Game-Theoretic Approach to Planning and Synthesis

Linear-Time Temporal Logic [1]

Giuseppe De Giacomo Antonio Di Stasio Giuseppe Perelli Shufang Zhu

ERC Advanced Grant
WhiteMech:
White-box Self Programming Mechanisms

PhD-AI Course
4-8 July, 2022

[1] Thanks to Alessandro Abate and Tom Melham (University of Oxford) for providing help with part of these slides.

SAPIENZA
UNIVERSITÀ DI ROMA

- given a system (e.g., a physical device, a hardware circuit, or software code), a model represents an abstraction of it
- models are *contextual*
- many *levels of abstraction* are possible: many models of a system can be made
- abstraction/refinement (choice of right model) ought to be part of the *modelling effort*
- formal verification asserts properties of a *model*, not of the underlying *system*

SAPIENZA
UNIVERSITÀ DI ROMA

Definition

A *transition system* is a tuple $\langle S, \rightarrow, I, \mathrm{Prop}, L\rangle$ consisting of

- S set of *states*,
- $\rightarrow \subseteq S \times S$ *transition relation*,
- $I \subseteq S$ set of *initial states*,
- Prop set of *atomic propositions* (alphabet), and
- $L : S \rightarrow 2^{\mathrm{Prop}}$ *labelling function*.

A TS is also known as *Kripke structure*.

Clarke, Grumberg, Peled - Model Checking. - MIT Press 1999

SAPIENZA
UNIVERSITÀ DI ROMA

Definition

A *transition system* is a tuple $\langle S, Act, \rightarrow, I, \mathrm{Prop}, L\rangle$ consisting of

- S set of *states*,
- Act set of *actions*,
- $\rightarrow \subseteq S \times Act \times S$ *transition relation*,
- $I \subseteq S$ set of *initial states*,
- Prop set of *atomic propositions* (alphabet), and
- $L : S \rightarrow 2^{\mathrm{Prop}}$ *labelling function*.

- different modelling choice
- *action enabled* TS are closely related to Moore machines
- we will use action enabled TS only when strictly needed

Baier, Katoen - Principles of Model Checking. - MIT Press 2008

Transition relation

- We also write $s \to s'$ instead of $(s, s') \in \to$.
- We consider exclusively transition relations such that each state has an outgoing transition, that is

$$\forall s \in S : \exists s' \in S : s \to s'$$

  known as *non-blocking* condition, as absence of *terminal* states.
- In this course we consider *finite* TS, namely S has finite cardinality.

System description: A traffic light can be red, green, amber or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.

Example: model for a traffic light

System description: A traffic light can be red, green, amber or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.

how many different states do we need to model the traffic light?

System description: A traffic light can be red, green, amber or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.

$$S = \{1, 2, 3, 4, 5\}$$

1	red
2	amber and red
3	green
4	amber
5	black

SAPIENZA
UNIVERSITÀ DI ROMA

System description: A traffic light can be red, green, amber or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.

$$S = \{1, 2, 3, 4, 5\}$$

1	red
2	amber and red
3	green
4	amber
5	black

what are the transitions?

Example: model of a traffic light

System description: A traffic light can be red, green, amber or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.

$$S = \{1, 2, 3, 4, 5\}$$

1	red
2	amber and red
3	green
4	amber
5	black

$$\rightarrow = \{(1, 2), (2, 3), (3, 4), (4, 1), (1, 5), (2, 5), (3, 5), (4, 5), (5, 1)\}$$

Example: model of a traffic light

System description: A traffic light can be red, green, amber or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.

$$S = \{1, 2, 3, 4, 5\}$$

1	red
2	amber and red
3	green
4	amber
5	black

$$\rightarrow = \{(1, 2), (2, 3), (3, 4), (4, 1), (1, 5), (2, 5), (3, 5), (4, 5), (5, 1)\}$$

what are the initial states?

System description: A traffic light can be red, green, amber or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.

$$
\begin{aligned}
S &= \{1,2,3,4,5\} \\
\rightarrow &= \{(1,2),(2,3),(3,4),(4,1),(1,5),(2,5),(3,5),(4,5),(5,1)\} \\
I &= \{1\} \\
\mathrm{Prop} &= \{r,a,g,b\}
\end{aligned}
$$

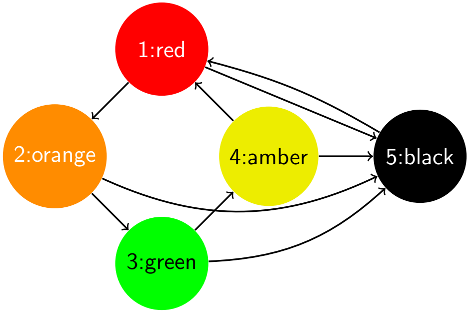
how is the labelling function defined?

SAPIENZA
UNIVERSITÀ DI ROMA

System description: A traffic light can be red, green, amber or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.

$$
\begin{aligned}
S &= \{1,2,3,4,5\} \\
\rightarrow &= \{(1,2),(2,3),(3,4),(4,1),(1,5),(2,5),(3,5),(4,5),(5,1)\} \\
I &= \{1\} \\
\mathrm{Prop} &= \{r,a,g,b\} \\
L &= \{1 \mapsto \{r\}, 2 \mapsto \{r,a\}, 3 \mapsto \{g\}, 4 \mapsto \{a\}, 5 \mapsto \{b\}\}
\end{aligned}
$$

System description: A traffic light can be red, green, amber or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red.



Definition

Consider the transition system $\langle S, \rightarrow, I, \mathrm{Prop}, L\rangle$

- A finite path is a finite state sequence $s_0 s_1 \dots s_n$ for some $n \geq 0$ such that $s_i \rightarrow s_{i+1}$ for all $0 \leq i < n$
- An infinite path is an infinite state sequence $s_0 s_1 \dots$ such that $s_i \rightarrow s_{i+1}$ for all $i \geq 0$
- A path is initial if $s_0 \in I$
- A path is an initial infinite path, contained in $Paths(TS)$
- A trace is the “output” of a path: $L(s_0)L(s_1)\dots$

- let $\pi = s_0 s_1 \ldots$ be an infinite path (applies to finite paths too)
- for $j \geq 0$, the jth state of π, s_j is denoted by $\pi[j]$ (initial state is indexed by 0)
- for $j \geq 0$, the jth prefix of π, $s_0 s_1 \ldots s_j$ is denoted by $\pi[..j]$

$$\overbrace{s_0 s_1 \ldots s_{j-1} s_j}^{\pi[..j]} s_{j+1} \ldots$$

- for $j \geq 0$, the jth suffix of π, $s_j s_{j+1} \ldots$ is denoted by $\pi[j..]$

$$s_0 s_1 \ldots s_{j-1} \overbrace{s_j s_{j+1} \ldots}^{\pi[j..]}$$

- the set of infinite path π with $\pi[0] = s$ is denoted by $\mathit{Paths}(s)$

A standard language for talking about infinite state sequences.

Amir Pnueli - The Temporal Logic of Programs. - FOCS'77

$\top$	truth constant
p	primitive propositions
$\neg\phi$	classical negation
$\phi \vee \psi$	classical disjunction
$\phi \wedge \psi$	classical conjunction

SAPIENZA
UNIVERSITÀ DI ROMA

A standard language for talking about infinite state sequences.

Amir Pnueli - The Temporal Logic of Programs. - FOCS'77

$\top$	truth constant	$\bigcirc\phi$	in the next state...
p	primitive propositions	$\Diamond\phi$	will eventually be the case
$\neg\phi$	classical negation	$\Box\phi$	is always the case
$\phi \vee \psi$	classical disjunction	$\phi \mathsf{U} \psi$	ϕ until ψ
$\phi \wedge \psi$	classical conjunction	$\phi \mathsf{R} \psi$	ϕ release ψ

SAPIENZA
UNIVERSITÀ DI ROMA

A standard language for talking about infinite state sequences.

Amir Pnueli - The Temporal Logic of Programs. - FOCS'77

$\top$	truth constant	$\bigcirc\phi$	in the next state...
p	primitive propositions	$\Diamond\phi$	will eventually be the case
$\neg\phi$	classical negation	$\Box\phi$	is always the case
$\phi \vee \psi$	classical disjunction	$\phi \mathsf{U} \psi$	ϕ until ψ
$\phi \wedge \psi$	classical conjunction	$\phi \mathsf{R} \psi$	ϕ release ψ

Minimal syntax

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \mathsf{U} \varphi$$

you may encounter the following notations:

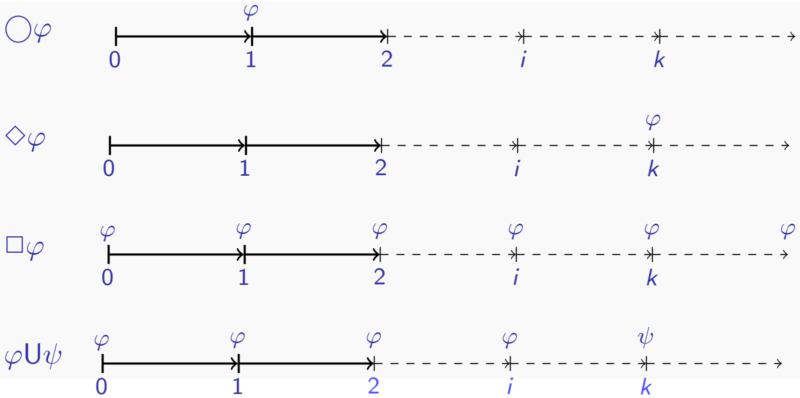
$$
\begin{array}{ccc}
\mathsf{X}\varphi & : & \bigcirc\varphi \\
\mathsf{F}\varphi & : & \Diamond\varphi \\
\mathsf{G}\varphi & : & \Box\varphi
\end{array}
$$

past operators are possible (though not strictly necessary)

Example LTL formulae

Eventually I will graduate	$\Diamond \mathtt{degree}$
The plane will never crash	$\Box \neg \mathtt{crash}$
I will eat pizza infinitely often	$\Box \Diamond \mathtt{eatPizza}$
... and they all lived happily ever after	$\Diamond \Box \mathtt{happy}$
We are not friends until you apologise	$(\neg \mathtt{friends}) \mathcal{U} \mathtt{youApologise}$
Every time it is requested, a document will be printed	$\Box(\mathtt{print\_req} \to \Diamond \mathtt{print})$
The two processes are never active at the same time	$\Box \neg (\mathtt{proc}_1 \wedge \mathtt{proc}_2)$

Semantics of LTL



LTL formulas are evaluated on infinite traces, that is, obtained from an infinite path.

The language defined by an LTL formula φ is $\mathcal{L}(\varphi) = \{w \in \Sigma^\omega : w \models \varphi\}$.

SAPIENZA
UNIVERSITÀ DI ROMA

how to express
“the light is infinitely often red”
by an LTL formula?

how to express
"the light is infinitely often red"
by an LTL formula?

$\Box\Diamond \mathit{red}$

how to express
"the light is infinitely often red"
by an LTL formula?

$\Box\Diamond red$

how to express
"once green, the light cannot become immediately red"
by an LTL formula?

how to express
"the light is infinitely often red"
by an LTL formula? $\Box\Diamond \mathit{red}$

how to express
"once green, the light cannot become immediately red"
by an LTL formula? $\Box(\mathit{green} \rightarrow \neg \bigcirc \mathit{red})$

SAPIENZA UNIVERSITÀ DI ROMA

Question: How do you express in LTL
"once green, the light cannot become red immediately after"?

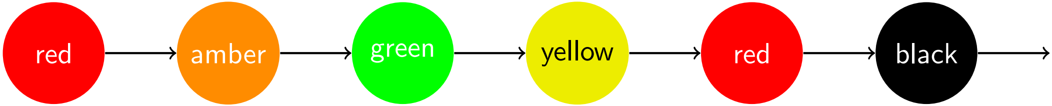
SAPIENZA UNIVERSITÀ DI ROMA

Question: How do you express in LTL
"once green, the light cannot become red immediately after"?

Answer: $\Box(\text{green} \rightarrow \neg \bigcirc \text{red})$.

Verification of LTL specs is over linear-time paths

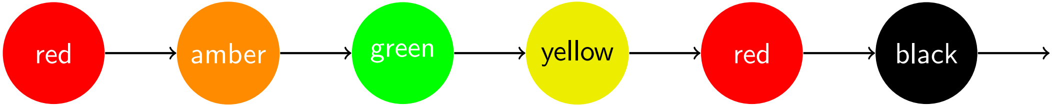
back to the traffic light model, consider the following path:



question: $\pi \models$ red?

Verification of LTL specs is over linear-time paths

back to the traffic light model, consider the following path:

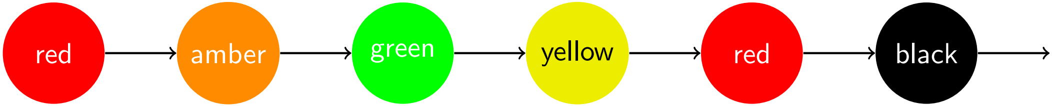


question: $\pi \models \text{red}$?

answer: yes

Verification of LTL specs is over linear-time paths

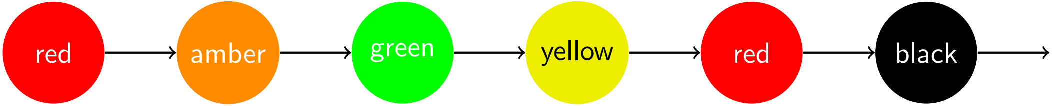
back to the traffic light model, consider the following path:



question: $\pi \models \bigcirc \bigcirc \text{red}$?

Verification of LTL specs is over linear-time paths

back to the traffic light model, consider the following path:



question: $\pi \models \bigcirc \bigcirc \textit{red}$?

answer: no

SAPIENZA
UNIVERSITÀ DI ROMA

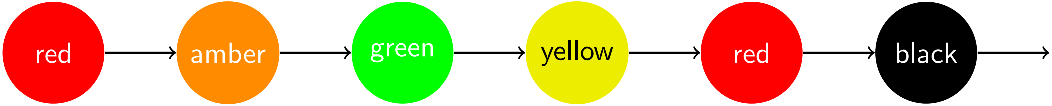
back to the traffic light model, consider the following path:

$$\text{red} \rightarrow \text{amber} \rightarrow \text{green} \rightarrow \text{yellow} \rightarrow \text{red} \rightarrow \text{black} \rightarrow$$

question: $\pi \models \text{red}\,\mathsf{U}\,\text{green}$?

Verification of LTL specs is over linear-time paths

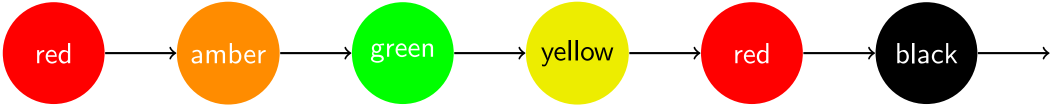
back to the traffic light model, consider the following path:



question: $\pi \models$ red$\mathsf{U}$green?

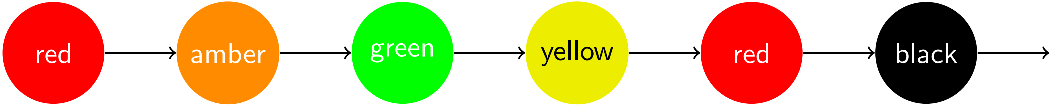
answer: yes, because $L(2) = \{$red, yellow$\}$

back to the traffic light model, consider the following path:



question: $\pi \models \Diamond \mathit{black}$?

back to the traffic light model, consider the following path:

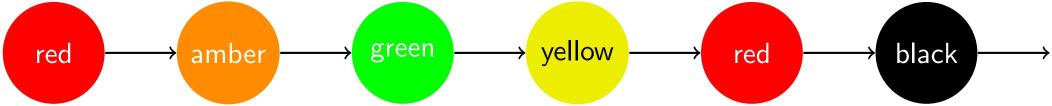


question: $\pi \models \Diamond \mathit{black}$?

answer: yes

Verification of LTL specs is over linear-time paths

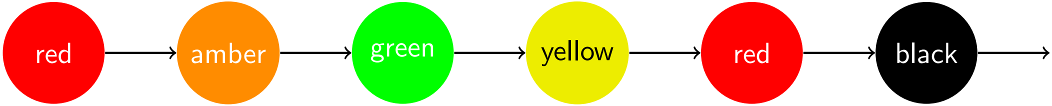
back to the traffic light model, consider the following path:



question: $\pi \models \Box \neg \mathit{red}$?

Verification of LTL specs is over linear-time paths

back to the traffic light model, consider the following path:



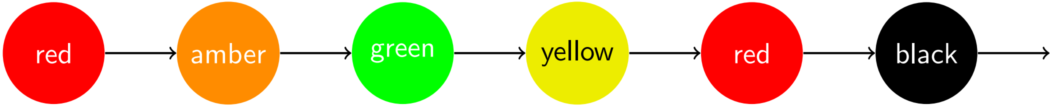
question: $\pi \models \Box \neg \mathsf{red}$?

answer: no

Verification of LTL specs is over linear-time paths

SAPIENZA
UNIVERSITÀ DI ROMA

back to the traffic light model, consider the following path:



question: $\pi \models (\Diamond \text{black}) \mathsf{U} (\bigcirc \text{red})$?

back to the traffic light model, consider the following path:

red $\longrightarrow$ amber $\longrightarrow$ green $\longrightarrow$ yellow $\longrightarrow$ red $\longrightarrow$ black $\longrightarrow$

question: $\pi \models (\Diamond \text{black}) U (\bigcirc \text{red})$?

answer: yes

Describe temporal modalities recursively

- $\varphi \mathsf{U} \psi \equiv \psi \vee (\varphi \wedge \bigcirc \varphi \mathsf{U} \psi)$ — $\varphi \mathsf{U} \psi$ is a "solution" of $\Psi = \psi \vee (\varphi \wedge \bigcirc \Psi)$
- $\Diamond \psi \equiv \psi \vee \bigcirc \Diamond \psi$ — $\Diamond \psi$ is a solution of $\Psi = \psi \vee \bigcirc \Psi$
- also $\Box \psi \equiv \neg \Diamond \neg \psi \equiv \psi \wedge \bigcirc \Box \psi$ — $\Box \psi$ is a solution of $\Psi = \psi \wedge \bigcirc \Psi$

SAPIENZA
UNIVERSITÀ DI ROMA

Define the Release operator R in a way that the following holds:

$\varphi \mathsf{R} \psi \equiv \neg(\neg\varphi \mathsf{U} \neg\psi)$

it also holds that

$\varphi \mathsf{U} \psi \equiv \neg(\neg\varphi \mathsf{R} \neg\psi)$ (Release is dual of Until)

SAPIENZA
UNIVERSITÀ DI ROMA

Define the Release operator R in a way that the following holds:

$\varphi \mathsf{R} \psi \equiv \neg(\neg\varphi \mathsf{U} \neg\psi)$

it also holds that

$\varphi \mathsf{U} \psi \equiv \neg(\neg\varphi \mathsf{R} \neg\psi)$ (Release is dual of Until)

PNF

Positive Normal Form for LTL: for $a \in AP$

$$\varphi ::= \mathsf{true} \mid \mathsf{false} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \varphi \mathsf{U} \varphi \mid \varphi \mathsf{R} \varphi$$

Release operator and PNF

Define the Release operator R in a way that the following holds:

$\varphi \mathsf{R} \psi \equiv \neg(\neg\varphi \mathsf{U} \neg\psi)$

it also holds that

$\varphi \mathsf{U} \psi \equiv \neg(\neg\varphi \mathsf{R} \neg\psi)$ (Release is dual of Until)

PNF

Positive Normal Form for LTL: for $a \in AP$

$$\varphi ::= \mathsf{true} \mid \mathsf{false} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \varphi \mathsf{U} \varphi \mid \varphi \mathsf{R} \varphi$$

Theorem

Each LTL formula φ admits an equivalent in PNF sometimes denoted $\mathsf{pnf}(\varphi)$

SAPIENZA UNIVERSITÀ DI ROMA

question: what class of LTL formulas capture invariants?

question: what class of LTL formulas capture invariants?

answer: $\Box\varphi$, where $\varphi ::= \mathsf{true} \mid a \mid \varphi \wedge \varphi \mid \neg\varphi$

question: what class of LTL formulas capture invariants?

answer: $\Box\varphi$, where $\varphi ::= \mathsf{true} \mid a \mid \varphi \wedge \varphi \mid \neg\varphi$

example: $\Box\neg\mathsf{red}$

question: how is the class of safety properties characterized?

question: how is the class of safety properties characterized?

answer: "nothing bad ever happens"

Classes of LTL specifications

question: how is the class of safety properties characterized?

answer: "nothing bad ever happens"

example: "every red light is immediately preceded by amber"

question: how can we express this property in LTL?

question: how is the class of safety properties characterized?

answer: “nothing bad ever happens”

example: “every red light is immediately preceded by amber”

question: how can we express this property in LTL?

answer: $\neg red \wedge \Box(\bigcirc red \rightarrow amber)$

question: how is the class of liveness properties characterized?

question: how is the class of liveness properties characterized?

answer: "something good eventually happens"

SAPIENZA UNIVERSITÀ DI ROMA

question: how is the class of liveness properties characterized?

answer: "something good eventually happens"

example: "the light is infinitely often red"

question: how can we express this property in LTL?

question: how is the class of liveness properties characterized?

answer: “something good eventually happens”

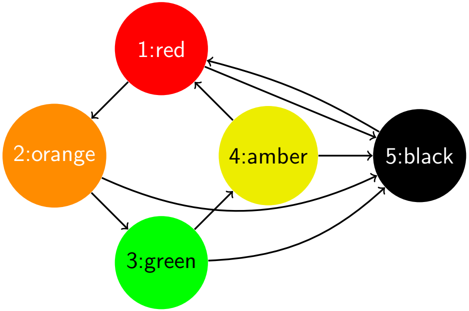
example: “the light is infinitely often red”

question: how can we express this property in LTL?

answer: $\Box\Diamond red$

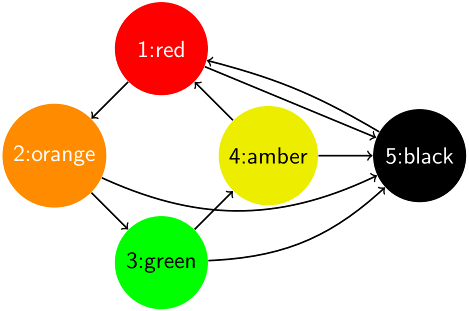
SAPIENZA
UNIVERSITÀ DI ROMA

consider traffic lights model



question: is $\psi := \Box(\mathit{black} \rightarrow \Diamond \mathit{red})$ a liveness property?

consider traffic lights model



question: is $\psi := \Box(\mathit{black} \to \Diamond \mathit{red})$ a liveness property?

answer: yes

does it hold $TS \models \psi$?

Fairness properties in LTL

unconditional fairness: "every transition is infinitely often taken"

$$\Box\Diamond\Psi$$

strong fairness: "if a transition is infinitely often enabled, then it is infinitely often taken"

$$\Box\Diamond\Phi \rightarrow \Box\Diamond\Psi$$

weak fairness: "if a transition is continuously enabled from a certain point in time, then it is infinitely often taken"

$$\Diamond\Box\Phi \rightarrow \Box\Diamond\Psi$$

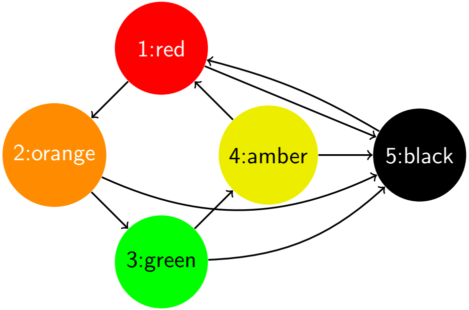
Fairness properties as LTL constraints

consider LTL constraint *fair*;
$FairPaths(s) = \{\pi \in Paths(s) \mid \pi \models fair\}$
$FairPaths(TS) = \{\pi \in Paths(TS) \mid \pi \models fair\}$

consider LTL specification φ;
$s \models_{fair} \varphi$ iff $\pi \models \varphi, \forall \pi \in FairPaths(s)$
$TS \models_{fair} \varphi$ iff $\pi \models \varphi, \forall \pi \in FairPaths(TS)$

fairness constraints are easily embedded with LTL verification:
$TS \models_{fair} \varphi \Leftrightarrow TS \models (fair \rightarrow \varphi)$

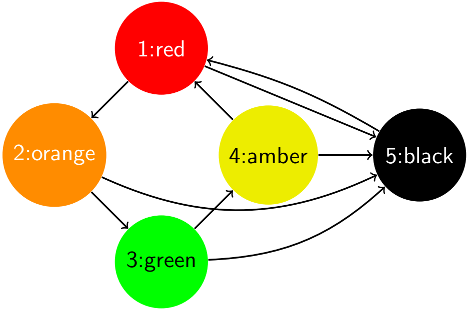
Fairness: an example



question: "is the traffic light infinitely often orange (amber and red)" under the strong fairness condition (if a transition is infinitely often enabled then it is infinitely often taken)?

express this in LTL:

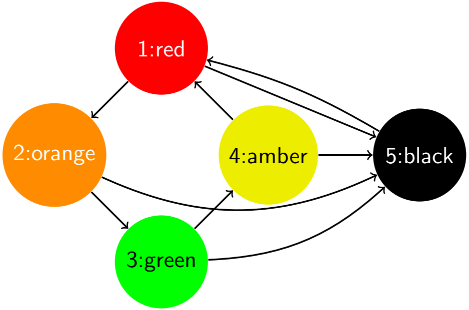
Fairness: an example



question: "is the traffic light infinitely often orange (amber and red)" under the strong fairness condition (if a transition is infinitely often enabled then it is infinitely often taken)? answer: no

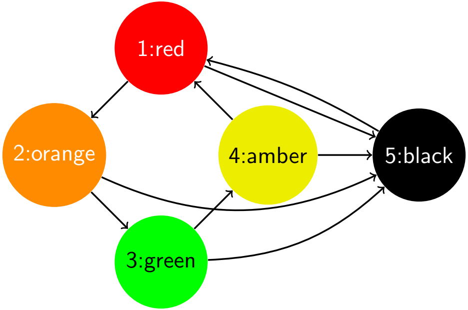
express this in LTL:

SAPIENZA UNIVERSITÀ DI ROMA



question: "is the traffic light infinitely often orange (amber and red)" under the strong fairness condition (if a transition is infinitely often enabled then it is infinitely often taken)? answer: no

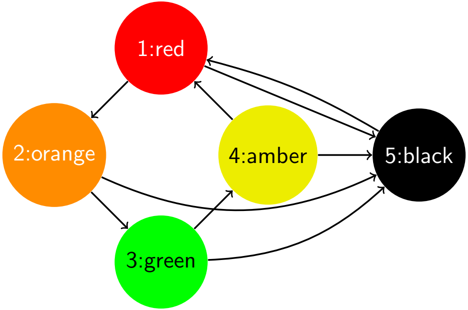
express this in LTL: $(\Box\Diamond red) \rightarrow \Box\Diamond(red \wedge \bigcirc(red \wedge amber))$



question: "is the traffic light infinitely often orange" under the weak fairness condition (if a transition is continuously enabled from a certain point in time then it is infinitely often taken)?

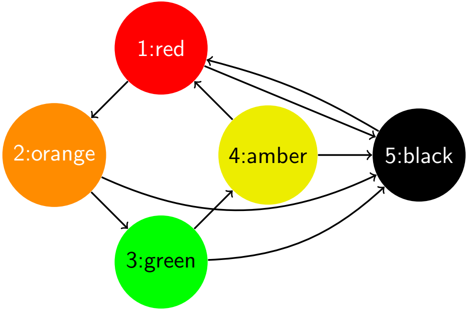
express this in LTL:

Fairness: a second example



question: "is the traffic light infinitely often orange" under the weak fairness condition (if a transition is continuously enabled from a certain point in time then it is infinitely often taken)? answer: yes

express this in LTL:



question: "is the traffic light infinitely often orange" under the weak fairness condition (if a transition is continuously enabled from a certain point in time then it is infinitely often taken)? answer: yes

express this in LTL: $(\Diamond\Box red) \rightarrow \Box\Diamond(red \wedge \bigcirc(red \wedge amber))$

(semantics of negation)

argue why $(TS \not\models \varphi) \not\equiv (TS \models \neg\varphi)$

(semantics of negation)

argue why $(TS \not\models \varphi) \not\equiv (TS \models \neg\varphi)$

and why instead $TS \models \neg\varphi \rightarrow TS \not\models \varphi$

Model Checking

Verifying that a system *satisfies* a given (temporal) specification.

Synthesis

Producing a system that *satisfies* a given (temporal) specification by construction.

Industry-strength approach to automated verification.

Idea: view the state transition graph of a program P as a model M_P, and express correctness criteria as logic formula ϕ

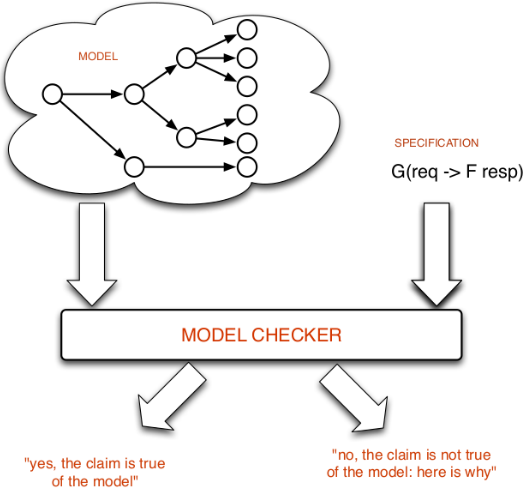
Verification then *reduces* to a model checking problem: $M_P \models \phi$

Most widely used logical specification languages: *LTL* and *CTL*;

Clarke and Emerson - Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. - LP'81

Can be (reasonably) efficiently automated, leading to many tools (SPIN, SMV, PRISM, MOCHA, MCMAS, EVE, ...).

Model Checking



Agents are powerful models in many areas of Computer Science.

Three characteristics

- Capabilities: actions and constraints
- Knowledge: information about environment
- Goal: specification of a task/objective to fulfill

Appears in many areas

Robotics	Planning
Software Engineering	Multi-Agent Systems
Process Management	Sequential decision making
Knowledge Representation	Reinforcement learning

Process

$$\xrightarrow{i} \boxed{f : I^* \to O} \xrightarrow{o}$$

Function f sends outputs according to the history of inputs.

Abadi, Lamport, Wolper - Realizable and Unrealizable Specifications of Reactive Systems. - ICALP'89

- Adhere to capabilities: actions always fulfill constraints
- Depend on knowledge: react on the stream of inputs
- Fulfill the specification

An agent satisfying these properties is correct.

Temporal specification setting

$$f \rightsquigarrow \mathcal{T}_f = \langle Q, I, O, \delta, \tau \rangle$$

Finite-state machines are expressive enough to implement agents correctly in a large class of temporal specifications.

SAPIENZA
UNIVERSITÀ DI ROMA

Instead of writing programs, we write specifications and run an automatic synthesis procedure that in turns produces the program.

Reactive Synthesis

- Self-programming mechanism.
- Specifying a problem is usually simpler than solving it.
- Aim: correct-by-construction.

- Pnueli and Rosner - On the Synthesis of a Reactive Module. - POPL'89
- Finkbeiner - Synthesis of Reactive Systems. - DSSE'16

SAPIENZA
UNIVERSITÀ DI ROMA

Synthesis problems as games

Agent vs environment	$\Longleftrightarrow$	Two-Player Game
Temporal specification	$\Longleftrightarrow$	Winning Condition
Correct program	$\Longleftrightarrow$	Winning Strategy

Solving synthesis = winning a game

Synthesizing a correct program reduces to winning a suitably defined formal game.
Solution techniques: Logic, Games, and Automata.

SAPIENZA UNIVERSITÀ DI ROMA

Synthesis problems as games

Agent vs environment	$\Longleftrightarrow$	Two-Player Game
Temporal specification	$\Longleftrightarrow$	Winning Condition
Correct program	$\Longleftrightarrow$	Winning Strategy

Solving synthesis = winning a game

Synthesizing a correct program reduces to winning a suitably defined formal game.
Solution techniques: Logic, Games, and Automata.

Stay tuned!