Model checking

parallel system

abstract model $M$

requirements

specification $spec$

model checker

fully automatic verification tool
checks whether $M \ satisifes \ spec$

no + counterexample

yes
Decidability of the model checking problem?
Decidability of the model checking problem?

- Parallel system
- Requirements
- Abstract model $M$
- Specification $spec$

Model checker

Decision algorithm for $M \text{ sat } spec$

No

Yes
Decidability of the model checking problem?

parallel system

abstract model $M$

Turing machine

requirements

specification $spec$

“halt for every input”

model checker

decision algorithm for $M$ sat $spec$

no

yes
General model checking problem is **undecidable**

- Parallel system
- Requirements
- Abstract model $M$
- Turing machine
- Specification $\text{spec}$
- "Halt for every input"

Model checker

Decision algorithm for $M \text{ sat spec}$

Outcome: yes or no
To ensure decidability...

real system

abstract model

requirements

model checker

“does $M \text{ sat spec}$ hold?”

yes

no
To ensure decidability...

real system

requirements

abstraction

semantics

abstract model

specification

model checker

“does $M \text{ sat spec}$ hold?”

yes

no
To ensure decidability . . .

real system

requirements

abstraction semantics

finite transition system

abstract model

model checker

"does $M\sat\spec$ hold?"

yes

no
To ensure decidability . . .

real system

abstraction semantics

abstract model

requirements

finite transition system

specification

finite transition system

model checker

“does $M \text{ sat spec}$ hold ?”

yes

no
To ensure decidability . . .

real system

abstraction semantics

abstract model

finite transition system

requirements

specification

finite transition system

or temporal formula, e.g.,
\( \Box (request \rightarrow \diamond enter\_crit) \)

model checker

"does \( M \) sat spec hold ?"

yes

no
The validation techniques (testing, simulation, deductive verification, model checking) are complementary to each other.
The validation techniques (testing, simulation, deductive verification, model checking) are complementary to each other.

**model checking**

- **most efficient** validation technique, **fully automatic**
- but mostly only applicable for **finite models** with "small" (or "sufficiently structured") state space
- **industrial applications:**
  - hardware systems
  - communication protocols
  - coordination protocols for distributed systems
Historical notes

1976  Keller  transition systems (TS) to model parallel systems

1977  Pnueli  temporal logic to specify parallel systems

1981  Clarke/Emerson  Queille/Sifakis  first model checker
<table>
<thead>
<tr>
<th>Year</th>
<th>Author(s)</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1976</td>
<td>Keller</td>
<td>transition systems (TS) to model parallel systems</td>
</tr>
<tr>
<td>1977</td>
<td>Pnueli</td>
<td>temporal logic $\mathbf{LTL}$ to specify parallel systems</td>
</tr>
<tr>
<td>1981</td>
<td>Clarke/Emerson, Queille/Sifakis</td>
<td>first model checker for $\mathbf{CTL}$</td>
</tr>
<tr>
<td>Year</td>
<td>Authors</td>
<td>Contribution</td>
</tr>
<tr>
<td>-------</td>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1976</td>
<td>Keller</td>
<td>transition systems (TS) to model parallel systems</td>
</tr>
<tr>
<td>1977</td>
<td>Pnueli</td>
<td>temporal logic <strong>LTL</strong> to specify parallel systems</td>
</tr>
<tr>
<td>1981</td>
<td>Clarke/Emerson</td>
<td>first model checker for <strong>CTL</strong></td>
</tr>
<tr>
<td></td>
<td>Queille/Sifakis</td>
<td></td>
</tr>
<tr>
<td>1983</td>
<td>Kanellakis/Smolka</td>
<td>model checking for homogeneous TS-based specifications</td>
</tr>
<tr>
<td>1985</td>
<td>Lichtenstein/Pnueli</td>
<td>model checking for <strong>LTL</strong></td>
</tr>
<tr>
<td>1986</td>
<td>Vardi/Wolper</td>
<td></td>
</tr>
</tbody>
</table>
1976 Keller
1977 Pnueli
1981 Clarke/Emerson
Queille/Sifakis

1985 Lichtenstein/Pnueli
1986 Vardi/Wolper

transition systems temporal logic LTL
first model checker for CTL

state explosion problem
model checking for LTL

state space of industrial systems too large
to be handled by naïve implementations of
model checking algorithms
### Historical notes

<table>
<thead>
<tr>
<th>Year</th>
<th>Name(s)</th>
<th>Transition Systems</th>
<th>Temporal Logic</th>
<th>First Model Checker</th>
<th>Model Checking</th>
</tr>
</thead>
<tbody>
<tr>
<td>1976</td>
<td>Keller</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1977</td>
<td>Pnueli</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1981</td>
<td>Clarke/Emerson</td>
<td></td>
<td>LTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Queille/Sifakis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1985</td>
<td>Lichtenstein/Pnueli</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1986</td>
<td>Vardi/Wolper</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### State explosion problem

- ca. since 1990
- “advanced techniques”
Historical notes

1976 Keller
1977 Pnueli
1981 Clarke/Emerson
  Queille/Sifakis
1985 Lichtenstein/Pnueli
1986 Vardi/Wolper

transition systems temporal logic LTL
first model checker for CTL

model checking for LTL

state explosion problem
ca. since 1990
“advanced techniques”

symbolic model checking with BDDs
partial order reduction
Historical notes

<table>
<thead>
<tr>
<th>Year</th>
<th>Name(s)</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1976</td>
<td>Keller</td>
<td>transition systems</td>
</tr>
<tr>
<td>1977</td>
<td>Pnueli</td>
<td>temporal logic LTL</td>
</tr>
<tr>
<td>1981</td>
<td>Clarke/Emerson</td>
<td>first model checker for CTL</td>
</tr>
<tr>
<td></td>
<td>Queille/Sifakis</td>
<td></td>
</tr>
<tr>
<td>1985</td>
<td>Lichtenstein/Pnueli</td>
<td>model checking for LTL</td>
</tr>
<tr>
<td>1986</td>
<td>Vardi/Wolper</td>
<td></td>
</tr>
</tbody>
</table>

state explosion problem

ca. since 1990

“advanced techniques”

model checking for infinite systems, quantitative analysis, e.g., real-time systems, probabilistic systems
Transition system (TS)

A transition system is a tuple

\[ T = (S, \text{Act}, \rightarrow, S_0, \text{AP}, L) \]

- \( S \) is the state space, i.e., set of states,
- \( \text{Act} \) is a set of actions,
- \( \rightarrow \subseteq S \times \text{Act} \times S \) is the transition relation, i.e., transitions have the form \( s \xrightarrow{\alpha} s' \)
  where \( s, s' \in S \) and \( \alpha \in \text{Act} \)
- \( S_0 \subseteq S \) the set of initial states,
- \( \text{AP} \) a set of atomic propositions,
- \( L : S \rightarrow 2^{\text{AP}} \) the labeling function
possible behaviours of a TS result from:

| select nondeterministically an initial state $s \in S_0$
| WHILE $s$ is non-terminal DO
| select nondeterministically a transition $s \xrightarrow{\alpha} s'$
| execute the action $\alpha$ and put $s := s'$
| OD
“Behaviour” of transition systems

possible behaviours of a TS result from:

select nondeterministically an initial state \( s \in S_0 \)

WHILE \( s \) is non-terminal DO

select nondeterministically a transition \( s \xrightarrow{\alpha} s' \)

execute the action \( \alpha \) and put \( s := s' \)

OD

executions: maximal “transition sequences”

\[ s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots \text{ with } s_0 \in S_0 \]
possible behaviours of a TS result from:

<table>
<thead>
<tr>
<th>select nondeterministically an initial state ( s \in S_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHILE ( s ) is non-terminal DO</td>
</tr>
<tr>
<td>select nondeterministically a transition ( s \xrightarrow{\alpha} s' )</td>
</tr>
<tr>
<td>execute the action ( \alpha ) and put ( s := s' )</td>
</tr>
<tr>
<td>OD</td>
</tr>
</tbody>
</table>

executions: maximal “transition sequences”

\[
s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots \text{ with } s_0 \in S_0
\]

reachable fragment:

\[\text{Reach}(T) = \text{set of all states that are reachable from an initial state through some execution}\]
Linear-time vs branching-time
transition system

\[ T = (S, Act, \rightarrow, S_0, AP, L) \]
Linear-time vs branching-time

transition system

\[ T = (S, \text{Act}, \rightarrow, S_0, AP, L) \]

abstraction from actions

state graph

+ labeling
Linear-time vs branching-time

Transition system

\[ T = (S, Act, \rightarrow, S_0, AP, L) \]

Abstraction from actions

State graph + labeling

Linear-time view

Branching-time view
Linear-time vs branching-time

transition system
\[ T = (S, Act, \rightarrow, S_0, AP, L) \]

abstraction from actions

state graph
+ labeling

linear-time view
path-based
state sequences
branching structure
irrelevant

branching-time view
nondeterministic branches
state & branches
Linear-time vs branching-time

transition system

\[ T = (S, Act, \rightarrow, S_0, AP, L) \]

abstraction from actions

state graph
+ labeling

linear-time view
state sequences

branching-time view
state & branches
Linear-time vs branching-time

Transition system:

\[ T = (S, Act, \rightarrow, S_0, AP, L) \]

Abstraction from actions:

State graph + labeling

Projection on \( AP \):

Linear-time view:

State sequences \( \Downarrow \) traces

Branching-time view:

State & branches \( \Downarrow \) computation tree
Traces

for TS with labeling function $L : S \rightarrow 2^{AP}$

**execution:** states + actions

$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$ infinite or finite

**paths:** sequences of states

$s_0 \ s_1 \ s_2 \ldots$ infinite or $s_0 \ s_1 \ldots \ s_n$ finite
for TS with labeling function $L : S \rightarrow 2^{AP}$

**Execution:** states + actions

$S_0 \xrightarrow{\alpha_1} S_1 \xrightarrow{\alpha_2} S_2 \xrightarrow{\alpha_3} \ldots$ infinite or finite

**Paths:** sequences of states

$S_0 S_1 S_2 \ldots$ infinite or $S_0 S_1 \ldots S_n$ finite

**Traces:** sequences of sets of atomic propositions

$L(S_0) L(S_1) L(S_2) \ldots$
for TS with labeling function $L : S \rightarrow 2^{AP}$

**execution:** states + actions

$$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$$ infinite or finite

**paths:** sequences of states

$$s_0 s_1 s_2 \ldots$$ infinite or $$s_0 s_1 \ldots s_n$$ finite

**traces:** sequences of sets of atomic propositions

$$L(s_0) L(s_1) L(s_2) \ldots \in (2^{AP})^\omega \cup (2^{AP})^+$$
Traces

for TS with labeling function \( L : S \rightarrow 2^{AP} \)

**execution:** states + actions

\[
S_0 \xrightarrow{\alpha_1} S_1 \xrightarrow{\alpha_2} S_2 \xrightarrow{\alpha_3} \ldots \text{ infinite or finite}
\]

**paths:** sequences of states

\[
S_0 \ S_1 \ S_2 \ldots \text{ infinite or } S_0 \ S_1 \ldots \ S_n \text{ finite}
\]

**traces:** sequences of sets of atomic propositions

\[
L(S_0) \ L(S_1) \ L(S_2) \ldots \in (2^{AP})^\omega \cup (2^{AP})^+
\]

for simplicity: we often assume that the given TS has no terminal states
for TS with labeling function $L : S \rightarrow 2^{AP}$

**Execution:** states + actions

$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$ infinite or finite

**Paths:** sequences of states

$s_0 s_1 s_2 \ldots$ infinite or $s_0 s_1 \ldots s_n$ finite

**Traces:** sequences of sets of atomic propositions

$L(s_0) L(s_1) L(s_2) \ldots \in (2^{AP})^\omega \cup (2^{AP})^*$

For simplicity: we often assume that the given TS has no terminal states
Example: traces

Let $T$ be a TS without terminal states.

$Traces(T) \overset{\text{def}}{=} \{ trace(\pi) : \pi \in Paths(T) \} \subseteq (2^{AP})^\omega$

$Traces_{\text{fin}}(T) \overset{\text{def}}{=} \{ trace(\widehat{\pi}) : \widehat{\pi} \in Paths_{\text{fin}}(T) \} \subseteq (2^{AP})^*$

$\{ a \}$

$\emptyset$

TS $T$ with a single atomic proposition $a$

$Traces(T) = \{ \{ a \} \emptyset^\omega, \emptyset^\omega \}$

$Traces_{\text{fin}}(T) = \{ \{ a \} \emptyset^n : n \geq 0 \} \cup \{ \emptyset^m : m \geq 1 \}$
Model checking

system $P_1 \parallel \ldots \parallel P_n$

transition system $\mathcal{T}$

requirements

specification $\text{spec}$

model checker does $\mathcal{T}$ satisfy $\text{spec}$ ?

yes

no + error indication
Model checking

Syntactic description of $P_1 \| \ldots \| P_n$

requirements

specification $spec$

SOS-rules abstraction from actions

state graph of transition system $\mathcal{T}$

model checker does $\mathcal{T}$ satisfy $spec$?

yes

no + error indication
Model checking

- syntactic description of $P_1 \parallel \ldots \parallel P_n$
- requirements
- specification $spec$, e.g., LT property
- model checker
  - does $\mathcal{T}$ satisfy $spec$?
- yes
- no $\pm$ error indication

SOS-rules abstraction from actions

state graph of transition system $\mathcal{T}$
Linear-time properties (LT properties)
Linear-time properties (LT properties)

for TS over $AP$ without terminal states

An LT property over $AP$ is a language $E$ of infinite words over the alphabet $\Sigma = 2^{AP}$, i.e., $E \subseteq (2^{AP})^\omega$. 
Linear-time properties (LT properties)

for TS over $AP$ without terminal states

An LT property over $AP$ is a language $E$ of infinite words over the alphabet $\Sigma = 2^AP$, i.e., $E \subseteq (2^AP)^\omega$.

E.g., for mutual exclusion problems and $AP = \{\text{crit}_1, \text{crit}_2, \ldots\}$

safety:

$MUTEX = \text{set of all infinite words } A_0 A_1 A_2 \ldots \text{ over } 2^AP \text{ such that for all } i \in \mathbb{N}: \text{crit}_1 \not\in A_i \text{ or } \text{crit}_2 \not\in A_i$
An LT property over $AP$ is a language $E$ of infinite words over the alphabet $\Sigma = 2^{AP}$, i.e., $E \subseteq (2^{AP})^\omega$. 
Satisfaction relation for LT properties

An LT property over $AP$ is a language $E$ of infinite words over the alphabet $\Sigma = 2^AP$, i.e., $E \subseteq (2^AP)^\omega$.

Satisfaction relation $\models$ for TS:

If $\mathcal{T}$ is a TS (without terminal states) over $AP$ and $E$ an LT property over $AP$ then

$\mathcal{T} \models E \text{ iff } \text{Traces}(\mathcal{T}) \subseteq E$
Satisfaction relation for LT properties

An LT property over $AP$ is a language $E$ of infinite words over the alphabet $\Sigma = 2^{AP}$, i.e., $E \subseteq (2^{AP})^\omega$.

Satisfaction relation $\models$ for TS and states:

If $\mathcal{T}$ is a TS (without terminal states) over $AP$ and $E$ an LT property over $AP$ then

$\mathcal{T} \models E \iff \text{Traces}(\mathcal{T}) \subseteq E$

If $s$ is a state in $\mathcal{T}$ then

$s \models E \iff \text{Traces}(s) \subseteq E$
LT properties and trace inclusion

An LT property over $AP$ is a language $E$ of infinite words over the alphabet $\Sigma = 2^{AP}$, i.e., $E \subseteq (2^{AP})^\omega$.

If $T$ is a TS over $AP$ then $T \models E$ iff $\text{Traces}(T) \subseteq E$.

If $T_1$ and $T_2$ are TS over $AP$ then the following statements are equivalent:

1. $\text{Traces}(T_1) \subseteq \text{Traces}(T_2)$

2. for all LT-properties $E$ over $AP$: whenever $T_2 \models E$ then $T_1 \models E$

(1) $\implies$ (2): $\sqrt{\top}$
An LT property over $\mathcal{AP}$ is a language $E$ of infinite words over the alphabet $\Sigma = 2^{\mathcal{AP}}$, i.e., $E \subseteq (2^{\mathcal{AP}})^\omega$.

If $\mathcal{T}$ is a TS over $\mathcal{AP}$ then $\mathcal{T} \models E$ iff $\text{Traces}(\mathcal{T}) \subseteq E$.

If $\mathcal{T}_1$ and $\mathcal{T}_2$ are TS over $\mathcal{AP}$ then the following statements are equivalent:

(1) $\text{Traces}(\mathcal{T}_1) \subseteq \text{Traces}(\mathcal{T}_2)$

(2) for all LT-properties $E$ over $\mathcal{AP}$: whenever $\mathcal{T}_2 \models E$ then $\mathcal{T}_1 \models E$

$(2) \implies (1)$: consider $E = \text{Traces}(\mathcal{T}_2)$
Trace equivalence
Trace equivalence

Transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$ over the same set $AP$ of atomic propositions are called trace equivalent iff

$$\text{Traces}(\mathcal{T}_1) = \text{Traces}(\mathcal{T}_2)$$
Trace equivalence

Transition systems $T_1$ and $T_2$ over the same set $AP$ of atomic propositions are called trace equivalent iff

$$\text{Traces}(T_1) = \text{Traces}(T_2)$$

i.e., trace equivalence requires trace inclusion in both directions
Trace equivalence

Transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$ over the same set $AP$ of atomic propositions are called trace equivalent iff

$$\text{Traces}(\mathcal{T}_1) = \text{Traces}(\mathcal{T}_2)$$

i.e., trace equivalence requires trace inclusion in both directions

Trace equivalent TS satisfy the same LT properties
LT properties and trace relations

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TS over $AP$.

The following statements are equivalent:

1. $Traces(\mathcal{T}_1) \subseteq Traces(\mathcal{T}_2)$
2. for all LT-properties $E$: $\mathcal{T}_2 \models E \implies \mathcal{T}_1 \models E$

The following statements are equivalent:

1. $Traces(\mathcal{T}_1) = Traces(\mathcal{T}_2)$
2. for all LT-properties $E$: $\mathcal{T}_1 \models E$ iff $\mathcal{T}_2 \models E$
Linear Temporal Logic (LTL)

\[ \varphi ::= \text{true} \mid a \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \mathsf{U} \varphi_2 \]

where \( a \in \mathit{AP} \)

atomic proposition \( a \in \mathit{AP} \)

next operator \( \bigcirc a \)

until operator \( a \mathsf{U} b \)
Derived operators in LTL

\[ \varphi ::= \text{true} \mid a \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \mathbin{\mathcal{U}} \varphi_2 \]

derived operators:

\( \lor, \rightarrow, \ldots \) as usual
Derived operators in LTL

\[ \varphi ::= \text{true} \mid a \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \Box \varphi \mid \varphi_1 \mathbf{U} \varphi_2 \]

derived operators:

\[ \Diamond \varphi \overset{\text{def}}{=} \text{true} \mathbf{U} \varphi \text{ eventually} \]

V, →, ... as usual
Derived operators in LTL

\[ \varphi ::= \text{true} \mid a \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \Box \varphi \mid \varphi_1 \mathcal{U} \varphi_2 \]

derived operators:

\[ \diamond \varphi \quad \text{def} \quad \text{true} \mathcal{U} \varphi \quad \text{eventually} \]

V, \rightarrow, \ldots \text{ as usual}

until operator \[ a \mathcal{U} b \]

eventually \[ \diamond b \]
Derived operators in LTL

\[ \varphi ::= \text{true} \mid a \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \square \varphi \mid \varphi_1 \mathsf{U} \varphi_2 \]

derived operators:

\[ \Diamond \varphi \overset{\text{def}}{=} \text{true} \mathsf{U} \varphi \quad \text{eventually} \]

\[ \Box \varphi \overset{\text{def}}{=} \neg \Diamond \neg \varphi \quad \text{always} \]

until operator \(a \mathsf{U} b\)

\[ a \quad a \quad a \quad b \]

eventually \(\Diamond b\)

\[ b \]

always \(\Box a\)

\[ a \quad a \quad a \quad a \quad a \quad a \quad a \quad a \]
Next $\bigcirc$, until U and eventually $\Diamond$

□ (try_to_send $\rightarrow$ $\bigcirc$ delivered)

... try del ...

LTLsf3.1-3
Next ◦, until U and eventually ♦

□ (try_to_send → ◦ delivered)

□ (try_to_send → try_to_send U delivered)
Next $\bigcirc$, until $U$ and eventually $\Diamond$

□ $(\text{try\_to\_send} \rightarrow \bigcirc \text{ delivered})$

... try del ...

□ $(\text{try\_to\_send} \rightarrow \text{try\_to\_send} \ U \text{ delivered})$

... try try try del ...

□ $(\text{try\_to\_send} \rightarrow \Diamond \text{ delivered})$

... try del ...

LTLFSF3.1-3
### Infinitely often and eventually forever

$$\varphi ::= \text{true} \mid a \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \Box \varphi \mid \varphi_1 \mathbf{U} \varphi_2$$

<table>
<thead>
<tr>
<th>Statement</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventually $\Diamond \varphi$</td>
<td>$\text{true U } \varphi$</td>
</tr>
<tr>
<td>always $\Box \varphi$</td>
<td>$\neg \Diamond \neg \varphi$</td>
</tr>
<tr>
<td>infinitely often $\Box \Diamond \varphi$</td>
<td></td>
</tr>
<tr>
<td>eventually forever $\Diamond \Box \varphi$</td>
<td></td>
</tr>
</tbody>
</table>

**Examples:**
- **Unconditional fairness**
  $$\Box \Diamond \text{crit}_i$$
- **Strong fairness**
  $$\Box \Diamond \text{wait}_i \rightarrow \Box \Diamond \text{crit}_i$$
- **Weak fairness**
  $$\Diamond \Box \text{wait}_i \rightarrow \Box \Diamond \text{crit}_i$$
interpretation of LTL formulas over traces, i.e., infinite words over \(2^{AP}\)
interpretation of LTL formulas over traces, i.e., infinite words over $2^{AP}$

formalized by a satisfaction relation $\models$ for

- LTL formulas and
- infinite words $\sigma = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega$
Semantics of LTL over infinite words

for $\sigma = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega$:

- $\sigma \models true$
- $\sigma \models a$ iff $A_0 \models a$, i.e., $a \in A_0$
- $\sigma \models \varphi_1 \land \varphi_2$ iff $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$
- $\sigma \models \neg \varphi$ iff $\sigma \not\models \varphi$
Semantics of LTL over infinite words

for \( \sigma = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \):

\[
\begin{align*}
\sigma & \models true \\
\sigma & \models a \quad \text{iff} \quad A_0 \models a, \text{i.e.,} \ a \in A_0 \\
\sigma & \models \varphi_1 \land \varphi_2 \quad \text{iff} \quad \sigma \models \varphi_1 \text{ and } \sigma \models \varphi_2 \\
\sigma & \models \neg \varphi \quad \text{iff} \quad \sigma \not\models \varphi \\
\sigma & \models \Diamond \varphi \quad \text{iff} \quad \text{suffix}(\sigma, 1) = A_1 A_2 A_3 \ldots \models \varphi
\end{align*}
\]
Semantics of LTL over infinite words

for $\sigma = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega$:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma \models true$</td>
<td>( A_0 \models a ), i.e., ( a \in A_0 )</td>
</tr>
<tr>
<td>$\sigma \models a$</td>
<td>$\sigma \models \varphi_1$ and $\sigma \models \varphi_2$</td>
</tr>
<tr>
<td>$\sigma \models \neg \varphi$</td>
<td>$\sigma \not\models \varphi$</td>
</tr>
<tr>
<td>$\sigma \models \Box \varphi$</td>
<td>$\text{suffix}(\sigma, 1) = A_1 A_2 A_3 \ldots \models \varphi$</td>
</tr>
<tr>
<td>$\sigma \models \varphi_1 U \varphi_2$</td>
<td>there exists $j \geq 0$ such that $\text{suffix}(\sigma, j) = A_j A_{j+1} A_{j+2} \ldots \models \varphi_2$ and $\text{suffix}(\sigma, i) = A_i A_{i+1} A_{i+2} \ldots \models \varphi_1$ for $0 \leq i &lt; j$</td>
</tr>
</tbody>
</table>
LTL semantics over paths of TS
LTL semantics over paths of TS

given: \( TS = (S, Act, \rightarrow, S_0, AP, L) \)

without terminal states

LTL formula \( \varphi \) over \( AP \)
LTL semantics over paths of TS

given: \( \mathcal{T} = (S, Act, \rightarrow, S_0, AP, L) \)
without terminal states
LTL formula \( \varphi \) over \( AP \)

interpretation of \( \varphi \) over infinite path fragments

\[ \pi = s_0 s_1 s_2 \ldots \models \varphi \text{ iff } \text{trace}(\pi) \models \varphi \]
LTL semantics over paths of TS

given: \( \mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L) \)
without terminal states

LTL formula \( \varphi \) over \( AP \)

interpretation of \( \varphi \) over infinite path fragments

\[
\pi = s_0 s_1 s_2 \ldots \models \varphi \text{ iff } \text{trace}(\pi) \models \varphi \text{ iff } \text{trace}(\pi) \in \text{Words}(\varphi)
\]
**LTL semantics over paths of TS**

*given:* \( \mathcal{T} = (S, Act, \rightarrow, S_0, AP, L) \)
without terminal states

LTL formula \( \varphi \) over \( AP \)

interpretation of \( \varphi \) over infinite path fragments

\[
\pi = s_0 s_1 s_2 \ldots \models \varphi \quad \text{iff} \quad \text{trace}(\pi) \models \varphi
\]

\[
\text{iff} \quad \text{trace}(\pi) \in \text{Words}(\varphi)
\]

*remind:* LT property of an LTL formula:

\[
\text{Words}(\varphi) = \{ \sigma \in (2^{AP})^\omega : \sigma \models \varphi \}
\]
LTL semantics over the states of a TS
LTL semantics over the states of a TS

given: \( T = (S, Act, \rightarrow, S_0, AP, L) \)
without terminal states
LTL formula \( \varphi \) over \( AP \)

interpretation of \( \varphi \) over infinite path fragments

\[ \pi = s_0 s_1 s_2 \ldots \models \varphi \quad \text{iff} \quad \text{trace}(\pi) \models \varphi \]

interpretation of \( \varphi \) over states:

\[ s \models \varphi \quad \text{iff} \quad \text{trace}(\pi) \models \varphi \quad \text{for all} \ \pi \in \text{Paths}(s) \]
LTL semantics over the states of a TS

given: \( \mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L) \)
without terminal states

LTL formula \( \varphi \) over \( AP \)

interpretation of \( \varphi \) over infinite path fragments

\[ \pi = s_0 s_1 s_2 \ldots \models \varphi \quad \text{iff} \quad \text{trace}(\pi) \models \varphi \]

interpretation of \( \varphi \) over states:

\[ s \models \varphi \quad \text{iff} \quad \text{trace}(\pi) \models \varphi \quad \text{for all} \quad \pi \in \text{Paths}(s) \]

\[ \text{iff} \quad s \models \text{Words}(\varphi) \]
LTL semantics over the states of a TS

**given:** TS $\mathcal{I} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states

LTL formula $\varphi$ over $AP$

interpretation of $\varphi$ over infinite path fragments

$$\pi = s_0 s_1 s_2 \ldots \models \varphi \iff \text{trace}(\pi) \models \varphi$$

interpretation of $\varphi$ over states:

$$s \models \varphi \iff \text{trace}(\pi) \models \varphi \text{ for all } \pi \in \text{Paths}(s)$$

$$\iff s \models \text{Words}(\varphi)$$

satisfaction relation for LT properties
LTL semantics over the states of a TS

given:  TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$
without terminal states
LTL formula $\varphi$ over $AP$

interpretation of $\varphi$ over infinite path fragments

$$\pi = s_0 s_1 s_2 \ldots \models \varphi \iff \text{trace}(\pi) \models \varphi$$

interpretation of $\varphi$ over states:

$\models \varphi \iff \text{trace}(\pi) \models \varphi$ for all $\pi \in \text{Paths}(s)$

iff $s \models \text{Words}(\varphi)$

iff $\text{Traces}(s) \subseteq \text{Words}(\varphi)$
Interpretation of LTL formulas over TS
Interpretation of LTL formulas over TS

Given: TS $\mathcal{I} = (S, \text{Act}, \rightarrow, S_0, AP, L)$ without terminal states

LTL formula $\varphi$ over $AP$

$\mathcal{I} \models \varphi$ iff $s_0 \models \varphi$ for all $s_0 \in S_0$
Interpretation of LTL formulas over TS

given: \( \mathcal{T} = (S, Act, \rightarrow, S_0, AP, L) \)

without terminal states

LTL formula \( \varphi \) over \( AP \)

\[ \mathcal{T} \models \varphi \quad \text{iff} \quad s_0 \models \varphi \quad \text{for all} \quad s_0 \in S_0 \]

\[ \text{iff} \quad \text{trace}(\pi) \models \varphi \quad \text{for all} \quad \pi \in \text{Paths}(\mathcal{T}) \]
Interpretation of LTL formulas over TS

given:  TS \( \mathcal{T} = (S, \text{Act}, \rightarrow, S_0, AP, L) \)
without terminal states
LTL formula \( \varphi \) over \( AP \)

\[ \mathcal{T} \models \varphi \iff s_0 \models \varphi \text{ for all } s_0 \in S_0 \]
iff \( \text{trace}(\pi) \models \varphi \text{ for all } \pi \in \text{Paths}(\mathcal{T}) \)
iff \( \text{Traces}(\mathcal{T}) \subseteq \text{Words}(\varphi) \)
Interpretation of LTL formulas over TS

given: \( \mathcal{I} = (S, Act, \rightarrow, S_0, AP, L) \)
without terminal states

LTL formula \( \varphi \) over \( AP \)

\[ \mathcal{I} \models \varphi \quad \text{iff} \quad s_0 \models \varphi \quad \text{for all} \quad s_0 \in S_0 \]

\[ \text{iff} \quad \text{trace}(\pi) \models \varphi \quad \text{for all} \quad \pi \in \text{Paths}(\mathcal{I}) \]

\[ \text{iff} \quad \text{Traces}(\mathcal{I}) \subseteq \text{Words}(\varphi) \]

\[ \text{iff} \quad \mathcal{I} \models \text{Words}(\varphi) \]
Interpretation of LTL formulas over TS

given: \( \mathcal{T} = (S, \text{Act}, \rightarrow, S_0, \text{AP}, L) \)
without terminal states

LTL formula \( \varphi \) over \( \text{AP} \)

\( \mathcal{T} \models \varphi \) iff \( s_0 \models \varphi \) for all \( s_0 \in S_0 \)

iff \( \text{trace}(\pi) \models \varphi \) for all \( \pi \in \text{Paths}(\mathcal{T}) \)

iff \( \text{Traces}(\mathcal{T}) \subseteq \text{Words}(\varphi) \)

iff \( \mathcal{T} \models \text{Words}(\varphi) \)

satisfaction relation for LT properties
Linear-time implementation relations
finite trace inclusion and equivalence:
  e.g., $Traces_{\text{fin}}(T_1) \subseteq Traces_{\text{fin}}(T_2)$

trace inclusion and trace equivalence:
  e.g., $Traces(T_1) \subseteq Traces(T_2)$
Linear-time implementation relations

finite trace inclusion and equivalence:

\[ \text{Traces}_{\text{fin}}(T_1) \subseteq \text{Traces}_{\text{fin}}(T_2) \]

preserves all linear-time safety properties

trace inclusion and trace equivalence:

\[ \text{Traces}(T_1) \subseteq \text{Traces}(T_2) \]
Linear-time implementation relations

finite trace inclusion and equivalence:
e.g., $\text{Traces}_{\text{fin}}(\mathcal{I}_1) \subseteq \text{Traces}_{\text{fin}}(\mathcal{I}_2)$

preserves all linear-time safety properties

trace inclusion and trace equivalence:
e.g., $\text{Traces}(\mathcal{I}_1) \subseteq \text{Traces}(\mathcal{I}_2)$

preserves all LTL properties
Linear-time implementation relations

finite trace inclusion and equivalence:
  e.g., \( \text{Traces}_{\text{fin}}(T_1) \subseteq \text{Traces}_{\text{fin}}(T_2) \)

  preserves all linear-time safety properties

trace inclusion and trace equivalence:
  e.g., \( \text{Traces}(T_1) \subseteq \text{Traces}(T_2) \)

  preserves all LTL properties

* none of the LT relations is compatible with CTL
Linear-time implementation relations

finite trace inclusion and equivalence:
  e.g., $\text{Traces}_{\text{fin}}(T_1) \subseteq \text{Traces}_{\text{fin}}(T_2)$
  preserves all linear-time safety properties

trace inclusion and trace equivalence:
  e.g., $\text{Traces}(T_1) \subseteq \text{Traces}(T_2)$
  preserves all LTL properties

* none of the LT relations is compatible with CTL
* checking LT relations is computationally hard
Linear-time implementation relations

finite trace inclusion and equivalence:
  e.g., $\text{Traces}_{\text{fin}}(T_1) \subseteq \text{Traces}_{\text{fin}}(T_2)$
  preserves all linear-time safety properties

trace inclusion and trace equivalence:
  e.g., $\text{Traces}(T_1) \subseteq \text{Traces}(T_2)$
  preserves all LTL properties

* none of the LT relations is compatible with CTL
* checking LT relations is computationally hard
* minimization ???
Minimization w.r.t. trace equivalence?

$\mathcal{T}_1$: 

$\mathcal{T}_2$: 

- $\text{Traces}(\mathcal{T}_1) = \text{Traces}(\mathcal{T}_2)$

but $\mathcal{T}_1$ and $\mathcal{T}_2$ are not isomorphic

- $\mathcal{T}_1$, $\mathcal{T}_2$ have 5 states and 7 transitions each

- there is no smaller TS that is trace-equivalent to $\mathcal{T}_i$
Classification of implementation relations

• linear vs. branching time
  * linear time: trace relations
  * branching time: (bi)simulation relations

• (nonsymmetric) preorders vs. equivalences:
  * preorders: trace inclusion, simulation
  * equivalences: trace equivalence, bisimulation

• strong vs. weak relations
  * strong: reasoning about all transitions
  * weak: abstraction from stutter steps
Summary: equivalences

- Bisimulation equivalence
- LTL equivalence
- CTL equivalence
- CTL* equivalence

for finitely branching TS
Summary: equivalences

- Trace equivalence
- LTL equivalence
- Bisimulation equivalence
- CTL equivalence
- CTL* equivalence

for finitely branching TS
Summary: equivalences

- Finite trace equivalence
- Trace equivalence
- Bisimulation equivalence
- CTL equivalence
- CTL* equivalence
- LTL equivalence

For finitely branching TS
Summary: equivalences

finite trace equivalence

trace equivalence

bisimulation equivalence

LTL equivalence

LTL safety properties

CTL equivalence

CTL* equivalence

for finitely branching TS