Master in Artificial Intelligence and Robotics (AIRO)
Electives in AI
Reasoning Agents

Fabio Patrizi

Sapienza University of Rome, Italy
patrizi@diag.uniroma1.it

A.Y. 2021-2022

Computation Tree Logic (CTL)

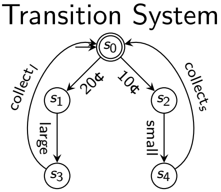
References:

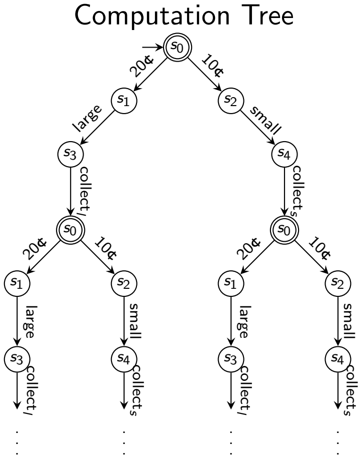
1. A fully detailed presentation of the topics discussed in these slides can be found in [CGP99]

Computation Tree Logic

- CTL [CE81, CGP99]: logic expressing properties about TSs seen as *computation trees*
- Computation tree: "unfolding" of TS
  - formally, tree containing all infinite paths of TS
- *Branching-time*, as opposed to *linear-time*, semantics
- CTL can express:
  - existence of a path satisfying certain properties
  - properties that mix *universal* and *existential* quantification over paths

Computation Trees

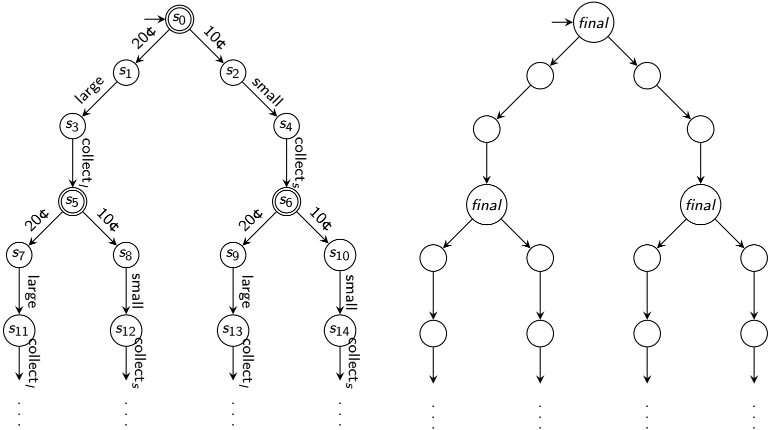




Observe:

- Computation trees are infinite (but have regular structure)
- Transition labels are irrelevant (and will be dropped)

Computation Trees



CTL Examples

With propositions $P = \{p, q, r, \ldots\}$:

1. There exists a path containing a state where p holds
2. There exists no path containing a state where q holds
3. Every path contains always contains either p or q
4. There exists a path such that all the past departing from its states contain a state where q holds

CTL Syntax

Definition (CTL: Syntax)

Let P be a countable set of *atomic propositions*

CTL formulas have the following syntax, with $p \in P$

$$\varphi = p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \textbf{EX}\,\varphi \mid \textbf{EG}\,\varphi \mid \varphi\,\textbf{EU}\,\varphi$$

Intuitions:

- φ: formula φ holds in current state
- $\textbf{EX}\,\varphi$: there exists a path s.t. in the next state φ holds
- $\textbf{EG}\,\varphi$: there exists a path s.t. φ always holds
- $\varphi\,\textbf{EU}\,\psi$: there exists a path s.t. ψ holds sometime in the future and until then φ always holds

Observe that both φ and ψ are CTL formulas themselves

CTL Semantics

- CTL semantics is provided over the computation tree of a TS $\mathcal{T}$
- Defined in terms of *satisfaction* relation $\models$
- For:
  - TS $\mathcal{T} = (P, A, S, s_0, \rightarrow, \lambda)$
  - state $s \in S$
  - a CTL formula φ over P

  we write $\mathcal{T}, s \models \varphi$ if the computation tree of $\mathcal{T}$ rooted in state s *satisfies* φ, as inductively defined next

CTL Semantics

Definition (CTL: Semantics)

Let $\mathcal{T} = (P, A, S, s_0, \rightarrow, \lambda)$ be a labelled transitions system, φ a CTL formula over P, and s_i a state of the computation tree of $\mathcal{T}$.

We inductively define $\mathcal{T}, s_i \models \varphi$ as follows:

- $\mathcal{T}, s_i \models p$ iff $p \in \lambda(s_i)$
- $\mathcal{T}, s_i \models \neg\varphi$ iff it is not the case that $\mathcal{T}, s_i \models \varphi$
- $\mathcal{T}, s_i \models \varphi \wedge \psi$ iff $\mathcal{T}, s_i \models \varphi$ and $\mathcal{T}, s_i \models \psi$
- $\mathcal{T}, s_i \models \mathbf{EX}\,\varphi$ iff $\exists \pi = s_i s_{i+1} \cdots$ s.t. $\mathcal{T}, s_{i+1} \models \varphi$
- $\mathcal{T}, s_i \models \mathbf{EG}\,\varphi$ iff $\exists \pi = s_i s_{i+1} \cdots$ s.t. $\mathcal{T}, s_j \models \varphi$, for all $j \geq i$
- $\mathcal{T}, s_i \models \varphi\,\mathbf{EU}\,\psi$ iff $\exists \pi = s_i s_{i+1} \cdots$ s.t. $\mathcal{T}, s_k \models \psi$, for some $k \geq i$ and $\mathcal{T}, s_j \models \varphi$ for all $j = i, \ldots, k-1$

Where $\pi = s_i s_{i+1} \cdots$ is an *infinite* path of $\mathcal{T}$ starting from s_i

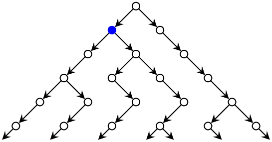
CTL Semantics

Definition (CTL: Semantics)

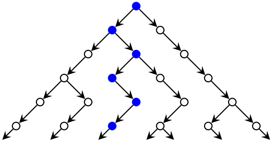
Let $\mathcal{T} = (P, A, S, s_0, \to, \lambda)$ be a LTS and φ a CTL formula over P.

We say that $\mathcal{T}$ *satisfies* φ, written $\mathcal{T} \models \varphi$, if $\mathcal{T}, s_0 \models \varphi$.

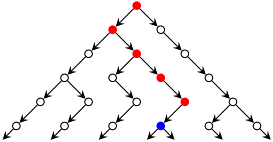
EX p



EG p



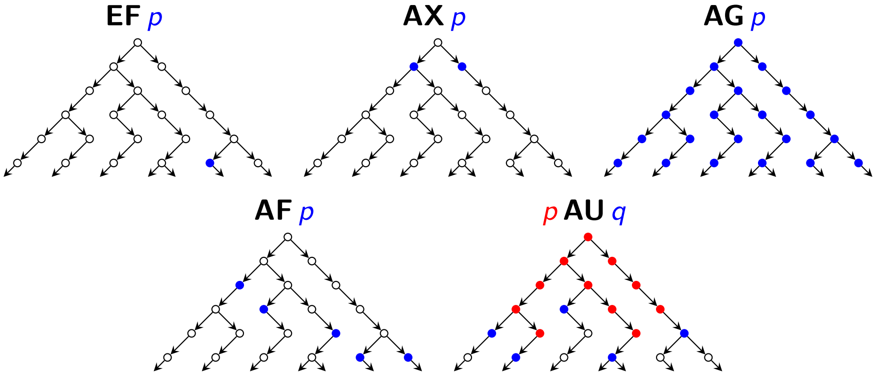
p **EU** q



CTL Syntax: Abbreviations

Abbreviations:

- $\varphi \lor \psi = \neg(\neg\varphi \land \neg\psi)$
- $\varphi \rightarrow \psi = \neg\varphi \lor \psi$
- $\mathbf{EF}\,\varphi = \top\,\mathbf{EU}\,\varphi$ (there exists a path s.t. φ eventually holds)
- $\mathbf{AX}\,\varphi = \neg\,\mathbf{EX}\,\neg\varphi$ (for all paths, φ holds next)
- $\mathbf{AG}\,\varphi = \neg\,\mathbf{EF}\,\neg\varphi$ (for all paths, φ always holds)
- $\mathbf{AF}\,\varphi = \neg\,\mathbf{EG}\,\neg\varphi$ (for all paths, φ eventually holds)
- $\varphi\,\mathbf{AU}\,\psi = \mathbf{AF}\,\psi \land \neg(\neg\psi\,\mathbf{EU}(\neg\varphi \land \neg\psi))$ (for all paths, φ holds until ψ)


EF p
AX p
AG p
AF p
p AU q

CTL: Examples

Safety properties (nothing bad will happen):

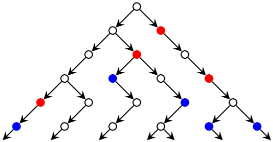
- $\mathbf{AG}\,\neg(green_1 \wedge green_2)$
  (traffic lights 1 and 2 are never green at the same time)
- $\mathbf{AG}\,\neg(altitude < 0)$
  (plane altitude is never negative)

Liveness properties (something good will happen):

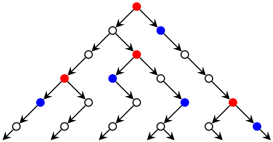
- $\mathbf{AF}(land \wedge stop)$
  (airplane will eventually land and stop)
- $\mathbf{AG}(work \rightarrow \mathbf{AF}\,get\_salary)$
  (it is always the case that if one works, (s)he is eventually paid)
- $\mathbf{AG}(play \rightarrow \mathbf{EX}\,win)$
  (it is always the case that if one plays, (s)he can win)

CTL: Examples

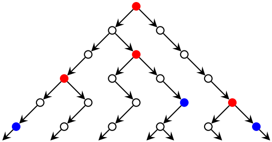
$\mathbf{AG}(\textit{work} \rightarrow \mathbf{AF}\, \textit{get\_salary})$



$\mathbf{AG}(\textit{play} \rightarrow \mathbf{EX}\, \textit{win})$



$\mathbf{AG}(\textit{play} \rightarrow \mathbf{EF}\, \textit{win})$



The CTL Model Checking Problem

CTL Model Checking

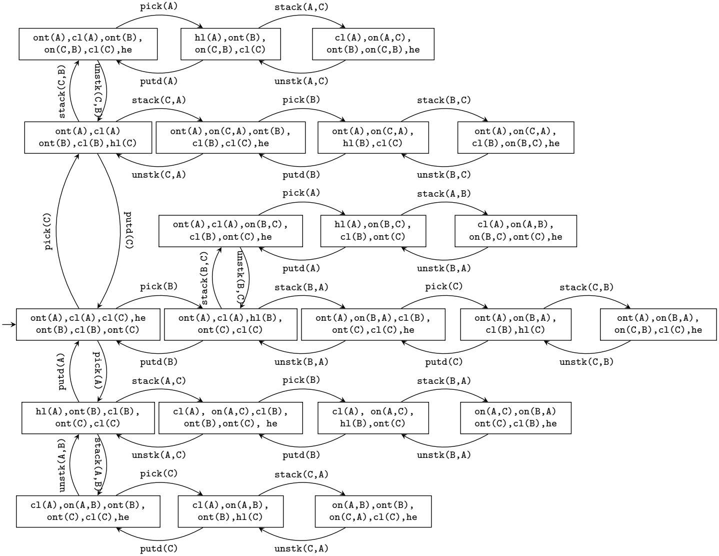
Given:

- A LTS $\mathcal{T} = (P, A, S, s_0, \rightarrow, \lambda)$
- A CTL formula φ

Check whether $\mathcal{T} \models \varphi$

CTL Model Checking

Example

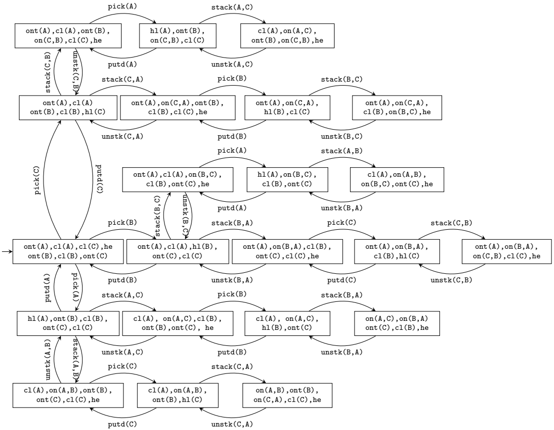


CTL Model Checking
Example

1. $\mathcal{T} \models \textbf{EF}(on(B,A) \wedge on(C,B) \wedge ont(A))$?
2. $\mathcal{T} \models \textbf{EF}(holding \wedge \textbf{EX}\, \neg he)$?
3. $\mathcal{T} \models \textbf{AG}(holding \rightarrow \textbf{AX}\, he)$?
4. $\mathcal{T} \models \textbf{AG}\,\textbf{AF}(on(B,A))$?
5. $\mathcal{T} \models \textbf{AG}(clear(c) \rightarrow \textbf{EF}\, \neg clear(C))$?
6. $\mathcal{T} \models \textbf{EG}\,\textbf{EF}(on(B,A))$?
7. $\mathcal{T} \models \textbf{EF}\,\textbf{AG}(\neg on(B,A))$?

with

- $holding = hl(A) \vee hl(B) \vee hl(C)$



Observe: must explore infinitely many, infinite-length paths!

Branching-time vs. linear-time

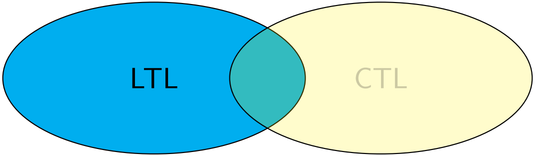
CTL has a *branching-time* semantics

- Properties of computation tree, not of single paths
- Cannot express *strong fairness*: for every path, if p occurs infinitely often, then q occurs infinitely often

Addressed by *linear-time temporal logic* (LTL [Pnu77, CGP99]):

- Expresses properties of *paths* (typically of a TS)
- Can express strong fairness
- Cannot quantify existentially over paths:
  - E.g., cannot express CTL formula: $\mathbf{AG}(p \rightarrow \mathbf{EF}\, q)$

LTL and CTL have incomparable expressive power



The Linear-time Temporal Logic

LTL (Linear-time temporal logic)

- Expresses properties of a single (infinite) path
- No path quantifiers

LTL Syntax

Definition (LTL: Syntax)

Let P be a countable set of *atomic propositions*

LTL formulas have the following syntax, with $p \in P$

$$\varphi = p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

Intuitions:

- φ: formula φ holds in current state
- $\mathbf{X}\,\varphi$: φ holds in next state
- $\varphi\,\mathbf{U}\,\psi$: ψ holds sometime in the future and until then φ always holds

Observe that both φ and ψ are LTL formulas

LTL Semantics

- LTL semantics is provided over infinite paths (of a TS)
- Defined in terms of *satisfaction* relation $\models$

LTL Semantics

Definition (LTL: Semantics)

Given

- A path $\pi = s_0 s_1 \cdots$ (of some LTS $\mathcal{T} = (P, A, S, s_0, \rightarrow, \lambda)$)
- A state s_i of π
- An LTL formula φ over P

we inductively define $\mathcal{T}, s_i \models \varphi$ as follows:

- $\pi, s_i \models p$ iff $p \in \lambda(s_i)$
- $\pi, s_i \models \neg\varphi$ iff it is not the case that $\pi, s_i \models \varphi$
- $\pi, s_i \models \varphi \wedge \psi$ iff $\pi, s_i \models \varphi$ and $\pi, s_i \models \psi$
- $\pi, s_i \models \mathbf{X}\,\varphi$ iff $\pi, s_{i+1} \models \varphi$
- $\pi, s_i \models \varphi\,\mathbf{U}\,\psi$ iff $\pi, s_k \models \psi$, for some $k \geq i$ and $\pi, s_j \models \varphi$ for all $j = i, \ldots, k-1$

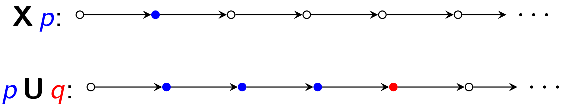
LTL Semantics

Definition (LTL: Semantics)

Let $\mathcal{T} = (P, A, S, s_0, \rightarrow, \lambda)$ be a LTS and φ an LTL formula over P

We say that $\mathcal{T}$ *satisfies* φ, written $\mathcal{T} \models \varphi$, if for all paths π of $\mathcal{T}$, we have that $\pi, s_0 \models \varphi$.

LTL Semantics

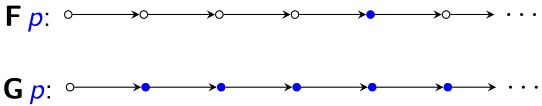


LTL Syntax: Abbreviations

Abbreviations:

- $\vee$ and $\rightarrow$ are as usual
- $\mathbf{F}\,\varphi = \top\,\mathbf{U}\,\varphi$ (φ eventually holds)
- $\mathbf{G}\,\varphi = \neg\,\mathbf{F}\,\neg\varphi = \neg(\top\,\mathbf{U}\,\neg\varphi)$ (φ always holds)

LTL Semantics



LTL: Examples

Safety properties (nothing bad will happen):

- $\mathbf{G}\,\neg(green_1 \wedge green_2)$
  (traffic lights 1 and 2 are never green at the same time)
- $\mathbf{G}\,\neg(altitude < 0)$
  (plane altitude is never negative)

Liveness properties (something good will happen):

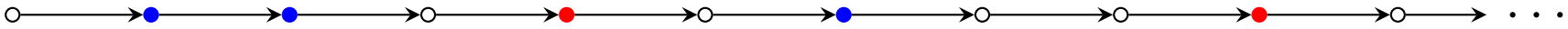
- $\mathbf{F}(land \wedge stop)$
  (airplane will eventually land and stop)
- $\mathbf{G}(work \rightarrow \mathbf{F}\, get\_salary)$
  (it is always the case that if one works, (s)he is eventually paid)
- $\mathbf{G}(play \rightarrow \mathbf{X}\, win)$
  (it is always the case that if one plays, (s)he can win)

Observe:

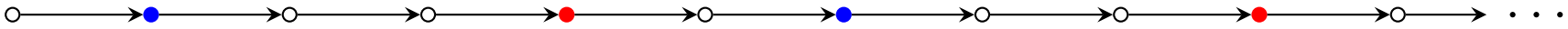
- This time formulas are interpreted over paths (not computation trees)

LTL: Examples

$$\mathbf{G}(\mathit{work} \rightarrow \mathbf{F}\ \mathit{get\_salary})$$



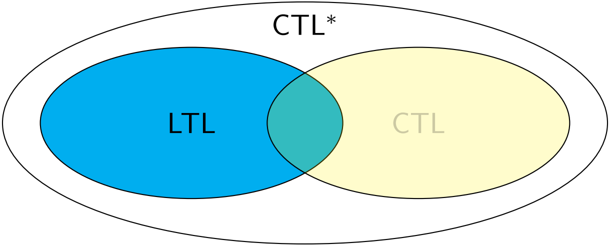
$$\mathbf{G}\,\mathbf{F}\ \mathit{play} \rightarrow \mathbf{G}\,\mathbf{F}\ \mathit{win}$$



Path Quantification and CTL*

CTL limitation: only certain combinations of *path quantifiers* and *temporal modalities* allowed, e.g.:

- Cannot express: for every path π s.t. eventually p there exists a path π' s.t. eventually q
- Solved by CTL*[EH83, EH86, CGP99] (not seen in this course)



Summary

- Computation-tree logic (CTL) can be used to express properties of TSs
- Interpreted over infinite computation trees of TSs
- Captures branching-time properties of practical interest
- CTL Model checking is the problem of checking whether a TS satisfies a CTL formula
- Other logics exist:
  - LTL (linear-time): incomparable to CTL (non-null intersection)
  - CTL^*: strictly more expressive than LTL and CTL

References I

Edmund M. Clarke and E. Allen Emerson.
Design and synthesis of synchronization skeletons using branching-time temporal logic.
In Dexter Kozen, editor, *Logics of Programs, Workshop, Yorktown Heights, New York, USA, May 1981*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.

Edmund M Clarke, Orna Grumberg, and Doron A. Peled.
Model checking.
MIT Press, London, Cambridge, 1999.

E. Allen Emerson and Joseph Y. Halpern.
"sometimes" and "not never" revisited: On branching versus linear time.
In John R. Wright, Larry Landweber, Alan J. Demers, and Tim Teitelbaum, editors, *Conference Record of the Tenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 1983*, pages 127–140. ACM Press, 1983.

References II

E. Allen Emerson and Joseph Y. Halpern.
"sometimes" and "not never" revisited: on branching versus linear time temporal logic.
J. ACM, 33(1):151–178, 1986.

Amir Pnueli.
The temporal logic of programs.
In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 46–57. IEEE Computer Society, 1977.