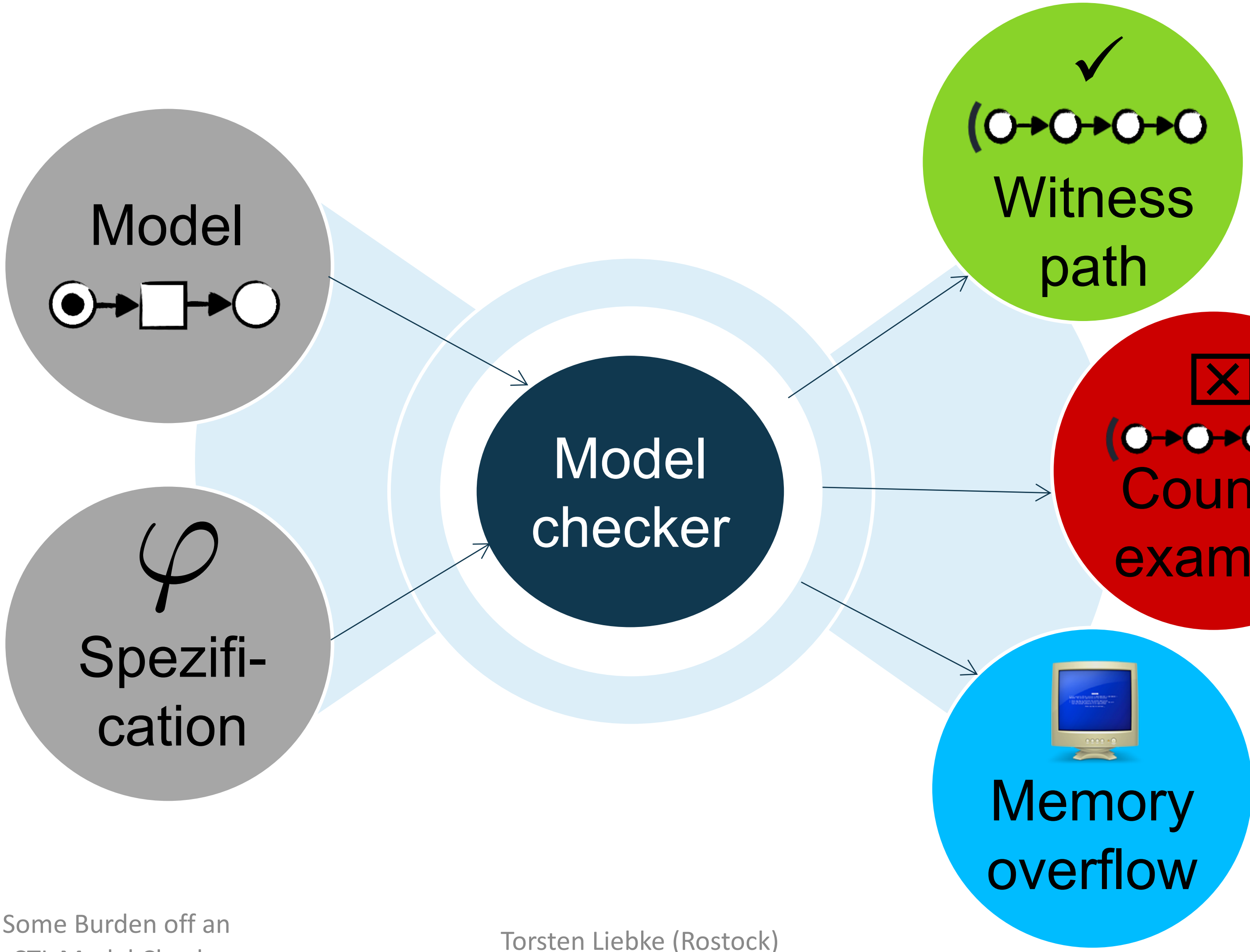
Taking Some Burden off an Explicit CTL Model Checker

Torsten Liebke and Karsten Wolf

University of Rostock

Model checking



Computational Tree logic (CTL)

$$
\begin{aligned}
\phi ::= {} & \mathtt{T} \mid \mathtt{F} \mid \mathtt{FIREABLE} \mid \mathtt{DEADLOCK} \mid \rho \\
& \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \longrightarrow \phi \\
& \mid \mathrm{AX}\ \phi \mid \mathrm{EX}\ \phi \\
& \mid \mathrm{AF}\ \phi \mid \mathrm{EF}\ \phi \\
& \mid \mathrm{AG}\ \phi \mid \mathrm{EG}\ \phi \\
& \mid \mathrm{A}\ (\phi\ \mathrm{U}\ \phi) \mid \mathrm{E}\ (\phi\ \mathrm{U}\ \phi)
\end{aligned}
$$

Path quantifier

A: inevitably (along all paths)

E: possibly (there exists a path)

Temporal operators

G: globally (always)

F: in future (eventually)

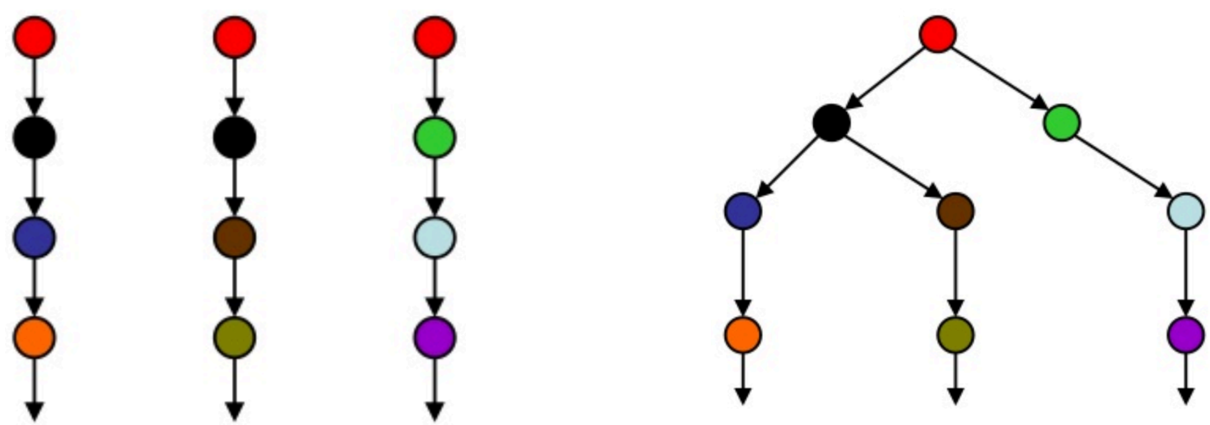
X: neXt state

U: until

Linear Time Logic (LTL)

$$\phi ::= T \mid F \mid \mathrm{FIREABLE} \mid \mathrm{DEADLOCK} \mid \rho$$
$$\mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \longrightarrow \phi$$
$$\mid X\ \phi \mid F\ \phi \mid G\ \phi \mid (\phi\ U\ \phi)$$

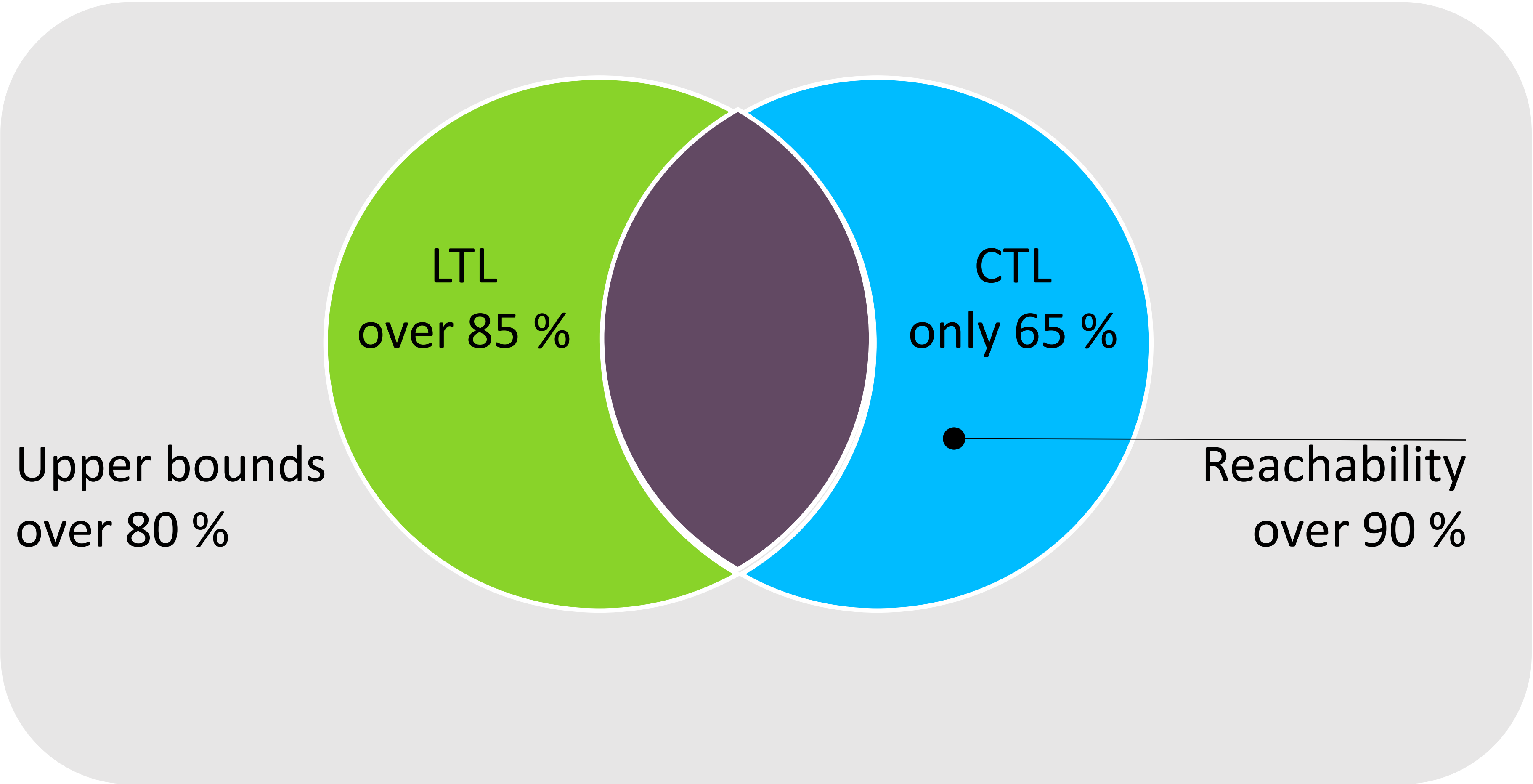
Similar to CTL but path quantifiers are not used.



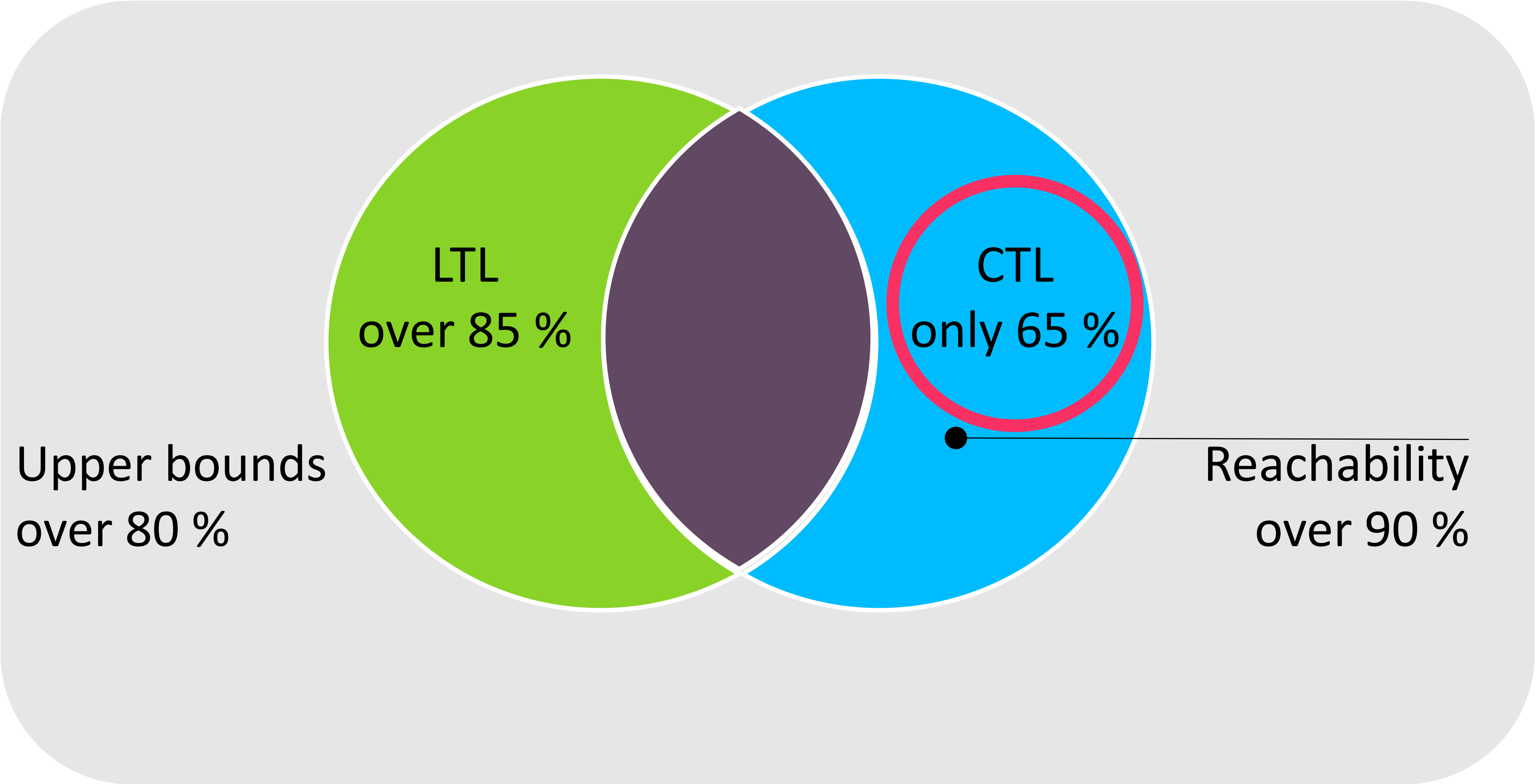
LTL: linear time

CTL: branching time

LoLA’s performance at the MCC



CTL performs worse than the rest: LoLA's performance at the MCC



Simple and frequently occurring formulas

Many CTL queries have a rather simple structure – only few temporal operators

- In the MCC this could be an artefact of the randomised formula generating mechanism
- Share same experience with LoLA users

Goal

Find for each formula an optimized algorithm

Systematic approach

- Most approaches we're using are well known
- Combining them in a systematic way, to push the limits further

Building a uniformly picture

$\Rightarrow$ Especially for stubborn sets

$\Rightarrow$ Pre-processing

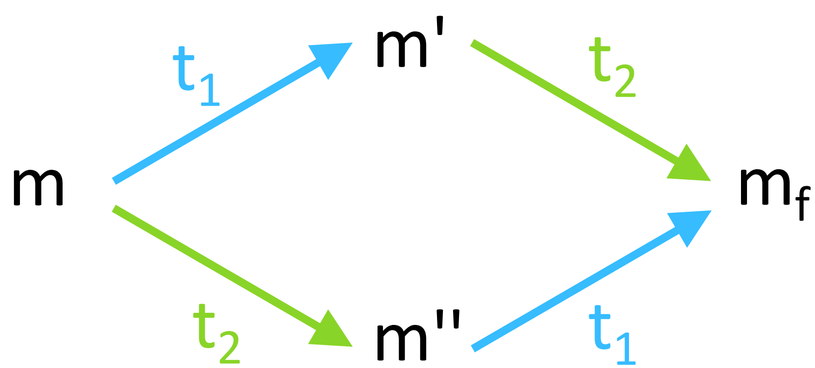
Partial order reduction: The stubborn set method

Given: Petri net $N = [P,T,F,W,m_0]$ and property ϕ

Goal: produce subgraph of the reachability graph

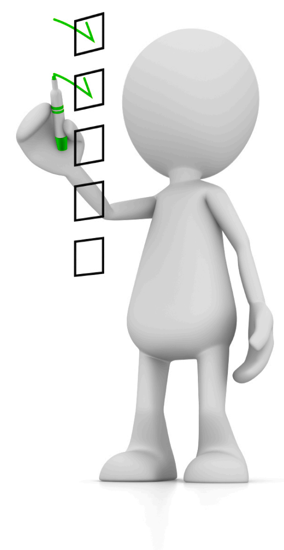
Condition: evaluation of ϕ yields same result

In any given marking, only a subset of the enabled transitions is explored = stubborn(m) $\subseteq T$



Principles

- There exists a list of principles to build the subgraph
- Based on the selected principles, all properties of a certain class are preserved



In the following:

$\boldsymbol{\pi}$‘ = Path in subgraph

$\boldsymbol{\pi}$ = Path in original graph

COM: The commutativity principle

- Transitions may be executed in another order
- Can shift transitions to the front of the path

$$m \overset{w}{\rightsquigarrow} m' \xrightarrow{t} m_f$$

$$m \xrightarrow{t} m'' \overset{w}{\rightsquigarrow} m_f$$

KEY: The key transition principle

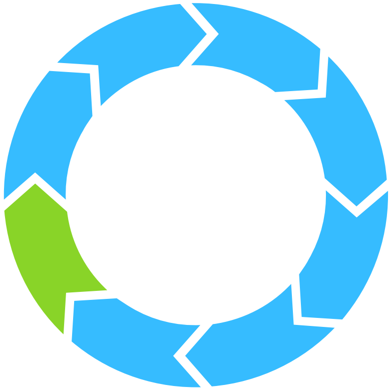
- Transition that stays enabled
- Can push transition to the right

VIS: The visibility principle

- VIS ensures the order of transitions visible for ϕ does not change
- Visible transitions in $\boldsymbol{\pi}$' appear in the same order as in $\boldsymbol{\pi}$, if they appear in $\boldsymbol{\pi}$'

IGN: The non-ignoring principle

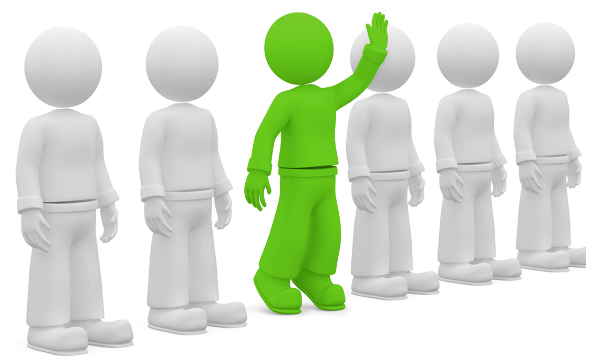
- All transitions are fired at least once in every circle
- Ensures that all transitions of $\boldsymbol{\pi}$ are eventually occurring in $\boldsymbol{\pi}$‘



All transitions are enabled

UPS: The up-set principle

- Stubborn set at m will always contain a transition of $\boldsymbol{\pi}$
- Between current marking and final marking there is a transition from up-set



BRA: The branching principle

- Ensures that visible transitions are not swapped with branches in the state space other than branches that are introduced by concurrency
- Enables reduction only in markings where just one (invisible) enabled transition is sufficient to meet all other principles

Partial order reduction for CTL

- Has severe restrictions
- Either a singleton set of an invisible transition, that satisfies all other criteria ...

Partial order reduction for CTL

- Has severe restrictions
- Either a singleton set of an invisible transition, that satisfies all other criteria ...

- Or we have to fire all enabled transitions
- Necessary to preserve BRA

Good news!

In all reported cases the very limiting BRA principle can be dropped.

BRA: enables reduction only in markings where just one (invisible) enabled transition is sufficient to meet all the other principles

Good news!

In all reported cases the very limiting BRA principle can be dropped.

In General:

less restrictive conditions (i.e. smaller set of principles to be met)

$\Rightarrow$ potentially smaller stubborn-sets

$\Rightarrow$ better reduction

BRA: enables reduction only in markings where just one (invisible) enabled transition is sufficient to meet all the other principles

AG ϕ, EF ϕ

AG ϕ = invariant, EF ϕ = reachability

- There are already well known stubborn sets
- Reachability: over 90 % - CTL only 65 %

$\Rightarrow$ Use the reachability stubborn sets

Structural analysis can also be applied:

- State equation with the CEGAR approach
- EF DEADLOCK – check for Commoner's theorem

EF ϕ – Exists a path, where finally ϕ holds?

AF ϕ, EG ϕ

AF ϕ ≡ F ϕ in LTL

⇒ Use LTL-X preserving stubborn sets (no BRA)

⇒ LTL: 90 % vs. CTL: 65 %

Drop IGN for visible transitions.

⇒ COM, KEY, VIS

⇒ Smaller stubborn sets

EG ϕ

EG ϕ – Exists a path, where permanently ϕ holds?

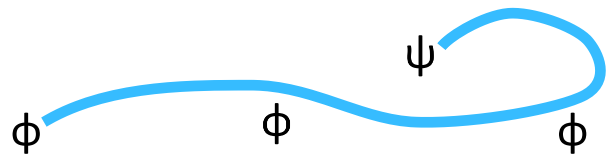
E (ϕ U ψ), A (ϕ R ψ)

E (ϕ U ψ) stubborn sets must preserve two properties:

1. Reachability of ψ
2. Non-violation of ϕ

Combining reachability (EF) and non-violation (EG) stubborn sets:

⇒ COM, UPS(ψ), VIS(ϕ)



E (ϕ U ψ) – Exists a path, where permanently ϕ holds until ψ holds?

EGEF ϕ, AFAG ϕ

No special stubborn sets → CTL-X stubborn sets

But: check for the pair of temporal operators can be folded into a single depth-first search

Two cases:

1. Deadlock: deadlock-state has to satisfy ϕ
2. Loop: from marking m on the loop, marking m’ satisfying ϕ is reachable

EGEF ϕ – Exists a path, where permanently EF ϕ holds ?

EFEG ϕ, AGAF ϕ

Nested depth-first search

Inner search: proceeds only through ϕ-markings and tries to find a cycle or a deadlock

$\Rightarrow$ COM, KEY, VIS(ϕ)

Outer search: proceeds through markings that have already proven not to be part of a ϕ-cycle (or a ϕ-deadlock)

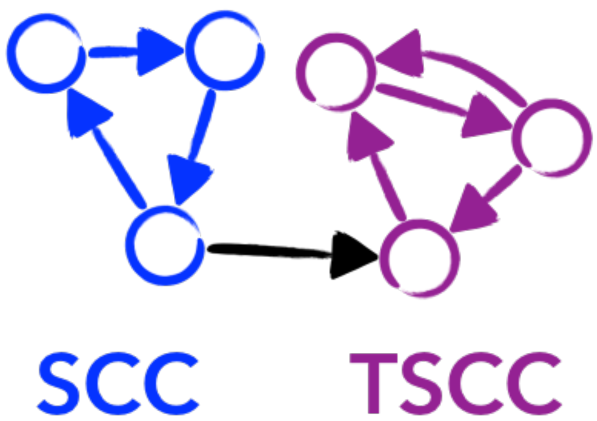
$\Rightarrow$ $m \not\models \phi$: COM, UPS(ϕ)

$\Rightarrow$ $m \models \phi$: COM, UPS(¬ϕ)

EFEG ϕ – Exists a path to a ϕ-loop or ϕ-deadlock?

AGEF ϕ, EFAG ϕ, AGEFAG ϕ, EFAGEF ϕ

- Only TSCC are relevant
- Use existing TSCC preserving stubborn sets



Formulas starting with EX and AX

Check the respective formula without the leading EX operator.

All we need to do is:

- explore all enabled transitions of m_0
- not store m_0

Whenever m_0 is visited during the search, it is treated as fresh marking.

Single-path formulas

- Aim: apply LTL model checking instead of CTL

$\Rightarrow$ BRA principle may be skipped

- Use rewriting system to recognise qualified formulas

Existential single-path formulas:

ϕ and ψ are existential single-path formulas, ω is a state predicate

- ω (base of inductive definition)
- $EG\ \omega$
- $EF\ \phi$
- $E\ (\omega\ U\ \phi)$
- $E\ (\phi\ R\ \omega)$
- $\phi \vee \psi$
- $\phi \wedge \omega$

Universal single-path formula are defined similar.

Boolean combinations

CTL formula is Boolean combination of subformulas

$\Rightarrow$ Check subformulas individual

Advantage:

- Smaller set of visible transition
- Apply stubborn sets to formulas without X-operator
- Some fall into the class considered above

Quick checks

For quite a few formulas we can add sufficient or necessary quick checks

E.g. AGEF ϕ → EF ϕ = nec., AG ϕ = suff.

State equation with CEGAR can be used

⇒ Not much memory used

⇒ Can run in parallel

⇒ Solved 1.24 % in MCC'2018

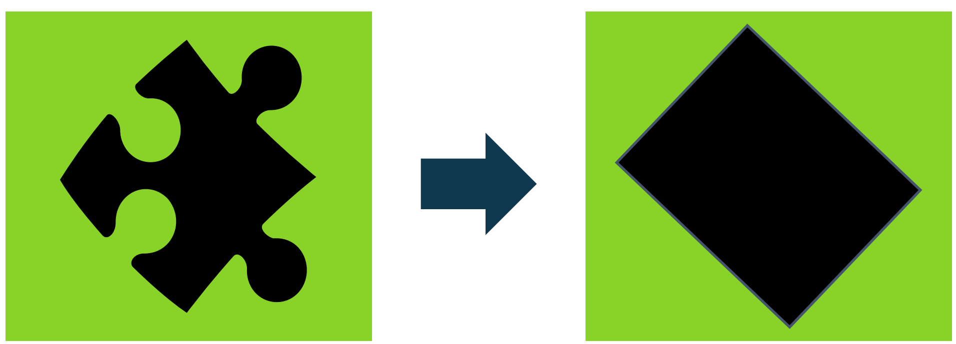
Simplify complex formulas

Tautologies – not all commonly known

- LoLA contains more than 100 rewrite rules based on CTL* tautologies

ILP-techniques using the Petri net state equation can be applied to atomic propositions

- Sometimes proving them invariantly true or false



Distribution in the MCC’2018 (Place/transition nets only)

	# of formulas	In %
All	24544	100,0
Special	13366	54,46
Preprocessing	3704	15,09
CTL	7474	30,45

Statistics

Formula	All	Solved CTL	Solved special	Solved more absolut	Solved more in %	Solved remaining in %
E(F(*)) / A(G(*))	2471	1438	2300	862	34,9	83,4
E(G(*)) / A(F(*))	1767	1625	1670	45	2,5	31,7
E((* R *)) / A((* U *))	168	157	160	3	1,8	27,3
E((* U *)) / A((* R *))	318	187	198	11	3,5	8,4
E(F(E(G(*)))) / A(G(A(F(*))))	515	340	431	91	17,7	52,0
E(G(E(F(*)))) / A(F(A(G(*))))	385	276	277	1	0,3	0,9
E(X(...)) / A(X(...))	602	407	539	132	21,9	67,7
Single-path formulas	421	275	295	20	4,8	13,7
TSCC-based	897	289	349	60	6,7	9,9
Boolean	5822	4250	5239	989	17,0	62,9
All	13366	9244	11458	2214	16,6	**53,7**

Summary

Simplify complex formulas

Use special features of the formula

Necessary / sufficient quick checks

CTL over 80%

Time for discussion!