Reducing Acceptance Marks in Emerson-Lei Automata by QBF Solving

Tereza Schwarzová Jan Strejček Juraj Major

Masaryk University, Brno, Czech Republic

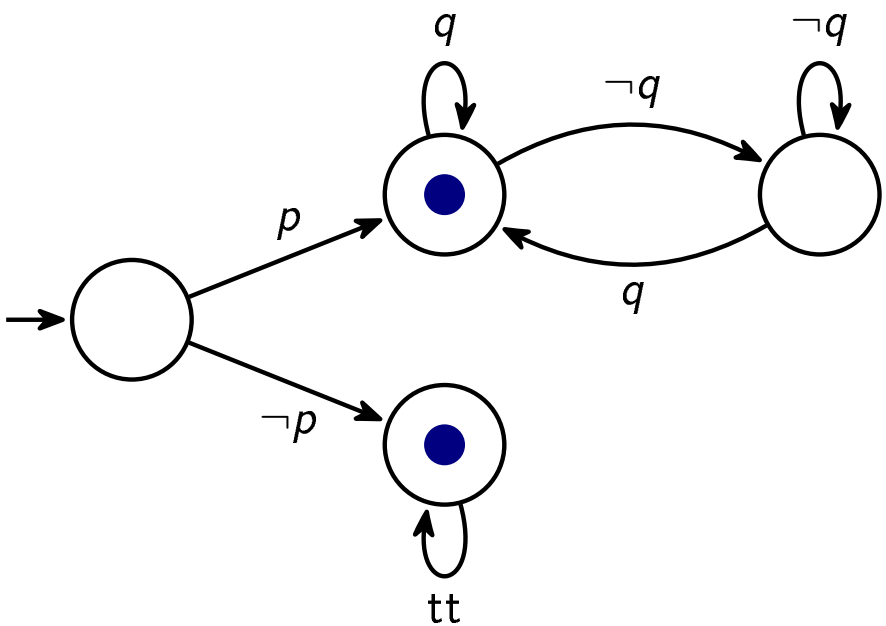
SAT23
July 28, 2023

Presentation Overview

Büchi automata

State-based acceptance

$p \implies \mathrm{GF}q$

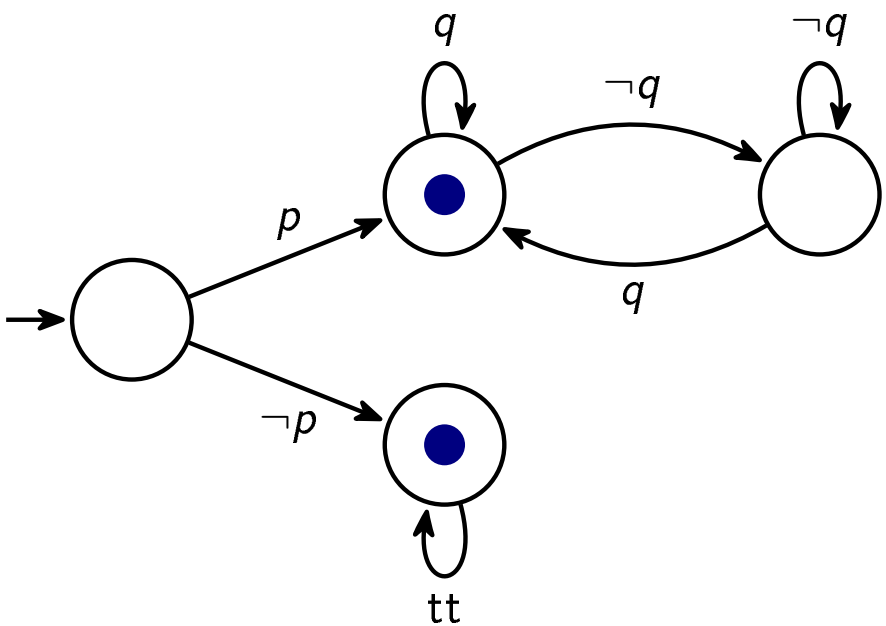


state-based

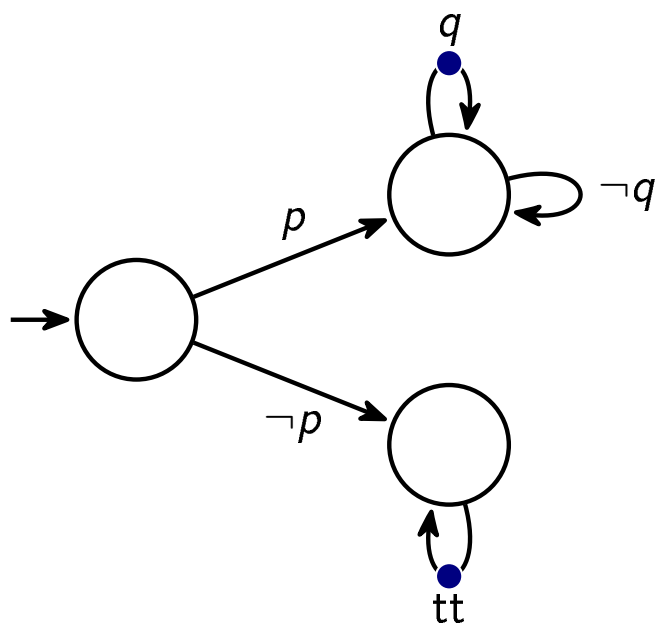
Büchi automata

State-based acceptance and transition-based acceptance

$p \implies \mathsf{GF}q$

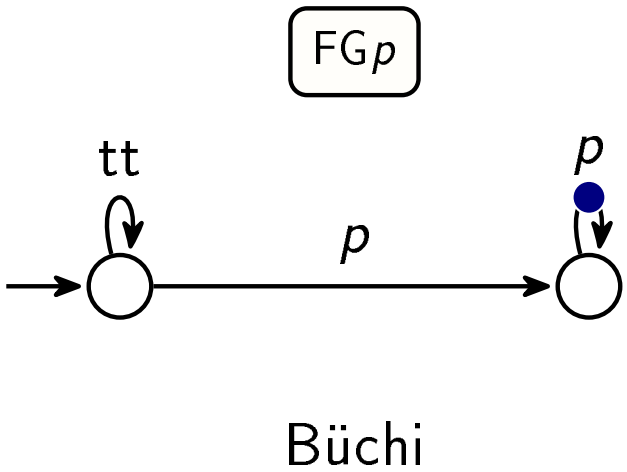


state-based



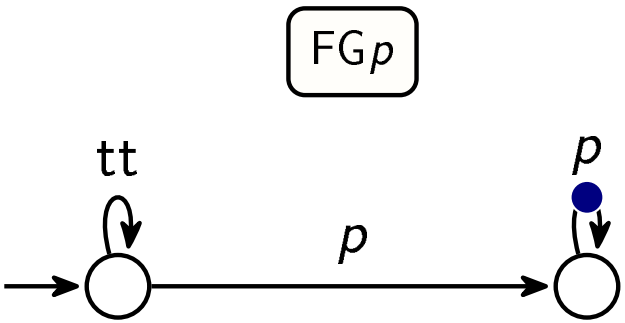
transition-based

Beyond Büchi Automata

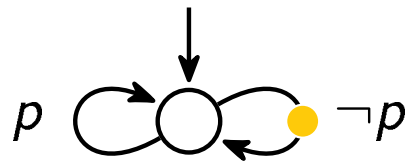


Büchi

Beyond Büchi Automata

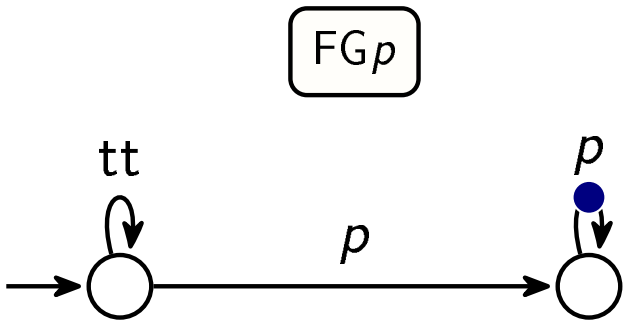


Büchi

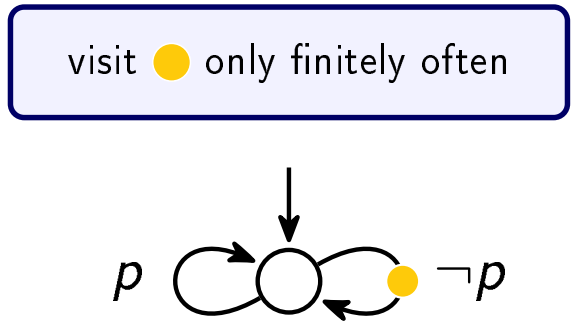


co-Büchi

Beyond Büchi Automata

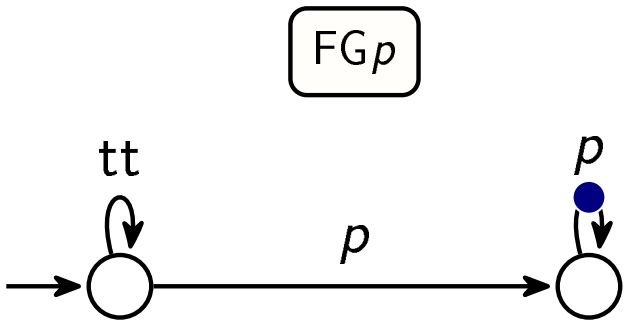


Büchi

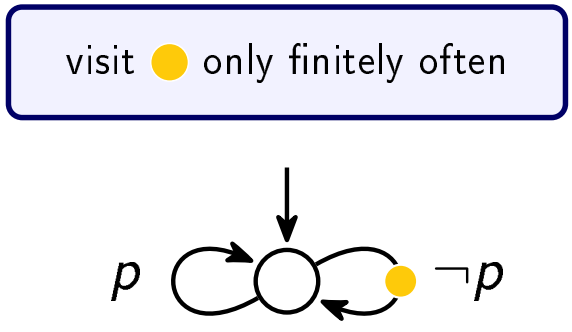


co-Büchi

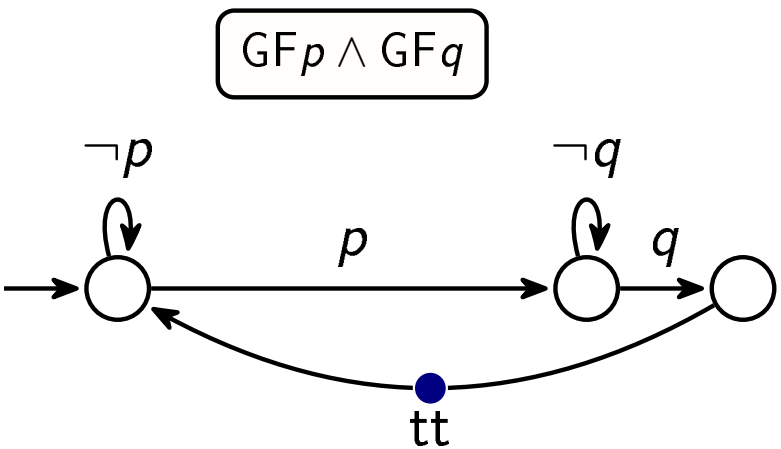
Beyond Büchi Automata



Büchi

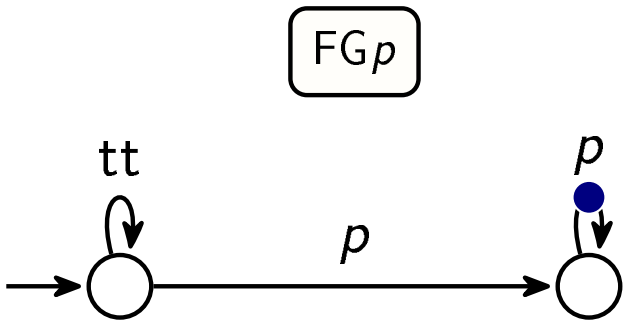


co-Büchi

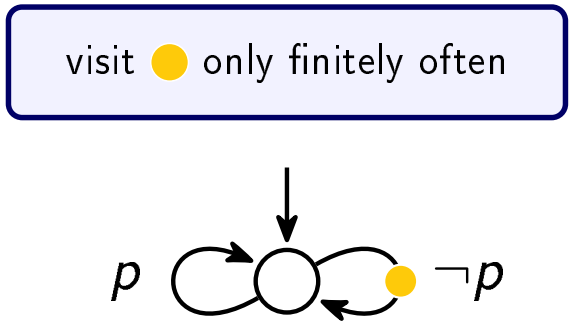


Büchi

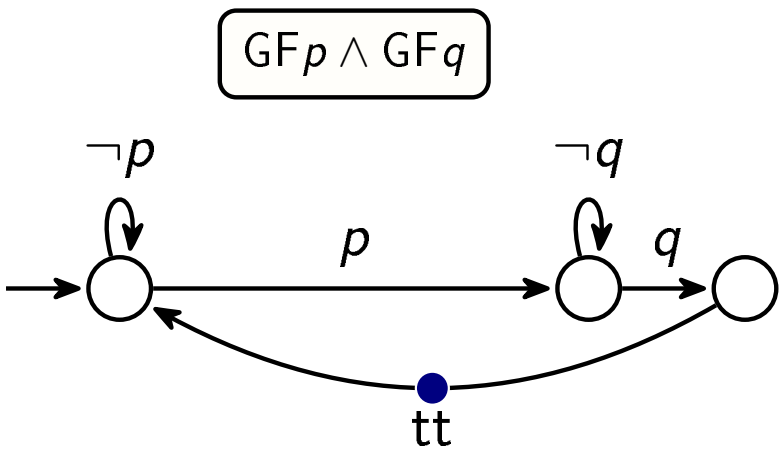
Beyond Büchi Automata



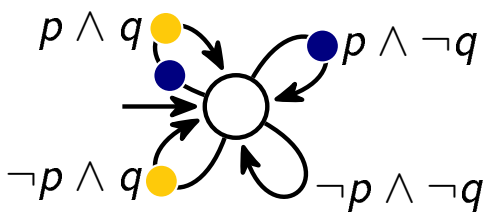
Büchi



co-Büchi

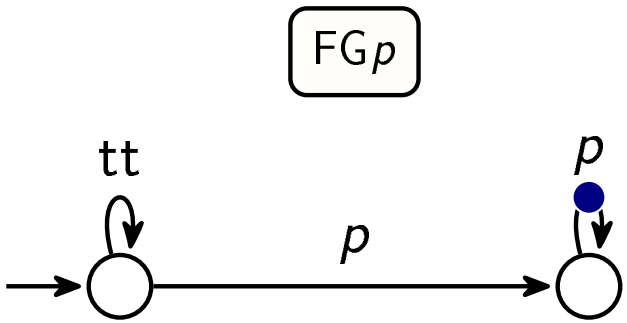


Büchi

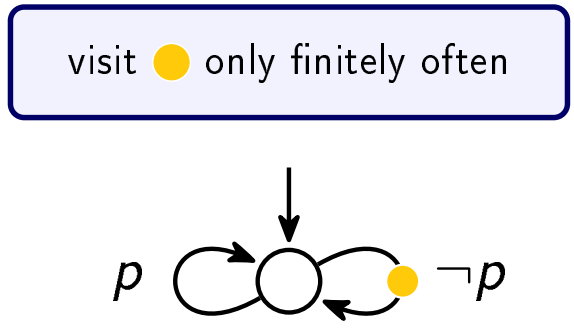


generalized Büchi

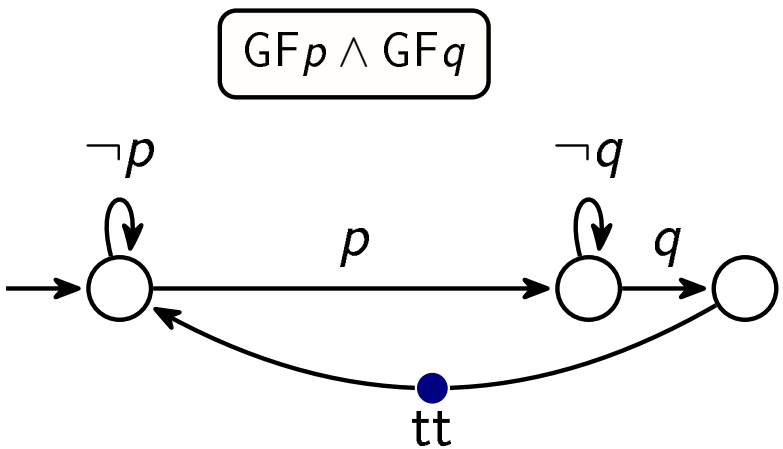
Beyond Büchi Automata



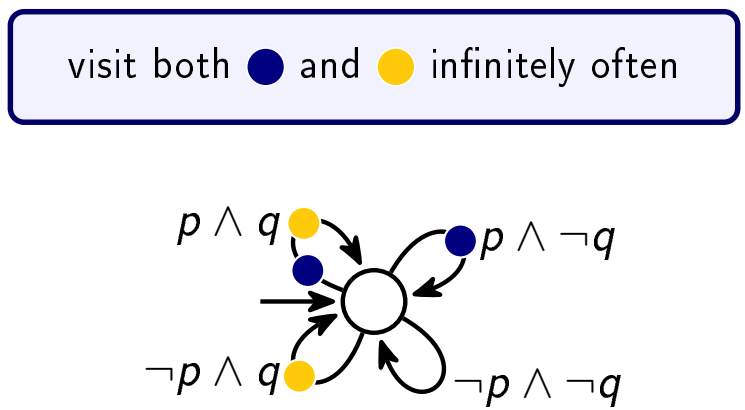
Büchi



co-Büchi

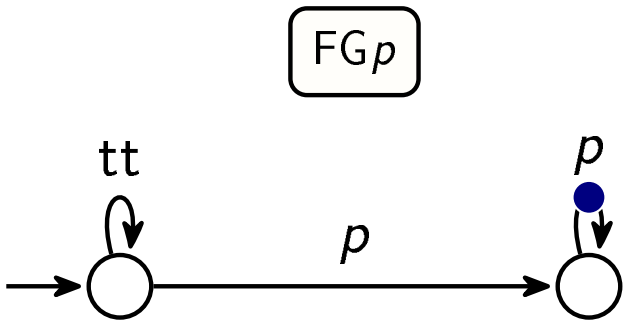


Büchi

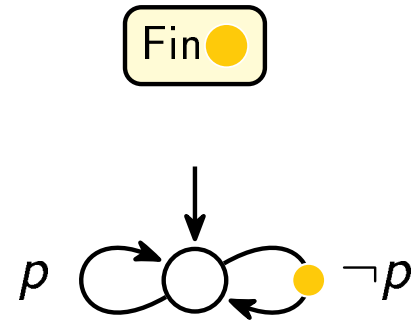


generalized Büchi

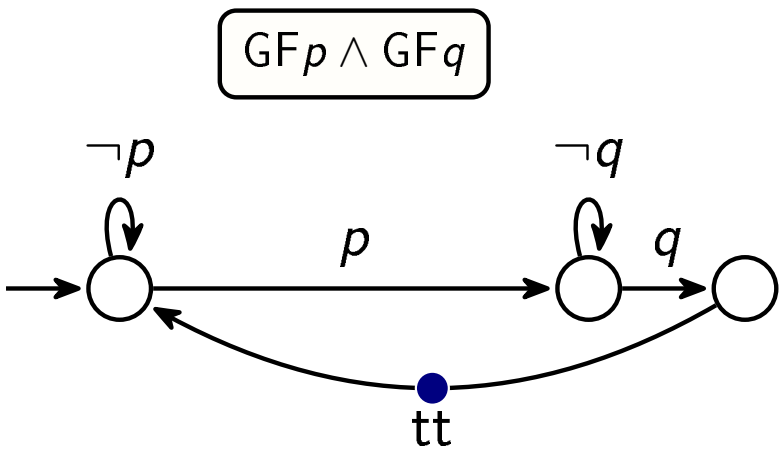
Beyond Büchi Automata



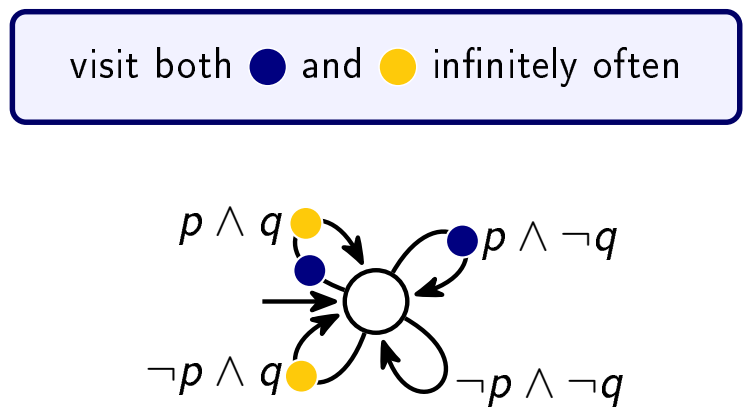
Büchi



TELA

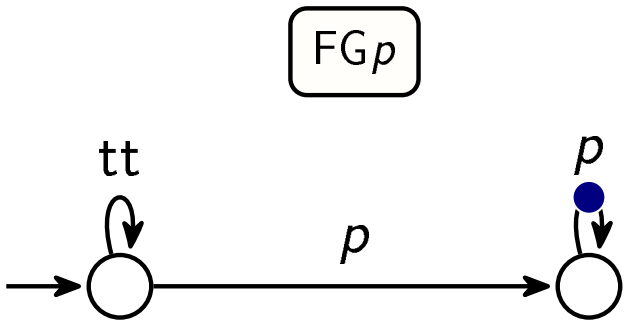


Büchi

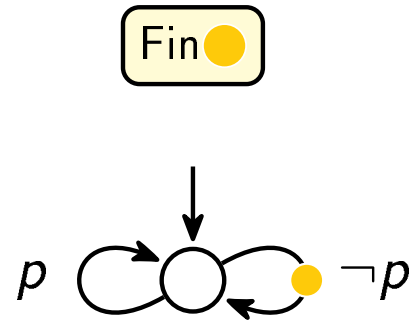


generalized Büchi

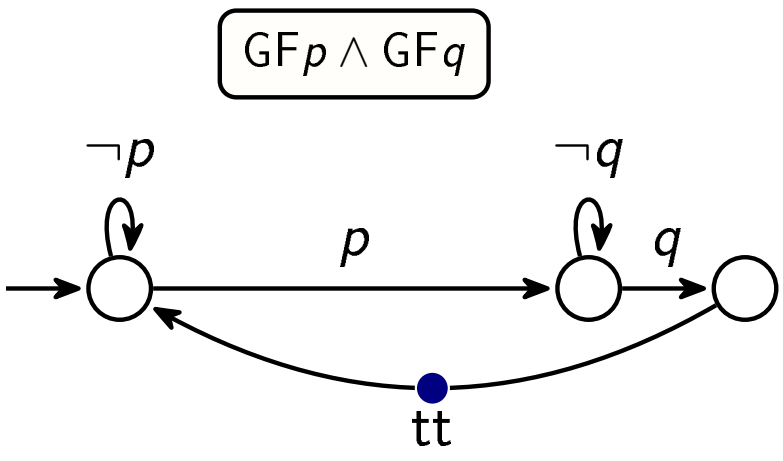
Beyond Büchi Automata



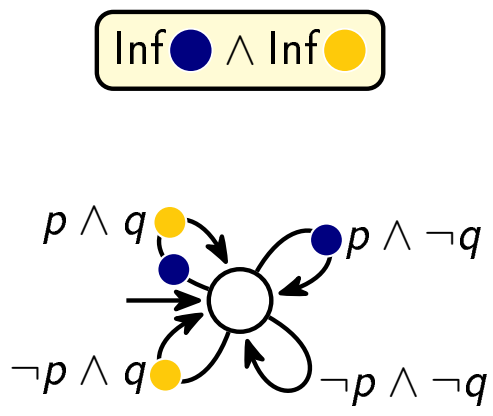
Büchi



TELA



Büchi



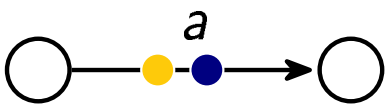
TELA

Transition-based Emerson-Lei Automata (TELA)

- Emerson and Lei, 1980s
- Reinvented in Hanoi Omega-Automata Format (2015)
- Same expressive power as nondeterministic Büchi with fewer states
- Tools: `delag`, `rabinizer4`, `spot` and `ltl3tela`
- Usage: emptiness check, translation of LTL to parity automata

Transition-based Emerson-Lei Automata

$\mathcal{A} = (Q, M, \Sigma, \delta, q_I, \varphi)$

- Q is a finite set of *states*
- M is a finite set of *acceptance marks*
- Σ is a finite *alphabet*
- $\delta \subseteq Q \times \Sigma \times 2^M \times Q$ is a *transition relation* 

- $q_I \in Q$ is an *initial state*
- φ is a *acceptance condition*, where

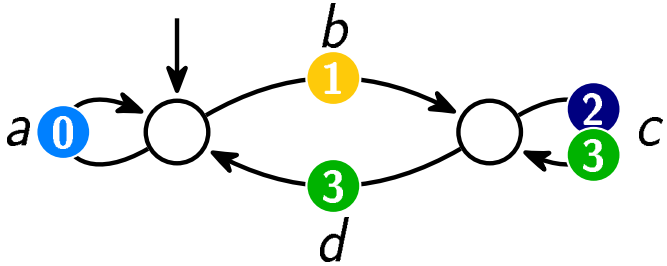
$$\varphi ::= \mathsf{tt} \mid \mathsf{ff} \mid \mathsf{Inf}(\bullet) \mid \mathsf{Fin}(\bullet) \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$$

Acceptance condition shape

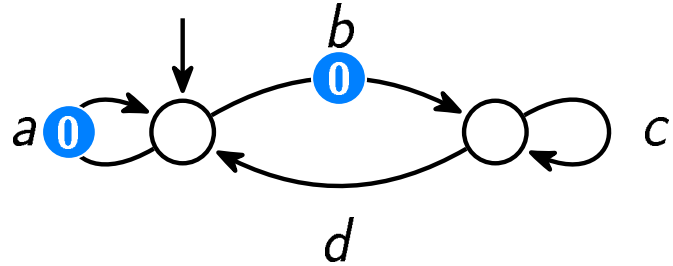
$$\varphi ::= \mathsf{tt} \mid \mathsf{ff} \mid \mathsf{Inf}(\bullet) \mid \mathsf{Fin}(\bullet) \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$$

Büchi	$\mathsf{Inf}(\bullet)$
generalized Büchi	$\mathsf{Inf}(1) \wedge \mathsf{Inf}(2) \wedge \mathsf{Inf}(3) \wedge \ldots$
co-Büchi	$\mathsf{Fin}(\bullet)$
Rabin	$(\mathsf{Fin}(0) \wedge \mathsf{Inf}(0)) \vee (\mathsf{Fin}(5) \wedge \mathsf{Inf}(5)) \vee \ldots$
Streett	$(\mathsf{Inf}(0) \vee \mathsf{Fin}(0)) \wedge (\mathsf{Inf}(5) \vee \mathsf{Fin}(5)) \vee \ldots$
$\vdots$	

Acceptance condition & Motivation

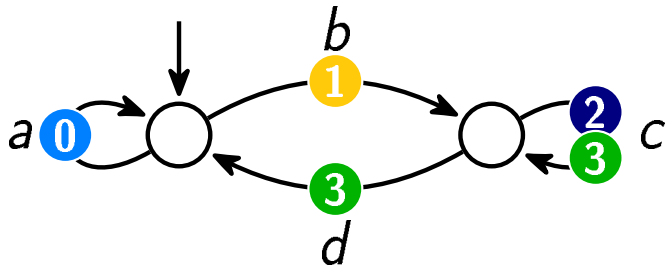


$(\mathsf{Fin}(1) \wedge \mathsf{Inf}(2)) \vee (\mathsf{Inf}(0) \wedge \mathsf{Fin}(1) \wedge \mathsf{Inf}(3))$

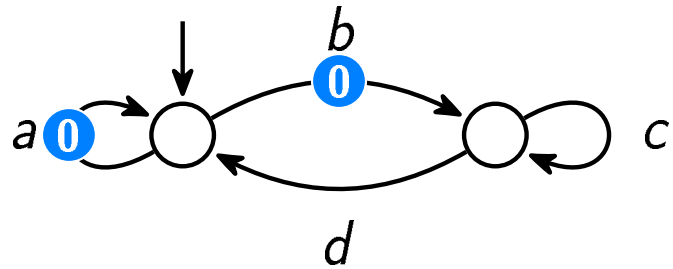


$\mathsf{Fin}(0)$

Acceptance condition & Motivation



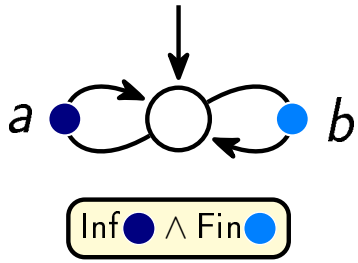
$(\mathrm{Fin}(1) \wedge \mathrm{Inf}(2)) \vee (\mathrm{Inf}(0) \wedge \mathrm{Fin}(1) \wedge \mathrm{Inf}(3))$



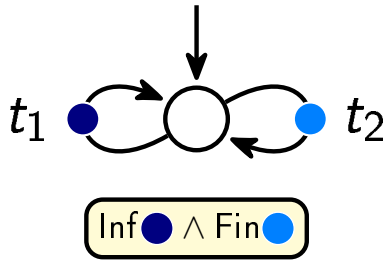
$\mathrm{Fin}(0)$

- *Color appearance record*:
  - TELA $\rightsquigarrow$ parity
  - Parity automaton with up to $m! \cdot s$ states
- *Emptiness-check algorithm*:
  - Exponential in the number of Fin marks

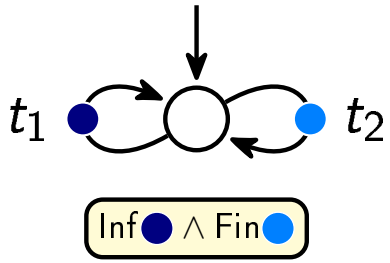
Automaton run



Automaton run



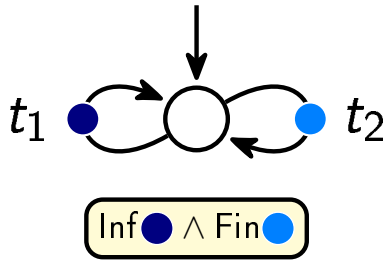
Automaton run



Run $\pi \in \delta^\omega$

$\pi_1 \in \{t_1, t_2\}^*.\ \{t_1\}^\omega$
$\pi_2 \in \{t_1, t_2\}^*.\ \{t_2\}^\omega$
$\pi_3 \in \{t_1, t_2\}^\omega$
$s.t.\ \pi_3 \models \mathsf{GF}t_1 \wedge \mathsf{GF}t_2$

Automaton run



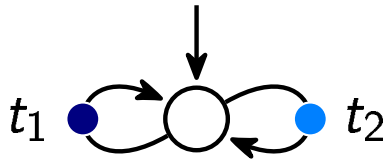
Run $\pi \in \delta^\omega$

π is evaluated over acceptance condition

Inf● ∧ Fin●

$\pi_1 \in \{t_1, t_2\}^*.\ \{t_1\}^\omega$
$\pi_2 \in \{t_1, t_2\}^*.\ \{t_2\}^\omega$
$\pi_3 \in \{t_1, t_2\}^\omega$
$s.t.\ \pi_3 \models \mathsf{GF}t_1 \wedge \mathsf{GF}t_2$

Automaton run



Inf● ∧ Fin●

Run $\pi \in \delta^\omega$

π is evaluated over acceptance condition

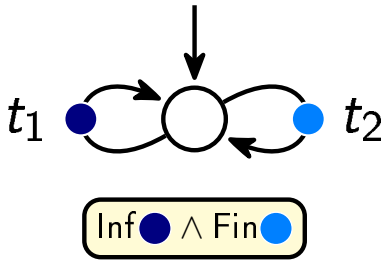
Inf● ∧ Fin●

Accepting

Rejecting

$\pi_1 \in \{t_1, t_2\}^*.\ \{t_1\}^\omega$
$\pi_2 \in \{t_1, t_2\}^*.\ \{t_2\}^\omega$
$\pi_3 \in \{t_1, t_2\}^\omega$
$s.t.\ \pi_3 \models \mathsf{GF}t_1 \wedge \mathsf{GF}t_2$

Automaton run



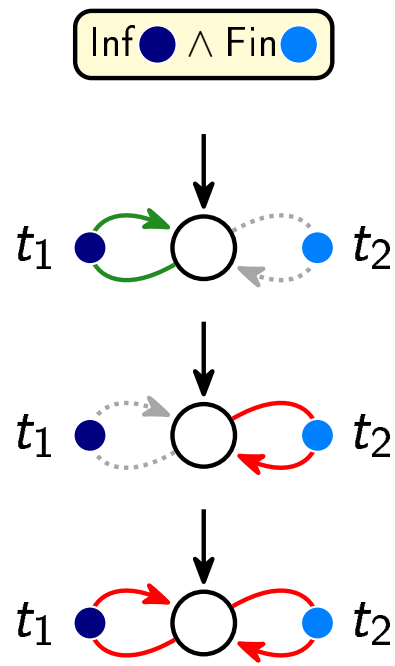
Run $\pi \in \delta^\omega$

π is evaluated over acceptance condition

Accepting

Rejecting

$\pi_1 \in \{t_1, t_2\}^*.\ \{t_1\}^\omega$
$\pi_2 \in \{t_1, t_2\}^*.\ \{t_2\}^\omega$
$\pi_3 \in \{t_1, t_2\}^\omega$
$s.t.\ \pi_3 \models \mathsf{GF}t_1 \wedge \mathsf{GF}t_2$



Problem specification

Problem

Find **acceptance condition with fewer marks** and **relabeling** of automaton that preserves acceptance of all runs.

Problem specification

Problem

Find **acceptance condition with fewer marks** and **relabeling** of automaton that preserves acceptance of all runs.

Solution:

1. Encode problem into QBF
2. Let QBF solver solve the query and obtain model
3. Decode model
   - Create **new acceptance condition**
   - **Relabel** automaton

Query structure

$$\forall T \subseteq \delta \;.\; \mathit{cycle}(T) \implies \left(\mathit{satisfies}_{\mathit{old}}(T) \iff \mathit{satisfies}_{\mathit{new}}(T)\right)$$

Query structure

For all possible combinations of edges

$$\forall T \subseteq \delta \;.\; cycle(T) \implies \big(satisfies_{old}(T) \iff satisfies_{new}(T)\big)$$

Query structure

For all possible combinations of edges

$$\forall T \subseteq \delta \;.\; cycle(T) \implies \left(satisfies_{old}(T) \iff satisfies_{new}(T)\right)$$

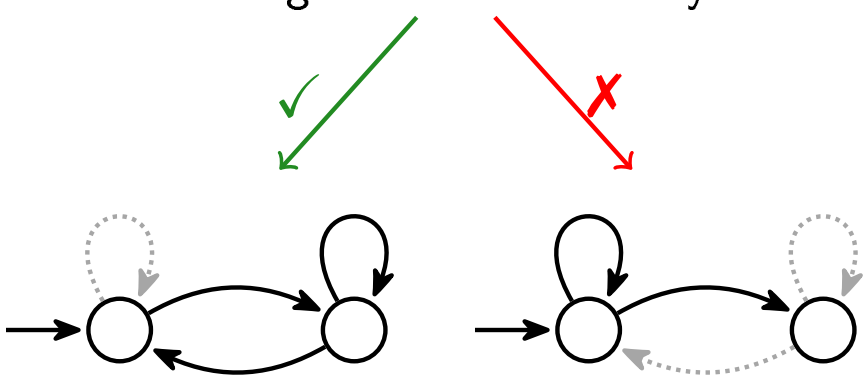
Do the edges in T create a cycle?

Query structure

For all possible combinations of edges

$$\forall T \subseteq \delta \;.\; cycle(T) \implies \left(satisfies_{old}(T) \iff satisfies_{new}(T)\right)$$

Do the edges in T create a cycle?



Query structure

For all possible combinations of edges

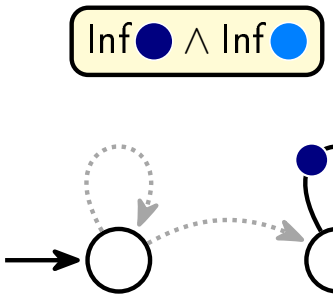
$$\forall T \subseteq \delta \;.\; cycle(T) \implies \left(satisfies_{old}(T) \iff satisfies_{new}(T)\right)$$

Do the edges in T create a cycle?

Is T accepting in the original automaton?

Query structure

For all possible combinations of edges



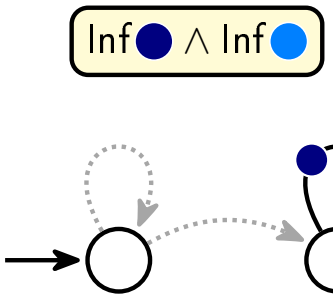
$$\forall T \subseteq \delta \;.\; cycle(T) \implies \left(satisfies_{old}(T) \iff satisfies_{new}(T)\right)$$

Do the edges in T create a cycle?

Is T accepting in the original automaton?

Query structure

For all possible combinations of edges



$$\forall T \subseteq \delta \;.\; cycle(T) \implies \left(satisfies_{old}(T) \iff satisfies_{new}(T)\right)$$

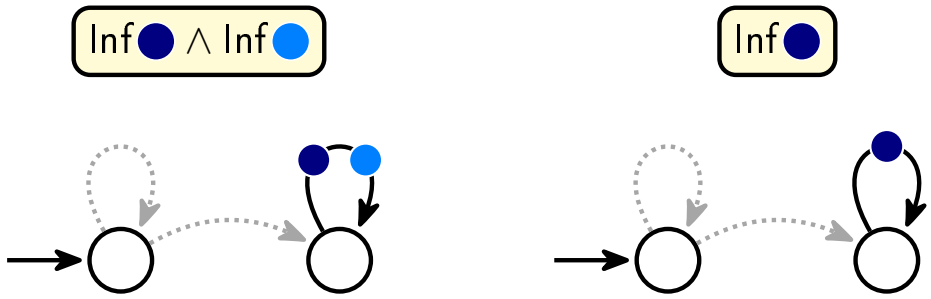
Do the edges in T create a cycle?

Is T accepting in the original automaton?

Is T accepting with fewer marks?

Query structure

For all possible combinations of edges



$$\forall T \subseteq \delta \;.\; cycle(T) \implies \big(satisfies_{old}(T) \iff satisfies_{new}(T)\big)$$

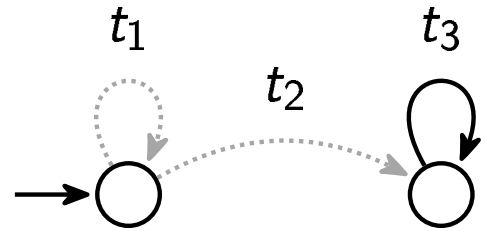
Do the edges in T create a cycle?

Is T accepting in the original automaton?

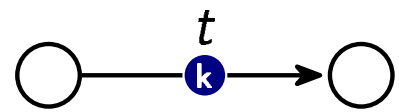
Is T accepting with fewer marks?

Variables

$$\mathsf{e}_t = \begin{cases} 1 & \text{if } t \in T \\ 0 & \text{otherwise} \end{cases}$$



$$\mathsf{n}_{t,k} = \begin{cases} 1 & \text{if relabeled } t \text{ contains } k \\ 0 & \text{otherwise} \end{cases}$$



After relabeling

$$\mathsf{i}_{c,k} = \begin{cases} 1 & \text{if the } c^{\text{th}} \text{ cube of } \psi \text{ contains Inf(k)} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathsf{f}_{c,k} = \begin{cases} 1 & \text{if the } c^{\text{th}} \text{ cube of } \psi \text{ contains Fin(k)} \\ 0 & \text{otherwise} \end{cases}$$

$$\psi \equiv \overbrace{\ldots \vee (\text{Inf(k)} \wedge \text{Fin(k)}) \vee \ldots}^{c^{\text{th}}\, cube}$$

New acc condition

Zoom in

$$\forall T \subseteq \delta \;.\; cycle(T) \implies \left(satisfies_{old}(T) \iff satisfies_{new}(T)\right)$$

$$\forall \vec{e} \;.\; cycle(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \vec{n}, \vec{i}, \vec{f})\right)$$

$$\vec{e} = (e_1, \ldots, e_t)$$
$$\vec{n} = (n_{1,1}, \ldots, n_{t,k})$$
$$\vec{i} = (i_{1,1}, \ldots, i_{c,k})$$
$$\vec{f} = (f_{1,1}, \ldots, f_{c,k})$$

Zoom in

$$\forall T \subseteq \delta \;.\; cycle(T) \implies \left(satisfies_{old}(T) \iff satisfies_{new}(T)\right)$$

$$\boxed{\forall \vec{e}} \;.\; \mathrm{cycle}(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \vec{n}, \vec{i}, \vec{f})\right)$$

$$\vec{e} = (e_1, \ldots, e_t)$$
$$\vec{n} = (n_{1,1}, \ldots, n_{t,k})$$
$$\vec{i} = (i_{1,1}, \ldots, i_{c,k})$$
$$\vec{f} = (f_{1,1}, \ldots, f_{c,k})$$

Zoom in

$$\forall T \subseteq \delta \;.\; cycle(T) \implies \left(satisfies_{old}(T) \iff satisfies_{new}(T)\right)$$

$$\underbrace{\boxed{\forall \vec{e}}}_{\forall e_1 \forall e_2 \ldots \forall e_t} \;.\; \text{cycle}(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \vec{n}, \vec{i}, \vec{f})\right)$$

$$\begin{aligned}
\vec{e} &= (e_1, \ldots, e_t) \\
\vec{n} &= (n_{1,1}, \ldots, n_{t,k}) \\
\vec{i} &= (i_{1,1}, \ldots, i_{c,k}) \\
\vec{f} &= (f_{1,1}, \ldots, f_{c,k})
\end{aligned}$$

Zoom in

$$\forall T \subseteq \delta \;.\; cycle(T) \implies \left(satisfies_{old}(T) \iff satisfies_{new}(T)\right)$$

$$\underbrace{\boxed{\forall \vec{e}}}_{\forall e_1 \forall e_2 \ldots \forall e_t} \;.\; \text{cycle}(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \underbrace{\vec{n}, \vec{i}, \vec{f}}_{\text{free variables}})\right)$$

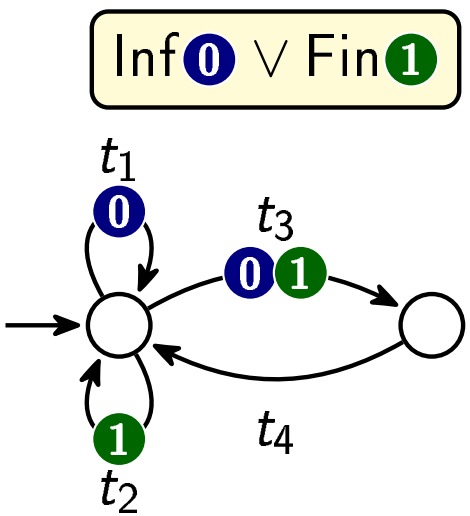
$$\begin{aligned}
\vec{e} &= (e_1, \ldots, e_t) \\
\vec{n} &= (n_{1,1}, \ldots, n_{t,k}) \\
\vec{i} &= (i_{1,1}, \ldots, i_{c,k}) \\
\vec{f} &= (f_{1,1}, \ldots, f_{c,k})
\end{aligned}$$

Original acceptance formula

$$\forall \vec{e} \;.\; cycle(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \vec{n}, \vec{i}, \vec{f})\right)$$

Original acceptance formula

$$\forall \vec{e} \;.\; cycle(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \vec{n}, \vec{i}, \vec{f})\right)$$



Original automaton

$$e_1 \vee e_3 \quad \vee \quad \neg e_2 \wedge \neg e_3$$

New acceptance condition

$$\forall \vec{e} \;.\; cycle(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \vec{n}, \vec{i}, \vec{f})\right)$$

New acceptance condition

$$\forall \vec{e} \;.\; cycle(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \underbrace{\vec{n}, \vec{i}, \vec{f}}_{\text{free variables}})\right)$$

Reminder

$n_{t,k}$ iff k is on edge t
$i_{c,k}$ iff Inf k in cube c
$f_{c,k}$ iff Fin k in cube c

New acceptance condition

$$\forall \vec{e} \;.\; cycle(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \underbrace{\vec{n}, \vec{i}, \vec{f}}_{\text{free variables}})\right)$$

$$\bigvee_{c=1}^{C} \bigwedge_{k=1}^{K}$$

$C = \#cubes, K = \#acc\ marks$

Reminder

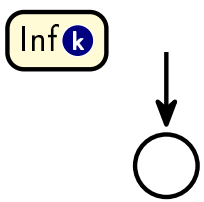
$n_{t,k}$ iff k is on edge t
$i_{c,k}$ iff Inf k in cube c
$f_{c,k}$ iff Fin k in cube c

New acceptance condition

$$\forall \vec{e} \;.\; cycle(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \underbrace{\vec{n}, \vec{i}, \vec{f}}_{\text{free variables}})\right)$$

$$\bigvee_{c=1}^{C} \bigwedge_{k=1}^{K} \left[i_{c,k} \implies \right]$$

$C = \#cubes, K = \#acc\ marks$



Reminder

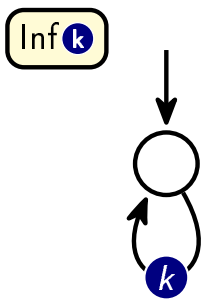
$n_{t,k}$ iff k is on edge t
$i_{c,k}$ iff Inf k in cube c
$f_{c,k}$ iff Fin k in cube c

New acceptance condition

$$\forall \vec{e} \;.\; cycle(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \underbrace{\vec{n}, \vec{i}, \vec{f}}_{\text{free variables}})\right)$$

$$\bigvee_{c=1}^{C} \bigwedge_{k=1}^{K} \left[i_{c,k} \implies \bigvee_{t \in \delta} (e_t \wedge n_{t,k}) \right]$$

$C = \#cubes, K = \#acc\ marks$



Reminder

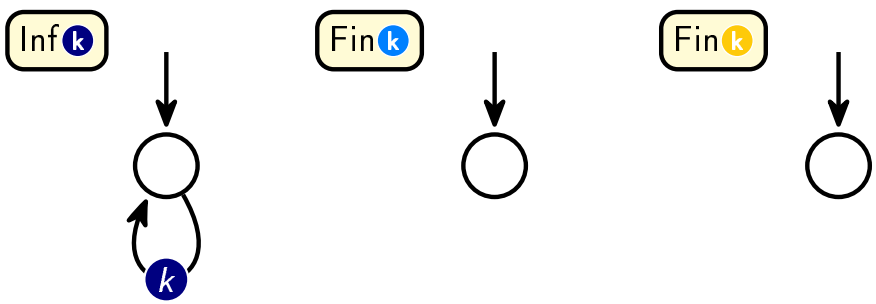
$n_{t,k}$ iff k is on edge t
$i_{c,k}$ iff Inf k in cube c
$f_{c,k}$ iff Fin k in cube c

New acceptance condition

$$\forall \vec{e} \;.\; cycle(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \underbrace{\vec{n}, \vec{i}, \vec{f}}_{\text{free variables}})\right)$$

$$\bigvee_{c=1}^{C} \bigwedge_{k=1}^{K} \left[i_{c,k} \implies \bigvee_{t \in \delta} (e_t \wedge n_{t,k}) \right] \wedge \left[f_{c,k} \implies \right]$$

$C = \#cubes, K = \#acc\ marks$



Reminder

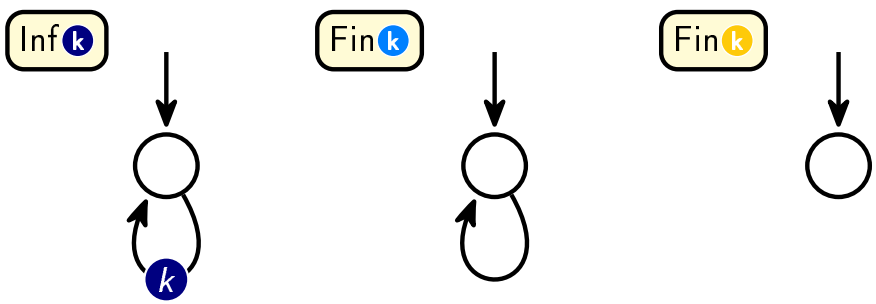
$n_{t,k}$ iff k is on edge t
$i_{c,k}$ iff Inf k in cube c
$f_{c,k}$ iff Fin k in cube c

New acceptance condition

$$\forall \vec{e} \;.\; cycle(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \underbrace{\vec{n}, \vec{i}, \vec{f}}_{\text{free variables}})\right)$$

$$\bigvee_{c=1}^{C} \bigwedge_{k=1}^{K} \left[i_{c,k} \implies \bigvee_{t \in \delta} (e_t \wedge n_{t,k})\right] \wedge \left[f_{c,k} \implies \bigwedge_{t \in \delta} (\neg n_{t,k} \vee \neg e_t)\right]$$

$C = \#cubes, K = \#acc\ marks$



Reminder

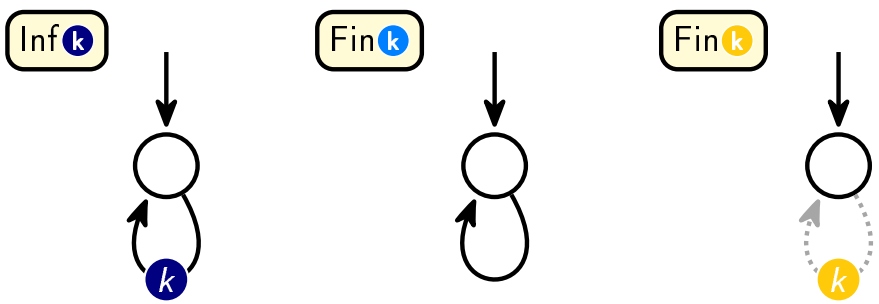
$n_{t,k}$ iff k is on edge t
$i_{c,k}$ iff Inf k in cube c
$f_{c,k}$ iff Fin k in cube c

New acceptance condition

$$\forall \vec{e} \;.\; cycle(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \underbrace{\vec{n}, \vec{i}, \vec{f}}_{\text{free variables}})\right)$$

$$\bigvee_{c=1}^{C} \bigwedge_{k=1}^{K} \left[i_{c,k} \implies \bigvee_{t \in \delta} (e_t \wedge n_{t,k})\right] \wedge \left[f_{c,k} \implies \bigwedge_{t \in \delta} (\neg n_{t,k} \vee \neg e_t)\right]$$

$C = \#cubes, K = \#acc\ marks$



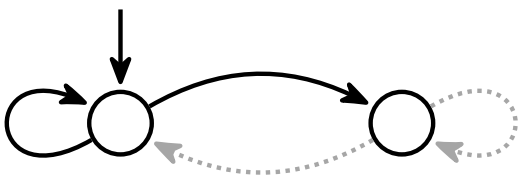
Reminder

$n_{t,k}$ iff k is on edge t
$i_{c,k}$ iff Inf k in cube c
$f_{c,k}$ iff Fin k in cube c

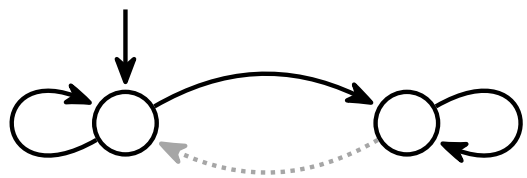
Cycle definition

$$\forall \vec{e} \;.\; cycle(\vec{e}) \implies \left(satisfies_{old}(\vec{e}) \iff satisfies_{new}(\vec{e}, \vec{n}, \vec{i}, \vec{f})\right)$$

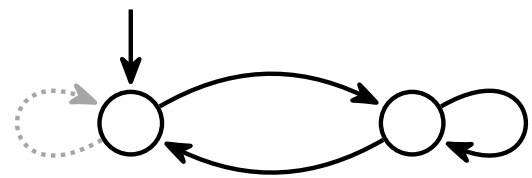
Three versions of cycles



Lightweighted ‘cycle‘



Continuous ‘cycle‘

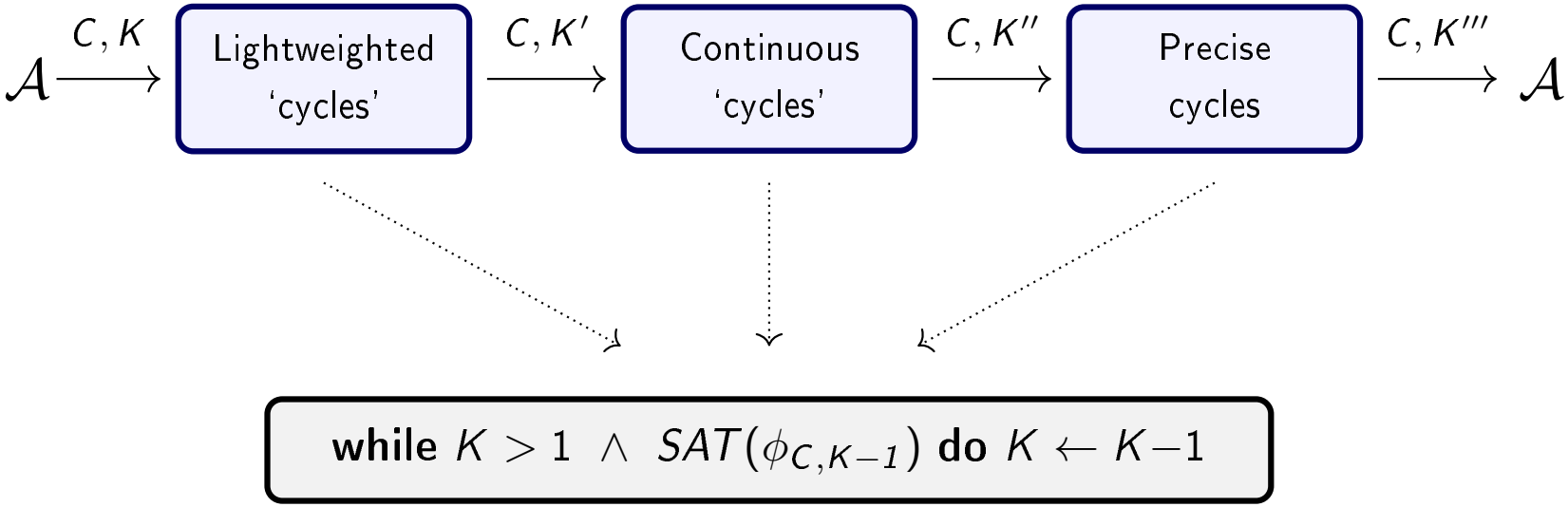


Precise cycle

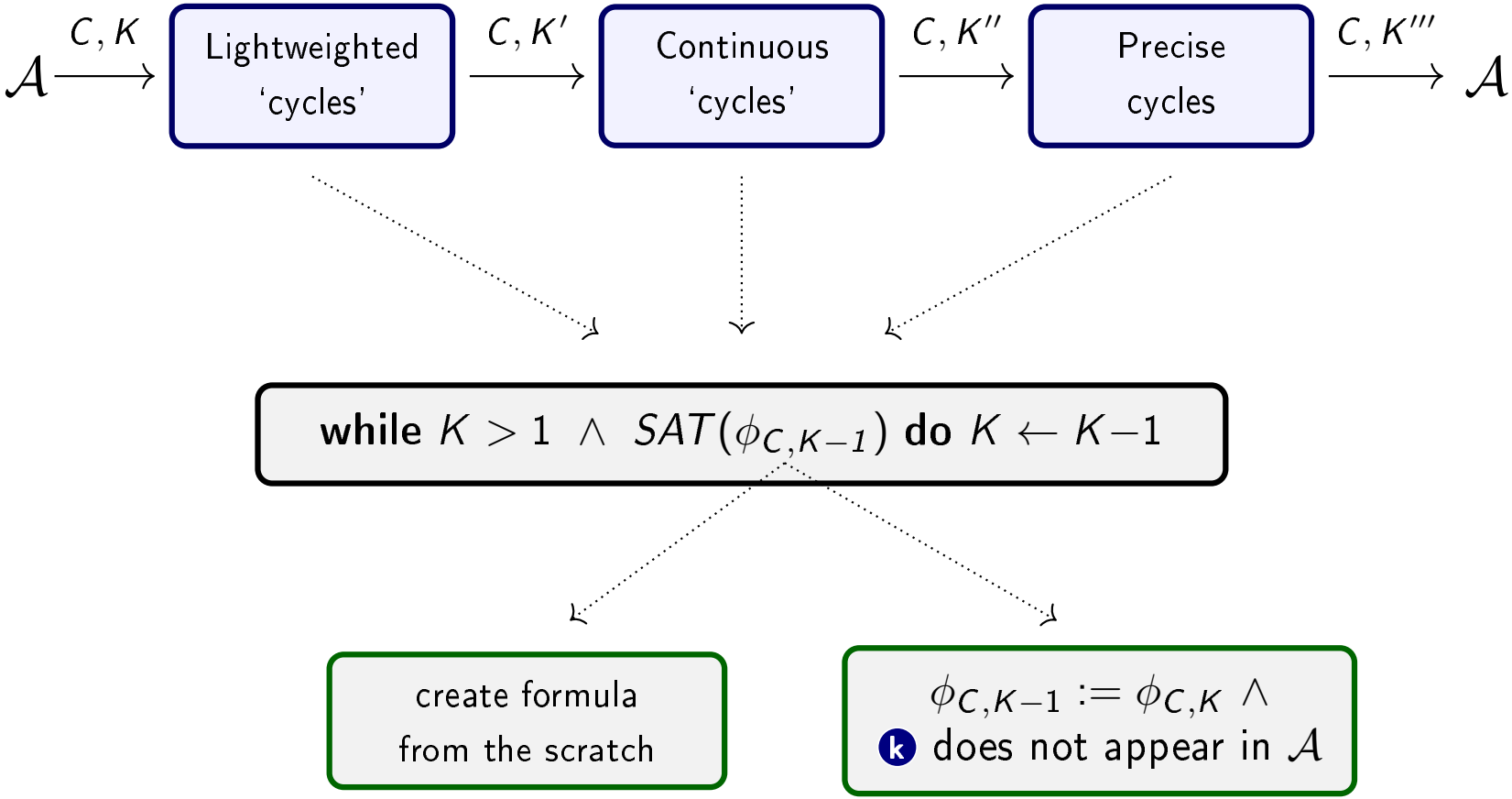
Reduction algorithm

while $K > 1 \ \wedge \ SAT(\phi_{C,K-1})$ **do** $K \leftarrow K-1$

Reduction algorithm



Reduction algorithm



TELAtko tool

- Python script
- usage of Spot library
- Z3 solver (prenex non-CNF, incremental)
- `https://gitlab.fi.muni.cz/xschwar3/telatko`

Experimental results

Contribution of each cycle-version during the reduction