarrow (y \vee z) \quad &\Leftrightarrow \quad (y \rightarrow x) \wedge (z \rightarrow x) \wedge (x \rightarrow (y \vee z)) \\
&\Leftrightarrow \quad (\overline{y} \vee x) \wedge (\overline{z} \vee x) \wedge (\overline{x} \vee y \vee z)
\end{aligned}
$$

Normalform Transformation: Definitions

Conjunction

$$
\begin{aligned}
x \leftrightarrow (y \wedge z) \quad &\Leftrightarrow \quad (x \rightarrow y) \wedge (x \rightarrow z) \wedge ((y \wedge z) \rightarrow x) \\
&\Leftrightarrow \quad (\overline{x} \vee y) \wedge (\overline{x} \vee z) \wedge (\overline{(y \wedge z)} \vee x) \\
&\Leftrightarrow \quad (\overline{x} \vee y) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z} \vee x)
\end{aligned}
$$

Normalform Transformation: Definitions

Equivalence

$$
\begin{aligned}
x \leftrightarrow (y \leftrightarrow z) \quad &\Leftrightarrow \quad (x \rightarrow (y \leftrightarrow z)) \wedge ((y \leftrightarrow z) \rightarrow x) \\
&\Leftrightarrow \quad (x \rightarrow ((y \rightarrow z) \wedge (z \rightarrow y)) \wedge ((y \leftrightarrow z) \rightarrow x) \\
&\Leftrightarrow \quad (x \rightarrow (y \rightarrow z)) \wedge (x \rightarrow (z \rightarrow y)) \wedge ((y \leftrightarrow z) \rightarrow x) \\
&\Leftrightarrow \quad (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{z} \vee y) \wedge ((y \leftrightarrow z) \rightarrow x) \\
&\Leftrightarrow \quad (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{z} \vee y) \wedge (((y \wedge z) \vee (\overline{y} \wedge \overline{z})) \rightarrow x) \\
&\Leftrightarrow \quad (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{z} \vee y) \wedge ((y \wedge z) \rightarrow x) \wedge ((\overline{y} \wedge \overline{z}) \rightarrow x) \\
&\Leftrightarrow \quad (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{z} \vee y) \wedge (\overline{y} \vee \overline{z} \vee x) \wedge (y \vee z \vee x)
\end{aligned}
$$

Transformation to PCNF: Example

$$\exists p, q, p', q' \forall x, y, x', y'.$$
$$(\bar{p} \land \bar{q}) \quad \land \quad (p' \land q') \quad \land$$
$$(((x \leftrightarrow p) \land (y \leftrightarrow q) \land (x' \leftrightarrow p') \land (y' \leftrightarrow q')) \rightarrow$$
$$((x' \leftrightarrow (\bar{x} \land \bar{y})) \land (y' \leftrightarrow 0)) \lor ((x' \leftrightarrow (x \land \bar{y})) \land (y' \leftrightarrow \bar{y})))$$

Transformation to PCNF: Example

$\exists p, q, p', q' \forall x, y, x', y'.$

$(\bar{p} \wedge \bar{q}) \quad \wedge \quad (p' \wedge q') \quad \wedge$

$(((x \leftrightarrow p) \wedge (y \leftrightarrow q) \wedge (x' \leftrightarrow p') \wedge (y' \leftrightarrow q')) \rightarrow$

$((x' \leftrightarrow (\bar{x} \wedge \bar{y})) \wedge (y' \leftrightarrow 0)) \vee ((x' \leftrightarrow (x \wedge \bar{y})) \wedge (y' \leftrightarrow \bar{y})))$

$\exists p, q, p', q' \forall x, y, x', y' \exists a, b, c, d, e, f, g, h, i, j, k.$

...

$(\bar{x} \vee \bar{a} \vee p) \wedge (\bar{x} \vee \bar{p} \vee a) \wedge (\bar{a} \vee \bar{p} \vee x) \wedge (a \vee p \vee x) \quad \wedge$

$(\bar{y} \vee \bar{b} \vee q) \wedge (\bar{y} \vee \bar{q} \vee b) \wedge (\bar{b} \vee \bar{q} \vee y) \wedge (b \vee q \vee y) \quad \wedge$

$(\bar{x}' \vee \bar{c} \vee p') \wedge (\bar{x}' \vee \bar{p}' \vee c) \wedge (\bar{c} \vee \bar{p}' \vee x') \wedge (c \vee p' \vee x') \quad \wedge$

$(\bar{y}' \vee \bar{d} \vee q') \wedge (\bar{y}' \vee \bar{q}' \vee d) \wedge (\bar{d} \vee \bar{q}' \vee y') \wedge (d \vee q' \vee y') \quad \wedge$

...

Transformation to PCNF: Example

$\exists p, q, p', q' \forall x, y, x', y'.$

$(\bar{p} \wedge \bar{q}) \quad \wedge \quad (p' \wedge q') \quad \wedge$

$(((x \leftrightarrow p) \wedge (y \leftrightarrow q) \wedge (x' \leftrightarrow p') \wedge (y' \leftrightarrow q')) \rightarrow$

$((x' \leftrightarrow (\bar{x} \wedge \bar{y})) \wedge (y' \leftrightarrow 0)) \vee ((x' \leftrightarrow (x \wedge \bar{y})) \wedge (y' \leftrightarrow \bar{y})))$

$\exists p, q, p', q' \forall x, y, x', y' \exists a, b, c, d, e, f, g, h, i, j, k.$

...

$(\bar{e} \vee \bar{x}) \wedge (\bar{e} \vee \bar{y}) \wedge (e \vee x \vee y) \quad \wedge$

$(\bar{f} \vee x) \wedge (\bar{f} \vee \bar{y}) \wedge (f \vee \bar{x} \vee y) \quad \wedge$

...

Transformation to PCNF: Example

$\exists p, q, p', q' \forall x, y, x', y'.$

$(\bar{p} \wedge \bar{q}) \quad \wedge \quad (p' \wedge q') \quad \wedge$

$(((x \leftrightarrow p) \wedge (y \leftrightarrow q) \wedge (x' \leftrightarrow p') \wedge (y' \leftrightarrow q')) \rightarrow$

$(((x' \leftrightarrow (\bar{x} \wedge \bar{y})) \wedge (y' \leftrightarrow 0)) \vee ((x' \leftrightarrow (x \wedge \bar{y})) \wedge (y' \leftrightarrow \bar{y})))$

$\exists p, q, p', q' \forall x, y, x', y' \exists a, b, c, d, e, f, g, h, i, j, k.$

...

$(\bar{x}' \vee \bar{g} \vee e \wedge (\bar{x}' \vee \bar{e} \vee g) \wedge (\bar{g} \vee \bar{e} \vee x') \wedge (g \vee e \vee x') \quad \wedge$

$(\bar{f} \vee \bar{h} \vee x') \wedge (\bar{f} \vee \bar{x}' \vee h) \wedge (\bar{h} \vee \bar{x}' \vee f) \wedge (h \vee x' \vee f) \quad \wedge$

$(\bar{y}' \vee \bar{i} \vee \bar{y}) \wedge (\bar{y}' \vee \bar{y} \vee i) \wedge (\bar{i} \vee y \vee y') \wedge (i \vee \bar{y} \vee y') \quad \wedge$

...

JKU

Transformation to PCNF: Example

$\exists p, q, p', q' \forall x, y, x', y'.$

$(\bar{p} \wedge \bar{q}) \quad \wedge \quad (p' \wedge q') \quad \wedge$

$(((x \leftrightarrow p) \wedge (y \leftrightarrow q) \wedge (x' \leftrightarrow p') \wedge (y' \leftrightarrow q')) \rightarrow$

$((x' \leftrightarrow (\bar{x} \wedge \bar{y})) \wedge (y' \leftrightarrow 0)) \vee ((x' \leftrightarrow (x \wedge \bar{y})) \wedge (y' \leftrightarrow \bar{y})))$

$\exists p, q, p', q' \forall x, y, x', y' \exists a, b, c, d, e, f, g, h, i, j, k.$

...

$(\bar{j} \vee \bar{g}) \wedge (\bar{j} \vee y') \wedge (j \vee g \vee \bar{y}') \quad \wedge$

$(\bar{k} \vee h) \wedge (\bar{k} \vee i) \wedge (k \vee \bar{i} \vee \bar{h}) \quad \wedge$

...

JKU

Transformation to PCNF: Example

$\exists p, q, p', q' \forall x, y, x', y'.$

$(\bar{p} \wedge \bar{q}) \quad \wedge \quad (p' \wedge q') \quad \wedge$

$(((x \leftrightarrow p) \wedge (y \leftrightarrow q) \wedge (x' \leftrightarrow p') \wedge (y' \leftrightarrow q')) \rightarrow$

$((x' \leftrightarrow (\bar{x} \wedge \bar{y})) \wedge (y' \leftrightarrow 0)) \vee ((x' \leftrightarrow (x \wedge \bar{y})) \wedge (y' \leftrightarrow \bar{y})))$

$\exists p, q, p', q' \forall x, y, x', y' \exists a, b, c, d, e, f, g, h, i, j, k.$

...

$(\bar{p} \wedge \bar{q}) \wedge (p' \wedge q') \quad \wedge$

$(\bar{a} \vee \bar{b} \vee \bar{c} \vee \bar{d} \vee j \vee k)$

QBF Solving

- Most QBF solvers support only QDIMACS format
  - like DIMACS but with quantifier prefix
- QBF solvers return true/false
  - assignments for variables in outermost quantifier block
  - functions for solving strategies
- Community platform: www.qbflib.org
- Examples for state-of-the-art solvers:
  - DepQBF
  - RAReQS

Bounded Model Checking (Fairness)

- Given a Kripke structure K. Is there a path such that a property $\neg p$ holds forever?
- In other words: there is a path such that Fp does not hold in K
- Observation 1: if Fp does not hold in K, there is an **infinite counter-example**.
- Observation 2: if the counter-example is infinite, then it has to be because of a cycle.

Bounded Model Checking (Fairness)

A bounded model checking (BMC) problem for Kripke structure K and fairness property Fp is encoded by

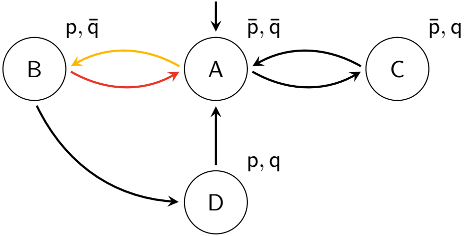
$$I(s_0) \wedge \bigwedge_{l=0}^{k-1} \mathcal{T}(s_l, s_{l+1}) \wedge \bigvee_{i=0}^{k} \mathcal{T}(s_k, s_i) \wedge \bigwedge_{j=0}^{k} F(s_j)$$

where

- $I(s_0)$ is true $\Leftrightarrow$ s_0 is an initial state
- $\mathcal{T}$ is the transition function of K
- $F(s_k)$ is true $\Leftrightarrow$ $\neg p$ holds in s_k

BMC Fairness

We want to know if $\mathsf{F}q$ holds for Kripke structure K:



Initial State:

$(\bar{p} \wedge \bar{q}) \quad \wedge$

One Step:

$(((p' \leftrightarrow (\bar{p} \wedge \bar{q})) \wedge (q' \leftrightarrow 0)) \vee ((p' \leftrightarrow (p \wedge \bar{q})) \wedge (q' \leftrightarrow \bar{q}))) \quad \wedge$

Cycle Check:

$((((p \leftrightarrow (\bar{p}' \wedge \bar{q}')) \wedge (q \leftrightarrow 0)) \vee ((p \leftrightarrow (p' \wedge \bar{q}')) \wedge (q \leftrightarrow \bar{q}'))) \quad \vee$

$(((p' \leftrightarrow (\bar{p}' \wedge \bar{q}')) \wedge (q' \leftrightarrow 0)) \vee ((p' \leftrightarrow (p' \wedge \bar{q}')) \wedge (q' \leftrightarrow \bar{q}')))) \quad \wedge$

Property Check:

$\bar{q} \wedge \bar{q}'$

JKU

BMC Summary

- BMC is incomplete ...
  - if all checked formulas are unsat, no insight
  - how to choose k? when to stop increasing k?

- ... very efficient (e.g., debugging)

- many tuning techniques
  - exploit similarities between two transition steps (structure sharing
  - simplification of formula by rewritings)

How to choose k for Safety?

Given Kripke structure K, the **diameter** is the smallest number d such that for every path $s_0, \ldots, s_{d+1}$ there exists a path $t_0, \ldots, t_l$ such that $l \leq d$ and $t_0 = s_0$ and $t_l = s_{d+1}$.

- If a state s is reachable from state t, then there is a path of length d of less where d is the diameter.
- The diameter is the maximum length between two states.
- Computing the diameter is difficult (solve a QBF).

How to choose k for Fairness?

Given Kripke structure K, the **recurrence diameter** is the smallest number d such that for every path $s_0, \ldots, s_{d+1}$, there exists $j \leq d$ with $s_{d+1} = s_j$.

- The recurrence diameter is the maximum length of a non-looping path.
- Can be formulated as validity checking problem.