Reactive Programs

Example 1: vending machine

Takes 3 coins to get a coke

If you put 3 coins, you can select a drink

(as long as you don't cancel in the middle)

After you select your drink you get your drink

If you press cancel, you get your money back

Example 2: ignition button

Engine starts and stops with button push

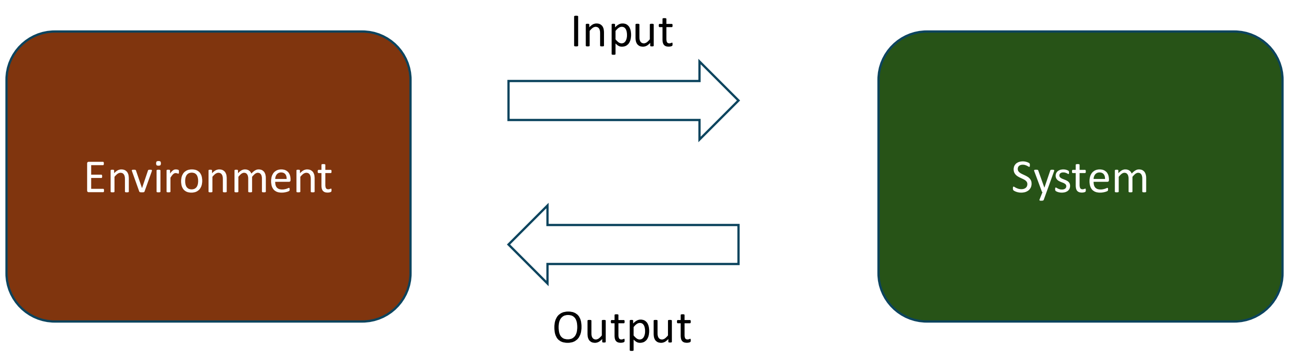
If engine is off, it stays off until I push

- If I never push it stays off forever

If engine is on, it stays on until I push

- If I never push it stays on forever

What is a reactive program?



Reactive systems as algorithms

How do we describe reactive systems?

- Finite state model

How do we specify their behavior?

- Monadic Second Order (MSO) logic, Linear Temporal Logic (LTL)

How to verify?

- Satisfiability of MSO => Emptiness of Automata

How to design?

- Reactive system as a game
- Program implemented as a strategies (Realizability vs. Synthesis)

Finite state models

A reactive system can be naturally described as a *finite state transducer*

- There are finitely many controls/states
- Input is from a finite set I
- Action is from a finite set A
- "coin-coin-coin-drink-coin-coin-cancel-return2coins-…"

The behaviors are commonly defined as a *finite state automaton*

- Pair interleaved input-action
- Alphabet: $(I \cup \{\epsilon\}) \times (A \cup \{\epsilon\})$
- "(coin, ϵ)-(coin, ϵ)-(coin, drink)-(coin, ϵ)-(coin, ϵ)-(cancel, return2coins)-…"

DFA

A *deterministic finite automaton* is $A = (\Sigma, Q, q_0, \delta, F)$ where

- Σ is the alphabet
- Q is a finite set of states
- $q_0 \in Q$ is the initial state
- $\delta: Q \times \Sigma \to Q$ is a deterministic transition table
- $F \subseteq Q$ is a set of accepting states

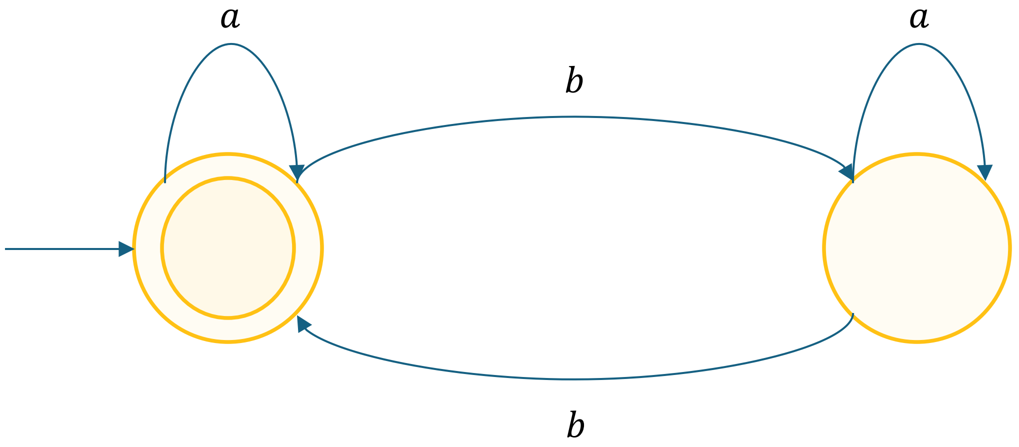
A word is accepted by A if running A over the word stops at an accepting state

$L(A)$: the language of A (the set of words accepted by A)

Example

$\Sigma = \{a, b\}$

Design a DFA A such that $L(A)$ is the set of all words containing even number of b's:



NFA

A *nondeterministic finite automaton* is $A = (\Sigma, Q, q_0, \delta, F)$ where

- Σ is the alphabet
- Q is a set of states
- $q_0 \in Q$ is the initial state
- $\delta \subseteq Q \times \Sigma \cup \{\epsilon\} \times Q$ is a nondeterministic transition table
- $F \subseteq Q$ is a set of accepting states

A word is accepted by A if a lucky run over the word stops at an accepting state

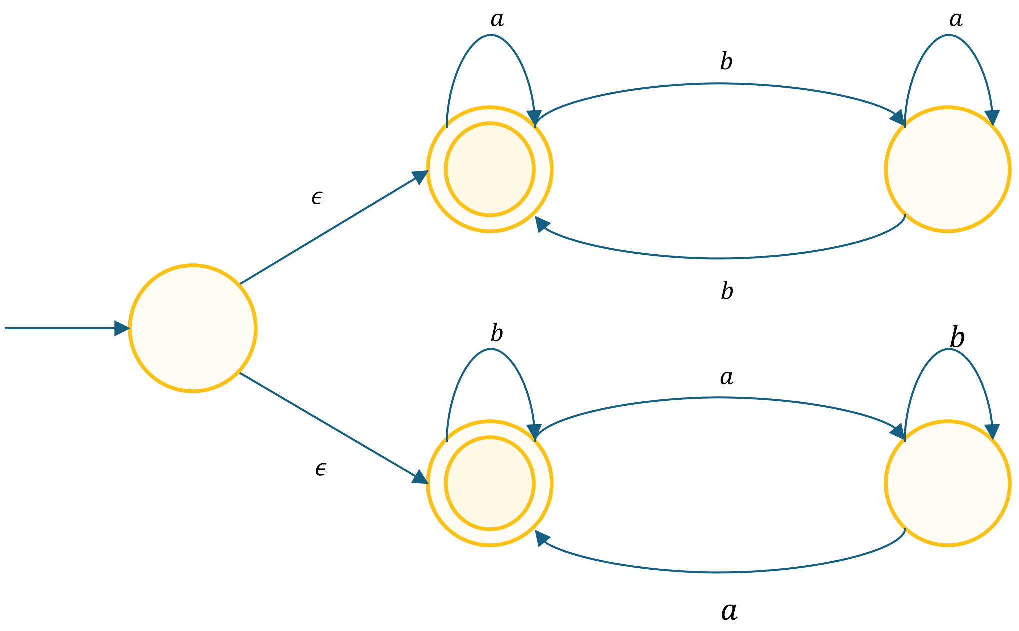
Theorems:

- NFA = DFA = Regular expressions
- NFA/DFA is closed under complementation, union, intersection
- Emptiness is decidable

Example

$\Sigma = \{a, b\}$

Design a DFA A such that $L(A)$ is the set of all words containing even number of a's or even number of b's:



Regular expression

R is a regular expression if R is

- a for some a in the alphabet Σ
- ϵ
- $\emptyset$
- $(R_1 \cup R_2)$ where R_1 and R_2 are regular expressions
- $(R_1 \circ R_2)$ where R_1 and R_2 are regular expressions
- (R_1^*) where R_1 is a regular expression

Theorems:

- Regular expressions => NFA
- DFA => Regular expressions

Monadic Second Order Logic

Second-Order Logic

Second-Order (SO) logic extends FOL with second-order variables

- E.g., R^1 (a set variable), R^2 (a relation variable), F^1 (a unary-function variable), F^2 (a binary-function variable) …
- $\exists F^2 \forall x, y(F^2(x, y) = F^2(y, x) \wedge \ \ldots)$
- Sound and complete proof system does not exist

Monadic Second-Order (MSO) logic allows set variables only

- E.g., $\exists S\big(\ldots \exists x(x \in S \wedge \ \ldots)\big)$

Example

the length of the word is even?

$$\exists R \exists G \begin{pmatrix} \forall x(\neg R(x) \vee \neg G(x)) \\ \wedge R(0) \\ \wedge \forall x \forall y(S(x,y) \wedge R(x) \rightarrow G(y)) \\ \wedge \forall x \forall y(S(x,y) \wedge G(x) \rightarrow R(y)) \\ \wedge \forall y(\text{last}(y) \rightarrow G(y)) \end{pmatrix}$$

Logic vs. Automata

Theorem (Buchi-Elgot-Trakhtenbrot 1960) : A language $L \in \Sigma^+$ is regular iff. L is MSO-definable.

- =>: Given a NFA A, Construct an MSO sentence over finite words that precisely describes L(A)
- <=: Given an MSO sentence φ, construct a NFA that accepts precisely the language defined by φ

Corollary: The satisfiability of MSO over finite words is decidable.

NFA to MSO

$A = (\Sigma, Q, q_0, \delta, F)$, $Q = \{q_0, \dots, q_k\}$, the accepting sequence is in Q^+.

$$\varphi_A = \exists X_0 X_1, \dots X_k:$$

$$\wedge \ \bigwedge_{i \neq j} \forall y \neg (X_i(y) \wedge X_j(y))$$

$$\wedge \ X_0(0)$$

$$\wedge \ \forall y, z \left(s(y,z) \Rightarrow \ \bigvee_{(i,a,j) \in \delta} \left(X_i(y) \wedge Q_a(y) \wedge X_j(z) \right) \right)$$

$$\wedge \ \forall y \left(last(y) \Rightarrow \bigvee_{j \in F} X_j(y) \right)$$

MSO to NFA

$x \in X \mid x = y \mid s(x, y) \mid Q_a(x) \mid \exists x \varphi(x) \mid \exists X \varphi(X) \mid \varphi \vee \varphi \mid \neg\varphi$

Remove first-order variables:

$Sing(X) \mid X \subseteq Y \mid Suc(X, Y) \mid X \subseteq Q_a \mid \exists X \varphi(X) \mid \varphi \vee \varphi \mid \neg\varphi$

Inductively convert:

- φ ➔ $A(\varphi)$ (DFA over Σ)
- $\varphi(X_1, \ldots X_n)$ ➔ $A(\varphi)$ (DFA over $\Sigma \times \{0,1\}^n$)

Büchi Automata

Infinite words

- DFA/NFA accept words of finite length.

- How about infinite words?
  - E.g., “abababababab…”

- An *infinite word* is a map $\alpha: \mathbb{N} \rightarrow \Sigma$.
  - E.g., $\alpha(i) = \begin{cases} a & if\ i \text{ is even} \\ b & \text{otherwise} \end{cases}$

- The set of all infinite words is denoted as Σ^{ω}.

Büchi automata

How can an automaton accept an infinite word?

- Büchi automata (named after Purdue’s Julius Richard Büchi)
- “an accepting state is hit infinitely many times”

Büchi automata

A *Büchi Automaton* (BA) is $A = (\Sigma, Q, q_0, \delta, F)$ where

- Σ is the alphabet
- Q is a finite set of states
- $q_0 \in Q$ is the initial state
- $\delta \subseteq Q \times \Sigma \times Q$ is a nondeterministic transition table
- $F \subseteq Q$ is a set of accepting states

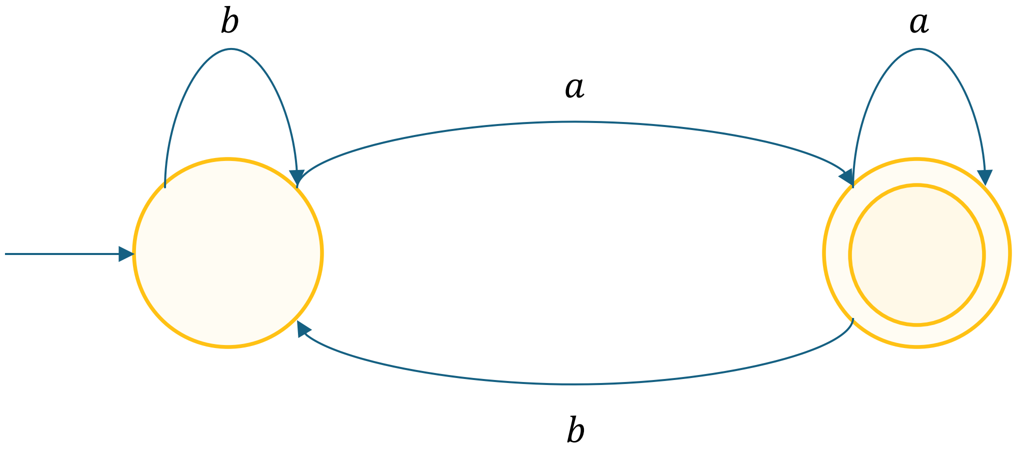
A run of A on an infinite word α is a map $r: \mathbb{N} \to Q$ such that $r(0) = q_0$, and for any $i \in \mathbb{N}$, $\big(r(i), \alpha(i), r(i+1)\big) \in \delta$

r is accepting if $r(i) \in F$ for infinitely many $i \in \mathbb{N}$

A word α is accepted by A if there is some accepting run of A on α

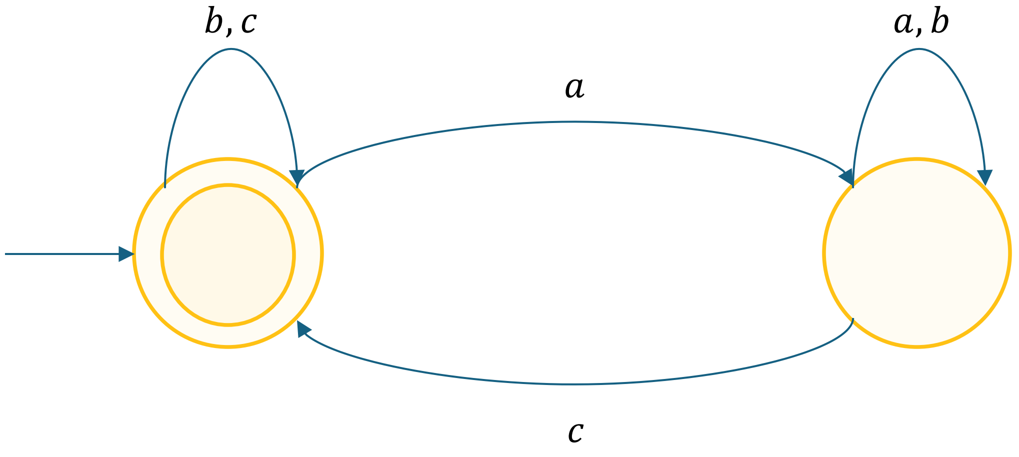
Example

$L = \{\alpha \in \{a, b\}^{\omega} \mid \alpha \text{ has infinitely many } a's\}$



Example

"Every 'a' must be followed eventually by a 'c'."



Properties of BA

- BA is closed under
  - Union
  - Intersection
  - Complementation ($2^{n \cdot \lg n}$ blow-up)
  - Projection
- BA is not determinizable!
  - E.g., infinite words over $\{0,1\}^\omega$ that contain finitely many 1's

- Emptiness of BA is decidable (NLOGSPACE-complete)

Decision Procedure for MSO

- Implementation:
  - Mona (https://www.brics.dk/mona/)

- How is the complexity?
  - Not elementary: $2^{2^{\cdot^{\cdot^{\cdot^{n}}}}}$

Linear Temporal Logic

Logic vs. Automata, round 2

Theorem (Büchi 1960): A language $L \in \Sigma^\omega$ is regular iff. L is MSO-definable.

Corollary: The satisfiability of MSO over infinite words is decidable.

“Algorithm not very efficient”

Implementation:

- Mona (https://www.brics.dk/mona/)

How is the complexity?

- Not elementary: $2^{2^{\cdot^{\cdot^{\cdot^{n}}}}}$

Linear Temporal Logic (Pnuerli 1977)

Syntax:

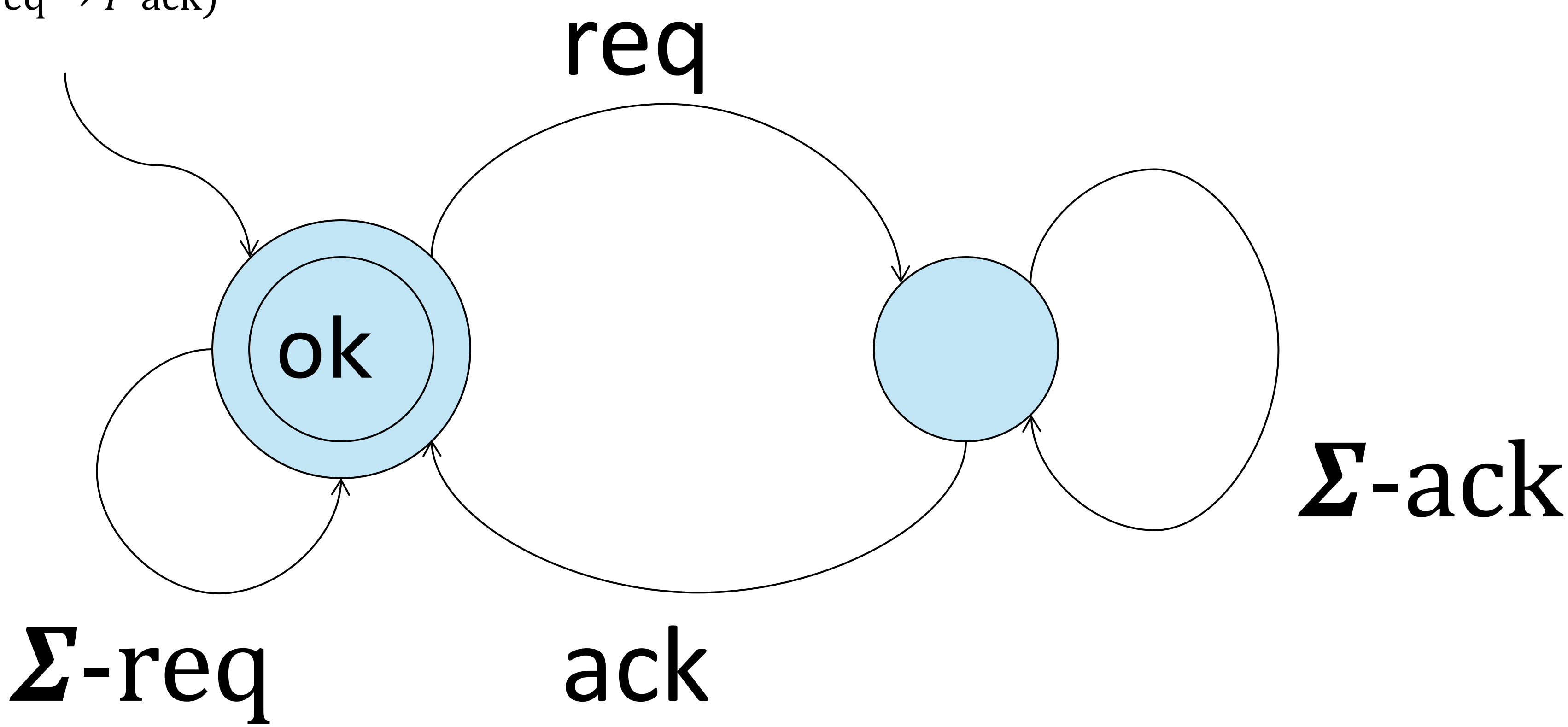
$$LTL ::- true \;|\; p \;|\; X\,\alpha \;|\; F\,\alpha \;|\; G\,\alpha \;|\; \alpha\, U\, \beta \;|\; \alpha \vee \beta \;|\; \neg\alpha$$

"next" *"eventually"* *"always"* *"until"*

Semantics: interpreted over infinite traces

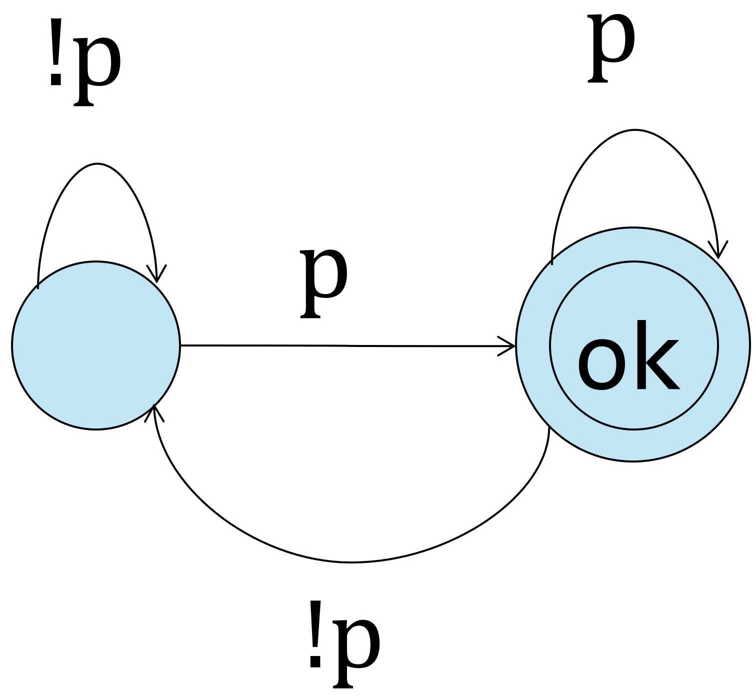
Example

$G\ (\text{req} \rightarrow F\ \text{ack})$



Example

$G\ F\ \mathrm{p}$



LTL to FO

For infinite words (or $(\mathbb{N}, <)$):

- LTL = FO = star-free regular language < MSO = BA = Regular language

φ_{LTL} to $\varphi_{FO}(x)$ by structural induction:

- $p \Rightarrow \bigvee_{p \in Y} Q_Y(x)$
- $X\,\varphi \Rightarrow \exists y \big(s(x,y) \wedge \varphi_{FO}(y)\big)$
- $F\,\varphi \Rightarrow \exists y \big(x \leq y \wedge \varphi_{FO}(y)\big)$
- $\varphi\, U\, \psi \Rightarrow \exists y \Big(x \leq y \wedge \psi_{FO}(y) \wedge \forall z \big(x \leq z < y \rightarrow \varphi_{FO}(z)\big)\Big)$
- $\varphi_1 \vee \varphi_2 \Rightarrow {\varphi_1}_{FO}(x) \vee {\varphi_2}_{FO}(x)$
- $\neg\varphi \Rightarrow \neg\varphi_{FO}(x)$

LTL to BA

Why LTL?

- More efficient algorithm!

A maximal-model-based algorithm (Wolper-Vardi-Sistla 1983)

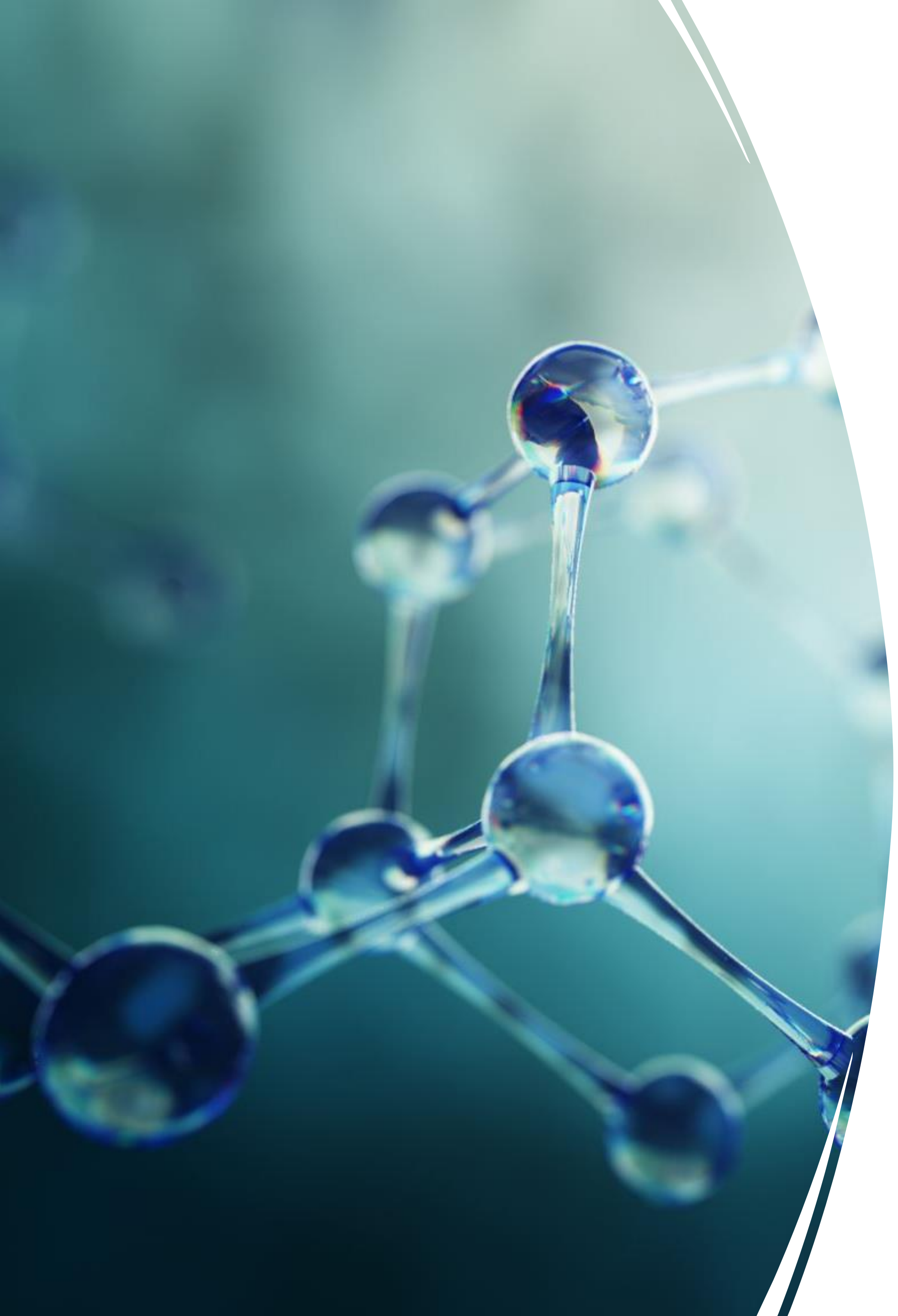
- Intuition: compute the maximal set of satisfied subformulae
- E.g., $\varphi : p\ U\ (\neg p \wedge q)$
- With input: $p,\quad pq,\quad p,\quad q,\quad p,\quad \emptyset,\quad q,\quad \ldots$

Closures

How to define the subformulae?

Let φ be LTL, then $CL(\varphi)$ is the smallest set satisfying:

- $\varphi \in CL(\varphi)$
- If $\neg\psi \in CL(\varphi)$, then $\psi \in CL(\varphi)$
- If $\varphi_1 \vee \varphi_2 \in CL(\varphi)$, then $\varphi_1, \varphi_2 \in CL(\varphi)$
- If $X\ \psi \in CL(\varphi)$, then $\psi \in CL(\varphi)$
- If $\varphi_1\ U\ \varphi_2 \in CL(\varphi)$, then $\varphi_1, \varphi_2, X\ \varphi_1\ U\ \varphi_2 \in CL(\varphi)$



Atoms

$A \subseteq CL(\varphi)$ is an atom (maximally consistent subset) if

- $\forall \neg\varphi' \in CL(\varphi),\ \varphi' \in A\ \ iff\ \ \neg\varphi' \notin A$
- $\forall \varphi_1 \vee \varphi_2 \in CL(\varphi), \varphi_1 \vee \varphi_2 \in A\ \ iff\ \ \varphi_1 \in A\ or\ \varphi_2 \in A$
- $\forall \varphi_1\ U\ \varphi_2 \in CL(\varphi), \varphi_1 U\ \varphi_2 \in A\ \ iff\ \ \varphi_2 \in A\ or\ (\varphi_1 \in A\ and\ X\ \varphi_1\ U\ \varphi_2 \in A)$

LTL to BA

States: set of atoms of φ

Transitions: $(A_1, X, A_2) \in \delta$ if and only if

- $A_1 \cap Voc(\varphi) = X$
- $\forall\, X\,\varphi_1 \in CL(\varphi),\;\; X\,\varphi_1 \in A_1 \;\; iff \;\; \varphi_1 \in A_2$

Initial states: $\{A \mid \varphi \in A\}$

LTL to BA

When to accept?

- For every $\varphi_1\ U\ \varphi_2$ in the atom, φ_2 has to eventually occur!
- A “good” state either does not have $\varphi_1\ U\ \varphi_2$ or has φ_2
- Good states must be infinitely many:
- $\mathrm{F}_i = \{A \in Atoms(\varphi) |\ \varphi_1\ U\ \varphi_2 \notin A\ or\ \varphi_2 \in A\}$
- Accepting states: F_i for every $\varphi_1\ U\ \varphi_2$

Complexity: $2^{O(|\varphi|)}$

Model Checking

Model Checking Problem: Given a finite transition system TS and an LTL formula φ, does every sequence α generated by TS satisfies φ?

Check: $L(A_{TS} \wedge A_{\neg\varphi}) = \emptyset$?

Complexity: $2^{O(|\varphi|)} \cdot |TS|$

Implementation: NuSMV

Reactive Synthesis

Cinderella Game

Cinderella

- Can empty two **adjacent** buckets
- If she can keep stepmother from winning, she wins

Stepmom

- Splits her water among all buckets
- If any overflows she wins

What is the B for which the advantage shifts from stepmother to Cinderella?

Stepmother's Perspective

She is trying to satisfy $F\ overflow$

What is the largest B* such that if B < B* she wins?

$\exists Strategy, \forall cind, B\ \ B < B^* \Rightarrow win(Strategy, cind)$

Cinderella’s perspective

She is trying to satisfy $G\ \neg overflow$

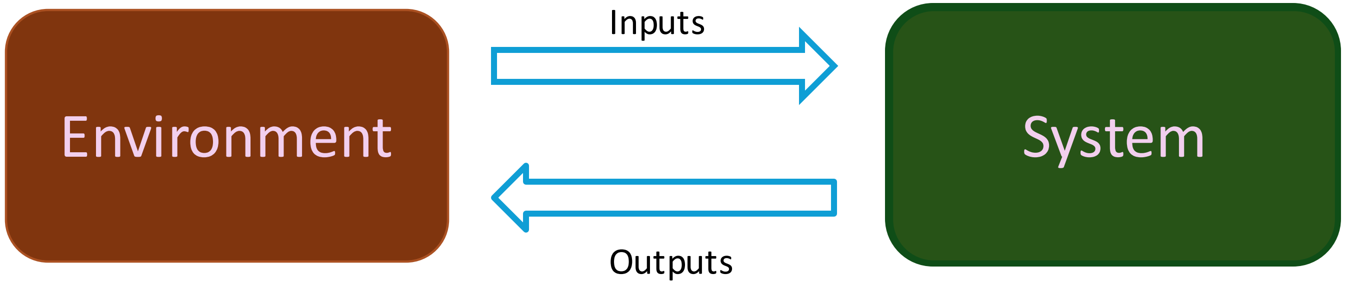
If B < 1 this is impossible

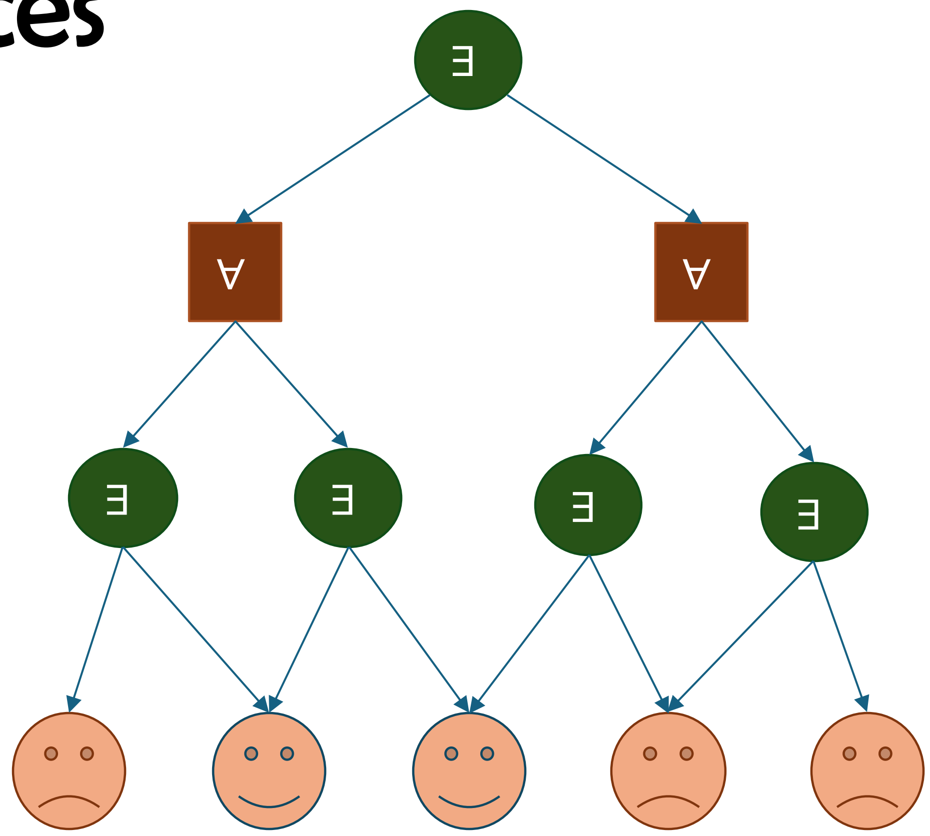
What is the smallest B* such that if B > B* she is safe?

$$\exists Strategy, \forall mom, B\ \ B > B^* \Rightarrow win(Strategy, mom)$$

$B^* = 2$

System evolution a tree of choices





Environment: choose an input from a set I
System: choose an output from a set O

Strategy: a function $f: I^* \to O$
Winning Condition: An MSO formula on $I \cup O$
Winning Strategy: all plays satisfy the winning condition

Realizability

Church's Problem (1957): the existence of winning strategy for specification expressed in MSO.

- Synthesis: obtaining such winning strategy

MSO Realizability (Büchi-Landweber 1969): the MSO Realizability problem is decidable.

- If a winning strategy exists, then a finite-state strategy exists.
- Realizability algorithm produces finite-state strategies.

Rabin's Realizability Algorithm (1972)

Rabin Tree Automata on infinite k-ary trees: $A = (\Sigma, Q, q_0, \delta, F)$ where

- Σ is the alphabet
- Q is a finite set of states
- $Q_0 \subseteq Q$ is the initial state set
- $\delta: Q \times \Sigma \to 2^{Q^k}$ is a nondeterministic transition table
- $\alpha \subseteq 2^{2^Q \times 2^Q}$ is a set of accepting conditions

Acceptance Condition:

- Let $\alpha = \{(G_1, B_1), \ldots, (G_l, B_l)\}$, $G_i, B_i \subseteq Q$
- Along every branch, for some i, G_i is visited infinitely often, and B_i is visited finitely often

Rabin's Realizability Algorithm (1972)

Emptiness of Tree Automata

- PTIME on finite trees (Doner 1965)
- NP-complete on infinite trees (Emerson-Jutla 1991)

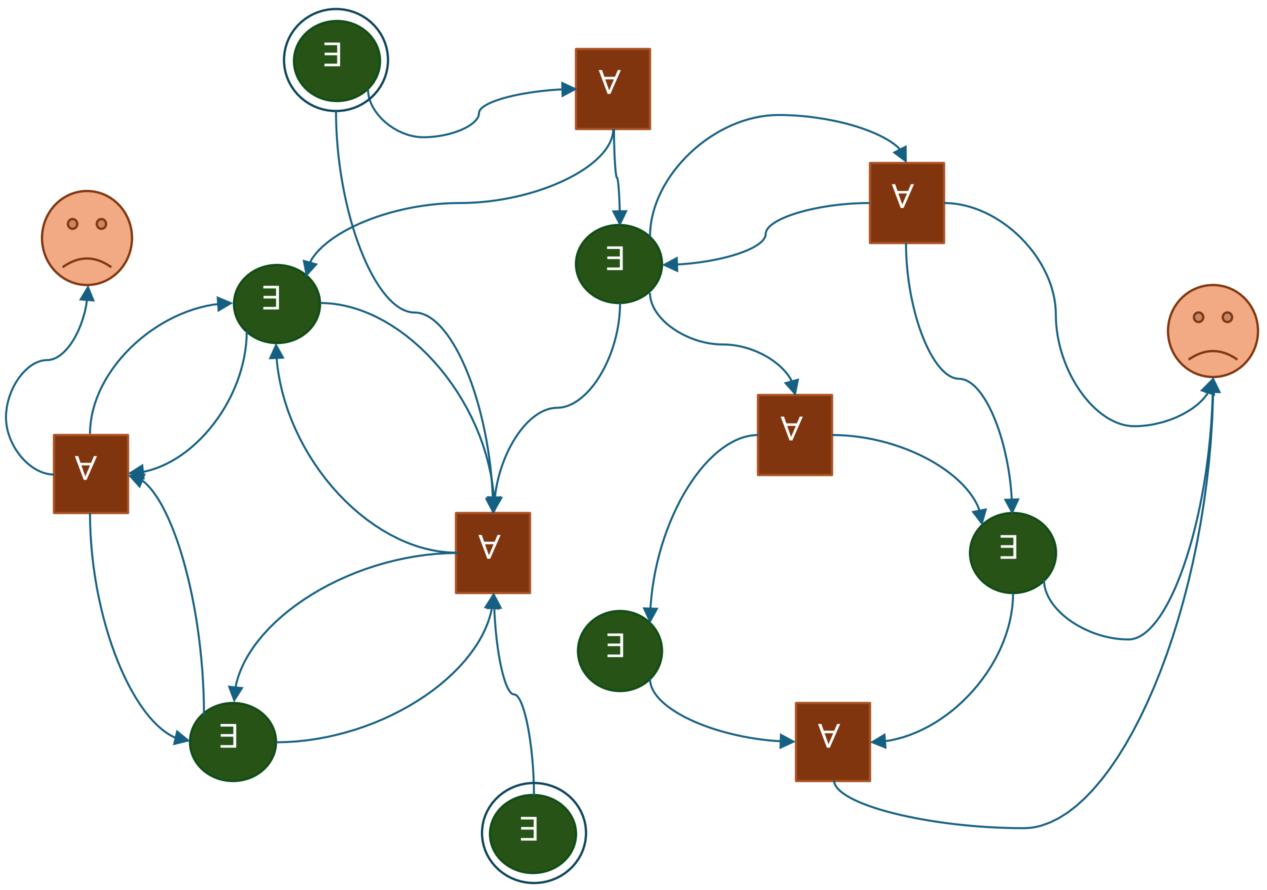
Real(φ):

- A strategy can be represented as labels on the infinite game tree
- Construct a Rabin tree automaton A_φ that accepts a labeled tree iff the labels represent a strategy, and the strategy is winning wrt to the condition φ
- Check the emptiness of A_φ; if nonempty, extra a strategy from the witness

Complexity:

- non-elementary (the construction of A_φ)
- 2EXPTIME-complete for LTL spec (Rosner 1990)

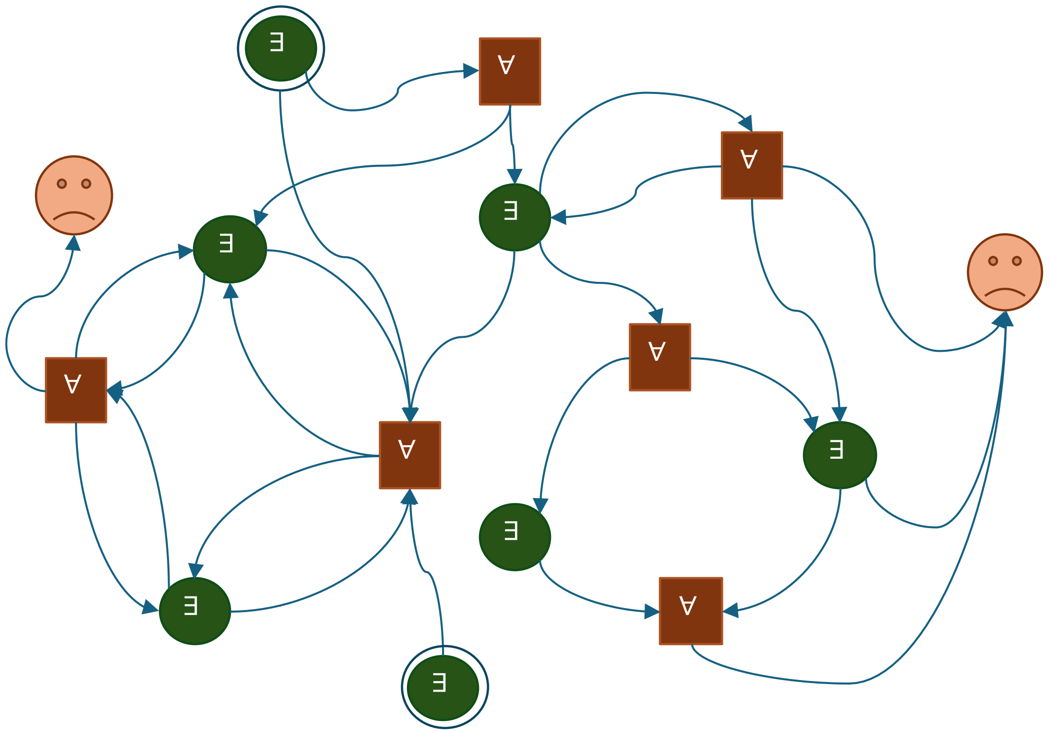
System/Environment as a graph



Two-person games

Winning condition:

- In general, LTL formulas
- Expensive! (double exponential)
- Reachability games
- Safety games
- ??

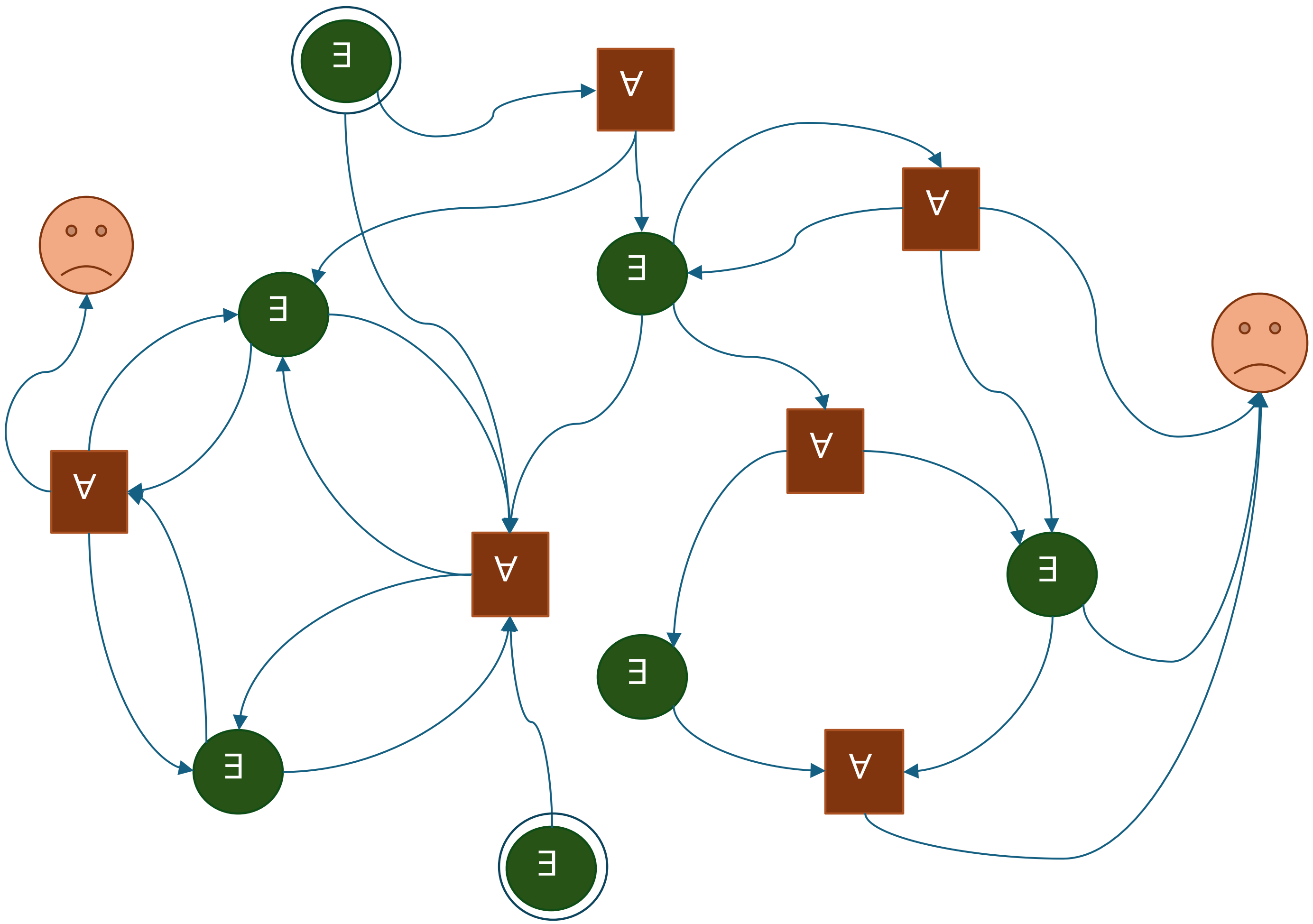


Specialized games

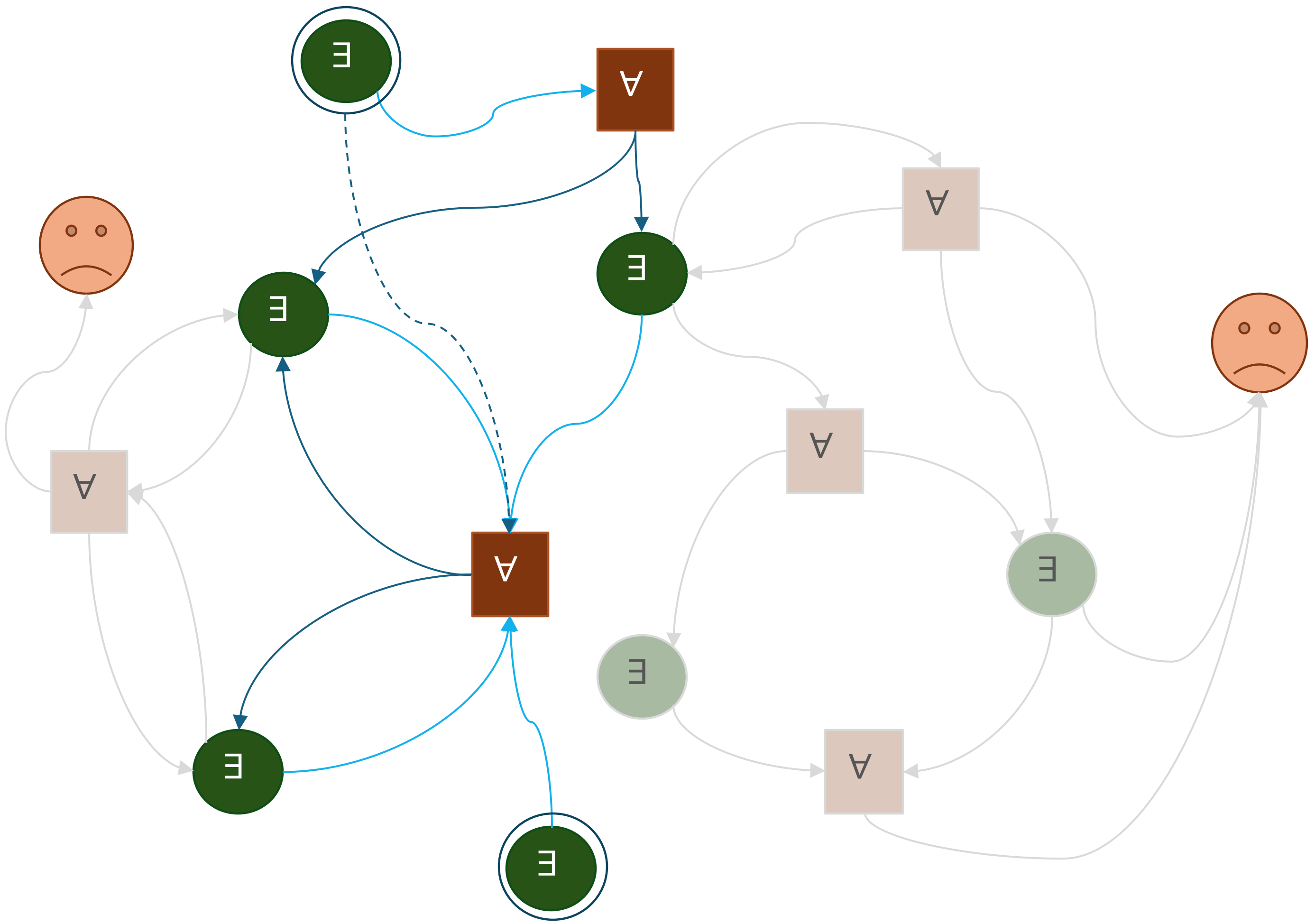
LTL provides a general language for specifications

- Safety Games ($G\ p$)
  - System "wins" if it can stay away from the bad states $\neg p$
- Reachability Games ($F\ p$)
  - System "wins" if it can reach a good state that satisfies p
  - $F\ p$ are often referred to as *liveness properties*

Solving reachability games



Solving reachability games



Reachability games enjoy memoryless strategies

At every ∃ state, the decision of what transition to make depends only on the current state

- easy to translate into code

Finite vs. Infinite Games

- Chess & Go
- Banach–Mazur game
- For finite games, one of the players has a winning strategy

Synthesis of AMBA AHB from Formal Spec

AMBA: Advanced Microcontroller Bus Architecture

AMBA AHB: a high-performance system backbone bus

- Formal Spec written in LTL
- Circuit automatically synthesized!
- AHB Slave synthesized in 21.5 second,
- (has 214 gates with area 429 square units)