

Introduction

Modelling parallel systems

Linear Time Properties

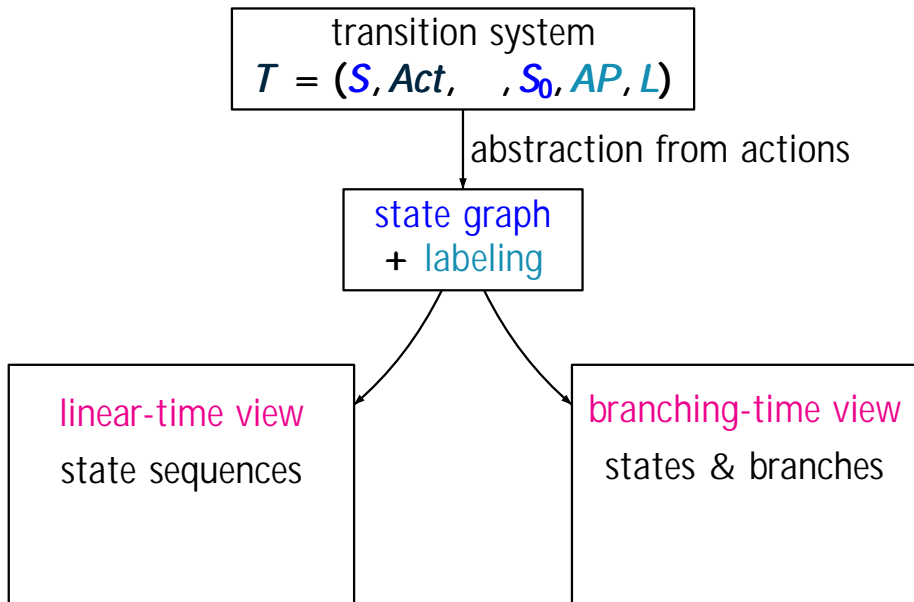
Regular Properties

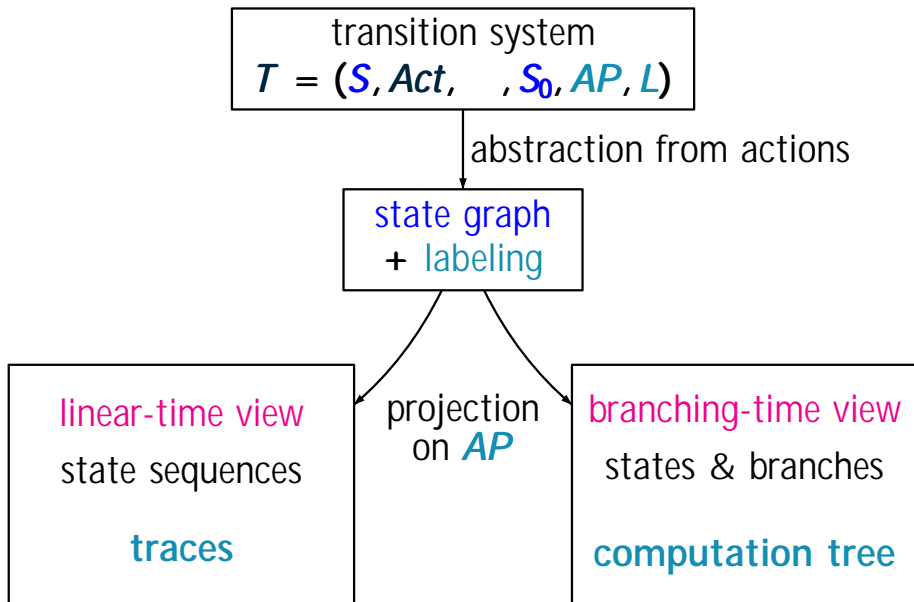
Linear Temporal Logic (LTL)

**Computation Tree Logic**

Equivalences and Abstraction









The computation tree of a transition system

$T = (S, Act, \rightarrow, s_0, AP, L)$  arises by:

$\hookrightarrow$  unfolding into a tree

$\hookrightarrow$  abstraction from the actions

$\hookrightarrow$  projection of the states  $s$  to their labels  $L(s)$   $AP$

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

• unfolding  $T_{s_0} = (S, Act, \rightarrow, s_0, AP, L)$  into a tree

• abstraction from the actions

• projection of the states  $s$  to their labels  $L(s)$   $AP$

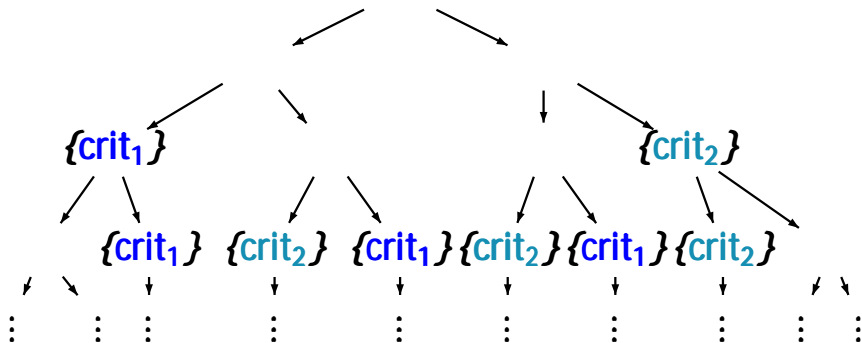




## Example: computation tree

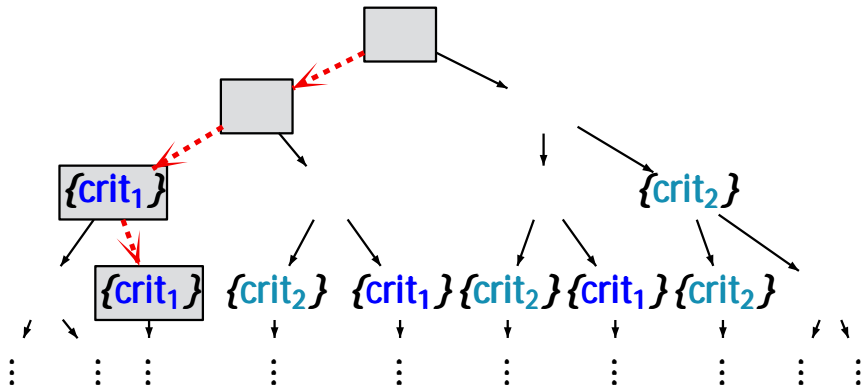
ctlss4.1-1a

mutual exclusion with semaphore and  $AP = \{\text{crit}_1, \text{crit}_2\}$ :



path	$\text{nc}_1, \text{nc}_2$	$\text{wait}_1, \text{nc}_2$	$\text{crit}_1, \text{nc}_2$	$\text{crit}_1, \text{wait}_2$	...
------	----------------------------	------------------------------	------------------------------	--------------------------------	-----

mutual exclusion with semaphore and  $AP = \{\text{crit}_1, \text{crit}_2\}$ :



path	$nc_1, nc_2$	$wait_1, nc_2$	$crit_1, nc_2$	$crit_1, wait_2$	...
trace			$\{crit_1\}$	$\{crit_1\}$	...

# Linear vs. branching time

ctlss4.1-2

	linear time	branching time
behavior	path based	state based

# Linear vs. branching time

ctlss4.1-2

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

# Linear vs. branching time

ctlss4.1-2

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

# Linear vs. branching time

ctlss4.1-2

	linear time	branching time
behavior	path based traces	state based computation tree
temporal logic	LTL path formulas	CTL state formulas
model checking	PSPACE-complete $O(\text{size}(T) \cdot 2^{\text{exp}( \Sigma )})$	PTIME $O(\text{size}(T) \cdot 2^{ Q })$

# Linear vs. branching time

ctlss4.1-2

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

# Linear vs. branching time

ctlss4.1-2

	linear time	branching time
behavior	path based traces	state based computation tree
temporal logic	LTL path formulas	CTL state formulas
model checking	PSPACE-complete $O(\text{size}(T)^{\text{Exp}})$	PTIME $O(\text{size}(T)^{\text{Poly}})$
impl. relation	trace inclusion trace equivalence PSPACE-complete	simulation bisimulation PTIME
fairness	can be encoded	requires special treatment



Introduction

Modelling parallel systems

Linear Time Properties

Regular Properties

Linear Temporal Logic (LTL)

**Computation Tree Logic**

syntax and semantics of CTL

expressiveness of CTL and LTL

CTL model checking

fairness, counterexamples/witnesses

CTL<sup>+</sup> and CTL

Equivalences and Abstraction





CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= 1 \text{ U } 2$

CTL (state) formulas:

$::= \text{true} \mid a \mid \text{CTL}_1 \mid \text{CTL}_2 \mid \neg \text{CTL} \mid \text{CTL} \text{ U } \text{CTL}$

CTL path formulas:

$::= \text{CTL}_1 \text{ U } \text{CTL}_2$

eventually:

$\stackrel{\text{def}}{=} (\text{true U } \text{ )}$

CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= 1 \text{ U } 2$

eventually:

$\stackrel{\text{def}}{=} (\text{true U } )$

$\stackrel{\text{def}}{=} ?$

CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= 1 \text{ U } 2$

eventually:

$\stackrel{\text{def}}{=} (\text{true U } )$

$\stackrel{\text{def}}{=} (\text{true U } )$

CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= 1 \text{ U } 2$

eventually:

$\stackrel{\text{def}}{=} (\text{true U } \phi)$

$\stackrel{\text{def}}{=} (\text{true U } \phi)$

always:

$\stackrel{\text{def}}{=} \neg ?$

CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= \phi_1 \text{ U } \phi_2$

eventually:

$\stackrel{\text{def}}{=} (\text{true U } \phi)$

$\stackrel{\text{def}}{=} (\text{true U } \phi)$

always:

$\stackrel{\text{def}}{=} \neg ?$

note:  $\neg \neg \phi$  is no CTL formula



CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= \phi_1 \text{ U } \phi_2$

eventually:

$\stackrel{\text{def}}{=} (\text{true U } \phi)$

$\stackrel{\text{def}}{=} (\text{true U } \phi)$

always:

$\stackrel{\text{def}}{=} \neg \neg \phi$

note:  $\neg \neg \neg \phi$  is no CTL formula

CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= 1 \text{ U } 2$

eventually:

$\stackrel{\text{def}}{=} (\text{true U } \phi)$

$\stackrel{\text{def}}{=} (\text{true U } \phi)$

always:

$\stackrel{\text{def}}{=} \neg \neg \phi$

$\stackrel{\text{def}}{=} \neg \neg \phi$

note:  $\neg \neg \neg \phi$  is no CTL formula

CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= 1 \text{ U } 2$

$=$  next

$\text{U}$  = until

CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= 1 \text{ U } 2$

$\text{U}$  = next

$\neg$  = eventually

$\text{U}$  = until

$\forall$  = always

CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= 1 \text{ U } 2$

mutual exclusion (safety)  $(\neg \text{crit}_1 \vee \neg \text{crit}_2)$

CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= 1 \text{ U } 2$

mutual exclusion (safety)  $(\neg \text{crit}_1 \vee \neg \text{crit}_2)$

Every request will be answered eventually  $\neg \text{eventually } \neg \text{response}$

$(\text{request} \text{ U } \text{response})$

CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= 1 \mid 2 \mid \phi \text{ U } \psi$

mutual exclusion (safety)  $(\neg \text{crit}_1 \wedge \neg \text{crit}_2)$

Every request will be answered eventually  $\Box \phi \text{ U } \psi$

$(\text{request} \text{ U } \text{response})$

traffic lights

$(\text{yellow} \text{ U } \text{red})$

CTL (state) formulas:

$::= \text{true} \mid a \mid 1 \mid 2 \mid \neg \phi$

CTL path formulas:

$::= 1 \mid 2 \mid \phi \text{ U } \psi$

mutual exclusion (safety)  $(\neg \text{crit}_1 \vee \neg \text{crit}_2)$

Every request will be answered eventually  $\Box \phi \text{ U } \psi$

$(\text{request} \text{ U } \text{response})$

traffic lights

$(\text{yellow} \vee \text{red})$

reset possibility

$\text{start}$



CTL (state) formulas:

$::= \text{true} \mid a \mid \neg a \mid \phi_1 \text{ U } \phi_2 \mid \phi_1 \text{ W } \phi_2$

CTL path formulas:

$::= \phi_1 \text{ U } \phi_2$

mutual exclusion (safety)  $(\neg \text{crit}_1 \vee \neg \text{crit}_2)$

Every request will be answered eventually  $\bigvee_{i \in \mathbb{N}} \text{request}_i \text{ U } \text{response}_i$

traffic lights  $(\text{request} \text{ U } \text{response})$

reset possibility  $(\text{yellow} \text{ U } \text{red})$

unconditional process fairness  $\text{start}$

unconditional process fairness  $\text{crit}_1 \text{ U } \text{crit}_2$

# Example: 15-puzzle

ct1ss4.1-5

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

?

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

# Example: 15-puzzle

ct1ss4.1-5

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

?

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

transition system has  $16!$   ~~$2^{16}$~~   $10^{13}$  states

# Example: 15-puzzle

ct1ss4.1-5

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

?

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

transition system has **16!** ~~2~~ ~~16!~~  $2^{16}$  states

states:	game configurations
transitions:	legal moves

# Example: 15-puzzle

ct1ss4.1-5

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

?

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

~~Pro~~ transition system has **16!** ~~Pro~~  $2^{14}$  states

~~Pro~~ representation as parallel system:

*left up down right*

with shared variables ~~Pro~~  $[i]$  for  $i = 1, \dots, 16$

# Example: 15-puzzle

ct1ss4.1-5

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

?

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

transition system has  $16!$  states

representation as parallel system:

*left up down right*

with shared variables  $[i]$  for  $i = 1, \dots, 16$

CTL specification:

$\forall i \in 1 \dots 15$  *piece i* on  $[i]$

# Example: 15-puzzle

ct1ss4.1-5

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

?

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

transition system has  $16!$  ~~2~~  $10^{13}$  states

representation as parallel system:

*left up down right*

with shared variables ~~id~~  $[i]$  for  $i = 1, \dots, 16$

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

*piece i* on ~~id~~  $[i]$

1 / 15





defn a satisfaction relation  $\models$  for CTL formulas  
over  $AP$  and a given TS  $T = (S, Act, S_0, AP, L)$

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

~~Def~~ a satisfaction relation  $\models$  for CTL formulas over **AP** and a given TS  **$T = (S, Act, S_0, AP, L)$**  without terminal states

~~Def~~ interpretation of **state formulas** over the **states**

~~Def~~ interpretation of **path formulas** over the **paths**  
(infinite path fragments)

for infinite path fragment  $= s_0 s_1 s_2 \dots$ :

$$\models \text{true}$$

$$\models a \quad \text{i} \quad s_0 \models a, \text{i.e., } a \in L(s_0)$$

$$\models \phi_1 \wedge \phi_2 \quad \text{i} \quad \models \phi_1 \text{ and } \models \phi_2$$

$$\models \neg \phi \quad \text{i} \quad \not\models \phi$$

$$\models \text{next } \phi \quad \text{i} \quad \text{su } x(, 1) = s_1 s_2 s_3 \dots \models \phi$$

$$\models \phi_1 \text{ U } \phi_2 \quad \text{i} \quad \text{there exists } j \geq 0 \text{ such that}$$

$$\text{su } x(, j) = s_j s_{j+1} s_{j+2} \dots \models \phi_2 \text{ and}$$

$$\text{su } x(, k) = s_k s_{k+1} s_{k+2} \dots \models \phi_1 \text{ for } 0 \leq k < j$$



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

Let  $\rho = s_0 s_1 s_2 \dots$  be an infinite path fragment.

$$\models \quad \text{ i } \quad s_1 \models$$

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

$$\models \varphi \text{ at } i \iff s_i \models \varphi$$

$$\models \varphi_1 \mathbf{U} \varphi_2 \text{ at } i \iff \text{there exists } j \geq 0 \text{ such that}$$

$$s_j \models \varphi_2$$

$$s_k \models \varphi_1 \text{ for } 0 \leq k < j$$



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

$$\models \quad i \quad s_1 \models$$

$$\models \quad 1 \text{ U } 2 \quad i \quad \text{there exists } j \geq 0 \text{ such that}$$

$$s_j \models 2$$

$$s_k \models 1 \text{ for } 0 \leq k < j$$

semantics of derived operators:

$$\models \quad i \quad \text{there exists } j \geq 0 \text{ with } s_j \models$$

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

$$\models \quad i \quad s_1 \models$$

$$\models \quad 1 \text{ U } 2 \quad i \quad \text{there exists } j \geq 0 \text{ such that}$$

$$s_j \models 2$$

$$s_k \models 1 \text{ for } 0 \leq k < j$$

semantics of derived operators:

$$\models \quad i \quad \text{there exists } j \geq 0 \text{ with } s_j \models$$

$$\models \quad i \quad \text{for all } j \geq 0 \text{ we have: } s_j \models$$



$$s \not\models \textit{true}$$

$s \models \text{true}$

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

$s \models \text{true}$

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

$s \models 1 \wedge 2$                       i     $s \models 1$  and  $s \models 2$

$$s \models \text{true}$$

$$s \models a \quad \text{iff} \quad a \in L(s)$$

$$s \models \phi_1 \wedge \phi_2 \quad \text{iff} \quad s \models \phi_1 \text{ and } s \models \phi_2$$

$$s \models \neg \phi \quad \text{iff} \quad s \not\models \phi$$

$$s \not\models \quad \text{i} \quad \text{there is a path} \quad \text{Paths}(s)$$



$$s \models \text{true}$$

$$s \models a \quad \text{i} \quad a \in L(s)$$

$$s \models \varphi_1 \wedge \varphi_2 \quad \text{i} \quad s \models \varphi_1 \text{ and } s \models \varphi_2$$

$$s \models \neg \varphi \quad \text{i} \quad s \not\models \varphi$$

$$s \models \exists x. \varphi \quad \text{i} \quad \text{there is a path } P \in \text{Paths}(s) \text{ s.t. } P \models \varphi$$

$$s \models \forall x. \varphi \quad \text{i} \quad \text{for each path } P \in \text{Paths}(s): P \models \varphi$$

$$s \models \text{true}$$

$$s \models a \quad \text{i} \quad a \in L(s)$$

$$s \models \phi_1 \wedge \phi_2 \quad \text{i} \quad s \models \phi_1 \text{ and } s \models \phi_2$$

$$s \models \neg \phi \quad \text{i} \quad s \not\models \phi$$

$$s \models \exists x. \phi \quad \text{i} \quad \text{there is a path } \gamma \in \text{Paths}(s) \text{ s.t. } \gamma \models \phi$$

$$s \models \forall x. \phi \quad \text{i} \quad \text{for each path } \gamma \in \text{Paths}(s): \gamma \models \phi$$

satisfaction set for state formula  $\phi$ :

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



satisfaction of state formulas over a TS  $T$ :

$$T \models \varphi \text{ i } S_0 \text{ Sat}( \varphi )$$

where  $S_0$  is the set of initial states

recall:  $\text{Sat}( \varphi ) = \{ s \in S : s \models \varphi \}$

satisfaction of state formulas over a TS  $T$ :

$$T \models \varphi \quad \text{iff} \quad S_0 \models \varphi$$

$$\text{iff} \quad s_0 \models \varphi \quad \text{for all initial states } s_0 \text{ of } T$$

where  $S_0$  is the set of initial states

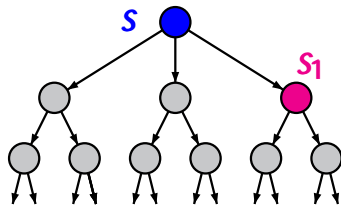
recall:  $Sat(\varphi) = \{s \in S : s \models \varphi\}$



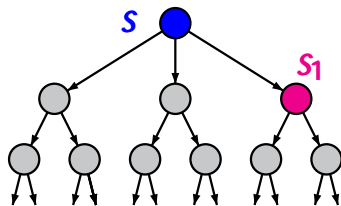
$$s \not\models \quad \text{ i } \quad \text{there exists} \quad = s s_1 s_2 \dots \quad \text{Paths}(s) \\ \text{ s.t. } \not\models$$

## Semantics of the next operator

ct1ss4.1-8

$$s \not\models \text{ i there exists } = s s_1 s_2 \dots \text{ Paths}(s) \\ \text{ s.t. } \not\models , \text{ i.e., } s_1 \not\models$$

$$Post(s) \quad Sat( ) =$$

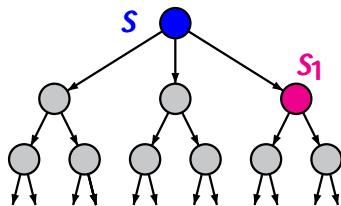
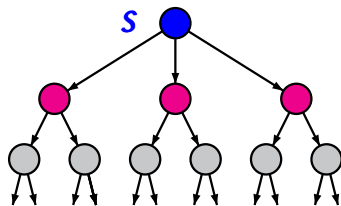


$$\begin{aligned}
 s \not\models & \quad \text{i} \quad \text{there exists } = s \, s_1 \, s_2 \, \dots \, \text{Paths}(s) \\
 & \quad \text{s.t. } \not\models \quad , \text{ i.e., } s_1 \not\models \\
 s \not\models & \quad \text{i} \quad \text{for all } = s \, s_1 \, s_2 \, \dots \, \text{Paths}(s): \\
 & \quad \not\models
 \end{aligned}$$

$$Post(s) \quad Sat( ) =$$

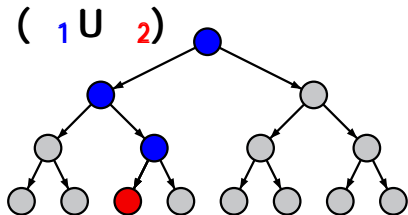
## Semantics of the next operator

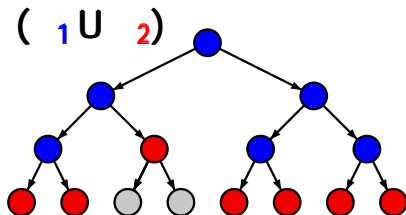
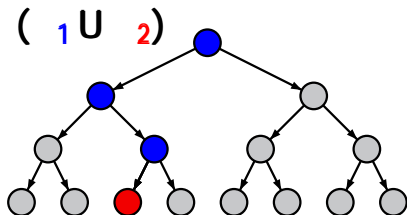
ct1ss4.1-8

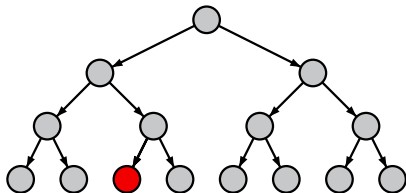
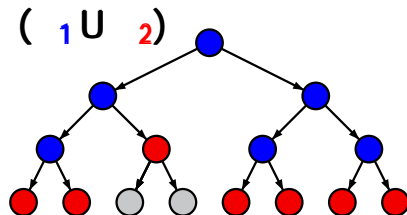
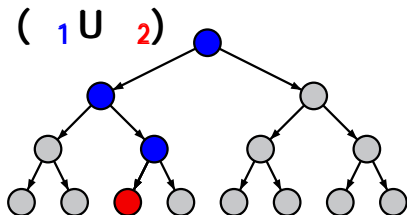
$$s \not\models \text{ i there exists } = s s_1 s_2 \dots \text{ Paths}(s) \\ \text{ s.t. } \not\models , \text{ i.e., } s_1 \not\models$$
$$s \not\models \text{if } \varphi \text{ then } \psi \text{ else } \chi \text{ fi} \quad \text{iff} \quad \text{for all } s = s_1 s_2 \dots \text{Paths}(s):$$

$$s \models \psi \text{ if } s_1 \models \varphi, \text{ i.e., } s_1 \models \varphi \Rightarrow s_1 \models \psi$$

$$Post(s) \quad Sat( ) =$$


*Post(s)*     *Sat( )*

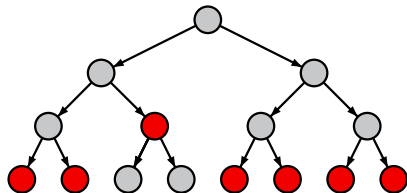
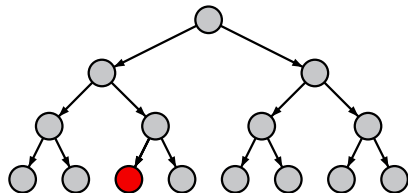
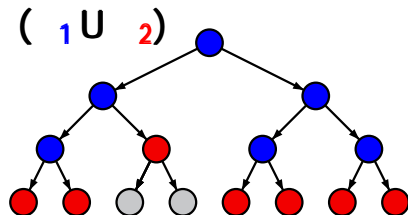
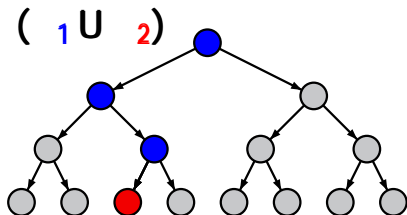


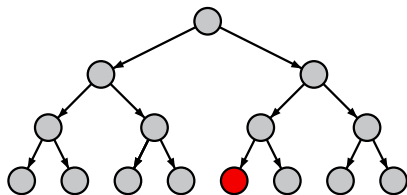
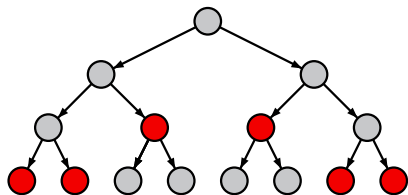


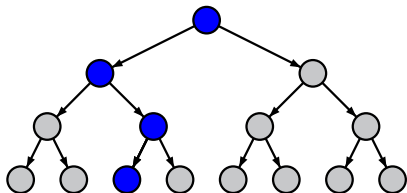
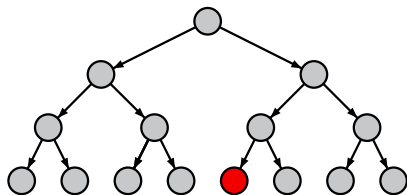
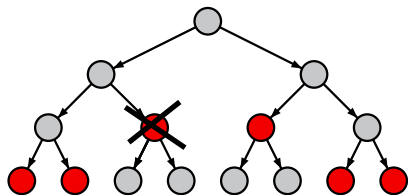


## Semantics of until and eventually

ctlss4.1-9



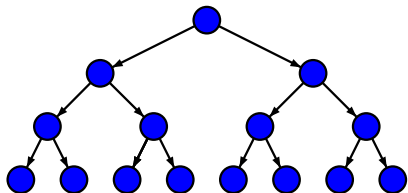
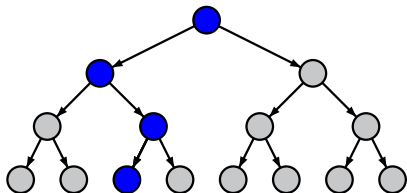
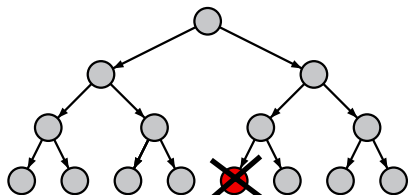
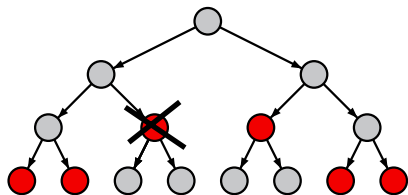


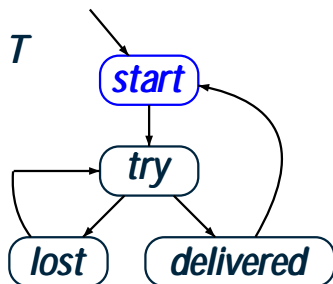
$\not\models$ 

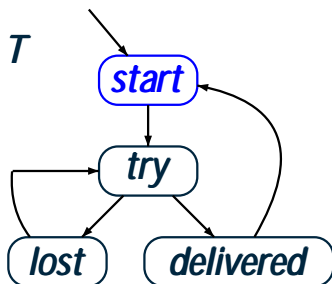


ctlss4.1-10

~~£00~~

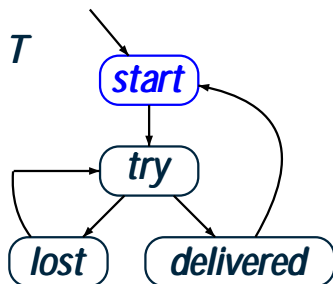






CTL formula

= *start*

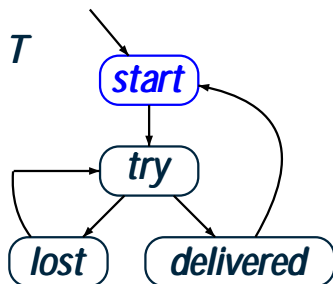


CTL formula

=

*start*

$Sat(\text{start}) = ?$

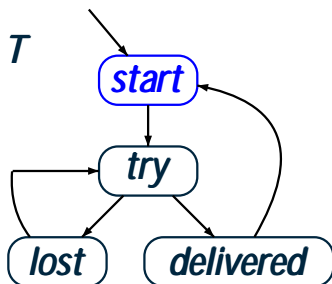


CTL formula

=

*start*

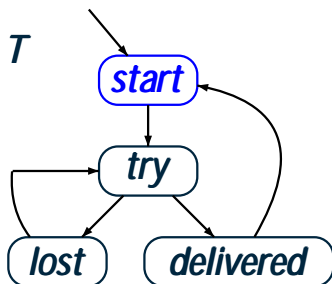
$Sat(\text{start}) = \text{start}, \text{delivered}$



CTL formula

$$= \text{start} = (\text{start} \text{ delivered})$$

$$\text{Sat}(\text{start}) = \text{start}, \text{delivered}$$

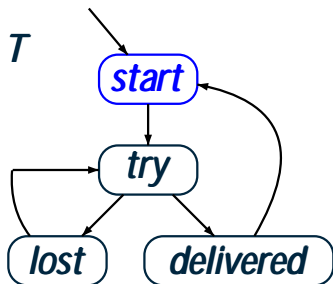


CTL formula

=     *start*     =     (*start*   *delivered*)

$Sat( \text{ } \textit{start} ) = \textit{start}, \textit{delivered}$

$Sat( \text{ } ) =$



$T \not\models \textit{start}$

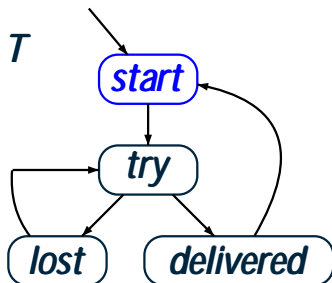
CTL formula

$= \textit{start} = (\textit{start} \textit{ delivered})$

$\textit{Sat}(\textit{start}) = \textit{start}, \textit{delivered}$

$\textit{Sat}(\ ) =$





$T \not\models \text{start}$

infinitely often  $\text{start}$

CTL formula

$= \text{start} = (\text{start } \text{delivered})$

$\text{Sat}(\text{start}) = \text{start}, \text{delivered}$

$\text{Sat}(\ ) =$



If  $s$  is a state in a TS and  $a \in AP$  then:

$$s \not\models_{\text{CTL}} a$$

i for all paths  $= s_0 s_1 s_2 \dots \in \text{Paths}(s)$ :

$$i \geq 0. \text{ s.t. } s_i \not\models a$$

If  $s$  is a state in a TS and  $a$   $AP$  then:

$$s \not\models_{\text{CTL}} a$$

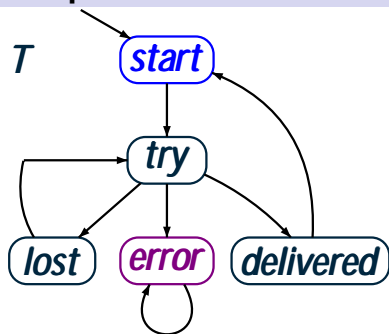
i for all paths  $= s_0 s_1 s_2 \dots$   $Paths(s)$ :

$$i \quad 0. \quad \text{s.t.} \quad s_i \not\models a$$

$$i \quad s \not\models_{\text{LTL}} a$$

# Example: CTL semantics

ctlss4.1-16

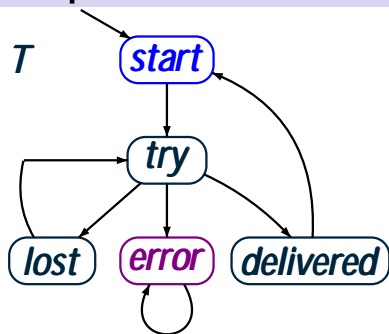


$T \models \neg \text{start} ?$

$1 = \neg \text{start}$

# Example: CTL semantics

ctlss4.1-16



$T \models \neg \text{start} ?$

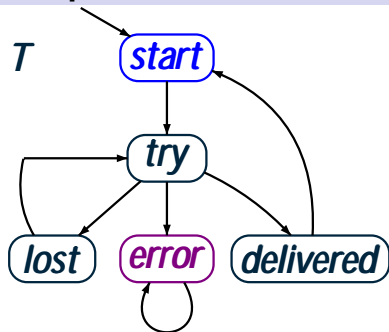
1 =

$\neg \text{start}$

$\text{Sat}(\neg \text{start}) = \text{error}$

# Example: CTL semantics

ctlss4.1-16



$T \models \neg \text{start} ?$

1 =

$\neg \text{start}$

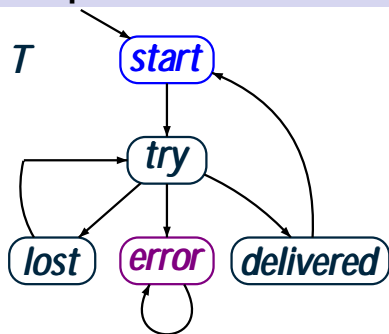
error

$\text{Sat}(\neg \text{start}) = \text{error}$

$\text{Sat}(\neg \text{start}) = ?$

# Example: CTL semantics

ctlss4.1-16



$T \models \neg \text{start}$

1 =

$\neg \text{start}$

error

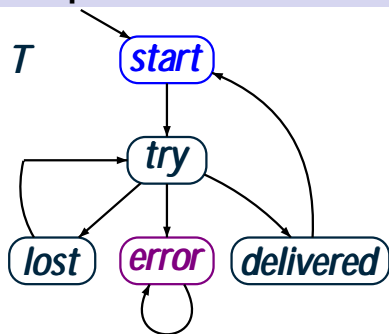
$\text{Sat}(\neg \text{start}) = \text{error}$

$\text{Sat}(\neg \text{start}) = \text{Sat}(\text{error}) = \text{all states}$



# Example: CTL semantics

ctlss4.1-16



2 =

~~$\forall$~~   $\text{start}$

$T \models$

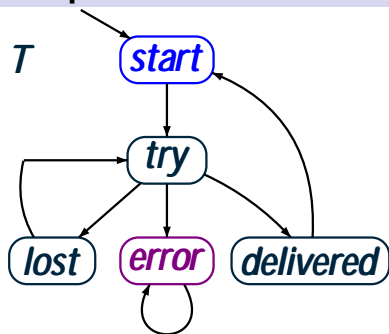
~~$\forall$~~   $\text{start}$

$T \models$

~~$\forall$~~   $\text{start} ?$

# Example: CTL semantics

ctlss4.1-16



$T \models \neg \text{start}$

$T \models \neg \text{start} ?$

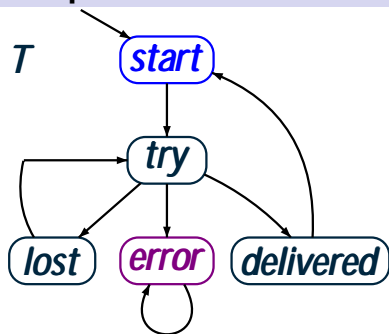
2 =

$\neg \text{start}$

$\text{Sat}(\neg \text{start}) = \{\text{error}\}$

# Example: CTL semantics

ctlss4.1-16



$T \models$

~~$\neg$~~   $\text{start}$

$T \models$

~~$\neg$~~   $\text{start} ?$

2 =

~~$\neg$~~   $\text{start}$

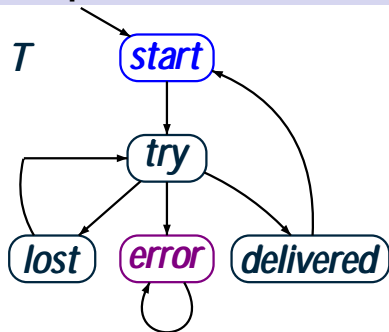
error

$\text{Sat}(\text{ } \neg \text{start}) = \{\text{error}\}$

$\text{Sat}(\text{ } \neg \text{start}) = \{\text{error}, \text{try}\}$

# Example: CTL semantics

ctlss4.1-16



$T \models \neg \text{start}$

$T \models \neg \text{start} ?$

2 =  $\neg \text{start}$

(error try)

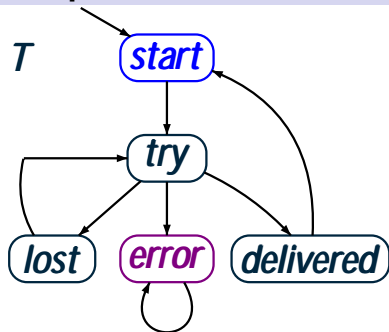
$\text{Sat}(\neg \text{start}) = \{\text{error}\}$

$\text{Sat}(\neg \text{start}) = \{\text{error}, \text{try}\}$

$\text{Sat}(\neg \text{start}) = ?$

# Example: CTL semantics

ctlss4.1-16



$T \models \neg \text{start}$

$T \models \neg \text{start}$

2 =  $\neg \text{start}$

(error try)

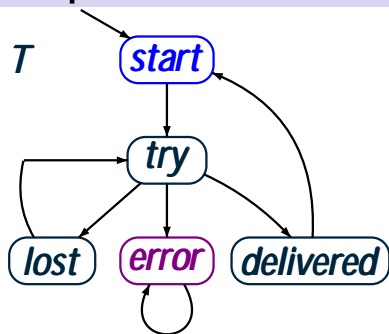
$\text{Sat}(\neg \text{start}) = \{\text{error}\}$

$\text{Sat}(\neg \text{start}) = \{\text{error}, \text{try}\}$

$\text{Sat}(\neg \text{start}) = \{\text{error}, \text{lost}, \text{start}\}$

# Example: CTL semantics

ctlss4.1-16



3 =

~~$\forall$~~   $\text{start}$

$T \models$

~~$\forall$~~   $\text{start}$

$T \models$

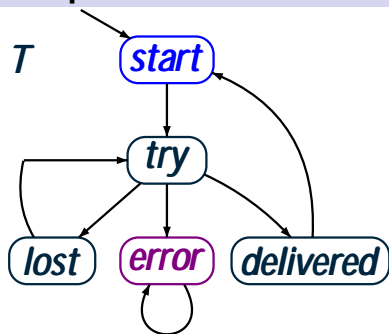
~~$\forall$~~   $\text{start}$

$T \models$

~~$\forall$~~   $\text{start} ?$

# Example: CTL semantics

ctlss4.1-16



$T \models \neg \text{start}$

$T \models \neg \text{start}$

$T \models \neg \text{start} ?$

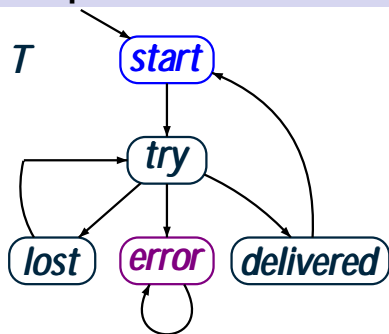
3 =

$\neg \text{start}$

$\text{Sat}(\neg \text{start}) = \{\text{error}\}$

# Example: CTL semantics

ctlss4.1-16



$T \models$

~~$\neg$~~ start

$T \models$

~~$\neg$~~ start

$T \models$

~~$\neg$~~ start ?

3 =

~~$\neg$~~ start

error

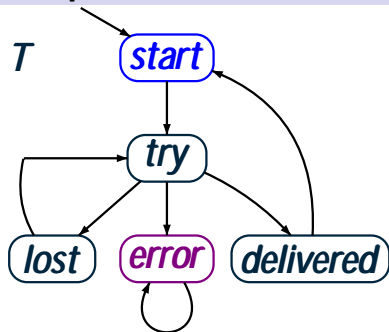
$Sat(\neg start) = \{error\}$

$Sat(\neg start) = \{error\}$



# Example: CTL semantics

ctlss4.1-16



$T \models \neg \text{start}$

$T \models \neg \text{start}$

$T \models \neg \text{start} ?$

3 =  $\neg \text{start}$  error

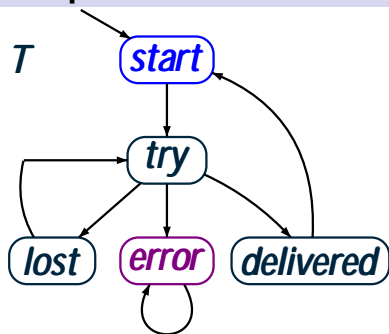
$\text{Sat}(\neg \text{start}) = \{\text{error}\}$

$\text{Sat}(\neg \text{start}) = \{\text{error}\}$

$\text{Sat}(\neg \text{start}) = ?$

# Example: CTL semantics

ctlss4.1-16



$$T \models \neg \text{start}$$

$$T \models \neg \text{start}$$

$$T \models \neg \text{start}$$

$$3 = \neg \text{start} \quad \boxed{\text{error}}$$

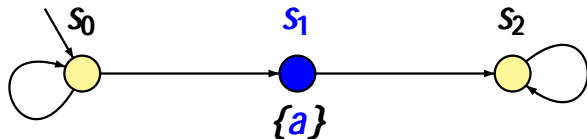
$$\text{Sat}(\neg \text{start}) = \{\text{error}\}$$

$$\text{Sat}(\neg \text{start}) = \{\text{error}\}$$

$$\text{Sat}(\neg \text{start}) = \{\text{error}, \text{try}\}$$

# Example: CTL semantics

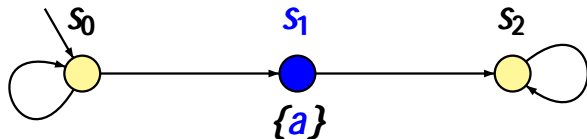
ctlss4.1-17



does  $T \models \neg a$  hold ?

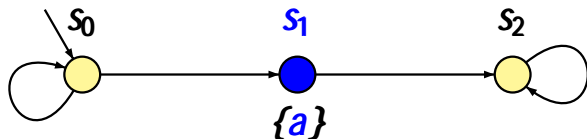
# Example: CTL semantics

ctlss4.1-17



does  $T \models \neg \mathcal{L}^a$  hold ?

answer: no



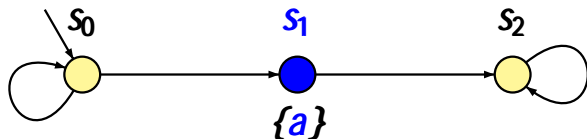
does  $T \models \neg P_{\infty} a$  hold ?

answer: no

$$\text{Sat}(\neg P_{\infty} a) = \{s_2\}$$

# Example: CTL semantics

ctlss4.1-17



does  $T \models \neg \mathcal{L}^a$  hold ?

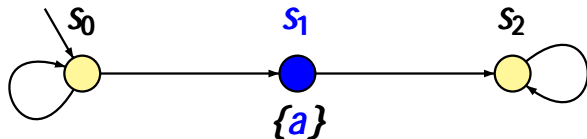
answer: no

$$\text{Sat}(\neg \mathcal{L}^a) = \{s_2\}$$

$$\text{Sat}(\mathcal{L}^a) = \{s_2, s_1\}$$

# Example: CTL semantics

ctlss4.1-17



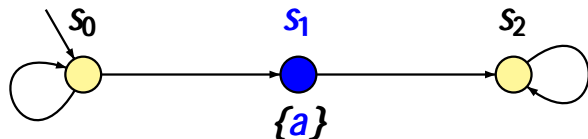
does  $T \models \neg \mathcal{L}^{a}$  hold ?

answer: no

does  $T \models \neg \mathcal{L}^{a}$  hold ?

# Example: CTL semantics

ctlss4.1-17



does  $T \models \neg \mathcal{L}^a$  hold ?

answer: no

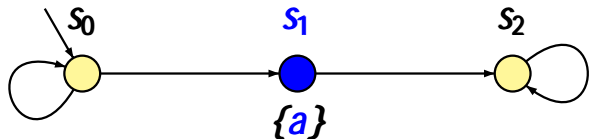
does  $T \models \mathcal{L}^a$  hold ?

answer: yes



# Example: CTL semantics

ctlss4.1-17



does  $T \models \neg \mathcal{L}^a$  hold ?

answer: no

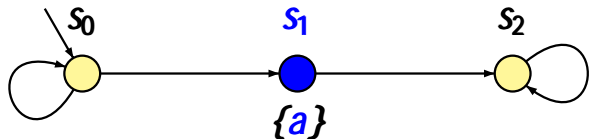
does  $T \models \mathcal{L}^a$  hold ?

answer: yes

$$\text{Sat}(\mathcal{L}^a) = \{s_0, s_1, s_2\}$$

# Example: CTL semantics

ctlss4.1-17



does  $T \models \neg \text{true} \wedge a$  hold ?

answer: no

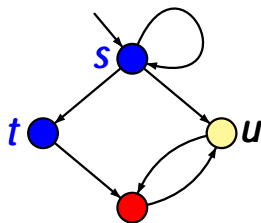
does  $T \models \neg \text{true} \wedge a$  hold ?

answer: yes

$$\begin{aligned} \text{Sat}(\neg \text{true} \wedge a) &= \{s_0, s_1, s_2\} \\ \text{Sat}(\neg \text{true} \wedge a) &= \{s_0, s_1, s_2\} \end{aligned}$$

# Example: CTL semantics

ctlss4.1-18



● =  $\{a\}$

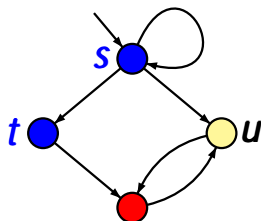
● =  $\{b\}$

● =

$T \models (a \mathbf{U} b) \quad ?$

# Example: CTL semantics

ctlss4.1-18



● =  $\{a\}$

● =  $\{b\}$

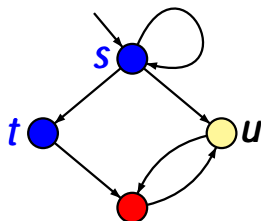
● =

$T \not\models (a \cup b)$

as  $s \not\models (a \cup b)$

# Example: CTL semantics

ctlss4.1-18



● =  $\{a\}$

● =  $\{b\}$

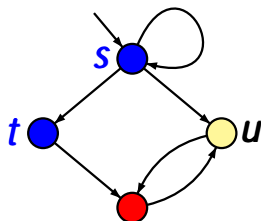
● =

$T \not\models (a \cup b)$

as  $s s s \dots \not\models (a \cup b)$

# Example: CTL semantics

ctlss4.1-18



● =  $\{a\}$

● =  $\{b\}$

● =

$T \models (a \cup b)$

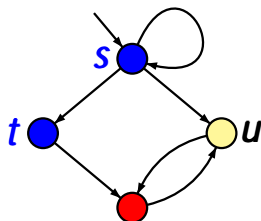
as  $s s s \dots \models (a \cup b)$

$T \models ((a) \cup b)$

?

# Example: CTL semantics

ctlss4.1-18



● = {*a*}

● = {*b*}

● =

$T \not\models (a \cup b)$

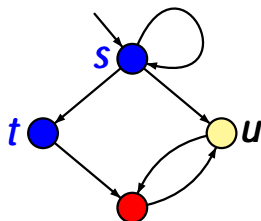
as  $s s s \dots \not\models (a \cup b)$

$T \models ((a) \cup b)$

as  $t \models a, u \models a$

# Example: CTL semantics

ctlss4.1-18



● = {*a*}

● = {*b*}

● =

$T \not\models (a \cup b)$

as  $s s s \dots \not\models (a \cup b)$

$T \not\models ((a) \cup b)$

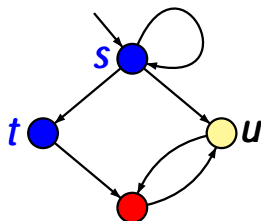
as  $t \models a, u \not\models a$

$T \models (a \cup (a \cup b))$  ?



# Example: CTL semantics

ctlss4.1-18



● = {*a*}

● = {*b*}

● =

$T \not\models (a \cup b)$

as  $s s s \dots \not\models (a \cup b)$

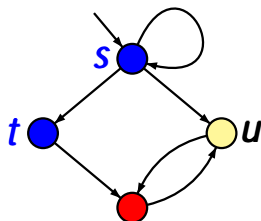
$T \not\models ((a) \cup b)$

as  $t \models a, u \models a$

$T \not\models (a \cup (\neg a \cup b))$

# Example: CTL semantics

ctlss4.1-18



● = {a}

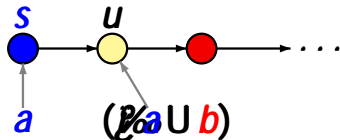
● = {b}

● =

$T \not\models (a \cup b)$  as  $s s s \dots \not\models (a \cup b)$

$T \not\models ((a) \cup b)$  as  $t \models a$ ,  $u \not\models a$

$T \models (a \cup (\neg a \cup b))$



$\models a \cup (\neg a \cup b)$

Let  $T$  be a transition system and  $\phi$  a CTL formula.  
Is the following statement correct ?

if  $T \models \neg \phi$  then  $T \models \phi$

Let  $T$  be a transition system and  $\phi$  a CTL formula.  
Is the following statement correct ?

if  $T \models \phi$  then  $T \models \neg \phi$

*answer:* no

Let  $T$  be a transition system and  $\varphi$  a CTL formula.  
Is the following statement correct ?

if  $T \models \varphi$  then  $T \models \varphi$

answer: no

$T \models \varphi$  i  $s_0 \models \varphi$  for all initial states  $s_0$

---

Let  $T$  be a transition system and  $\varphi$  a CTL formula.  
Is the following statement correct ?

if  $T \models \neg \varphi$  then  $T \models \varphi$

answer: no

$T \models \varphi$  i  $s_0 \models \varphi$  for all initial states  $s_0$

---

$T \models \neg \varphi$  i there exists an initial state  $s_0$  with  
 $s_0 \models \neg \varphi$

Let  $T$  be a transition system and  $\phi$  a CTL formula.  
Is the following statement correct ?

if  $T \models \neg \phi$  then  $T \models \neg \phi$

answer: no

$T \models \phi$  i  $s_0 \models \phi$  for all initial states  $s_0$

---

$T \models \neg \phi$  i there exists an initial state  $s_0$  with  
 $s_0 \models \neg \phi$

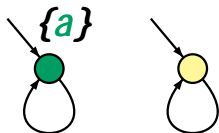
i there exists an initial state  $s_0$  with  
 $s_0 \models \phi$

Let  $T$  be a transition system and  $\phi$  a CTL formula.  
Is the following statement correct ?

if  $T \models \neg \phi$  then  $T \models \phi$

answer: no

transition system  $T$  with 2 initial states:



$$T \models a$$

$$T \models \neg \phi$$





*given:* ~~Not~~ a directed graph  $G = (V, E)$

*question:* does  $G$  have a **Hamilton path**, i.e., a path that visits each node exactly once ?

*given:*     ~~No~~ directed graph  $G = (V, E)$

*question:*   does  $G$  have a **Hamilton path**, i.e., a path  
that visits each node exactly once ?

*goal:*   provide an encoding of the Hamilton path problem  
in **CTL**

*given:* ~~Give~~ a directed graph  $G = (V, E)$

*question:* does  $G$  have a **Hamilton path**, i.e., a path that visits each node exactly once ?

*goal:* provide an encoding of the Hamilton path problem in **CTL** by means of a transformation

~~Give~~  
digraph  $G$

~~Give~~ TS  $T_G$   
+ CTL formula

# Hamilton path problem

ctlss4.1-20

*given:* ~~Not~~ a directed graph  $G = (V, E)$

*question:* does  $G$  have a **Hamilton path**, i.e., a path that visits each node exactly once ?

*goal:* provide an encoding of the Hamilton path problem in **CTL** by means of a transformation

<del>Not</del> digraph $G$	<del>Not</del> TS $T_G$ + CTL formula
-------------------------------	--

s.t.  $G$  has a **Hamilton path**  $\iff T_G \models$

~~Note~~  
 digraph  $G$

~~Note~~ TS  $T_G$   
 + CTL formula

s.t.  $G$  has a Hamilton path  $\iff T_G \models$

# CTL-encoding of the Hamilton path problem

ctlss4.1-20

~~Note~~  
digraph  $G$

~~Note~~ TS  $T_G$   
+ CTL formula

s.t.  $G$  has a Hamilton path i  $T_G \models$



digraph  $G$

# CTL-encoding of the Hamilton path problem ctlss4.1-20

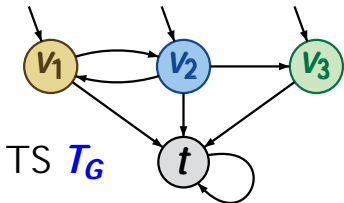
Not  
digraph  $G$

Not TS  $T_G$   
+ CTL formula

s.t.  $G$  has a Hamilton path i  $T_G \models$



digraph  $G$



TS  $T_G$



# CTL-encoding of the Hamilton path problem ctlss4.1-20

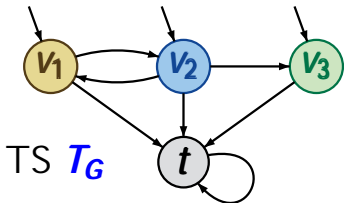
Given  
digraph  $G$

Given TS  $T_G$   
+ CTL formula

s.t.  $G$  has a Hamilton path  $\iff T_G \models$



digraph  $G$



TS  $T_G$

CTL formula

$(v_1$	$(v_2$	$v_3))$	$(v_1$	$(v_3$	$v_2))$
$(v_2$	$(v_1$	$v_3))$	$(v_2$	$(v_3$	$v_1))$
$(v_3$	$(v_1$	$v_2))$	$(v_3$	$(v_2$	$v_1))$

# CTL-encoding of the Hamilton path problem ctlss4.1-20

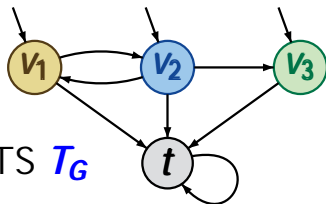
Given  
digraph  $G$

Given TS  $T_G$   
+ CTL formula

s.t.  $G$  has a Hamilton path  $\iff T_G \models$



digraph  $G$



TS  $T_G$

CTL formula = negation of the formula

$(v_1$	$(v_2$	$v_3))$	$(v_1$	$(v_3$	$v_2))$
$(v_2$	$(v_1$	$v_3))$	$(v_2$	$(v_3$	$v_1))$
$(v_3$	$(v_1$	$v_2))$	$(v_3$	$(v_2$	$v_1))$



1      2      i      for all transition systems  $T$ :

$T \not\models 1$        $T \not\models 2$

1 2 i for all transition systems  $T$ :

$$T \models 1 \quad T \models 2$$

quantification over all transition systems  $T$

~~no~~ without terminal states

~~no~~ over  $AP$  if 1 and 2 are CTL formulas over  $AP$

- 1      2      i      for all transition systems  $T$ :  
 $T \models 1$        $T \models 2$
- i      for all transition systems  $T$ :  
 $Sat(1) = Sat(2)$

quantification over all transition systems  $T$

$\forall$  without terminal states

$\forall$  over  $AP$  if 1 and 2 are CTL formulas over  $AP$

- 1      2      i      for all transition systems  $T$ :
- $T \models 1$        $T \models 2$
- i      for all transition systems  $T$ :
- $Sat(1) = Sat(2)$

Examples:

~~$\forall \phi$~~

~~$\forall \phi$~~       )       ~~$\forall \phi$~~   ~~$\exists \phi$~~

- 1      2      i      for all transition systems  $T$ :
- $T \models 1$        $T \models 2$
- i      for all transition systems  $T$ :
- $Sat(1) = Sat(2)$

Examples:

$\neg \phi$

$\neg(\phi \wedge \psi)$        $\neg \phi \vee \neg \psi$

$\vdots$

$\neg \phi$        $\neg \phi$



# Correct or wrong?

ctlss4.1-23

(*a* *b*)      *a*      *b*

# Correct or wrong?

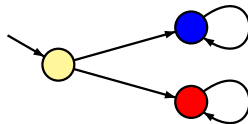
ctlss4.1-23

(*a* *b*)

*a*

*b*

wrong, e.g.,



# Correct or wrong?

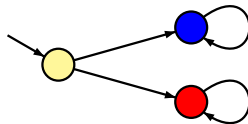
ctlss4.1-23

(*a* *b*)

*a*

*b*

wrong, e.g.,



---

(*a* *b*)

*a*

*b*

# Correct or wrong?

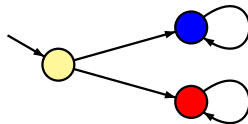
ctlss4.1-23

(*a* *b*)

*a*

*b*

wrong, e.g.,



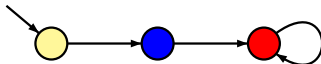
---

(*a* *b*)

*a*

*b*

wrong, e.g.,



# Correct or wrong?

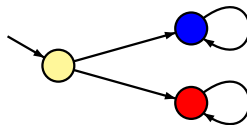
ctlss4.1-23

(*a* *b*)

*a*

*b*

wrong, e.g.,

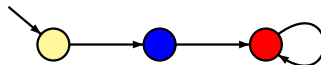


(*a* *b*)

*a*

*b*

wrong, e.g.,



but:

( *1* *2* )

*1*

*2*

( *1* *2* )

*1*

*2*

# Correct or wrong?

ctlss4.1-24

*a*

*a*

# Correct or wrong?

ctlss4.1-24

*a*

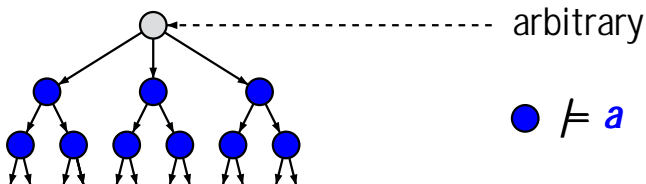
*a*

correct.

*a**a*

correct.

both formulas require computation trees  
of the form:





# Correct or wrong?

ctlss4.1-24

*a*

*a*

correct.

---

*a*

*a*

# Correct or wrong?

ctlss4.1-24

*a*

*a*

correct.

---

*a*

*a*

wrong,

# Correct or wrong?

ctlss4.1-24

*a*

*a*

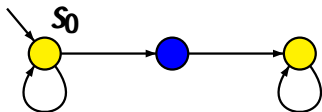
correct.

---

*a*

*a*

wrong, e.g.,



# Correct or wrong?

ctlss4.1-24

*a*

*a*

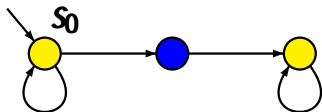
correct.

---

*a*

*a*

wrong, e.g.,



$s_0 \not\models$

*a*

# Correct or wrong?

ctlss4.1-24

*a*

*a*

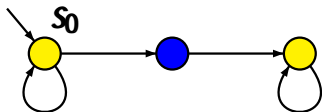
correct.

---

*a*

*a*

wrong, e.g.,



$s_0 \not\models a$

note:  $Sat(a) =$

# Correct or wrong?

ctlss4.1-24

*a*

*a*

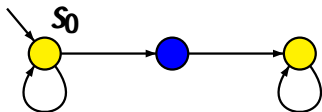
correct.

---

*a*

*a*

wrong, e.g.,



$s_0 \not\models$

*a*

$s_0 \not\models$

*a*

# Correct or wrong?

ctlss4.1-24

*a*

*a*

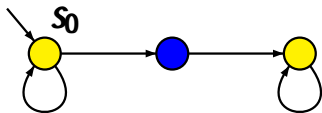
correct.

---

*a*

*a*

wrong, e.g.,



$s_0 \not\models a$

$s_0 \not\models a$

$s_0 \models a$

# Correct or wrong?

ctlss4.1-24

*a*

*a*

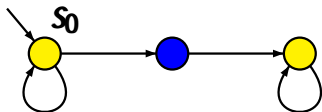
correct.

---

*a*

*a*

wrong, e.g.,



$s_0 \not\models a$

$s_0 \not\models a$

$$\begin{aligned} & s_0 \not\models a \\ = & s_0 s_0 s_0 \dots \not\models a \end{aligned}$$



# Correct or wrong?

ctlss4.1-24

*a*

*a*

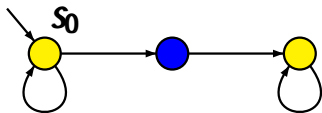
correct.

---

*a*

*a*

wrong, e.g.,



$s_0 \not\models a$

$s_0 \not\models a$

$s_0 \not\models a$

$= s_0 s_0 s_0 \dots \not\models a$

$= s_0 \not\models a$



in LTL:  $W \stackrel{\text{def}}{=} ( U )$

in CTL: ?

in LTL:  $W \stackrel{\text{def}}{=} ( U )$

duality of U and W:

$$\begin{aligned}
 \neg (U) & \equiv (\neg \neg U) & \neg (W) & \equiv (\neg \neg W) \\
 \neg (W) & \equiv (\neg \neg W) & \neg (U) & \equiv (\neg \neg U)
 \end{aligned}$$

in CTL: ?

in **LTL**:  $W \stackrel{\text{def}}{=} ( U )$

duality of **U** and **W**:

$$\neg ( U ) \quad ( \neg W )$$

$$\neg ( W ) \quad ( \neg U )$$

definition of **W** in **CTL** on the basis of duality rules:

$$( W ) \stackrel{\text{def}}{=} \neg ( \neg U )$$

in **LTL**:  $W \stackrel{\text{def}}{=} ( U )$

duality of **U** and **W**:

$$\neg ( U ) \quad ( \neg W )$$

$$\neg ( W ) \quad ( \neg U )$$

definition of **W** in **CTL** on the basis of duality rules:

$$( W ) \stackrel{\text{def}}{=} \neg ( \neg U )$$

$$( W ) \stackrel{\text{def}}{=} \neg ( \neg U )$$

definition of  $W$  in **CTL** on the basis of duality rules:

$$(W) \stackrel{\text{def}}{=} \neg((\neg W) \cup (\neg W))$$

$$(W) \stackrel{\text{def}}{=} \neg((\neg W) \cup (\neg W))$$

definition of  $W$  in **CTL** on the basis of duality rules:

$$(W) \stackrel{\text{def}}{=} \neg((\neg W) \cup (\neg W))$$

$$(W) \stackrel{\text{def}}{=} \neg((\neg W) \cup (\neg W))$$

note that:

$$(W) \quad (U)$$



definition of  $W$  in **CTL** on the basis of duality rules:

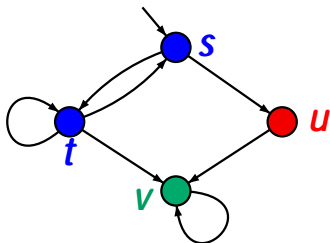
$$(W) \stackrel{\text{def}}{=} \neg((\neg W) \cup (\neg W))$$

$$(W) \stackrel{\text{def}}{=} \neg((\neg W) \cup (\neg W))$$

note that:

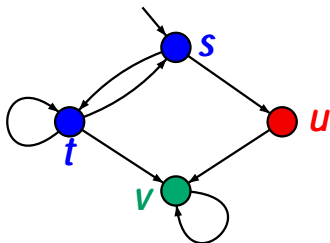
$$(W) \quad (U)$$

$$(W) \quad (U)$$



$T \models (a \text{ W } c) \quad ?$

~~$P \models \{a\}$~~   
 ~~$P \models \{b\}$~~   
 ~~$P \models \{c\}$~~



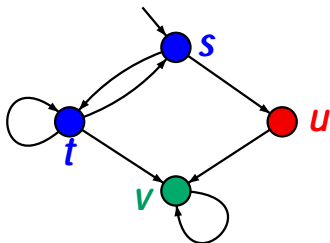
$T \models (a \text{ W } c)$

as  $s \models (a \text{ W } c)$

~~$p \models \{a\}$~~   
 ~~$p \models \{b\}$~~   
 ~~$p \models \{c\}$~~

# Weak until W in CTL

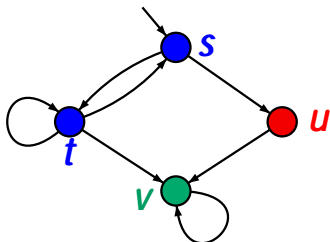
ctlss4.1-21b



$T \not\models (a \text{ W } c)$

as  $s \ s_1 \ s_2 \ \dots \not\models (a \text{ W } c)$

~~$p \models \{a\}$~~   
 ~~$p \models \{b\}$~~   
 ~~$c \models \{c\}$~~



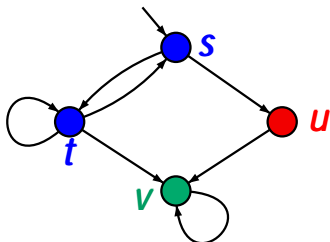
~~$p \models \{a\}$~~   
 ~~$p \models \{b\}$~~   
 ~~$c \models \{c\}$~~

$T \models (a \text{ W } c)$

as  $s \models (a \text{ W } c)$

$T \models (a \text{ W } b)$

?



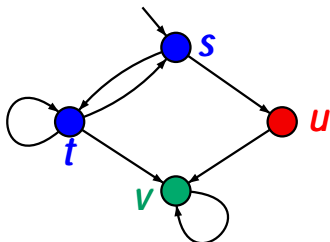
~~$P \models \{a\}$~~   
 ~~$P \models \{b\}$~~   
 ~~$P \models \{c\}$~~

$T \models (a \text{ W } c)$

as  $s \models (a \text{ W } c)$

$T \models (a \text{ W } b)$

as  $s \models a$

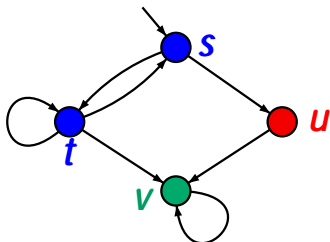


~~$P \models \{a\}$~~   
 ~~$P \models \{b\}$~~   
 ~~$P \models \{c\}$~~

$T \models (a \text{ W } c)$  as  $s \models (a \text{ W } c)$

$T \models (a \text{ W } b)$  as  $s \models a$

$T \models ((b \text{ } c)) \text{ W } (a \text{ } b) \text{ ?}$



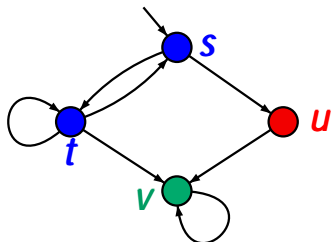
~~$P \models \{a\}$~~   
 ~~$P \models \{b\}$~~   
 ~~$P \models \{c\}$~~

$T \not\models (a \text{ W } c)$  as  $s \not\models (a \text{ W } c)$

$T \not\models (a \text{ W } b)$  as  $s \not\models a$

$T \not\models ((b \text{ } c)) \text{ W } (a \text{ } b)$





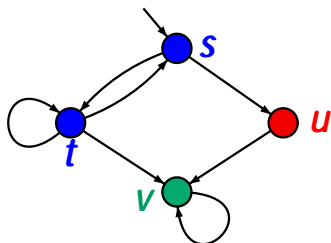
$$\begin{aligned}
 &\cancel{p} \neq \{a\} \\
 &\cancel{p} \neq \{b\} \\
 &\cancel{p} \neq \{c\}
 \end{aligned}$$

$$T \models (a \text{ W } c) \quad \text{as } s \models (a \text{ W } c)$$

$$T \models (a \text{ W } b) \quad \text{as } s \models a$$

$$T \models ((b \text{ } c)) \text{ W } (a \text{ } b)$$

three types of paths:  $(st)$  or  $(st)^+ v$  or  $(st) suv$



$$\begin{aligned}
 &\cancel{p} \models \{a\} \\
 &\cancel{p} \models \{b\} \\
 &\cancel{p} \models \{c\}
 \end{aligned}$$

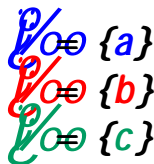
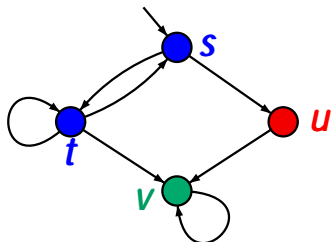
$$T \models (a \text{ W } c) \quad \text{as } s \models (a \text{ W } c)$$

$$T \models (a \text{ W } b) \quad \text{as } s \models a$$

$$T \models ((b \text{ } c)) \text{ W } (a \text{ } b)$$

three types of paths:  $(st)$  or  $(st)^+v$  or  $(st) \text{ } suv$

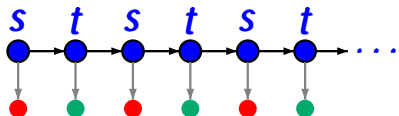
in all three cases:  $\models (b \text{ } c)$



$T \models (a \text{ W } c)$  as  $s \models (a \text{ W } c)$

$T \models (a \text{ W } b)$  as  $s \models a$

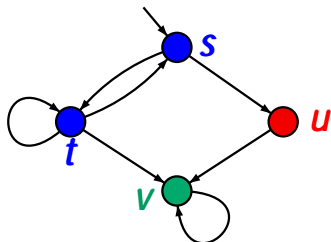
$T \models ((b \text{ } c)) \text{ W } (a \text{ } b)$



$\models (b \text{ } c)$

# Weak until W in CTL

ctlss4.1-21b

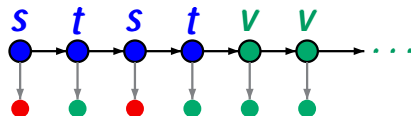


~~$P \models \{a\}$~~   
 ~~$P \models \{b\}$~~   
 $P \models \{c\}$

$T \models (a \text{ W } c)$  as  $s \models (a \text{ W } c)$

$T \models (a \text{ W } b)$  as  $s \models a$

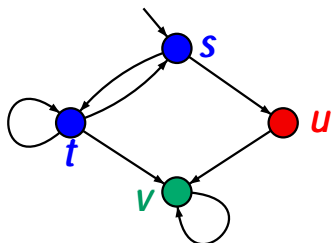
$T \models ((b \text{ } c)) \text{ W } (a \text{ } b)$



$\models (b \text{ } c)$

# Weak until W in CTL

ctlss4.1-21b

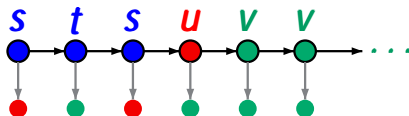


~~$P \models \{a\}$~~   
 ~~$P \models \{b\}$~~   
 $P \models \{c\}$

$T \models (a \text{ W } c)$  as  $s \models (a \text{ W } c)$

$T \models (a \text{ W } b)$  as  $s \models a$

$T \models ((b \text{ } c)) \text{ W } (a \text{ } b)$



$\models (b \text{ } c)$



( U )

( ( U ))

# Expansion laws

ctlss4.1-26

( U )

( ( U ))

( U )

?



# Expansion laws

ctlss4.1-26

( U )

( ( U ))

( U )

( ( U ))

# Expansion laws

ctlss4.1-26

( U )

( ( U ))

( U )

( ( U ))

( U )

( ( U ))

( U )

( ( U ))

( W )

( U )

( ( U ))

( U )

( ( U ))

( W )

( ( W ))

( U )

( ( U ))

( U )

( ( U ))

( W )

( ( W ))

( W )

( ( W ))

( U )

( ( U ))

( U )

( ( U ))

( W )

( ( W ))

( W )

( ( W ))

?

( U )

( ( U ))

( U )

( ( U ))

( W )

( ( W ))

( W )

( ( W ))

( U )

( ( U ))

( U )

( ( U ))

( W )

( ( W ))

( W )

( ( W ))



duality of      and      :

$$\overline{P \vee Q} \equiv \overline{P} \wedge \overline{Q}$$

$$\overline{P \wedge Q} \equiv \overline{P} \vee \overline{Q}$$

# Duality laws

ctlss4.1-27

duality of      and      :

$$\neg \phi \equiv \neg \phi$$

$$\neg \phi \equiv \neg \phi$$

self-duality of      :

$$\neg \phi \equiv \neg \phi \quad \neg \phi \equiv \neg \phi$$

$$\neg \phi \equiv \neg \phi \quad \neg \phi \equiv \neg \phi$$

duality of  $\neg$  and  $\vee$  :

$$\neg(\neg P \vee \neg Q) \equiv P \wedge Q$$

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

self-duality of  $\neg$  :

$$\neg(\neg P) \equiv P$$

$$\neg(\neg P \vee \neg Q) \equiv P \wedge Q$$

duality of **U** and **W**, e.g.:

duality of  $\neg$  and  $\vee$  :

$$\neg(\neg P \vee \neg Q) \equiv P \wedge Q$$

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

self-duality of  $\neg$  :

$$\neg(\neg P) \equiv P$$

$$\neg(\neg P \vee \neg Q) \equiv P \wedge Q$$

duality of **U** and **W**, e.g.:

$$(\neg U) \equiv W$$

duality of  $\mathcal{P}$  and  $\mathcal{Q}$  :

$$\mathcal{P}(\mathcal{Q}) = \mathcal{Q}(\mathcal{P})$$

$$\mathcal{P}(\mathcal{Q}) = \mathcal{Q}(\mathcal{P})$$

self-duality of  $\mathcal{P}$  :

$$\mathcal{P}(\mathcal{P}) = \mathcal{P}$$

$$\mathcal{P}(\mathcal{P}) = \mathcal{P}$$

duality of  $\mathbf{U}$  and  $\mathbf{W}$ , e.g.:

$$(\mathbf{U}) = \mathcal{P}(\mathcal{Q}(\mathcal{P}(\mathcal{Q}(\mathbf{W}(\mathcal{P}(\mathcal{Q}(\mathbf{U})))))$$

duality of  $\mathcal{P}$  and  $\mathcal{Q}$  :

$$\mathcal{P}(\mathcal{Q}) = \mathcal{Q}(\mathcal{P})$$

$$\mathcal{Q}(\mathcal{P}) = \mathcal{P}(\mathcal{Q})$$

self-duality of  $\mathcal{P}$  :

$$\mathcal{P}(\mathcal{P}) = \mathcal{P}$$

$$\mathcal{Q}(\mathcal{Q}) = \mathcal{Q}$$

duality of  $\mathbf{U}$  and  $\mathbf{W}$ , e.g.:

$$(\mathbf{U}) = \mathcal{P}(\mathcal{Q}(\mathbf{W})) \quad \mathbf{W}(\mathcal{P}(\mathcal{Q}))$$

$$\mathcal{P}(\mathcal{Q}(\mathbf{W})) = \mathbf{W}(\mathcal{P}(\mathcal{Q}))$$

duality of  $\neg$  and  $\neg$  :

$$\neg \neg A \equiv A$$

$$\neg \neg A \equiv A$$

self-duality of  $\neg$  :

$$\neg \neg A \equiv A$$

$$\neg \neg A \equiv A$$

duality of  $\cup$  and  $\cap$ , e.g.:

$$(\cup) \quad \neg (\neg A \cap \neg B) \equiv A \cup B$$

$$\neg (\neg A) \equiv A$$

$$\neg (\neg A) \cup (\neg B) \equiv A \cap B$$

# Duality laws

ctlss4.1-27a

duality of  $\mathcal{P}$  and  $\mathcal{Q}$  :

$$\begin{aligned} \mathcal{P}(\mathcal{Q}(x)) &= \mathcal{Q}(\mathcal{P}(x)) \\ \mathcal{Q}(\mathcal{P}(x)) &= \mathcal{P}(\mathcal{Q}(x)) \end{aligned}$$

self-duality of  $\mathcal{P}$  :

$$\begin{aligned} \mathcal{P}(\mathcal{P}(x)) &= x \\ \mathcal{Q}(\mathcal{Q}(x)) &= x \end{aligned}$$

duality of  $\mathbf{U}$  and  $\mathbf{W}$  yields

$$(\mathbf{U} \circ \mathbf{W})(x) = \mathbf{W}(\mathbf{U}(x)) \cup (\mathbf{U} \circ \mathbf{W})(x) = \mathbf{W}(\mathbf{U}(x))$$



# Duality laws

ctlss4.1-27a

duality of  $\neg$  and  $\vee$  :

$$\neg(\neg p \vee \neg q) \equiv p \wedge q$$

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$

self-duality of  $\oplus$  :

$$\neg(\neg p \oplus \neg q) \equiv p \oplus q$$

$$\neg(p \oplus q) \equiv \neg p \oplus \neg q$$

derivation of  $\neg$  from  $\vee$  and  $\oplus$  :

$$(\neg p \vee \neg q) \oplus (\neg p \vee \neg q) \equiv \neg p \vee \neg q$$

$$\neg p \vee \neg q \equiv \neg(p \oplus q)$$

$\neg$  and  $\vee$  are expressible via  $\oplus$ ,  $\wedge$  and  $\neg$

For each **CTL** formula  $\varphi$  there is an equivalent **CTL** formula  $\varphi'$  built by

using only the operators of propositional logic

and the modalities  $\mathbf{E}$ ,  $\mathbf{U}$  and  $\mathbf{F}$ .

For each **CTL** formula  $\phi$  there is an equivalent **CTL** formula  $\phi'$  built by

$\forall$  operators of propositional logic

$\forall$  the modalities  $\mathbf{A}$ ,  $\mathbf{U}$  and  $\mathbf{X}$ .

$$\begin{aligned} ::= & \text{true} \quad a \quad \mathbf{X} \phi \quad \mathbf{A} \phi_1 \mathbf{U} \phi_2 \quad \mathbf{E} \phi_1 \mathbf{U} \phi_2 \end{aligned}$$

For each **CTL** formula  $\phi$  there is an equivalent **CTL** formula  $\phi_{\text{ENF}}$  built by

replacing all occurrences of operators of propositional logic

by the modalities  $\mathbf{E}$ ,  $\mathbf{U}$  and  $\mathbf{A}$ .

$$\begin{aligned} \phi_{\text{ENF}} ::= & \text{true} \mid a \mid \neg \phi_1 \mid \neg \phi_2 \mid \mathbf{E} \phi_1 \\ & \mid (\phi_1 \mathbf{U} \phi_2) \end{aligned}$$

transformation  $\phi \mapsto \phi_{\text{ENF}}$  relies on:

$$\begin{aligned} \mathbf{E}(\neg \phi_1 \mathbf{U} \neg \phi_2) &= \mathbf{E}(\neg(\phi_1 \mathbf{U} \phi_2)) \\ &= \mathbf{E}(\neg(\mathbf{E}(\phi_1 \mathbf{U} \phi_2))) \\ &= \mathbf{E}(\neg \mathbf{E}(\phi_1 \mathbf{U} \phi_2)) \\ &= \mathbf{E}(\mathbf{A}(\neg(\phi_1 \mathbf{U} \phi_2))) \\ &= \mathbf{E}(\mathbf{A}(\neg \phi_1 \wedge \neg \phi_2)) \\ &= \mathbf{E}(\mathbf{A}(\neg \phi_1) \wedge \mathbf{A}(\neg \phi_2)) \\ &= \mathbf{E}(\mathbf{A}(\neg \phi_1) \mathbf{U} \mathbf{A}(\neg \phi_2)) \end{aligned}$$

# Example: $\neg$ -normal form for CTL formula

ctlss4.1-28a

CTL formula	CTL formula in $\neg$ -normal form
	<del><math>\forall</math></del> <del><math>\exists</math></del>
$(\phi_1 \text{ U } \phi_2)$	<del><math>\forall</math></del> ( <del><math>\forall</math></del> $\phi_2 \text{ U } (\phi_1 \text{ } \neg \phi_2)$ ) $\neg \phi_2$

$((\phi_1 \text{ U } \phi_2) \text{ U } \phi_3)$

# Example: -normal form for CTL formula

ctlss4.1-28a

CTL formula	CTL formula in -normal form
	$\neg \phi \rightarrow \neg \psi$
$(\phi_1 \text{ U } \phi_2)$	$\neg \phi \rightarrow (\neg \phi_2 \text{ U } (\neg \phi_1 \rightarrow \neg \phi_2))$

$$\begin{aligned}
 & ((\phi_1 \text{ U } \phi_2) \rightarrow \neg \phi) \\
 & \neg \phi \rightarrow (\neg \phi_2 \text{ U } ((\neg \phi_1 \rightarrow \neg \phi_2) \rightarrow \neg \phi))
 \end{aligned}$$

# Example: -normal form for CTL formula

ctlss4.1-28a

CTL formula	CTL formula in -normal form
	$\neg \phi \vee \neg \psi$
$(\phi_1 \vee \phi_2)$	$\neg \phi \vee (\neg \phi \vee \phi_2 \vee (\neg \phi \vee \phi_1 \vee \neg \phi_2)) \vee \neg \phi \vee \phi_2$

$$\begin{aligned}
 & ((\phi \vee a) \vee \neg \phi) \\
 & \neg \phi \vee (\neg \phi \vee \neg \phi \vee ((\neg \phi \vee a) \vee \neg \phi)) \vee \neg \phi \vee \neg \phi \vee \neg \phi \\
 & \neg \phi \vee (\neg \phi \vee ((\neg \phi \vee a) \vee \neg \phi)) \vee \neg \phi \vee \neg \phi
 \end{aligned}$$

# Example: -normal form for CTL formula

ctlss4.1-28a

CTL formula	CTL formula in -normal form
	$\neg \phi \rightarrow \neg \psi$
$(\phi_1 \text{ U } \phi_2)$	$\neg \phi \rightarrow (\neg \phi_2 \text{ U } (\neg \phi_1 \rightarrow \neg \phi_2))$

$$\begin{aligned}
 & ((\phi_1 \text{ U } \phi_2) \rightarrow \neg \phi) \\
 & \neg \phi \rightarrow (\neg \phi_2 \text{ U } ((\neg \phi_1 \rightarrow \neg \phi_2) \rightarrow \neg \phi)) \quad \neg \phi \rightarrow \neg \phi_2 \\
 & \neg \phi \rightarrow (\neg \phi_2 \text{ U } ((\neg \phi_1 \rightarrow \neg \phi_2) \rightarrow \neg \phi)) \quad \neg \phi \rightarrow \neg \phi_2 \\
 & \neg \phi \rightarrow (\neg \phi_2 \text{ U } ((\neg \phi_1 \rightarrow \neg \phi_2) \rightarrow \neg \phi)) \quad \neg \phi \rightarrow \neg \phi_2
 \end{aligned}$$





~~No~~negation only on the level of literals

~~Do~~ negation only on the level of literals

~~Do~~ uses for each operator its dual

~~Does~~ negation only on the level of literals

~~Does~~ uses for each operator its dual

*true* and *false*

conjunction and disjunction

~~Do~~ negation only on the level of literals

~~Do~~ uses for each operator its dual

*true* and *false*

conjunction and disjunction

existential and universal path quantification

until and weak until

~~Do~~ **negation** only on the level of literals

~~Do~~ uses for each operator (one of) its dual(s)

*true* and *false*

conjunction and disjunction

**existential** and **universal** path quantification

**until** and **weak until**

*alternatively*: until and release

~~no~~ **negation** only on the level of literals

~~no~~ uses for each operator (one of) its dual(s)

*true* and *false*

conjunction and disjunction

**existential** and **universal** path quantification

**until** and **weak until**

*alternatively*: until and release

but no additional operator for      required

syntax of **CTL** formulas in **PNF**:

state formulas:

$::=$  *true* *false* *a* ~~*!a*~~  $\bigwedge_1$   $\bigvee_2$   $\bigwedge_1$   $\bigvee_2$

path formulas:

$::=$   $\bigwedge_1 U_2$   $\bigwedge_1 W_2$



syntax of **CTL** formulas in **PNF**:

state formulas:

$::=$  *true* *false* *a* ~~*!a*~~  $\bigwedge_{i=1}^n \phi_i$   $\bigvee_{i=1}^n \phi_i$

path formulas:

$::=$   $\phi_1 \mathbf{U} \phi_2$   $\phi_1 \mathbf{W} \phi_2$

For each **CTL** formula there is an equivalent **CTL** formula in **PNF**

CTL formula

CTL formula in PNF

CTL formula

CTL formula in PNF

~~*not*~~ *true*     *false*

CTL formula      CTL formula in PNF

~~$\neg$~~  true      false

~~$\neg$~~  1      2)       ~~$\neg$~~  1       ~~$\neg$~~  2      de Morgan laws

CTL formula      CTL formula in PNF

~~$\neg$~~  true      false

~~$\neg$~~  1      2)       ~~$\neg$~~  1       ~~$\neg$~~  2      de Morgan laws

~~$\neg$~~        ~~$\neg$~~

CTL formula      CTL formula in PNF

~~$P$~~  true      false

~~$P$~~  1      2)       ~~$P$~~  1       ~~$i$~~  2      de Morgan laws

~~$P$~~        ~~$P$~~

~~$P$~~        ~~$P$~~

CTL formula      CTL formula in PNF

~~$\neg$~~  true      false

~~$\neg$~~  1      2)       ~~$\neg$~~  1       ~~$\neg$~~  2      de Morgan laws

~~$\neg$~~  1       ~~$\neg$~~  2

~~$\neg$~~  1       ~~$\neg$~~  2

~~$\neg$~~  1 U 2      (( 1  ~~$\neg$~~  2) W ( ~~$\neg$~~  1  ~~$\neg$~~  2))

CTL formula      CTL formula in PNF

~~$P$~~  true      false

~~$P$~~  1      2)       ~~$P$~~  1       ~~$i$~~   ~~$P$~~  2      de Morgan laws

~~$P$~~  0       ~~$P$~~  0

~~$P$~~  0       ~~$P$~~  0

~~$P$~~  0 ( 1 U 2 )      (( ( 1  ~~$P$~~  2 ) W (  ~~$P$~~  1  ~~$P$~~  2 ))

~~$P$~~  0 ( 1 U 2 )      (( ( 1  ~~$P$~~  2 ) W (  ~~$P$~~  1  ~~$P$~~  2 ))



CTL formula      CTL formula in PNF

~~$P$~~  true      false

~~$P$~~  1      2)       ~~$P$~~  1       ~~$i$~~  2      de Morgan laws

~~$P$~~        ~~$P$~~

~~$P$~~        ~~$P$~~

~~$P$~~  ( 1 U 2 )      (( 1  ~~$P$~~  2 ) W (  ~~$P$~~  1       ~~$P$~~  2 ))

~~$P$~~  ( 1 U 2 )      (( 1  ~~$P$~~  2 ) W (  ~~$P$~~  1       ~~$P$~~  2 ))

CTL formula      CTL formula in PNF

~~$P$~~  true      false

~~$P$~~   $\neg$   $(P_1 \wedge P_2)$        ~~$P$~~   $\neg$   $(P_1 \wedge P_2)$       de Morgan laws

~~$P$~~   $\neg$   $(P_1 \vee P_2)$        ~~$P$~~   $\neg$   $(P_1 \vee P_2)$

~~$P$~~   $\neg$   $(P_1 \wedge P_2)$        ~~$P$~~   $\neg$   $(P_1 \wedge P_2)$

~~$P$~~   $\neg$   $(P_1 \vee P_2)$        $(\neg P_2 \wedge (\neg P_1 \vee \neg P_2))$

~~$P$~~   $\neg$   $(P_1 \wedge P_2)$        $(\neg P_2 \wedge (\neg P_1 \vee \neg P_2))$

CTL formula      CTL formula in PNF

~~$P$~~  true      false

~~$P$~~   $\neg$  1      2)       ~~$P$~~   $\neg$  1       ~~$\neg$~~   $\neg$  2      de Morgan laws

~~$P$~~   $\neg$   $\neg$        ~~$P$~~   $\neg$   $\neg$

~~$P$~~   $\neg$   $\neg$        ~~$P$~~   $\neg$   $\neg$

~~$P$~~   $\neg$  ( 1 U 2 )      (  ~~$P$~~   $\neg$  2 W (  ~~$P$~~   $\neg$  1       ~~$P$~~   $\neg$  2 ) )

~~$P$~~   $\neg$  ( 1 U 2 )      (  ~~$P$~~   $\neg$  2 W (  ~~$P$~~   $\neg$  1       ~~$P$~~   $\neg$  2 ) )

... exponential blowup possible ...