Specification of Systems with Temporal Logic

Simon Robillard

Autumn 2019

Section 1

Introduction

Before We Begin

1. Exercises during the lecture
2. Keep track of answers, as they might be re-used in later exercises

Contact: simon.robillard@imt-atlantique.fr

Verification, Abstraction and Specification

- In previous part of course: modelling of systems
- To do verification, we need a way to express what is expected of the system
- The expected behavior is often described in natural language

Example: RFC 793 (Transmission Control Protocol)

"The TCP then waits until its own FIN is acknowledged whereupon it deletes the connection."

- Natural language cannot be understood by computers
- Even for humans, it can be ambiguous
- We need a formal language, with precise semantics, to describe what we want

Properties of a Logic of Time

What is the nature of time?

- Discrete or continuous?
- Deterministic or not?
- Does it have a beginning? An end?
- If two events happen simultaneously, are they the same?

What properties do we need to specify?

- Time points (instants) or time intervals?
- Describe the future, the past, or both?
- Need to describe that something is true sometimes? a certain (precise) number of times? infinitely often?
- Probabilities?

LTL and CTL

- The two most common temporal logics for computer science
- Discrete time in both (common view in computer science, since computers follow a clock)

Linear Temporal Logic

- interpreted over timelines, where every moment has a unique successor
- one timeline = one run of a program/system
- we are generally interested in checking that a formula is true for all possible runs

Computational Tree Logic

- interpreted over "time trees": at a given point, there can be more than one possible future
- one tree already captures the different possible runs of a program/system

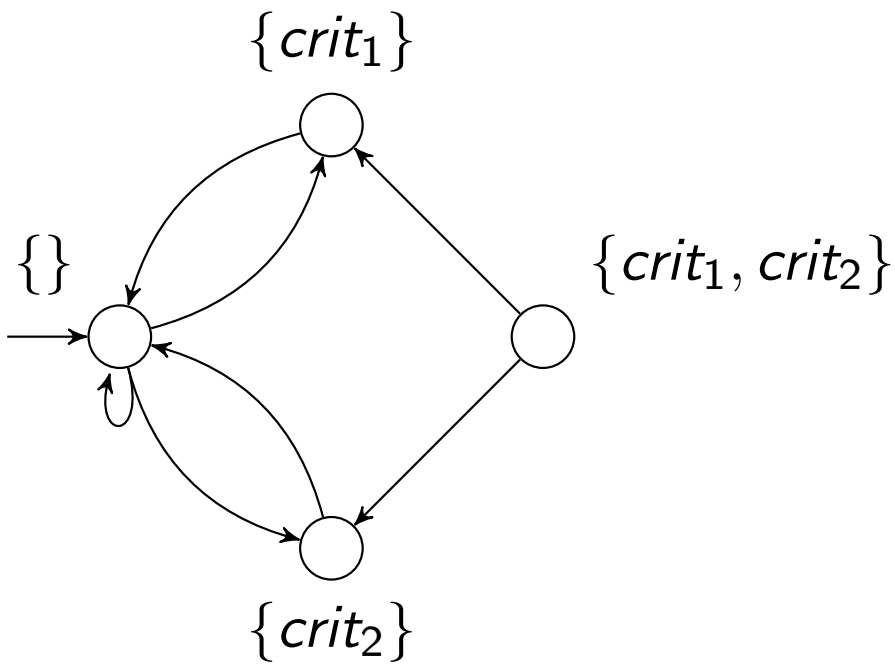
Labelled Transition Systems (LTS)

- until now, we have mainly considered transitions (actions)
- this can make it difficult to express some properties
- example: to check that 2 processes are never in a critical section at the same time, we need to check all the sequences of actions that can lead to this situation
- instead, we can label the states of our system with some properties (e.g., "process X is in the critical section")

Boolean Abstractions

- we use Boolean variables for our labels: for a set of variables P, states are labelled with elements of 2^P
- if $p \in L(\sigma)$, then p is true in state σ, otherwise $\neg p$ is true
- in theory, we can use a countably infinite set of variables
- in practice, you may need a coarse abstraction

Example: two processes in a critical section, $P = \{crit_1, crit_2\}$



Section 2

Linear Temporal Logic: Syntax and Semantics

Syntax of LTL

Let P be a set of propositional variables. The syntax of LTL is defined inductively:

- propositions: if $p \in P$, then p is an LTL formula
- Boolean operators: if φ and ψ are LTL formulas, then
  - $\neg\varphi$
  - $\varphi \vee \psi$

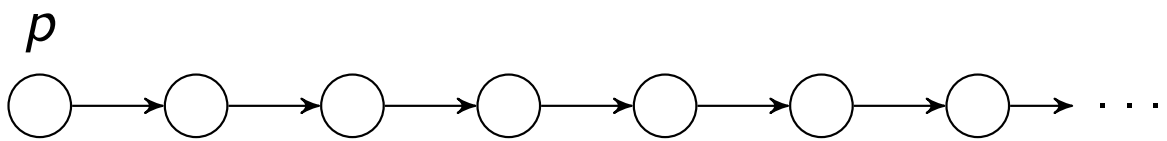
  are LTL formulas
- modalities: if φ and ψ are LTL formulas, then
  - $\mathbf{X}\varphi$ ("next φ")
  - $\varphi \mathbf{U} \psi$ ("φ until ψ")

  are LTL formulas

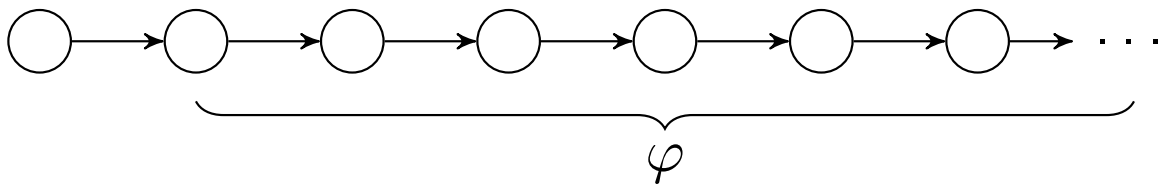
Precedence: unary operators bind stronger than binary ones

Intuitive Semantics

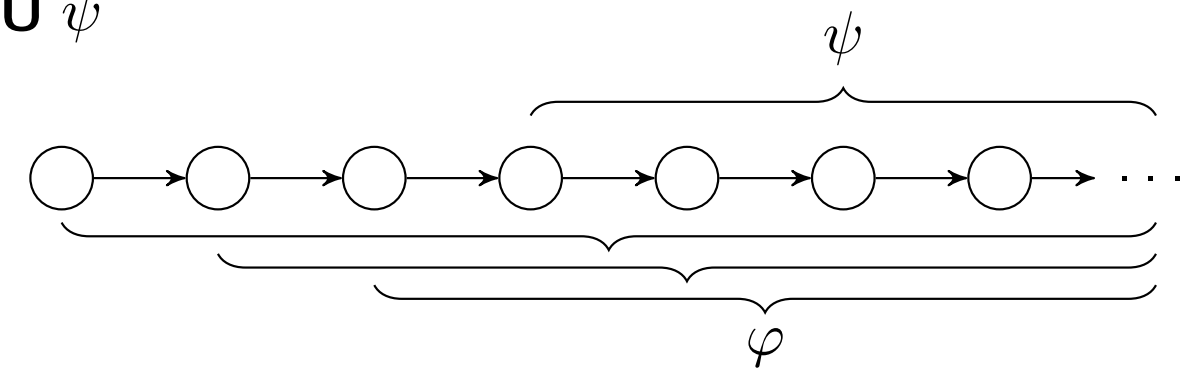
Atomic proposition: p



Next state: $\mathbf{X}\varphi$



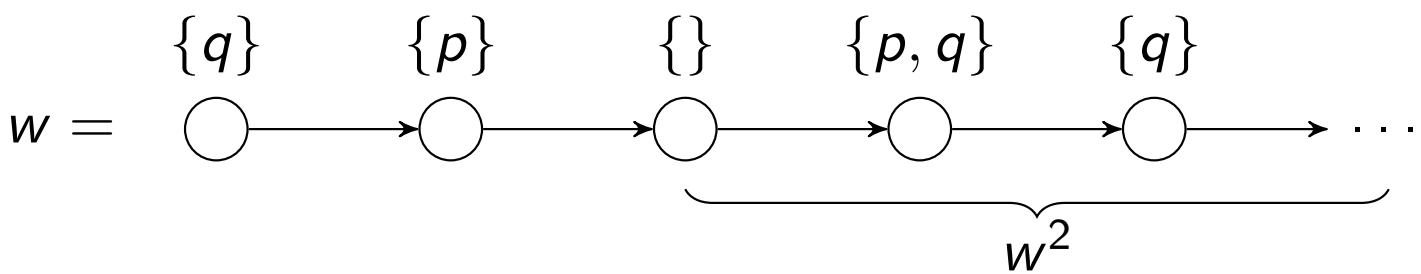
Until: $\varphi \mathbf{U} \psi$



Interpretations

- LTL interpretations = infinite words over the alphabet 2^P (infinite sequences of interpretations of the propositional variables)
- one path in a LTS = one interpretation
- let $w = w_0 w_1 \ldots$ be an infinite word. We denote by w^i the word $w_i w_{i+1} \ldots$

Example with $P = \{p, q\}$



Definition of Semantics

The satisfaction relation $\models$ between w and an LTL formula is defined inductively:

$$
\begin{array}{lcl}
w \models p & \equiv & p \in w_0 \\
w \models \mathbf{X}\varphi & \equiv & w^1 \models \varphi \\
w \models \varphi \,\mathbf{U}\, \psi & \equiv & \text{there exists } i \geq 0 \text{ such that:} \\
 & & \bullet\ w^i \models \psi \\
 & & \bullet\ \text{for all } 0 \leq j < i, w^j \models \varphi \\
w \models \neg\varphi & \equiv & w \models \varphi \text{ is not true} \\
w \models \varphi \vee \psi & \equiv & w \models \varphi \text{ or } w \models \psi
\end{array}
$$

Derived Operators

Derived Boolean operators:

$$
\begin{array}{rcl}
\top & \equiv & p \vee \neg p \\
\bot & \equiv & \neg \top \\
\varphi \wedge \psi & \equiv & \neg(\neg\varphi \vee \neg\psi) \\
\varphi \implies \psi & \equiv & \neg\varphi \vee \psi \\
\varphi \iff \psi & \equiv & (\varphi \implies \psi) \wedge (\psi \implies \varphi)
\end{array}
$$

Derived modalities

$$
\begin{array}{rcl}
\mathbf{F}\varphi & \equiv & \top \,\mathbf{U}\, \varphi \\
\mathbf{G}\varphi & \equiv & \neg(\mathbf{F}\neg\varphi) \\
\varphi \,\mathbf{R}\, \psi & \equiv & \neg(\neg\varphi \,\mathbf{U}\, \neg\psi) \\
\varphi \,\mathbf{W}\, \psi & \equiv & (\varphi \,\mathbf{U}\, \psi) \vee \mathbf{G}\varphi \\
\varphi \,\mathbf{M}\, \psi & \equiv & (\varphi \,\mathbf{R}\, \psi) \wedge \mathbf{F}\varphi
\end{array}
$$

Exercise 1

Give a direct description of the semantics of **F**, **G**, **R**, **W**, **M** and justify it.

Exercise 1

Give a direct description of the semantics of **F**, **G**, **R**, **W**, **M** and justify it.

Answer

- $\mathbf{F}\varphi$ ("eventually φ"): φ will be satisfied at some point
- $\mathbf{G}\varphi$ ("globally φ"): φ is always satisfied
- $\varphi\ \mathbf{R}\ \psi$ ("φ release ψ"): ψ must remain true up to (including) the point when φ becomes true, which may never occur
- $\varphi\ \mathbf{W}\ \psi$ ("weak until"): same as "until", but the stop condition ψ may never occur
- $\varphi\ \mathbf{M}\ \psi$ ("strong release"): same as "release", but the release condition φ must occur

Validity, Equivalence, Models

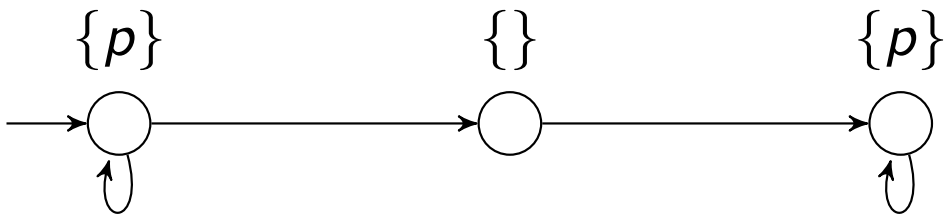
- a formula φ is valid (denoted $\models \varphi$) if for any word w, $w \models \varphi$
- two formulas φ and ψ are equivalent ($\varphi \equiv \psi$) if for any word w, $w \models \varphi \iff w \models \psi$
- a transition system TS satisfies a formula φ ($TS \models \varphi$), if, for any word $w \in L(TS)$, $w \models \varphi$

A Note on Negation

- for a word, by definition

$$w \not\models \varphi \equiv w \models \neg\varphi$$

- the same is not true for a transition system!



This system satisfies neither $\mathbf{G}p$, nor $\neg\mathbf{G}p$

Equivalence Rules

Negation

$$\neg \mathbf{X}\varphi \equiv \mathbf{X}\neg\varphi$$
$$\neg \mathbf{G}\varphi \equiv \mathbf{F}\neg\varphi$$
$$\neg \mathbf{F}\varphi \equiv \mathbf{G}\neg\varphi$$
$$\neg(\varphi \,\mathbf{U}\, \psi) \equiv \neg\varphi \,\mathbf{R}\, \neg\psi$$
$$\neg(\varphi \,\mathbf{R}\, \psi) \equiv \neg\varphi \,\mathbf{U}\, \neg\psi$$
$$\neg(\varphi \,\mathbf{W}\, \psi) \equiv \neg\varphi \,\mathbf{M}\, \neg\psi$$
$$\neg(\varphi \,\mathbf{M}\, \psi) \equiv \neg\varphi \,\mathbf{W}\, \neg\psi$$

Distributivity

$$\mathbf{X}(\varphi \vee \psi) \equiv \mathbf{X}\varphi \vee \mathbf{X}\psi$$
$$\mathbf{X}(\varphi \wedge \psi) \equiv \mathbf{X}\varphi \wedge \mathbf{X}\psi$$
$$\mathbf{X}(\varphi \,\mathbf{U}\, \psi) \equiv \mathbf{X}\varphi \,\mathbf{U}\, \mathbf{X}\psi$$
$$\mathbf{F}(\varphi \vee \psi) \equiv \mathbf{F}\varphi \vee \mathbf{F}\psi$$
$$\mathbf{G}(\varphi \wedge \psi) \equiv \mathbf{G}\varphi \wedge \mathbf{G}\psi$$
$$\rho \,\mathbf{U}\, (\varphi \vee \psi) \equiv (\rho \,\mathbf{U}\, \varphi) \vee (\rho \,\mathbf{U}\, \psi)$$
$$(\varphi \wedge \psi) \,\mathbf{U}\, \rho \equiv (\varphi \,\mathbf{U}\, \rho) \wedge (\psi \,\mathbf{U}\, \rho)$$

Idempotency

$$\mathbf{G}\varphi \equiv \mathbf{G}\mathbf{G}\varphi$$
$$\mathbf{F}\varphi \equiv \mathbf{F}\mathbf{F}\varphi$$
$$\varphi \,\mathbf{U}\, \psi \equiv \varphi \,\mathbf{U}\, (\varphi \,\mathbf{U}\, \psi)$$

Unrolling

$$\mathbf{G}\varphi \equiv \varphi \wedge \mathbf{X}\mathbf{G}\varphi$$
$$\mathbf{F}\varphi \equiv \varphi \vee \mathbf{X}\mathbf{F}\varphi$$
$$\varphi \,\mathbf{U}\, \psi \equiv \psi \vee (\varphi \wedge \mathbf{X}(\varphi \,\mathbf{U}\, \psi))$$

Exercise 2

Show that the following formulas are not equivalent:

1. $\mathbf{G}(p \vee q) \not\equiv \mathbf{G}p \vee \mathbf{G}q$
2. $\mathbf{F}(p \wedge q) \not\equiv \mathbf{F}p \wedge \mathbf{F}q$
3. $p \,\mathbf{U}\, (q \,\mathbf{U}\, r) \not\equiv (p \,\mathbf{U}\, q) \,\mathbf{U}\, r$

Negation Normal Form

A LTL formula is in negation normal form if:

1. negations appear only in front of propositional variables
2. $\top$, $\bot$, $\wedge$ and $\vee$ are the only Boolean operators allowed
3. $\mathbf{X}$, $\mathbf{U}$ and $\mathbf{R}$ are the only operators allowed

Examples

$\neg \mathbf{X} p$	✗	$\mathbf{X} \neg p$	✓
$p \Longrightarrow \mathbf{X} q$	✗	$\neg p \vee \mathbf{X} a$	✓
$\mathbf{F} p$	✗	$\top \mathbf{U} p$	✓

Exercise 3

Put the following formulas in NNF:

1. $\mathbf{G}p$
2. $\neg\mathbf{F}\mathbf{G}p$
3. $\neg\mathbf{G}(p \implies \mathbf{F}q)$

Formalizing Properties of Systems: Examples

- safety (mutual exclusion invariant):

$$\mathbf{G}\neg(\mathit{crit}_1 \wedge \mathit{crit}_2)$$

- a safety property that is not an invariant

$$\neg \mathit{access}\ \mathbf{W}\ \mathit{authenticated}$$

- liveness (request):

$$\mathbf{G}(\mathit{request} \implies \mathbf{F}\mathit{grant})$$

Liveness vs Safety

- liveness properties describe what the program should do
- often easier to formulate than safety properties
- harder to check (counter-examples are infinite)
- for a given liveness property, there is often a corresponding safety property

Exercise 4

1. Write LTL formulas corresponding to the following properties:
   - if p becomes true at any point, then q must be true until r becomes true
   - p can only be true if q has been true at least once before
   - p is true exactly every other state
2. Consider a traffic light. Formalize the property that the light must cycle through colors in the usual order.

Section 3

LTL Model-Checking

LTL and ω-Languages

- a LTL formula defines an ω-language: the set of words that satisfy it
- a labelled transition system defines an ω-language: the set of words (paths) through it

Verifying that a system TS satisfies a formula φ is the same thing as checking

$$\mathcal{L}(TS) \subseteq \mathcal{L}(\varphi)$$

We will do this by converting both TS and φ to Büchi automata.

Büchi Automata: Reminders

Definition

- transition system
- no labels on states, but actions (labels on transitions) instead
- set of accepting states

Acceptance condition

- accepted run = at least one accepting state occurs infinitely often

Determinism

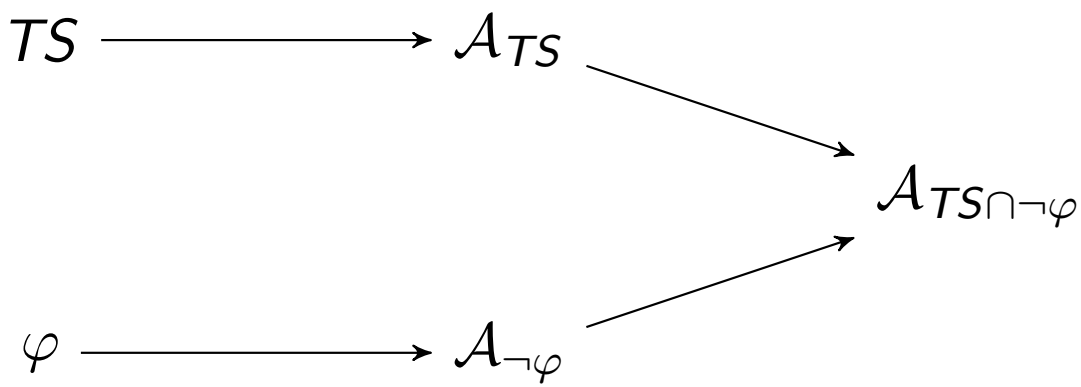
- unlike finite automata, not all non-deterministic Büchi automata (NDA) have an equivalent deterministic Büchi automaton (DBA)

Why Use an Intermediate Representation?

Advantages

- NBA can be used to represent the language of
  - any LTL formula
  - any labelled transition system
- NBA are closed under intersection (also: union, complement, ω-closure, concatenation)
- checking that the language of a BA is empty is easy

Automata-Based Verification

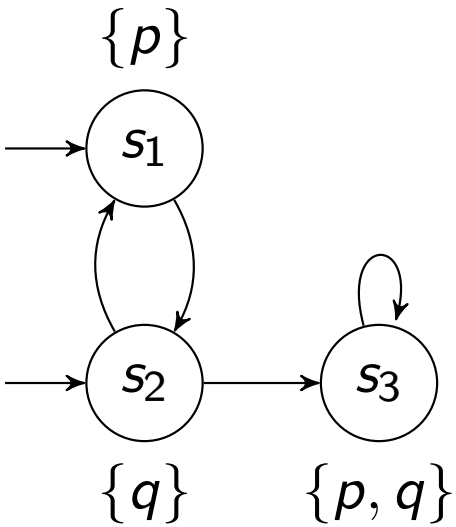


- any word accepted by $\mathcal{A}_{TS \cap \neg\varphi}$ is both in $\mathcal{L}(TS)$ and $\mathcal{L}(\neg\varphi)$
- it corresponds to a run of TS that violates φ
- otherwise, if the language of $\mathcal{A}_{TS \cap \neg\varphi}$ is empty, $TS \models \varphi$

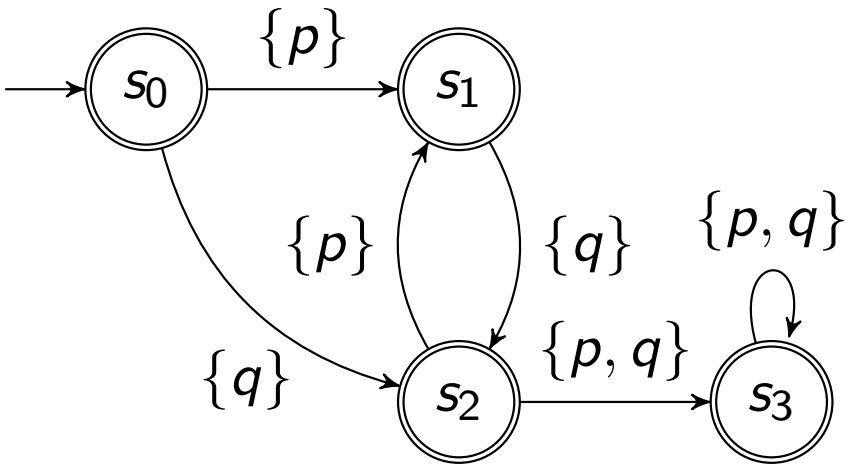
From a LTS to a Büchi Automaton

- given a LTS $TS = (S, \rightarrow, S_0, L : S \rightarrow 2^P)$
- the corresponding BA is $A = (S \cup \{s_0\}, 2^P, \delta, \{s_0\}, S \cup \{s_0\})$
  - $(s_0, \alpha, s) \in \delta$ iff $s \in S_0$ and $\alpha = L(s)$
  - $(s, \alpha, s') \in \delta$ iff $s \rightarrow s'$ and $\alpha = L(s')$

LTS



Büchi automaton



Closure of a Formula

$$cl(\varphi) = \{\psi \mid \psi \text{ is a sub-formula of } \varphi \text{ or its negation}\}$$

Example

$$cl(p \,\mathbf{U}\, \neg q) = \{p \,\mathbf{U}\, \neg q, \neg p \,\mathbf{R}\, q, p, \neg p, q, \neg q\}$$

note: we put all subformulas in NNF for convenience later

Maximally Consistent Subset

Let us consider the subsets $S \subset cl(\varphi)$ that are *maximally consistent*: the largest combinations of subformulas that can be true at the same time.

Enumeration

- consider all the subsets $S \subset cl(\varphi)$ that contain exactly one of ψ or $\neg\psi$ for each subformula ψ
- keep only those that are consistent:
  - $\bot$ must not be in S
  - if $\varphi \wedge \psi \in S$, then φ and ψ must be in S
  - if $\varphi \vee \psi \in S$, then φ must be in S or ψ must be in S
  - if $\varphi \mathbf{U} \psi \in S$, then φ must be in S or ψ must be in S
  - if $\varphi \mathbf{R} \psi \in S$, then ψ must be in S

The set of maximally consistent subsets is denoted $mcs(\varphi)$

Maximally Consistent Subsets: Example

- we have $cl(p \mathbf{U} \neg q) = \{p \mathbf{U} \neg q, \neg p \mathbf{R} q, p, \neg p, q, \neg q\}$
- let us enumerate the maximally consistent subsets

subset	consistent?
$\{p \mathbf{U} \neg q, p, q\}$	✓
$\{p \mathbf{U} \neg q, p, \neg q\}$	✓
$\{p \mathbf{U} \neg q, \neg p, q\}$	✗
$\{p \mathbf{U} \neg q, \neg p, \neg q\}$	✓
$\{\neg p \mathbf{R} q, p, q\}$	✓
$\{\neg p \mathbf{R} q, p, \neg q\}$	✗
$\{\neg p \mathbf{R} q, \neg p, q\}$	✓
$\{\neg p \mathbf{R} q, \neg p, \neg q\}$	✗

Exercise 5

- list the elements of $mcs(\neg \mathbf{FG} p)$
- reminder: $NNF(\neg \mathbf{FG} p) = \bot \, \mathbf{R} \, (\top \, \mathbf{U} \, \neg p)$

Sequences of Maximally Consistent Sets

Idea: for any word w such that $w \models \varphi$ we can construct a unique sequence H of elements of $mcs(\varphi)$ such that for all i, w^i satisfies the formulas in H_i and no other formulas in $cl(\varphi)$

Example with $\varphi = p \mathbf{U} \neg q$

$$\begin{array}{lccccccc} w = & \{p,q\} & \{p,q\} & \{p\} & \{q\} & \{q\} & \dots \\ H = & \{\varphi,p,q\} & \{\varphi,p,q\} & \{\varphi,p,\neg q\} & \{\neg\varphi,\neg p,q\} & \{\neg\varphi,\neg p,q\} & \dots \end{array}$$

Corollaries

- for any i, $w_i = H_i \cap P$
- $\varphi \in H_0$

Constructing the Büchi Automaton

Idea:

- build BA with states $Q = mcs(\varphi)$
- infinite paths starting in q_i should be accepted iff they satisfy exactly the formulas in q_i
- initial states q such that $\varphi \in q$
- words accepted by the BA are those that satisfy φ

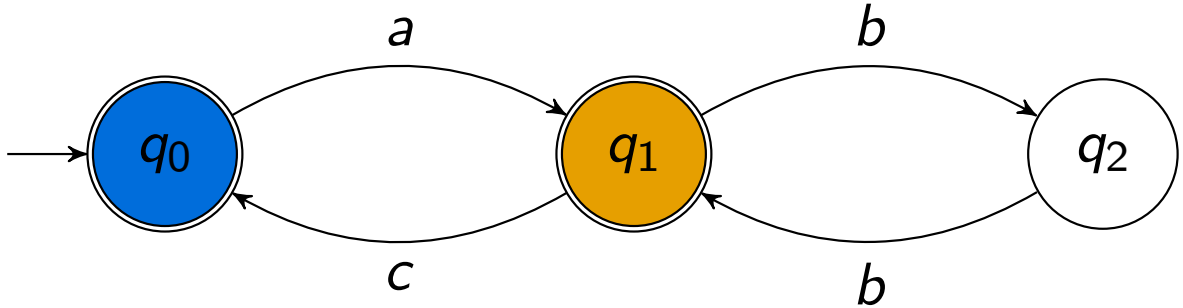
Difficulty:

- the acceptance condition for $\varphi \mathbf{U} \psi$ is not local

Generalized Büchi Automata

- difference from Büchi automata: acceptance condition
  - the acceptance condition $\mathcal{F}$ is a set of sets of states
  - a run is accepting iff the set of infinitely often occurring states contains at least a state from each accepting set $F_i \in \mathcal{F}$
  - if $\mathcal{F}$ is empty, any run is accepted
- same expressive power: every GBA has a corresponding BA (trivially, the other way around too)
- mostly used as an intermediate step during the translation from LTL

Generalized Büchi Automata: Example



- $F = \Big\{\{q_0\}, \{q_1\}\Big\}$
- the word $(abbc)^\omega$ is accepted
- the word $a(bb)^\omega$ is not, because q_0 is not visited infinitely often

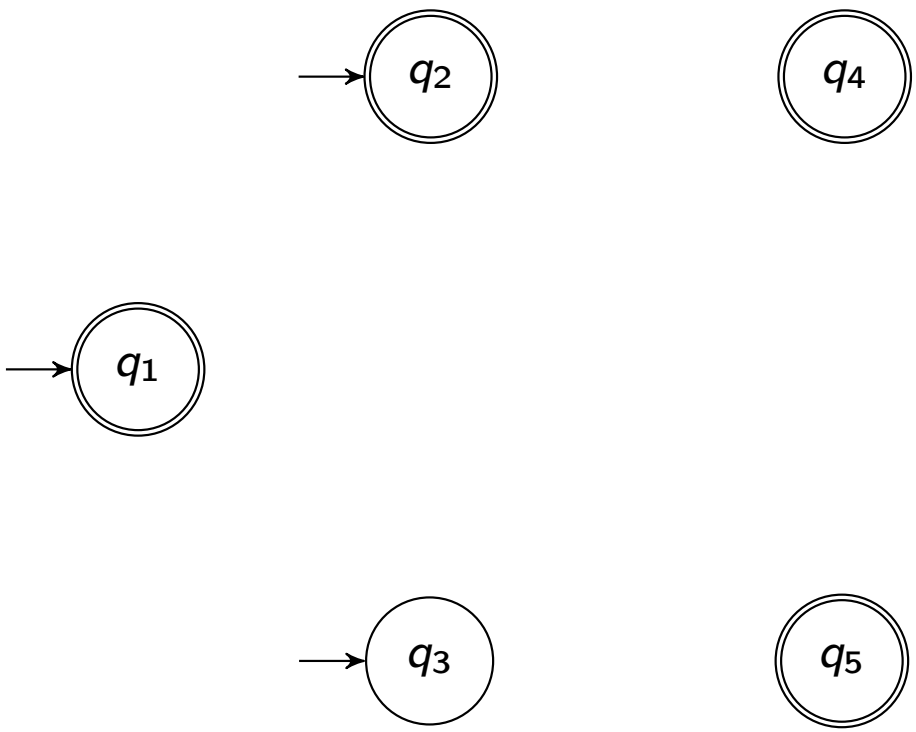
LTL to GBA

For a formula φ, let $\mathcal{A}_\varphi = (Q, 2^P, \delta, Q_0, F)$ with

- $Q = mcs(\varphi)$
- $Q_0 = \{q \mid \varphi \in q\}$
- $(q, \alpha, q') \in \delta$ iff
  - $\alpha = q \cap P$
  - if $\mathbf{X}\psi \in q$, then $\psi \in q'$
  - if $\psi_1 \mathbf{U} \psi_2 \in q$ and $\neg\psi_2 \in q$, then $\psi_1 \mathbf{U} \psi_2 \in q'$
  - if $\psi_1 \mathbf{R} \psi_2 \in q$ and $\neg\psi_1 \in q$, then $\psi_1 \mathbf{R} \psi_2 \in q'$
- $F = \{F_1, \ldots, F_n\}$
- one accepting set $F_i \in F$ for each $\psi_{i1} \mathbf{U} \psi_{i2} \in cl(\varphi)$
- $q \in F_i$ if:
  - $\psi_{i1} \mathbf{U} \psi_{i2} \notin q$
  - OR $\psi_{i2} \in q$

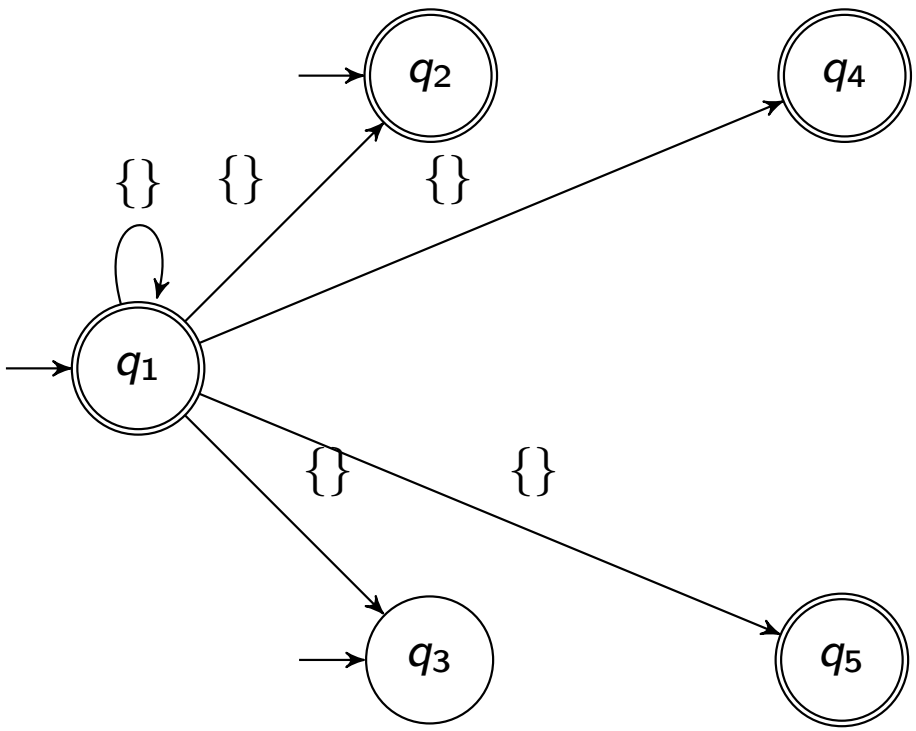
Automaton for $\varphi = p \mathbf{U} \neg q$

$$\underbrace{\{p \mathbf{U} \neg q, \neg p, \neg q\}}_{q_1}, \underbrace{\{p \mathbf{U} \neg q, p, \neg q\}}_{q_2}, \underbrace{\{p \mathbf{U} \neg q, p, q\}}_{q_3}, \underbrace{\{\neg p \mathbf{R} q, p, q\}}_{q_4}, \underbrace{\{\neg p \mathbf{R} q, \neg p, q\}}_{q_5},$$



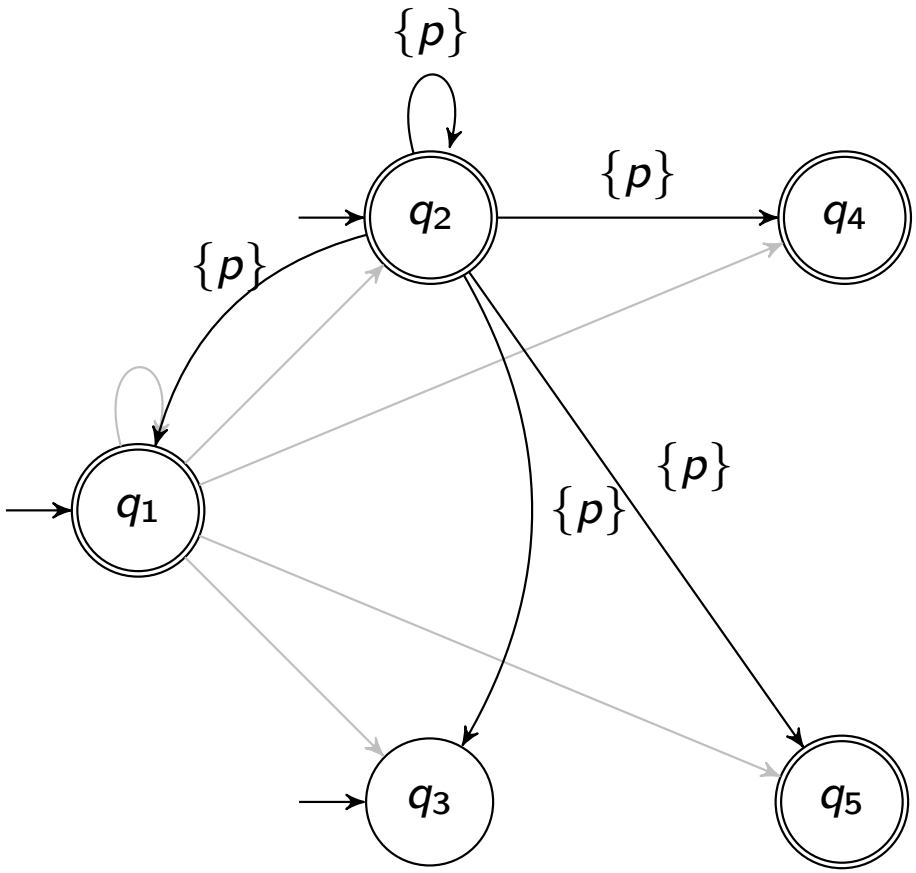
Automaton for $\varphi = p \mathbf{U} \neg q$

$$\underbrace{\{p \mathbf{U} \neg q, \neg p, \neg q\}}_{q_1}, \underbrace{\{p \mathbf{U} \neg q, p, \neg q\}}_{q_2}, \underbrace{\{p \mathbf{U} \neg q, p, q\}}_{q_3}, \underbrace{\{\neg p \mathbf{R} q, p, q\}}_{q_4}, \underbrace{\{\neg p \mathbf{R} q, \neg p, q\}}_{q_5},$$



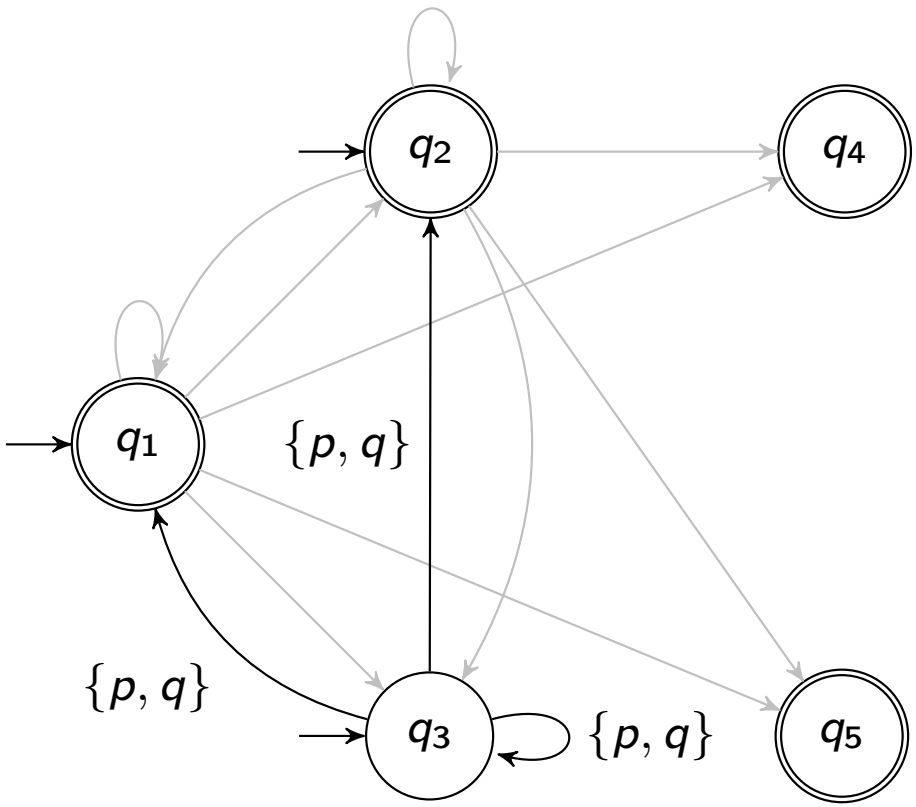
Automaton for $\varphi = p \mathbf{U} \neg q$

$$\underbrace{\{p \mathbf{U} \neg q, \neg p, \neg q\}}_{q_1}, \underbrace{\{p \mathbf{U} \neg q, p, \neg q\}}_{q_2}, \underbrace{\{p \mathbf{U} \neg q, p, q\}}_{q_3}, \underbrace{\{\neg p \mathbf{R} q, p, q\}}_{q_4}, \underbrace{\{\neg p \mathbf{R} q, \neg p, q\}}_{q_5},$$



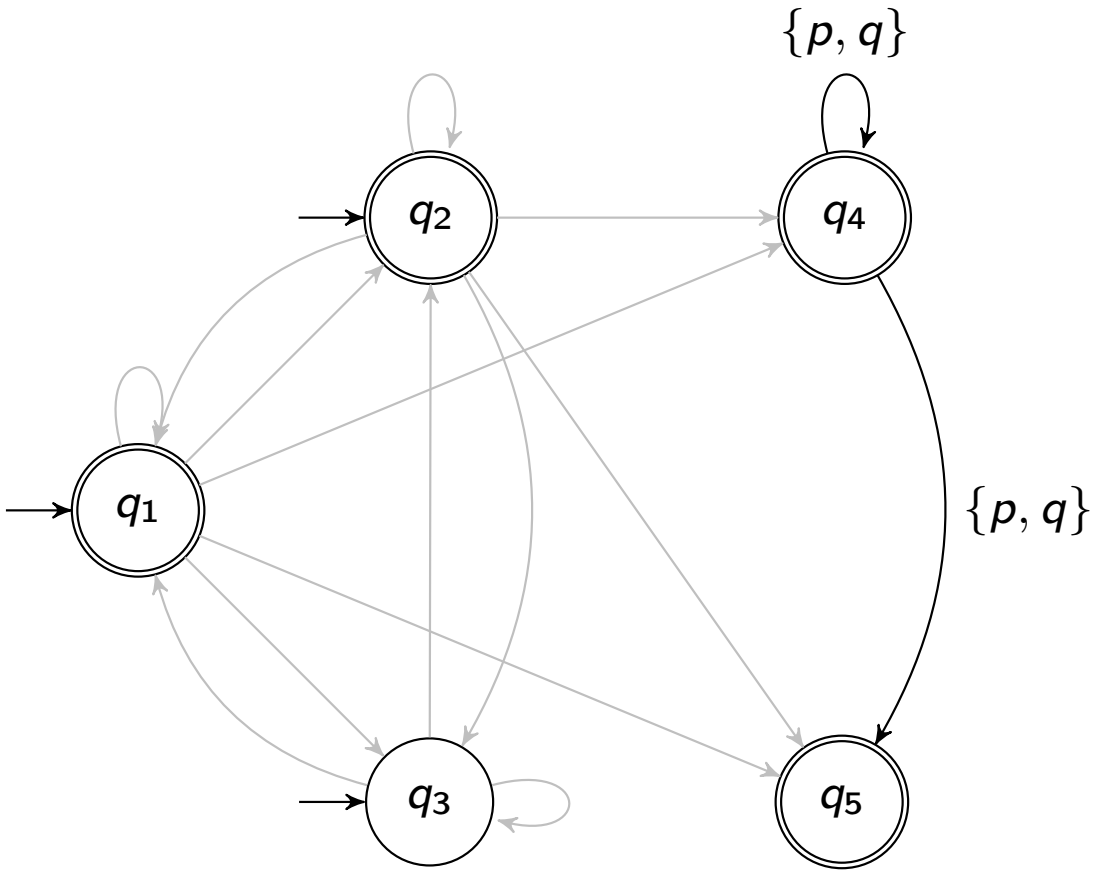
Automaton for $\varphi = p \mathbf{U} \neg q$

$$\underbrace{\{p \mathbf{U} \neg q, \neg p, \neg q\}}_{q_1}, \underbrace{\{p \mathbf{U} \neg q, p, \neg q\}}_{q_2}, \underbrace{\{p \mathbf{U} \neg q, p, q\}}_{q_3}, \underbrace{\{\neg p \mathbf{R} q, p, q\}}_{q_4}, \underbrace{\{\neg p \mathbf{R} q, \neg p, q\}}_{q_5},$$



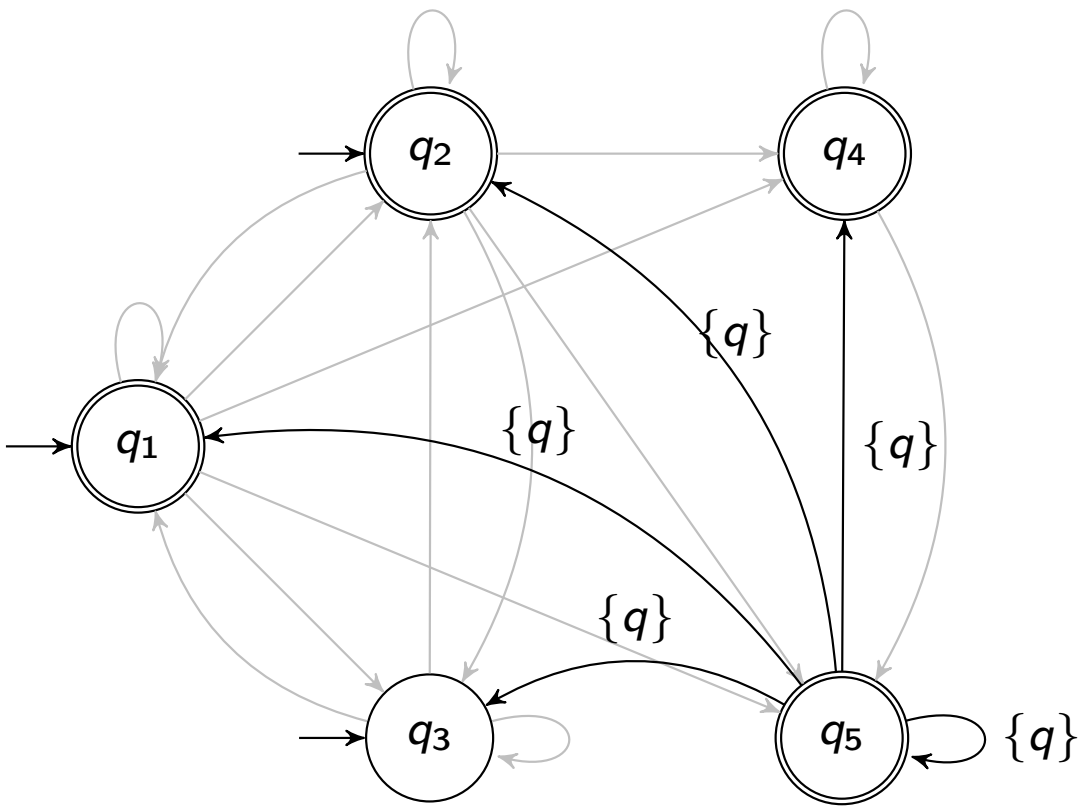
Automaton for $\varphi = p \mathbf{U} \neg q$

$$\underbrace{\{p \mathbf{U} \neg q, \neg p, \neg q\}}_{q_1}, \underbrace{\{p \mathbf{U} \neg q, p, \neg q\}}_{q_2}, \underbrace{\{p \mathbf{U} \neg q, p, q\}}_{q_3}, \underbrace{\{\neg p \mathbf{R} q, p, q\}}_{q_4}, \underbrace{\{\neg p \mathbf{R} q, \neg p, q\}}_{q_5},$$



Automaton for $\varphi = p \mathbf{U} \neg q$

$$\underbrace{\{p \mathbf{U} \neg q, \neg p, \neg q\}}_{q_1}, \underbrace{\{p \mathbf{U} \neg q, p, \neg q\}}_{q_2}, \underbrace{\{p \mathbf{U} \neg q, p, q\}}_{q_3}, \underbrace{\{\neg p \mathbf{R} q, p, q\}}_{q_4}, \underbrace{\{\neg p \mathbf{R} q, \neg p, q\}}_{q_5},$$

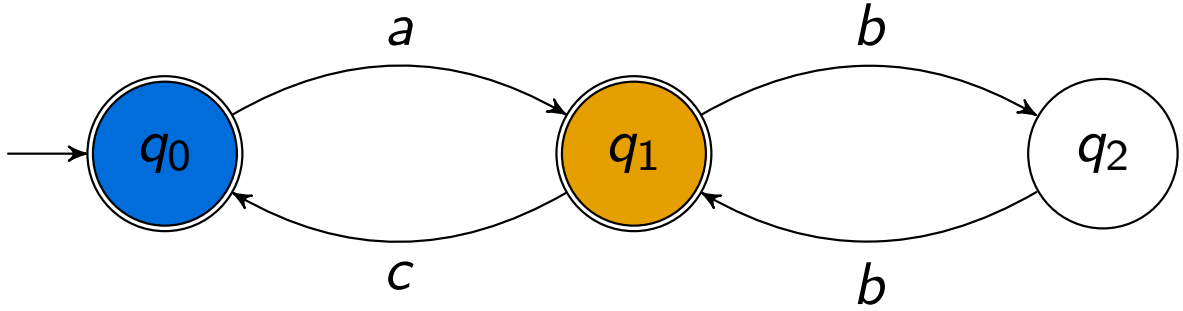


Un-Generalizing the Büchi Automaton

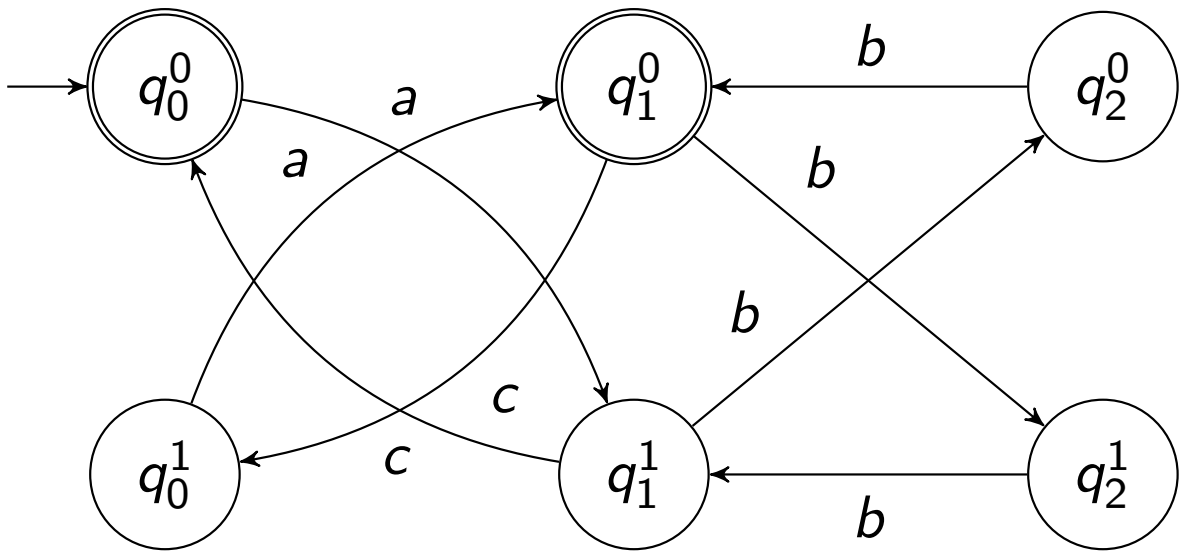
- if $|F| = 0$, simply mark all states as accepting
- else, when $F = \{F_0, \ldots, F_{k-1}\}$, let $\mathcal{A}' = (Q', 2^P, \delta', Q'_0, F')$ with
  - $Q' = Q \times \{0, \ldots, k-1\}$
  - $Q'_o = \{\langle q, 0\rangle \mid q \in Q_0\}$
  - $F' = \{\langle q, 0\rangle \mid q \in F_0\}$
  - for any $(q, \alpha, q') \in \delta$
    - if $q \notin F_i$, $(\langle q, i\rangle, \alpha, \langle q', i\rangle) \in \delta'$
    - otherwise, $(\langle q, i\rangle, \alpha, \langle q', i+1 \mod k\rangle) \in \delta'$
- idea: create k "copies" of the automaton
  - transitions out of a state in F_i lead to the copy $i+1$
  - a path looping through an accepting state $\langle q, 0\rangle$ must necessarily go through accepting states in each of the copies

Un-Generalizing the Büchi Automaton: Example

Generalized automaton

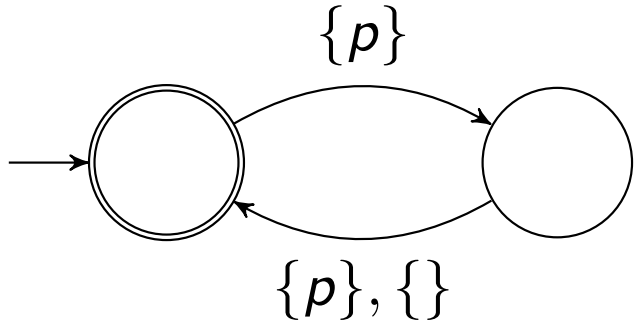


Equivalent standard automaton



Expressivity Büchi Automata vs LTL

Büchi automata are strictly more expressive than LTL



- no LTL formula (over $P = \{p\}$) is equivalent to this BA
- Büchi automata can define ω-regular languages (similar to regular languages, but for infinite words)

Intersection Automaton: the Special Case

- given two Büchi automata over alphabet Σ
  - $\mathcal{A} = (Q_A, \Sigma, \delta_A, Q_{0A}, F_A)$ and $\mathcal{B} = (Q_B, \Sigma, \delta_B, Q_{0B}, F_B)$
- we want an automaton $\mathcal{A}_\cap = (Q', \Sigma, \delta', Q'_0, F')$ that accepts the language $L(\mathcal{A}) \cap L(\mathcal{B})$
- **for our purpose**, all states of $\mathcal{A}$ are accepting ($F_A = Q_A$), so we can simply take
  - $Q' = Q_A \times Q_B$
  - $Q'_0 = Q_{0A} \times Q_{0B}$
  - $F' = \{\langle q, q' \rangle \mid q' \in F_B\}$
  - for any $(q_A, \alpha, q'_A) \in \delta_A$ and any $(q_B, \alpha, q'_B) \in \delta_B$, $(\langle q_A, q_B, \rangle, \alpha, \langle q'_A, q'_B, \rangle) \in \delta'$
- this construction is not correct for arbitrary automata!

Intersection Automaton: the General Case

- in general, the automaton should only accept runs that go infinitely often through states of F_A and F_B
- create two "copies" of the automaton with product states to emulate the acceptance condition (similar to the idea used for the un-generalized automaton)
- $\mathcal{A}_\cap = (Q', \Sigma, \delta', Q'_0, F')$
  - $Q' = Q_A \times Q_B \times \{0, 1\}$
  - $Q'_o = Q_{0A} \times Q_{0B} \times \{0\}$
  - $F' = \{\langle q, q', 0\rangle \mid q' \in F_B\}$
  - for any $(q_A, \alpha, q'_A) \in \delta_A$ and any $(q_B, \alpha, q'_B) \in \delta_B$
    - $(\langle q_A, q_B, 0\rangle, \alpha, \langle q'_A, q'_B, i\rangle) \in \delta'$, with $i = 1$ if $q_A \in F_A$ and $i = 0$ otherwise
    - $(\langle q_A, q_B, 1\rangle, \alpha, \langle q'_A, q'_B, i\rangle) \in \delta'$, with $i = 0$ if $q_B \in F_A$ and $i = 1$ otherwise

Emptiness of Büchi Automaton

To check that a Büchi automaton accepts a word:

1. find a path from an initial state to an accepting state q
2. find a path from q to itself

Simple DFS

Negation vs Complement

- instead of computing the automaton $\mathcal{A}_{\neg\varphi}$
- why not compute the automaton $\mathcal{A}_{\varphi}$ and then its complement?

Negation vs Complement

- instead of computing the automaton $\mathcal{A}_{\neg\varphi}$
- why not compute the automaton $\mathcal{A}_{\varphi}$ and then its complement?
- complementation of a BA is exponential
- whereas $\mathcal{A}_{\neg\varphi}$ has the same size as $\mathcal{A}_{\varphi}$
  - $cl(\varphi) = cl(\neg\varphi)$ hence $mcs(\varphi) = mcs(\neg\varphi)$
  - the two automata have the same states, transitions and acceptance conditions
  - only the initial states are different

Complexity

- size of closure: $|cl(\varphi)| \in \mathcal{O}(|\varphi|)$
- number of m.c. sets: $|mcs(\varphi)| \in \mathcal{O}(2^{|cl(\varphi)|}) = \mathcal{O}(2^{|\varphi|})$
- size of intersection automata for a LTS with n states: $\mathcal{O}(n \times 2^{|\varphi|})$

Optimizations

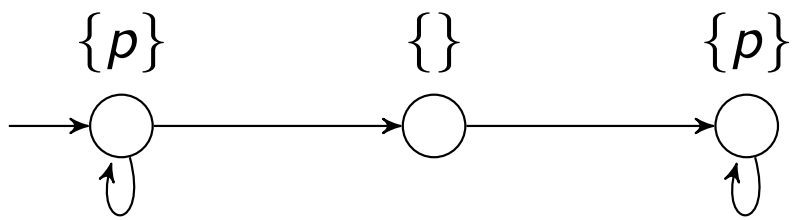
- there are more efficient algorithms to construct $\mathcal{A}_\varphi$
- they remain exponential
- we only care about the emptiness test in the intersection automaton $\implies$ we can construct the automaton lazily (on-the-fly) while we visit it searching for an accepting cycle
- in general, LTL model-checking is PSPACE-complete

Exercise 6

- Compute the generalized Büchi automaton corresponding to the formula

$$\neg \mathbf{FG} p$$

- We want to check that the following transition system satisfies the formula $\mathbf{FG}p$



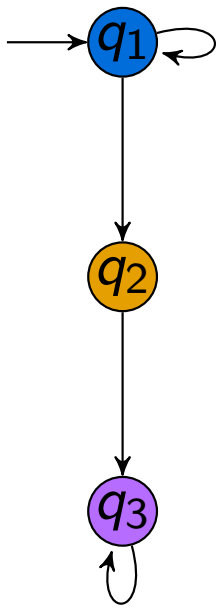
How many states are there in the "intersection" automaton?

Section 4

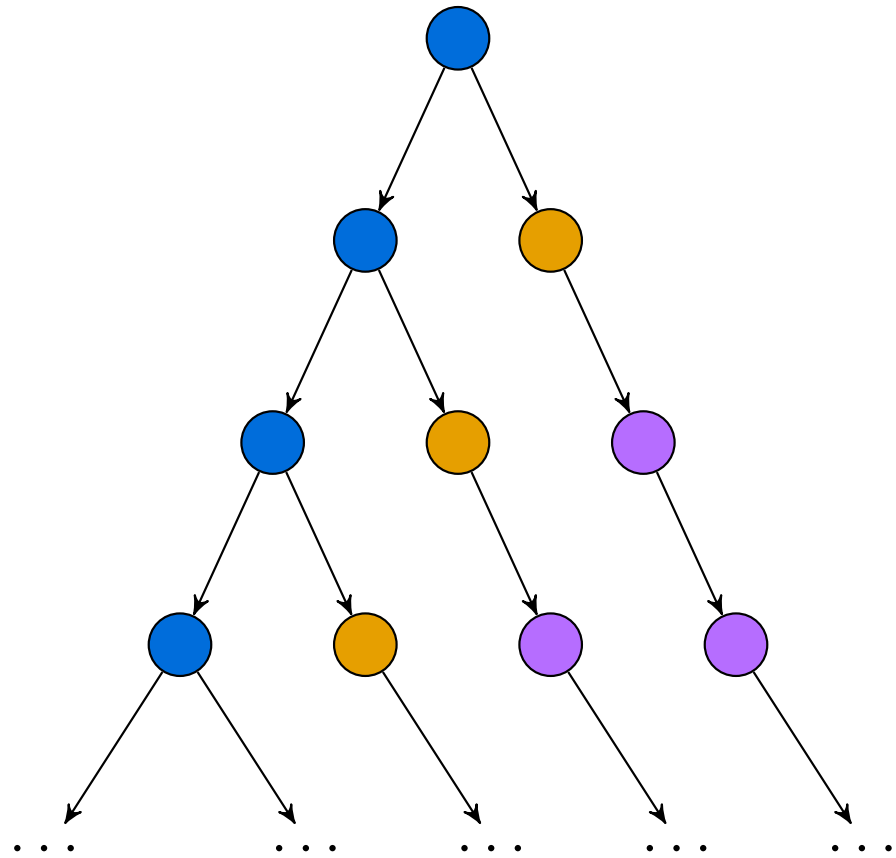
Computational Tree Logic

Computational Trees

Transition System



Unfolding into tree



In some textbooks, you might read that CTL is interpreted over **Kripke structures**: different name for labelled transition systems.

Syntax of CTL

Let P be a set of propositional variables. The syntax of CTL is defined inductively:

- propositions: if $p \in P$, then p is a CTL formula
- Boolean operators: as usual
- modalities: if φ and ψ are LTL formulas, then
  - $\mathbf{AX}\varphi$
  - $\mathbf{EX}\varphi$
  - $\mathbf{AF}\varphi$
  - $\mathbf{EF}\varphi$
  - $\mathbf{AG}\varphi$
  - $\mathbf{EG}\varphi$
  - $\mathbf{A}[\varphi\ \mathbf{U}\ \psi]$
  - $\mathbf{E}[\varphi\ \mathbf{U}\ \psi]$

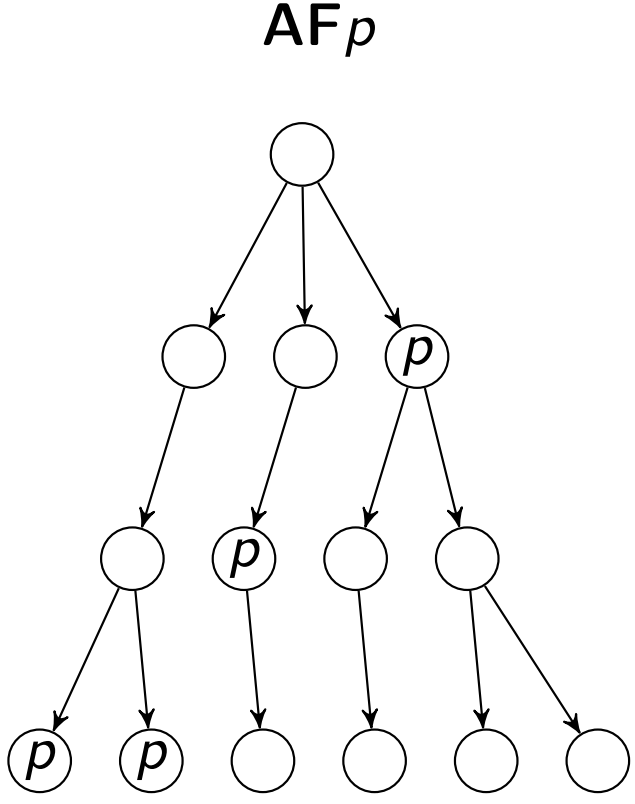
  are LTL formulas

Intuitive Semantics: Path Quantification

- **X**, **F**, **G**, **U** have the same intuitive meaning as in LTL

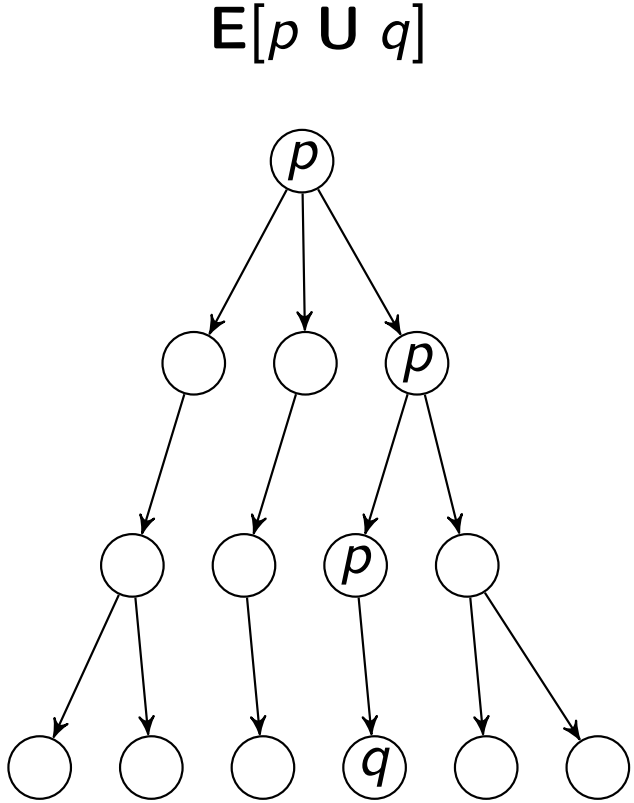
Intuitive Semantics: Path Quantification

- **X**, **F**, **G**, **U** have the same intuitive meaning as in LTL
- **A** = "for all paths"



Intuitive Semantics: Path Quantification

- **X**, **F**, **G**, **U** have the same intuitive meaning as in LTL
- **A** = "for all paths"
- **E** = "there exists a path"



Intuitive Semantics: Path Quantification

- **X**, **F**, **G**, **U** have the same intuitive meaning as in LTL
- **A** = "for all paths"
- **E** = "there exists a path"
- note that **A** and **E** cannot appear at arbitrary positions in formulas

$$\mathbf{AG}(p\ \mathbf{U}\ q)$$

is not a syntactically correct CTL formula!

Definition of Semantics

- let $S = (Q, \rightarrow, Q_0, L)$ be a LTS and $q \in Q$
- S satisfies φ iff $\forall q \in Q_0, \langle S, q \rangle \models \varphi$
- satisfaction relation between a pair $\langle S, q \rangle$ and a CTL formula:

$$\langle S, q \rangle \models p \quad \equiv \quad p \in L(q)$$

Definition of Semantics

- let $S = (Q, \rightarrow, Q_0, L)$ be a LTS and $q \in Q$
- S satisfies φ iff $\forall q \in Q_0, \langle S, q \rangle \models \varphi$
- satisfaction relation between a pair $\langle S, q \rangle$ and a CTL formula:

$$
\begin{array}{lcl}
\langle S, q \rangle \models p & \equiv & p \in L(q) \\
\langle S, q \rangle \models \mathbf{EX}\varphi & \equiv & \exists q' \text{ s.t. } q \rightarrow q' \text{ and } \langle S, q' \rangle \models \varphi \\
\langle S, q_0 \rangle \models \mathbf{EF}\varphi & \equiv & \exists q_n \text{ s.t. } q_0 \rightarrow \cdots \rightarrow q_n \text{ and } \langle S, q_n \rangle \models \varphi \\
\langle S, q_0 \rangle \models \mathbf{EG}\varphi & \equiv & \exists \langle q_0 \rightarrow q_1 \rightarrow \ldots \rangle \text{ s.t. } \forall i, \langle S, q_i \rangle \models \varphi \\
\langle S, q_0 \rangle \models \mathbf{E}[\varphi \, \mathbf{U} \, \psi] & \equiv & \exists q_n \text{ s.t. } q_0 \rightarrow \cdots \rightarrow q_n \text{ and } \langle S, q_n \rangle \models \psi \\
 & & \text{and } \forall i \text{ s.t. } 0 \leq i < n, \langle S, q_i \rangle \models \varphi
\end{array}
$$

Definition of Semantics

- let $S = (Q, \rightarrow, Q_0, L)$ be a LTS and $q \in Q$
- S satisfies φ iff $\forall q \in Q_0, \langle S, q \rangle \models \varphi$
- satisfaction relation between a pair $\langle S, q \rangle$ and a CTL formula:

$$
\begin{array}{lcl}
\langle S, q \rangle \models p & \equiv & p \in L(q) \\
\langle S, q \rangle \models \mathbf{EX}\varphi & \equiv & \exists q' \text{ s.t. } q \rightarrow q' \text{ and } \langle S, q' \rangle \models \varphi \\
\langle S, q_0 \rangle \models \mathbf{EF}\varphi & \equiv & \exists q_n \text{ s.t. } q_0 \rightarrow \cdots \rightarrow q_n \text{ and } \langle S, q_n \rangle \models \varphi \\
\langle S, q_0 \rangle \models \mathbf{EG}\varphi & \equiv & \exists \langle q_0 \rightarrow q_1 \rightarrow \ldots \rangle \text{ s.t. } \forall i, \langle S, q_i \rangle \models \varphi \\
\langle S, q_0 \rangle \models \mathbf{E}[\varphi \ \mathbf{U} \ \psi] & \equiv & \exists q_n \text{ s.t. } q_0 \rightarrow \cdots \rightarrow q_n \text{ and } \langle S, q_n \rangle \models \psi \\
 & & \text{and } \forall i \text{ s.t. } 0 \leq i < n, \langle S, q_i \rangle \models \varphi \\
\langle S, q \rangle \models \mathbf{AX}\varphi & \equiv & \forall q' \text{ s.t. } q \rightarrow q', \langle S, q' \rangle \models \varphi \\
\langle S, q_0 \rangle \models \mathbf{AG}\varphi & \equiv & \forall q_n \text{ s.t. } q_0 \rightarrow \cdots \rightarrow q_n, \langle S, q_n \rangle \models \varphi \\
\langle S, q_0 \rangle \models \mathbf{AF}\varphi & \equiv & \forall \langle q_0 \rightarrow q_1 \rightarrow \ldots \rangle, \exists i \text{ s.t. } \langle S, q_i \rangle \models \varphi \\
\langle S, q_0 \rangle \models \mathbf{A}[\varphi \ \mathbf{U} \ \psi] & \equiv & \forall \langle q_0 \rightarrow q_1 \rightarrow \ldots \rangle, \exists j \text{ s.t. } \langle S, q_j \rangle \models \psi \\
 & & \text{and } \forall i \text{ s.t. } 0 \leq i < j, \langle S, q_i \rangle \models \varphi
\end{array}
$$

Exercise 7

- we want to show that the operators $\top$, $\vee$, $\neg$, **EX**, **EG** and **EU** are sufficient to express any CTL formula
- find equivalences to remove operators **EF**, **AX**, **AG**, **AF**, **AU**.

Exercise 7

- we want to show that the operators $\top$, $\vee$, $\neg$, **EX**, **EG** and **EU** are sufficient to express any CTL formula
- find equivalences to remove operators **EF**, **AX**, **AG**, **AF**, **AU**.

Solution

$$
\begin{aligned}
\mathbf{EF}\varphi &\equiv \mathbf{E}[\top\ \mathbf{U}\ \varphi] \\
\mathbf{AX}\varphi &\equiv \neg\mathbf{EX}\neg\varphi \\
\mathbf{AG}\varphi &\equiv \neg\mathbf{EF}\neg\varphi \\
\mathbf{AF}\varphi &\equiv \neg\mathbf{EG}\neg\varphi \\
\mathbf{AF}[\varphi\ \mathbf{U}\ \psi] &\equiv \mathbf{AF}\psi \wedge \neg\mathbf{E}[\neg\psi\ \mathbf{U}\ (\neg\varphi \wedge \neg\psi)] \\
&\equiv \neg(\mathbf{EG}\neg\psi \vee \mathbf{E}[\neg\psi\ \mathbf{U}\ \neg(\varphi \vee \psi)])
\end{aligned}
$$

A formula that uses only those operators is said to be in **existential normal form**.

Satisfaction Set

- given $S = (Q, \to, Q_0, L)$, the *satisfaction set* of a formula φ is

$$[\![\varphi]\!] = \{q \in Q \mid \langle S, q\rangle \models \varphi\}$$

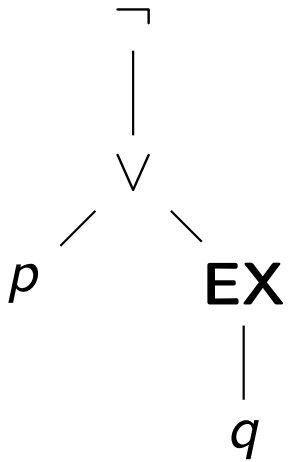
- recursive definition
  - $[\![\top]\!] = Q$
  - $[\![p]\!] = \{q \in Q \mid p \in L(q)\}$
  - $[\![\varphi \vee \psi]\!] = [\![\varphi]\!] \cup [\![\psi]\!]$
  - $[\![\neg\varphi]\!] = Q \setminus [\![\varphi]\!]$
  - $[\![\mathbf{EX}\varphi]\!] = \{q \in Q \mid \exists q' \in [\![\varphi]\!], q \to q'\}$
- easy to compute since Q is finite
- not so obvious for **EG** and **EU**: infinite number of paths!
  - let's ignore those two operators for a moment

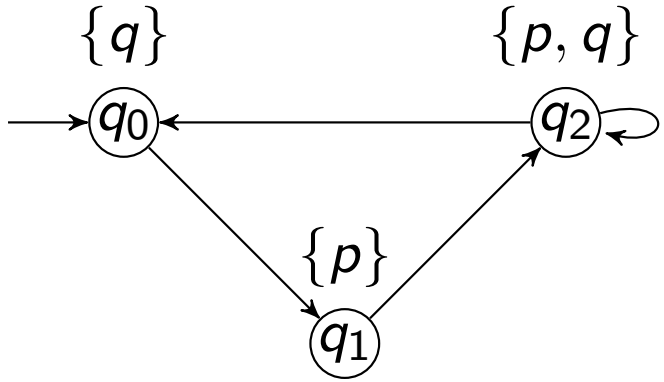
CTL Model Checking

1. put formula φ in $\exists$-normal form
2. traverse the syntactic tree of the formula in a bottom-up fashion (starting with atomic Boolean propositions and finishing with the whole formula)
3. during that traversal, compute the satisfaction set of each subformula
4. check whether $Q_0 \subseteq [\![\varphi]\!]$

CTL Model Checking: Example

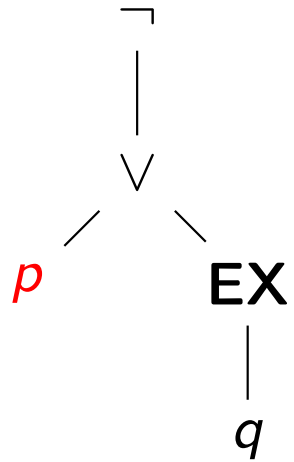
$$\varphi = \neg(p \vee \mathbf{EX} q)$$

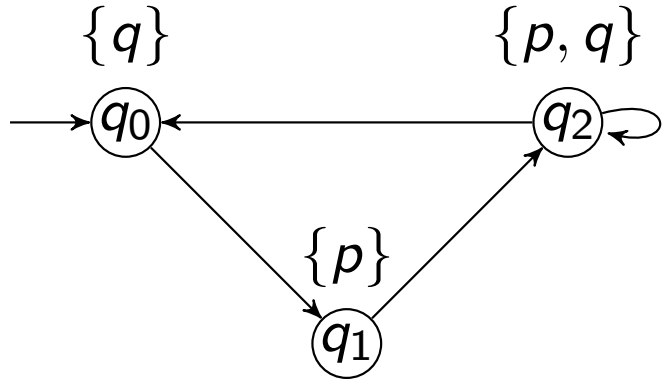




CTL Model Checking: Example

$$\varphi = \neg(p \vee \mathbf{EX} q)$$

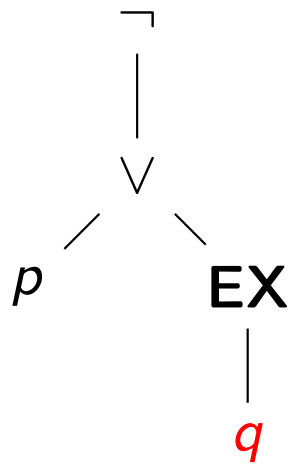


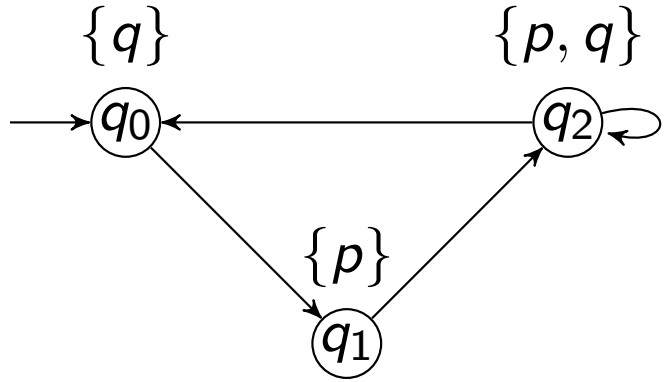


$$[\![p]\!] = \{q_1, q_2\}$$

CTL Model Checking: Example

$$\varphi = \neg(p \vee \mathbf{EX} q)$$

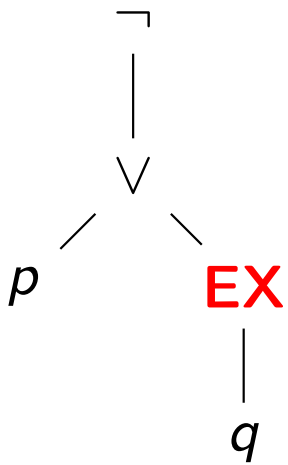


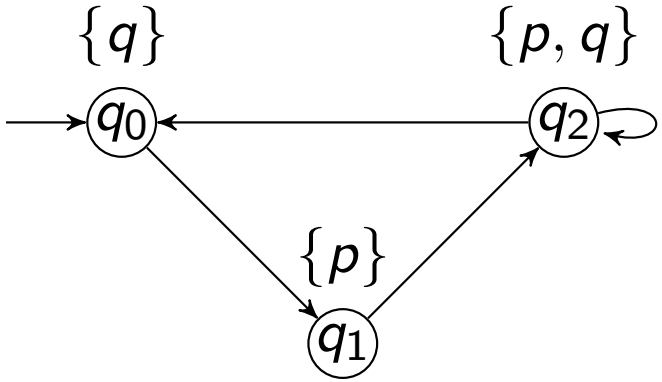


$$[\![p]\!] = \{q_1, q_2\}$$
$$[\![q]\!] = \{q_0, q_2\}$$

CTL Model Checking: Example

$$\varphi = \neg(p \vee \mathbf{EX} q)$$

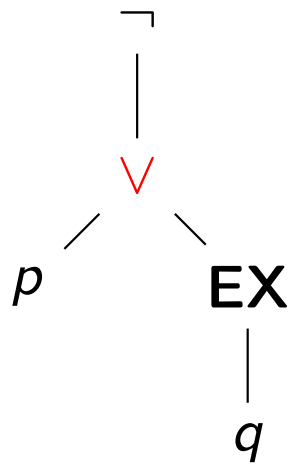


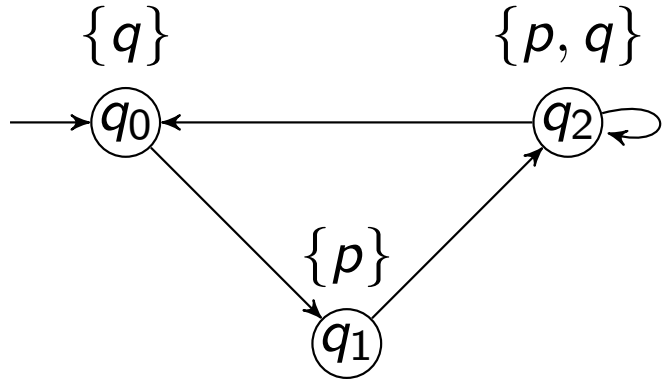


$$\begin{aligned} [\![p]\!] &= \{q_1, q_2\} \\ [\![q]\!] &= \{q_0, q_2\} \\ [\![\mathbf{EX} q]\!] &= \{q_1, q_2\} \end{aligned}$$

CTL Model Checking: Example

$$\varphi = \neg(p \vee \mathbf{EX} q)$$

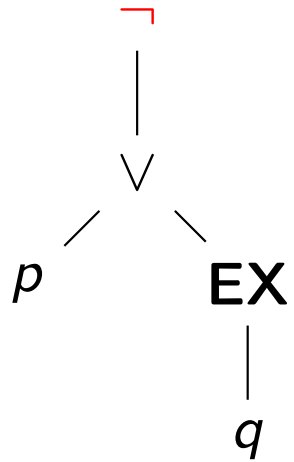


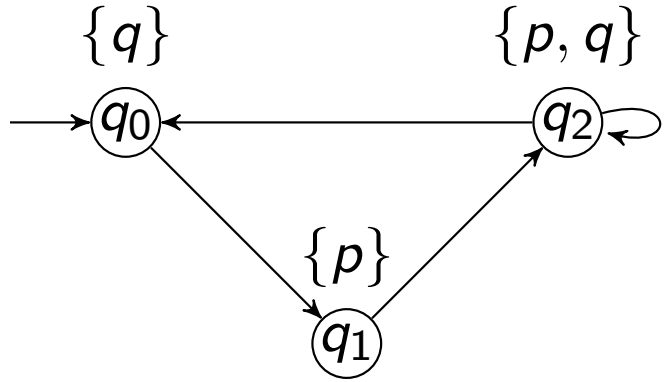


$$\begin{aligned} [\![p]\!] &= \{q_1, q_2\} \\ [\![q]\!] &= \{q_0, q_2\} \\ [\![\mathbf{EX} q]\!] &= \{q_1, q_2\} \\ [\![p \vee \mathbf{EX} q]\!] &= \{q_1, q_2\} \end{aligned}$$

CTL Model Checking: Example

$$\varphi = \neg(p \vee \mathbf{EX} q)$$

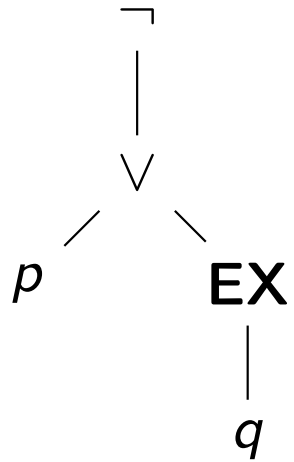


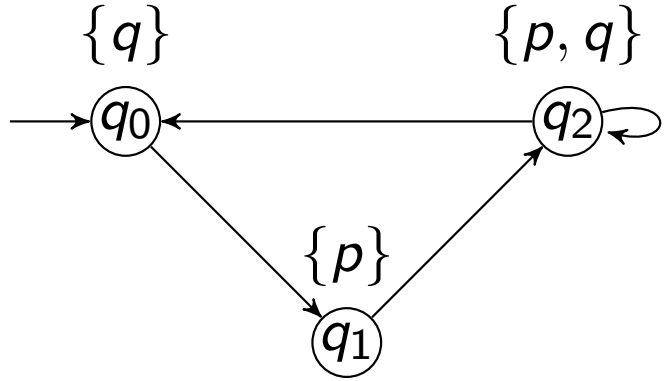


$$\begin{aligned}
[\![p]\!] &= \{q_1, q_2\} \\
[\![q]\!] &= \{q_0, q_2\} \\
[\![\mathbf{EX} q]\!] &= \{q_1, q_2\} \\
[\![p \vee \mathbf{EX} q]\!] &= \{q_1, q_2\} \\
[\![\neg(p \vee \mathbf{EX} q)]\!] &= \{q_0\}
\end{aligned}$$

CTL Model Checking: Example

$$\varphi = \neg(p \vee \mathbf{EX} q)$$





$$\begin{aligned}
[\![p]\!] &= \{q_1, q_2\} \\
[\![q]\!] &= \{q_0, q_2\} \\
[\![\mathbf{EX} q]\!] &= \{q_1, q_2\} \\
[\![p \vee \mathbf{EX} q]\!] &= \{q_1, q_2\} \\
[\![\neg(p \vee \mathbf{EX} q)]\!] &= \{q_0\}
\end{aligned}$$

$$Q_0 = \{q_0\} \subseteq [\![\varphi]\!] \checkmark$$

Satisfaction Set Computation for $\mathbf{EG}\varphi$

Let's try to compute $[\![\mathbf{EG}\varphi]\!]$

- idea: check if φ holds for a finite number of steps
- $[\![\mathbf{EG}\varphi]\!]_i$ to denote states from which there exists a path on which the i first states satisfy φ
- recursive definition
  - $[\![\mathbf{EG}\varphi]\!]_0 = [\![\varphi]\!]$
  - $[\![\mathbf{EG}\varphi]\!]_{i+1} = \{q \in [\![\varphi]\!] \mid \exists q' \text{ s.t. } q \rightarrow q' \text{ and } q' \in [\![\mathbf{EG}\varphi]\!]_i\}$

Satisfaction Set Computation for $\mathbf{EG}\varphi$

Let's try to compute $[\![\mathbf{EG}\varphi]\!]$

- idea: check if φ holds for a finite number of steps
- $[\![\mathbf{EG}\varphi]\!]_i$ to denote states from which there exists a path on which the i first states satisfy φ
- recursive definition
  - $[\![\mathbf{EG}\varphi]\!]_0 = [\![\varphi]\!]$
  - $[\![\mathbf{EG}\varphi]\!]_{i+1} = \{q \in [\![\varphi]\!] \mid \exists q' \text{ s.t. } q \to q' \text{ and } q' \in [\![\mathbf{EG}\varphi]\!]_i\}$
- our LTS has a finite number of states! $(|Q| = n)$
  - after exploring at depth n, there are no more possibilities: $[\![\mathbf{EG}\varphi]\!] \subseteq [\![\mathbf{EG}\varphi]\!]_n$
  - a path of length n among n states must have a loop: $[\![\mathbf{EG}\varphi]\!]_n \subseteq [\![\mathbf{EG}\varphi]\!]$
- $[\![\mathbf{EG}\varphi]\!]_n = [\![\mathbf{EG}\varphi]\!]$ so we can compute the satisfying set with a finite number of steps

Satisfaction Set Computation for $\mathbf{E}[\varphi\ \mathbf{U}\ \psi]$

Same idea

- $[\![\mathbf{E}[\varphi\ \mathbf{U}\ \psi]]\!]_0 = [\![\psi]\!]$
- $[\![\mathbf{E}[\varphi\ \mathbf{U}\ \psi]]\!]_{i+1} =$

$$\{q \in [\![\varphi]\!] \mid \exists q' \text{ s.t. } q \to q' \text{ and } q' \in [\![\mathbf{E}[\varphi\ \mathbf{U}\ \psi]]\!]_i\}$$

- for $|Q| = n$

$$[\![\mathbf{E}[\varphi\ \mathbf{U}\ \psi]]\!] = [\![\mathbf{E}[\varphi\ \mathbf{U}\ \psi]]\!]_n \cup [\![\mathbf{E}[\varphi\ \mathbf{U}\ \psi]]\!]_{n-1} \cup \cdots \cup [\![\mathbf{E}[\varphi\ \mathbf{U}\ \psi]]\!]_0$$

Complexity

- computing the satisfaction set of one formula (assuming we already have done it for subformulas)
  - worst case, $[\![\mathbf{EG}\varphi]\!]$ and $[\![\mathbf{E}[\varphi\ \mathbf{U}\ \psi]]\!]$
  - computing one approximation level: $\mathcal{O}(|Q| + | \to |)$
  - $|Q|$ approximation levels needed to reach fixpoint
- overall complexity: $\mathcal{O}(|\varphi| \times |Q| \times (|Q| + | \to |))$
- can be optimized further with graph algorithms
  - strongly connected components
  - backward search

Section 5

LTL vs CTL: Comparison & Relation

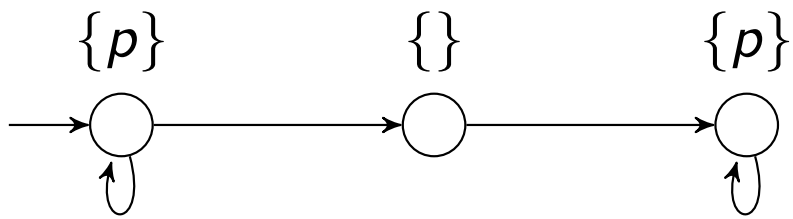
LTL vs CTL

- LTL = statements about (all) paths starting in a state
- CTL = statements about some or all paths starting in a state
- yet some LTL formulas have no equivalent in CTL!

Exercise 8

No CTL formula corresponds to the LTL formula $\mathbf{FG}p$

1. show that $\mathbf{AF}(\mathbf{EG}p)$ is too weak: find a transition system that satisfies the CTL formula, but such that a path violates the LTL formula
2. show that $\mathbf{AF}(\mathbf{AG}p)$ is too strong: prove that the transition system below does not satisfy the CTL formula (previously, we have shown that it satisfies the LTL formula)



Fairness

- some algorithms are correct only under fairness assumptions
- example: Peterson's algorithm for mutual exclusion
  - an unfair scheduler might select process 1 indefinitely right before process 0 executes line 6

Process 0

```
1 flag[0]:= true ;
2 turn := 1 ;
3 while flag[1] & turn == 1 do
4 end
5 critical section ;
6 flag[0]:= false;
```

Process 1

```
1 flag[1]:= true ;
2 turn := 0 ;
3 while flag[0] & turn == 0 do
4 end
5 critical section ;
6 flag[1]:= false;
```

Unfair runs = artifacts of the model that should be ignored?

Fairness: LTL vs CTL

In LTL

- fairness assumptions can be encoded in formulas

$$\neg \mathbf{FG} \mathit{Proc}_1 \implies \varphi$$

In CTL

- no equivalent to $\mathbf{FG}\psi$
- unfair runs cannot be excluded
- requires a special semantics
  - fairness must be encoded in the LTS: set of accepting states
  - a path is fair if it visits **all** accepting states infinitely often
  - **A** and **E** quantifiers apply only to fair paths
- requires adapted algorithm to model-check fair CTL

State Explosion Problem

$$18,446,744,073,709,551,616$$

State Explosion Problem

$$18,446,744,073,709,551,616$$

- the number of distinct states of *one* 64-bit word
- number of states is exponential in size of system
- for distributed systems, parallel composition is another source of exponential growth (number of interleavings)

Model-checking needs to cope with this. Some solutions:

1. **represent** states efficiently
2. **ignore** unreachable/irrelevant parts of the model
3. **abstract** states and use symbolic operations

Other Temporal Logics

- CTL*
  - includes both CTL and LTL
  - path quantification (**A** and **E**) can appear anywhere
  - model-checking is PSPACE-complete (not worse than LTL!)
- μ-calculus
  - logic with fixpoints (recursive definitions)
  - more expressive than CTL*
  - again, model-checking is PSPACE-complete
- Interval temporal logics
  - various versions
  - typically, linear time only
  - time points may be special cases (intervals $[x, x]$) or the logic may be restricted to non-empty intervals