

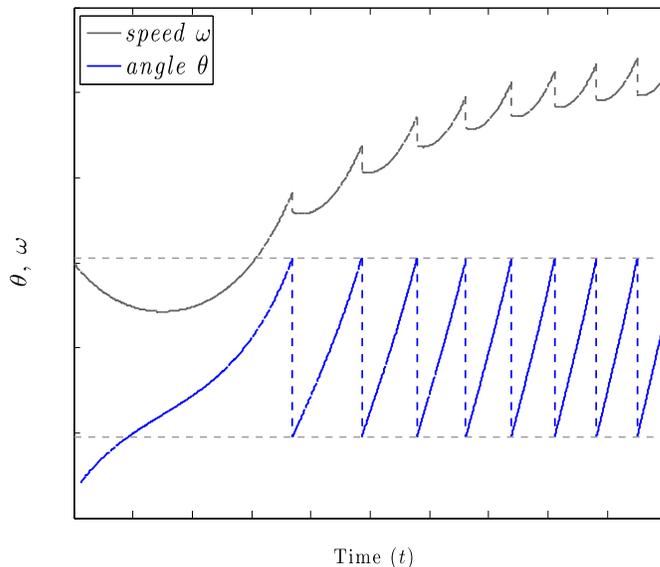
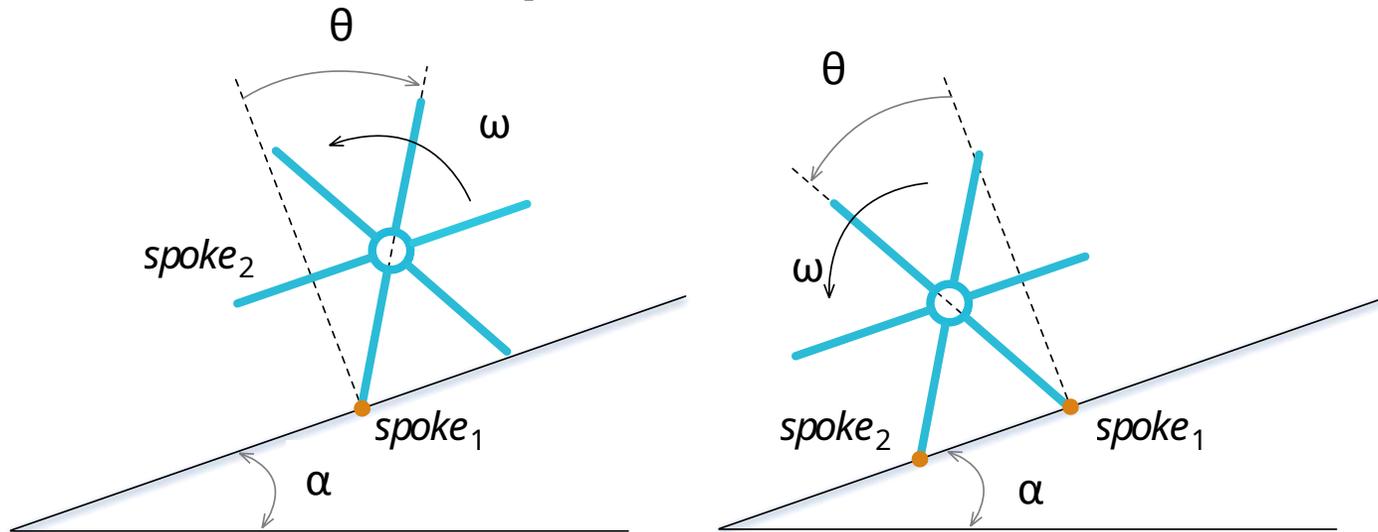
Lecture 18: Invariants in Cyberphysical Systems

Verification requirements, Temporal logics

Huan Zhang

huan@huan-zhang.com

Review: Hybrid Automaton



automaton RimlessWheel(α, μ : Real, n : Nat)

const β : Real := $2 \pi / n$

type Spokes: enumeration [1,...,n]

actions

impact

variables

pivot: Spokes := 1

θ : Real := 0

ω : Real := 0

transitions

impact

pre $\theta \geq \beta / 2$

eff pivot := pivot + 1 mod n

$\theta := -\beta / 2$

$\omega := \mu \omega$

trajectories

swing

evolve

$d(\theta) = \omega$

$d(\omega) = \sin(\theta + \alpha)$

invariant $\theta \leq \frac{\beta}{2}$

Invariants and reachability

- A state x of automaton \mathcal{A} is **reachable** if there exists an execution α with $\alpha.lstate = x$
- $Reach_{\mathcal{A}}(\Theta)$ is the set of all reachable state from Θ
- $Reach_{\mathcal{A}}(\Theta, T)$ is the set of states reachable within time T
- $Reach_{\mathcal{A}}(\Theta, k)$ is the set of states reachable within k transitions
- $Reach_{\mathcal{A}}(\Theta, T, k)$ is the set of states reachable up to time k transitions and time T
- An invariant $I \subseteq val(X)$ is a set of states that contains $Reach_{\mathcal{A}}(\Theta)$

Review: Hybrid Automaton

$$\mathcal{A} = (X, \Theta, A, \mathcal{D}, \mathcal{T})$$

- X : set of **state variables**
 - $Q \subseteq \text{val}(X)$ set of **states**
- $\Theta \subseteq Q$ set of **start states**
- set of **actions**, $A = E \cup H$
- $\mathcal{D} \subseteq Q \times A \times Q$
- \mathcal{T} : set of **trajectories**

automaton RimlessWheel(α, μ : Real, n : Nat)

const β : Real := $2 \pi / n$

type Spokes: enumeration [1,...,n]

actions

impact

variables

pivot: Spokes := 1

θ : Real := 0

ω : Real := 0

transitions

impact

pre $\theta \geq \beta / 2$

eff pivot := pivot + 1 mod n

$\theta := -\beta / 2$

$\omega := \mu \omega$

trajectories

swing

evolve

$d(\theta) = \omega$

$d(\omega) = \sin(\theta + \alpha)$

invariant $\theta \leq \frac{\beta}{2}$

How to prove invariants of hybrid automata?

Theorem 7.1. Given an HIOA $\mathcal{A} = \langle X, \Theta, A, \mathbf{D}, \mathbf{T} \rangle$, if a set of states $I \subseteq \text{val}(X)$ satisfies the following:

- (Start condition) For any starting state $x \in \Theta$, $x \in I$ and
- (Transition closure) For any action $a \in A$, if $x \rightarrow_a x'$ and $x \in I$ then $x' \in I$, and
- (**Trajectory closure**) For **any** trajectory $\tau \in \mathbf{T}$ if $\tau.fstate \in I$ then $\tau.lstate \in I$

Then I is an inductive invariant of \mathcal{A} .

How to prove invariants of hybrid automata

Theorem 7.1. Given an HIOA $\mathcal{A} = \langle X, \Theta, A, \mathbf{D}, \mathbf{T} \rangle$, if a set of states $I \subseteq \text{val}(X)$ satisfies the following:

- (Start condition) For any starting state $x \in \Theta$, $x \in I$ and
- (Transition closure) For any action $a \in A$, if $x \rightarrow_a x'$ and $x \in I$ then $x' \in I$, and
- (Trajectory closure) For any trajectory $\tau \in \mathbf{T}$ if $\tau.fstate \in I$ then $\tau.lstate \in I$

Then I is an inductive invariant of \mathcal{A} .

Proof. Consider any reachable state $x \in \text{Reach}_{\mathcal{A}}$. By the definition of a reachable state, there exists an execution α of \mathcal{A} with $\alpha.lstate = x$. We proceed by induction on the length of the execution α . For the base case, α consists of a single starting state $x \in \Theta$, and, by the *start condition*, $x \in I$. For the inductive step, we consider two subcases:

Case 1: $\alpha = \alpha' a p(x)$, where $a \in A$ and $p(x)$ is a point trajectory at x .

By the induction hypothesis, we know that $\alpha'.lstate \in I$.

By invoking the *transition closure*, we obtain $x \in I$.

Case 2: $\alpha = \alpha' \tau$, where τ is a trajectory of \mathcal{A} and $\tau.lstate = x$

By the *induction hypothesis*, $\alpha'.lstate \in I$ and by

invoking the *trajectory closure*, we deduce that $\tau.lstate = x \in I$

An application

automaton Bouncingball(c,h,g)

variables: x : Reals := h , v : Reals := 0

actions: bounce

transitions:

bounce

pre $x = 0 \wedge v < 0$

eff $v := -cv$

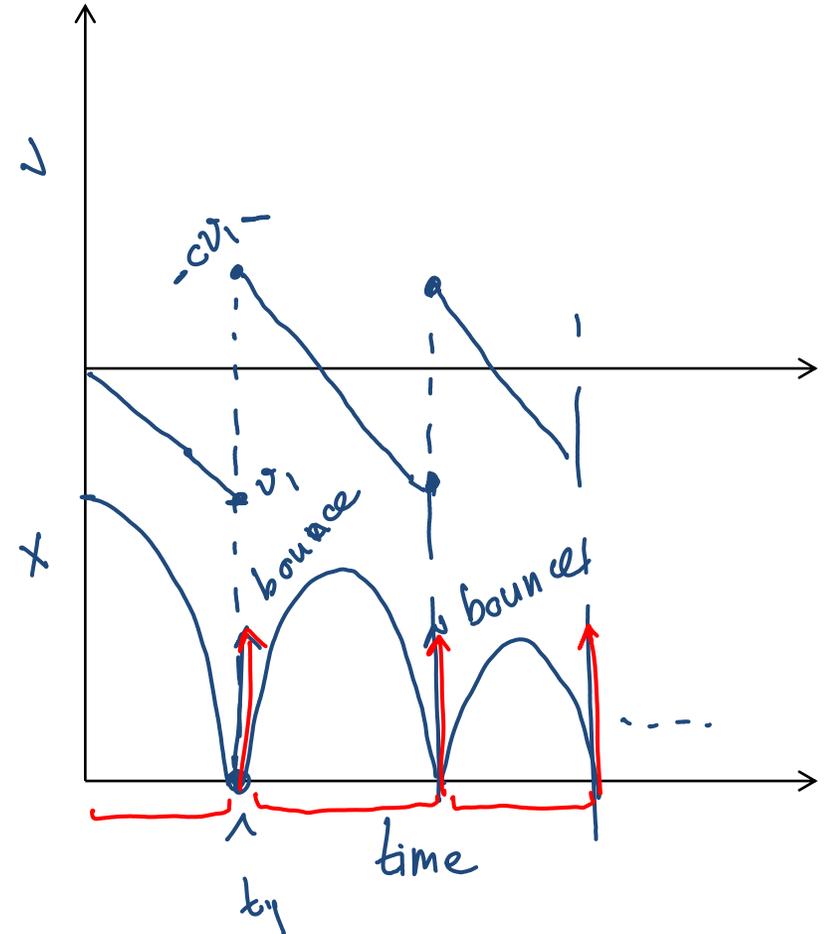
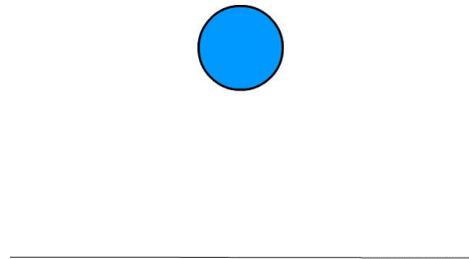
trajectories:

freefall

evolve $d(x) = v$; $d(v) = -g$

invariant $x \geq 0$

Can you find some candidate invariant for the simple Bouncingball hybrid automaton?



An application

automaton Bouncingball(c,h,g)

variables: x: Reals := h, v: Reals := 0

actions: bounce

transitions:

bounce

pre $x = 0 \wedge v < 0$

eff $v := -cv$

trajectories:

freefall

evolve $d(x) = v; d(v) = -g$

invariant $x \geq 0$

Candidate invariant: “stays above ground”

$I_0: x \geq 0 \equiv \{ \mathbf{u} \in \text{val}(\{x, v\}) \mid \mathbf{u}[x \geq 0] \}$

Applying Theorem 7.1:

- Consider any initial state $\mathbf{u} \in \Theta; \mathbf{u}[x = h \geq 0]$
 - $\mathbf{u} \in I_0$
- Consider any transition $\mathbf{u} \rightarrow_{\text{bounce}} \mathbf{u}'$
 - From precondition we know $\mathbf{u}[x = 0]$; from effect we know $\mathbf{u}'.x = \mathbf{u}.x$ therefore $\mathbf{u}'[x = 0 \geq 0]$
 - $\mathbf{u}' \in I_0$
- Consider any trajectory $\tau \in T$
 - From mode invariant we know that for $\forall t \in \tau. \text{dom}, \tau(t)[x \geq 0]$
 - It follows that $\tau. \text{lstate}[x \geq 0]$

An application

automaton Bouncingball(c,h,g)

variables: x: Reals := h, v: Reals := 0

actions: bounce

transitions:

bounce

pre $x = 0 \wedge v < 0$

eff $v := -cv$

trajectories:

freefall

evolve $d(x) = v; d(v) = -g$

invariant $x \geq 0$

Candidate invariant: “stays above ground and below h”

$$I_h: h \geq x \geq 0$$

Applying Theorem 7.1:

- Consider any initial state $\mathbf{u} \in \Theta$; $\mathbf{u}[x = h]$
 - $\mathbf{u} \in I_h$
- Consider any transition $\mathbf{u} \rightarrow_{\text{bounce}} \mathbf{u}'$
 - From precondition we know $\mathbf{u}[x = 0]$; from effect we know $\mathbf{u}'.x = \mathbf{u}.x$ therefore $\mathbf{u}'[x = 0]$
 - $\mathbf{u}' \in I_h$
- Consider any trajectory $\tau \in T$
 - From mode invariant and inductive hypothesis we know that for $\forall t \in \tau. \text{dom}$, $\tau(t)[x \geq 0$ and $\tau(0)[x \in [0, h]$ and that τ is a solution of $d(x) = v; d(v) = -g$
 - Is this adequate to infer $\tau. \text{lstate} \in I_h$?

An application

automaton Bouncingball(c,h,g)

variables: x: Reals := h, v: Reals := 0

actions: bounce

transitions:

bounce

pre $x = 0 \wedge v < 0$

eff $v := -cv$

trajectories:

freefall

evolve $d(x) = v; d(v) = -g$

invariant $x \geq 0$

Candidate invariant: “stays above ground and below h”

$I_h: h \geq x \geq 0$

Applying Theorem 7.1:

- Consider any initial state $\mathbf{u} \in \Theta$; $\mathbf{u}[x = h]$
 - $\mathbf{u} \in I_h$
- Consider any transition $\mathbf{u} \rightarrow_{\text{bounce}} \mathbf{u}'$
 - From precondition we know $\mathbf{u}[x = 0]$; from effect we know $\mathbf{u}'.x = \mathbf{u}.x$ therefore $\mathbf{u}'[x = 0]$
 - $\mathbf{u}' \in I_h$
- Consider any trajectory $\tau \in T$
 - From mode invariant and inductive hypothesis we know that for $\forall t \in \tau. \text{dom}$, $\tau(t)[x \geq 0$ and $\tau(0)[x \in [0, h]$ and that τ is a solution of $d(x) = v; d(v) = -g$

No, $I_h: h \geq x \geq 0$ is not an **inductive invariant!** velocity unconstrained. How to fix it?

Strengthened invariant

automaton Bouncingball(c,h,g)

variables: x: Reals := h, v: Reals := 0

k: Nat := 0

actions: bounce

transitions:

bounce

pre $x = 0 \wedge v < 0$

eff $v := -cv; k := k + 1$

trajectories:

freefall

evolve $d(x) = v; d(v) = -g$

invariant $x \geq 0$

Candidate invariant: “stays above ground and below h”

To prove using inductive invariant, strength it!

$$I_v: v^2 - 2g(hc^{2k} - x) = 0$$

k is the number of bounces capturing the velocity information.

Applying Theorem 7.1:

- Consider any initial state $\mathbf{u} \in \Theta$; $\mathbf{u}[x = h; \mathbf{u}[k = 0$
 - $\mathbf{u} \in I_v$
- **Exercise:** Finish the rest

Strengthened invariant

automaton Bouncingball(c,h,g)

variables: x: Reals := h, v: Reals := 0

k: Nat := 0

actions: bounce

transitions:

bounce

pre $x = 0 \wedge v < 0$

eff $v := -cv; k := k + 1$

trajectories:

freefall

evolve $d(x) = v; d(v) = -g$

invariant $x \geq 0$

Candidate invariant: “stays above ground and below h”

To prove using inductive invariant, strength it!

$$I_v: v^2 - 2g(hc^{2k} - x) = 0$$

$$I_h: h \geq x \geq 0$$

Applying Theorem 7.1 you can prove that $I_v \wedge I_h$ is an (stronger) inductive invariant. So I_h is proved.

Summary: invariants of hybrid automaton

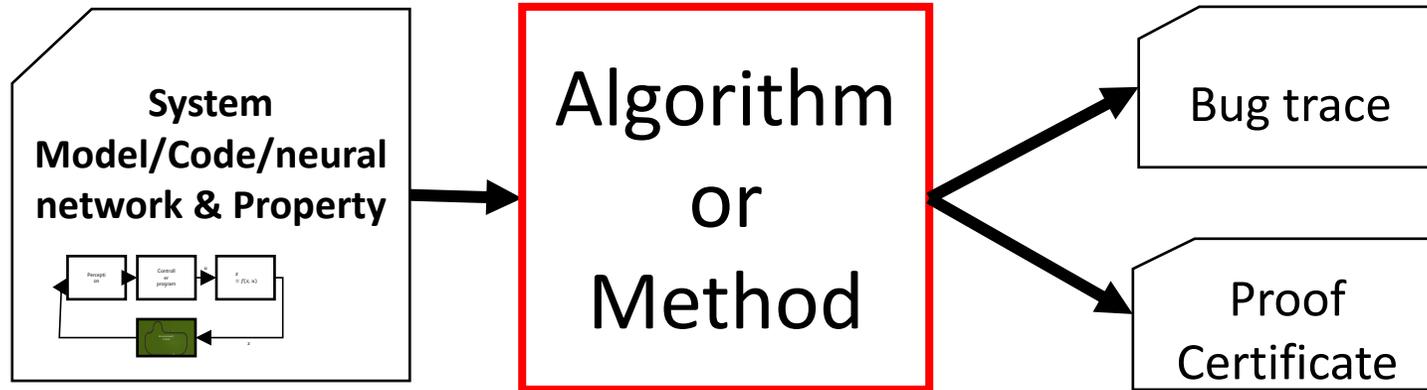
- Theorem 7.1 gives a sufficient condition for proving **inductive** invariants
- Not all invariants are inductive
- We often have to **strengthen** invariants to make them inductive
- Invariants can also be proven using alternative methods (e.g., barrier functions, subangular conditions)
- Read examples in Chapter 7

Back to the verification problem

System. A program/system for *lane keeping control* for vehicles

Model/assumptions: automaton, ODEs, hybrid automata

Requirement. The vehicle does not go outside the lane boundaries



counterexample. A particular environment situation (lane geometry, sensor failure, computer configuration) that makes the vehicle go outside lanes

A mathematical proof that establishes that for all *allowed* inputs and environments the vehicle stays with the lane

Verification tool

When can we build such a tool? How expensive is it? How well is it going to work? Under what assumptions?

Requirements and safety in the real world

Requirements analysis: Set of tasks that ultimately lead to the determination and documentation of the design requirements that the product must meet:

E.g. “0 to 60 mph in 4 seconds on flat road”,

“Petrol car can emit no more than 60mg/km” EURO 6.

Safety standards: Provide *guidelines* and *processes* for developing safety-critical systems.

E.g. DO-178C standard is enforced by the FAA for certifying aviation software

ISO2626 is used for functional safety of cars

Standards for Advanced Autonomous and AI-enabled systems are being developed

Requirements thus far: Invariants and stability

Models **automaton, hybrid automaton** $\mathcal{A} = \langle X, \Theta, A, \mathcal{D}, T \rangle$

Requirements: $I \subseteq \text{val}(X)$, such that $\text{Reach}_{\mathcal{A}}(\Theta) \subseteq I$

Given an **unsafe set** $U \subseteq \text{val}(X)$ we can check whether $I \cap U = \emptyset$ to infer that $\text{Reach}_{\mathcal{A}}(\Theta) \cap U = \emptyset$

Asymptotic stability: Does $\alpha(x_0, t) \rightarrow 0$ as $t \rightarrow \infty$.

What are the requirements that we haven't discussed yet?

What about more general types of requirements, e.g.,

“**Eventually** the light turns red and prior to that the orange light blinks”

“After failures, **eventually** there is just one token in the system”

How to express and verify such properties?

Introduction to temporal logics

Temporal logics: Formal language for representing, and reasoning about, propositions qualified in terms of a **sequence**

Amir Pnueli received the ACM Turing Award (1996) for seminal work introducing temporal logic into computer science and for outstanding contributions to program verification.



Large follow-up literature, e.g., different temporal logics MTL, MITL, PCTL, ACTL, STL, applications in synthesis and monitoring

Setup: States are labeled

We have a set of **atomic propositions (AP)**

These are the properties that hold in each state, e.g., “light is green”, “has 2 tokens”

We have a **labeling function** that assigns to each state, a set of propositions that hold at that state

$$L: Q \rightarrow 2^{AP}$$

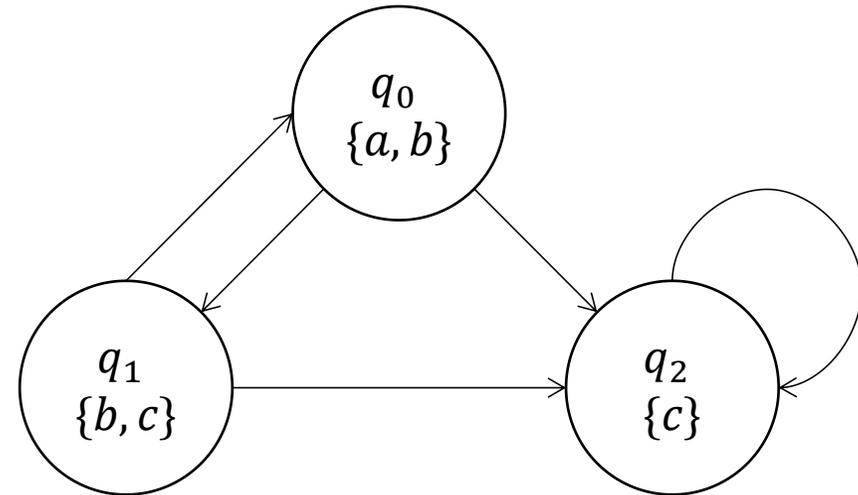
Notations

Automata with state labels but no action labels (“Kripke structure”)

$$\mathcal{A} = \langle Q, Q_0, T, L \rangle$$

$$AP = \{a, b, c\}$$

$$L(q_0) = \{a, b\}$$

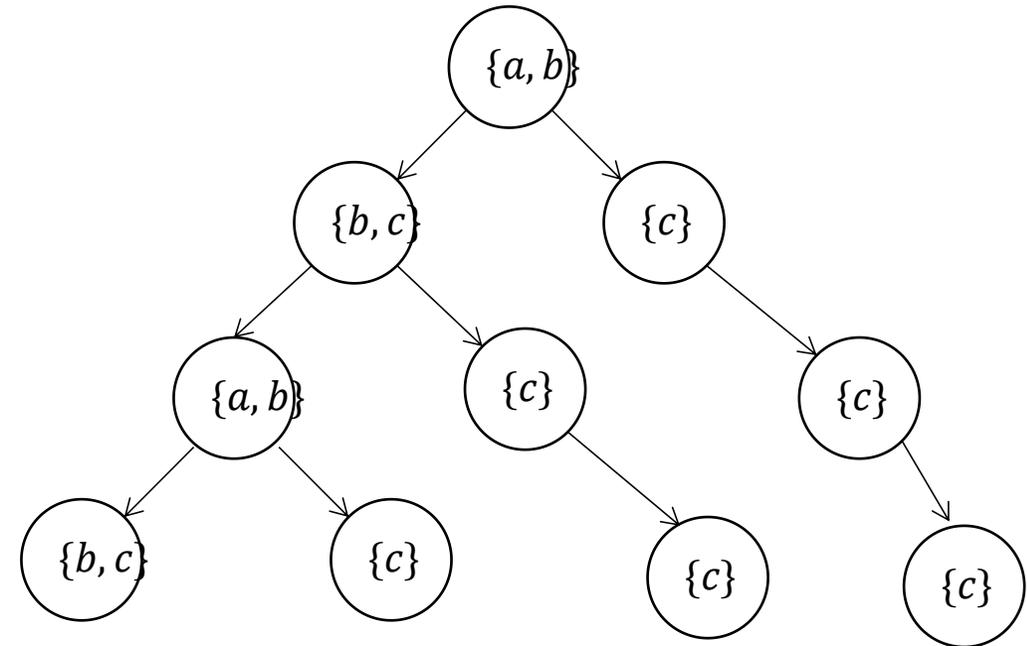
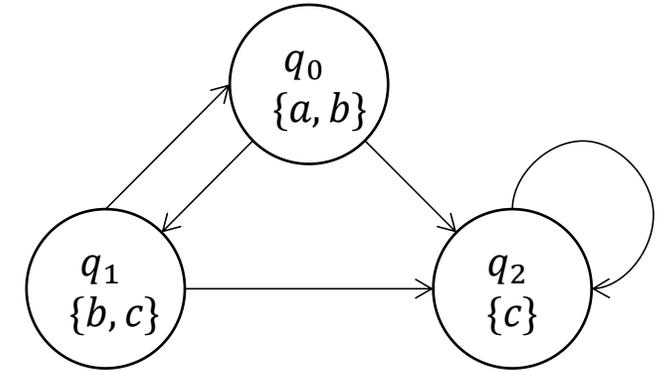


Computational tree logic (CTL)

Unfolding the automaton

We get a tree, representing all possible computations

A **CTL formula** allows us to specify subsets of paths in this tree



CTL quantifiers

Path quantifiers

E: Exists some path

A: All paths

Temporal operators

X: Next state

U: Until

F: Eventually

G: Globally (Always)

ϕ : “no collision”

Invariance: $AG\phi$

ϕ : “one token”

Stabilization: $AF\phi$

Visualizing CTL semantics

Path quantifiers

E: Exists some path

A: All paths

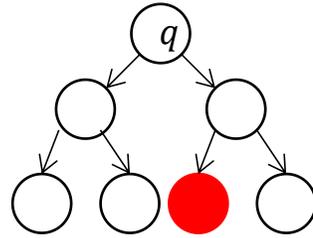
Temporal operators

X: Next state

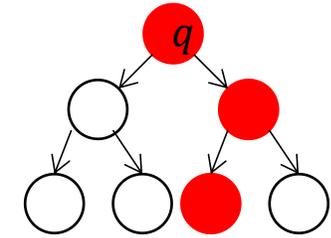
U: Until

F: Eventually

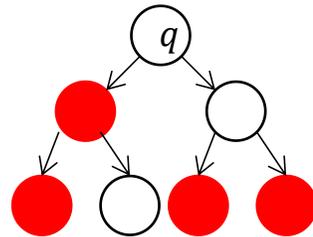
G: Globally (Always)



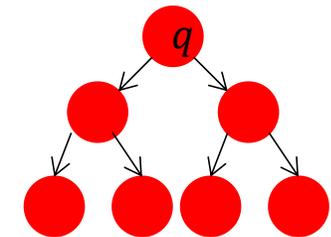
$q \models EF \text{ red}$



$q \models EG \text{ red}$



$q \models AF \text{ red}$



$q \models AG \text{ red}$

Visualizing CTL semantics

Path quantifiers

E: Exists some path

A: All paths

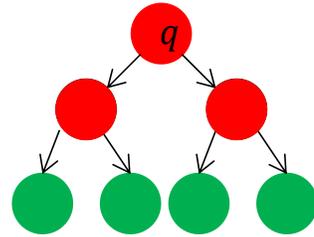
Temporal operators

X: Next state

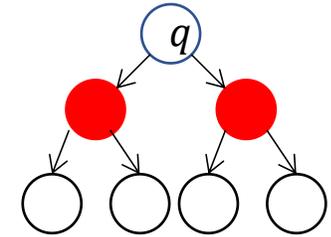
U: Until

F: Eventually

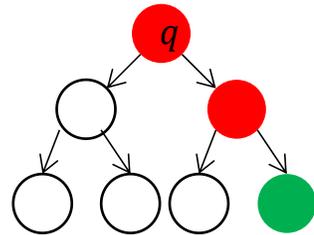
G: Globally (Always)



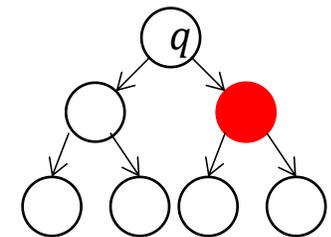
$$q \models A [red \ U \ green]$$



$$q \models AX \ red$$



$$q \models E [red \ U \ green]$$



$$q \models EX \ red$$

CTL syntax

CTL syntax

State Formula (SF) ::= $true \mid p \mid \neg f_1 \mid f_1 \wedge f_2 \mid E \phi \mid A \phi$

Path Formula (PF) ::= $Xf_1 \mid f_1 U f_2 \mid Gf_1 \mid Ff_1$

where $p \in AP$, $f_1, f_2 \in SF$, $\phi \in PF$

Examples:

everything in the previous two slides;

AG ($p_1 \Rightarrow$ **AF** p_2)

Non-examples

$AXX a$; path and state operators must alternate in CTL

CTL syntax

CTL syntax

State Formula (SF) ::= $true \mid p \mid \neg f_1 \mid f_1 \wedge f_2 \mid E \phi \mid A \phi$

Path Formula (PF) ::= $Xf_1 \mid f_1 U f_2 \mid Gf_1 \mid Ff_1$

where $p \in AP$, $f_1, f_2 \in SF$, $\phi \in PF$

Depth of formula: number of production rules used

$EX a$; (depth 3)

$AX EX a$; (depth 5)

$AG AF \text{ green}$; (depth 5)

$AF AG \text{ single token}$ (depth 5)