Key points of this lecture

Computation Tree Logic (CTL) as a logic to reason about transition systems, in particular about their computation trees.

Grammars for CTL: standard, minimalistic and existential normal form.

Formal semantics of CTL: satisfaction relation for states and paths.

Derived operators and equivalences between CTL formulas.

CTL* as a generalisation of CTL (and LTL).

Lecture 02 - Computation Tree Logic (CTL)

- CTL formulas, grammar and intuition
- Formal semantics of CTL
- Formula equivalences and alternative grammars
- CTL in PRISM
- Beyond CTL: CTL* and LTL

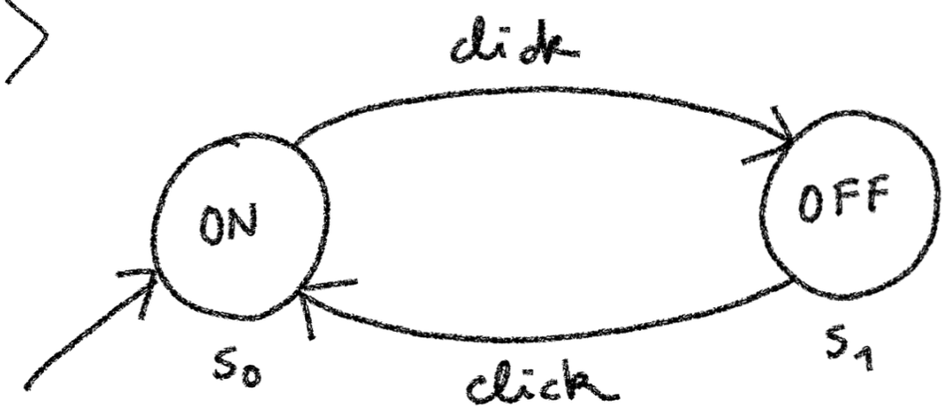
Transition Systems

Def. A transition system is a tuple

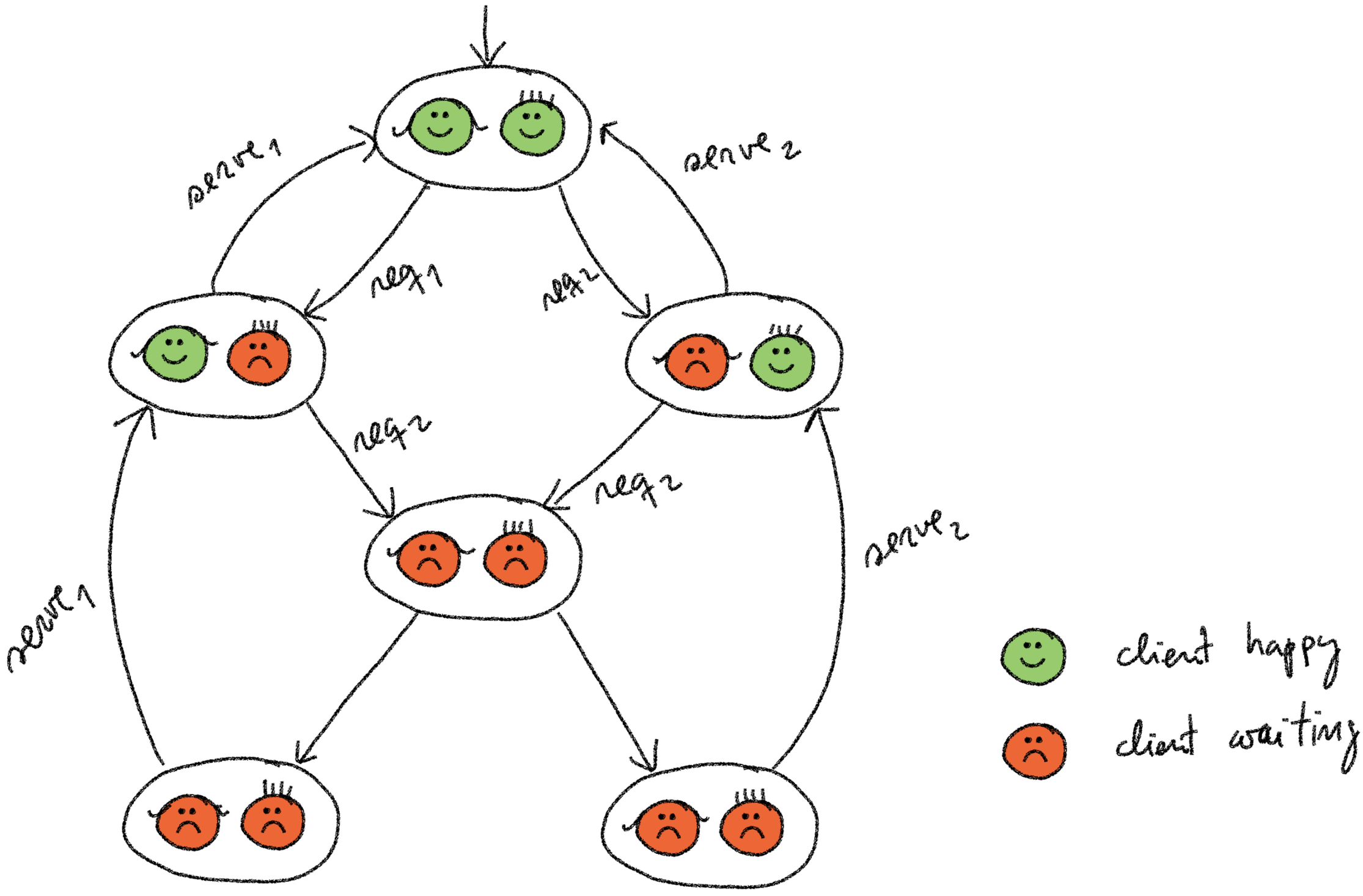
$$\langle S, A, \rightarrow, L, AP, I \rangle$$

such that

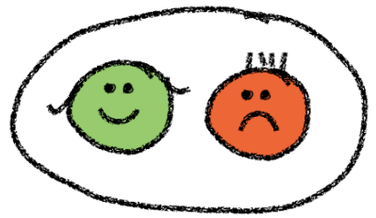
- S is a set of states
- A is a set of "Actions"
- $\rightarrow \subseteq S \times A \times S$ is a set of transitions
- $L : S \longrightarrow 2^{AP}$ is a labelling function
- AP is a finite state of "Atomic Propositions"
- $I \subseteq S$ is the set of initial states



Example: 2 clients, 1 server



Propositional Logic

We can use propositional logic to reason about a state 

$true$

$false = \neg true$

p

$\neg \phi$

$\phi_1 \vee \phi_2$

$\phi_1 \wedge \phi_2$

$\phi_1 \rightarrow \phi_2$

$\ldots$

the state satisfies p

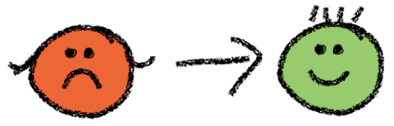
the state does not satisfy ϕ

the state satisfies at least one of ϕ_1, ϕ_2

the state satisfies both of ϕ_1, ϕ_2

if the state satisfies ϕ_1
then it also satisfies ϕ_2

Some examples



Derived operators and grammar for propositional logic

We don't need all operators, some can be derived

$$true$$

$$false = \neg true$$

$$p$$

$$\neg\phi$$

$$\phi_1 \vee \phi_2$$

$$\phi_1 \wedge \phi_2 \equiv \neg(\neg\phi_1 \vee \neg\phi_2)$$

$$\phi_1 \rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$$

We can then use a minimalistic grammar for formulas

$$\phi ::= true \mid p \mid \neg\phi \mid \phi_1 \vee \phi_2$$

Adding "temporal" operators

Propositional logic is not enough if we want to talk about executions.

Whether some or all executions have some property

$\exists\psi$ Some execution satisfies ψ

$\forall\psi$ All executions satisfy ψ

and properties of executions

next $\bigcirc\phi$ the next state of the execution satisfies ϕ

eventually or finally $\Diamond\phi$ some state of the execution satisfies ϕ

always $\Box\phi$ all states of the execution satisfies ϕ

until $\phi_1 \mathsf{U} \phi_2$ ϕ_2 holds in some state of the execution and until then all states satisfy ϕ_1

…

CTL grammar

The grammar CTL extends the one of propositional logic with path quantifiers

$$\phi ::= true \mid p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \exists\psi \mid \forall\psi$$

and path formulas

$$\psi ::= \bigcirc\phi \mid \Diamond\phi \mid \Box\phi \mid \phi_1 \mathsf{U} \phi_2$$

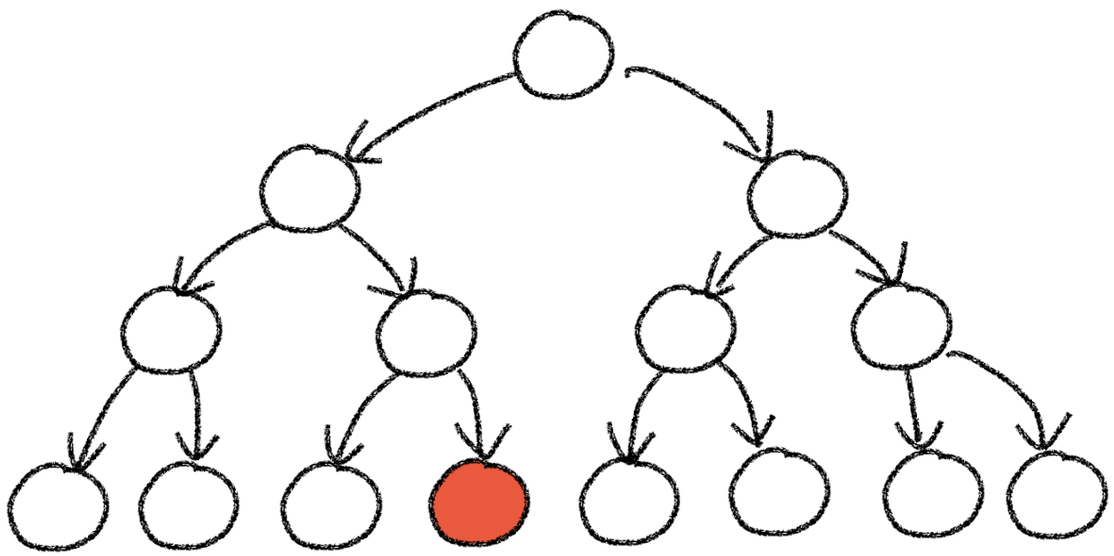
Alternative / equivalent notation

math/LaTeX style		ascii style
$\exists$	E	
$\forall$	A	
$\bigcirc$	X	
$\Diamond$	F	
$\Box$	G	

Intuition of “eventually” and “always” operators

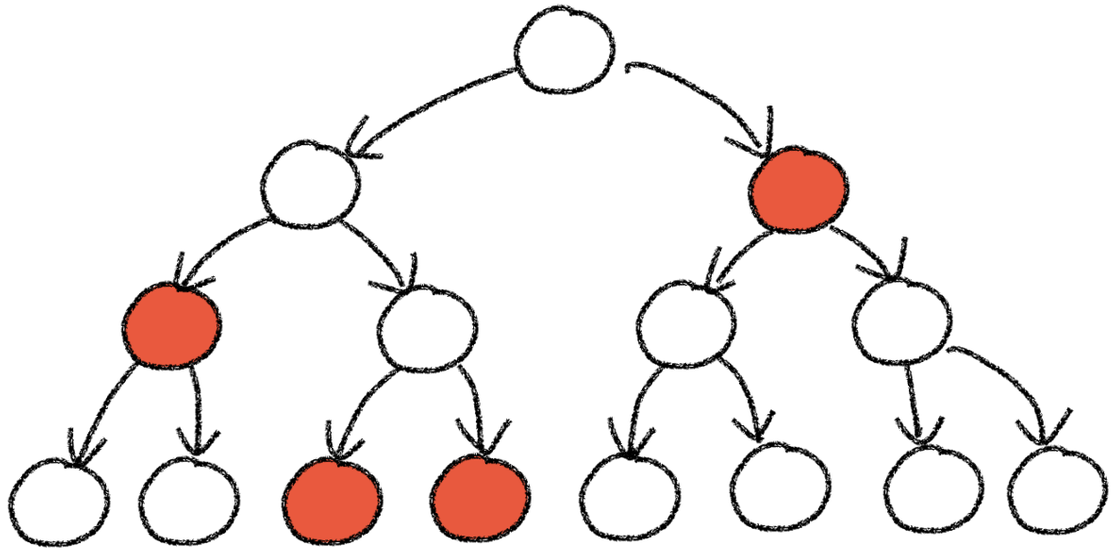
$\exists \Diamond \phi$

"ϕ holds potentially"



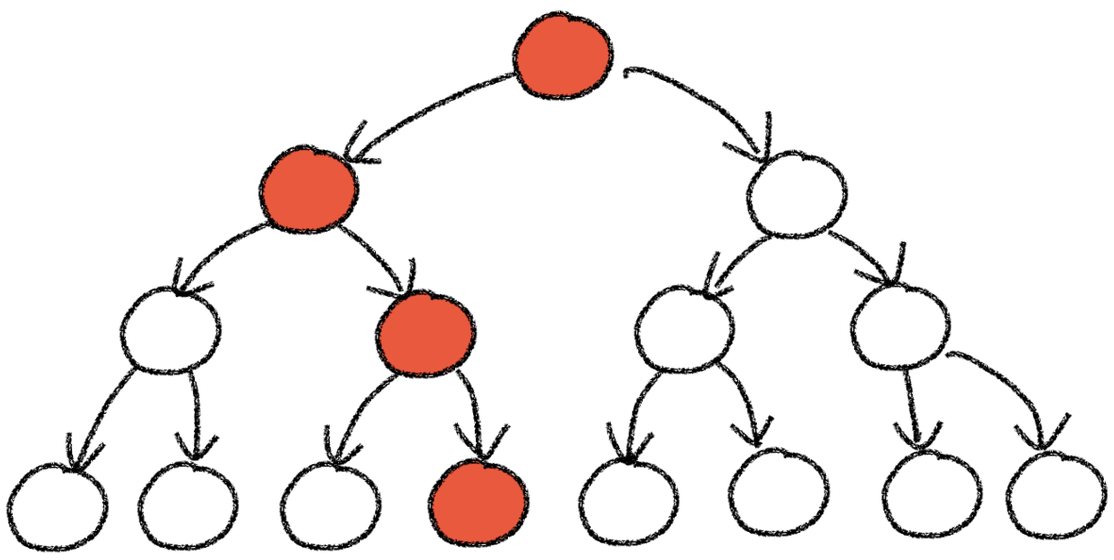
$\forall \Diamond \phi$

"ϕ is inevitable"



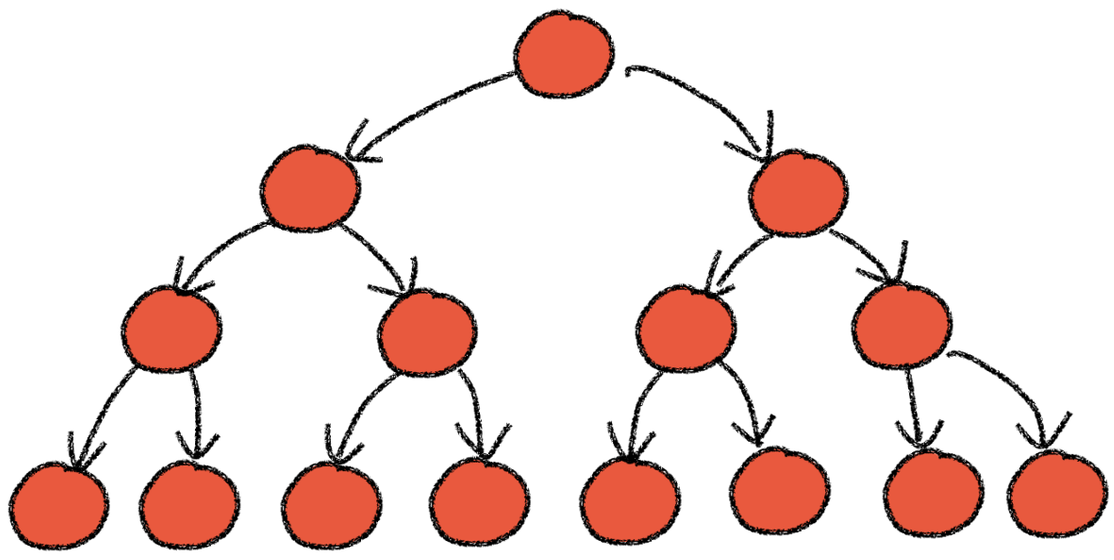
$\exists \Box \phi$

"potentially always ϕ"



$\forall \Box \phi$

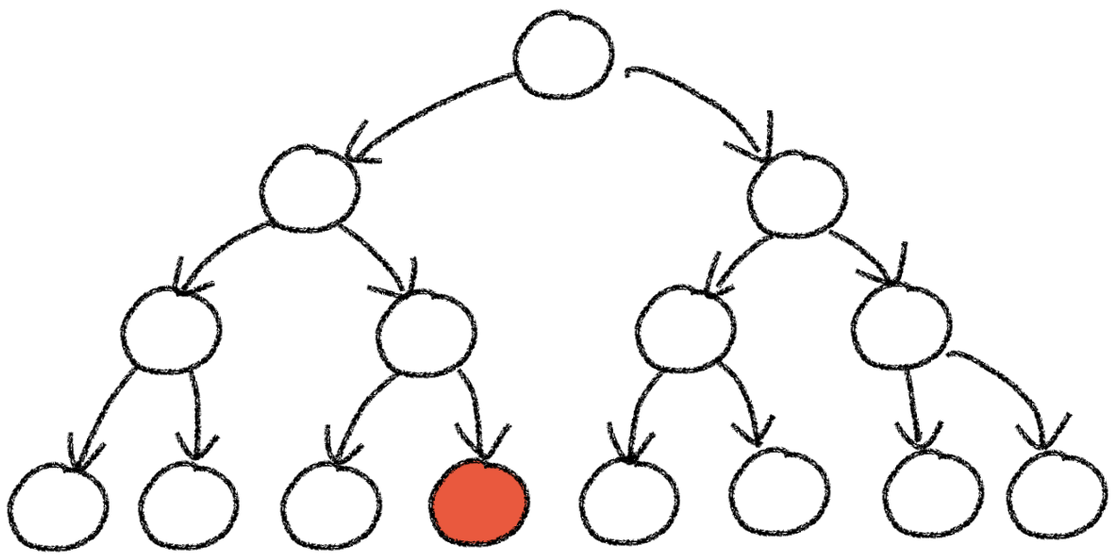
"Globally always ϕ"



Intuition of the “Until” operator

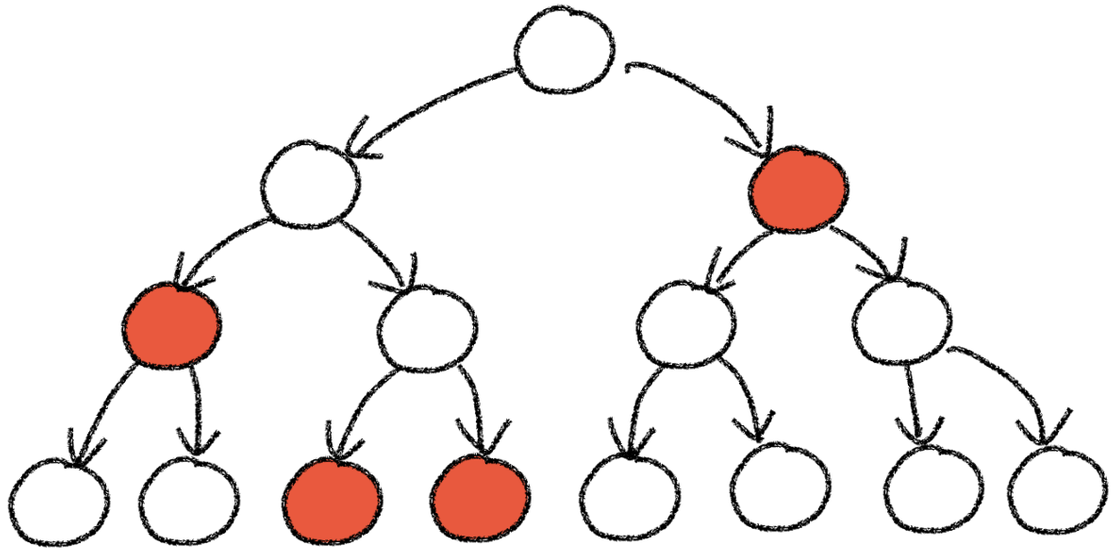
$\exists \Diamond \phi$

"ϕ holds potentially"



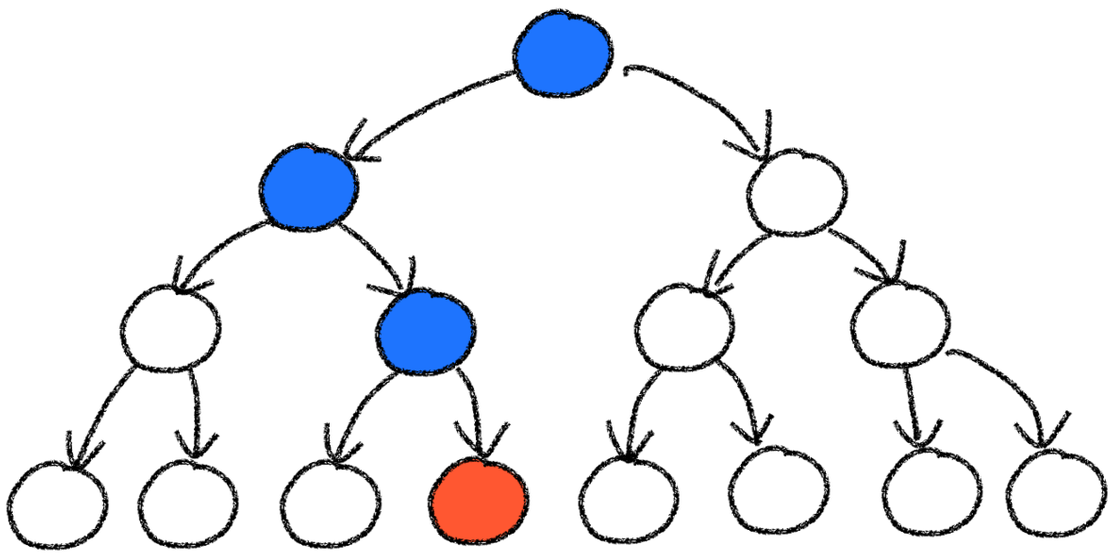
$\forall \Diamond \phi$

"ϕ is inevitable"



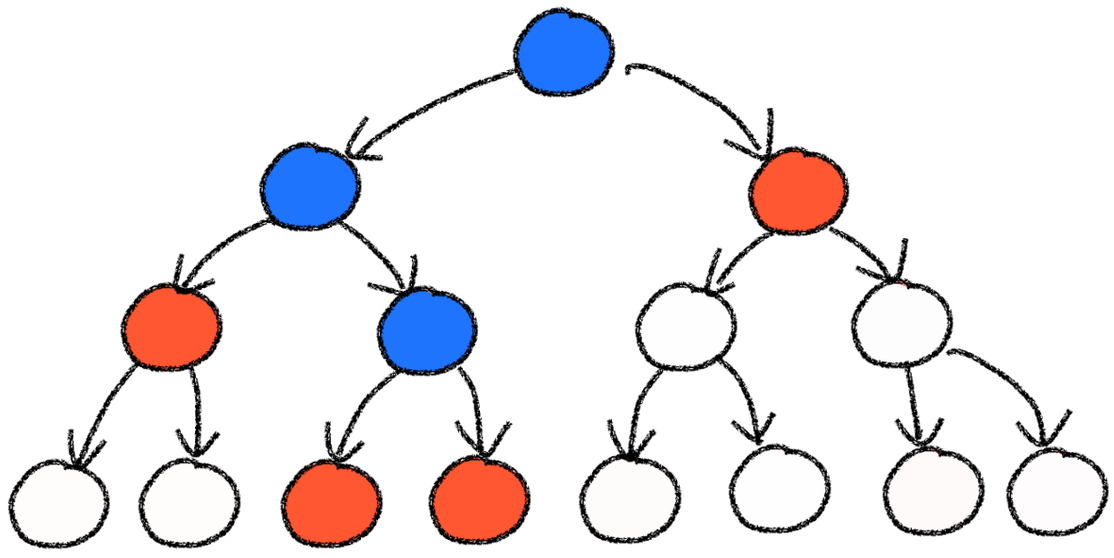
$\exists \ \phi_1 \cup \phi_2$

"there is a path s.t. $\phi_1 \cup \phi_2$ holds"

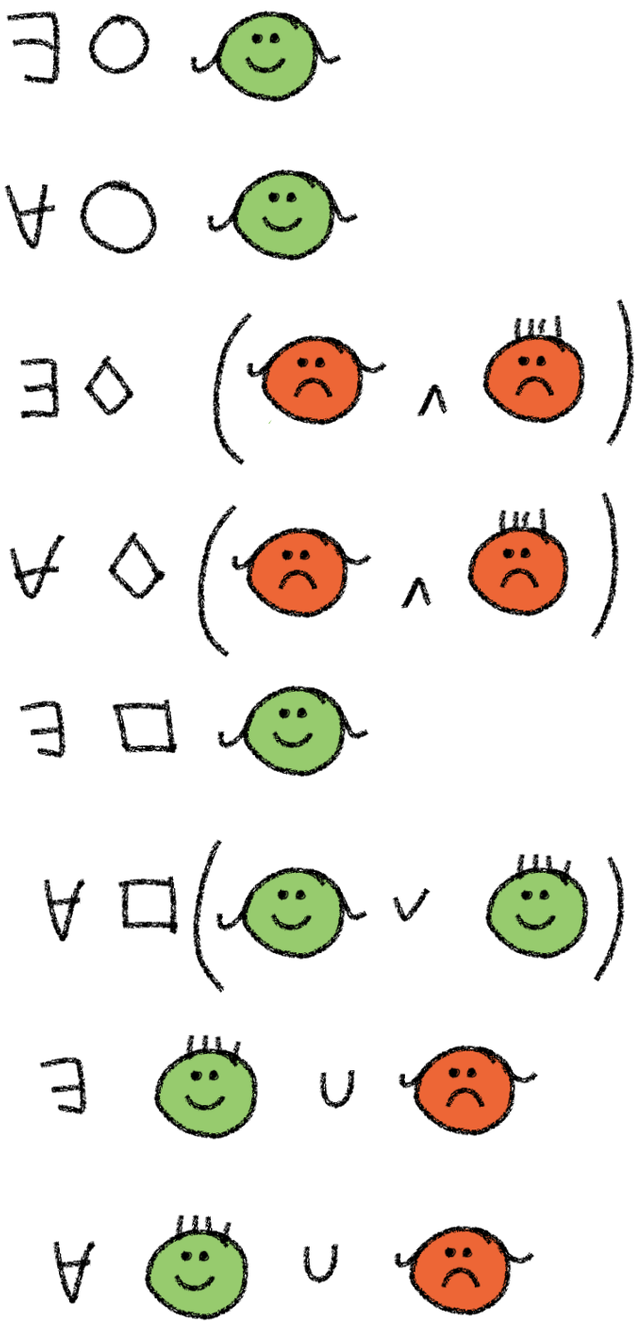


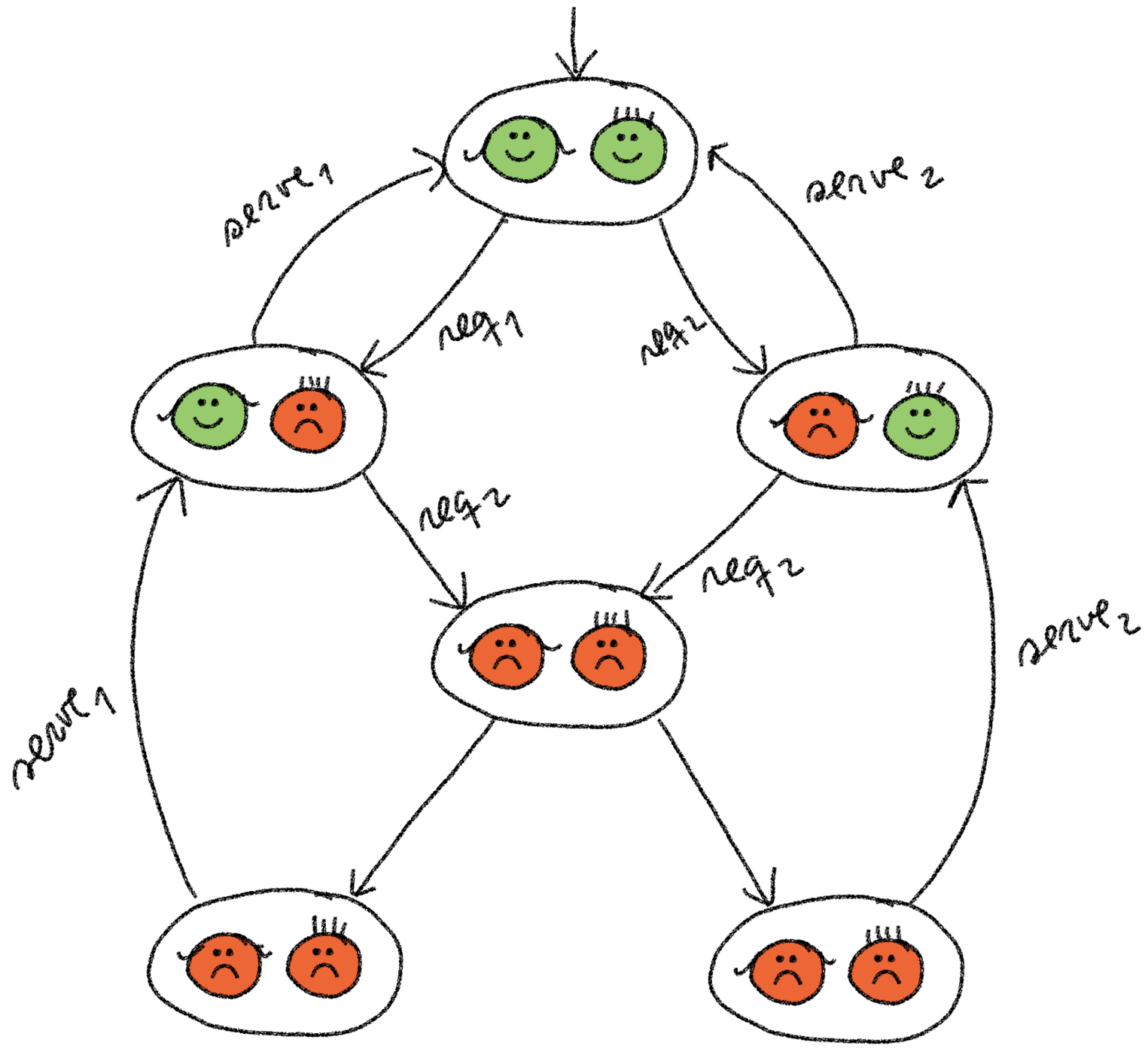
$\forall \ \phi_1 \cup \phi_2$

"in all paths $\phi_1 \cup \phi_2$ holds"



Some examples





Typical patterns of formulas

"inv is an invariant"

$$\forall \Box\, inv$$

"every request if followed by a response"

$$\forall \Box\, (request \rightarrow \exists \Diamond response)$$

"infinitely often p holds"

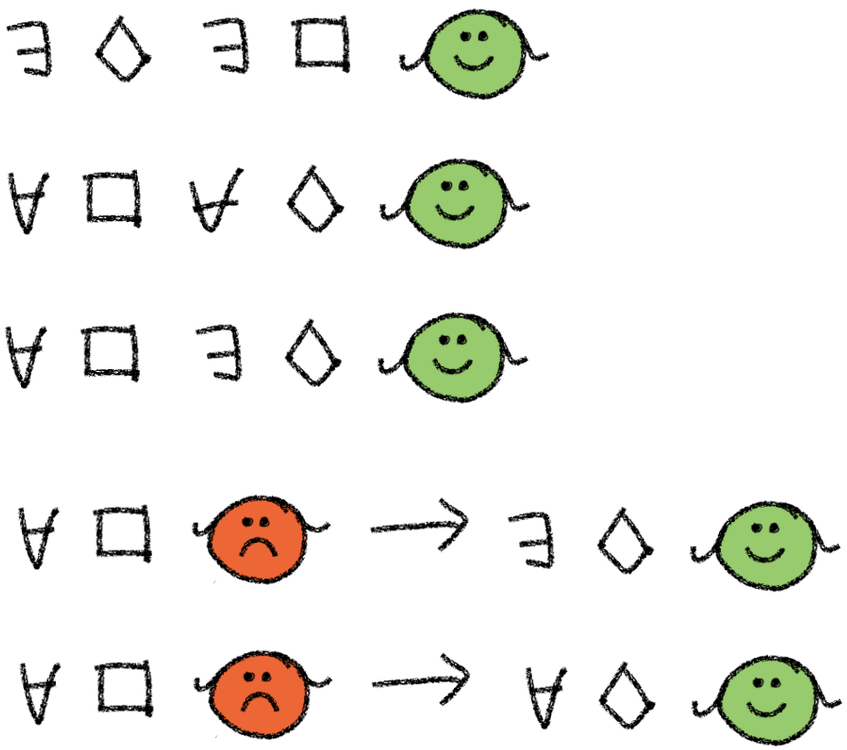
$$\forall \Box \forall \Diamond p$$

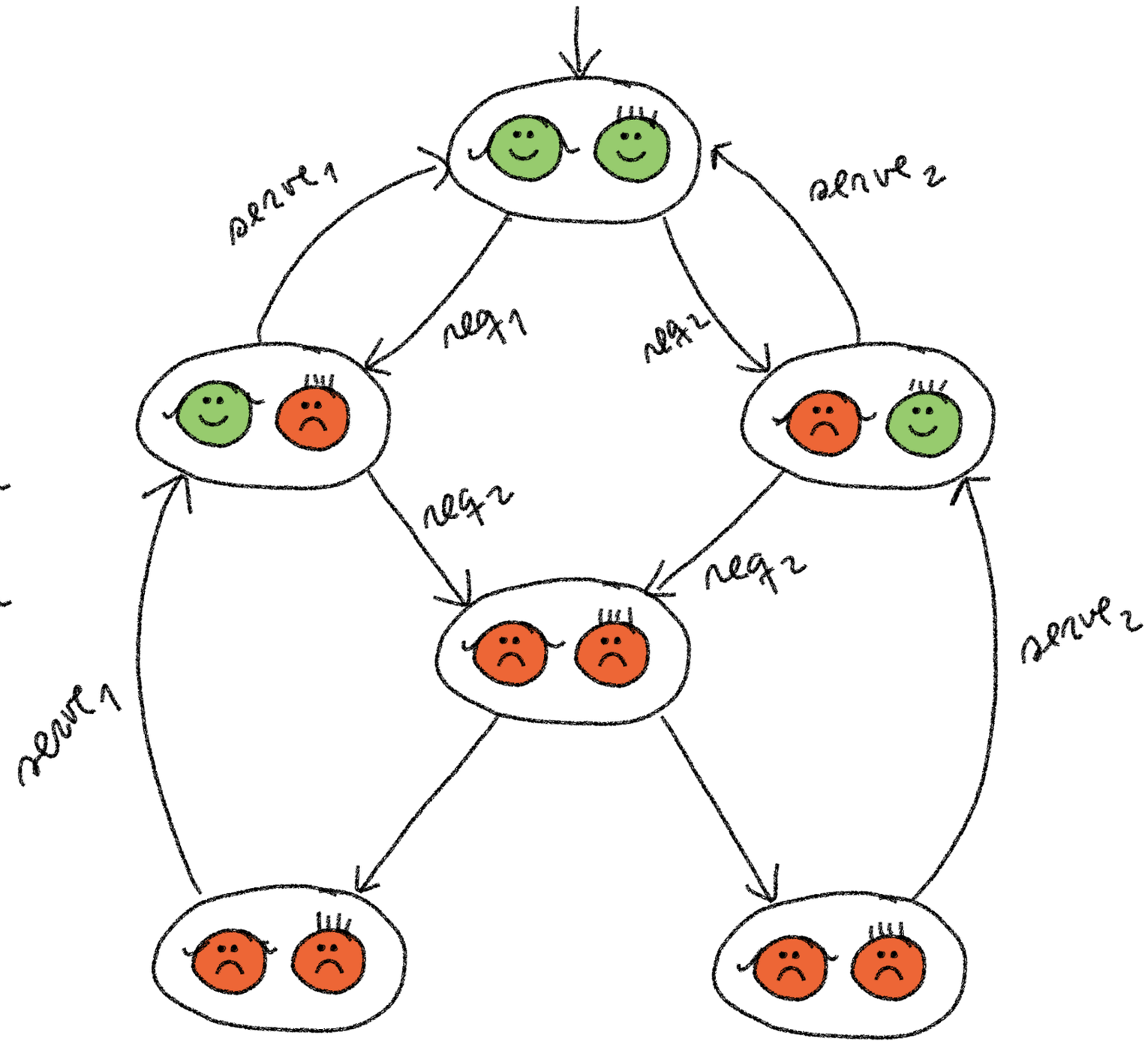
"q is persistent"

$$\forall \Diamond \exists \Box\, q$$

NOTE: A collection of property patterns is available at https://matthewbdwyer.github.io/psp/

Some examples





Lecture 02 - Computation Tree Logic (CTL)

- CTL formulas, grammar and intuition
- Formal semantics of CTL
- Formula equivalences and alternative grammars
- CTL in PRISM
- Beyond CTL: CTL* and LTL

Some notation: paths, prefixes, states

For a state *s* we define the set of all paths starting from *s*, as the set of all maximal executions starting from *s*.

$$Paths(s) = \{s_0, s_1, s_2, \ldots \mid s_0 = s_1 \text{ and } s_i \rightarrow s_{i+1}\}$$

For a path $\pi = s_0, s_1, s_2, \ldots$ we define the (i-1)th state by

$$\pi[i] = s_i$$

We can also define the prefix starting at the (i-1)th state

$$\pi[i\,..\,] = s_i, s_{i+1}, \ldots$$

CTL - formal semantics

We define the formal semantics of CTL as 2 relations.

A relation between states and state formulas

$s \models true$ (holds always)

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

$s \models \neg\phi$ iff $s \not\models \phi$

$s \models \phi_1 \wedge \phi_2$ iff $s \models \phi_1$ and $s \models \phi_2$

$s \models \exists\phi$ iff $\exists \pi \in Paths(s) \,.\, \pi \models \psi$

$s \models \forall\phi$ iff $\forall \pi \in Paths(s) \,.\, \pi \models \psi$

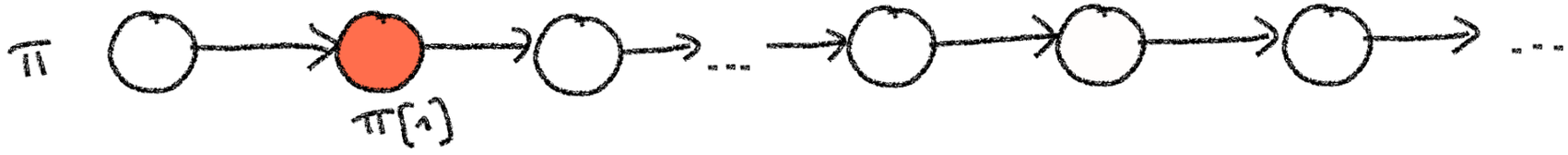
and a relation between paths and path formulas (next slide)

$\pi \models \psi$ iff …

Semantics of path formulas

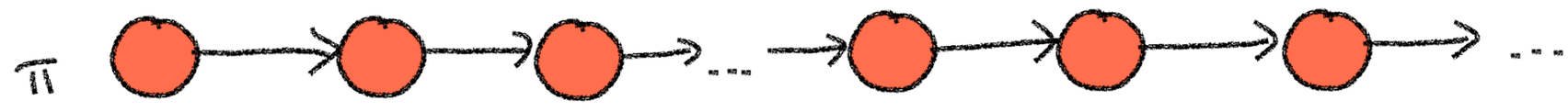
And here is the formal semantics of path formulas

$\pi \vDash \bigcirc \phi$ iff $\pi[1] \vDash \phi$

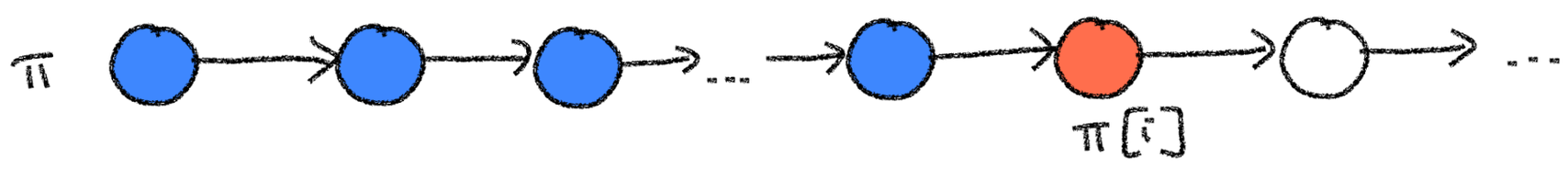


$\pi \vDash \Diamond \phi$ iff $\exists i \in \mathbb{N} \,.\, \pi[i] \vDash \phi$

$\pi \vDash \Box \phi$ iff $\forall i \in \mathbb{N} \,.\, \pi[i] \vDash \phi$



$\pi \vDash \phi_1 \mathsf{U} \phi_2$ iff $\exists i \in \mathbb{N} \,.\, \pi[i] \vDash \phi_2 \wedge \forall 0 \leq j < i \,.\, \pi[j] \vDash \phi_1$



Satisfaction sets

We define the satisfaction set of a CTL state formula as the set of states that satisfy the formula

$$sat(\phi) = \{s \mid s \vDash \phi\}$$

Semantics over a TS

We say that a transition system satisfies a formula if all its initial states satisfy the formula:

$$T \models \phi \text{ iff } \forall s \in I \,.\, s \models \phi$$

or, equivalently,

$$T \models \phi \text{ iff } I \subseteq sat(\phi)$$

Lecture 02 - Computation Tree Logic (CTL)

- CTL formulas, grammar and intuition
- Formal semantics of CTL
- Formula equivalences and alternative grammars
- CTL in PRISM
- Beyond CTL: CTL* and LTL

Equivalences

Two formulas are equivalent iff their validity (hold/does not hold) is the same for all transitions systems, i.e. for all transition systems *T* we have:

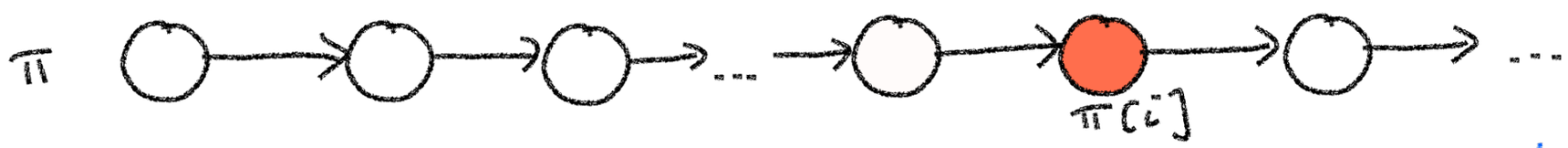
$$T \models \phi_1 \qquad \text{iff} \qquad T \models \phi_2$$

“Eventually” is a case of “Until”

A simple equivalence is allows use to define “eventually” with “until”:

$$\Diamond \phi \equiv \mathit{true}\ \mathsf{U}\ \phi$$

$\pi \vDash \Diamond \phi_2$ iff $\exists i \in \mathbb{N}\,.\,\pi[i] \vDash \phi_2$

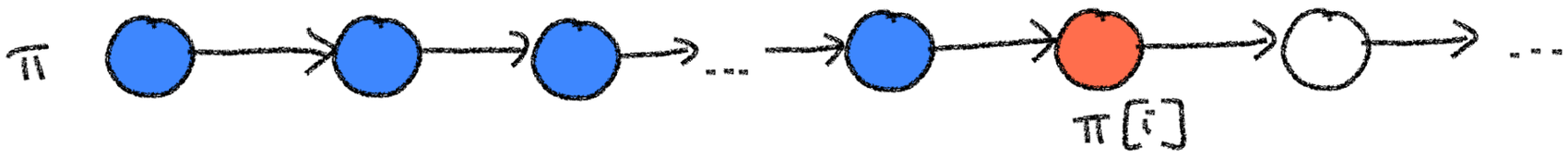


$\pi \vDash \phi_1 \mathsf{U} \phi_2$ iff $\exists i \in \mathbb{N}\,.\,\pi[i] \vDash \phi_2 \wedge \forall 0 \leq j < 1.\pi[j] \vDash \phi_1$

true!

true

true

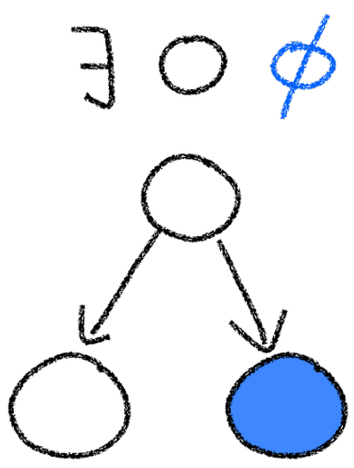


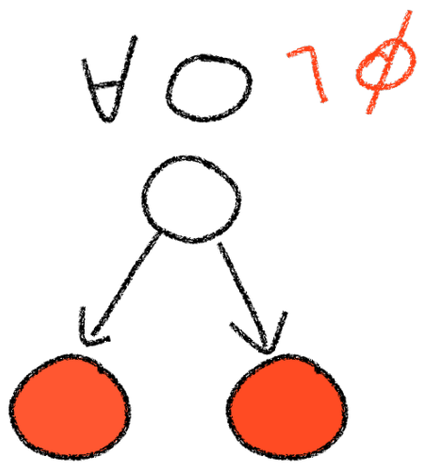
Dualities for “next”

Another example of simple equivalences is

$$\exists \bigcirc \phi \equiv \neg \forall \bigcirc \neg \phi \qquad \forall \bigcirc \phi \equiv \neg \exists \bigcirc \neg \phi$$

Easy to see graphically





Dualities for “eventually” and “always”

As for propositional logic we could get rid of some operators:

$$\exists \Box \phi \equiv \neg \forall \Diamond \neg \phi$$

$$\forall \Box \phi \equiv \neg \exists \Diamond \neg \phi$$

Note that we need the quantifiers. We can’t do this:

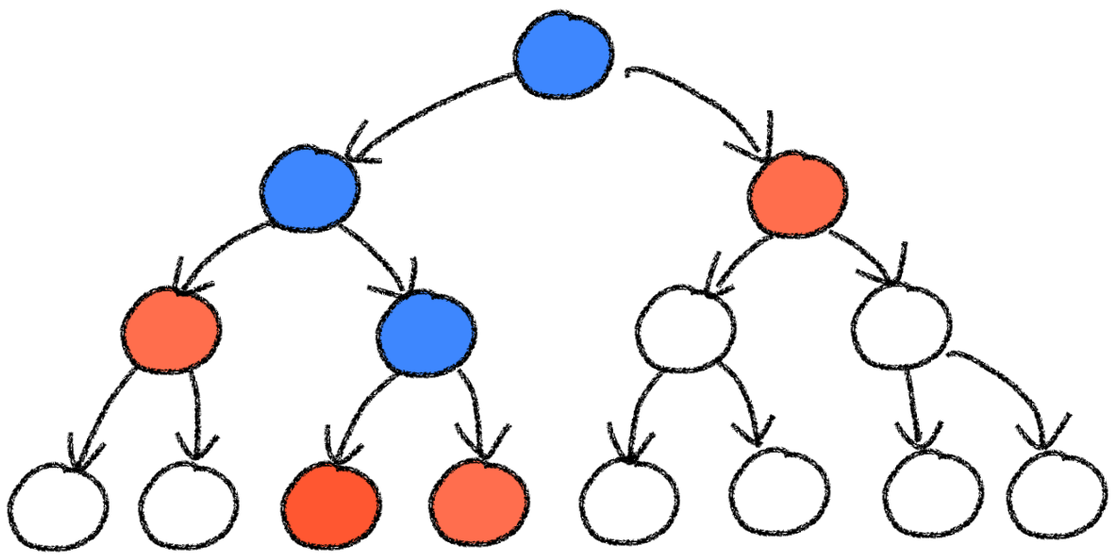
$$\Box \phi \equiv \neg \Diamond \neg \phi$$

Why?

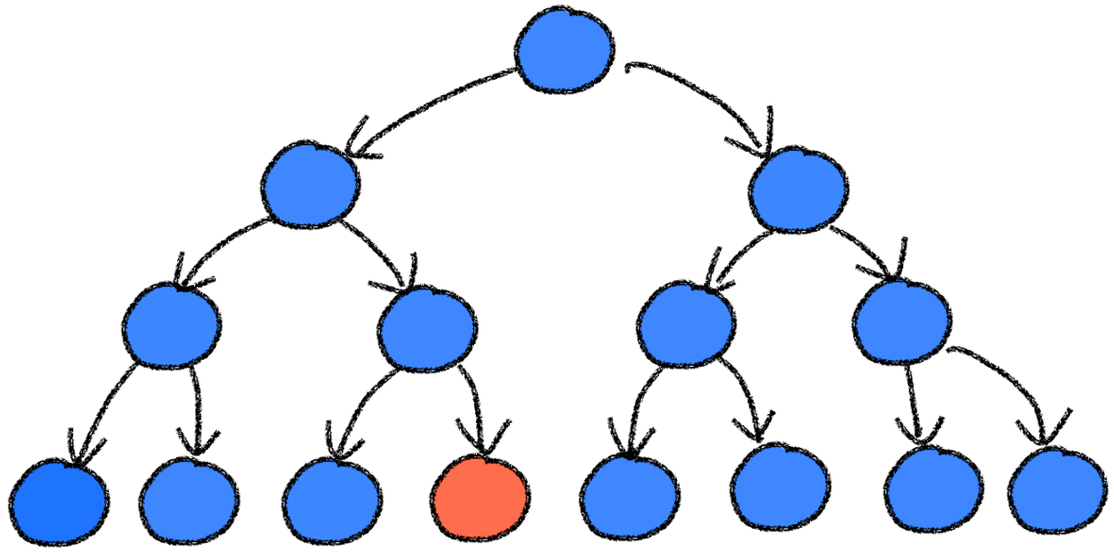
Intuition of dualities for “eventual” and “always”

$\forall \Diamond \neg \phi$

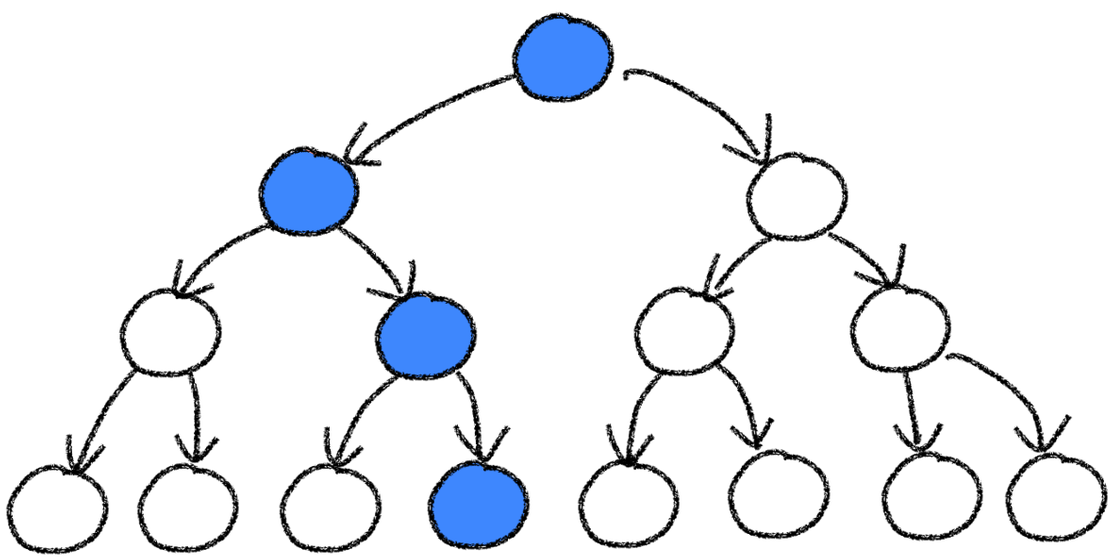
"ϕ holds potentially"



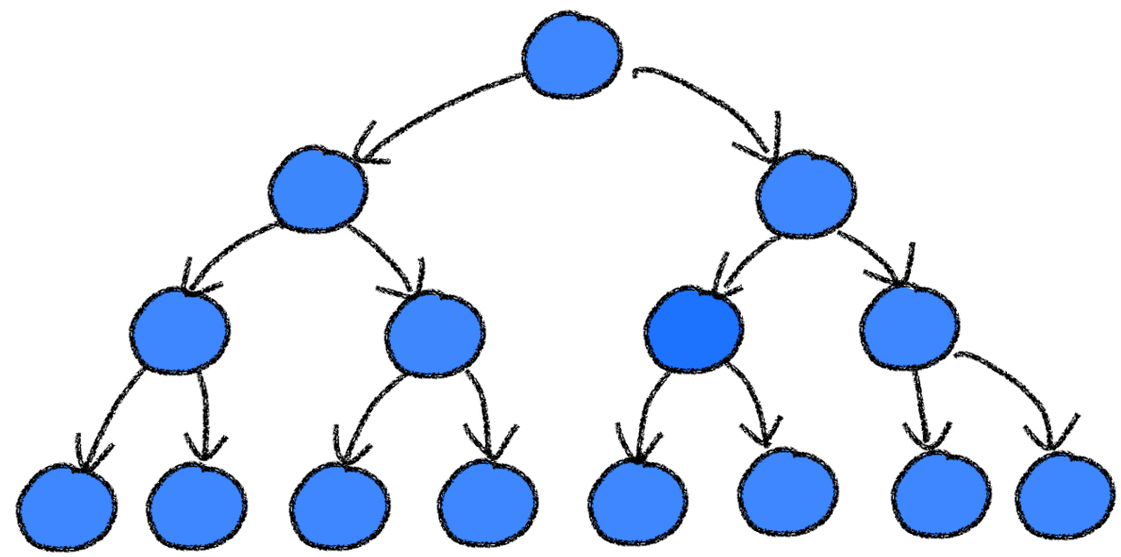
$\exists \Diamond \neg \phi$



$\exists \Box \phi$



$\forall \Box \phi$



CTL - minimal syntax

So, we can get rid of the “always” and “eventually” operators

$$\phi ::= true \mid p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \exists\psi \mid \forall\psi$$

$$\psi ::= \bigcirc\phi \mid \Diamond\phi \mid \Box\phi \mid \phi_1 \mathsf{U} \phi_2$$

Alternatively, we can get rid of the universal quantifier

$$\phi ::= true \mid p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \exists\psi \mid \forall\psi$$

$$\psi ::= \bigcirc\phi \mid \Diamond\phi \mid \Box\phi \mid \phi_1 \mathsf{U} \phi_2$$

The above grammar yields CTL formulas in so-called “existential normal form” (we shall consider it when we will see the model-checking algorithms)

More equivalences

Expansion laws

$$\exists \Box \phi \equiv \phi \wedge \exists \bigcirc \exists \Box \phi$$

$$\exists \Diamond \phi \equiv \phi \vee \exists \bigcirc \exists \Diamond \phi$$

$$\exists \phi_1 \mathsf{U} \phi_2 \equiv \phi_2 \vee (\phi_1 \wedge \exists \bigcirc \exists \phi_1 \mathsf{U} \phi_2)$$

basis for model checking algorithms (next lecture)

Distributive laws

$$\exists \Diamond (\phi_1 \vee \phi_2) \equiv (\exists \Diamond \phi_1) \vee (\exists \Diamond \phi_2)$$

$$\forall \Box (\phi_1 \wedge \phi_2) \equiv (\forall \Box \phi_1) \wedge (\forall \Box \phi_2)$$

More Equivalences?

The following equivalence do not hold

$$\exists \Diamond (\phi_1 \wedge \phi_2) \equiv (\exists \Diamond \phi_1) \wedge (\exists \Diamond \phi_2)$$

Why?

$$\exists \Diamond(\phi_1 \wedge \phi_2) \overset{?}{\equiv} (\exists \Diamond \phi_1) \wedge (\exists \Diamond \phi_2)$$

$\Rightarrow$

Let $\mathcal{T}$ be any TS s.t. $\mathcal{T} \models \exists \Diamond (\phi_1 \wedge \phi_2)$.

By definition (semantics) we know that for all initial states $s_0 \in I$ there is a path $\pi = s_0 s_1 \dots$ and an $i \in \mathbb{N}$ s.t.

$\pi[i] \models \phi_1 \wedge \phi_2$

$\Downarrow$ $\wedge$-elimination

$\pi[i] \models \phi_1$

$\Downarrow$ semantics of $\Diamond$

$\pi \models \Diamond \phi_1$

$\Downarrow$ semantics of $\exists$

$s_0 \models \exists \Diamond \phi_1$

$\Downarrow$

$\pi[i] \models \phi_2$

$\Downarrow$

$\pi \models \Diamond \phi_2$

$\Downarrow$

$s_0 \models \exists \Diamond \phi_2$

$\Downarrow$ sem of $\wedge$

$s_0 \models \exists \Diamond \phi_1 \wedge \exists \Diamond \phi_2$

$\Downarrow$

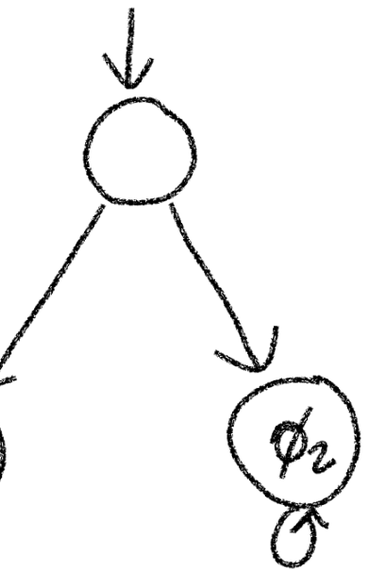
$\mathcal{T} \models \exists \Diamond \phi_1 \wedge \exists \Diamond \phi_2$

$$\exists\Diamond(\phi_1 \wedge \phi_2) \not\equiv (\exists\Diamond\phi_1) \wedge (\exists\Diamond\phi_2)$$

$$\not\Leftarrow$$

We need to find a TS $\mathcal{T}$ s.t.

$\mathcal{T} \models \exists\Diamond\phi_1 \wedge \exists\Diamond\phi_2$ $\Rightarrow$



<u>but</u>

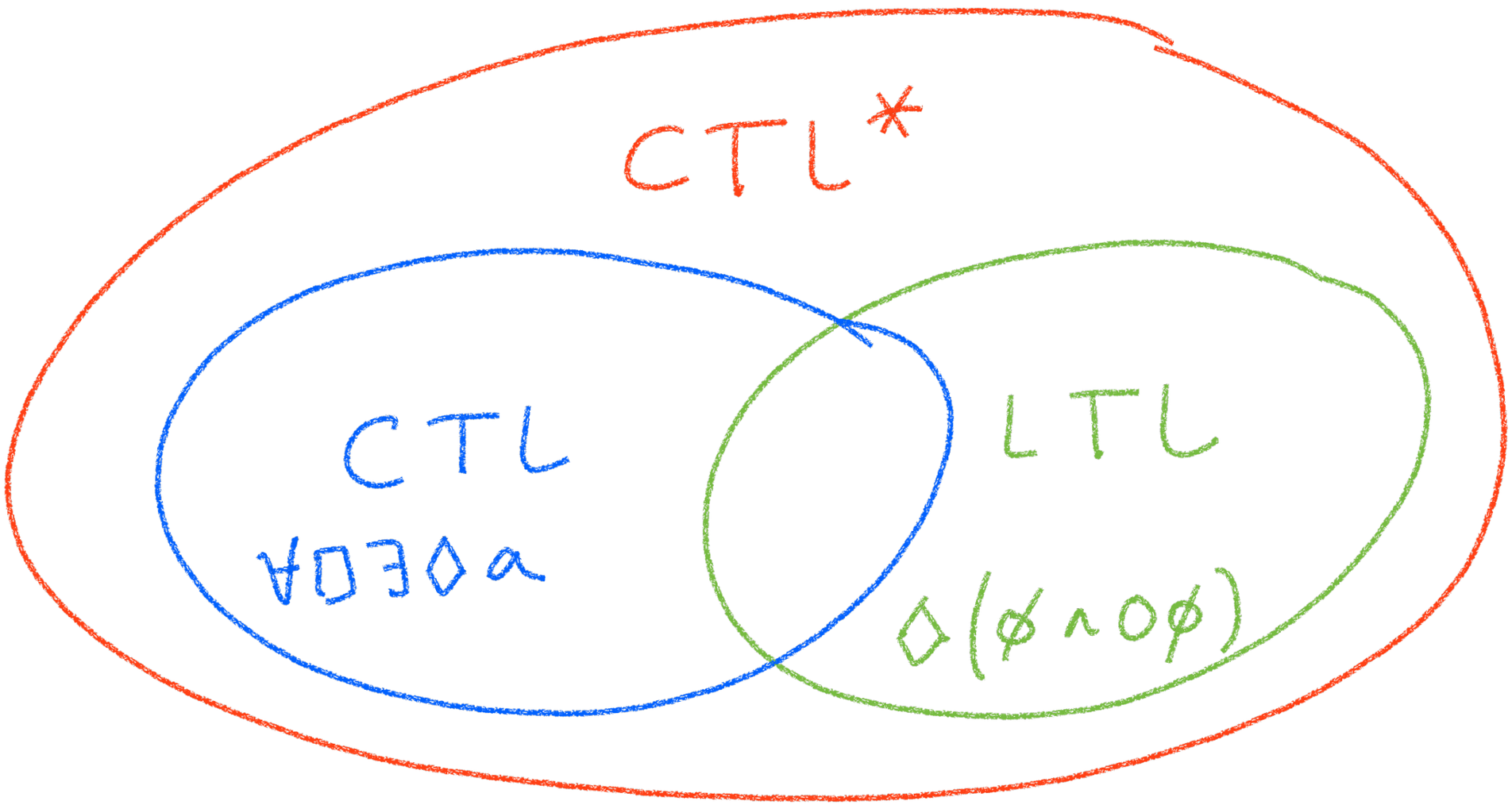
$\mathcal{T} \not\models \exists\Diamond(\phi_1 \wedge \phi_2)$ $\not\Rightarrow$

Lecture 02 - Computation Tree Logic (CTL)

- CTL formulas, grammar and intuition
- Formal semantics of CTL
- Formula equivalences and alternative grammars
- Beyond CTL: CTL* and LTL

The CTL* family

CTL is not the only logic for transition systems…



CTL* grammar

The grammar of CTL* has the same state formulas

$$\phi ::= true \mid p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \exists\psi \mid \forall\psi$$

But (subtly) extended path formulas

$$\psi ::= \bigcirc\psi \mid \Diamond\psi \mid \Box\psi \mid \psi_1 \mathsf{U} \psi_2 \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \phi$$

Main differences:

(1) no need to interleave quantifiers with temporal operators

(2) All state formulas are also path formulas

Linear-time Temporal Logic (LTL)

LTL formulas are CTL* of the form

$$\forall\psi$$

Usually presented with this grammar (where the initial universal quantifier is implicit)

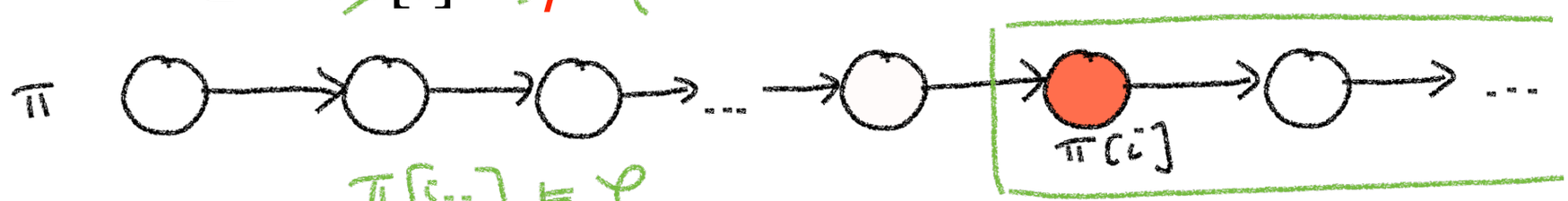
$$\psi ::= true \mid p \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \bigcirc\psi \mid \Diamond\psi \mid \Box\psi \mid \psi_1 \mathsf{U} \psi_2$$

Semantics of path formulas

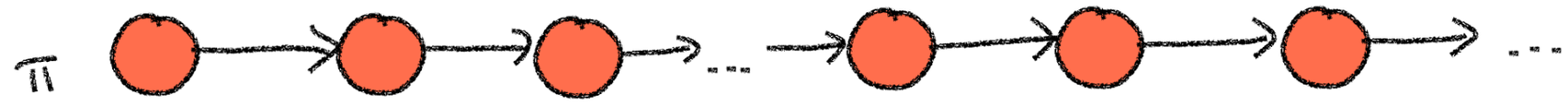
And here is the formal semantics of path formulas

$\pi \vDash \bigcirc \phi$ iff $\pi[1] \vDash \phi$

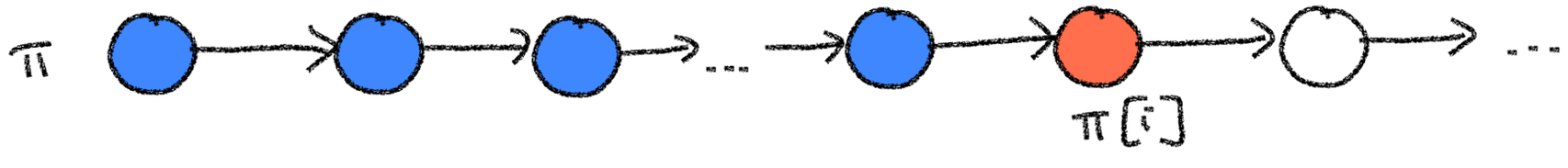
$\pi \vDash \Diamond \phi$ iff $\exists i \in \mathbb{N} . \pi[i] \vDash \phi$



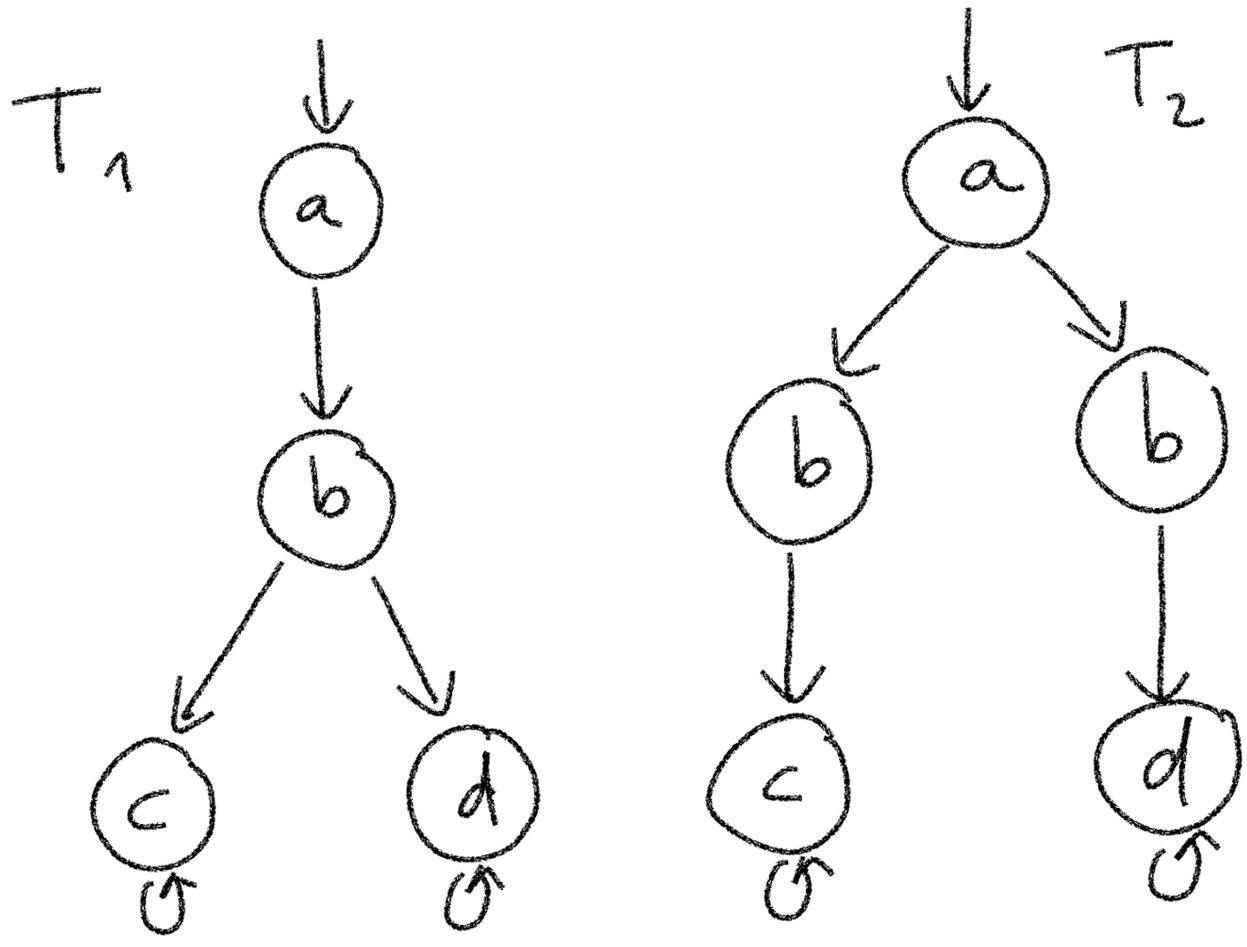
$\pi \vDash \Box \phi$ iff $\forall i \in \mathbb{N} . \pi[i] \vDash \phi$



$\pi \vDash \phi_1 \mathsf{U} \phi_2$ iff $\exists i \in \mathbb{N} . \pi[i] \vDash \phi_2 \wedge \forall 0 \leq j < 1 . \pi[j] \vDash \phi_1$



CTL can distinguish more than LTL



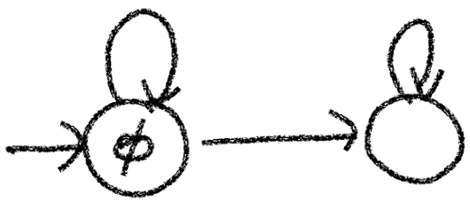
T_1 and T_2 have the same traces $\Rightarrow$ satisfy the same LTL formulas

but ... $T_1 \not\models \forall \Box (b \rightarrow \exists \bigcirc c)$

$T_2 \models$

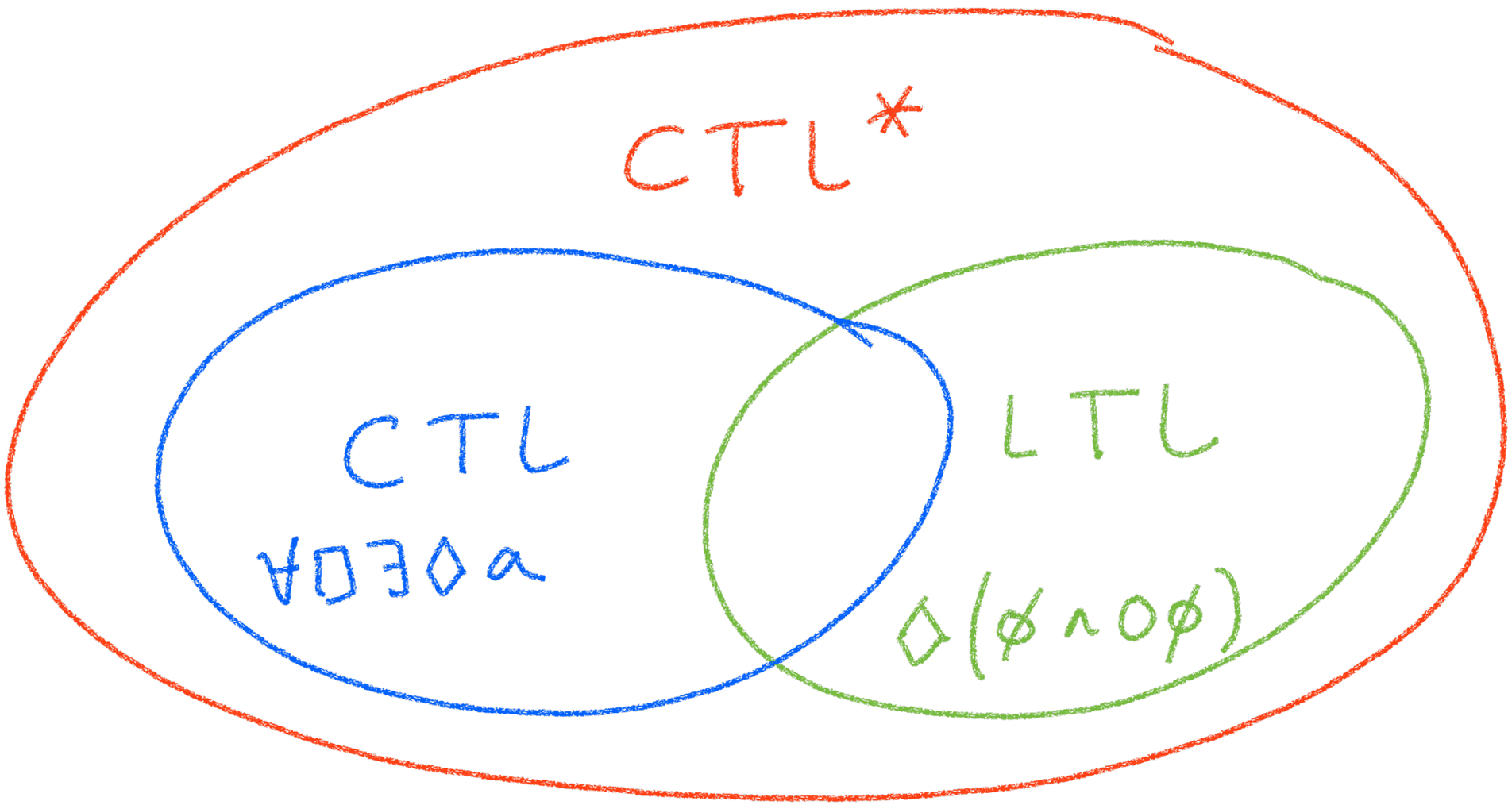
↑ CTL formula

Still there are LTL formulas not expressible in CTL!

$\models$?	LTL $\forall\Box(\phi \wedge \bigcirc\phi)$	CTL $\forall\Box(\phi \wedge \exists\bigcirc\phi)$	CTL $\forall\Box(\phi \wedge \forall\bigcirc\phi)$
	×	✓	
	✓		×

The CTL* family

CTL* is more expressive than both



Example: just take some combination of the above two formulas

Lecture 02 - Computation Tree Logic (CTL)

- CTL formulas, grammar and intuition
- Formal semantics of CTL
- Formula equivalences and alternative grammars
- Beyond CTL: CTL* and LTL

Key points so far

Computation Tree Logic (CTL) as a logic to reason about transition systems, in particular about their computation trees.

Grammars for CTL: standard, minimalistic and existential normal form.

Formal semantics of CTL: satisfaction relation for states and paths.

Derived operators and equivalences between CTL formulas.

CTL* as a generalisation of CTL (and LTL).