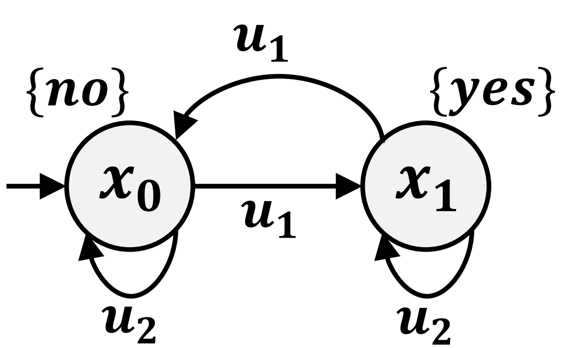
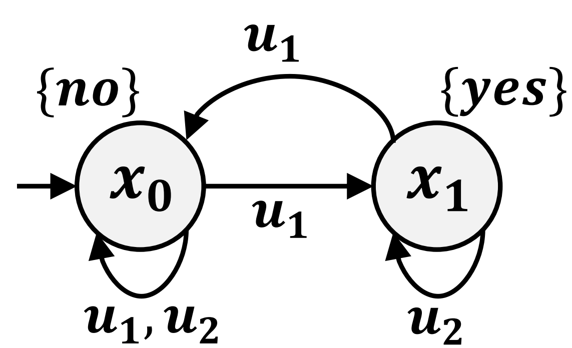
Game-Based LTL Control Synthesis

Role of Inputs

- Control input is not important in the verification problem since we want to check the satisfaction for all runs
- Some “bad” runs can be avoided by suitably choosing inputs
- $T_1 \nvDash \Diamond\Box yes$, e.g., $(x_0 x_1)^\omega$ yields $(\{no\}\{yes\})^\omega$
- We can choose input sequence $u_1(u_2)^\omega$, which gives $\{no\}(\{yes\})^\omega$
- In general, the effect of an input is non-deterministic $|Post(x,u)| > 1$
- We also need to handle all possible consequences of the input



Deterministic LTS T_1 with $U = \{u_1, u_2\}$



Non-Deterministic LTS T_2

Synthesis Problem

- A **control strategy** is a function $C: X(X)^* \to U$
- State run under C: $x_0 x_1 \cdots x_n \cdots$ such that $x_{i+1} \in Post(x_i, C(x_0 \cdots x_n))$
- The set of all infinite runs of T under C: $Run(C/T)$
- The set of all infinite traces of T under C: $Trace(C/T)$

Control Synthesis Problem

Given an LTS T and a property $P \subseteq \left(2^{AP}\right)^{\omega}$, find a control strategy $C: X(X)^* \to U$ such that $C/T \vDash P$, i.e., $Trace(C/T) \subseteq P$.

- for LTL formula ϕ, $C/T \vDash \phi$ means $Trace(C/T) \subseteq Word(\phi)$

Case of Deterministic System

- A control strategy is not a single input sequence
- A controller should be **reactive** to non-determinism
- A singe (infinite) input sequence is enough **when it is deterministic**
- Deterministic synthesis problem is essentially a **path planning problem or open-loop control** problem and can be solved by model checking (returns a counter-example if negative)

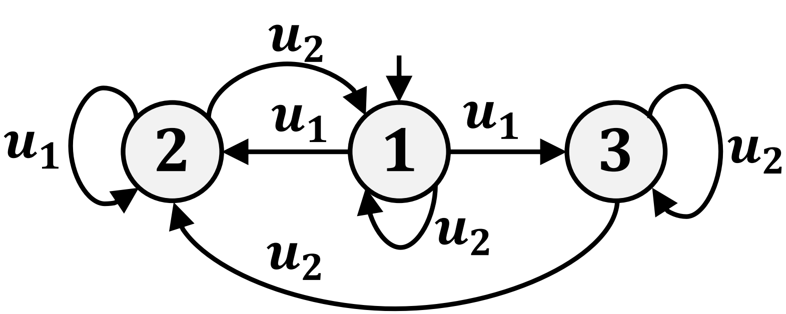
LTL Path Planning by Model Checking

Given: T and LTL Formula ϕ

- Use model checker to verify whether or not $T \vDash \neg\phi$
- If $\textsc{Check}(T, \neg\phi) = \text{"Yes"}$, then return "no controller exists"
- If $\textsc{Check}(T, \neg\phi) = \text{"No"}$, then **the model-checker will provide an infinite run $\rho \in X^\omega$ as counter-example**, i.e., $L(\rho) \nvDash \neg\phi$. Return "ρ" as the planned path

General Case as a Two-Player Game

- **Player-C: controller** chooses an input $u \in U$ that is defined at $x \in X$
- **Player-A: adversary** chooses a successor $x' \in Post(x, u)$ and the system moves to x'; then Player-C chooses and so forth…
- The strategy of Player-C is actually a controller $C: X(X)^* \to U$
- The strategy of Player-A is a function $A: X(X)^* U \to X$ that resolves non-deter.
- If C and A are given, the initial-state $x_0 \in X_0$ is given, then the run is uniquely determined, denoted by $\rho(A, C, x_0)$ and denote the trace by $\sigma(A, C, x_0)$
- Then $C/T \models P$ iff $\forall A, \forall x_0 \in X_0: \sigma(A, C, x_0) \in P$



- $C/T \models \Box\neg 3$ can be achieved by fixing u_2
- $C/T \models \Diamond 2$ cannot be achieved by any C

control with non-determinism as a two-player game

Two-Player Games

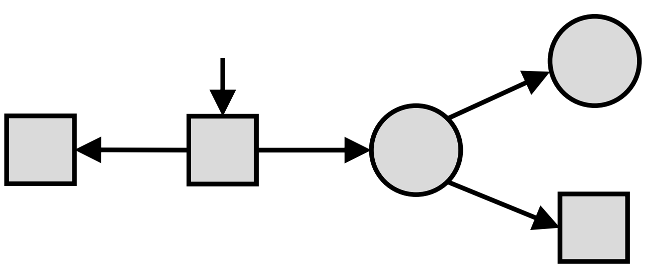
- **Safety Game: stay within safe states (not to reach unsafe states)**
- **Reachability Game: reach desired states within finite number of steps**
- **Büchi Game: visit desired states infinitely often**
- **Rabin Game: visit desired states infinitely often avoiding rejected states**

➢ **Safety game is related to regular safety**

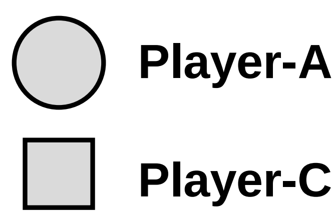
➢ **Reachability game is related to scLTL**

➢ **Büchi game and Rabin game are related to general LTL**

➢ **Two-player game can also be formulated by explicitly partitioning the state-space for each player**



A different formulation of two-player game

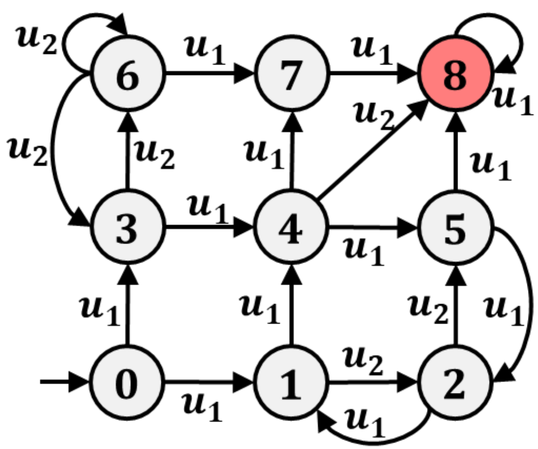


Safety Game

Safety Game (Reach Avoid Game)

For LTS T and a set of unsafe region $B \subseteq X$, find a controller C such that the run **never reaches** B under any possible adversary A.

- **Winning Region:** the set of states from which Player-C can win
- Player-C wins (exists a controller) if $X_0 \subseteq X_{win}$
- By definition, control strategy is history based, but state-based strategy (memoryless) strategy is sufficient for many games



Can you avoid state 8?

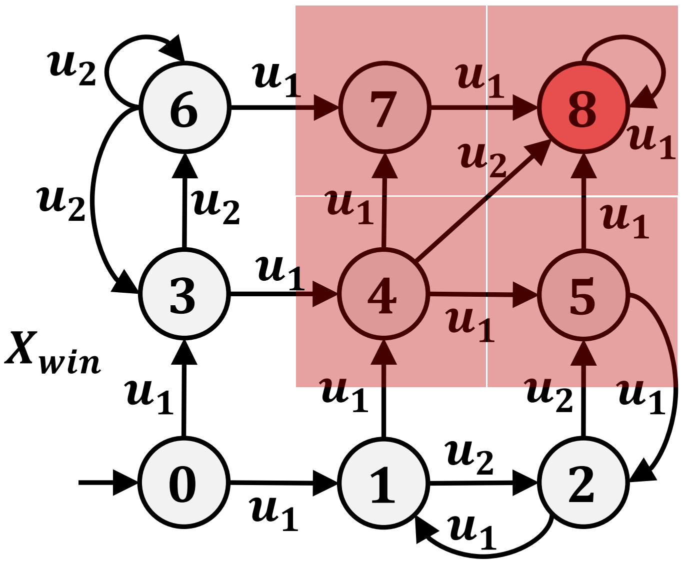
Solving Safety Game

- We need to avoid unsafe states $B \subseteq X$
- To avoid B, we need to avoid states that cannot avoid B

$$B_1 = \text{Avoid}(B) = \{x \in X: \forall u \in U, Post(x,u) \cap B \neq \emptyset\}$$

- Then we also need to avoid $B_2 = \text{Avoid}(B_1)$
- Keep deleting states until we get the winning region $X_{win} \subseteq X$ s.t.

$$X_{win} \cap B = \emptyset \quad \text{and} \quad \forall x \in X_{win}, \exists u \in U: Post(x,u) \subseteq X_{win}$$



Safety Game Algorithm

- Delete B from X, $B \leftarrow B \cup \text{Avoid}(B)$
- Repeat the above until $B = B \cup Avoid(B)$
- State remained are X_{win}
- If $X_0 \not\subseteq X_{win}$, then "no controller"
- Otherwise, C chooses an input $u \in U$ at each $x \in X$ s.t. $Post(x,u) \subseteq X_{win}$

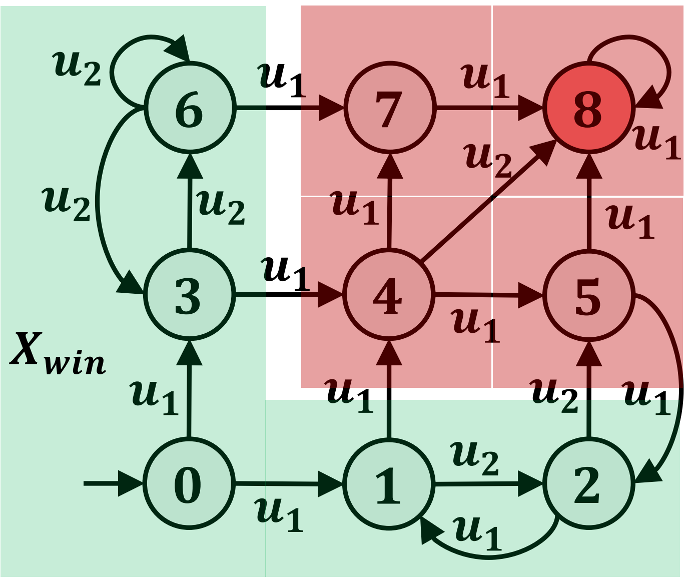
Solving Safety Game

- We need to avoid unsafe states $B \subseteq X$
- To avoid B, we need to avoid states that cannot avoid B

$$B_1 = \text{Avoid}(B) = \{x \in X: \forall u \in U, Post(x,u) \cap B \neq \emptyset\}$$

- Then we also need to avoid $B_2 = \text{Avoid}(B_1)$
- Keep deleting states until we get the winning region $X_{win} \subseteq X$ s.t.

$$X_{win} \cap B = \emptyset \quad \text{and} \quad \forall x \in X_{win}, \exists u \in U: Post(x,u) \subseteq X_{win}$$



Safety Game Algorithm

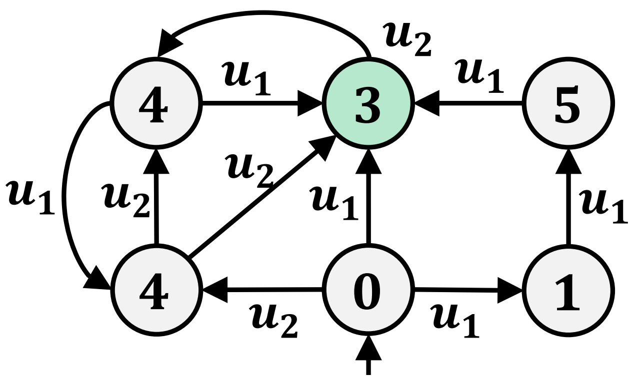
- Delete B from X, $B \leftarrow B \cup \text{Avoid}(B)$
- Repeat the above until $B = B \cup Avoid(B)$
- State remained are X_{win}
- If $X_0 \not\subseteq X_{win}$, then "no controller"
- Otherwise, C chooses an input $u \in U$ at each $x \in X$ s.t. $Post(x,u) \subseteq X_{win}$

Reachability Game

Reachability Game

For LTS T and a set of desired region $D \subseteq X$, find a controller C such that the run can always reaches D within finite steps under any possible A.

- **Player-C losses the game iff the adversary can let the system loop in a cycle in which there is no desired state**
- **If Player-C wins the game, then it can always reaches D within $|X|$ steps**



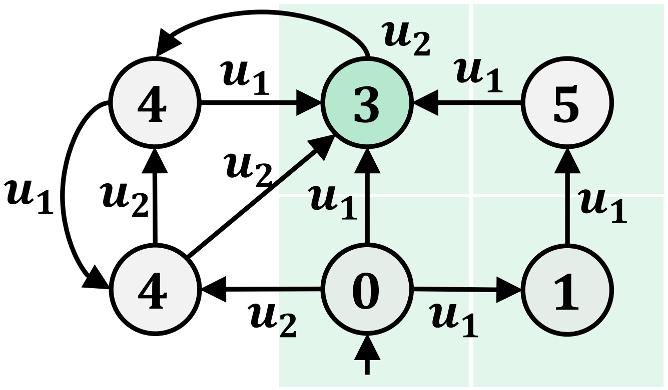
Can you reach state 3?

Solving Reachability Game

- To guarantee reaching $D \subseteq X$ in one step, we must in states $D_1 = D \cup \mathrm{CPre}(D)$

$$\mathrm{CPre}(D) = \{x \in X: \exists u \in U, Post(x,u) \subseteq D\}$$

- To guarantee reach D in two steps, we must in states $D_2 = D_1 \cup \mathrm{CPre}(D_1)$
- By keep expending the region of attraction, we get the winning region $X_{win} = \mathrm{Attr}(D) := D \cup D_1 \cup \cdots \cup D_n = D_n$. For each D_{i+1} we can always move to D_i to be "closer" to the target region



- $D_0 = \{3\}, D_1 = \{3,5\}, D_3 = \{1,3,5\}$
- $D_4 = X_{win} = \{0,1,3,5\}$
- $C(0) = C(1) = C(5) = u_1, C(3) = u_2$

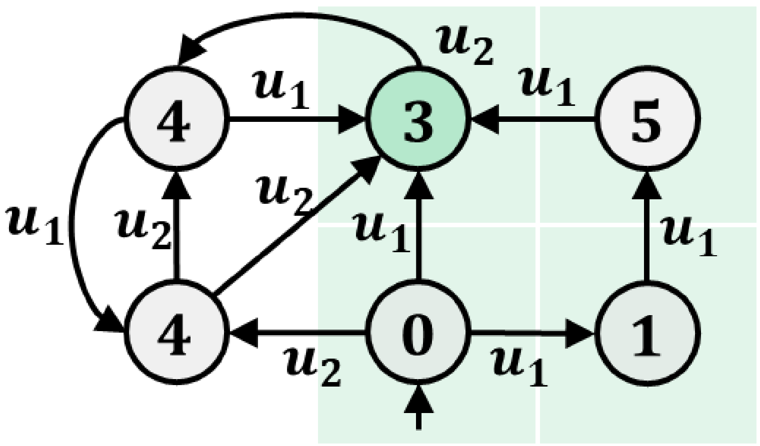
Reachability Game Algorithm

- Define $D_0 = D$
- Repeat $D_{i+1} = D_i \cup \mathrm{CPre}(D_i)$ until $D_i = \mathrm{CPre}(D_i)$
- If $X_0 \nsubseteq X_{win} = D_n$, then "no controller"
- Otherwise, C chooses an input $u \in U$ at each $x \in D_i$ s.t. $Post(x,u) \subseteq D_0 \cup \cdots \cup D_{i-1}$

Büchi Game

Büchi Game

For LTS T and a set of accepting states $F \subseteq X$, find a controller C such that the run can always visits F infinitely often under any possible A.



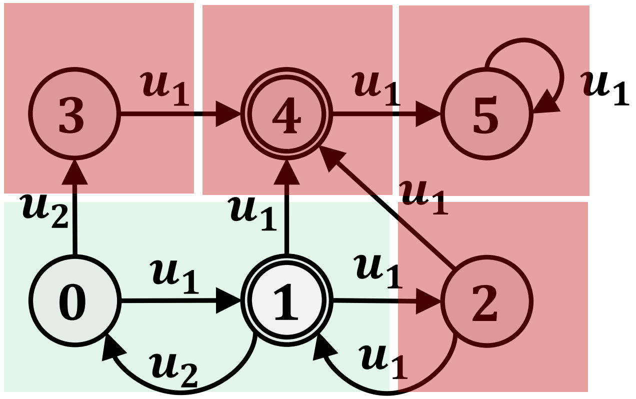
- Player-C wins the reachability game but cannot win the Büchi game for $F = \{3\}$
- State 3 can only be guaranteed to be visited once
- We should also take care of recurrence for what happens after reaching F

Solving Büchi Game

- To visit F again, we must in $\mathrm{Attr}(F)$
- We need to avoid $W_A = X \setminus \mathrm{Attr}(F)$ from F
- Therefore, we shrink accepting states to $F = F \setminus \mathrm{APre}(W_A)$, where

$$\mathrm{APre}(W_A) = \{x : \forall u \in U, Post(x, u) \cap W_A \neq \emptyset\}$$

- Since F is changed, we need to computed $\mathrm{Attr}(F)$ and $\mathrm{APre}(W_A)$ again



- $F = \{1, 4\}, W_A = \{5\}, \mathrm{APre}(W_A) = \{4, 5\}$
- $F = \{1\}, \mathrm{Attr}(F) = \{0, 1\}, W_A = \{2, 3, 4, 5\}, \mathrm{APre}(W_A) = \{3, 4, 5\}$
- $F \setminus \{3, 4, 5\} = \{1\}$

Büchi Game Algorithm

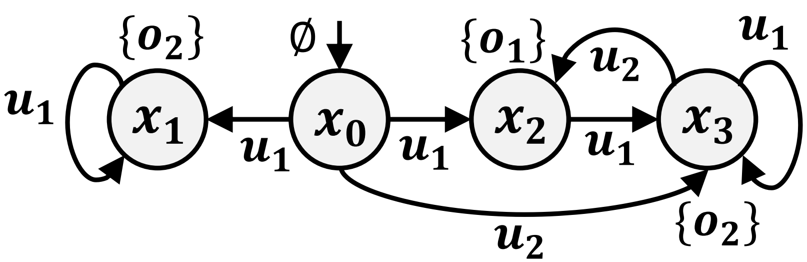
- $F = F \setminus \mathrm{APre}\big(X \setminus \mathrm{Attr}(F)\big)$
- Repeat above until $\mathrm{APre}\big(X \setminus \mathrm{Attr}(F)\big) \cap F = \emptyset$
- If $X_0 \nsubseteq X_{win} = Attr(F)$, then “no controller”
- Otherwise, C chooses an input $u \in U$ based on the reachability game for F

LTL Synthesis: Deterministic Case

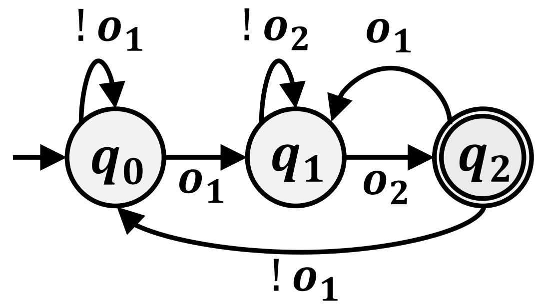
- Suppose we have an LTS T and an LTL formula ϕ
- We want to find a controller C such that $C/T \models \phi$
- Assume LTL formula ϕ can be accepted by a DBA A_ϕ (dLTL)
- Then we build $T \otimes A_\phi$ with accepting states $F_\otimes$
- Solve the Büchi game for $T \otimes A_\phi$ with $F_\otimes$
- The winning strategy in $T \otimes A_\phi$ can be mapped directly to T by looking at the first component

➢ Note: we cannot use NBA for ϕ because the LTS T may be non-deterministic; otherwise, we cannot really control the system

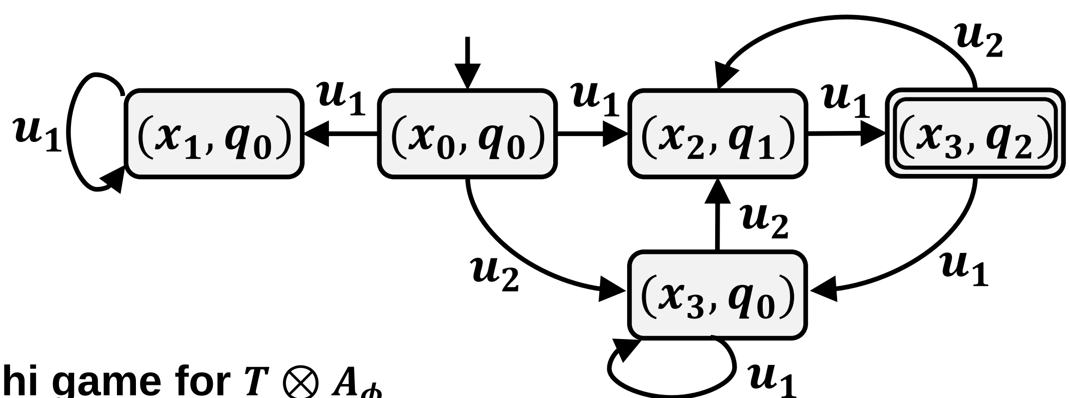
LTL Synthesis: Deterministic Case



LTS T



DBA for dLTL $\phi = \Box(\Diamond o_1 \wedge \Diamond o_2)$



Büchi game for $T \otimes A_\phi$

LTL Synthesis: General Case

- Suppose we have an LTS T and an LTL formula ϕ
- We want to find a controller C such that $C/T \vDash \phi$
- We first build DRA A_ϕ such that $\mathcal{L}^\omega(A_\phi) = Word(\phi)$
- Then we build $T \otimes A_\phi$ with $Acc_\otimes = \{(X \times L_1, X \times K_1), \ldots, (X \times L_n, X \times K_n)\}$
- Solve the Rabin game for $T \otimes A_\phi$ with $Acc_\otimes$
- The winning strategy in $T \otimes A_\phi$ can be mapped directly to T by looking at the first component

➢ Note: we cannot use NBA for ϕ because the LTS T may be non-deterministic; otherwise, we cannot really control the system

Rabin Automata

- In many problems we do need deterministic mechanism, but the expressiveness of DFA is limited
- Using different accepting condition: Rabin acceptance

A Deterministic Rabin Automata (DRA) is a tuple

$$A = (Q, q_0, \delta, \Sigma, Acc)$$

- Q is a finite set of states, $q_0 \in Q$ is the initial state, Σ is the alphabet
- $\delta: Q \times \Sigma \rightarrow Q$ is a partial deterministic transition function
- $Acc = \{(L_1, K_1), \ldots, (L_n, K_n)\} \subseteq 2^Q \times 2^Q$ is the acceptance condition.

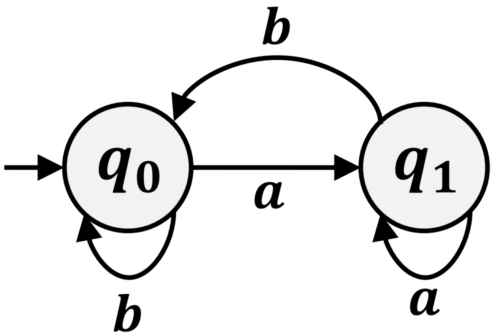
- A run $\rho = q_0 q_1 q_2 \ldots$ is accepting if there exists a pair $(L, K) \in Acc$ s.t.

$$[Inf(\rho) \cap L = \emptyset] \wedge [Inf(\rho) \cap K \neq \emptyset]$$

- Accepted language of DBA A is

$$\mathcal{L}^\omega(A) = \{w \in \Sigma^\omega: \text{the run induced by } w \text{ is accepting in } A\}$$

Rabin Automata



- DRA for "$\Diamond\Box a$"
- $Acc = \{(\{q_0\}, \{q_1\})\}$
- Looping between q_0 and q_1 is rejected

- The class of languages accepted by DRA is the same as that of NBA
- For any LTL formula ϕ over AP, there exists a DRA A_ϕ with alphabet $\Sigma = 2^{AP}$ such that $\mathcal{L}^\omega(A_\phi) = Word(\phi)$
- Cost: we may need $2^{2^{|\phi|\cdot\log|\phi|}}$ states and $2^{|\phi|}$ pairs
- Tools: ltl2dstar https://www.ltl2dstar.de/

Rabin Game

Rabin Game

For LTS T and a set of accepting pairs $Acc = \{(L_1, K_1), \dots, (L_n, K_n)\} \subseteq 2^X \times 2^X$, find a controller C such that for any adversary A there exists a pair (L_i, K_i) such that the run visits K_i infinite times and L_i only finite times.

General Idea:

- **For each pair (L_i, K_i), consider a Büchi Game for K_i + Safety Game for L_i**
- **Then we get $K_i' \subseteq K_i$ that can be visited infinitely often without visiting L_i**
- **Then we consider a reachability game for $\cup_{i=1,\dots,n} K_i'$**
- **The winning region is actually $\mathbf{Attr}(\cup_{i=1,\dots,n} K_i')$**

Stage Summary

- Control problem can be viewed as a two-player game
- Safety game can be solved by inductively extending the unsafe region
- Reachability game can be solved by using n-step attractor
- Büchi game can be solved by identifying recurrent accepting states
- Rabin game can be solved by combing safety, reachability and Büchi
- LTL control synthesis can be solved as a game over the product
- General LTL needs to solve Rabin game
- dLTL can be solved by Büchi game
- scLTL can be solved by reachability game

Course Summary

- How to describe dynamic systems using formal models
  - labeled transition systems
  - bisimulation and quotient-based abstraction

- How to describe formal specifications/requirements
  - linear-time properties
  - linear-temporal logics, computation tree logics

- How to formally verify whether a model satisfies a specification
  - automata-based LTL model checking
  - finite-state automata, Büchi automata, Rabin automata

- How to synthesize a reactive controller to enforce a specification
  - game-based LTL controller synthesis
  - safety game, reachability game, Büchi game, Rabin game

Advanced Topics

- **Timed & hybrid dynamic systems**
- **Formal abstraction of continuous dynamic systems**
- **Stochastic systems and probabilistic verification/synthesis**
- **Real-valued & real-time logics, e.g., MTL and STL**
- **Information-flow analysis or hyper-properties**
- **Control synthesis under imperfect information**
- **Verification & synthesis for multi-agent systems**
- **Temporal-logic-guided learning**

Thank You!

yinxiang@sjtu.edu.cn
http://xiangyin.sjtu.edu.cn