Causality-based Verification of Multi-threaded Programs

joint work with Bernd Finkbeiner

Andrey Kupriyanov

Saarland University
Reactive Systems Group

September 25, 2014

Overview

LTL Model Checking

- ✓ Full LTL
- ✓ Infinite-state
- ✓ Multi-threading

$$T_1 \ \| \ \dots \ \| \ T_n \ \models$$
$$\Box\Diamond(at_{l_2} \wedge r > 0) \implies \Box\Diamond\, at_{l_3}$$

Overview

Safety/Reachability

$\pm$ $\forall\pi.\Box\Phi$ / $\exists\pi.\Diamond\Phi$

✓ Infinite-state

✓ Multi-threading

$$T_1 \parallel \ldots \parallel T_n \models$$
$$\Box\neg(at_{l_3} \wedge at_{m_3})$$

$\Longrightarrow$

LTL Model Checking

✓ Full LTL

✓ Infinite-state

✓ Multi-threading

$$T_1 \parallel \ldots \parallel T_n \models$$
$$\Box\Diamond(at_{l_2} \wedge r > 0) \Longrightarrow \Box\Diamond\, at_{l_3}$$

Overview

Safety/Reachability

$\pm$ $\forall\pi.\Box\Phi$ / $\exists\pi.\Diamond\Phi$

✓ Infinite-state

✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models$
$\Box\neg(at_{l_3} \wedge at_{m_3})$

$\Longrightarrow$

LTL Model Checking

✓ Full LTL

✓ Infinite-state

✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models$
$\Box\Diamond(at_{l_2} \wedge r > 0) \implies \Box\Diamond\, at_{l_3}$

$\Longleftarrow$

Liveness/Termination

$\pm$ $\forall\pi.\Diamond\Phi$ / $\exists\pi.\Box\Phi$

✓ Infinite-state

✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models$
$at_{l_2} \implies \Diamond\, at_{l_3}$

Overview

Safety/Reachability

$\pm$ $\forall\pi.\Box\Phi$ / $\exists\pi.\Diamond\Phi$
✓ Infinite-state
✓ Multi-threading

$$T_1 \parallel \ldots \parallel T_n \models \Box\neg(at_{l_3} \wedge at_{m_3})$$

$\Longrightarrow$

LTL Model Checking

✓ Full LTL
✓ Infinite-state
✓ Multi-threading

$$T_1 \parallel \ldots \parallel T_n \models \Box\Diamond(at_{l_2} \wedge r > 0) \Longrightarrow \Box\Diamond\, at_{l_3}$$

$\Longleftarrow$

Liveness/Termination

$\pm$ $\forall\pi.\Diamond\Phi$ / $\exists\pi.\Box\Phi$
✓ Infinite-state
✓ Multi-threading

$$T_1 \parallel \ldots \parallel T_n \models at_{l_2} \Longrightarrow \Diamond\, at_{l_3}$$

$\Uparrow$

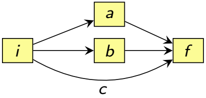
LTL Satisfiability/Validity

✓ Full LTL
$\mp$ Boolean
— No system

$$\models \Diamond p \wedge \Box(p \Longrightarrow \bigcirc p) \Longrightarrow \Diamond\Box p$$

Our approach

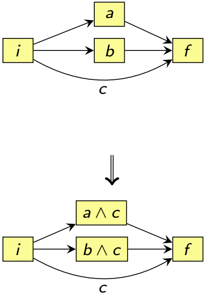
- **Proof objects: concurrent traces**
  allow to capture temporal order, constraints, independence



Our approach

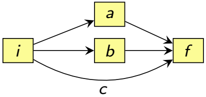
- **Proof objects: concurrent traces**
  allow to capture temporal order, constraints, independence

- **Proof rules based on causality**
  causality $\equiv$ language-preserving trace transformations

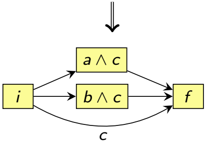


Our approach

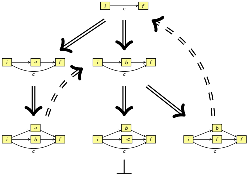
- **Proof objects: concurrent traces**
  allow to capture temporal order, constraints, independence



- **Proof rules based on causality**
  causality $\equiv$ language-preserving trace transformations



- **Proof construction: tableau search based on causal loops**
  causal loops $\equiv$ infinitely-looping trace transformations



Example: mutual exclusion for semaphores

Thread 1

```
while (true) {
  l1: noncritical;
  l2: request r;
  l3: critical;
  l4: release r;
}
```

Thread 2

```
while (true) {
  m1: noncritical;
  m2: request r;
  m3: critical;
  m4: release r;
}
```

Thread 3

```
while (true) {
  n1: noncritical;
  n2: request r;
  n3: critical;
  n4: release r;
}
```

Example: mutual exclusion for semaphores

Thread 1

```
while (true) {
  l1:  noncritical;
  l2:  request r;
  l3:  critical;
  l4:  release r;
}
```

Thread 2

```
while (true) {
  m1:  noncritical;
  m2:  request r;
  m3:  critical;
  m4:  release r;
}
```

Thread 3

```
while (true) {
  n1:  noncritical;
  n2:  request r;
  n3:  critical;
  n4:  release r;
}
```

Definition (Most general semaphore class)

Simple semaphore class +

- arbitrary control flow
- arbitrary number of semaphore variables

Example: mutual exclusion for semaphores

Thread 1

```
while (true) {
  l1:  noncritical;
  l2:  request r;
  l3:  critical;
  l4:  release r;
}
```

Thread 2

```
while (true) {
  m1:  noncritical;
  m2:  request r;
  m3:  critical;
  m4:  release r;
}
```

Thread 3

```
while (true) {
  n1:  noncritical;
  n2:  request r;
  n3:  critical;
  n4:  release r;
}
```

Definition (Most general semaphore class)

Simple semaphore class +

- arbitrary control flow
- arbitrary number of semaphore variables

Open problem

Is the most general semaphore class polynomially verifiable for a fixed number of locks?

Example: mutual exclusion for semaphores

Thread 1

```
while (true) {
  l1:  noncritical;
  l2:  request r;
  l3:  critical;
  l4:  release r;
}
```

Thread 2

```
while (true) {
  m1:  noncritical;
  m2:  request r;
  m3:  critical;
  m4:  release r;
}
```

Thread 3

```
while (true) {
  n1:  noncritical;
  n2:  request r;
  n3:  critical;
  n4:  release r;
}
```

Definition (Most general semaphore class)

Simple semaphore class +

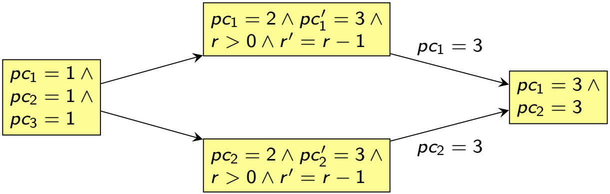
- arbitrary control flow
- arbitrary number of semaphore variables

Open problem

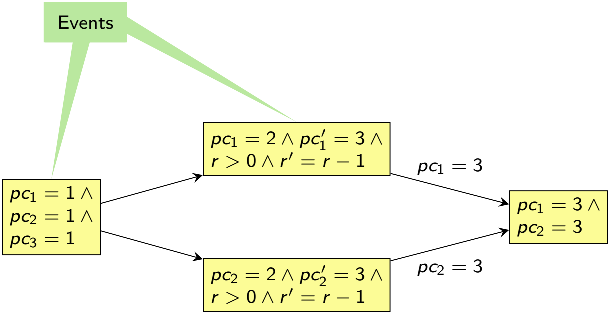
Is the most general semaphore class polynomially verifiable for a fixed number of locks?

Our causality-based reachability analysis algorithm
has settled this question affirmatively.

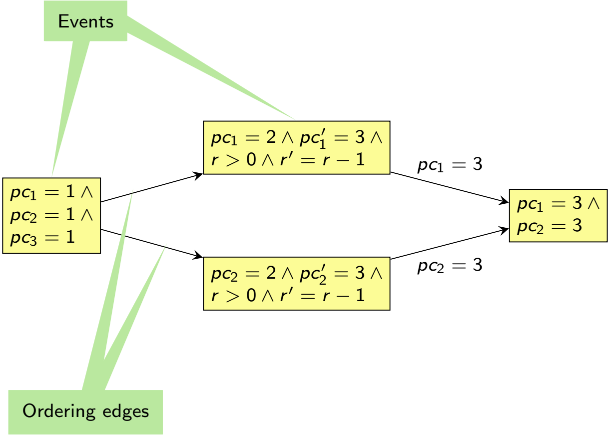
Finite concurrent traces



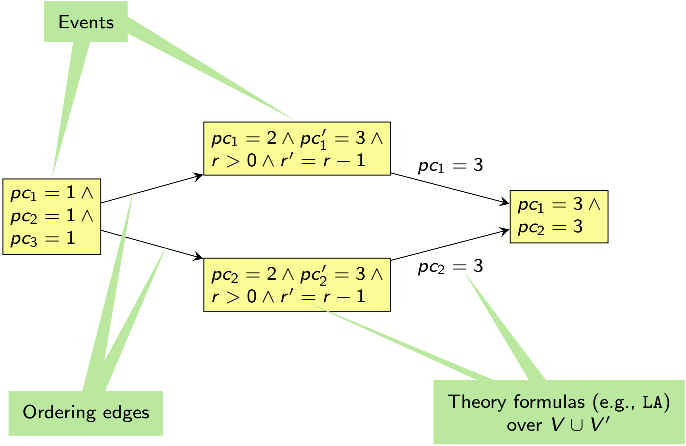
Finite concurrent traces



Finite concurrent traces



Finite concurrent traces



Mutual exclusion for semaphores: a causal proof

Thread 1

while (true) {
  l_1: `noncritical;`
  l_2: `request r;`
  l_3: `critical;`
  l_4: `release r;`
}

Thread 2

while (true) {
  m_1: `noncritical;`
  m_2: `request r;`
  m_3: `critical;`
  m_4: `release r;`
}

Thread 3

while (true) {
  n_1: `noncritical;`
  n_2: `request r;`
  n_3: `critical;`
  n_4: `release r;`
}

Mutual exclusion for semaphores: a causal proof

Thread 1

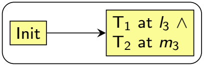
```
while (true) {
  l1:  noncritical;
  l2:  request r;
  l3:  critical;
  l4:  release r;
}
```

Thread 2

```
while (true) {
  m1:  noncritical;
  m2:  request r;
  m3:  critical;
  m4:  release r;
}
```

Thread 3

```
while (true) {
  n1:  noncritical;
  n2:  request r;
  n3:  critical;
  n4:  release r;
}
```



Mutual exclusion for semaphores: a causal proof

Thread 1

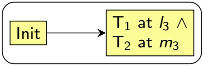
```
while (true) {
  l1:  noncritical;
  l2:  request r;
  l3:  critical;
  l4:  release r;
}
```

Thread 2

```
while (true) {
  m1:  noncritical;
  m2:  request r;
  m3:  critical;
  m4:  release r;
}
```

Thread 3

```
while (true) {
  n1:  noncritical;
  n2:  request r;
  n3:  critical;
  n4:  release r;
}
```



What is necessary?

Mutual exclusion for semaphores: a causal proof

Thread 1

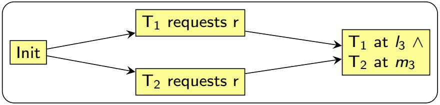
```
while (true) {
  l1:  noncritical;
  l2:  request r;
  l3:  critical;
  l4:  release r;
}
```

Thread 2

```
while (true) {
  m1:  noncritical;
  m2:  request r;
  m3:  critical;
  m4:  release r;
}
```

Thread 3

```
while (true) {
  n1:  noncritical;
  n2:  request r;
  n3:  critical;
  n4:  release r;
}
```



Mutual exclusion for semaphores: a causal proof

Thread 1

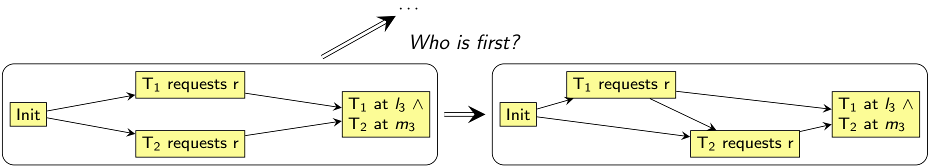
```
while (true) {
  l1:  noncritical;
  l2:  request r;
  l3:  critical;
  l4:  release r;
}
```

Thread 2

```
while (true) {
  m1:  noncritical;
  m2:  request r;
  m3:  critical;
  m4:  release r;
}
```

Thread 3

```
while (true) {
  n1:  noncritical;
  n2:  request r;
  n3:  critical;
  n4:  release r;
}
```



Mutual exclusion for semaphores: a causal proof

Thread 1

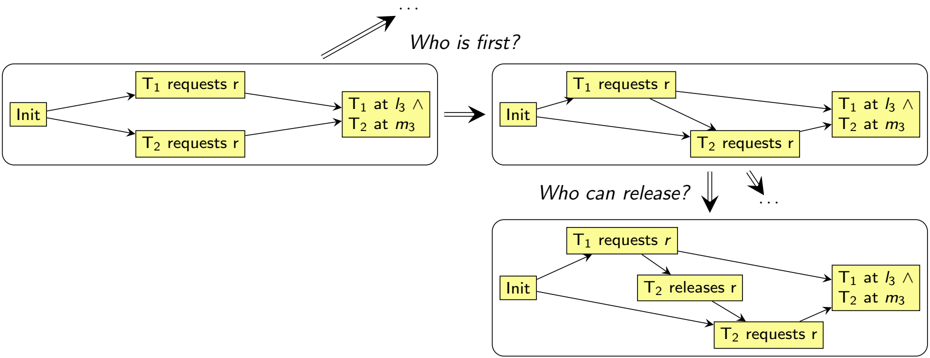
```
while (true) {
  l1:  noncritical;
  l2:  request r;
  l3:  critical;
  l4:  release r;
}
```

Thread 2

```
while (true) {
  m1:  noncritical;
  m2:  request r;
  m3:  critical;
  m4:  release r;
}
```

Thread 3

```
while (true) {
  n1:  noncritical;
  n2:  request r;
  n3:  critical;
  n4:  release r;
}
```



Mutual exclusion for semaphores: a causal proof

Thread 1

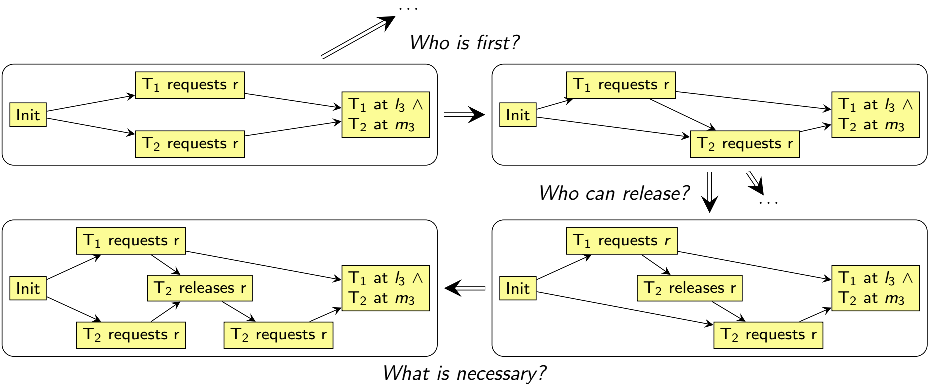
```
while (true) {
  l1:  noncritical;
  l2:  request r;
  l3:  critical;
  l4:  release r;
}
```

Thread 2

```
while (true) {
  m1:  noncritical;
  m2:  request r;
  m3:  critical;
  m4:  release r;
}
```

Thread 3

```
while (true) {
  n1:  noncritical;
  n2:  request r;
  n3:  critical;
  n4:  release r;
}
```



Mutual exclusion for semaphores: a causal proof

Thread 1

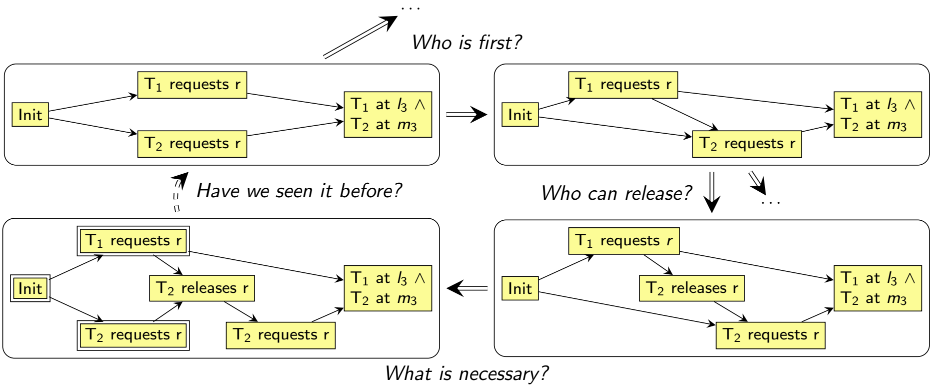
```
while (true) {
  l1:  noncritical;
  l2:  request r;
  l3:  critical;
  l4:  release r;
}
```

Thread 2

```
while (true) {
  m1:  noncritical;
  m2:  request r;
  m3:  critical;
  m4:  release r;
}
```

Thread 3

```
while (true) {
  n1:  noncritical;
  n2:  request r;
  n3:  critical;
  n4:  release r;
}
```



Finite concurrent traces

Transition system $\mathcal{S} = \langle V, I, T \rangle$

- V: variables
- $I \in \Phi(V')$: initialization
- $T \subseteq \Phi(V \cup V')$: transitions

Finite concurrent traces

- $I \equiv x = 0 \wedge y = 0$
- $T \equiv \{\ \boldsymbol{x^+}\colon\ x' = x + 1 \wedge y' = y$
  $\boldsymbol{x^-}\colon\ x' = x - 1 \wedge y' = y$
  $\boldsymbol{y^+}\colon\ y' = y + 1 \wedge x' = x$
  $\boldsymbol{y^-}\colon\ y' = y - 1 \wedge x' = x\ \}$

Finite concurrent traces

- $I \;\equiv x = 0 \land y = 0$
- $T \equiv \{\, \boldsymbol{x^+}\colon\ x' = x + 1 \land y' = y$
  $\qquad \boldsymbol{x^-}\colon\ x' = x - 1 \land y' = y$
  $\qquad \boldsymbol{y^+}\colon\ y' = y + 1 \land x' = x$
  $\qquad \boldsymbol{y^-}\colon\ y' = y - 1 \land x' = x \,\}$
- $F \;\equiv x = 1 \land y = 1$

Finite concurrent traces

Finite trace $\mathcal{A} = \langle N, E, \nu, \eta \rangle$

- $\langle N, E \rangle$ is a DAG
- $\nu : N \to \Phi(V \cup V')$
- $\eta : E \to \Phi(V \cup V')$

- $I \equiv x = 0 \land y = 0$
- $T \equiv \{\ \boldsymbol{x^+}\colon\ x' = x + 1 \land y' = y$
  $\boldsymbol{x^-}\colon\ x' = x - 1 \land y' = y$
  $\boldsymbol{y^+}\colon\ y' = y + 1 \land x' = x$
  $\boldsymbol{y^-}\colon\ y' = y - 1 \land x' = x\ \}$
- $F \equiv x = 1 \land y = 1$

Finite concurrent traces

Finite trace $\mathcal{A} = \langle N, E, \nu, \eta \rangle$

- $\langle N, E \rangle$ is a DAG
- $\nu : N \to \Phi(V \cup V')$
- $\eta : E \to \Phi(V \cup V')$

- $I \ \equiv x = 0 \wedge y = 0$
- $T \equiv \{ \ \boldsymbol{x}^{+}\!: \ x' = x + 1 \wedge y' = y$
  $\boldsymbol{x}^{-}\!: \ x' = x - 1 \wedge y' = y$
  $\boldsymbol{y}^{+}\!: \ y' = y + 1 \wedge x' = x$
  $\boldsymbol{y}^{-}\!: \ y' = y - 1 \wedge x' = x \ \}$
- $F \ \equiv x = 1 \wedge y = 1$

Language of a finite concurrent trace

A set of system runs such that a *linearization* of a concurrent trace can be mapped into a *subsequence* of a run, *respecting constraints*

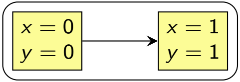
Finite concurrent traces

- $I \equiv x = 0 \land y = 0$
- $T \equiv \{ \; \boldsymbol{x^+}\colon x' = x + 1 \land y' = y$
  $\boldsymbol{x^-}\colon x' = x - 1 \land y' = y$
  $\boldsymbol{y^+}\colon y' = y + 1 \land x' = x$
  $\boldsymbol{y^-}\colon y' = y - 1 \land x' = x \; \}$
- $F \equiv x = 1 \land y = 1$

Language of a finite concurrent trace

A set of system runs such that a *linearization* of a concurrent trace can be mapped into a *subsequence* of a run, *respecting constraints*

Finite concurrent traces



- $I \;\equiv\; x = 0 \wedge y = 0$
- $T \equiv \{\; \boldsymbol{x^+}\colon\; x' = x + 1 \wedge y' = y$
  $\boldsymbol{x^-}\colon\; x' = x - 1 \wedge y' = y$
  $\boldsymbol{y^+}\colon\; y' = y + 1 \wedge x' = x$
  $\boldsymbol{y^-}\colon\; y' = y - 1 \wedge x' = x \;\}$
- $F \;\equiv\; x = 1 \wedge y = 1$

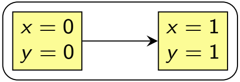
Accepted runs

- $I, \boldsymbol{x^+}, \boldsymbol{y^+}, F$
- $I, \boldsymbol{y^+}, \boldsymbol{x^+}, F$
- $I, \boldsymbol{y^+}, \boldsymbol{x^+}, \boldsymbol{x^-}, \boldsymbol{x^+}, F$
- ...

Rejected runs

- $I, \boldsymbol{x^+}, F$
- $I, \boldsymbol{x^+}, \boldsymbol{y^+}, \boldsymbol{x^+}, F$
- $I, \boldsymbol{x^-}, \boldsymbol{y^-}, F$
- ...

Finite concurrent traces



- $I \;\equiv x = 0 \wedge y = 0$
- $T \equiv \{\, \boldsymbol{x^+}\colon\ x' = x + 1 \wedge y' = y$
  $\quad\ \boldsymbol{x^-}\colon\ x' = x - 1 \wedge y' = y$
  $\quad\ \boldsymbol{y^+}\colon\ y' = y + 1 \wedge x' = x$
  $\quad\ \boldsymbol{y^-}\colon\ y' = y - 1 \wedge x' = x \,\}$
- $F \;\equiv x = 1 \wedge y = 1$

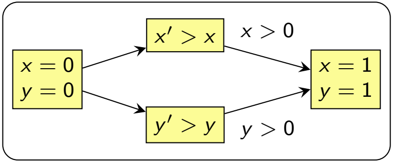
Accepted runs

- $I, \boldsymbol{x^+}, \boldsymbol{y^+}, F$
- $I, \boldsymbol{y^+}, \boldsymbol{x^+}, F$
- $I, \boldsymbol{y^+}, \boldsymbol{x^+}, \boldsymbol{x^-}, \boldsymbol{x^+}, F$
- ...

Rejected runs

- $I, \boldsymbol{x^+}, F$
- $I, \boldsymbol{x^+}, \boldsymbol{y^+}, \boldsymbol{x^+}, F$
- $I, \boldsymbol{x^-}, \boldsymbol{y^-}, F$
- ...

Finite concurrent traces



- $I \;\equiv\; x = 0 \wedge y = 0$
- $T \equiv \{\; \boldsymbol{x^+}\colon\; x' = x + 1 \wedge y' = y$
  $\boldsymbol{x^-}\colon\; x' = x - 1 \wedge y' = y$
  $\boldsymbol{y^+}\colon\; y' = y + 1 \wedge x' = x$
  $\boldsymbol{y^-}\colon\; y' = y - 1 \wedge x' = x \;\}$
- $F \;\equiv\; x = 1 \wedge y = 1$

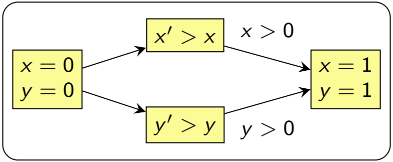
Accepted runs

- $I, \boldsymbol{x^+}, \boldsymbol{y^+}, F$
- $I, \boldsymbol{y^+}, \boldsymbol{x^+}, F$
- $I, \boldsymbol{y^+}, \boldsymbol{x^+}, \boldsymbol{x^-}, \boldsymbol{x^+}, F$
- ...

Rejected runs

- $I, \boldsymbol{x^+}, F$
- $I, \boldsymbol{x^+}, \boldsymbol{y^+}, \boldsymbol{x^+}, F$
- $I, \boldsymbol{x^-}, \boldsymbol{y^-}, F$
- ...

Finite concurrent traces

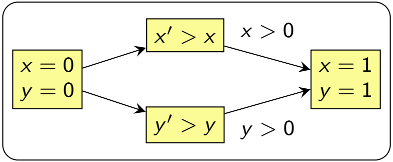


- $I \;\equiv\; x = 0 \wedge y = 0$
- $T \equiv \{\, \boldsymbol{x^+}\!: x' = x + 1 \wedge y' = y$
  $\boldsymbol{x^-}\!: x' = x - 1 \wedge y' = y$
  $\boldsymbol{y^+}\!: y' = y + 1 \wedge x' = x$
  $\boldsymbol{y^-}\!: y' = y - 1 \wedge x' = x \,\}$
- $F \;\equiv\; x = 1 \wedge y = 1$

Accepted run

$I \qquad \boldsymbol{x^+} \qquad \boldsymbol{y^+} \qquad F$

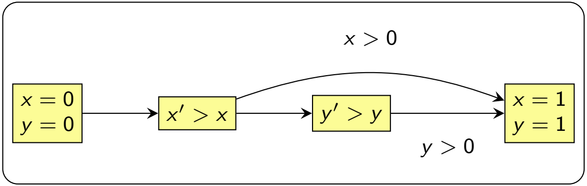
Finite concurrent traces



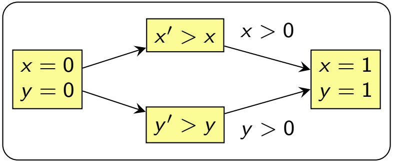
- $I \equiv x = 0 \wedge y = 0$
- $T \equiv \{\, \boldsymbol{x^+}\!: x' = x + 1 \wedge y' = y$
  $\boldsymbol{x^-}\!: x' = x - 1 \wedge y' = y$
  $\boldsymbol{y^+}\!: y' = y + 1 \wedge x' = x$
  $\boldsymbol{y^-}\!: y' = y - 1 \wedge x' = x \,\}$
- $F \equiv x = 1 \wedge y = 1$

Accepted run

I $\boldsymbol{x^+}$ $\boldsymbol{y^+}$ F



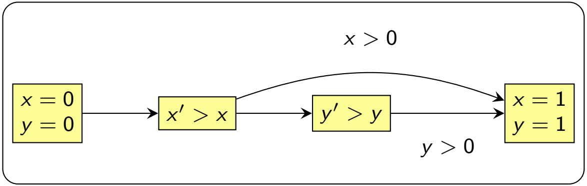
Finite concurrent traces



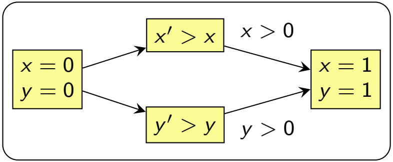
- $I \equiv x = 0 \land y = 0$
- $T \equiv \{\ \boldsymbol{x^+}\colon\ x' = x + 1 \land y' = y$
  $\boldsymbol{x^-}\colon\ x' = x - 1 \land y' = y$
  $\boldsymbol{y^+}\colon\ y' = y + 1 \land x' = x$
  $\boldsymbol{y^-}\colon\ y' = y - 1 \land x' = x\ \}$
- $F \equiv x = 1 \land y = 1$

Accepted run

I $\quad$ $\boldsymbol{y^+}$ $\quad$ $\boldsymbol{x^+}$ $\quad$ F



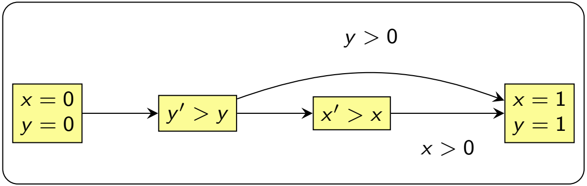
Finite concurrent traces



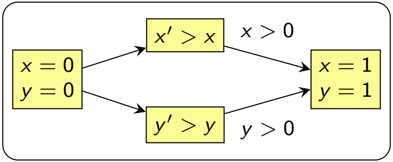
- $I \equiv x = 0 \wedge y = 0$
- $T \equiv \{\ \boldsymbol{x^+}\colon\ x' = x + 1 \wedge y' = y$
  $\boldsymbol{x^-}\colon\ x' = x - 1 \wedge y' = y$
  $\boldsymbol{y^+}\colon\ y' = y + 1 \wedge x' = x$
  $\boldsymbol{y^-}\colon\ y' = y - 1 \wedge x' = x\ \}$
- $F \equiv x = 1 \wedge y = 1$

Accepted run

$I \qquad \boldsymbol{y^+} \qquad \boldsymbol{x^+} \qquad F$



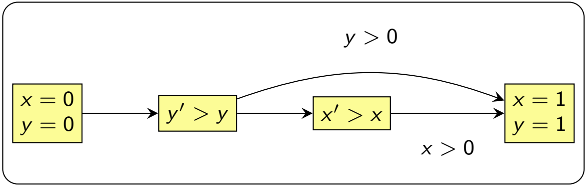
Finite concurrent traces



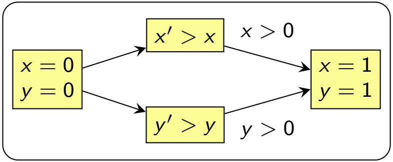
- $I \equiv x = 0 \wedge y = 0$
- $T \equiv \{\, \boldsymbol{x^+}\colon\ x' = x + 1 \wedge y' = y$
  $\boldsymbol{x^-}\colon\ x' = x - 1 \wedge y' = y$
  $\boldsymbol{y^+}\colon\ y' = y + 1 \wedge x' = x$
  $\boldsymbol{y^-}\colon\ y' = y - 1 \wedge x' = x \,\}$
- $F \equiv x = 1 \wedge y = 1$

Accepted run

I	$\boldsymbol{y^+}$	$\boldsymbol{x^+}$	$\boldsymbol{x^-}$	$\boldsymbol{x^+}$	F



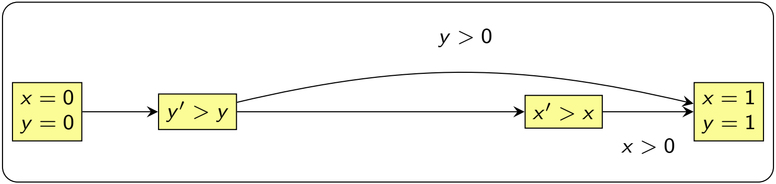
Finite concurrent traces



- $I \;\equiv\; x = 0 \wedge y = 0$
- $T \equiv \{\; \boldsymbol{x^+}\!: \; x' = x + 1 \wedge y' = y$
  $\quad \boldsymbol{x^-}\!: \; x' = x - 1 \wedge y' = y$
  $\quad \boldsymbol{y^+}\!: \; y' = y + 1 \wedge x' = x$
  $\quad \boldsymbol{y^-}\!: \; y' = y - 1 \wedge x' = x \;\}$
- $F \;\equiv\; x = 1 \wedge y = 1$

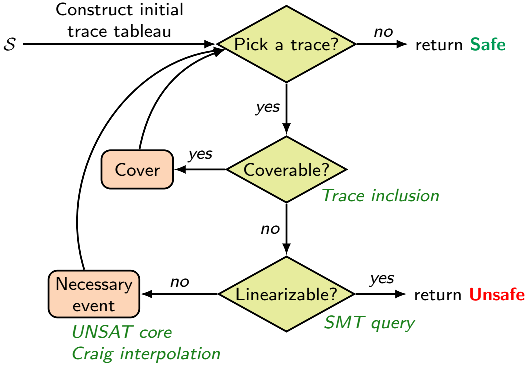
Accepted run

I	$\boldsymbol{y^+}$	$\boldsymbol{x^+}$	$\boldsymbol{x^-}$	$\boldsymbol{x^+}$	F



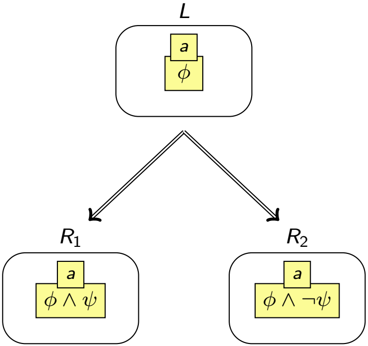
Reachability analysis algorithm

[K., Finkbeiner, CONCUR 2013]

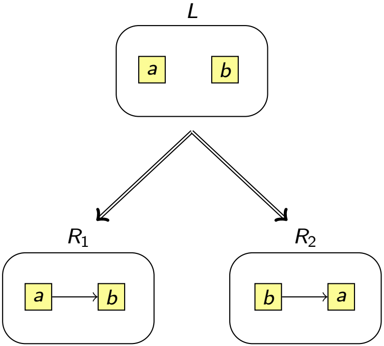


Proof rules: finite traces

Event split

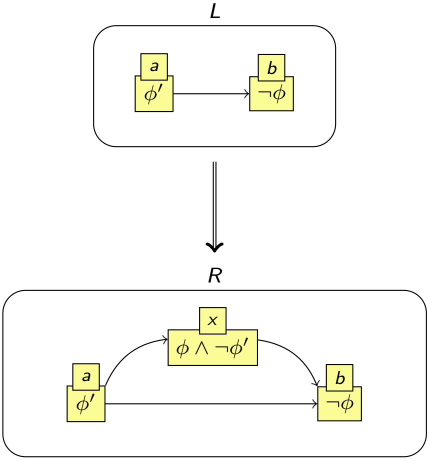


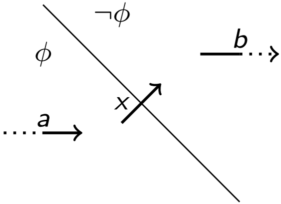
Order split



Proof rules: finite traces

Necessary event





Safety

Theorem (Soundness)

If there exists a correct and complete causal trace tableau for a transition system $\mathcal{S}$, then $\mathcal{S}$ is safe.

Theorem (Relative completeness)

If a transition system $\mathcal{S}$ is safe, then a correct and complete causal trace tableau for $\mathcal{S}$ can be constructed, provided that all necessary first-order formulas are given.

Theorem (Polynomiality for semaphore programs)

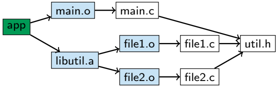
Causality-based verification algorithm proves the safety of the most general class of multi-threaded semaphore programs in deterministic polynomial time with respect to the number of threads and locks.

Termination of multi-threaded programs

- Parallel compilation (e.g. GNU Make)

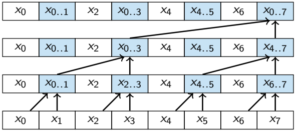
  gnu.org/software/make/

  `make -j N`



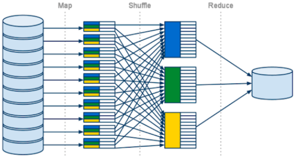
- Parallel computations in GPUs (OpenCL, CUDA)

  developer.nvidia.com/cuda-zone/



- Distributed processing (e.g. the Map-Reduce architecture)

  developers.google.com/appengine/docs/java/dataprocessing/



- Device drivers, leader election, ...

Producer-Consumer (Map-Reduce architecture)

Producer 1
```
while (p1>0) {
  if(*) q1++;
  else  q2++;
  p1--;
}
```

Producer 2
```
while (p2>0) {
  if(*) q1++;
  else  q2++;
  p2--;
}
```

Consumer 1
```
while (true) {
  await(q1>0);
  skip; //step 1
  skip; //step 2
  q1--;
}
```

Consumer 2
```
while (true) {
  await(q2>0);
  skip; //step 1
  skip; //step 2
  q2--;
}
```

Producer-Consumer (Map-Reduce architecture)

Producer 1
```
while (p1>0) {
  if(*) q1++;
  else  q2++;
  p1--;
}
```

Producer 2
```
while (p2>0) {
  if(*) q1++;
  else  q2++;
  p2--;
}
```

Consumer 1
```
while (true) {
  await(q1>0);
  skip; //step 1
  skip; //step 2
  q1--;
}
```

Consumer 2
```
while (true) {
  await(q2>0);
  skip; //step 1
  skip; //step 2
  q2--;
}
```

	Terminator		T2		AProVE	
Threads	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)
1	3.37	26	2.42	38	3.17	237
2	1397	1394	3.25	44	6.79	523
3	×	MO	U(29.2)	253	U(26.6)	1439
4	×	MO	U(36.6)	316	U(71.2)	1455
5	×	MO	U(30.7)	400	U(312)	1536
10	×	MO	Z3-TO	×	×	MO
20	×	MO	Z3-TO	×	×	MO
40	×	MO	Z3-TO	×	×	MO
60	×	MO	Z3-TO	×	×	MO
80	×	MO	Z3-TO	×	×	MO
100	×	MO	Z3-TO	×	×	MO

Producer-Consumer (Map-Reduce architecture)

Producer 1
```
while (p1>0) {
  if(*) q1++;
  else  q2++;
  p1--;
}
```

Producer 2
```
while (p2>0) {
  if(*) q1++;
  else  q2++;
  p2--;
}
```

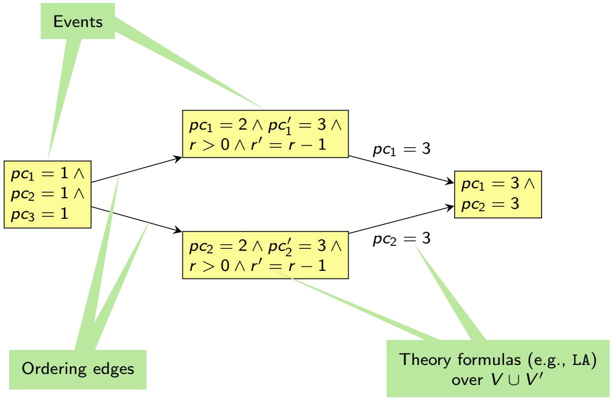
Consumer 1
```
while (true) {
  await(q1>0);
  skip; //step 1
  skip; //step 2
  q1--;
}
```

Consumer 2
```
while (true) {
  await(q2>0);
  skip; //step 1
  skip; //step 2
  q2--;
}
```

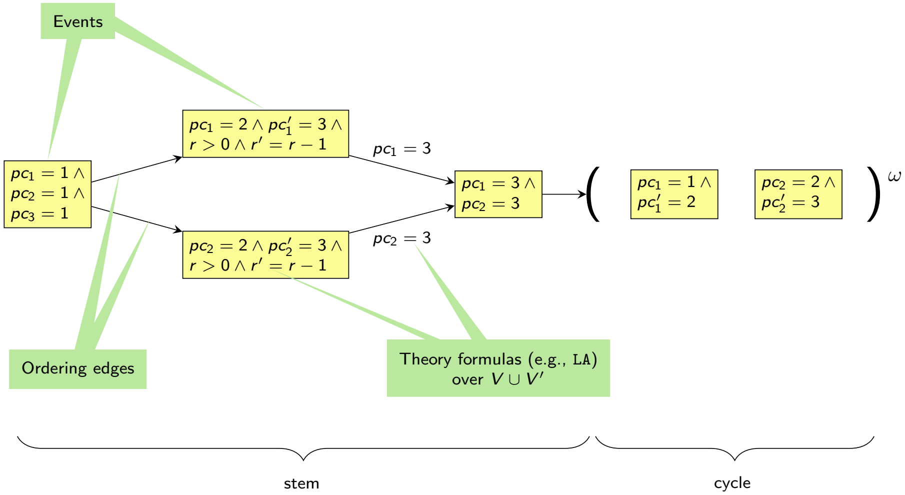
	Terminator		T2		AProVE		**Arctor** [1]	
Threads	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)
1	3.37	26	2.42	38	3.17	237	**0.002**	**2.3**
2	1397	1394	3.25	44	6.79	523	**0.002**	**2.6**
3	×	MO	U(29.2)	253	U(26.6)	1439	**0.002**	**2.6**
4	×	MO	U(36.6)	316	U(71.2)	1455	**0.003**	**2.7**
5	×	MO	U(30.7)	400	U(312)	1536	**0.007**	**2.7**
10	×	MO	Z3-TO	×	×	MO	**0.027**	**3.0**
20	×	MO	Z3-TO	×	×	MO	**0.30**	**4.2**
40	×	MO	Z3-TO	×	×	MO	**4.30**	**12.7**
60	×	MO	Z3-TO	×	×	MO	**20.8**	**35**
80	×	MO	Z3-TO	×	×	MO	**67.7**	**145**
100	×	MO	Z3-TO	×	×	MO	**172**	**231**

[1] **Arctor** : **A**bstraction **R**efinement of **C**oncurrent **T**emporal **Or**derings (react.uni-saarland.de/tools/arctor/)

Infinite concurrent traces

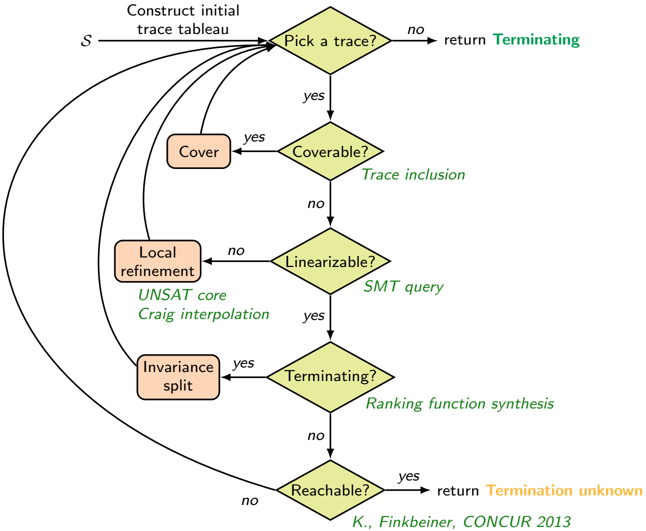


Infinite concurrent traces



Termination analysis algorithm

[K., Finkbeiner, CAV 2014]



Termination: soundness and completeness

Theorem (Soundness)

If there exists a correct and complete causal trace tableau for a transition system $\mathcal{S}$, then $\mathcal{S}$ is terminating.

Theorem (Relative completeness)

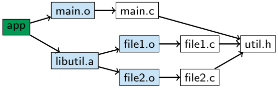
If a transition system $\mathcal{S}$ is terminating, then a correct and complete causal trace tableau for $\mathcal{S}$ can be constructed, provided that all necessary first-order formulas are given.

Experimental results: simple programs

	Terminator		T2		AProVE		Arctor		
Benchmark	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)	Vertices
Chain 2	0.65	20	0.52	20	1.58	131	0.002	2.0	3
Chain 4	1.45	25	0.54	22	2.13	153	0.002	2.2	7
Chain 6	24.4	57	0.58	24	2.58	171	0.002	2.5	11
Chain 8	×	MO	0.63	26	3.48	210	0.002	2.5	15
Chain 20	×	MO	2.36	55	16.5	941	0.007	2.5	39
Chain 40	×	MO	40.5	288	536	1237	0.023	2.8	79
Chain 60	×	MO	Z3-TO	×	×	MO	0.063	3.0	119
Chain 80	×	MO	Z3-TO	×	×	MO	0.145	3.3	159
Chain 100	×	MO	Z3-TO	×	×	MO	0.320	3.9	199
Phase 1	×	MO	U(4.53)	48	1.60	132	0.002	2.4	2
Phase 2	×	MO	U(4.53)	48	2.16	144	0.002	2.4	11
Phase 3	×	MO	U(30.6)	301	3.83	199	0.002	2.5	20
Phase 4	×	MO	×	MO	8.89	336	0.003	2.6	29
Phase 8	×	MO	×	MO	47.0	1506	0.003	2.6	65
Phase 10	×	MO	×	MO	×	MO	0.012	2.7	83
Phase 20	×	MO	×	MO	×	MO	0.061	3.3	173
Phase 40	×	MO	×	MO	×	MO	0.35	4.0	353
Phase 60	×	MO	×	MO	×	MO	1.18	4.2	533
Phase 80	×	MO	×	MO	×	MO	3.21	5.1	713
Phase 100	×	MO	×	MO	×	MO	7.38	6.1	893
Semaphore 1	3.05	26	2.81	46	3.22	230	0.002	2.6	8
Semaphore 2	622	691	U(20.7)	219	U(6.52)	465	0.002	2.6	16
Semaphore 3	×	MO	U(15.8)	239	U(10.42)	1138	0.003	2.6	24
Semaphore 10	×	MO	U(83.5)	470	U(246)	1287	0.023	2.8	80
Semaphore 20	×	MO	×	MO	×	MO	0.073	3.3	160
Semaphore 40	×	MO	×	MO	×	MO	0.264	4.0	320
Semaphore 60	×	MO	×	MO	×	MO	0.58	4.0	480
Semaphore 80	×	MO	×	MO	×	MO	1.02	4.6	640
Semaphore 100	×	MO	×	MO	×	MO	1.59	5.1	800

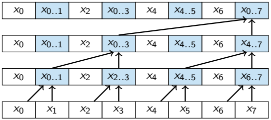
Experimental results: models of industrial programs

- Parallel compilation (GNU Make)



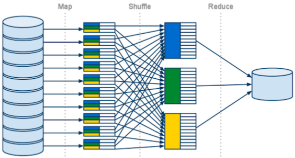
Threads	Time(s)	Mem.(MB)	Vertices
2	0.04	3.6	126
3	0.10	4.3	189
4	0.17	4.5	252
5	0.26	4.5	315
6	0.36	4.5	378
7	0.48	4.5	441
8	0.62	4.6	504
9	0.79	5.5	567
10	0.97	5.5	630

- Parallel computations in GPUs (CUDA)



Threads	Time(s)	Mem.(MB)	Vertices
2	0.04	3.3	86
3	0.09	3.7	129
4	0.15	4.3	172
5	0.24	4.5	215
6	0.33	4.5	258
7	0.45	4.6	301
8	0.58	5.5	344
9	0.72	5.5	387
10	0.88	5.5	430

- Distributed processing (Map-Reduce)



Threads	Time(s)	Mem.(MB)	Vertices
2	0.42	4.5	238
3	2.50	4.5	393
4	8.22	5.5	547
5	31.3	6.5	767
6	78.7	6.5	986
7	219	7.3	1271
8	457	8.3	1555
9	1053	9.3	1905
10	1924	11.4	2254

No other termination prover can handle even 2 threads!

Arctor

LTL Satisfiability/Validity

Applications

Specification debugging:

- detection of unsatisfiable specifications
- detection of vacuous specifications

Specification understanding:

- small models/countermodels

Can be used for finite-state LTL model checking by a simple reduction

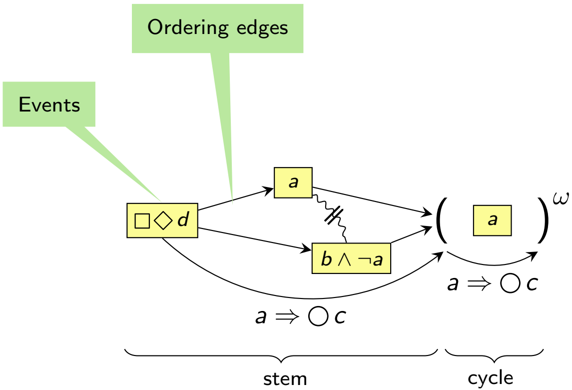
Captures the LTL complexity

PSPACE-complete even for simple fragments $L(F, X)$, $L(U)$

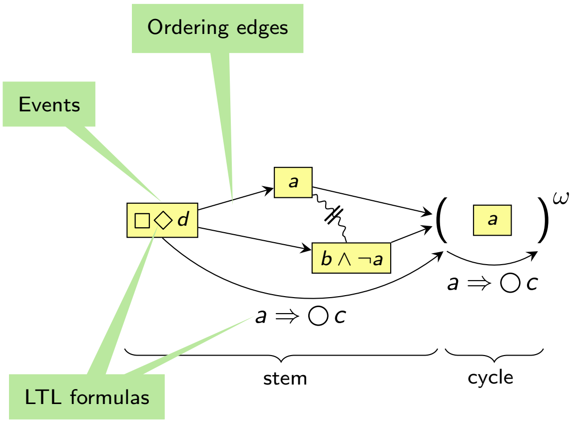
Decision algorithms

- Tableau calculus [Schwendimann, 1998]
- Clausal temporal resolution [Fischer, Dixon, Peim, 2001]
- Reduction to automata-based model checking [Rozier, Vardi, 2007]

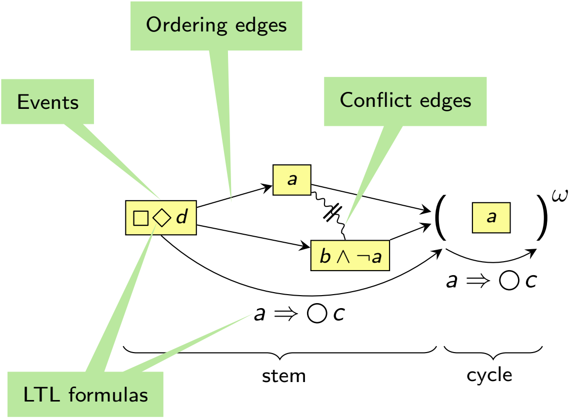
LTL concurrent traces



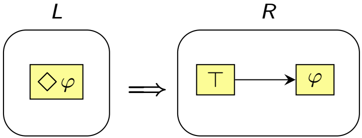
LTL concurrent traces



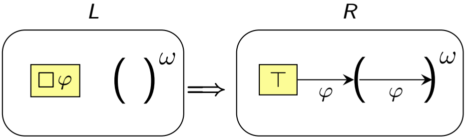
LTL concurrent traces



LTL proof rules

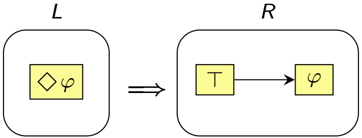


Finally

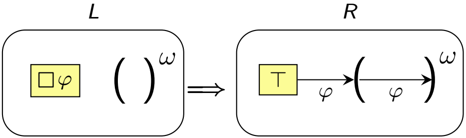


Globally

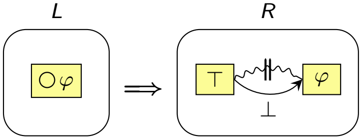
LTL proof rules



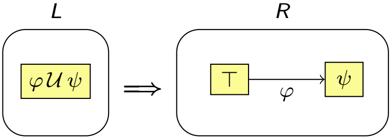
Finally



Globally



Next



Until

LTL satisfiability: $\Box\Diamond p \;\wedge\; \Box\Diamond\neg p$

LTL satisfiability: $\Box\Diamond p \ \wedge\ \Box\Diamond\neg p$

$$(\mathsf{G\,F}\,p \wedge \mathsf{G\,F}\,\neg p) \mid \ldots \mid \ldots$$
$$(\mathsf{G\,F}\,p), \mathsf{G\,F}\,\neg p \mid \ldots \mid \ldots$$
$$\mathsf{F}\,p, \mathsf{X\,G\,F}\,p, (\mathsf{G\,F}\,\neg p) \mid \ldots \mid \ldots$$
$$(\mathsf{F}\,p), \mathsf{X\,G\,F}\,p, \mathsf{F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid \ldots$$
$$p, \mathsf{X\,G\,F}\,p, (\mathsf{F}\,\neg p), \mathsf{X\,G\,F}\,\neg p \mid \{p\}\,;\,.\mid \ldots \qquad \mathrm{Sub}_1$$
$$p, \neg p, \ldots \qquad \boxed{p, \mathsf{X\,G\,F}\,p, \mathsf{X\,F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \{p\}\,;\,.\mid \ldots} \qquad (\mathsf{X})$$
$$(\mathsf{G\,F}\,p), \mathsf{F}\,\neg p, \mathsf{G\,F}\,\neg p \mid \ldots \mid \ldots$$
$$\mathsf{F}\,p, \mathsf{X\,G\,F}\,p, \mathsf{F}\,\neg p, (\mathsf{G\,F}\,\neg p) \mid \ldots \mid \ldots$$
$$(\mathsf{F}\,p), \mathsf{X\,G\,F}\,p, \mathsf{F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid \ldots$$
$$\mathrm{Sub}_2 \qquad \mathsf{X\,F}\,p, \mathsf{X\,G\,F}\,p, (\mathsf{F}\,\neg p), \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid \ldots$$
$$\boxed{\mathsf{X\,F}\,p, \mathsf{X\,G\,F}\,p, \neg p, \mathsf{X\,G\,F}\,\neg p \mid \{\neg p\}\,;\,.\mid \ldots} \qquad (\mathsf{X}) \qquad \mathrm{Sub}_3$$
$$\mathsf{F}\,p, (\mathsf{G\,F}\,p), \mathsf{G\,F}\,\neg p \mid \ldots \mid \ldots$$
$$\mathsf{F}\,p, \mathsf{X\,G\,F}\,p, (\mathsf{G\,F}\,\neg p) \mid \ldots \mid \ldots$$
$$(\mathsf{F}\,p), \mathsf{X\,G\,F}\,p, \mathsf{F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid \ldots$$
$$p, \mathsf{X\,G\,F}\,p, (\mathsf{F}\,\neg p), \mathsf{X\,G\,F}\,\neg p \mid \{p\}\,;\,.\mid \ldots \qquad \mathrm{Sub}_4$$
$$p, \neg p, \ldots \qquad p, \mathsf{X\,G\,F}\,p, \mathsf{X\,F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid (0, \emptyset) \quad \text{(loop)}$$

[Schwendimann, 1998,
A New One-Pass Tableau Calculus for PLTL]
Tools: **LWB**, **pltl**, **LTL Tableau**,. . .

LTL satisfiability: $\Box\Diamond p \wedge \Box\Diamond\neg p$

$$\boxed{\Box\Diamond p \ \wedge\ \Box\Diamond\neg p} \longrightarrow \left(\ \boxed{\top} \ \right)^{\omega}$$

$(\mathsf{G\,F}\,p \wedge \mathsf{G\,F}\,\neg p) \mid \ldots \mid \ldots$

$(\mathsf{G\,F}\,p), \mathsf{G\,F}\,\neg p \mid \ldots \mid \ldots$

$\mathsf{F}\,p, \mathsf{X\,G\,F}\,p, (\mathsf{G\,F}\,\neg p) \mid \ldots \mid \ldots$

$(\mathsf{F}\,p), \mathsf{X\,G\,F}\,p, \mathsf{F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid \ldots$

$p, \mathsf{X\,G\,F}\,p, (\mathsf{F}\,\neg p), \mathsf{X\,G\,F}\,\neg p \mid \{p\}\,;\,.\mid \ldots$ $\quad \mathrm{Sub}_1$

$p, \neg p, \ldots \qquad p, \mathsf{X\,G\,F}\,p, \mathsf{X\,F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \{p\}\,;\,.\mid \ldots$ $\quad (\mathsf{X})$

$(\mathsf{G\,F}\,p), \mathsf{F}\,\neg p, \mathsf{G\,F}\,\neg p \mid \ldots \mid \ldots$

$\mathsf{F}\,p, \mathsf{X\,G\,F}\,p, \mathsf{F}\,\neg p, (\mathsf{G\,F}\,\neg p) \mid \ldots \mid \ldots$

$(\mathsf{F}\,p), \mathsf{X\,G\,F}\,p, \mathsf{F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid \ldots$

$\mathrm{Sub}_2 \qquad \mathsf{X\,F}\,p, \mathsf{X\,G\,F}\,p, (\mathsf{F}\,\neg p), \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid \ldots$

$\mathsf{X\,F}\,p, \mathsf{X\,G\,F}\,p, \neg p, \mathsf{X\,G\,F}\,\neg p \mid \{\neg p\}\,;\,.\mid \ldots$ $\quad (\mathsf{X}) \quad \mathrm{Sub}_3$

$\mathsf{F}\,p, (\mathsf{G\,F}\,p), \mathsf{G\,F}\,\neg p \mid \ldots \mid \ldots$

$\mathsf{F}\,p, \mathsf{X\,G\,F}\,p, (\mathsf{G\,F}\,\neg p) \mid \ldots \mid \ldots$

$(\mathsf{F}\,p), \mathsf{X\,G\,F}\,p, \mathsf{F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid \ldots$

$p, \mathsf{X\,G\,F}\,p, (\mathsf{F}\,\neg p), \mathsf{X\,G\,F}\,\neg p \mid \{p\}\,;\,.\mid \ldots$ $\quad \mathrm{Sub}_4$

$p, \neg p, \ldots \qquad p, \mathsf{X\,G\,F}\,p, \mathsf{X\,F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid (0, \emptyset)$ $\quad$ (loop)

[Schwendimann, 1998,
A New One-Pass Tableau Calculus for PLTL]

Tools: **LWB**, **pltl**, **LTL Tableau**,. . .

LTL satisfiability: $\Box\Diamond p \;\wedge\; \Box\Diamond\neg p$

$$\boxed{\Box\Diamond p \;\wedge\; \Box\Diamond\neg p} \longrightarrow \left(\;\boxed{\top}\;\right)^{\omega}$$

$(\top)^{\omega}$

$(\mathsf{G\,F}\,p \wedge \mathsf{G\,F}\,\neg p) \mid \ldots \mid \ldots$

$(\mathsf{G\,F}\,p), \mathsf{G\,F}\,\neg p \mid \ldots \mid \ldots$

$\mathsf{F}\,p, \mathsf{X\,G\,F}\,p, (\mathsf{G\,F}\,\neg p) \mid \ldots \mid \ldots$

$(\mathsf{F}\,p), \mathsf{X\,G\,F}\,p, \mathsf{F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid \ldots$

$p, \mathsf{X\,G\,F}\,p, (\mathsf{F}\,\neg p), \mathsf{X\,G\,F}\,\neg p \mid \{p\}\,;\,.\mid \ldots$ $\quad \mathrm{Sub}_1$

$p, \neg p, \ldots \qquad p, \mathsf{X\,G\,F}\,p, \mathsf{X\,F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \{p\}\,;\,.\mid \ldots$

(X)

$(\mathsf{G\,F}\,p), \mathsf{F}\,\neg p, \mathsf{G\,F}\,\neg p \mid \ldots \mid \ldots$

$\mathsf{F}\,p, \mathsf{X\,G\,F}\,p, \mathsf{F}\,\neg p, (\mathsf{G\,F}\,\neg p) \mid \ldots \mid \ldots$

$(\mathsf{F}\,p), \mathsf{X\,G\,F}\,p, \mathsf{F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid \ldots$

$\mathrm{Sub}_2 \qquad \mathsf{X\,F}\,p, \mathsf{X\,G\,F}\,p, (\mathsf{F}\,\neg p), \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid \ldots$

$\mathsf{X\,F}\,p, \mathsf{X\,G\,F}\,p, \neg p, \mathsf{X\,G\,F}\,\neg p \mid \{\neg p\}\,;\,.\mid \ldots$ $\quad \mathrm{Sub}_3$

(X)

$\mathsf{F}\,p, (\mathsf{G\,F}\,p), \mathsf{G\,F}\,\neg p \mid \ldots \mid \ldots$

$\mathsf{F}\,p, \mathsf{X\,G\,F}\,p, (\mathsf{G\,F}\,\neg p) \mid \ldots \mid \ldots$

$(\mathsf{F}\,p), \mathsf{X\,G\,F}\,p, \mathsf{F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid \ldots$

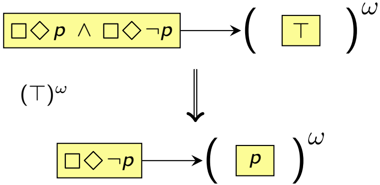
$p, \mathsf{X\,G\,F}\,p, (\mathsf{F}\,\neg p), \mathsf{X\,G\,F}\,\neg p \mid \{p\}\,;\,.\mid \ldots$ $\quad \mathrm{Sub}_4$

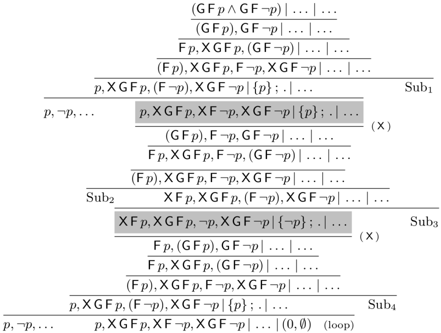
$p, \neg p, \ldots \qquad p, \mathsf{X\,G\,F}\,p, \mathsf{X\,F}\,\neg p, \mathsf{X\,G\,F}\,\neg p \mid \ldots \mid (0, \emptyset)$ $\quad$ (loop)

[Schwendimann, 1998,
A New One-Pass Tableau Calculus for PLTL]

Tools: **LWB**, **pltl**, **LTL Tableau**,...

LTL satisfiability: $\Box\Diamond p \wedge \Box\Diamond\neg p$

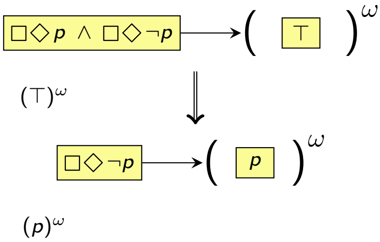


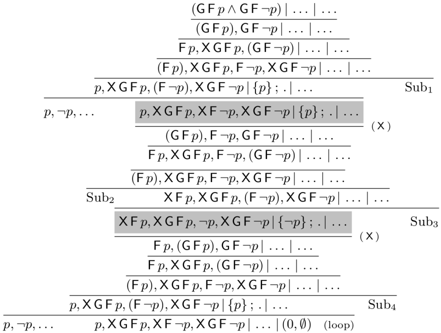


[Schwendimann, 1998,
A New One-Pass Tableau Calculus for PLTL]

Tools: **LWB**, **pltl**, **LTL Tableau**,. . .

LTL satisfiability: $\Box\Diamond p \wedge \Box\Diamond\neg p$

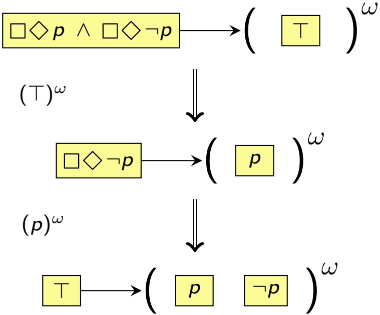


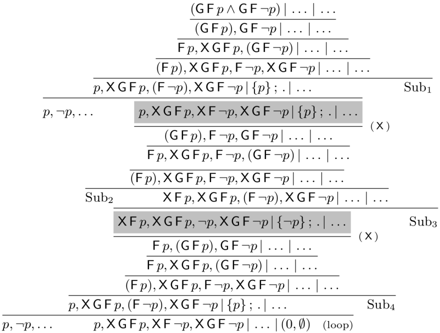


[Schwendimann, 1998,
A New One-Pass Tableau Calculus for PLTL]

Tools: **LWB**, **pltl**, **LTL Tableau**,...

LTL satisfiability: $\square\Diamond p \wedge \square\Diamond\neg p$

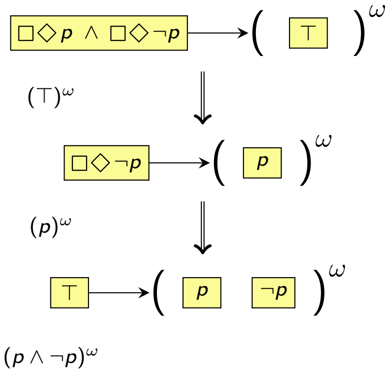


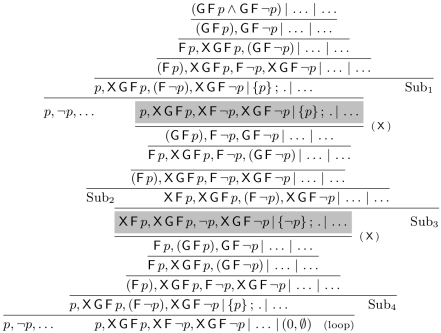


[Schwendimann, 1998,
A New One-Pass Tableau Calculus for PLTL]

Tools: **LWB**, **pltl**, **LTL Tableau**,. . .

LTL satisfiability: $\Box\Diamond p \,\wedge\, \Box\Diamond\neg p$

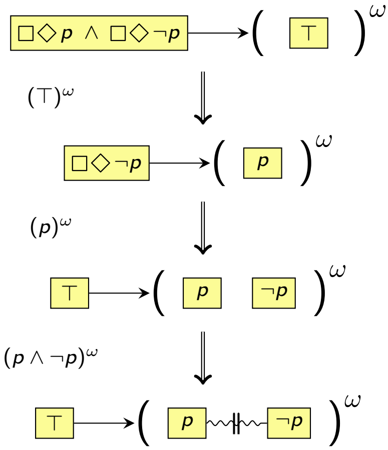


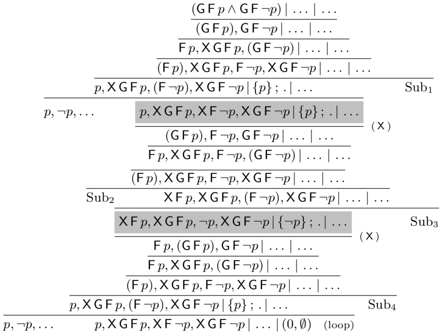


[Schwendimann, 1998,
A New One-Pass Tableau Calculus for PLTL]

Tools: **LWB**, **pltl**, **LTL Tableau**,...

LTL satisfiability: $\Box\Diamond p \wedge \Box\Diamond\neg p$

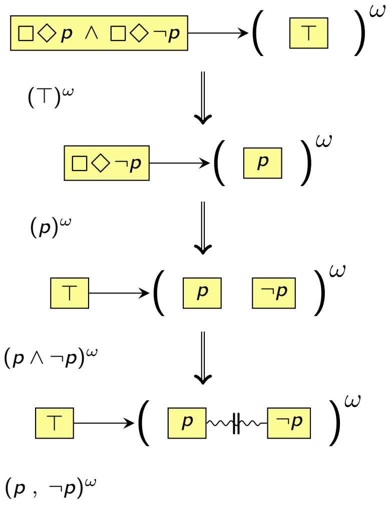


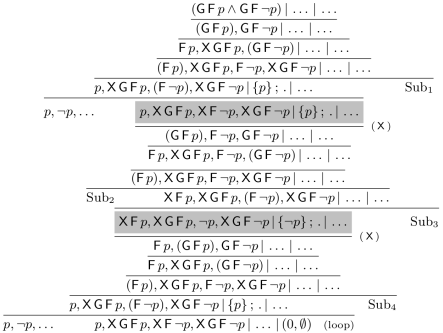


[Schwendimann, 1998,
A New One-Pass Tableau Calculus for PLTL]

Tools: **LWB**, **pltl**, **LTL Tableau**,. . .

LTL satisfiability: $\Box\Diamond p \;\wedge\; \Box\Diamond\neg p$





[Schwendimann, 1998,
A New One-Pass Tableau Calculus for PLTL]

Tools: **LWB**, **pltl**, **LTL Tableau**,. . .

LTL model checking

Automata-based LTL Model Checking

The standard way to model check a program P against an LTL property φ:

1. translate $\neg\varphi$ into a Büchi automaton A
2. check for emptiness the synchronized product of A and P

LTL model checking

Automata-based LTL Model Checking

The standard way to model check a program P against an LTL property φ:

1. translate $\neg\varphi$ into a Büchi automaton A
2. check for emptiness the synchronized product of A and P

Main problem: LTL formulas are often not small!

They describe necessary assumptions of, e.g.:

- fairness
- termination
- allowed request/response pairs

Example: individual accessibility for semaphores

Thread 1

```
while (true) {
  l1:  noncritical;
  l2:  request r;
  l3:  critical;
  l4:  release r;
}
```

Thread 2

```
while (true) {
  m1:  noncritical;
  m2:  request r;
  m3:  critical;
  m4:  release r;
}
```

Thread 3

```
while (true) {
  n1:  noncritical;
  n2:  request r;
  n3:  critical;
  n4:  release r;
}
```

LTL Properties

Fair scheduling: $\varphi_F \equiv \Box\Diamond(at_2 \wedge r_{free}) \implies \Box\Diamond at_3$

Termination of critical sections: $\varphi_T \equiv \Box(at_3 \implies \Diamond at_1)$

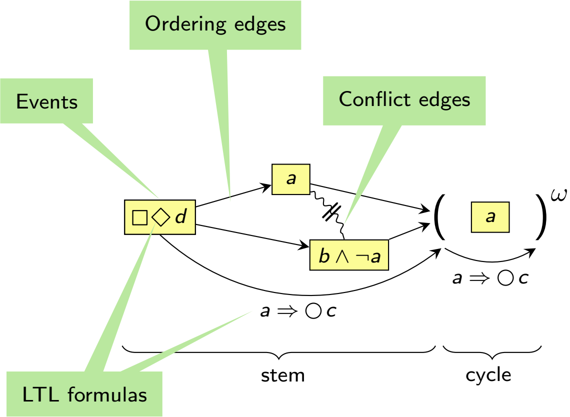
Individual **A**ccessibility: $\varphi_A \equiv \Box(at_2 \implies \Diamond at_3)$

$$\varphi \equiv \bigwedge_{i \in 1..n}(\varphi_{F_i} \wedge \varphi_{T_i}) \implies \varphi_{A_1}$$

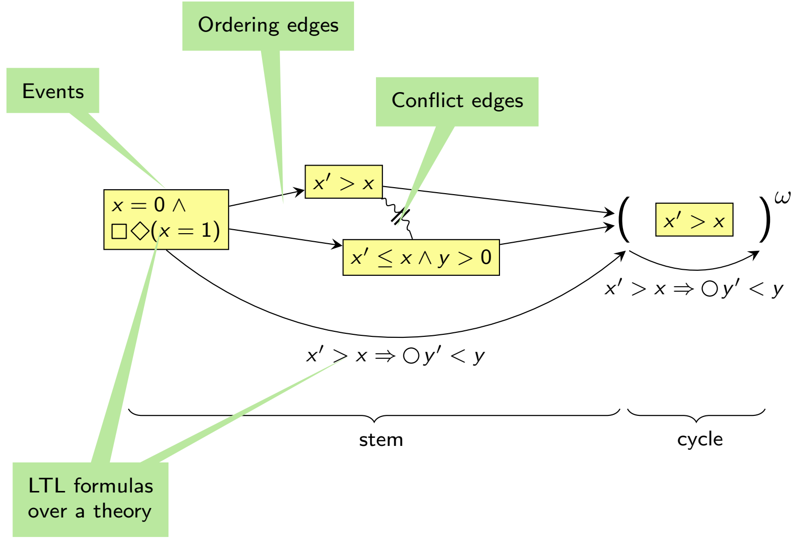
Translation of $\neg\varphi$ into a Büchi automaton: **ltl3ba**

Threads	Time (sec)	Memory (MB)	|Automaton| (MB)
2	0.005	4.2	0.002
3	0.09	5.0	0.38
4	9.6	14.7	8.6
5	1295	139	185
6	TO	X	X

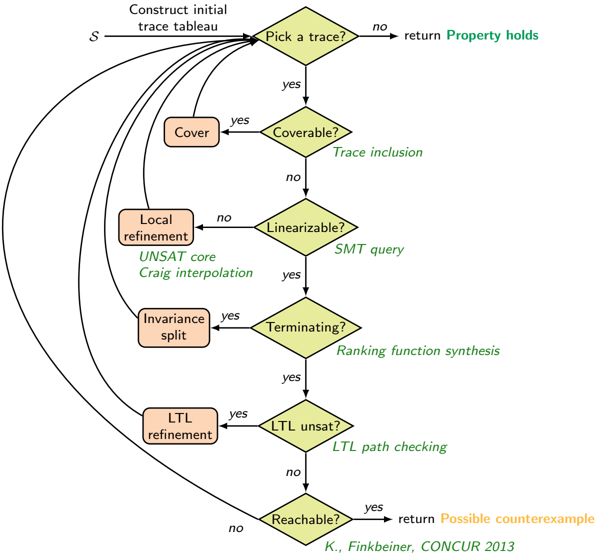
LTL concurrent traces over a theory



LTL concurrent traces over a theory



LTL model checking algorithm



Conclusion

Safety/Reachability

- $\pm$ $\forall\pi.\Box\Phi$ / $\exists\pi.\Diamond\Phi$
- ✓ Infinite-state
- ✓ Multi-threading

$$T_1 \parallel \dots \parallel T_n \models \Box\neg(at_{l_3} \wedge at_{m_3})$$

$\Longrightarrow$

LTL Model Checking

- ✓ Full LTL
- ✓ Infinite-state
- ✓ Multi-threading

$$T_1 \parallel \dots \parallel T_n \models \Box\Diamond(at_{l_2} \wedge r > 0) \implies \Box\Diamond\, at_{l_3}$$

$\Longleftarrow$

Liveness/Termination

- $\pm$ $\forall\pi.\Diamond\Phi$ / $\exists\pi.\Box\Phi$
- ✓ Infinite-state
- ✓ Multi-threading

$$T_1 \parallel \dots \parallel T_n \models at_{l_2} \implies \Diamond\, at_{l_3}$$

$\Uparrow$

LTL Satisfiability/Validity

- ✓ Full LTL
- $\mp$ Boolean
- $-$ No system

$$\models \Diamond p \wedge \Box(p \implies \bigcirc p) \implies \Diamond\Box p$$

Conclusion

Safety/Reachability

$\pm$ $\forall\pi.\Box\Phi$ / $\exists\pi.\Diamond\Phi$
$\checkmark$ Infinite-state
$\checkmark$ Multi-threading

$$T_1 \parallel \ldots \parallel T_n \models \Box\neg(at_{l_3} \wedge at_{m_3})$$

$\Longrightarrow$

LTL Model Checking

$\checkmark$ Full LTL
$\checkmark$ Infinite-state
$\checkmark$ Multi-threading

$$T_1 \parallel \ldots \parallel T_n \models \Box\Diamond(at_{l_2} \wedge r > 0) \implies \Box\Diamond\, at_{l_3}$$

$\Longleftarrow$

Liveness/Termination

$\pm$ $\forall\pi.\Diamond\Phi$ / $\exists\pi.\Box\Phi$
$\checkmark$ Infinite-state
$\checkmark$ Multi-threading

$$T_1 \parallel \ldots \parallel T_n \models at_{l_2} \implies \Diamond\, at_{l_3}$$

$\Uparrow$

LTL Satisfiability/Validity

$\checkmark$ Full LTL
$\mp$ Boolean
$-$ No system

$$\models \Diamond p \wedge \Box(p \implies \bigcirc p) \implies \Diamond\Box p$$

Thank you for attention!