Caltech

Lecture 7
Synthesis of Reactive Control Protocols

Richard M. Murray
Caltech

Ufuk Topcu
UT Austin

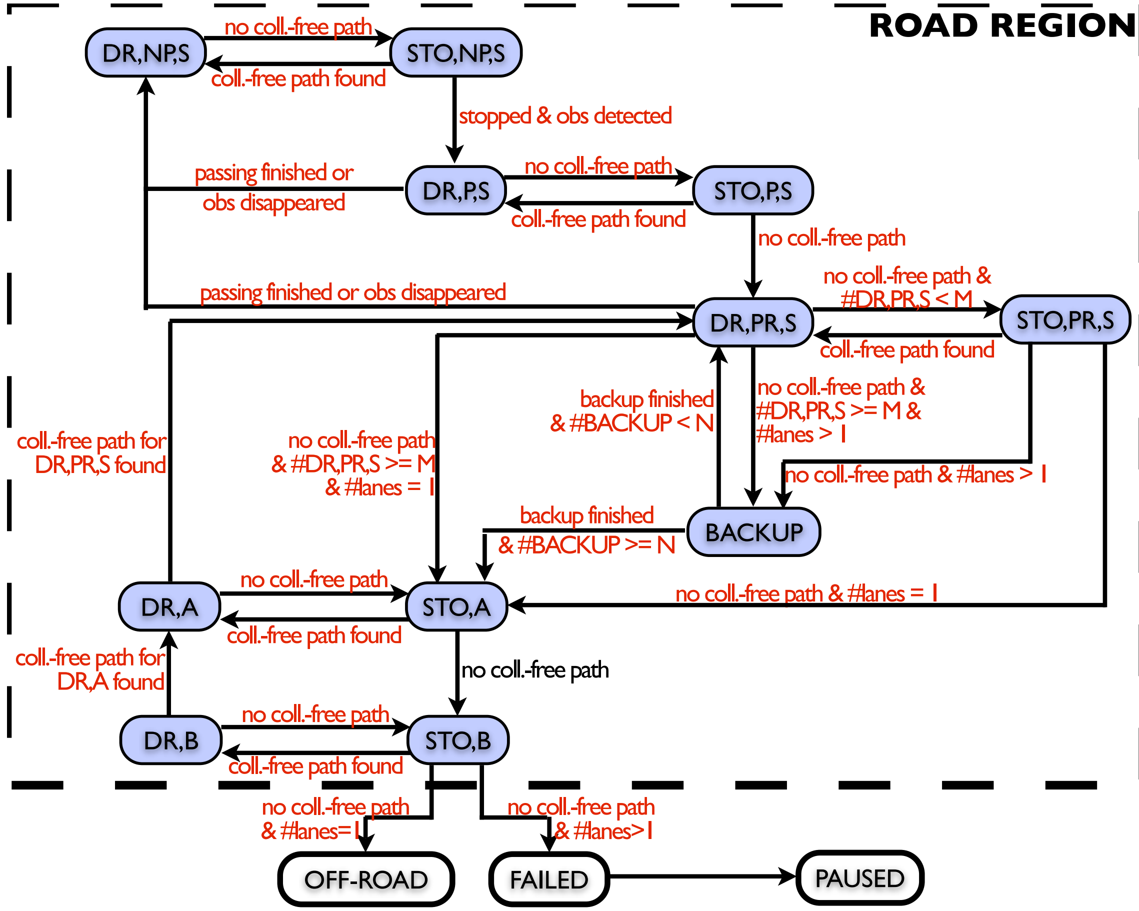
Nok Wongpiromsarn
UT Austin/Iowa State

EECI-IGSC, 11 Mar 2020

Outline

- Open System Synthesis: definition of open systems and open system synthesis problem
- Reactive System Synthesis: problem statement, realizability, games, solving games, complexity
- General Reactivity(1) Games

Alice's Logic Planner

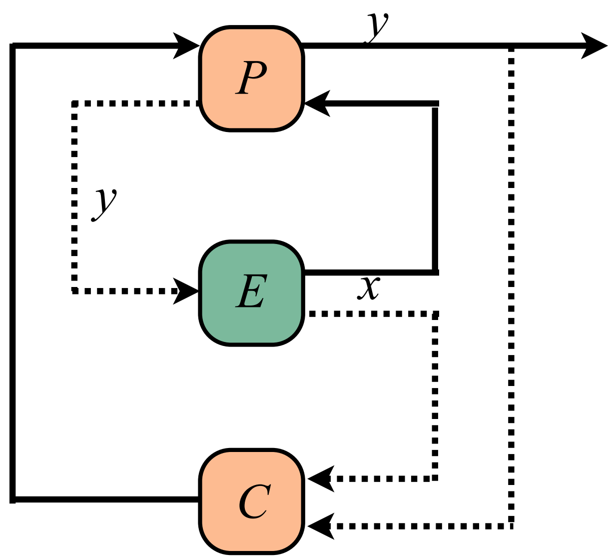


Given a specification Φ, whether the planner is correct with respect to Φ depends on the environment's actions (e.g., how obstacles move)

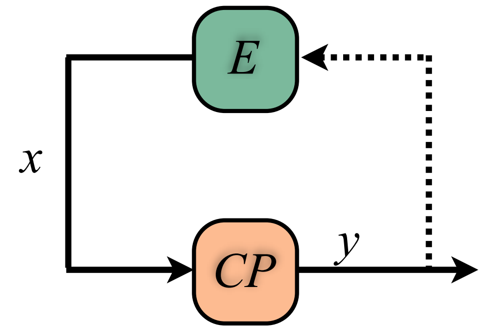
- a "correct" planner needs to ensure that Φ is satisfied for all the possible valid behaviors of the environment

How to design such a correct planner?

Open System Synthesis



An *open system* is a system whose behaviors can be affected by external influence



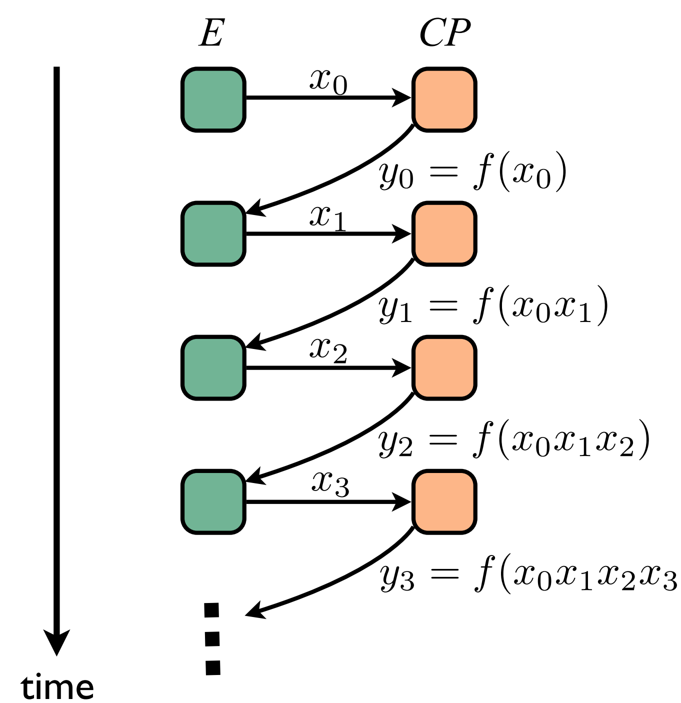
Open (synchronous) synthesis:

Given

- a system that describes all the possible actions
  - plant actions y are controllable
  - environment actions x are uncontrollable
- a specification $\Phi(x, y)$

find a strategy $f(x)$ for the controllable actions which will maintain the specification against all possible adversary moves, i.e.,

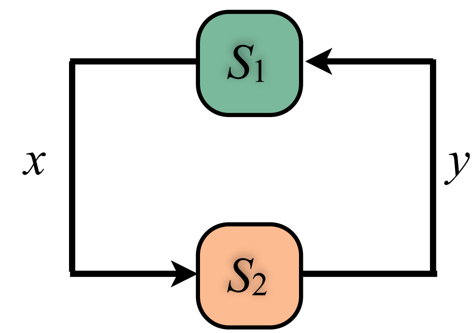
$$\forall x \cdot \Phi(x, f(x))$$



Reactive System Synthesis

Reactive systems are open systems that maintain an ongoing interaction with their environment rather than producing an output on termination.

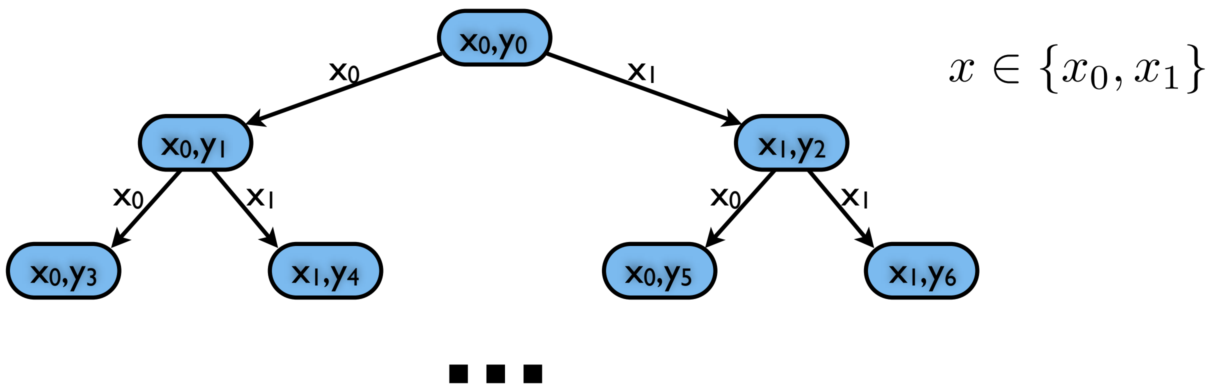
Consider the synthesis of a reactive system with input x and output y, specified by the linear temporal formula $\Phi(x, y)$.



- The system contains 2 components S_1 (i.e., "environment") and S_2 (i.e., "reactive module")
  - Only S_1 can modify x
  - Only S_2 can modify y
- Want to show that S_2 has a *winning* strategy for y against all possible x scenarios the environment may present to it.
  - Two-person game: treat environment as adversary
    - S_2 does its best, by manipulating y, to maintain $\Phi(x, y)$
    - S_1 does its best, by manipulating x, to falsify $\Phi(x, y)$
- If a winning strategy for S_2 exists, we say that $\Phi(x, y)$ is *realizable*

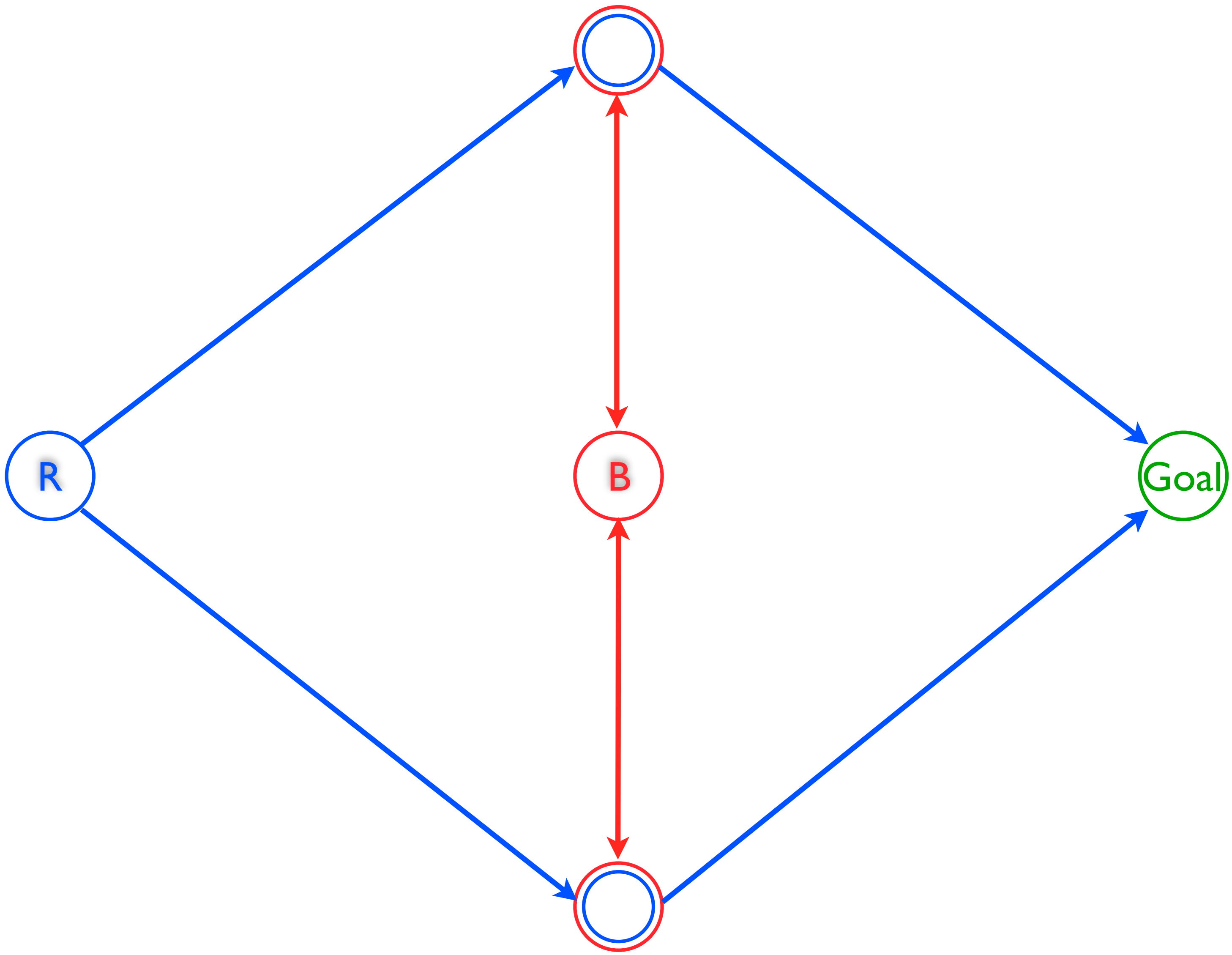
Satisfiability ≠ Realizability

- Realizability should guarantee the specification against all possible (including adversarial) environment (Rosner 98)
  - To solve the problem one must find a satisfying tree where the branching represents all possible inputs



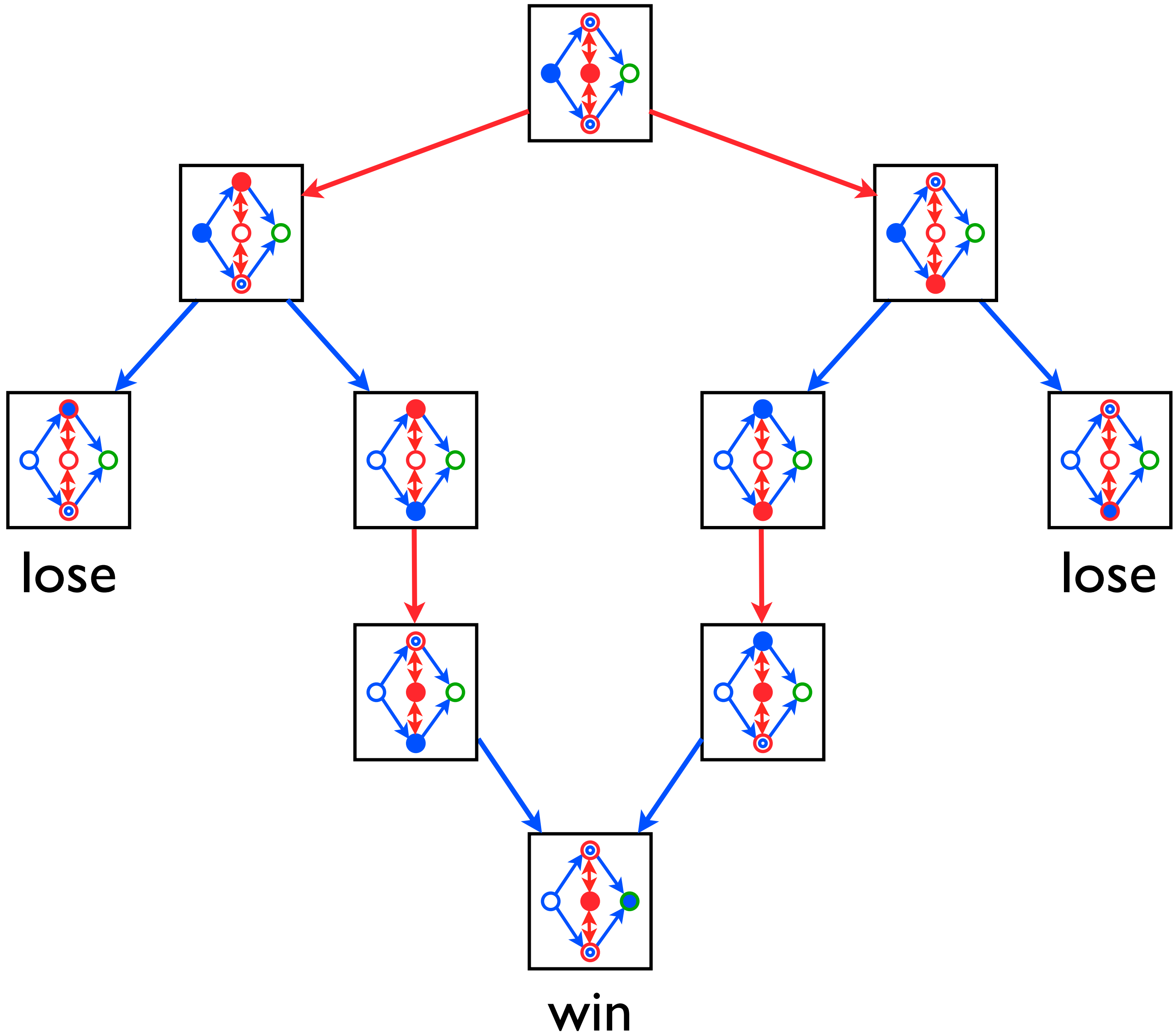
- *Satisfiability* of $\Phi(x, y)$ only ensures that there exists at least one behavior, listing the running values of x and y that satisfies $\Phi(x, y)$
  - There is a way for the plant and the environment to cooperate to achieve $\Phi(x, y)$
- Existence of a winning strategy for S_2 can be expressed by the AE-formula $\forall x \exists y \cdot \Phi(x, y)$

The Runner Blocker System



Runner R tries to reach Goal. Blocker B tries to intercept and stop R.

The Runner Blocker System



Solving Reactive System Synthesis

- Solution is typically given as the winning set
  - The winning set is the set of states starting from which there exists a strategy for S_2 to satisfy the specification for all the possible behaviors of S_1
  - A winning strategy can then be constructed by saving intermediate values in the winning set computation
- Worst case complexity is **double exponential**
  - Construct a nondeterministic Buchi automaton from $\Phi(x, y)$ ⇒ first exponent
  - Determinize Buchi automaton into a deterministic Rabin automaton ⇒ second exponent
  - Follow a similar procedure as in closed system synthesis and construct the product of the system and the deterministic Rabin automaton
  - Find the set of states starting from which all the possible runs in the product automaton are accepting ⇒ This set can be obtained by computing the *recurrent* and the *attractor* sets
- **Special Cases of Lower Complexity**
  - For a specification of the form $\Box p, \Diamond p, \Box\Diamond p$ or $\Diamond\Box p$, the controller can be synthesized in $O(N^2)$ time where N is the size of the state space
  - Avoid translation of the formula to an automaton and determinization of the automaton

Special Case: Satisfiability

- Transition system $TS = (S, \mathrm{Act}, \rightarrow, I, \mathrm{AP}, L)$
- Specification $\Phi = \Diamond p$
- Define the set $WIN \triangleq \{s \in S : s \models p\}$
- Define the predecessor operator $Pre_{\exists} : 2^S \rightarrow 2^S$ by

$$Pre_{\exists}(R) = \{s \in S : \exists r \in R \text{ s.t. } s \rightarrow r\}$$

- The set of all the states starting from which WIN is satisfiable (if the plant and the environment to cooperate) can be computed efficiently by the iteration sequence

$$\begin{aligned} R_0 &= WIN \\ R_i &= R_{i-1} \cup Pre_{\exists}(R_{i-1}), \forall i > 0 \end{aligned}$$

From **Tarski-Knaster Theorem:**

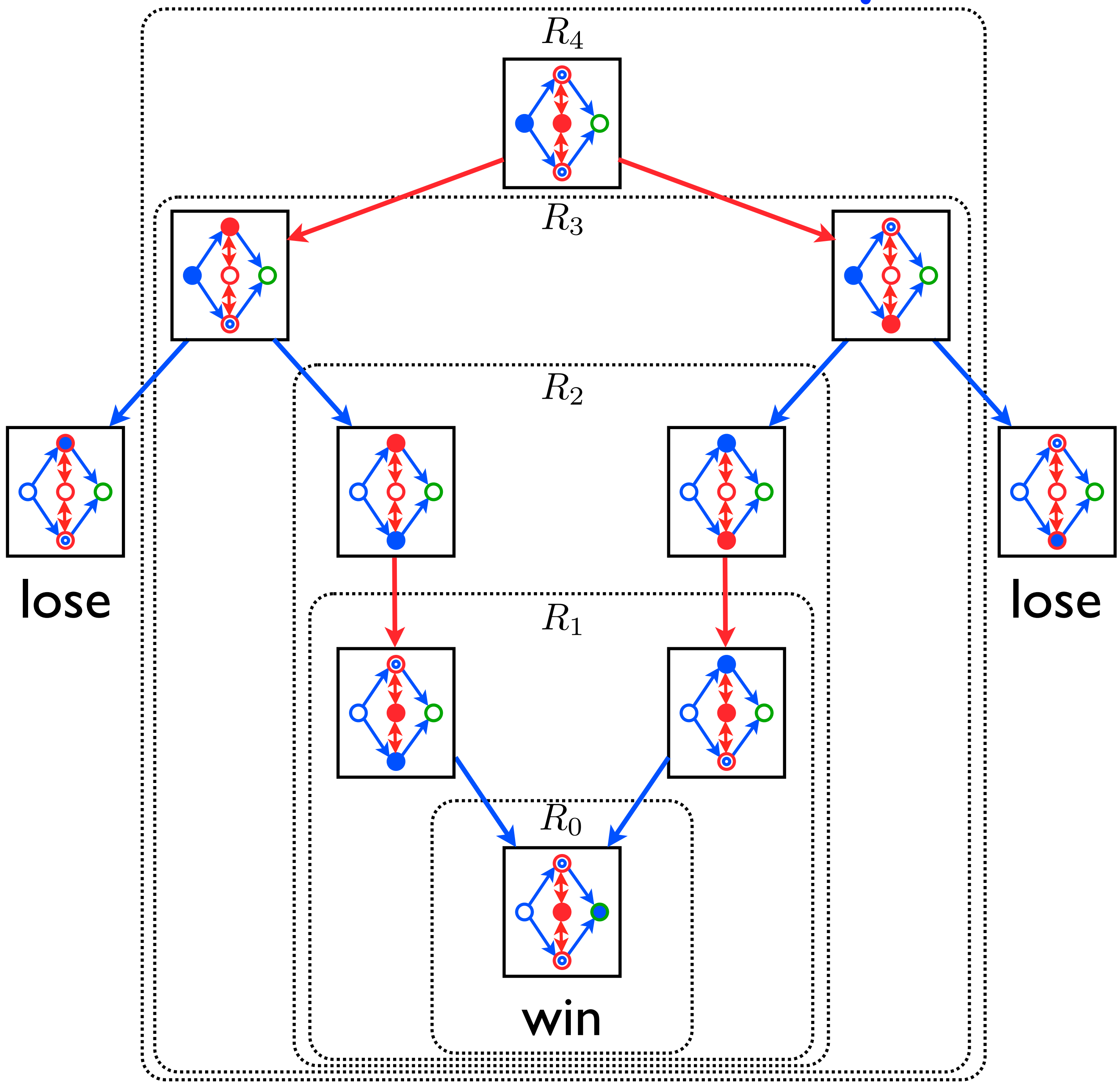
- There exists a natural number n such that $R_n = R_{n\text{-}1}$
- Such an R_n is the minimal solution of the fix-point equation

$$R = WIN \cup Pre_{\exists}(R)$$

- The minimal solution of the above fix-point equation is denoted by

$$\mu R.(WIN \cup Pre_{\exists}(R))$$

The Runner Blocker System



Reachability in Adversarial Setting

- Transition system $TS = (S, \mathrm{Act}, \rightarrow, I, \mathrm{AP}, L)$
- Specification $\Phi = \Diamond p$
- Define the set $WIN \triangleq \{s \in S : s \models p\}$
- Define the operator $Pre_{\forall} : 2^S \rightarrow 2^S$ and $Pre_{\exists\forall} : 2^S \rightarrow 2^S$ by

$$
\begin{aligned}
Pre_{\forall}(R) &= \{s \in S : \forall r \in S \text{ if } s \rightarrow r, \text{then } r \in R\} \\
&= \text{the set of states whose all successors are in } R \\
Pre_{\forall\exists}(R) &= Pre_{\forall}(Pre_{\exists}(R)) \\
&= \text{the set of states whose all successors} \\
&\quad\ \text{have at least one successor in } R
\end{aligned}
$$

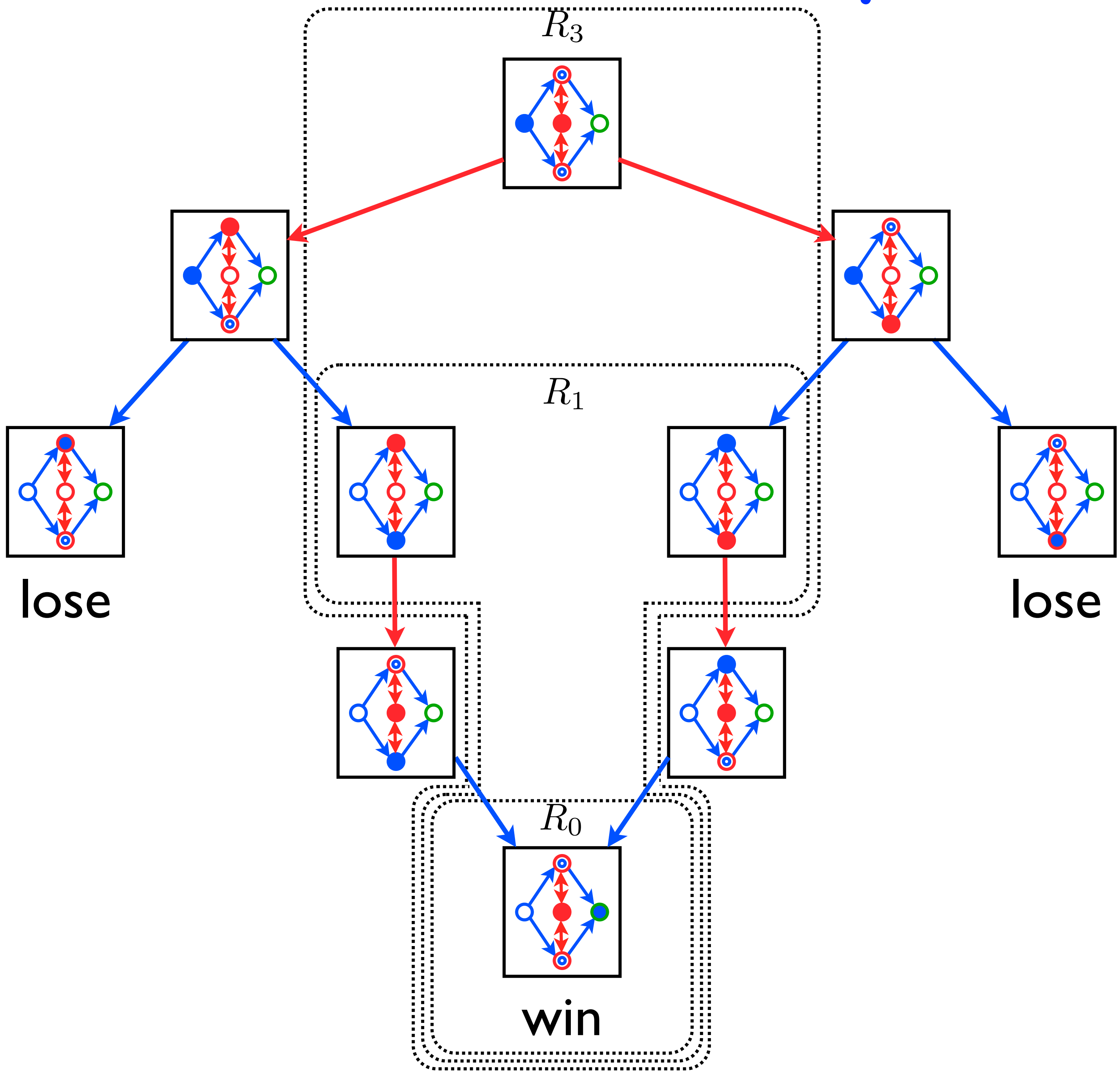
- The set of all the states starting from which the controller can force the system into WIN can be computed efficiently by the iteration sequence

$$
\begin{aligned}
R_0 &= WIN \\
R_i &= R_{i-1} \cup Pre_{\forall\exists}(R_{i-1}), \forall i > 0
\end{aligned}
$$

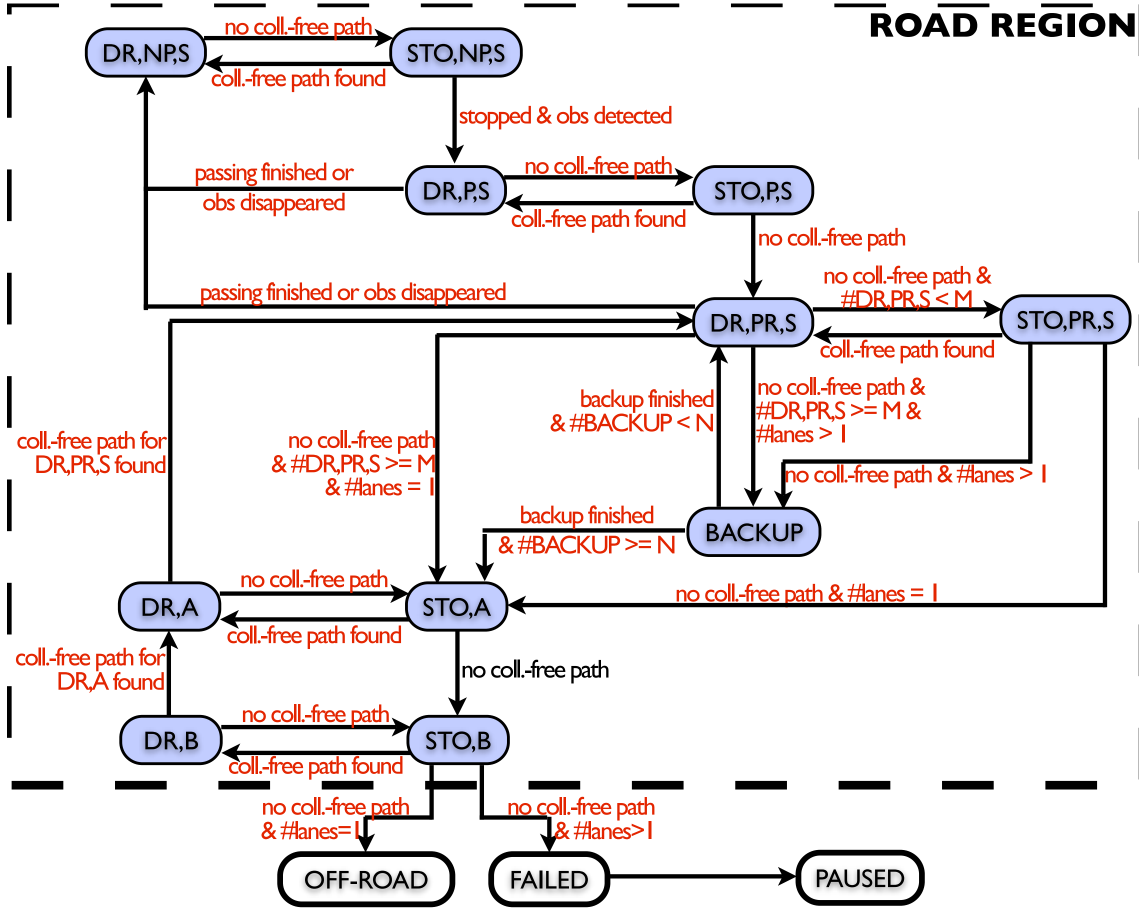
  - There exists a natural number n such that $R_n = R_{n\text{-}1}$
  - Such R_n is the minimal solution of the fix-point equation $R = WIN \cup Pre_{\forall\exists}(R)$
  - The minimal solution of the above fix-point equation is denoted by

$$\mu R.(WIN \cup Pre_{\forall\exists}(R))$$

The Runner Blocker System



More Complicated Case



Game Automata Approach

- Consider the specification as the winning condition in an infinite two-person game between input player (S_1) and output player (S_2).
- Decide whether player S_2 has a *winning* strategy, and if this is the case construct a finite state winning strategy.

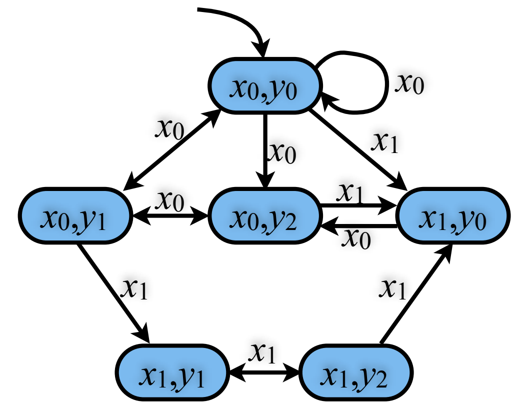
Game Structures

A game structure is a tuple $G = (\mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, AP, L, \varphi)$

- $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a finite set of state variables. $\Sigma_{\mathcal{V}}$ is the set of all the possible assignments to variables in $\mathcal{V}$
- $\mathcal{X} \subseteq \mathcal{V}$ is a set of input variables
- $\mathcal{Y} = \mathcal{V} \setminus \mathcal{X}$ is a set of output variables
- $\theta_e(\mathcal{X})$ is a proposition characterizing the initial states of the environment
- $\theta_s(\mathcal{V})$ is a proposition characterizing the initial states of the system

primed copy of $\mathcal{X}$ represents the set of *next* input variables

- $\rho_e(\mathcal{V}, \mathcal{X}')$ is a proposition characterizing the transition relation of the environment
- $\rho_s(\mathcal{V}, \mathcal{X}', \mathcal{Y}')$ is a proposition characterizing the transition relation of the system
- AP is a set of atomic propositions
- $L : \Sigma_{\mathcal{V}} \to 2^{\mathcal{AP}}$ is a labeling function
- φ is an LTL formula characterizing the winning condition



$\mathcal{V} = \{x, y\}$,
$\mathcal{X} = \{x\}, \Sigma_{\mathcal{X}} = \{x_0, x_1\}$,
$\mathcal{Y} = \{y\}, \Sigma_{\mathcal{Y}} = \{y_0, y_1, y_2\}$,
$x_0 \models \theta_e, x_1 \not\models \theta_e$,
$(x_0, y_0) \models \theta_s$,
$(x_i, y_j) \not\models \theta_s, \forall i, j \neq 0$,
$((x_0, y_i), x_j) \models \rho_e, \forall i, j$,
$((x_1, y_0), x_0) \models \rho_e$,
$((x_1, y_0), x_1) \not\models \rho_e$,
$((x_1, y_i), x_0) \not\models \rho_e, \forall i \in \{1, 2\}$,
$((x_1, y_i), x_1) \models \rho_e, \forall i \in \{1, 2\}$,
$((x_0, y_0), x_0, y_i) \models \rho_s, \forall i$,
$((x_0, y_0), x_1, y_0) \models \rho_s$,
$((x_0, y_0), x_1, y_i) \not\models \rho_s, \forall i \neq 0$,
$\ldots$

Autonomous Car Example

Game Structure $G = (\mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, AP, L, \varphi)$

- $\mathcal{X}$ (environment): obstacles, other cars, pedestrians
- $\mathcal{Y}$ (plant): vehicle state (drive VS stop, passing?, reversing?, etc)
- θ_e describes the valid initial states of the environment, e.g., where obstacles can be
- θ_s describes the valid initial states of the vehicle, e.g., the stop state
- ρ_e describes how obstacles may move
- ρ_s describes the valid transitions of the vehicle state
- φ describes the winning condition, e.g., vehicle does not get stuck

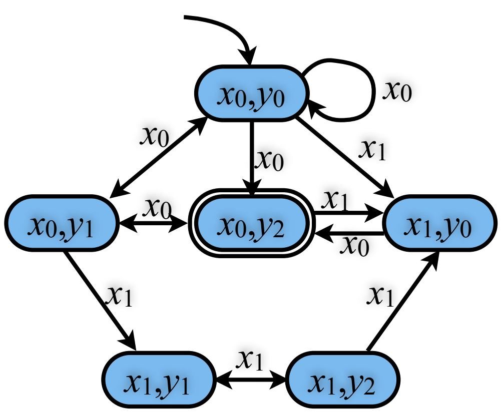
Plays

Game structure $G = (\mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, AP, L, \varphi)$

infinite or the last state in the sequence has no valid successor

- A *play* of G is a maximal sequence of states $\sigma = s_0 s_1 \ldots$ satisfying $s_0 \models \theta_e \wedge \theta_s$ and $(s_j, s_{j+1}) \models \rho_e \wedge \rho_s, \forall j \geq 0$.
  - Initially, the environment chooses an assignment $s_\mathcal{X} \in \Sigma_\mathcal{X}$ such that $s_\mathcal{X} \models \theta_e$ and the system chooses an assignment $s_\mathcal{Y} \in \Sigma_\mathcal{Y}$ such that $(s_\mathcal{X}, s_\mathcal{Y}) \models \theta_e \wedge \theta_s$.
  - From a state s_j, the environment chooses an input $s_\mathcal{X} \in \Sigma_\mathcal{X}$ such that $(s_j, s_\mathcal{X}) \models \rho_e$ and the system chooses an output $s_\mathcal{Y} \in \Sigma_\mathcal{Y}$ such that $(s, s_\mathcal{X}, s_\mathcal{Y}) \models \rho_s$.
- A play σ is *winning for the system* if either
  - $\sigma = s_0 s_1 \ldots s_n$ is finite and $(s_n, s_\mathcal{X}) \not\models \rho_e, \forall s_\mathcal{X} \in \Sigma_\mathcal{X}$, or
  - σ is infinite and $\sigma \models \varphi$.

  Otherwise σ is *winning for the environment*.



$$\varphi = \Box\Diamond(x = x_0 \wedge y = y_2)$$

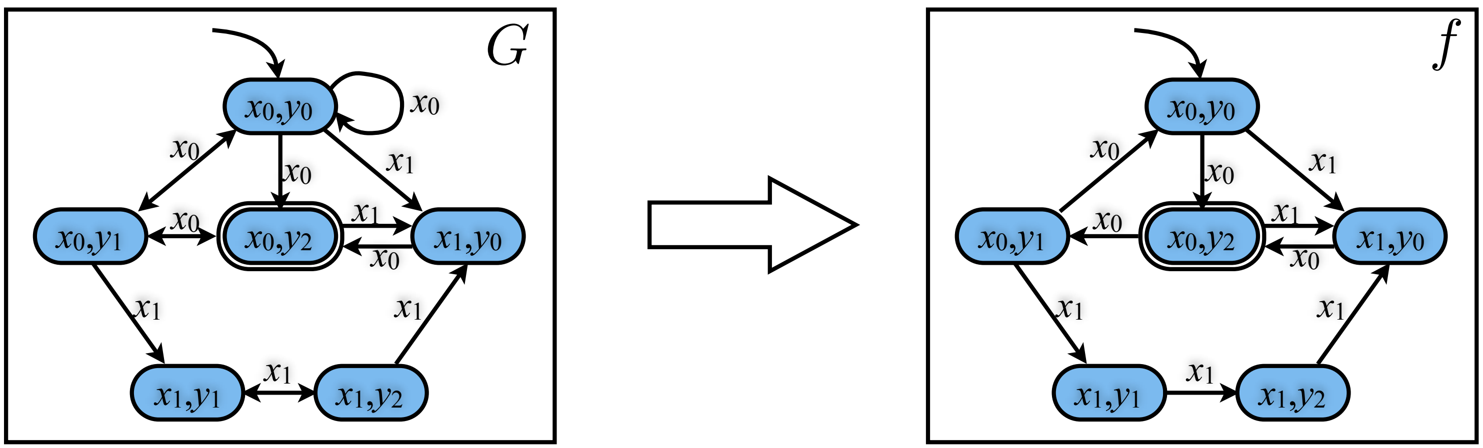
- $\sigma = \big((x_0, y_0), (x_0, y_2), (x_0, y_1)\big)^\omega$ is winning for the system
- $\sigma = \big((x_0, y_0)\big)^\omega$ is winning for the environment

Strategies

Game structure $G = (\mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, AP, L, \varphi)$

memory domain

- A *strategy for the system* is a function $f : M \times \Sigma_{\mathcal{V}} \times \Sigma_{\mathcal{X}} \to M \times \Sigma_{\mathcal{Y}}$ such that for all $s \in \Sigma_{\mathcal{V}}, s_{\mathcal{X}} \in \Sigma_{\mathcal{X}}, m \in M$, if $f(m, s, s_{\mathcal{X}}) = (m', s_{\mathcal{Y}})$ and $(s, s_{\mathcal{X}}) \models \rho_e$, then $(s, s_{\mathcal{X}}, s_{\mathcal{Y}}) \models \rho_s$.
- A play $\sigma = s_0 s_1 \ldots$ is *compliant* with strategy f if $f(m_i, s_i, s_{i+1}|_{\mathcal{X}}) = (m_{i+1}, s_{i+1}|_{\mathcal{Y}}), \forall i$.
- A strategy f is *winning for the system* from state $s \in \Sigma_{\mathcal{V}}$ if all plays that start from s and are compliant with f are winning for the system. If such a winning strategy exists, we call s a *winning state for the system*.

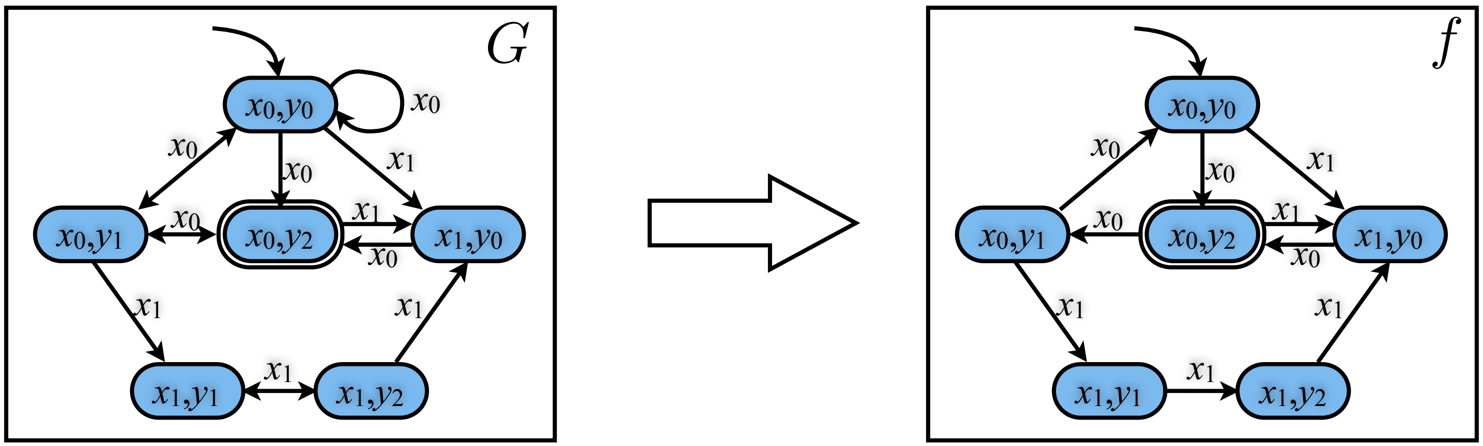


Is f winning for the system?

$$
\begin{aligned}
f(m,(x_0,y_0),x_0) &= (m,y_2) & f(m,(x_0,y_2),x_0) &= (m,y_1) & f(m,(x_1,y_1),x_0) &= (m,y_2)\\
f(m,(x_0,y_0),x_1) &= (m,y_0) & f(m,(x_0,y_2),x_1) &= (m,y_0) & f(m,(x_1,y_1),x_1) &= (m,y_2)\\
f(m,(x_0,y_1),x_0) &= (m,y_0) & f(m,(x_1,y_0),x_0) &= (m,y_2) & f(m,(x_1,y_2),x_0) &= (m,y_2)\\
f(m,(x_0,y_1),x_1) &= (m,y_1) & f(m,(x_1,y_0),x_1) &= (m,y_2) & f(m,(x_1,y_2),x_1) &= (m,y_0)
\end{aligned}
$$

Winning Games

A game structure $G = (\mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, AP, L, \varphi)$ is *winning for the system* if for each $s_\mathcal{X} \in \Sigma_\mathcal{X}$ such that $s_\mathcal{X} \models \theta_e$, there exists $s_\mathcal{Y} \in \Sigma_\mathcal{Y}$ such that $(s_\mathcal{X}, s_\mathcal{Y}) \models \theta_s$ and $(s_\mathcal{X}, s_\mathcal{Y})$ is a winning state for the system



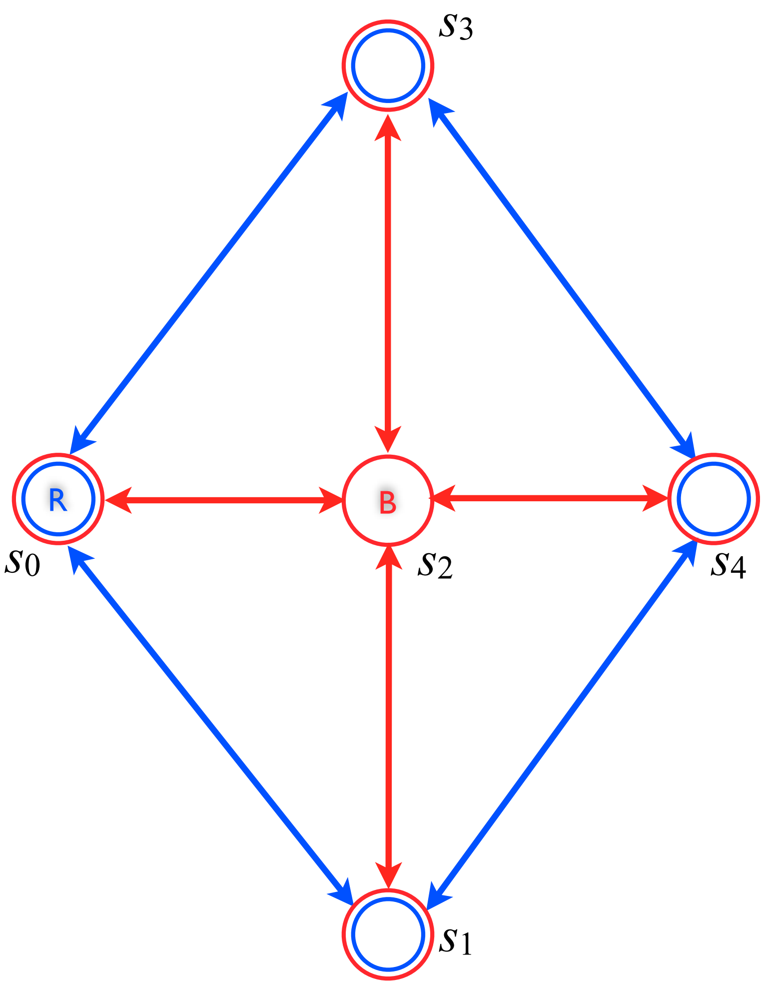
(x_0, y_0) is a winning state for the system

$x_0 \models \theta_e$ but $x_1 \not\models \theta_e$

$(x_0, y_0) \models \theta_s$

G is winning for the system

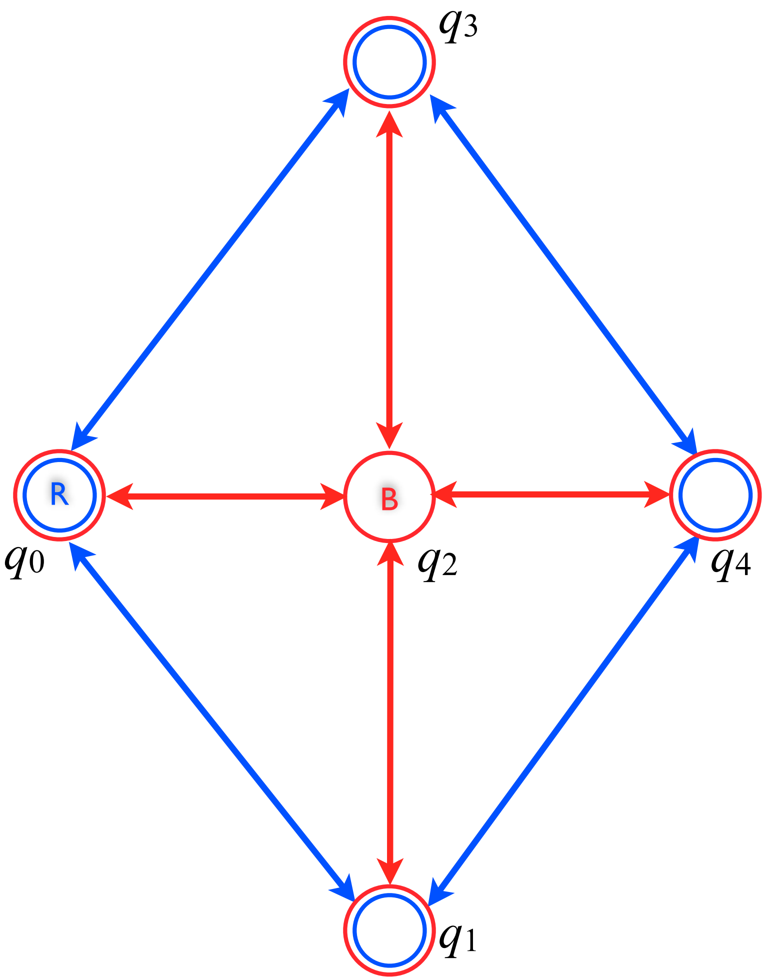
Runner Blocker Example



Game Structure $G = (\mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, AP, L, \varphi)$

- $\mathcal{X} := \{x\}$, $\Sigma_{\mathcal{X}} = \{s_0, s_1, s_2, s_3, s_4\}$
- $\mathcal{Y} := \{y\}$, $\Sigma_{\mathcal{Y}} = \{s_0, s_1, s_3, s_4\}$
- $\theta_e := (x = s_2)$
- $\theta_s := (y = s_0)$
- $\rho_e := \big((x = s_2) \implies (x' \neq s_2)\big) \wedge \big((x \neq s_2) \implies (x' = s_2)\big)$
- $\rho_s := \big((y = s_0 \vee y = s_4) \implies (y' = s_1 \vee y' = s_3)\big) \wedge \big((y = s_1 \vee y = s_3) \implies (y' = s_0 \vee y' = s_4)\big) \wedge (y' \neq x')$
- φ describes the winning condition, e.g., $\diamond(y = s_4)$

Runner Blocker Example



Play: An infinite sequence $\sigma = s_0 s_1 \ldots$ of system (blocker + runner) states such that s_0 is a valid initial state and (s_j, s_{j+1}) satisfies the transition relation of the blocker and the runner

Strategy: A function that gives the next runner state, given a finite number of previous system states of the current play, the current system state and the next blocker state

Winning state: A state starting from which there exists a strategy for the runner to satisfy the winning condition for all the possible behaviors of the blocker

Winning game: For any valid initial blocker state s_x, there exists a valid initial runner state s_y such that (s_x, s_y) is a winning state

Solving game: Identify the set of winning states

Solving Game Structures

General solutions are hard

- Worst case complexity is double exponential (roughly in number of states)

Special cases are easier

- For a specification of the form $\Box p, \Diamond p, \Box\Diamond p$ or $\Diamond\Box p$, the controller can be synthesized in $O(N^2)$ time where N is the size of the state space

Another special case: GR(1) formulas

$$\varphi = \underbrace{(\Box\Diamond p_1 \wedge \ldots \wedge \Box\Diamond p_m)}_{\varphi_e} \implies \underbrace{(\Box\Diamond q_1 \wedge \ldots \wedge \Box\Diamond q_n)}_{\varphi_s}$$

Thm (Piterman, Sa'ar, Pneuli, 2007) A game structure G with a GR(1) winning condition can be solved by a symbolic algorithm in time proportional to $nm|\Sigma_{\mathcal{V}}|^3$

More useful form:

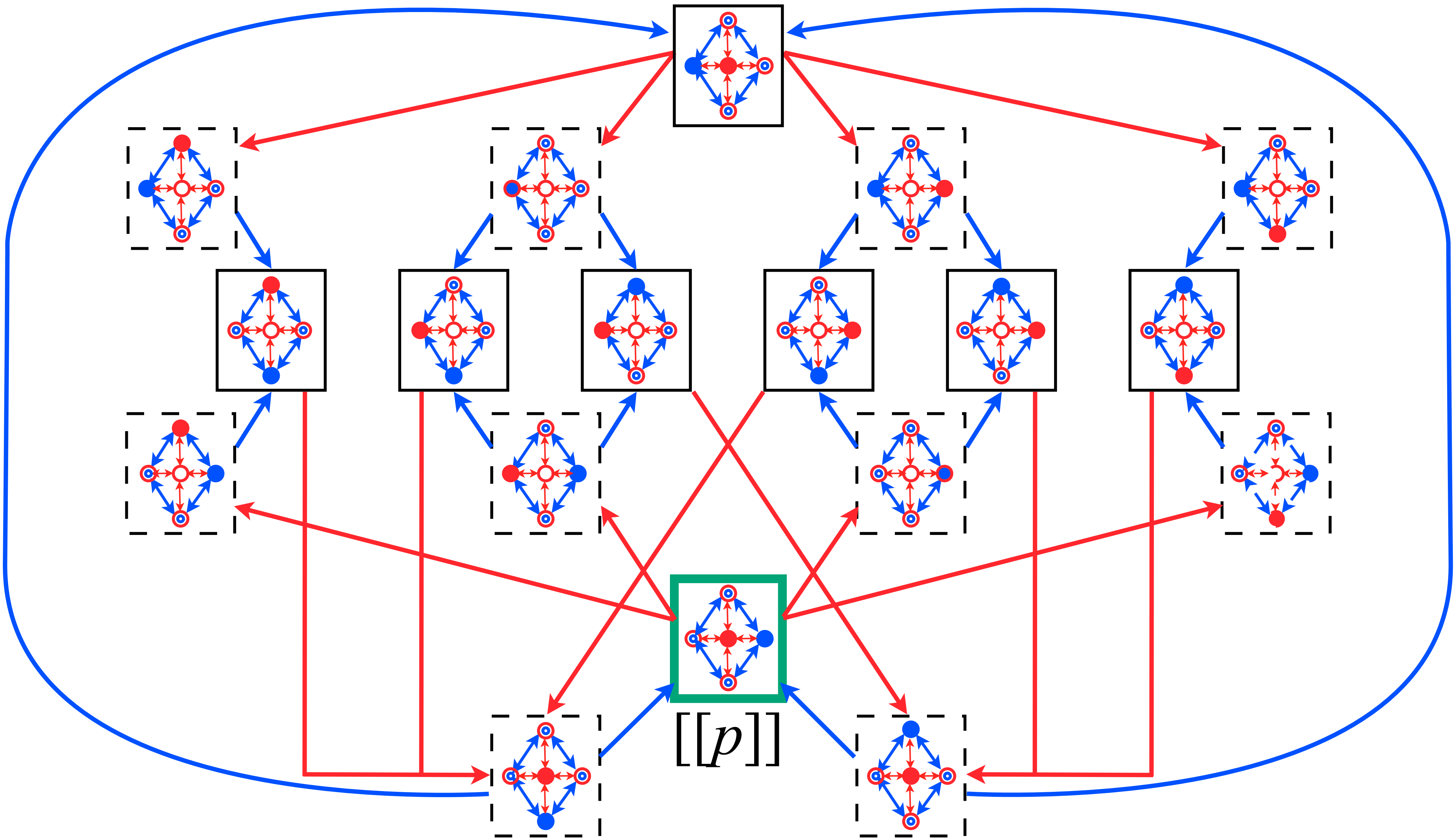
$$\varphi = \big(\underbrace{\psi^e_{init}}_{\substack{\text{assumptions on}\\ \text{initial condition}}} \wedge \underbrace{\Box\psi^e_s \wedge \bigwedge_{i\in I_f} \Box\Diamond\psi^e_{f,i}}_{\substack{\text{assumptions on}\\ \text{environment}}} \big) \implies \underbrace{\big(\psi^s_{init} \wedge \Box\psi^s_s \wedge \bigwedge_{i\in I_g} \Box\Diamond\psi^s_{g,i} \big)}_{\substack{\text{desired}\\ \text{behavior}}}$$

- Can show that this can be "converted" to GR(1) form

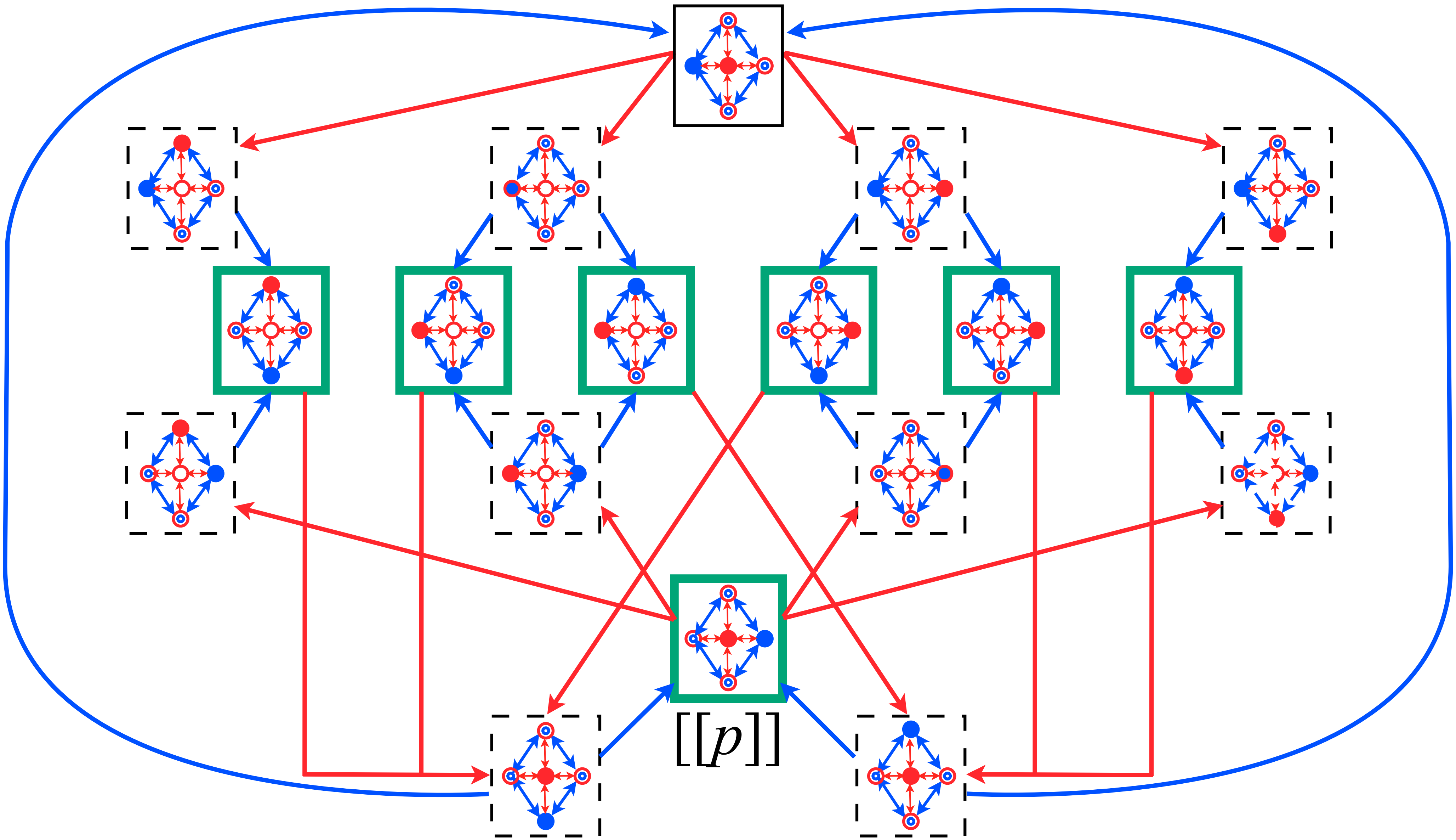
Solving Reachability Games

- Game structure $G = (\mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, AP, L, \varphi)$
- For a proposition p, let
$$[[p]] = \{s \in \Sigma_{\mathcal{V}} \mid s \models p\}$$
- For a set R, let
$$[[\otimes R]] = \left\{s \in \Sigma_{\mathcal{V}} \mid \forall s'_{\mathcal{X}} \in \Sigma_{\mathcal{X}}, (s, s'_{\mathcal{X}}) \models \rho_e \Rightarrow \exists s'_{\mathcal{Y}} \in \Sigma_{\mathcal{Y}} \text{ s.t. } (s, s'_{\mathcal{X}}, s'_{\mathcal{Y}}) \models \rho_s \text{ and } (s'_{\mathcal{X}}, s'_{\mathcal{Y}}) \in R\right\}$$
similar to the $Pre_{\forall\exists}$ operator we saw earlier
- Reachability game: $\varphi = \diamond p$
- The set of winning states can be computed efficiently by the iteration sequence
$$\begin{aligned} R_0 &= \emptyset \\ R_{i+1} &= [[p]] \cup [[\otimes R_i]], \forall i \geq 0 \end{aligned}$$
  - R_{i+1} is the set of states starting from which the system can force the play to reach a state satisfying p within i steps
  - There exists a natural number n such that $R_n = R_{n-1}$
  - Such R_n is the minimal solution of the fix-point equation $R = [[p]] \cup [[\otimes R]]$
  - In μ-calculus, the minimal solution of the above fix-point equation is denoted by $\mu R(p \vee \otimes R)$
least fixpoint

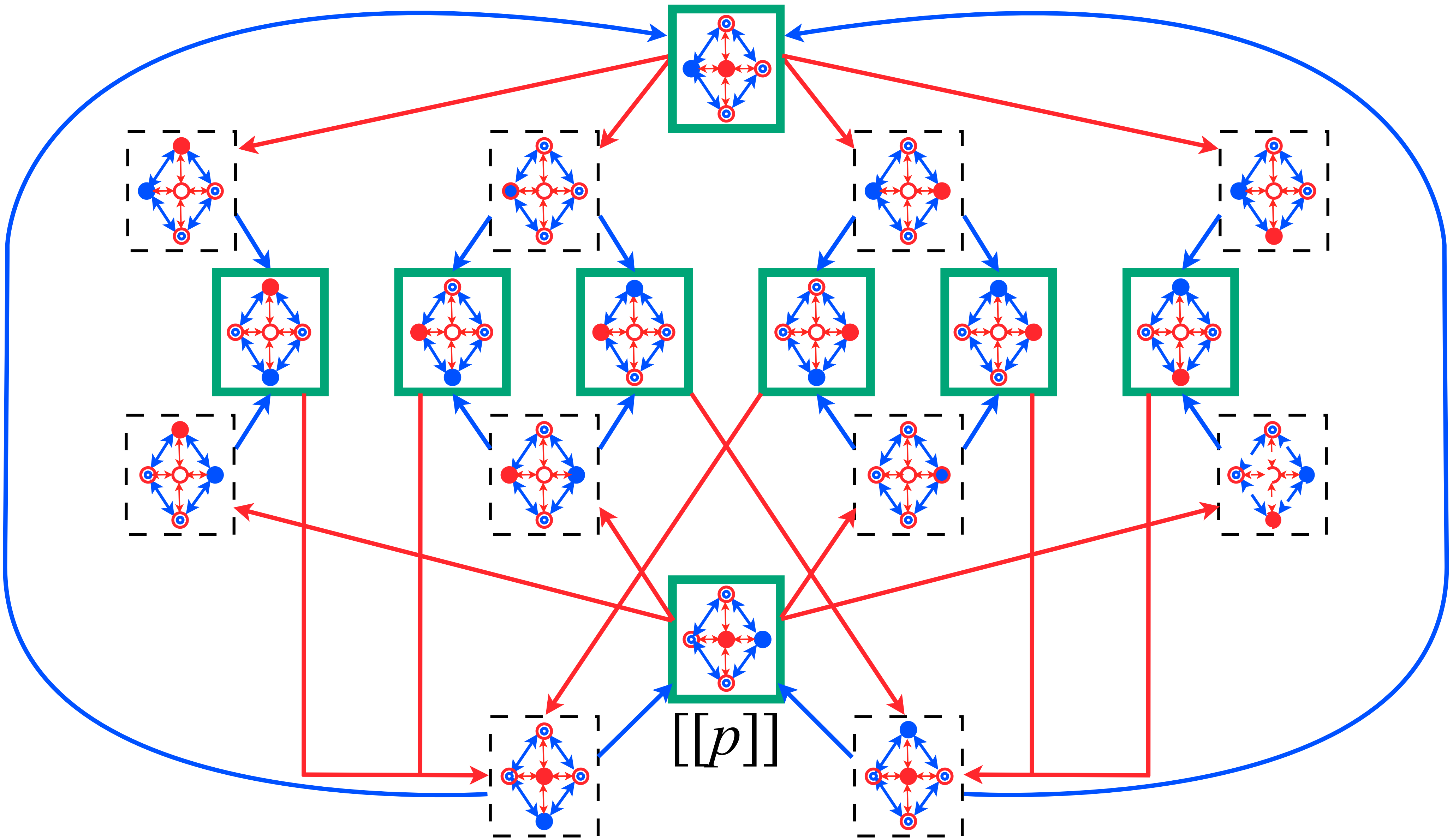
Runner Blocker Example: R_1



Runner Blocker Example: R_2



Runner Blocker Example: $R_3 = R_4 = \ldots$



Solving Safety Games

- Game structure $G = (\mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, AP, L, \varphi)$
- For a proposition p, let

$$[[p]] = \{s \in \Sigma_{\mathcal{V}} \mid s \models p\}$$

- For a set R, let

$$[[\circledast R]] = \left\{ s \in \Sigma_{\mathcal{V}} \mid \forall s'_{\mathcal{X}} \in \Sigma_{\mathcal{X}}, (s, s'_{\mathcal{X}}) \models \rho_e \Rightarrow \exists s'_{\mathcal{Y}} \in \Sigma_{\mathcal{Y}} \text{ s.t. } (s, s'_{\mathcal{X}}, s'_{\mathcal{Y}}) \models \rho_s \text{ and } (s'_{\mathcal{X}}, s'_{\mathcal{Y}}) \in R \right\}$$

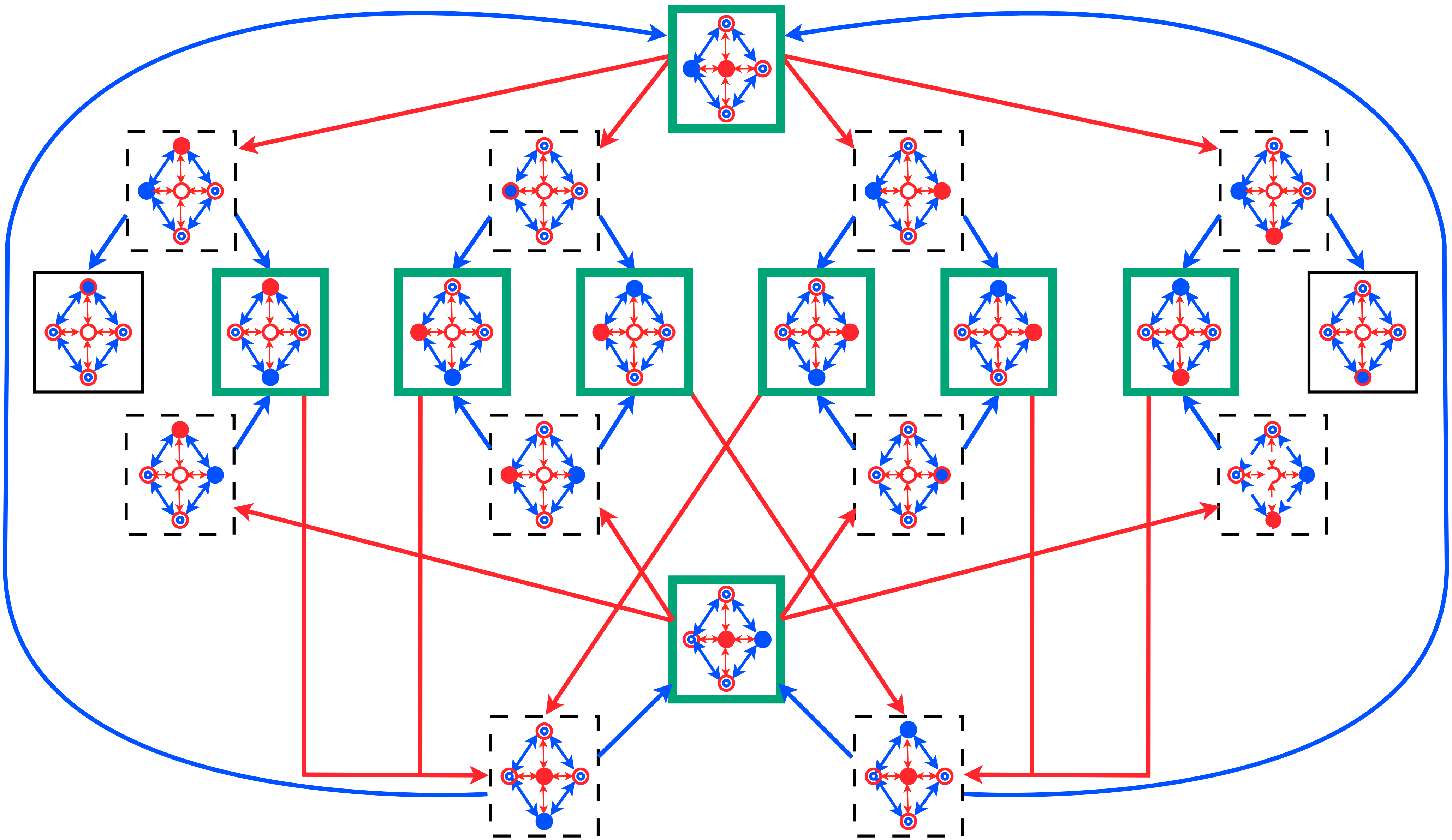
- Safety game: $\varphi = \Box p$
- The set of winning states can be computed efficiently by the iteration sequence

$$\begin{array}{rcl} R_0 & = & \Sigma_{\mathcal{V}} \\ R_{i+1} & = & [[p]] \cap [[\circledast R_i]], \forall i \geq 0 \end{array}$$

  - R_{i+1} is the set of states starting from which the system can force the play to stay in states satisfying p for i steps
  - There exists a natural number n such that $R_n = R_{n-1}$
  - Such R_n is the maximal solution of the fix-point equation $R = [[p]] \cap [[\circledast R]]$
  - In μ-calculus, the minimal solution of the above fix-point equation is denoted by $\nu R(p \wedge \circledast R)$

greatest fixpoint

Runner Blocker Example: $R_1 = R_2 = \ldots$



Solving Games

Game structure $G = (\mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, AP, L, \varphi)$

φ	The set of winning states for the system
$\Diamond p$	$\mu X(p \vee \circledDiamond X)$
$\Box p$	$\nu X(p \wedge \circledDiamond X)$
$\Box\Diamond p$	$\nu X \mu Y\big((p \wedge \circledDiamond X) \vee \circledDiamond Y\big)$

- $\nu X(p \wedge \circledDiamond X)$ is the largest set S of states such that
  - all the states in S satisfy p, and
  - starting from a state in S, the system can force the play to transition to a state in S
- $\nu X \mu Y\big((p \wedge \circledDiamond X) \vee \circledDiamond Y\big)$ is the set of state starting from which the system can force the play to satisfy p infinitely often
  - The disjunction and μY operators ensure that the system is in a state where it can force the play to reach a state satisfying p
  - The conjunction and the νX operators ensure that the above statement is true at all time

Games and Realizability

Game structure $G = (\mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, AP, L, \varphi)$

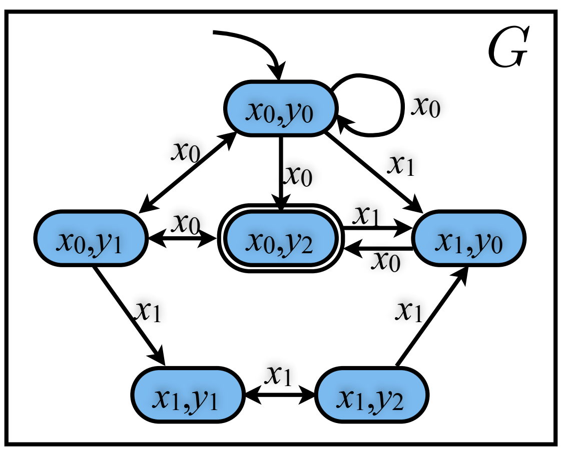
The system wins in G iff the specification

$$\psi = (\theta_e \implies \theta_s) \wedge \big(\theta_e \implies \Box((\boxminus\rho_e) \implies \rho_s)\big) \wedge \big((\theta_e \wedge \Box\rho_e) \implies \varphi\big)$$

is realizable.

Given an LTL specification ψ, we construct G as follows

- θ_e and θ_s include the non-temporal specification parts of ψ
- ρ_e and ρ_s include the local limitations on the next values of variables in $\mathcal{X}$ and $\mathcal{Y}$
- φ includes all the remaining properties in ψ that are not included in θ_e, θ_s, ρ_e and ρ_s



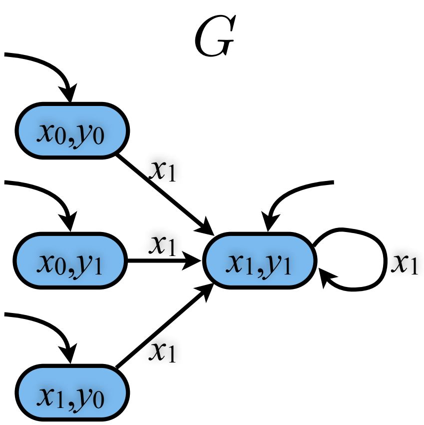
$$\begin{aligned}
X_i &\triangleq (x = x_i),\ \ Y_i \triangleq (y = y_i),\ \ X_i' \triangleq (x' = x_i),\ \ Y_i' \triangleq (y' = y_i) \\
\theta_e &\triangleq X_0,\ \ \theta_s \triangleq Y_0 \\
\rho_e &\triangleq ((X_1 \wedge Y_0) \implies X_0') \wedge ((X_1 \wedge Y_1) \implies X_1') \wedge ((X_1 \wedge Y_2) \implies X_1') \\
\rho_s &\triangleq ((X_0 \wedge Y_0 \wedge X_0') \implies (Y_1' \vee Y_2')) \wedge ((X_0 \wedge Y_0 \wedge X_1') \implies (Y_0')) \wedge \\
&\quad ((X_0 \wedge Y_1 \wedge X_0') \implies (Y_0' \vee Y_2')) \wedge ((X_0 \wedge Y_1 \wedge X_1') \implies (Y_1')) \wedge \\
&\quad ((X_0 \wedge Y_2 \wedge X_0') \implies Y_1') \wedge ((X_0 \wedge Y_2 \wedge X_1') \implies Y_0') \wedge \\
&\quad ((X_1 \wedge Y_0 \wedge X_0') \implies Y_2') \wedge ((X_1 \wedge Y_1 \wedge X_1') \implies Y_2') \wedge \\
&\quad ((X_1 \wedge Y_2 \wedge X_1') \implies (Y_0' \vee Y_1')) \\
\varphi &\triangleq \Box\Diamond(X_0 \wedge Y_2)
\end{aligned}$$

Games and Realizability

More intuitive specification

$$\psi' = (\theta_e \wedge \Box \rho_e \wedge \varphi_e) \implies (\theta_s \wedge \Box \rho_s \wedge \varphi_s)$$

- Fulfillment of the system safety depends on the liveness of the environment
  - The system may violate its safety if it ensures that the environment cannot fulfill its liveness
- ψ implies ψ'
  - If ψ is realizable, a controller for ψ is also a controller for ψ' (but not vice versa)
  - If the system wins in $G = (\mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, AP, L, \varphi_e \implies \varphi_s)$, then ψ' is realizable (but not vice versa)
- By adding extra output variables that represent the memory of whether the system or the environment violate their initial requirements or their safety requirements, we can construct a game G' such that G' is won by the system iff ψ' is realizable



$\mathcal{X} = \{x\}, \Sigma_{\mathcal{X}} = \{x_0, x_1\},$

$\mathcal{Y} = \{y\}, \Sigma_{\mathcal{Y}} = \{y_0, y_1\},$

$\theta_e, \theta_s \triangleq \texttt{true}$

$\rho_e \triangleq (x' = x_1)$

$\rho_s \triangleq ((x' = x_1) \implies (y' = y_1))$

$\varphi_e \triangleq \Box\Diamond((x = x_1) \wedge (y = y_1))$

$\varphi_s \triangleq \Box\Diamond(y = y_0)$

- ψ' is realizable
  - The system always picks $y = y_0$
- ψ is not realizable
- The system does not win in G

General Reactivity(1) Games

GR(1) game is a game $G = (\mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, AP, L, \varphi)$ with the winning condition

$$\varphi = \underbrace{(\Box\Diamond p_1 \wedge \ldots \wedge \Box\Diamond p_m)}_{\varphi_e} \implies \underbrace{(\Box\Diamond q_1 \wedge \ldots \wedge \Box\Diamond q_n)}_{\varphi_s}$$

The winning states in a GR(1) game can be computed using the fixpoint expression

$$\nu \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_n \end{bmatrix} \begin{bmatrix} \mu Y \left(\bigvee_{i=1}^{m} \nu X \big((q_1 \wedge \circledDiamond Z_2) \vee \circledDiamond Y \vee (\neg p_i \wedge \circledDiamond X) \big) \right) \\ \mu Y \left(\bigvee_{i=1}^{m} \nu X \big((q_2 \wedge \circledDiamond Z_3) \vee \circledDiamond Y \vee (\neg p_i \wedge \circledDiamond X) \big) \right) \\ \vdots \\ \mu Y \left(\bigvee_{i=1}^{m} \nu X \big((q_n \wedge \circledDiamond Z_1) \vee \circledDiamond Y \vee (\neg p_i \wedge \circledDiamond X) \big) \right) \end{bmatrix}$$

- $\mu Y \nu X \big(\circledDiamond Y \vee (\neg p_i \wedge \circledDiamond X) \big)$ characterizes the set of states from which the system can force the play to stay indefinitely in $\neg p_i$ states
- The two outer fixpoints make sure that the system wins from the set $q_j \wedge \circledDiamond Z_{j \oplus 1} \vee \circledDiamond Y$
  - The disjunction and μY operators ensure that the system is in a state where it can force the play to reach a $q_j \wedge \circledDiamond Z_{j \oplus 1}$ state in a finite number of steps
  - The conjunction and νZ_j operators ensure that after visiting q_j, we can loop and visit $q_{j \oplus 1}$

Extracting GR(1) Strategies

The intermediate values in the computation of the fixpoint can be used to compute a strategy, represented by a finite transition system, for a GR(1) game.

This strategy does one of the followings

- Iterates over strategies $f_1, \ldots, f_n$ where f_j ensures that the play reaches a q_j state
- Eventually uses a fixed strategy ensuring that the play does not satisfy one of the liveness assumptions p_j

Complexity: A game structure G with a GR(1) winning condition can be solved by a symbolic algorithm in time proportional to $nm|\Sigma_{\mathcal{V}}|^3$

Extensions

The algorithm for solving GR(1) game can be applied to any game with the winning condition of the form

$$\varphi = \underbrace{(\Box\Diamond p_1 \wedge \ldots \wedge \Box\Diamond p_m)}_{\varphi_e} \implies \underbrace{(\Box\Diamond q_1 \wedge \ldots \wedge \Box\Diamond q_n)}_{\varphi_s}$$

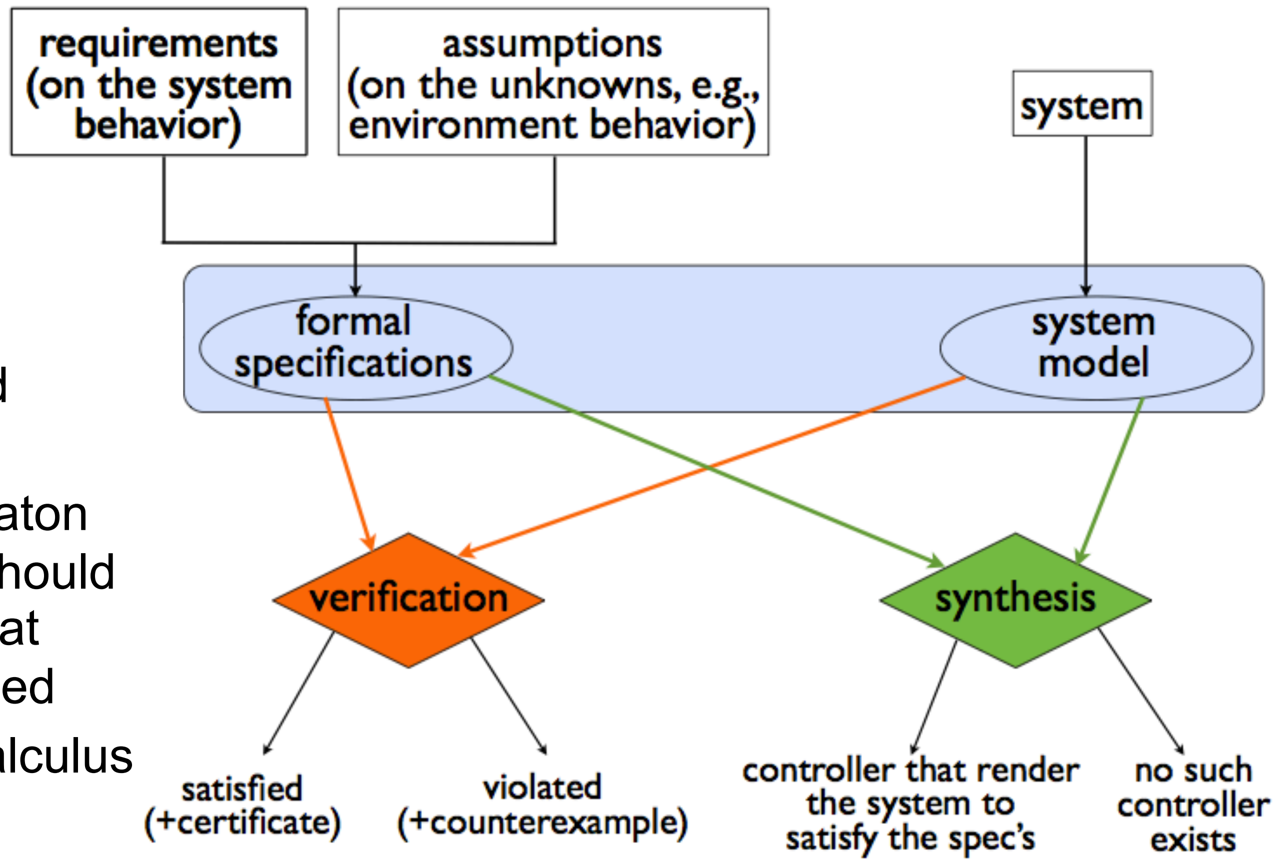
where p_i, q_j are past formulas.

- Add to the game additional variables and a transition relation which encodes the deterministic Buchi automaton
- Examples: $\Box(p \implies \Diamond q)$
  - Introduce a Boolean variable x
  - Initial condition: $x = 1$
  - Transition relation for the environment: $\rho_e \wedge \big(x' = (q \vee x \wedge \neg p)\big)$
  - Winning condition: $\Box \Diamond x$

Summary

Reactive controller synthesis for discrete systems

- System model: discrete transition system, with actions unspecified
- Specification: LTL formula giving desired properties and allow environment actions
- Controller: finite state automaton that describes how system should react to environment such that specification is always satisfied
- Approach: winning sets, μ calculus



Implementation

- In general, synthesis for a general LTL formula can be doubly exponentially complex size of the formula => intractable
- For GR(1) formulas, this reduces to cubic complexity in # of states => *big* reduction

Next steps

- Figure out how to make use of these results for control of hybrid systems