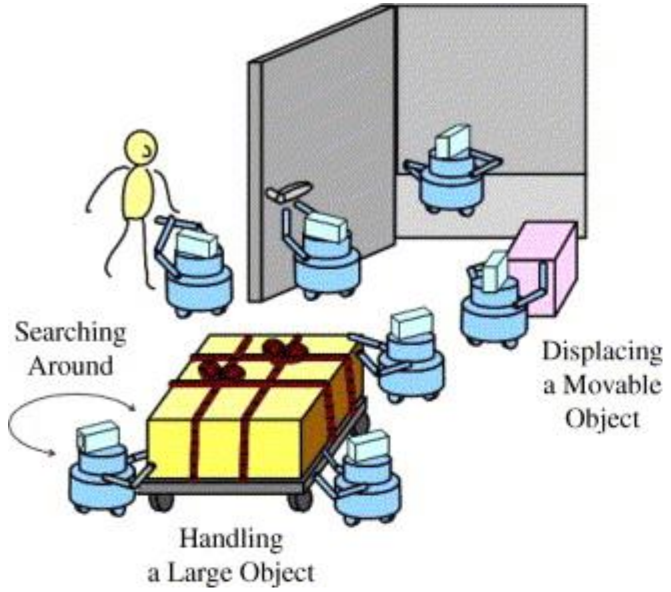
Prism Games 3.0

Stochastic Game Verification with Concurrency and Equilibria

Marco Lavorini
CMCS 24/25

Why Probabilistic Verification on Games?

- Agents operate in a noisy, stochastic environment
- Systems increasingly involve concurrently acting agents
- Game-Theory approaches can be used to reason about the competitive or collaborative behaviour of multiple rational agents



"Program testing can be used to show the presence of bugs, but never to show their absence"

-Dijkstra

What is a Game?

- A game can be formally defined as a tuple

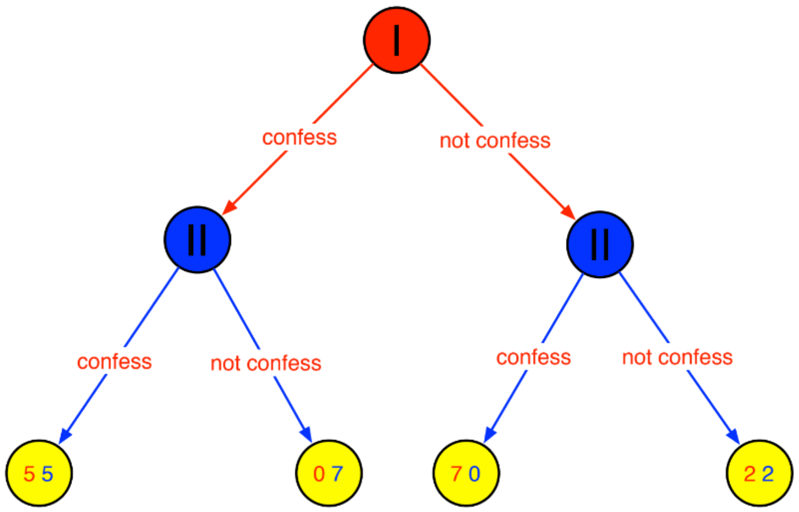
$$\mathbf{G} = \langle N, S, u, R \rangle$$

- $N = \{1,\dots,n\}$ finite set of players
- S_i set of strategies for player $i \in N$, S is the set for all players.
- $u : S \to \mathbb{R}$ payoff function
- R set of possible rules

More about Games

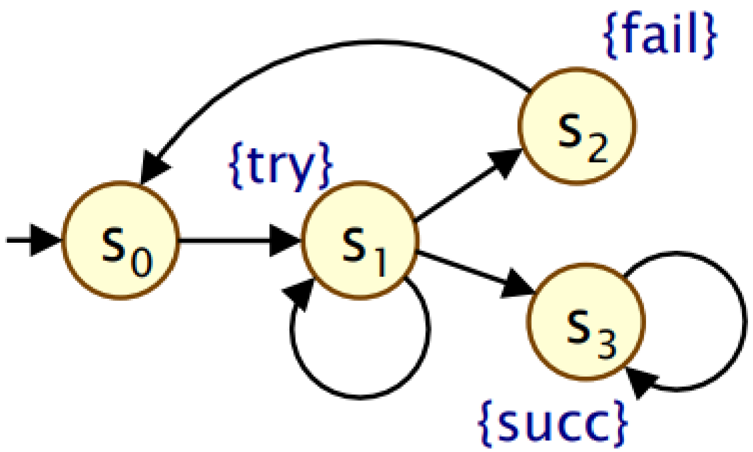
- The same game can be represented in many way, for example a matrix or a graph.
- There are many types of games:
  - pure, mixed, stochastic, …
  - Zero-sum, bi-games, …
- A game can be competitive, collaborative, or even both.

I/II	not confess	confess
not confess	(2,2)	(7,0)
confess	(0,7)	(5,5)

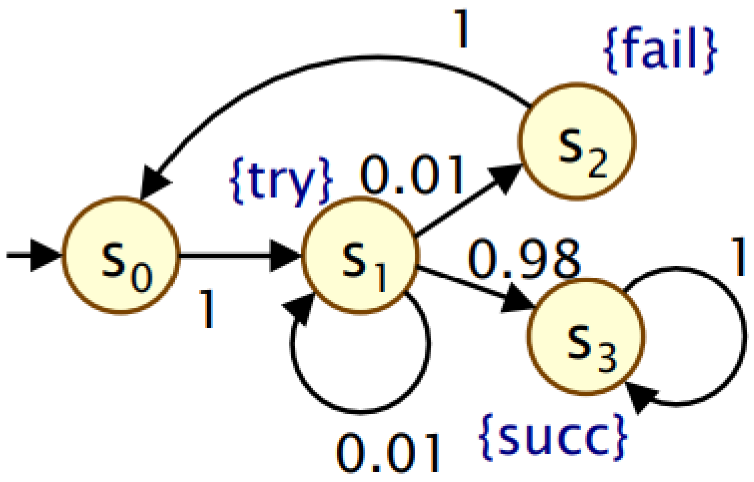


From Simple Models to Games

- Labelled Transition System
  - We use CTL
  - Example:
    - ❖ E [fail U succ]
      there exists a path that reaches a fail state until it first reaches succ

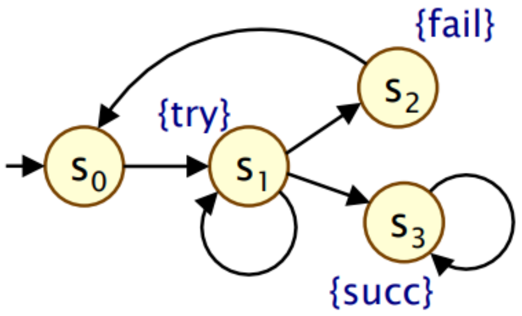


- Discrete-Time Markov Chain
  - We use PCTL
  - Example:
    - ❖ P≥ 0.95 [F succ]
      the probability or reaching succ is at least 95%

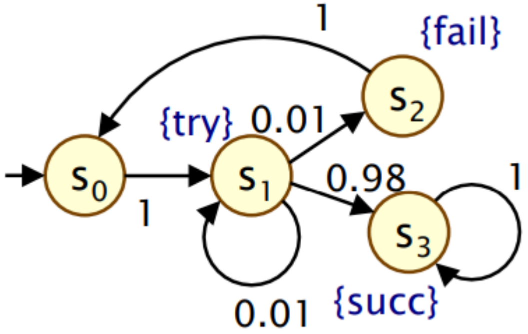


Markov Decision Process

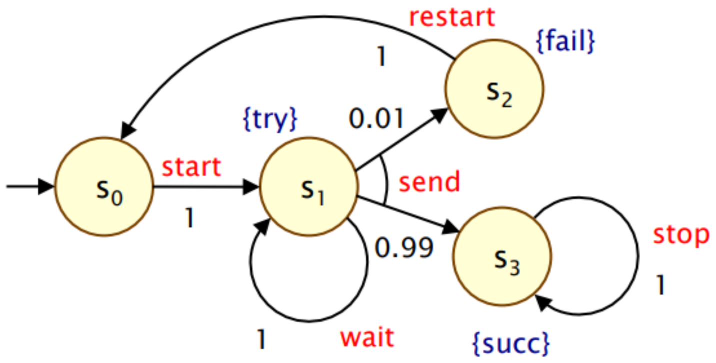
Nondeterministic



Probabilistic



Probabilities and nondeterminism

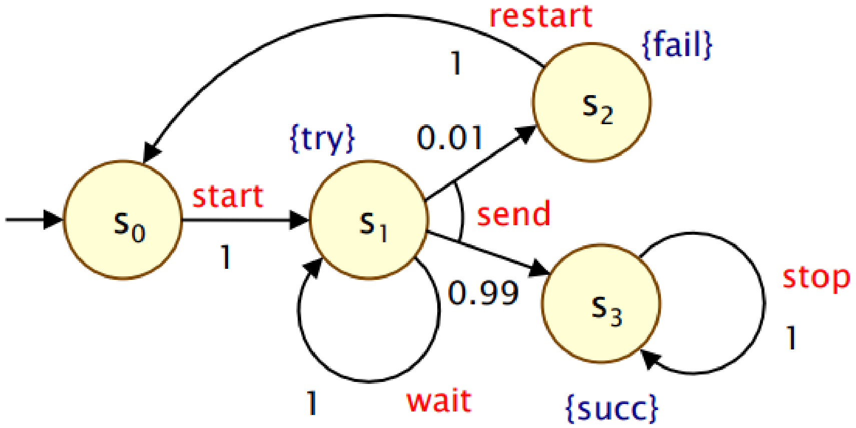


- In each state of the MDP, we have a nondeterministic choice between several discrete probability distributions over the successor states

Example taken from: **Slide 12 of "Lecture 12 – Markov Decision Processes"** from Dave Parker's *Probabilistic Model Checking* course at Oxford.

Strategies in an MDP

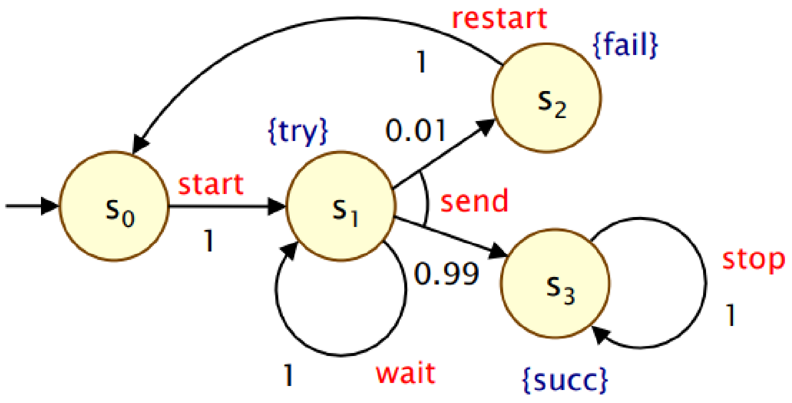
- In any given state we can choose an action
- The choice is still probabilistic
- A series of actions is called a **Strategy σ**
- An **optimal strategy** is a strategy that maximise probability
- Example:
  - ❖ What is the maximum probability of reaching state s_3, achievable by any strategy σ?
  - ❖ $\sup_\sigma Pr_s^\sigma(F\ s_3)$
    "The supremum over all the possible strategies of the probability from a particular state s under that strategy σ"



Value Iteration

- Value Iteration is a dynamic programming approach used for probabilistic model checking

$$p(s) = \begin{cases} 1, & if\ s = goal \\ max_a \sum_{s'} \delta(s,a)(s') \cdot p(s'), & otherwise \end{cases}$$

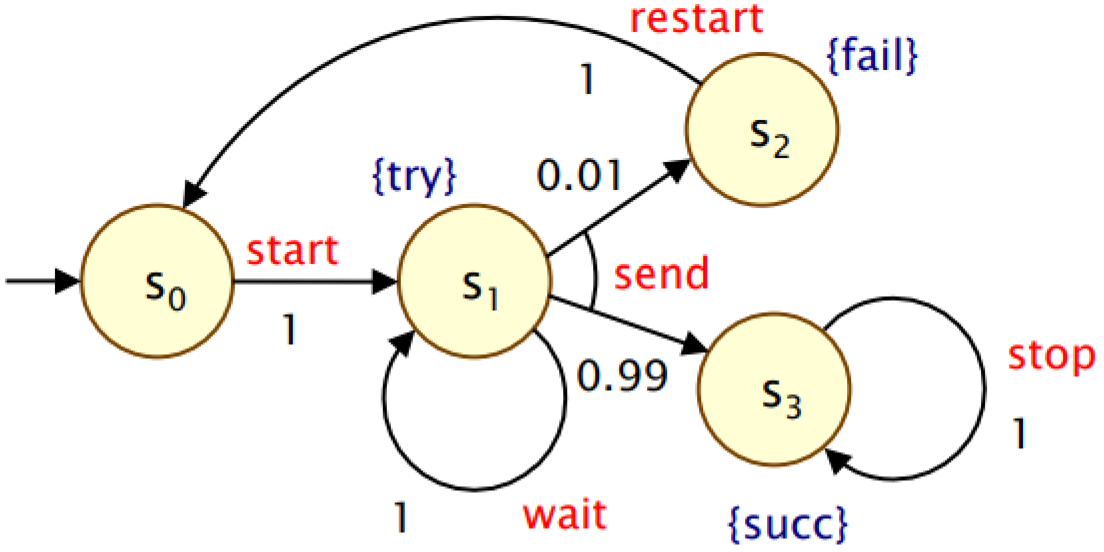


$\sup_{\sigma} Pr_s^{\sigma}(F\ s_3)$

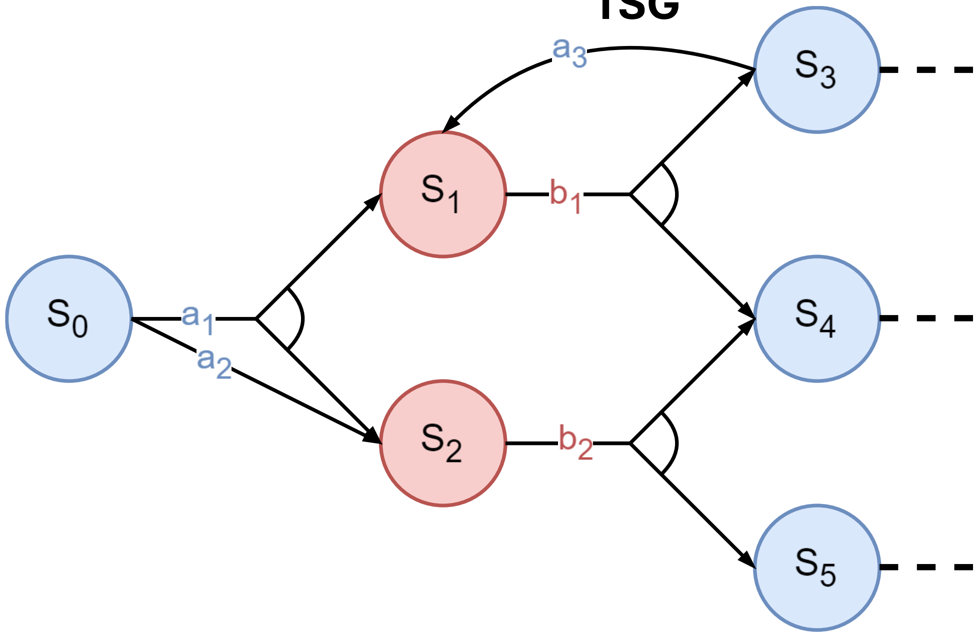
Stochastic Multiplayer Games

- If we add Players to a MDP, we can easily obtain a Turn-based Stochastic Game. Where each player control a subset of the states set.

MDP



TSG

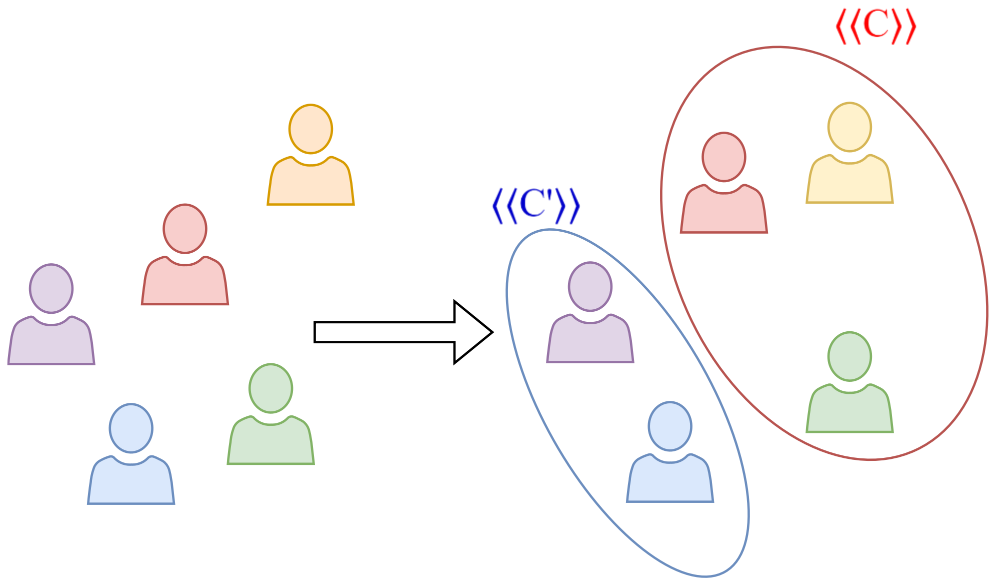


Prism Games

- To model a game on Prism, we will define the **rPATL**,
  «reward probabilistic alternating temporal logic».
  Based on the computational tree logic (**CTL**), it is extended with:
  - Coalition operator $\langle\langle C \rangle\rangle$ from Altertating-time Temporal Logic (ATL)
  - Probability operator **P** from pCTL
  - Reward operator **R** form PRISM
- Example:
  - $\langle\langle \{player_1, player_2\} \rangle\rangle$ $\mathbf{P}_{\geq 90}$ [$F^{\leq 5}$ $goal_1$] & **R**{"cost"}≤15 [F $goal_1$]
    "player 1 and 2 have a strategy to ensure that the probability of reaching the $goal_1$ state within 5 steps is at least 90%. The expected cost incurred before that happens is **no more than 15**."

Coalition for Zero-Sum games

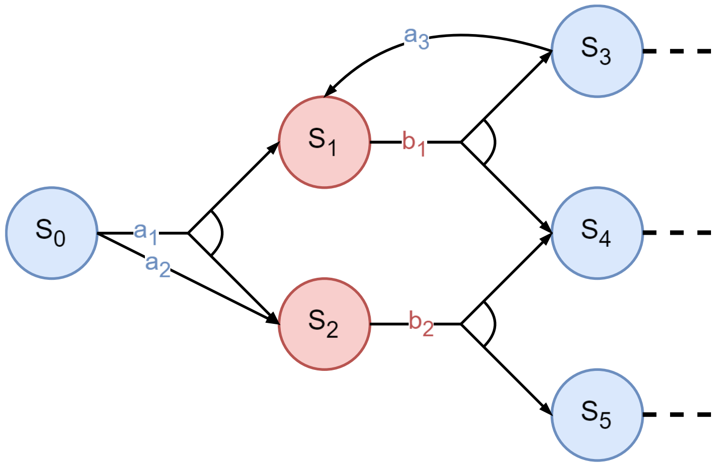
- The coalition operator ⟨⟨C⟩⟩ from ATL, allows us to reduce every turn-based game into a two plater Zero-Sum Game for the desired property, the coalition C againts the coalition of every other player.
- What we want to solve now is
  - $\sup_{\sigma_1}\inf_{\sigma_2}Pr_s^{\sigma_1\sigma_2}(F\ s')$



Value Iteration for TSG

- Similar recursive expression used to solve MDP, but now we have to separate the states between the players

$$p(s) = \begin{cases} 1, & if\ s = goal \\ max_a \sum_{s'} \delta(s, a)(s') \cdot p(s'), & if\ s \neq goal\ and\ s \in S_1 \\ min_a \sum_{s'} \delta(s, a)(s') \cdot p(s'), & if\ s \neq goal\ and\ s \in S_2 \end{cases}$$



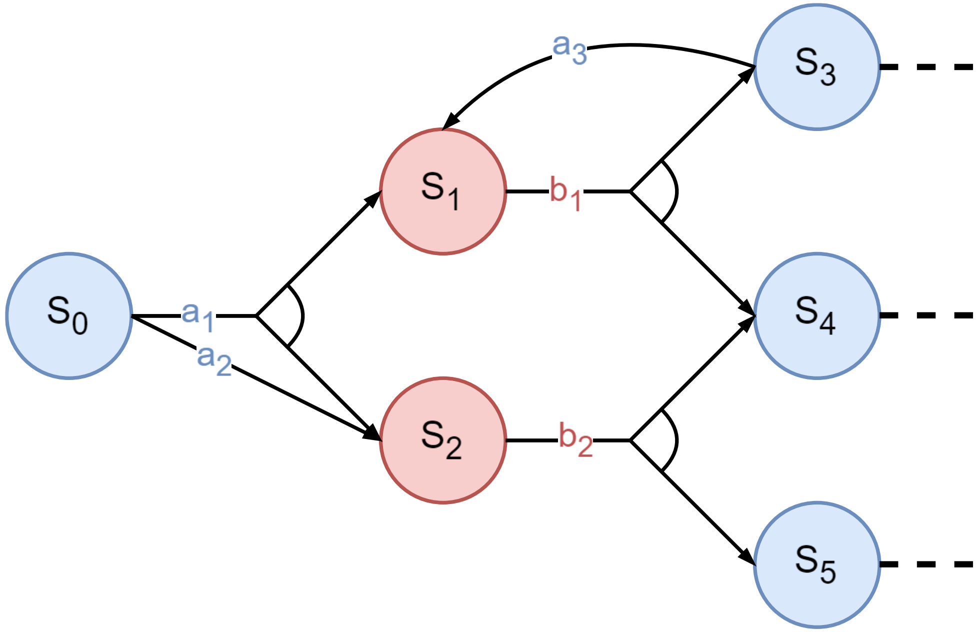
$\sup_{\sigma_1} \inf_{\sigma_2} Pr_s^{\sigma_1 \sigma_2}(F\ s')$

Concurrent Stochastic Games

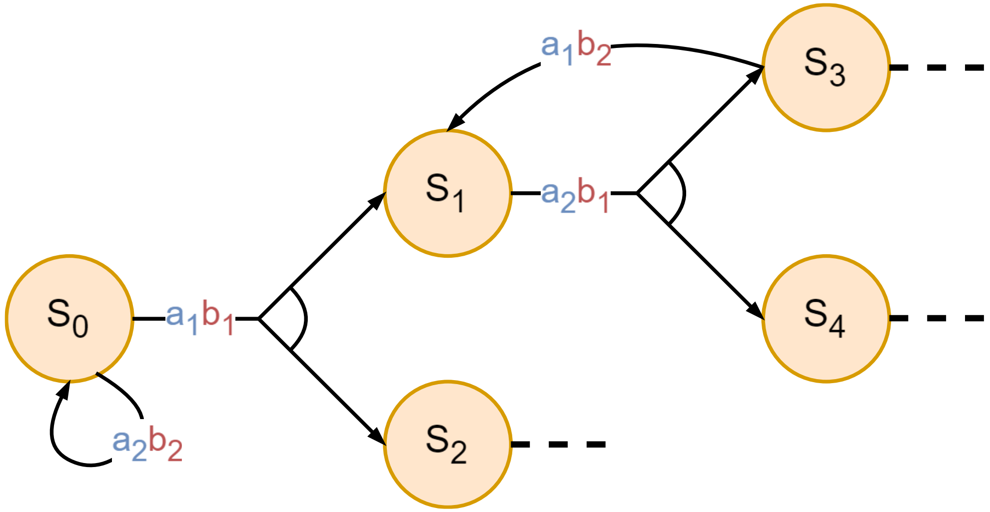
- In a Turn-based Stochastic Game, only **one player at the time** can make an action
- Each player **knows the other player current state** and makes a decision based on that
- A more realistic model have all its agent operating **concurrently**, making strategical choice indipendently

Concurrent Stochastic Games

Turn-based Stochastic Games (TSG)

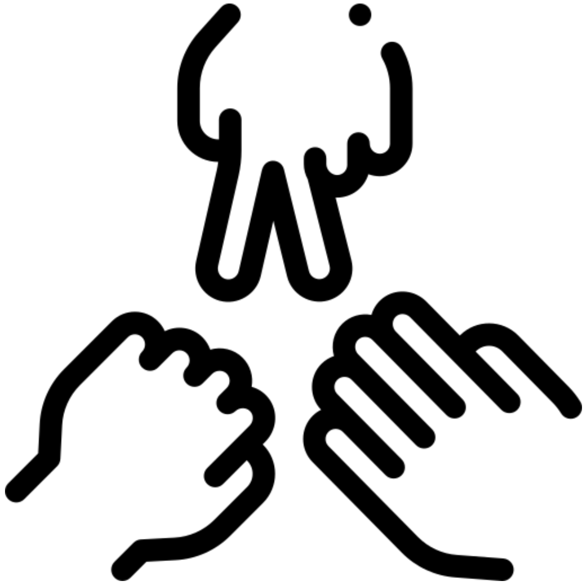


Concurrent Stochastic Games (CSG)



Concurrent Stochastic Games

- CSG are a generalization of TSG, in which we jointly determine the probabilistic successor state.
- In TSG the best strategy was deterministic, now the best strategy can be randomized
- The logic used to model CSG properties is still rPATL but withsome changes



Value Iteration for CSG

- What we want to solve is still:
  - $\sup_{\sigma_1}\inf_{\sigma_2}\Pr_s^{\sigma_1\sigma_2}(F\ s')$
- And the recursive expression is now:

I/II	not confess	confess
not confess	(2,2)	(7,0)
confess	(0,7)	(5,5)

$$p(s) = \begin{cases} 1, & if\ s = goal \\ val(G), & otherwise \end{cases}$$

Where G is the bimatrix game with

$$g_{ij} = \sum_{s'} \delta\big(s, (a_i, b_j)\big)(s') \cdot p(s')$$

- The bimatrix game can be solved by Linear Programming

Modelling CSG

- Players are now associated with **modules, instead of states**, this way a state is not bound to a single player.
- Modules represent individial agents’ behaviour and choices, modules with no nondeterministic choice (like channels) **do not need to be tied to a player**.
- Player concurrently select between multiple **possible** actions
- This results in a complex, interdependent probabilistic distribution for the resulting joint-actions.

Medium Access Control in CSG

```
csg
// Player specification
player p1 mac1 endplayer
player p2 mac2 endplayer
// Max energy per user
const int emax;
// User 1
module mac1
        s1 : [0..1] init 0; // Has user 1 sent?
        e1 : [0..emax] init emax; // Energy level of user 1
        [w1] true -> (s1'=0); // Wait
        [t1] e1>0 -> (s1'=c'?0:1) & (e1'=e1-1); // Transmit
endmodule
// Define second user using module renaming
module mac2 = mac1 [ s1=s2, e1=e2, w1=w2, t1=t2 ] endmodule
```

Player are associated to modules.

```
// Probability qi for transmission success when i users send
const double q1;
const double q2;
// Channel (computes joint transmission probabilities)
module channel
        c : bool init false; // Did a collision occur during transmission?
        [t1,w2] true -> q1:(c'=false) + (1-q1):(c'=true); // User 1 transmits
        [w1,t2] true -> q1:(c'=false) + (1-q1):(c'=true); // User 2 transmits
        [t1,t2] true -> q2:(c'=false) + (1-q2):(c'=true); // Both transmit
endmodule
```

Transition and **Guarded Command** are now a list since the players move concurrently.

Modules with no nondeterministic choice do not need to be tied to a player.

```
// Reward structures
rewards "mess1" // Number of messages sent by user 1
        s1=1 : 1;
endrewards
rewards "mess2" // Number of messages sent by user 2
        s2=1 : 1;
endrewards
rewards "send2" // Number of times users 1 and 2 transmit simultaneously
        [t1,t2] true : 1;
endrewards
```

From Zero-Sum to Equilibria

- With rPATL we solved **zero-sum** game with the maximisation of the probability of reaching the wanted goal for a coalition.
- In Prism game 3.0 we can compute the ***social-welfare optimal Nash equilibria*** in which we maximise (or minimize) the sum of the values associated to the objectives for each player.
- The general idea is that player can have distinct objective which are not directly opposing (zero-sum)
- To do this we add to rPATL the **+** operator

What is an equilibria?

- Let $G = (N, S, \mu)$ be a Game.
- A Nash Equilibrium is a strategy profile $x^* \in S$ such that the strategy x_i^* is a best response to the strategy profile x_{-i}^* for all $i \in N$

$$u_i(x_i^*, x_{-i}^*) \geq u_i(x_i, x_{-i}^*) \qquad \text{for all } x_i \in S_i$$

I/II	not confess	confess
not confess	(2,2)	(7,0)
confess	(0,7)	(5,5)

(confess, confess) is the unique Nash Equilibrium

Not Pareto optimal: both player could get less years
Not socially optimal: it is not the best joint best result

Equilibria II

- The *social-welfare optimal Nash equilibria* is a kind of equilibria where there is no incentive for any player to unilaterally change strategy, given the maximisation of the **sum** of the players' payoff.

Zero-sum

$\langle\langle\{player_1\}\rangle\rangle_{max=?}\ \mathbf{P}\,[F^{\leq k}\ goal_1]$

SWNE

$\langle\langle\{player_1,player_2\}\rangle\rangle_{max=?}\ (\mathbf{P}\,[F^{\leq k}\ goal_1]+\mathbf{P}\,[F^{\leq k}\ goal_2])$

Value Iteration for CSG with equilibria

- The recursive expression is now:

$$p(s) = \begin{cases} (1,1), & if\ s_1 = goal_1, s_2 = goal_2 \\ (P_{max}(s, goal_1), 1), & if\ s_1 \neq goal_1, s_2 = goal_2 \\ (1, P_{max}(s, goal_2)), & if\ s_1 = goal_1, s_2 \neq goal_2 \\ SWNE(G(s)), & if\ s_1 \neq goal_1, s_2 \neq goal_2 \end{cases}$$

- Now we have pairs of value in order to consider both player objectives
- The first three cases are a MDP

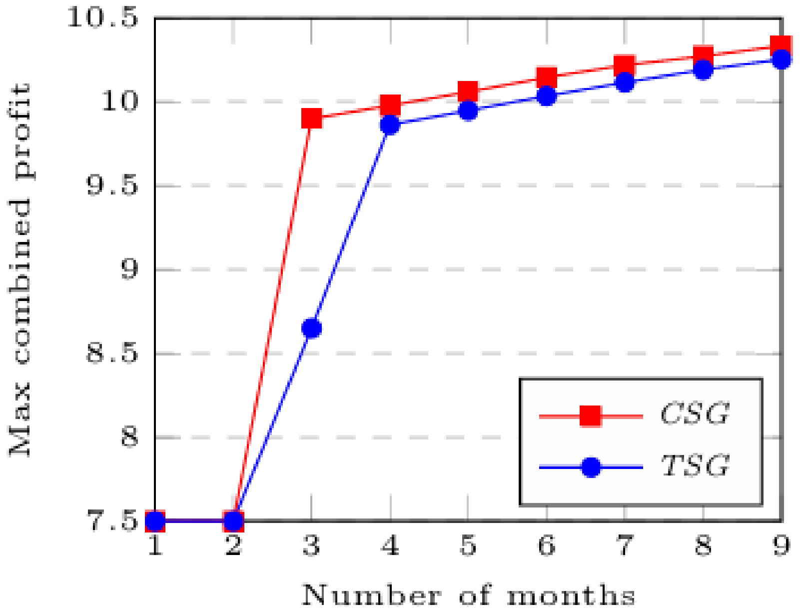
Equilibria for unbounded properties

- ‘Unbounded properties’ are properties that do not specify a time or step limit. They require reasoning about behavior potentially extending indefinitely into the future
  - ❖ Example:
    “probability of eventually reaching state ***s’*** (regardless of the number of steps).
- Nash Equilibria might not exist or might not be computable, hence PRISM-games computes **ε-Nash equilibria**
  - A strategy profile $\sigma = (\sigma_1, \dots, \sigma_n)$ for a CSG is a ***ε-NE*** for state ***s*** and objective $X_1, \dots, X_n$ iff:

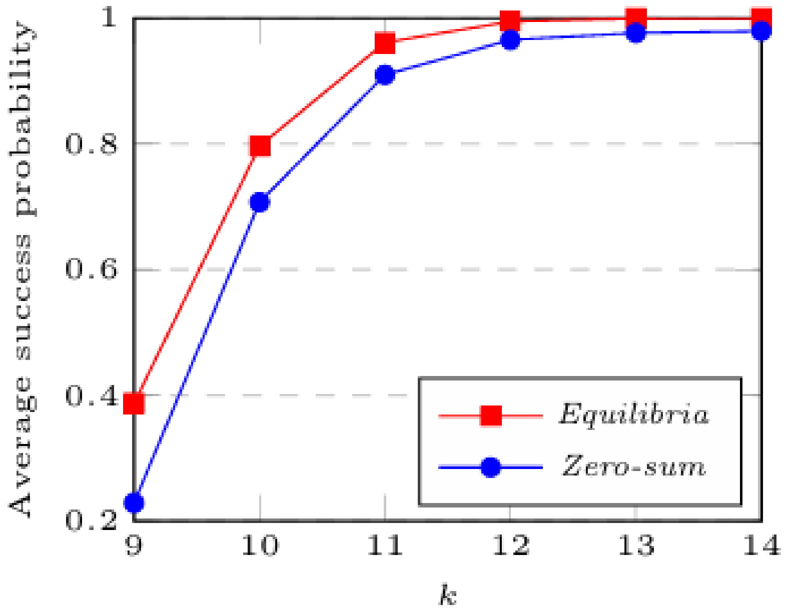
$$\Pr_{\boldsymbol{s}}^{\boldsymbol{\sigma}}(\mathrm{X_i}) \geq \sup_{\boldsymbol{\sigma}_i'} \Pr_{\boldsymbol{s}}^{(\sigma_{-i}, \boldsymbol{\sigma}')}(\mathrm{X_i}) - \varepsilon, \quad \forall i.$$

Case studies

Future markets investor

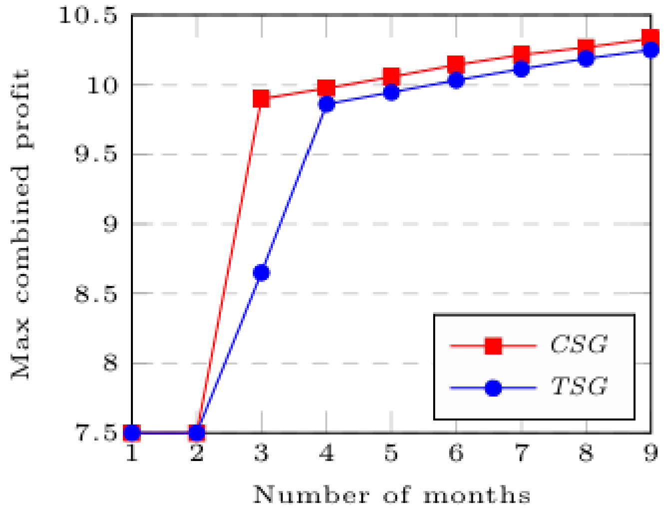


Robot coordination



Future Market Invesors

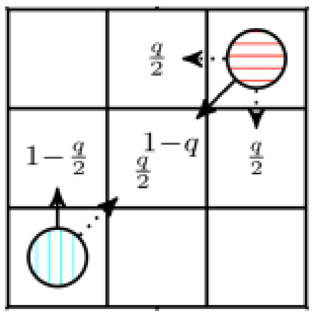
- A stock market (the model) evolves stochastically
- Two investors decides when to invest
- The market can refuse the investments
- Making concurrent decision is more realistic, the market can not observe the investors strategies.

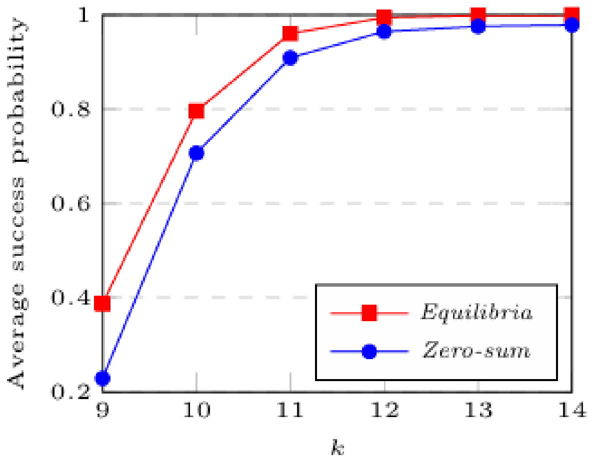


(a) **Future markets investor:** avoiding unrealistic strategy choices using CSGs

Robot coordination

- Two robots navigating in a $l \times l$ grid
- Starting from the opposite corners, the goal is to reach the opposite corners.
- Each step has a probability q to fail to simulate an obstacle
- When robots collide, they will need to wait before moving

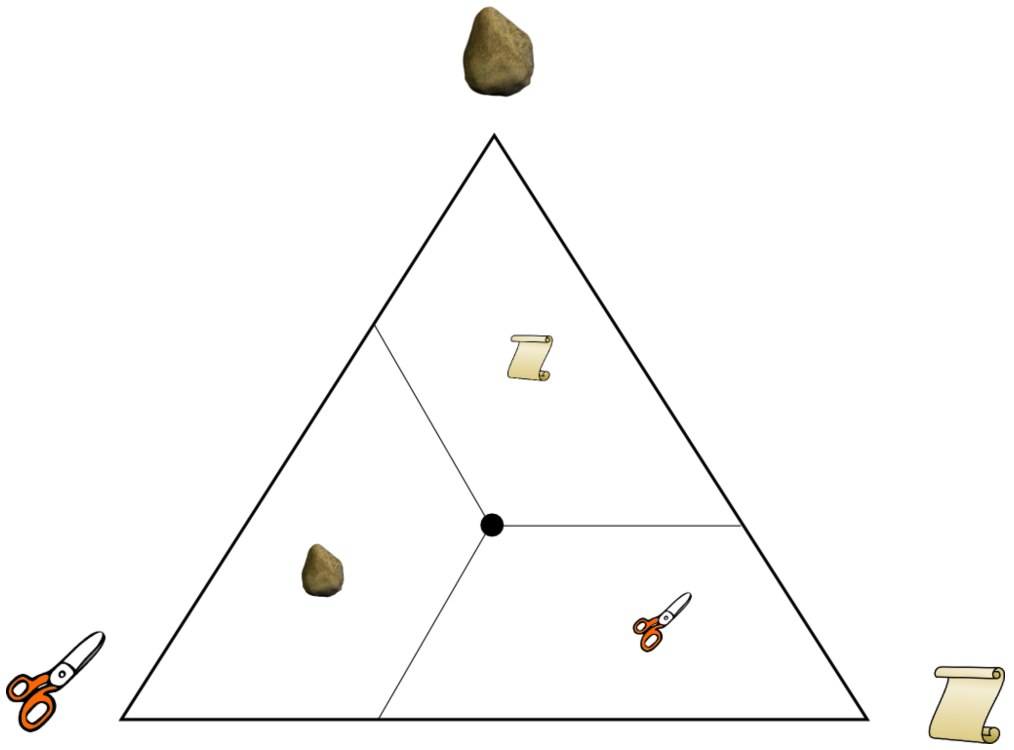




(b) **Robot coordination:** using equilibria for mutually beneficial navigation plans

Example: Rock Paper Scissor

- Very intuitive game with simple rules.
- Zero-sum game with no *Pure strategy Nash Equilibria*
- By modelling the game in Prism 3.0 we can observe the expected probability of the simple model, and calculate the *Mixed strategies Nash Equilibria*



DEMO

Example: Rock Paper Scissor

- “What is the maximum probability that Player 1 wins before ever losing?”

<<p1>>Pmax=? [win!=2 U win=1]

$$p = \frac{1}{3} \cdot 1 \ \ (\text{win}) + \frac{1}{3} \cdot 0 \ \ (\text{loss}) + \frac{1}{3} \cdot p \ \ (\text{draw, retry})$$

$$p = \frac{1}{3} + \frac{1}{3}p \implies p - \frac{1}{3}p = \frac{1}{3} \implies \frac{2}{3}p = \frac{1}{3} \implies p = \frac{1}{2}$$

- “What is the maximum probability that Player 1 gets exactly 1 win and exactly 2 draws in 3 rounds?”

<<p1>>Pmax=? [F (win_count=1 & draw_count=2)]

$$3 \times \frac{1}{27} = \frac{3}{27} = \frac{1}{9}$$