Formal Methods in software development

a.y.2017/2018

Prof. Anna Labella

LTL: what is expressible

Safety properties $G \neg \varphi$

Liveness properties $GF\psi$ or $G(\varphi \rightarrow F\psi)$

Semantics: transition systems

Abstract models: states and transitions

LTS: Automata without terminal states

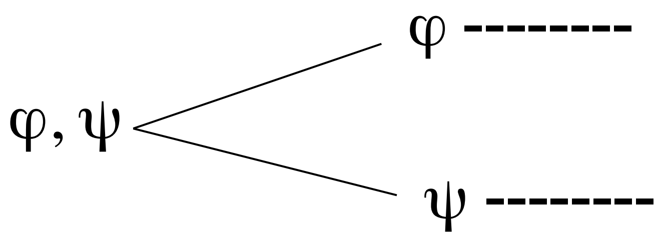
Kripke structures

$$\rightarrow \subseteq S \times S$$

LTL

Characterising linear time

$$G\,((\varphi \vee G\psi) \wedge (G\varphi \vee \psi)) \cong (G\varphi \vee G\psi)$$



Lefthand holds, righthand does not

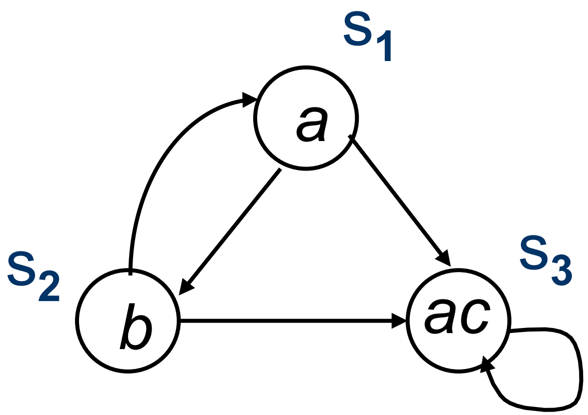
LTL: what is expressible

- It is impossible to get to a state where **started** holds, but **ready** does not hold: $G\neg(\mathsf{started} \wedge \neg\mathsf{ready})$
- For any state, if a **request** (of some resource) occurs, then it will eventually be acknowledged:
  $\mathrm{G}\,(\mathsf{requested} \rightarrow \mathrm{F}\,\mathsf{acknowledged})$.
- A certain process is **enabled** infinitely often on every computation path:
  $\mathrm{G}\,\mathrm{F}\,\mathsf{enabled}$.
- Whatever happens, a certain process will eventually be permanently **deadlock**ed:
  $\mathrm{F}\,\mathrm{G}\,\mathsf{deadlock}$.
- If the process is enabled infinitely often, then it runs infinitely often.
  $\mathrm{G}\,\mathrm{F}\,\mathsf{enabled} \rightarrow \mathrm{G}\,\mathrm{F}\,\mathsf{running}$.
- An upwards travelling lift at the second floor does not change its direction when it has passengers wishing to go to the fifth floor:
  $\mathrm{G}\,(\mathsf{floor2} \wedge \mathsf{directionup} \wedge \mathsf{ButtonPressed5} \rightarrow (\mathsf{directionup}\ \mathrm{U}\ \mathsf{floor5}))$
  Here, our atomic descriptions are boolean expressions built from system variables, e.g., **floor2**.

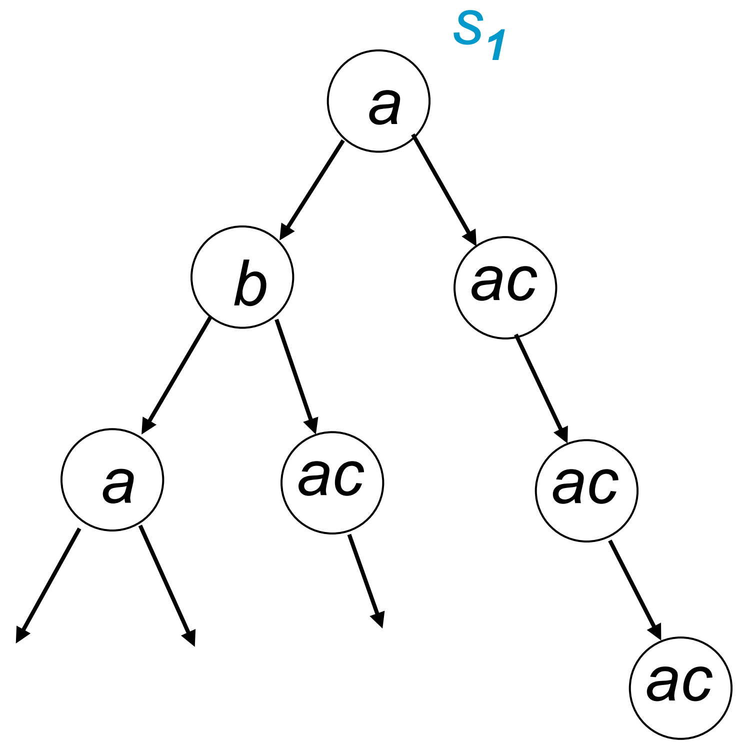
LTL: what is not expressible

- From any state *it is possible* to get to a restart state (i.e., there is a path from all states to a state satisfying restart).
- The lift *can* remain idle on the third floor with its doors closed (i.e., from the state in which it is on the third floor, there is a path along which it stays there).

Computation Trees: labelled TS



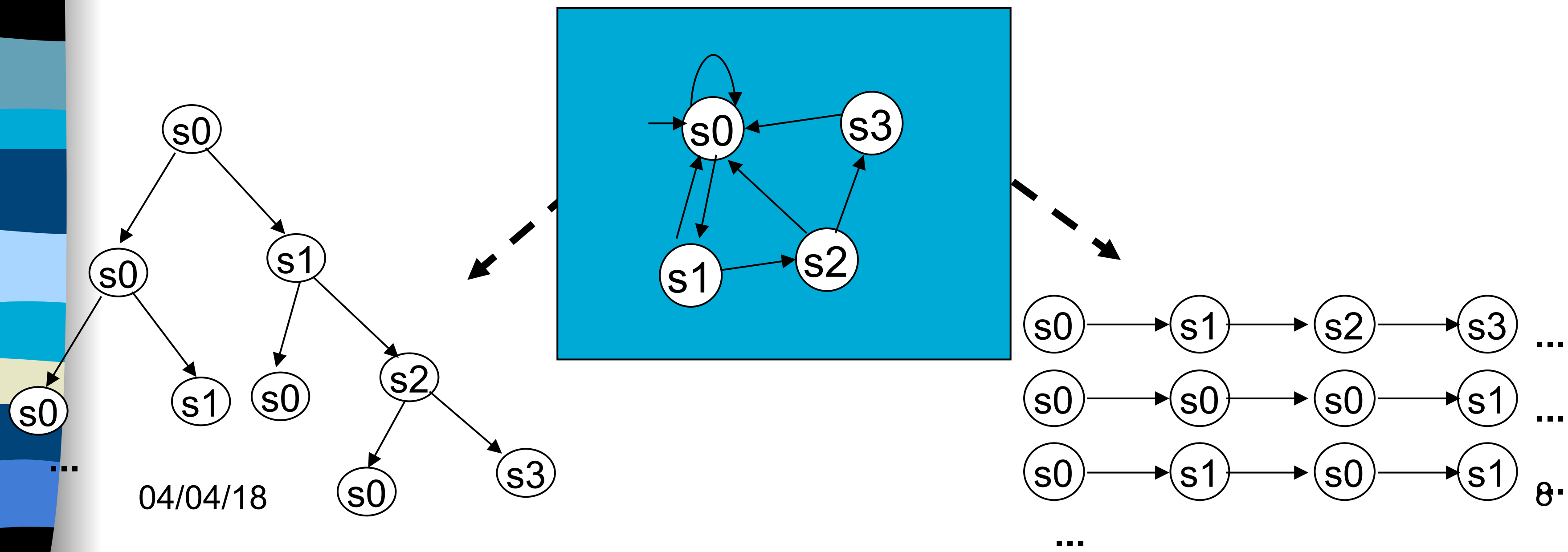
State transition structure
(*Kripke Model*)



Infinite computation tree
for initial state s_1

Linear time and Branching time

- **Linear**: only one possible future in a moment
  - Look at individual computations
- **Branching**: may split to different courses depending on possible futures
  - Look at the tree of computations



Operators and Quantifiers

- State operators
  - G ϕ : ϕ holds globally
  - F ϕ : ϕ holds eventually
  - X ϕ : ϕ holds at the next state
  - ϕ U ψ : ϕ holds until ψ holds
  - ϕ W ψ : ϕ holds until ψ *possibly* holds

- Path quantifiers
  - E: along at least one path (there *exists* …)
  - A: along all paths (for *all* …)

CTL characterisation

- Temporal operators must be immediately preceded by a path quantifier

Typical CTL Formulas

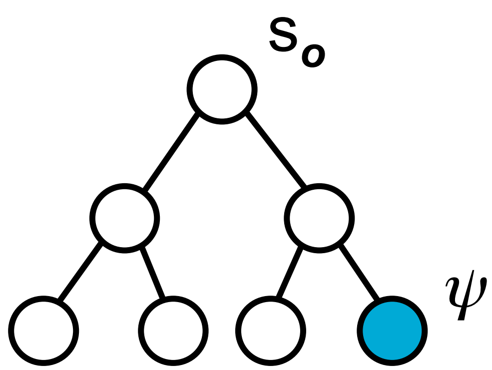
- E F (*start* ∧ ¬ *ready*)
  - eventually a state is reached where *start* holds and *ready* does not hold
- A G (*req* → A F *ack*)
  - any time *request* occurs, it will be eventually *ack*nowledged
- A G (E F *restart*)
  - from any state it is possible to get to the *restart* state

CTL semantics

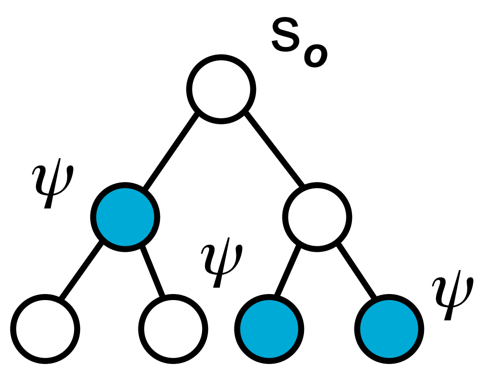
- E X (ϕ)
  - true in state *s* if ϕ is true in some successor of *s* (there *exists* a next state of s for which ϕ holds)
- A X (ϕ)
  - true in state *s* if ϕ is true for all successors of *s (*for *all* next states of *s* ϕ is true)
- E G (ϕ)
  - true in *s* if ϕ holds in *every* state along *some* path emanating from *s (there exists a path ….)*
- A G (ϕ)
  - true in *s* if ϕ holds in every state along *all* paths emanating from *s* (*for all* paths ….*globally*)

- E F (ψ)
  - there *exists* a path which *eventually* contains a state in which ψ is true
- A F (ψ)
  - for *all* paths, eventually there is state in which ψ holds
- E F (ϕ U ψ)
  - there *exists* a path where (ϕ U ψ) is true
- A F (ϕ U ψ)
  - *for all* paths (ϕ U ψ) is true
- E F, A F are special cases of E [ϕ Uψ], A [ϕ Uψ]
  - E F (ψ) = E [*true* U ψ], A F (ψ) = A [*true* U ψ]

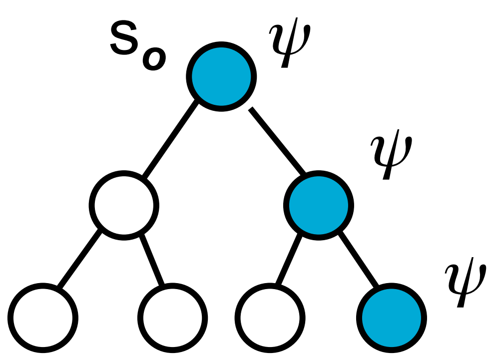
CTL Operators - examples



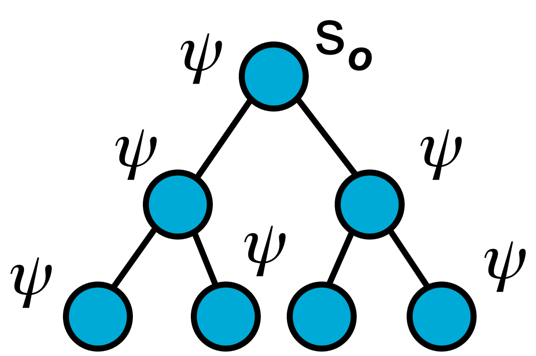
$s_o \models E\,F\,\psi$



$s_o \models A\,F\,\psi$



$s_o \models E\,G\,\psi$



$s_o \models A\,G\,\psi$

Minimal set of CTL Formulas

- Full set of operators
  - Boolean: $\neg, \wedge, \vee, \oplus, \rightarrow$
  - temporal: E, A, X, F, G, U, W
- Minimal set sufficient to express any CTL formula
  - Boolean: $\neg, \vee$
  - temporal: E, X, U
- Examples:

$\phi \wedge \psi = \neg(\neg\phi \vee \neg\psi)$, F ϕ = *true* U ϕ , A (ϕ) = $\neg$E($\neg\phi$)

CTL* – Computation Tree Logic

- Path quantifiers - describe branching structure of the tree
  - A (for *all* computation paths)
  - E (for *some* computation path = there *exists* a path)
- Temporal operators - describe properties of a path through the tree
  - X (next time, next state)
  - F (eventually, finally)
  - G (always, globally)
  - U (until)
  - W (weak until)

CTL* Formulas

- Temporal logic formulas are evaluated w.r.to a state in the model

- State formulas
  – apply to a specific state

- Path formulas
  – apply to all states along a specific path

CTL* Syntax

- An atomic proposition *p* is a state formula
- A state formula is also a path formula
- If ϕ, ψ are state formulae, so are $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$,
- If α is a path formula, E α is a state formula
- If α, β are path formulae, so are $\neg\alpha$, $\alpha \wedge \beta$, $\alpha \vee \beta$
- If α, β are path formulae, so are X α, αUβ

Summing up (CTL*)

- *state formulas*, which are evaluated in states:

$$\phi ::= \top \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid \mathrm{A}[\alpha] \mid \mathrm{E}[\alpha]$$

  where p is any atomic formula and α any path formula; and
- *path formulas*, which are evaluated along paths:

$$\alpha ::= \phi \mid (\neg\alpha) \mid (\alpha \wedge \alpha) \mid (\alpha\ \mathrm{U}\ \alpha) \mid (\mathrm{G}\,\alpha) \mid (\mathrm{F}\,\alpha) \mid (\mathrm{X}\,\alpha)$$

  where ϕ is any state formula.

CTL* Semantics

- If formula ϕ holds at state s (path π), we write:
  s $\models \phi$ ($\pi \models \alpha$)
- s $\models p$, p is an atomic formula, iff $p \in L(s)$ [label of s]
- s $\models \neg \phi$, iff s $\not\models \phi$
- s $\models \phi \wedge \psi$, iff s $\models \phi$ and s $\models \psi$
- s $\models E\ \phi$, iff $\exists\ \pi$ from state s, s.t. $\pi \models \phi$
- $\pi \models \neg \alpha$, iff $\pi \not\models \alpha$
- $\pi \models \alpha \wedge \beta$, iff $\pi \models \alpha$ and $\pi \models \beta$
- $\pi \models X\ \alpha$, iff $\pi^1 \models \alpha$ (α reachable in next state)
- $\pi \models \alpha\ U\ \beta$, iff $\pi \models \alpha$ until $\pi \models \beta$

CTL – Computational Tree Logic

- CTL* - a powerful branching-time temporal logic
- CTL – a branching-time fragment of CTL*
- In CTL every *temporal operator* (G,F,X,U,W) must be immediately preceded by a *path quantifier* (A,E)
- We need both state formulae and path formulae to recursively define the logic

More expressivity in CTL than in LTL?

Quantifying on paths

In LTL formulas are always quantified through A

LTL: blocks of operators must be thought as preceded by A always

- Linear time operators.
- The following are a complete set

$\neg\phi$, $\phi \vee \psi$, $X\phi$, $\phi\, U\, \psi$

Others can be derived

- $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$
- $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$
- $F\ \phi \equiv (\text{true}\ U\ \phi)$
- $G\ \phi \equiv (\phi\ U\ \text{false})$

CTL: what is expressible? 1

- It is possible to get to a state where **started** holds, but **ready** doesn't: EF (started $\wedge$ $\neg$ready). To express impossibility, we simply negate the formula.
- For any state, if a **request** (of some resource) occurs, then it will eventually be acknowledged:
  AG (requested $\rightarrow$ AF acknowledged).
- The property that if the process is enabled infinitely often, then it runs infinitely often, is not expressible in CTL. In particular, it is not expressed by AG AF enabled $\rightarrow$ AG AF running, or indeed any other insertion of A or E into the corresponding LTL formula. The CTL formula just given expresses that if every path has infinitely often enabled, then every path is infinitely often taken; this is much weaker than asserting that every path which has infinitely often enabled is infinitely often taken.
- A certain process is **enabled** infinitely often on every computation path:
  AG (AF enabled).
- Whatever happens, a certain process will eventually be permanently **deadlocked**:
  AF (AG deadlock).

CTL: what is expressible? 2

- From any state it is possible to get to a restart state:
  AG (EF restart).
- An upwards travelling lift at the second floor does not change its direction when it has passengers wishing to go to the fifth floor:
  AG (floor2 $\wedge$ directionup $\wedge$ ButtonPressed5 $\rightarrow$ A[directionup U floor5])
  Here, our atomic descriptions are boolean expressions built from system variables, e.g., floor2.
- The lift can remain idle on the third floor with its doors closed:
  AG (floor3 $\wedge$ idle $\wedge$ doorclosed $\rightarrow$ EG (floor3 $\wedge$ idle $\wedge$ doorclosed)).
- A process can always request to enter its critical section. Recall that this was not expressible in LTL. Using the propositions of Figure 3.8, this may be written AG $(n_1 \rightarrow \mathrm{EX}\, t_1)$ in CTL.
- Processes need not enter their critical section in strict sequence. This was also not expressible in LTL, though we expressed its negation. CTL allows us to express it directly: EF $(c_1 \wedge \mathrm{E}[c_1 \mathrm{U} (\neg c_1 \wedge \mathrm{E}[\neg c_2 \mathrm{U}\, c_1])])$.

Expressivity of LTL and CTL

- Safety $G \neg(c_1 \wedge c_2)$
- Liveness $G (t_i \rightarrow F c_i)$

- Safety $AG \neg(c_1 \wedge c_2)$
- Liveness $AG (t_i \rightarrow AF c_i)$

LTL and CTL

- **LTL** (Linear Temporal Logic) - Reasoning about infinite sequence of states $\pi: s_0, s_1, s_2, \ldots$
- **CTL** (Computation Tree Logic) – Reasoning on a computation tree.
  - Temporal operators are immediately preceded by a path quantifier (e.g. A F p)
- **CTL vs. LTL – different expressive power**
  - EFp is not expressible in LTL
  - FGp is not expressible in CTL

Comparing logics

PLTL	state-formulas	$\Phi ::= \mathsf{A}\,\varphi$
	path-formulas	$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi$
CTL	state-formulas	$\Phi ::= p \mid \neg\Phi \mid \Phi \vee \Phi \mid \mathsf{E}\,\varphi \mid \mathsf{A}\,\varphi$
	path-formulas	$\varphi ::= \mathsf{X}\,\Phi \mid \Phi\,\mathsf{U}\,\Phi$
CTL*	state-formulas	$\Phi ::= p \mid \neg\Phi \mid \Phi \vee \Phi \mid \mathsf{E}\,\varphi \mid \mathsf{A}\,\varphi$
	path-formulas	$\varphi ::= \Phi \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi$

Comparing logics

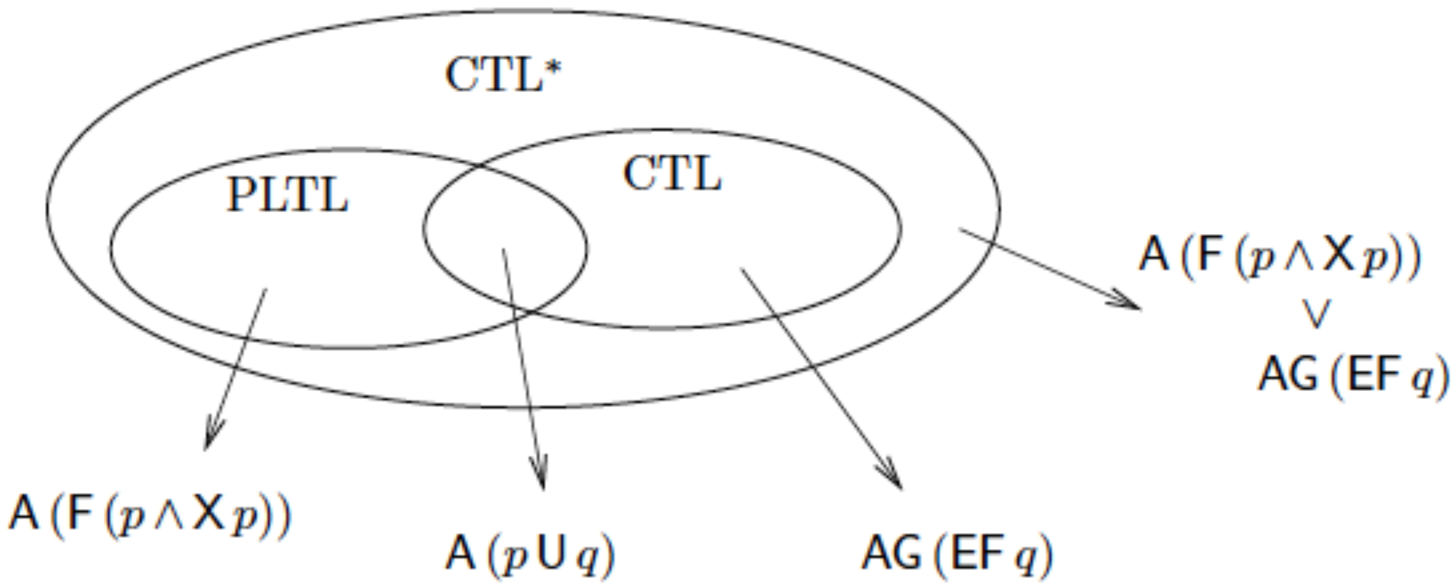
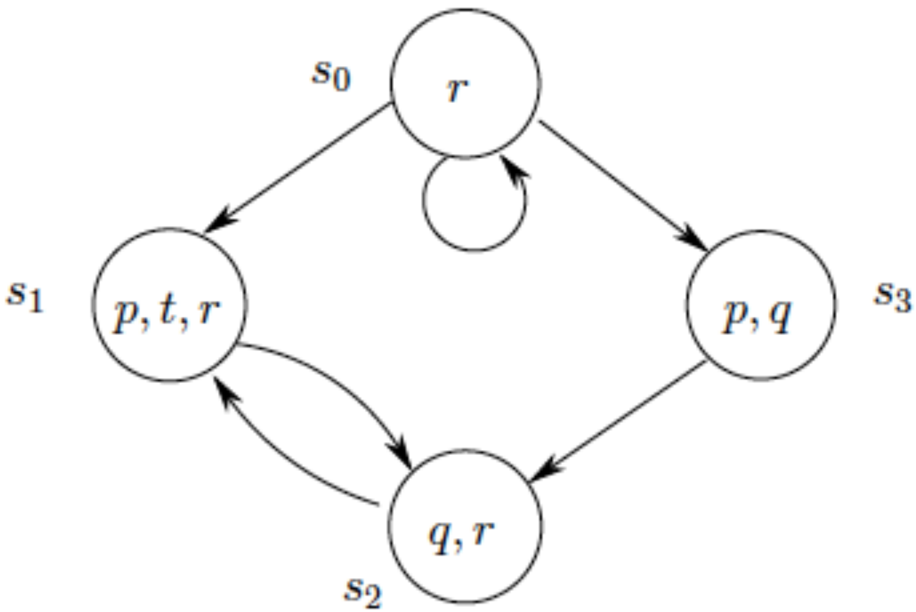


Figure 6.4: Relationship between PLTL, CTL and CTL*

Exercises



Determine whether $\mathcal{M}, s_0 \vDash \phi$ and $\mathcal{M}, s_2 \vDash \phi$ hold and justify your answer, where ϕ is the LTL or CTL formula:

* (i) $\neg p \to r$

(ii) $\mathrm{F}\, t$

*(iii) $\neg \mathrm{EG}\, r$

(iv) $\mathrm{E}\,(t\ \mathrm{U}\ q)$

(v) $\mathrm{F}\, q$

(vi) $\mathrm{EF}\, q$

(vii) $\mathrm{EG}\, r$

(viii) $\mathrm{G}\,(r \vee q)$.

Exercises

Which of the following pairs of CTL formulas are equivalent? For those which are not, exhibit a model of one of the pair which is not a model of the other:

(a) $\mathrm{EF}\,\phi$ and $\mathrm{EG}\,\phi$

(b) $\mathrm{EF}\,\phi \vee \mathrm{EF}\,\psi$ and $\mathrm{EF}\,(\phi \vee \psi)$

(c) $\mathrm{AF}\,\phi \vee \mathrm{AF}\,\psi$ and $\mathrm{AF}\,(\phi \vee \psi)$

(d) $\mathrm{AF}\,\neg\phi$ and $\neg\mathrm{EG}\,\phi$

(e) $\mathrm{EF}\,\neg\phi$ and $\neg\mathrm{AF}\,\phi$

(f) $\mathrm{A}[\phi_1\ \mathrm{U}\ \mathrm{A}[\phi_2\ \mathrm{U}\ \phi_3]]$ and $\mathrm{A}[\mathrm{A}[\phi_1\ \mathrm{U}\ \phi_2]\ \mathrm{U}\ \phi_3]$, hint: it might make it simpler if you think first about models that have just one path

(g) $\top$ and $\mathrm{AG}\,\phi \to \mathrm{EG}\,\phi$

(h) $\top$ and $\mathrm{EG}\,\phi \to \mathrm{AG}\,\phi$.

LTL: what does hold (exercises)

$G\ (\varphi \rightarrow \psi) \rightarrow (G\ \varphi \rightarrow G\ \psi)$
$G\ \varphi \rightarrow \varphi$
$\varphi \rightarrow F\ \varphi$
$G\ \varphi \rightarrow X\varphi$
$X\ \varphi \rightarrow F\varphi$
$G\ \varphi \rightarrow F\varphi$
$X(\varphi \wedge \psi) \cong X\varphi \wedge X\psi$