Overview

Introduction

Modelling parallel systems

Linear Time Properties

Regular Properties

Linear Temporal Logic (LTL)

Computation Tree Logic

Equivalences and Abstraction

Linear vs branching time

Linear vs branching time

transition system
$\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$

abstraction from actions

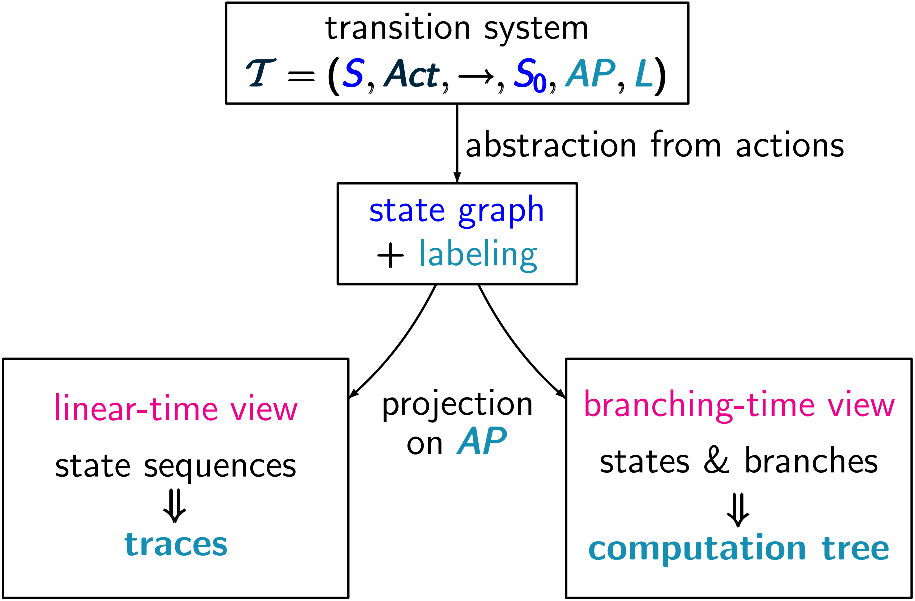
state graph
+ labeling

linear-time view
state sequences

branching-time view
states & branches

Linear vs branching time

CTLSS4.1-1



Computation tree

Computation tree

The computation tree of a transition system $\mathcal{T} = (S, Act, \rightarrow, s_0, AP, L)$ arises by:

- unfolding into a tree
- abstraction from the actions
- projection of the states s to their labels $L(s) \subseteq AP$

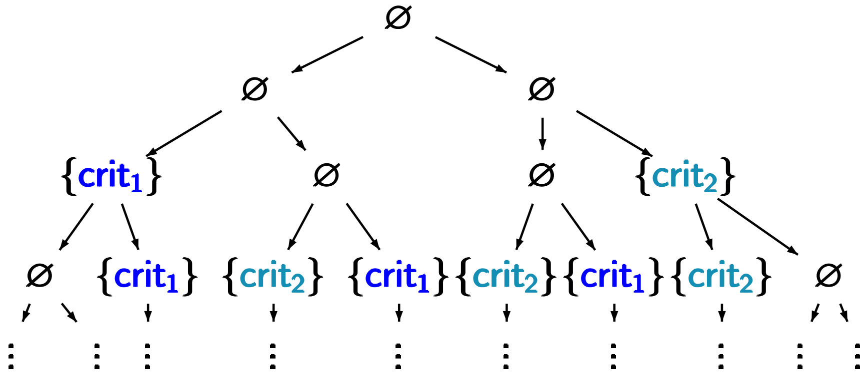
Computation tree

The computation tree of state s_0 in a transition system $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ arises by:

- unfolding $\mathcal{T}_{s_0} = (S, Act, \rightarrow, s_0, AP, L)$ into a tree
- abstraction from the actions
- projection of the states s to their labels $L(s) \subseteq AP$

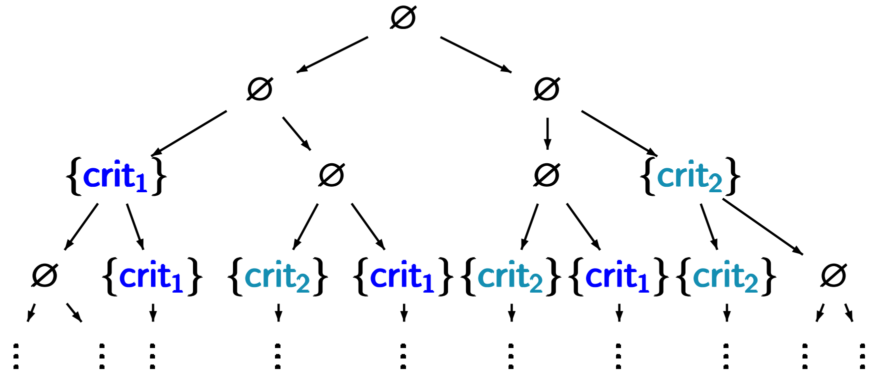
Example: computation tree

mutual exclusion with semaphore and $AP = \{crit_1, crit_2\}$:



Example: computation tree

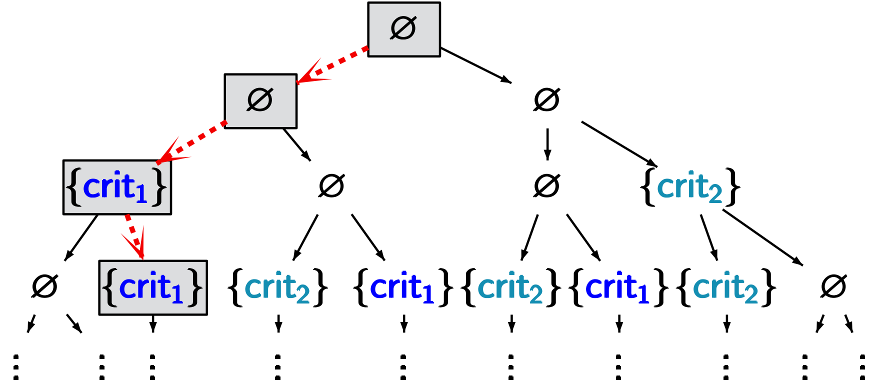
mutual exclusion with semaphore and $AP = \{crit_1, crit_2\}$:



path $\langle nc_1, nc_2\rangle$ $\langle wait_1, nc_2\rangle$ $\langle crit_1, nc_2\rangle$ $\langle crit_1, wait_2\rangle$ ...

Example: computation tree

mutual exclusion with semaphore and $AP = \{crit_1, crit_2\}$:



path	$\langle nc_1, nc_2\rangle$	$\langle wait_1, nc_2\rangle$	$\langle crit_1, nc_2\rangle$	$\langle crit_1, wait_2\rangle$	...
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
trace	$\varnothing$	$\varnothing$	$\{crit_1\}$	$\{crit_1\}$	...

Linear vs. branching time

	linear time	branching time
behavior	path based	state based

Linear vs. branching time

	linear time	branching time
behavior	path based traces	state based computation tree

Linear vs. branching time

	linear time	branching time
behavior	path based traces	state based computation tree
temporal logic	**LTL** path formulas	**CTL** state formulas

Linear vs. branching time

	linear time	branching time
behavior	path based traces	state based computation tree
temporal logic	**LTL** path formulas	**CTL** state formulas
model checking	PSPACE-complete $\mathcal{O}\big(\mathit{size}(\mathcal{T}) \cdot \exp(\lvert\varphi\rvert)\big)$	PTIME $\mathcal{O}\big(\mathit{size}(\mathcal{T}) \cdot \lvert\Phi\rvert\big)$

Linear vs. branching time

	linear time	branching time
behavior	path based traces	state based computation tree
temporal logic	**LTL** path formulas	**CTL** state formulas
model checking	PSPACE-complete $\mathcal{O}\big(\mathit{size}(\mathcal{T}) \cdot \exp(\lvert\varphi\rvert)\big)$	PTIME $\mathcal{O}\big(\mathit{size}(\mathcal{T}) \cdot \lvert\Phi\rvert\big)$
impl. relation	trace inclusion trace equivalence PSPACE-complete	simulation bisimulation PTIME

Linear vs. branching time

	linear time	branching time
behavior	path based traces	state based computation tree
temporal logic	**LTL** path formulas	**CTL** state formulas
model checking	PSPACE-complete $\mathcal{O}\big(\mathit{size}(\mathcal{T}) \cdot \exp(\lvert\varphi\rvert)\big)$	PTIME $\mathcal{O}\big(\mathit{size}(\mathcal{T}) \cdot \lvert\Phi\rvert\big)$
impl. relation	trace inclusion trace equivalence PSPACE-complete	simulation bisimulation PTIME
fairness	can be encoded	requires special treatment

Overview

Introduction

Modelling parallel systems

Linear Time Properties

Regular Properties

Linear Temporal Logic (LTL)

Computation Tree Logic

- syntax and semantics of CTL $\longleftarrow$
- expressiveness of CTL and LTL
- CTL model checking
- fairness, counterexamples/witnesses
- CTL$^+$ and CTL*

Equivalences and Abstraction

Computation Tree Logic (CTL)

Computation Tree Logic (CTL)

CTL (state) formulas:

$$\Phi \quad ::= \quad \mathit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

CTL path formulas:

$$\varphi \quad ::= \quad \bigcirc\Phi \mid \Phi_1 \,\mathsf{U}\, \Phi_2$$

Computation Tree Logic (CTL)

CTL (state) formulas:

$$\Phi \quad ::= \quad \mathit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

CTL path formulas:

$$\varphi \quad ::= \quad \bigcirc\Phi \mid \Phi_1 \mathsf{U} \Phi_2$$

eventually:

$$\exists\Diamond\Phi \quad \stackrel{\text{def}}{=} \quad \exists(\mathit{true}\,\mathsf{U}\,\Phi)$$

Computation Tree Logic (CTL)

CTL (state) formulas:

$$\Phi \quad ::= \quad \mathit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

CTL path formulas:

$$\varphi \quad ::= \quad \bigcirc\Phi \mid \Phi_1 \,\mathsf{U}\, \Phi_2$$

eventually:

$$\exists\Diamond\Phi \quad \stackrel{\text{def}}{=} \quad \exists(\mathit{true}\,\mathsf{U}\,\Phi)$$

$$\forall\Diamond\Phi \quad \stackrel{\text{def}}{=} \quad ?$$

CTL (state) formulas:

$$\Phi \quad ::= \quad \mathit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

CTL path formulas:

$$\varphi \quad ::= \quad \bigcirc\Phi \mid \Phi_1 \, \mathsf{U} \, \Phi_2$$

eventually:

$$\exists\Diamond\Phi \quad \overset{\text{def}}{=} \quad \exists(\mathit{true} \, \mathsf{U} \, \Phi)$$

$$\forall\Diamond\Phi \quad \overset{\text{def}}{=} \quad \forall(\mathit{true} \, \mathsf{U} \, \Phi)$$

Computation Tree Logic (CTL)

CTL (state) formulas:

$$\Phi \ ::= \ \textit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

CTL path formulas:

$$\varphi \ ::= \ \bigcirc\Phi \mid \Phi_1 \,\mathsf{U}\, \Phi_2$$

eventually:

$$\exists\Diamond\Phi \ \stackrel{\text{def}}{=} \ \exists(\textit{true}\,\mathsf{U}\,\Phi)$$

$$\forall\Diamond\Phi \ \stackrel{\text{def}}{=} \ \forall(\textit{true}\,\mathsf{U}\,\Phi)$$

always:

$$\exists\Box\Phi \ \stackrel{\text{def}}{=} \ ?$$

Computation Tree Logic (CTL)

CTL (state) formulas:

$$\Phi \quad ::= \quad \mathit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

CTL path formulas:

$$\varphi \quad ::= \quad \bigcirc\Phi \mid \Phi_1 \,\mathsf{U}\, \Phi_2$$

eventually:

$$\exists\Diamond\Phi \stackrel{\text{def}}{=} \exists(\mathit{true}\,\mathsf{U}\,\Phi)$$

$$\forall\Diamond\Phi \stackrel{\text{def}}{=} \forall(\mathit{true}\,\mathsf{U}\,\Phi)$$

always:

$$\exists\Box\Phi \stackrel{\text{def}}{=} \ ?$$

note: $\exists\neg\Diamond\neg\Phi$ is <u>no</u> **CTL** formula

CTL (state) formulas:

$$\Phi \quad ::= \quad \mathit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

CTL path formulas:

$$\varphi \quad ::= \quad \bigcirc\Phi \mid \Phi_1 \,\mathsf{U}\, \Phi_2$$

eventually:

$$\exists\Diamond\Phi \;\stackrel{\text{def}}{=}\; \exists(\mathit{true}\,\mathsf{U}\,\Phi)$$

$$\forall\Diamond\Phi \;\stackrel{\text{def}}{=}\; \forall(\mathit{true}\,\mathsf{U}\,\Phi)$$

always:

$$\exists\Box\Phi \;\stackrel{\text{def}}{=}\; \neg\forall\Diamond\neg\Phi$$

note: $\exists\neg\Diamond\neg\Phi$ is <u>no</u> **CTL** formula

Computation Tree Logic (CTL)

CTL (state) formulas:

$$\Phi \quad ::= \quad \mathit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

CTL path formulas:

$$\varphi \quad ::= \quad \bigcirc\Phi \mid \Phi_1 \,\mathsf{U}\, \Phi_2$$

eventually:

$$\exists\Diamond\Phi \stackrel{\text{def}}{=} \exists(\mathit{true}\,\mathsf{U}\,\Phi)$$

$$\forall\Diamond\Phi \stackrel{\text{def}}{=} \forall(\mathit{true}\,\mathsf{U}\,\Phi)$$

always:

$$\exists\Box\Phi \stackrel{\text{def}}{=} \neg\forall\Diamond\neg\Phi$$

$$\forall\Box\Phi \stackrel{\text{def}}{=} \neg\exists\Diamond\neg\Phi$$

note: $\exists\neg\Diamond\neg\Phi$ is <u>no</u> **CTL** formula

Syntax of CTL

CTL (state) formulas:

$$\Phi \quad ::= \quad \mathit{true} \;\big|\; a \;\big|\; \Phi_1 \wedge \Phi_2 \;\big|\; \neg\Phi \;\big|\; \exists\varphi \;\big|\; \forall\varphi$$

CTL path formulas:

$$\varphi \quad ::= \quad \bigcirc\Phi \;\big|\; \Phi_1 \,\mathsf{U}\, \Phi_2$$

$\bigcirc \quad \widehat{=} \quad$ next

$\mathsf{U} \quad \widehat{=} \quad$ until

CTL (state) formulas:

$$\Phi \;::=\; \textit{true} \;\mid\; a \;\mid\; \Phi_1 \wedge \Phi_2 \;\mid\; \neg\Phi \;\mid\; \exists\varphi \;\mid\; \forall\varphi$$

CTL path formulas:

$$\varphi \;::=\; \bigcirc\Phi \;\mid\; \Phi_1 \,\mathsf{U}\, \Phi_2 \;\mid\; \Diamond\Phi \;\mid\; \Box\Phi$$

$\bigcirc \;\widehat{=}\;$ next

$\mathsf{U} \;\widehat{=}\;$ until

$\Diamond \;\widehat{=}\;$ eventually

$\Box \;\widehat{=}\;$ always

Examples for CTL formulas

CTL (state) formulas:

$$\Phi \quad ::= \quad \mathit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

CTL path formulas:

$$\varphi \quad ::= \quad \bigcirc\Phi \mid \Phi_1 \mathsf{U} \Phi_2 \mid \Diamond\Phi \mid \Box\Phi$$

mutual exclusion (safety) $\forall\Box(\neg \mathit{crit}_1 \vee \neg \mathit{crit}_2)$

Examples for CTL formulas

CTL (state) formulas:

$$\Phi \ ::= \ \textit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

CTL path formulas:

$$\varphi \ ::= \ \bigcirc\Phi \mid \Phi_1 \, \mathsf{U} \, \Phi_2 \mid \Diamond\Phi \mid \Box\Phi$$

mutual exclusion (safety) $\forall\Box(\neg \mathit{crit}_1 \vee \neg \mathit{crit}_2)$

"every request will be answered eventually"

$$\forall\Box(\mathit{request} \rightarrow \forall\Diamond \mathit{response})$$

CTL (state) formulas:

$$\Phi \quad ::= \quad \mathit{true} \;\big|\; a \;\big|\; \Phi_1 \wedge \Phi_2 \;\big|\; \neg\Phi \;\big|\; \exists\varphi \;\big|\; \forall\varphi$$

CTL path formulas:

$$\varphi \quad ::= \quad \bigcirc\Phi \;\big|\; \Phi_1 \,\mathsf{U}\, \Phi_2 \;\big|\; \Diamond\Phi \;\big|\; \Box\Phi$$

mutual exclusion (safety) $\forall\Box(\neg \mathit{crit}_1 \vee \neg \mathit{crit}_2)$

"every request will be answered eventually"

$\forall\Box(\mathit{request} \rightarrow \forall\Diamond\mathit{response})$

traffic lights $\forall\Box(\mathit{yellow} \rightarrow \forall\bigcirc\mathit{red})$

Examples for CTL formulas

CTL (state) formulas:

$$\Phi \quad ::= \quad \mathit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

CTL path formulas:

$$\varphi \quad ::= \quad \bigcirc\Phi \mid \Phi_1 \mathsf{U} \Phi_2 \mid \Diamond\Phi \mid \Box\Phi$$

mutual exclusion (safety) $\forall\Box(\neg \mathit{crit}_1 \vee \neg \mathit{crit}_2)$

"every request will be answered eventually"

$\forall\Box(\mathit{request} \rightarrow \forall\Diamond \mathit{response})$

traffic lights $\forall\Box(\mathit{yellow} \rightarrow \forall\bigcirc \mathit{red})$

reset possiblity $\forall\Box\exists\Diamond \mathit{start}$

CTL (state) formulas:

$$\Phi \ ::= \ \mathit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

CTL path formulas:

$$\varphi \ ::= \ \bigcirc\Phi \mid \Phi_1 \mathsf{U} \Phi_2 \mid \Diamond\Phi \mid \Box\Phi$$

mutual exclusion (safety) $\forall\Box(\neg \mathit{crit}_1 \vee \neg \mathit{crit}_2)$

"every request will be answered eventually"

$\forall\Box(\mathit{request} \rightarrow \forall\Diamond \mathit{response})$

traffic lights $\forall\Box(\mathit{yellow} \rightarrow \forall\bigcirc \mathit{red})$

reset possiblity $\forall\Box\exists\Diamond \mathit{start}$

unconditional process fairness $\forall\Box\forall\Diamond \mathit{crit}_1 \wedge \forall\Box\forall\Diamond \mathit{crit}_2$

Example: 15-puzzle

CTLSS4.1-5

6	8	2	12
4	1	13	5
	9	10	14
7	11	15	3

$\overset{?}{\rightsquigarrow}$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Example: 15-puzzle

6	8	2	12
4	1	13	5
	9	10	14
7	11	15	3

$\overset{?}{\rightsquigarrow}$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

transition system has $\mathbf{16!} \approx \mathbf{2 \cdot 10^{13}}$ states

Example: 15-puzzle

6	8	2	12
4	1	13	5
	9	10	14
7	11	15	3

$\overset{?}{\rightsquigarrow}$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

transition system has $\mathbf{16!} \approx \mathbf{2 \cdot 10^{13}}$ states

$\uparrow$

states:	game configurations
transitions:	legal moves

Example: 15-puzzle

6	8	2	12
4	1	13	5
	9	10	14
7	11	15	3

$\overset{?}{\rightsquigarrow}$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

- transition system has $16! \approx 2 \cdot 10^{13}$ states
- representation as parallel system:

  $$\mathit{left} \parallel \mathit{up} \parallel \mathit{down} \parallel \mathit{right}$$

  with shared variables $\mathit{field}[i]$ for $i = 1, \ldots, 16$

Example: 15-puzzle

6	8	2	12
4	1	13	5
	9	10	14
7	11	15	3

$\overset{?}{\rightsquigarrow}$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

- transition system has $\mathbf{16!} \approx \mathbf{2 \cdot 10^{13}}$ states
- representation as parallel system:

  $$\mathit{left} \parallel \mathit{up} \parallel \mathit{down} \parallel \mathit{right}$$

  with shared variables $\mathit{field}[i]$ for $i = 1, \ldots, 16$

CTL specification:

$$\exists\Diamond \bigwedge_{1 \leq i \leq 15} \text{“}\mathit{piece}\ i \text{ on } \mathit{field}[i]\text{”}$$

6	8	2	12
4	1	13	5
	9	10	14
7	11	15	3

$\overset{?}{\rightsquigarrow}$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

- transition system has $16! \approx 2 \cdot 10^{13}$ states
- representation as parallel system:

  $$\mathit{left} \,\|\, \mathit{up} \,\|\, \mathit{down} \,\|\, \mathit{right}$$

  with shared variables $\mathit{field}[i]$ for $i = 1, \ldots, 16$

CTL specification: seeking for a **witness** for

$$\exists\Diamond \bigwedge_{1 \leq i \leq 15} \text{“}\mathit{piece}\ i \text{ on } \mathit{field}[i]\text{”}$$

Semantics of CTL

define a satisfaction relation $\models$ for CTL formulas over AP and a given TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$

Semantics of CTL

define a satisfaction relation $\models$ for CTL formulas over AP and a given TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ without terminal states

Semantics of CTL

define a satisfaction relation $\models$ for CTL formulas over AP and a given TS $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ without terminal states

- interpretation of state formulas over the states
- interpretation of path formulas over the paths (infinite path fragments)

Satisfaction relation for path formulas

Satisfaction relation for path formulas

Let $\pi = s_0\, s_1\, s_2\, \ldots$ be an infinite path fragment.

Satisfaction relation for path formulas

Let $\pi = s_0\, s_1\, s_2\, \ldots$ be an infinite path fragment.

$\pi \models \bigcirc \Phi$ iff $s_1 \models \Phi$

Satisfaction relation for path formulas

Let $\pi = s_0\, s_1\, s_2\, \ldots$ be an infinite path fragment.

$$
\begin{array}{lll}
\pi \models \bigcirc \Phi & \text{iff} & s_1 \models \Phi \\
\pi \models \Phi_1 \, \mathsf{U} \, \Phi_2 & \text{iff} & \text{there exists } j \geq 0 \text{ such that} \\
 & & \qquad s_j \models \Phi_2 \\
 & & \qquad s_k \models \Phi_1 \text{ for } 0 \leq k < j
\end{array}
$$

Satisfaction relation for path formulas

Let $\pi = s_0\, s_1\, s_2 \, \ldots$ be an infinite path fragment.

$\pi \models \bigcirc \Phi$	iff	$s_1 \models \Phi$
$\pi \models \Phi_1 \, \mathsf{U} \, \Phi_2$	iff	there exists $j \geq 0$ such that $s_j \models \Phi_2$ and $s_k \models \Phi_1$ for $0 \leq k < j$

semantics of derived operators:

$\pi \models \Diamond \Phi$ iff there exists $j \geq 0$ with $s_j \models \Phi$

Satisfaction relation for path formulas

Let $\pi = s_0\, s_1\, s_2\, \ldots$ be an infinite path fragment.

$\pi \models \bigcirc\Phi$	iff	$s_1 \models \Phi$
$\pi \models \Phi_1 \mathsf{U} \Phi_2$	iff	there exists $j \geq 0$ such that
		$s_j \models \Phi_2$
		$s_k \models \Phi_1$ for $0 \leq k < j$

semantics of derived operators:

$\pi \models \Diamond\Phi$ iff there exists $j \geq 0$ with $s_j \models \Phi$

$\pi \models \Box\Phi$ iff for all $j \geq 0$ we have: $s_j \models \Phi$

Satisfaction relation for state formulas

Satisfaction relation for state formulas

$s \models true$

Satisfaction relation for state formulas

$s \models \mathit{true}$

$s \models a \quad$ iff $\quad a \in L(s)$

Satisfaction relation for state formulas

$s \models \mathit{true}$

$s \models a$ iff $a \in L(s)$

$s \models \Phi_1 \wedge \Phi_2$ iff $s \models \Phi_1$ and $s \models \Phi_2$

Satisfaction relation for state formulas

$s \models \mathit{true}$

$s \models a$ iff $a \in L(s)$

$s \models \Phi_1 \wedge \Phi_2$ iff $s \models \Phi_1$ and $s \models \Phi_2$

$s \models \neg\Phi$ iff $s \not\models \Phi$

Satisfaction relation for state formulas

$$
\begin{array}{lll}
s \models \mathit{true} & & \\
s \models a & \text{iff} & a \in L(s) \\
s \models \Phi_1 \wedge \Phi_2 & \text{iff} & s \models \Phi_1 \text{ and } s \models \Phi_2 \\
s \models \neg\Phi & \text{iff} & s \not\models \Phi \\
s \models \exists\varphi & \text{iff} & \text{there is a path } \pi \in \mathit{Paths}(s) \\
 & & \qquad\qquad \text{s.t. } \pi \models \varphi
\end{array}
$$

Satisfaction relation for state formulas

$s \models \mathit{true}$		
$s \models a$	iff	$a \in L(s)$
$s \models \Phi_1 \wedge \Phi_2$	iff	$s \models \Phi_1$ and $s \models \Phi_2$
$s \models \neg\Phi$	iff	$s \not\models \Phi$
$s \models \exists\varphi$	iff	there is a path $\pi \in \mathit{Paths}(s)$ s.t. $\pi \models \varphi$
$s \models \forall\varphi$	iff	for each path $\pi \in \mathit{Paths}(s)$: $\pi \models \varphi$

Satisfaction relation for state formulas

$s \models \mathit{true}$

$s \models a$ iff $a \in L(s)$

$s \models \Phi_1 \wedge \Phi_2$ iff $s \models \Phi_1$ and $s \models \Phi_2$

$s \models \neg\Phi$ iff $s \not\models \Phi$

$s \models \exists\varphi$ iff there is a path $\pi \in \mathit{Paths}(s)$ s.t. $\pi \models \varphi$

$s \models \forall\varphi$ iff for each path $\pi \in \mathit{Paths}(s)$: $\pi \models \varphi$

satisfaction set for state formula Φ:

$$\mathit{Sat}(\Phi) \stackrel{\text{def}}{=} \{s \in S : s \models \Phi\}$$

Semantics of the next operator

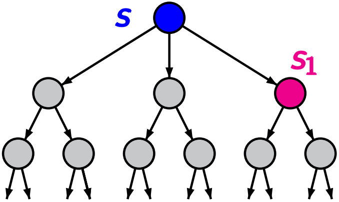
Semantics of the next operator

$$s \models \exists \bigcirc \Phi \quad \text{iff} \quad \text{there exists } \pi = s\, s_1\, s_2 \ldots \in Paths(s) \text{ s.t. } \pi \models \bigcirc \Phi$$

Semantics of the next operator

$s \models \exists\bigcirc\Phi$ iff there exists $\pi = s\, s_1\, s_2\, \ldots \in \mathit{Paths}(s)$
s.t. $\pi \models \bigcirc\Phi$, i.e., $s_1 \models \Phi$

$\exists\bigcirc\Phi$



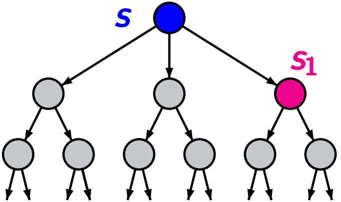
$\mathit{Post}(s) \cap \mathit{Sat}(\Phi) \neq \varnothing$

Semantics of the next operator

$s \models \exists \bigcirc \Phi$ iff there exists $\pi = s\, s_1\, s_2 \ldots \in Paths(s)$
s.t. $\pi \models \bigcirc \Phi$, i.e., $s_1 \models \Phi$

$s \models \forall \bigcirc \Phi$ iff for all $\pi = s\, s_1\, s_2 \ldots \in Paths(s)$:
$\pi \models \bigcirc \Phi$

$\exists \bigcirc \Phi$



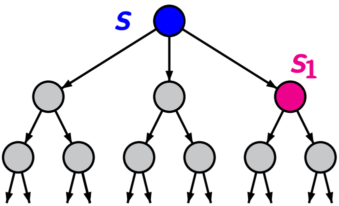
$Post(s) \cap Sat(\Phi) \neq \varnothing$

Semantics of the next operator

$s \models \exists\bigcirc\Phi$ iff there exists $\pi = s\, s_1\, s_2 \ldots \in Paths(s)$ s.t. $\pi \models \bigcirc\Phi$, i.e., $s_1 \models \Phi$

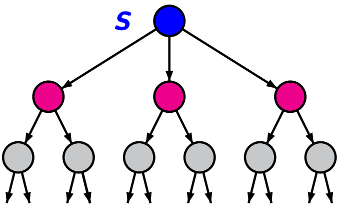
$s \models \forall\bigcirc\Phi$ iff for all $\pi = s\, s_1\, s_2 \ldots \in Paths(s)$: $\pi \models \bigcirc\Phi$, i.e., $s_1 \models \Phi$

$\exists\bigcirc\Phi$



$Post(s) \cap Sat(\Phi) \neq \varnothing$

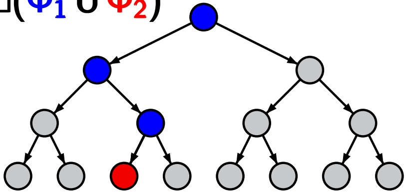
$\forall\bigcirc\Phi$



$Post(s) \subseteq Sat(\Phi)$

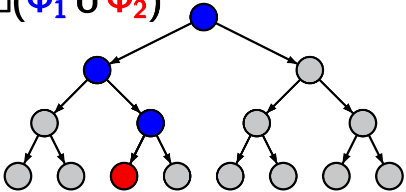
Semantics of until

$\exists(\Phi_1 \, \mathsf{U} \, \Phi_2)$

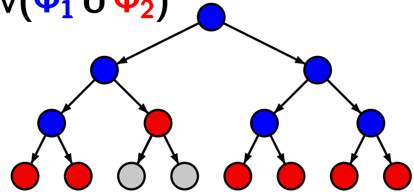


Semantics of until

$\exists(\Phi_1 \, \mathsf{U} \, \Phi_2)$

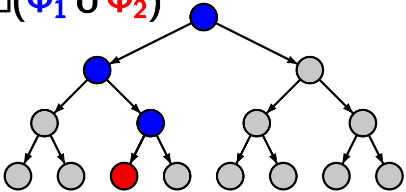


$\forall(\Phi_1 \, \mathsf{U} \, \Phi_2)$

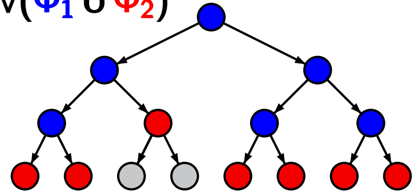


Semantics of until and eventually

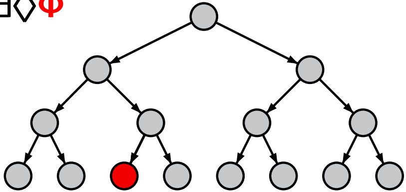
$\exists(\Phi_1 \,\mathsf{U}\, \Phi_2)$



$\forall(\Phi_1 \,\mathsf{U}\, \Phi_2)$

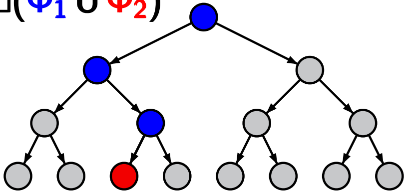


$\exists\Diamond\Phi$

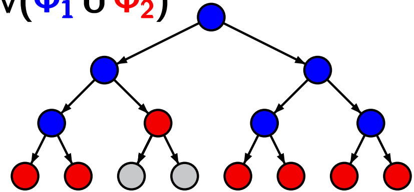


Semantics of until and eventually

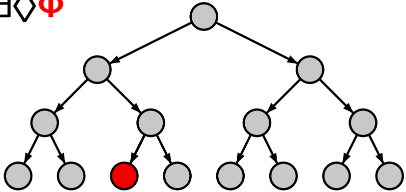
$\exists(\Phi_1 \, \mathsf{U} \, \Phi_2)$



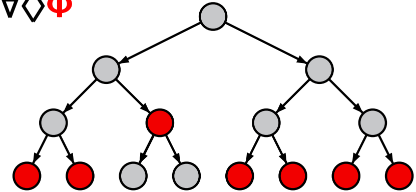
$\forall(\Phi_1 \, \mathsf{U} \, \Phi_2)$



$\exists\lozenge\Phi$

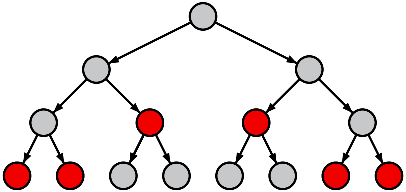


$\forall\lozenge\Phi$

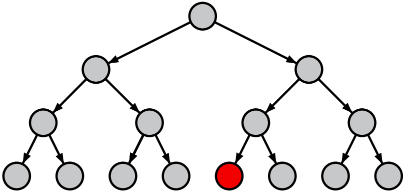


Semantics of eventually and always

$\forall\Diamond\Phi$

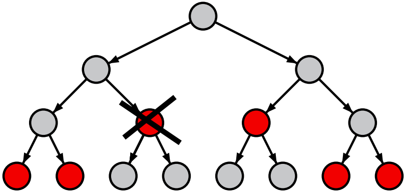


$\exists\Diamond\Phi$

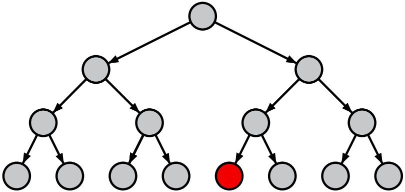


Semantics of eventually and always

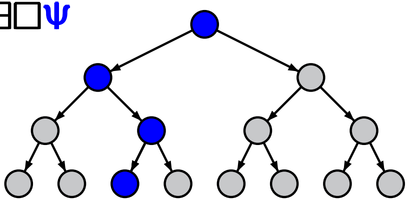
$\neg \forall \Diamond \Phi$



$\exists \Diamond \Phi$

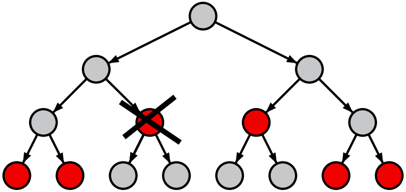


$\exists \Box \Psi$

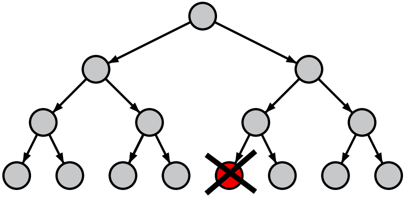


Semantics of eventually and always

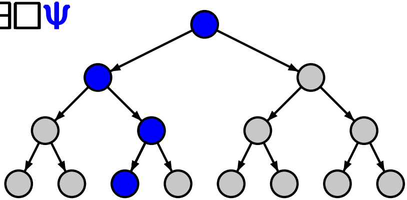
$\neg\forall\Diamond\Phi$



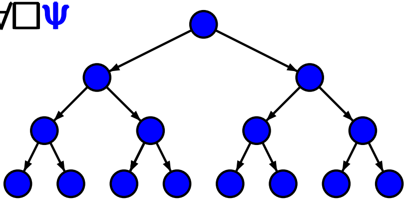
$\neg\exists\Diamond\Phi$



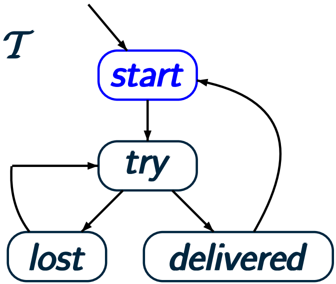
$\exists\Box\Psi$



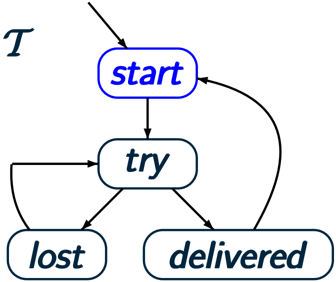
$\forall\Box\Psi$



Example for CTL semantics



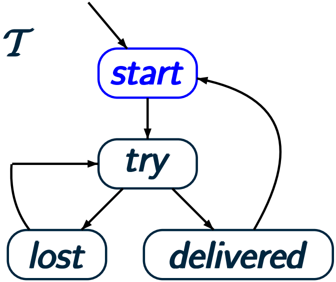
Example for CTL semantics



CTL formula

$\Phi = \forall\Box\forall\Diamond \textit{start}$

Example for CTL semantics

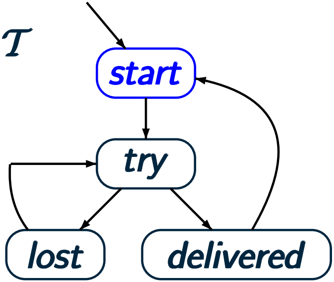


CTL formula

$\Phi = \forall\Box \boxed{\forall\Diamond start}$

$Sat(\forall\Diamond start) = \textbf{?}$

Example for CTL semantics

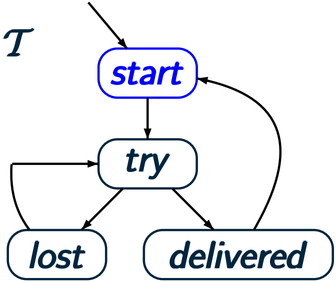


CTL formula

$\Phi = \forall\Box \boxed{\forall\Diamond start}$

$Sat(\forall\Diamond start) = \{start, delivered\}$

Example for CTL semantics

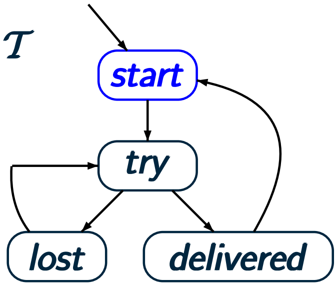


CTL formula

$$\Phi = \forall\Box\forall\Diamond start \quad \widehat{=} \quad \forall\Box(start \vee delivered)$$

$$Sat(\forall\Diamond start) = \{start, delivered\}$$

Example for CTL semantics



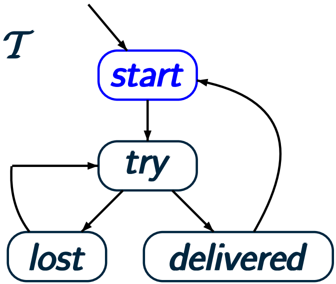
CTL formula

$$\Phi = \forall\Box\forall\Diamond start \quad \widehat{=} \quad \forall\Box(start \vee delivered)$$

$$Sat(\forall\Diamond start) = \{start, delivered\}$$

$$Sat(\Phi) = \varnothing$$

Example for CTL semantics



$\mathcal{T} \not\models \forall\Box\forall\Diamond start$

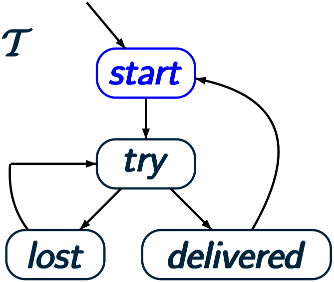
CTL formula

$\Phi = \forall\Box\forall\Diamond start \quad \widehat{=} \quad \forall\Box(start \vee delivered)$

$$Sat(\forall\Diamond start) = \{start, delivered\}$$

$$Sat(\Phi) = \varnothing$$

Example for CTL semantics



$\mathcal{T} \not\models \forall\Box\forall\Diamond start$

"infinitely often *start*"

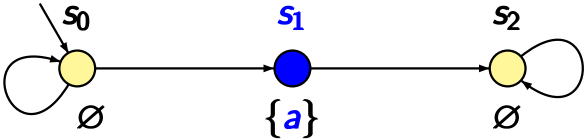
CTL formula

$\Phi = \forall\Box\forall\Diamond start \quad \widehat{=} \quad \forall\Box(start \vee delivered)$

$$Sat(\forall\Diamond start) = \{start, delivered\}$$

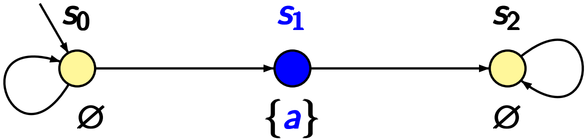
$$Sat(\Phi) = \varnothing$$

Example: CTL semantics



does $\mathcal{T} \models \exists\bigcirc\forall\Box\neg a$ hold ?

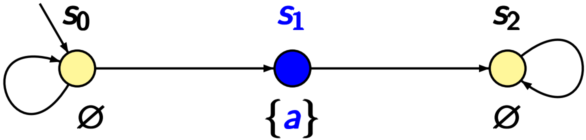
Example: CTL semantics



does $\mathcal{T} \models \exists\bigcirc\forall\Box\neg a$ hold **?**

answer: no

Example: CTL semantics

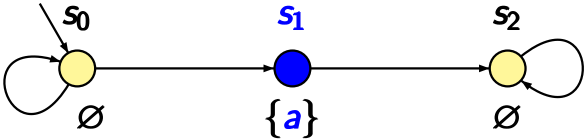


does $\mathcal{T} \models \exists\bigcirc\forall\Box\neg a$ hold ?

answer: no

$$\boxed{Sat(\forall\Box\neg a) \;=\; \{s_2\}}$$

Example: CTL semantics

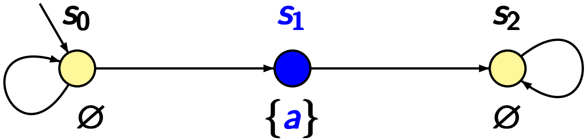


does $\mathcal{T} \models \exists\bigcirc\forall\Box\neg a$ hold ?

answer: no

$$\begin{array}{rcl} Sat(\forall\Box\neg a) & = & \{s_2\} \\ Sat(\exists\bigcirc\forall\Box\neg a) & = & \{s_2, s_1\} \end{array}$$

Example: CTL semantics

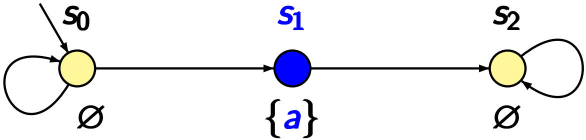


does $\mathcal{T} \models \exists\bigcirc\forall\Box\neg a$ hold ?

answer: no

does $\mathcal{T} \models \forall\Box\exists\bigcirc\neg a$ hold ?

Example: CTL semantics



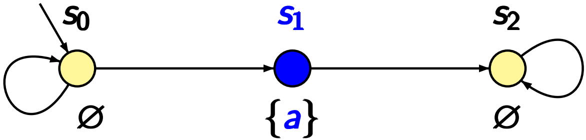
does $\mathcal{T} \models \exists\bigcirc\forall\Box\neg a$ hold ?

answer: no

does $\mathcal{T} \models \forall\Box\exists\bigcirc\neg a$ hold ?

answer: yes

Example: CTL semantics



does $\mathcal{T} \models \exists\bigcirc\forall\Box\neg a$ hold ?

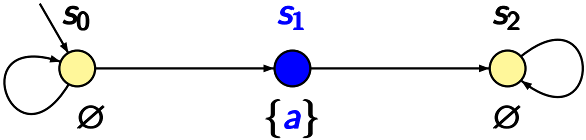
answer: no

does $\mathcal{T} \models \forall\Box\exists\bigcirc\neg a$ hold ?

answer: yes

$$\boxed{\mathit{Sat}(\exists\bigcirc\neg a) = \{s_0, s_1, s_2\}}$$

Example: CTL semantics



does $\mathcal{T} \models \exists\bigcirc\forall\Box\neg a$ hold ?

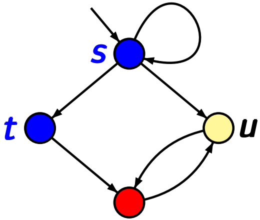
answer: no

does $\mathcal{T} \models \forall\Box\exists\bigcirc\neg a$ hold ?

answer: yes

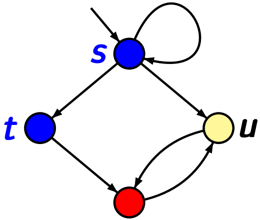
$$\boxed{\begin{array}{rcl} Sat(\exists\bigcirc\neg a) & = & \{s_0, s_1, s_2\} \\ Sat(\forall\Box\exists\bigcirc\neg a) & = & \{s_0, s_1, s_2\} \end{array}}$$

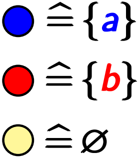
Example: CTL semantics



$\mathcal{T} \models \exists\Box\exists(a \cup b)$?

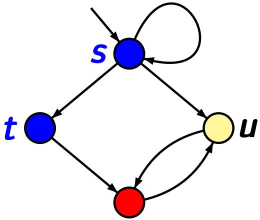
Example: CTL semantics





$\mathcal{T} \models \exists\Box\exists(a \,\mathsf{U}\, b)$ $\quad\checkmark$ as $s \models \exists(a \,\mathsf{U}\, b)$

Example: CTL semantics



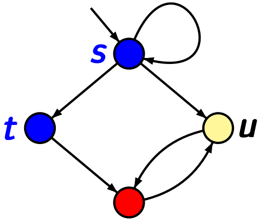
$\bullet \mathrel{\widehat{=}} \{a\}$

$\bullet \mathrel{\widehat{=}} \{b\}$

$\circ \mathrel{\widehat{=}} \varnothing$

$\mathcal{T} \models \exists\Box\exists(a \,\mathsf{U}\, b)$ $\quad\surd\quad$ as $s\,s\,s\ldots \models \Box\exists(a \,\mathsf{U}\, b)$

Example: CTL semantics



$\bullet \mathrel{\widehat{=}} \{a\}$

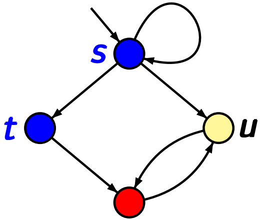
$\bullet \mathrel{\widehat{=}} \{b\}$

$\circ \mathrel{\widehat{=}} \varnothing$

$\mathcal{T} \models \exists\Box\exists(a \,\mathsf{U}\, b)$ $\quad\checkmark\quad$ as $s\,s\,s\,\ldots \models \Box\exists(a \,\mathsf{U}\, b)$

$\mathcal{T} \models \exists((\exists\bigcirc a) \,\mathsf{U}\, b)$ $\quad$ **?**

Example: CTL semantics



$\bullet \mathrel{\widehat{=}} \{a\}$ (blue)

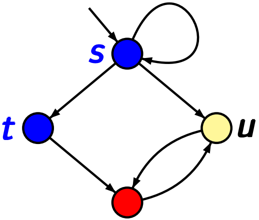
$\bullet \mathrel{\widehat{=}} \{b\}$ (red)

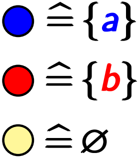
$\bullet \mathrel{\widehat{=}} \varnothing$ (yellow)

$\mathcal{T} \models \exists\Box\exists(a \,\mathsf{U}\, b)$ $\quad \surd \quad$ as $s\,s\,s\,\ldots \models \Box\exists(a \,\mathsf{U}\, b)$

$\mathcal{T} \not\models \exists((\exists\bigcirc a) \,\mathsf{U}\, b)$ $\quad$ as $t \not\models \exists\bigcirc a$, $u \not\models \exists\bigcirc a$

Example: CTL semantics



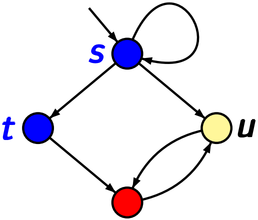


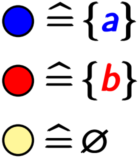
$\mathcal{T} \models \exists\Box\exists(a \,\mathsf{U}\, b)$ $\quad\surd\quad$ as $s\,s\,s\,\ldots \models \Box\exists(a \,\mathsf{U}\, b)$

$\mathcal{T} \not\models \exists((\exists\bigcirc a) \,\mathsf{U}\, b)$ $\quad$ as $t \not\models \exists\bigcirc a$, $u \not\models \exists\bigcirc a$

$\mathcal{T} \models \exists(a \,\mathsf{U}\, \forall(\neg a \,\mathsf{U}\, b))$ $\quad$?

Example: CTL semantics

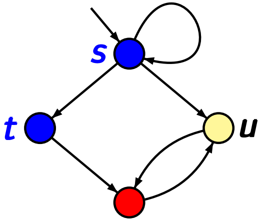


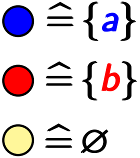


$\mathcal{T} \models \exists\Box\exists(a \,\mathsf{U}\, b)$ $\quad\surd\quad$ as $s\,s\,s\,\ldots \models \Box\exists(a \,\mathsf{U}\, b)$

$\mathcal{T} \not\models \exists((\exists\bigcirc a) \,\mathsf{U}\, b)$ $\quad$ as $t \not\models \exists\bigcirc a$, $u \not\models \exists\bigcirc a$

$\mathcal{T} \models \exists(a \,\mathsf{U}\, \forall(\neg a \,\mathsf{U}\, b))$ $\quad\surd$

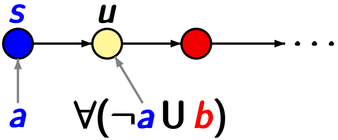




$\mathcal{T} \models \exists\Box\exists(a \,\mathsf{U}\, b)$ $\quad\checkmark\quad$ as $s\,s\,s\ldots \models \Box\exists(a \,\mathsf{U}\, b)$

$\mathcal{T} \not\models \exists((\exists\bigcirc a) \,\mathsf{U}\, b)$ $\quad$ as $t \not\models \exists\bigcirc a$, $u \not\models \exists\bigcirc a$

$\mathcal{T} \models \exists(a \,\mathsf{U}\, \forall(\neg a \,\mathsf{U}\, b))$ $\quad\checkmark$



$\models a \,\mathsf{U}\, \forall(\neg a \,\mathsf{U}\, b)$