Reactive Synthesis

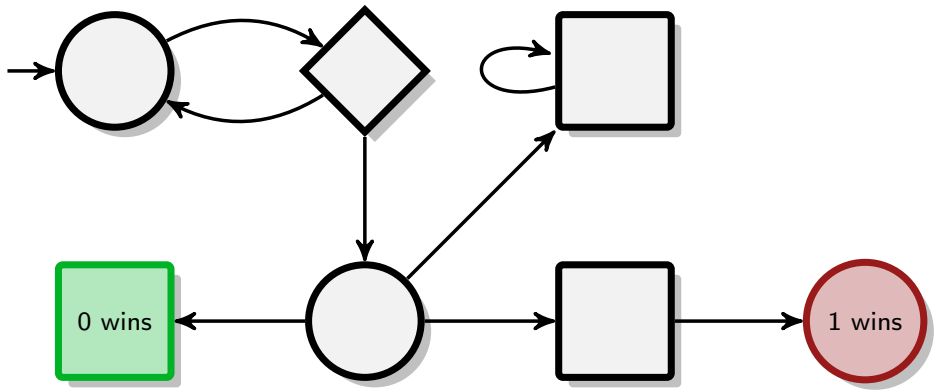
Lecture 12

Swen Jacobs and Martin Zimmermann
(Saarland University)

Outlook

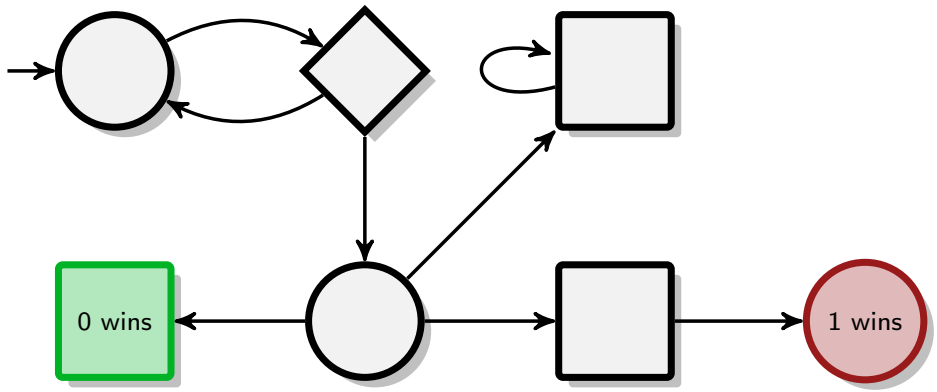
(Simple) Stochastic Games

Enter a new player Nature (◇): it flips a coin to pick a successor.



(Simple) Stochastic Games

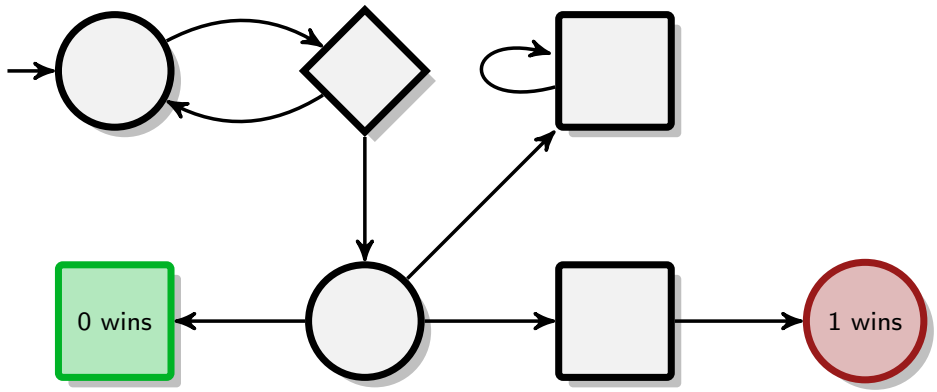
Enter a new player Nature (◇): it flips a coin to pick a successor.



- No (sure) winning strategy...

(Simple) Stochastic Games

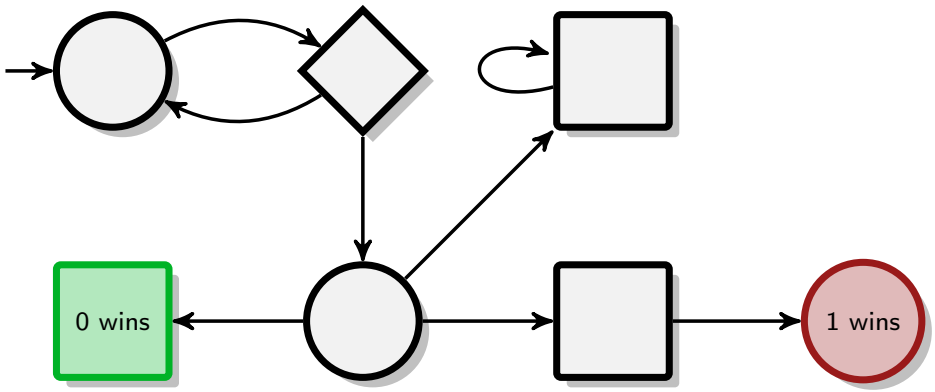
Enter a new player Nature (◇): it flips a coin to pick a successor.



- No (sure) winning strategy... but one with probability 1.

(Simple) Stochastic Games

Enter a new player Nature (◇): it flips a coin to pick a successor.



- No (sure) winning strategy... but one with probability 1.
- Value of the game for Player 0:

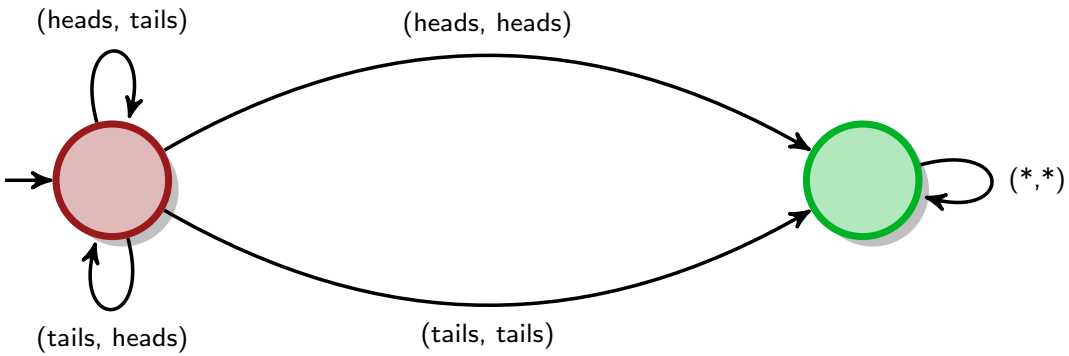
$$\max_{\sigma} \min_{\tau} p_{\sigma,\tau}$$

where $p_{\sigma,\tau}$ is the probability that Player 0 wins when using strategy σ and Player 1 uses strategy τ.

Concurrent Games

Both players choose their moves simultaneously.

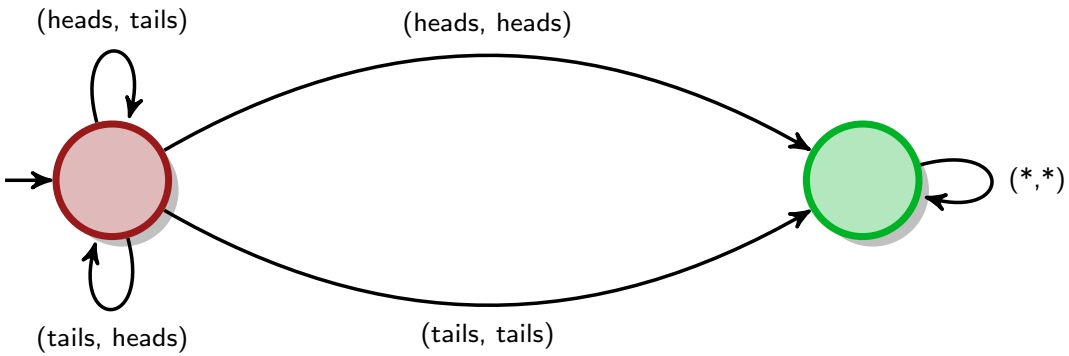
- Matching pennies:



Concurrent Games

Both players choose their moves simultaneously.

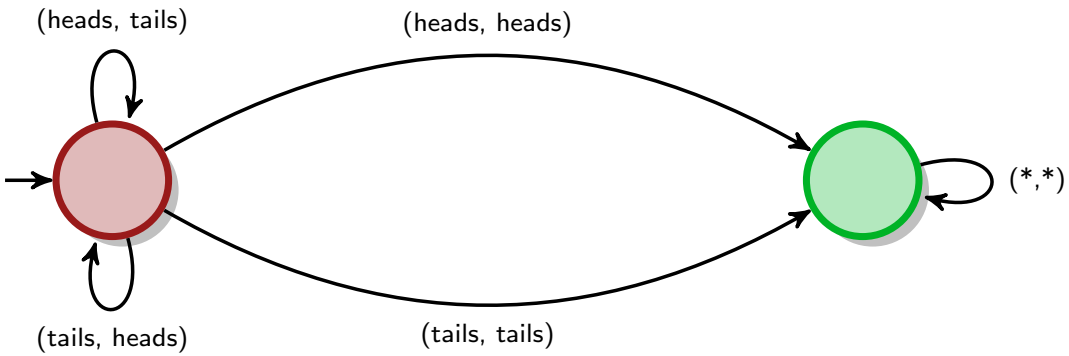
- Matching pennies: randomized strategy winning with probability 1.



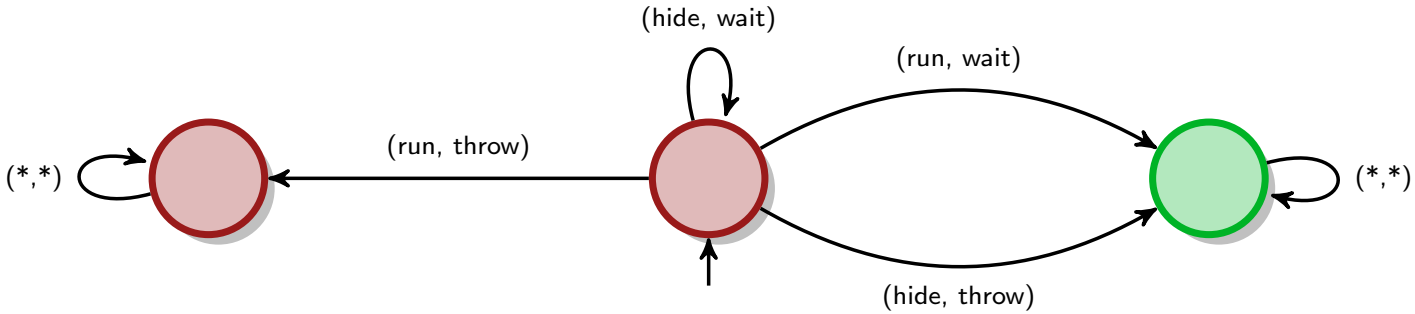
Concurrent Games

Both players choose their moves simultaneously.

- Matching pennies: randomized strategy winning with probability 1.



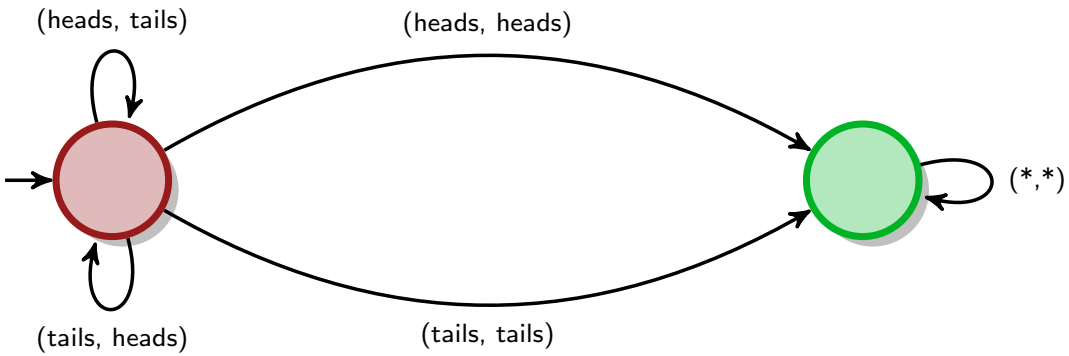
- The "Snowball Game":



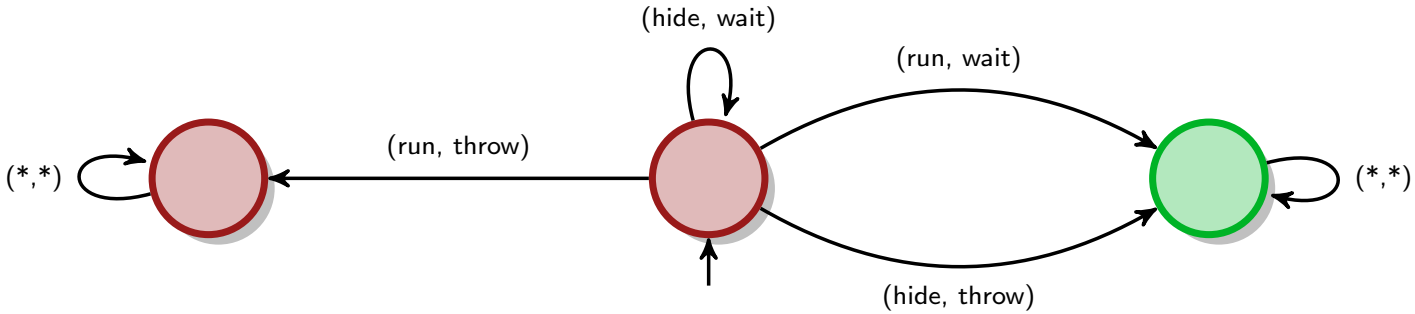
Concurrent Games

Both players choose their moves simultaneously.

- Matching pennies: randomized strategy winning with probability 1.

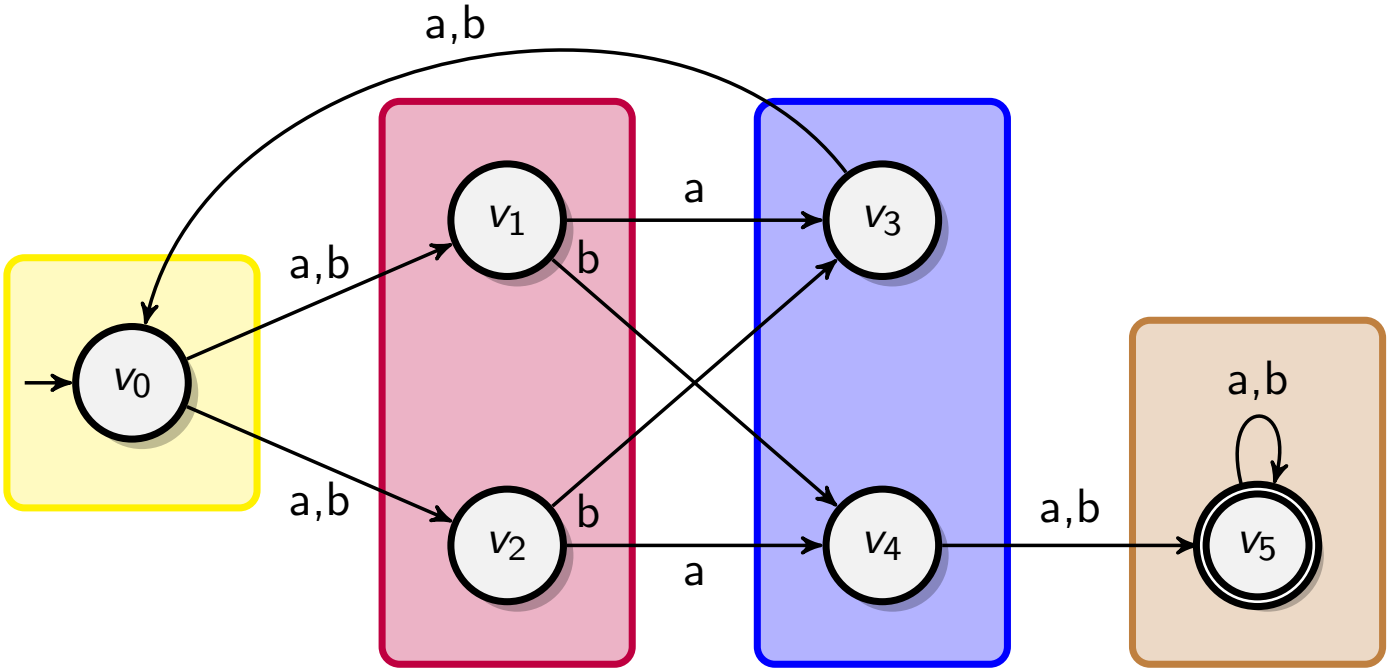


- The "Snowball Game": for every ε, randomized strategy winning with probability $1 - \varepsilon$.



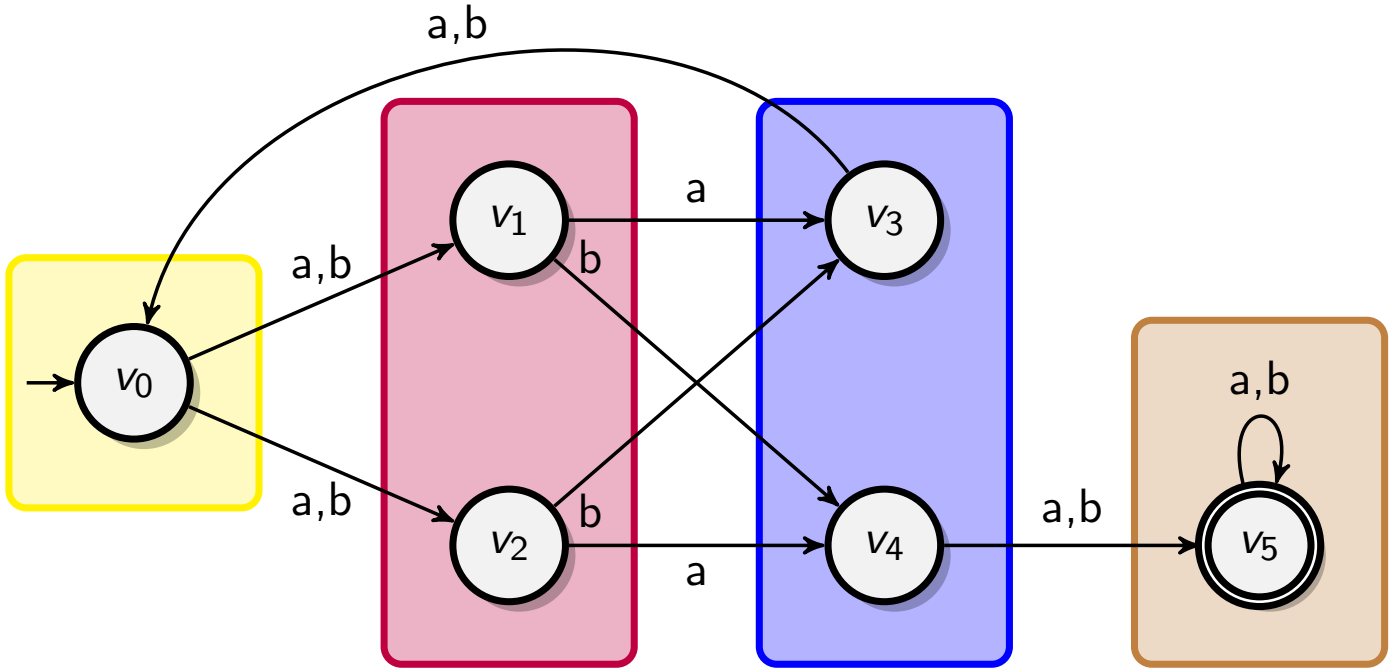
Games of Imperfect Information

- Players do not observe sequence of states, but sequence of non-unique observations (yellow, purple, blue, brown).
- Player 0 picks action a/b, Player 1 resolves non-determinism.



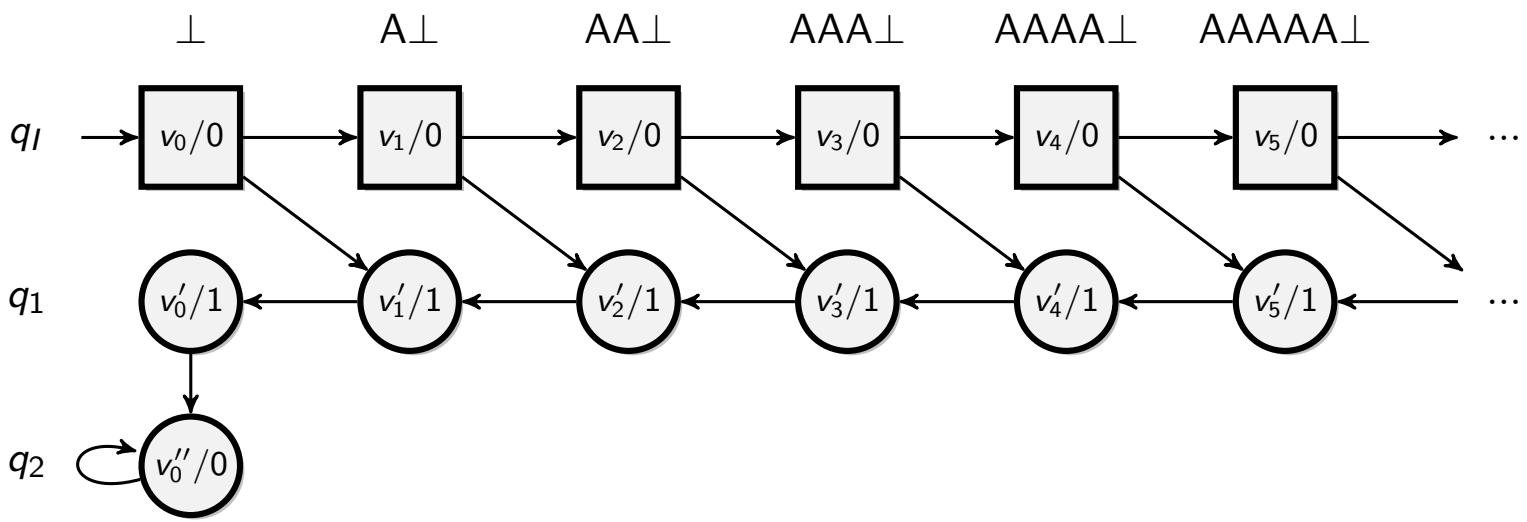
Games of Imperfect Information

- Players do not observe sequence of states, but sequence of non-unique observations (yellow, purple, blue, brown).
- Player 0 picks action a/b, Player 1 resolves non-determinism.

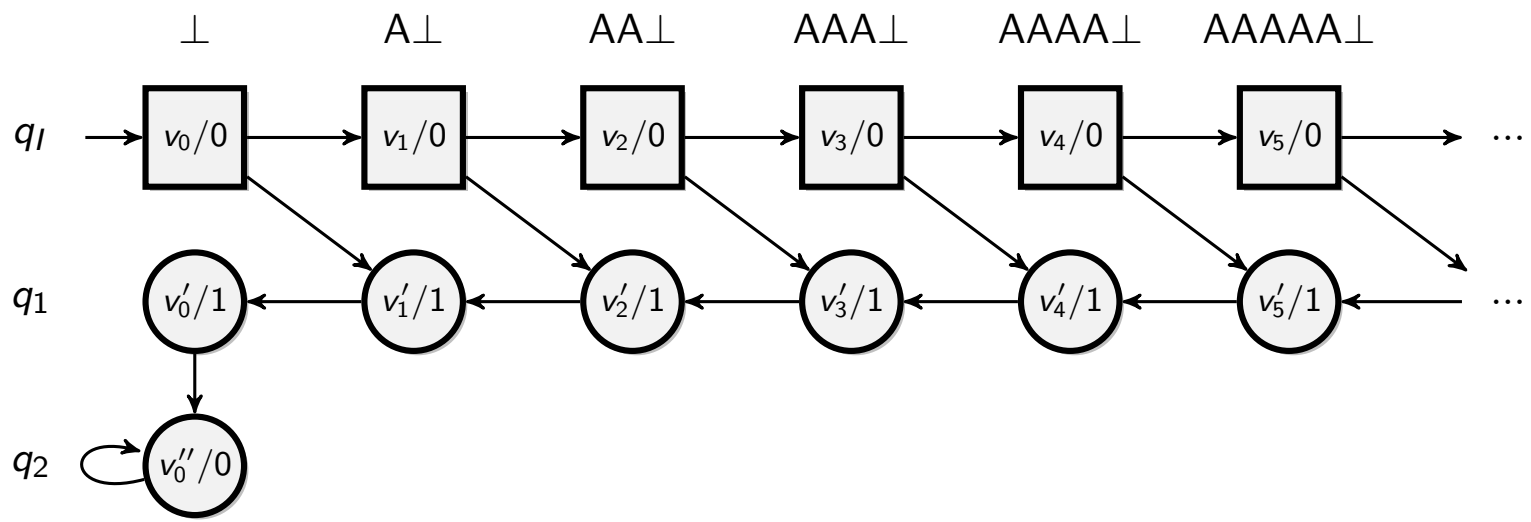


No winning strategy for Player 0: every fixed choice of actions to pick at ( )*( ) can be countered by going to v_1 or v_2.

Pushdown Games



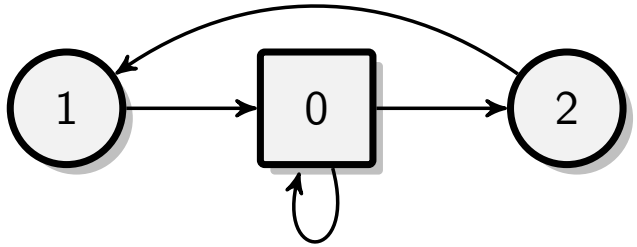
Pushdown Games



- Pushdown Parity Games can be reduced to parity games in exponentially sized arenas $\Rightarrow$ solvable in exponential time.
- Both players have positional winning strategies (but these are now infinite objects!).
- Finite representation of winning strategies: pushdown automata with output.

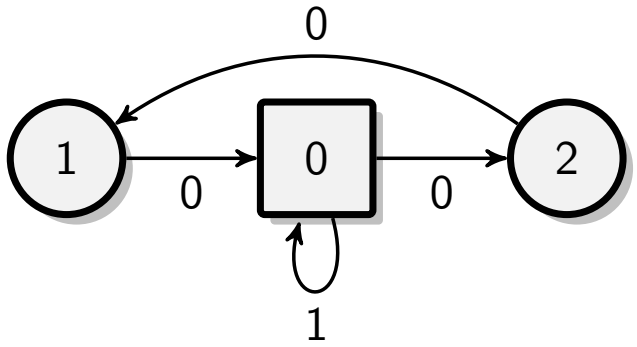
Quantitative Games

- Parity game: Player 0 wins from everywhere, but it takes arbitrarily long to "answer" 1 with 2.



Quantitative Games

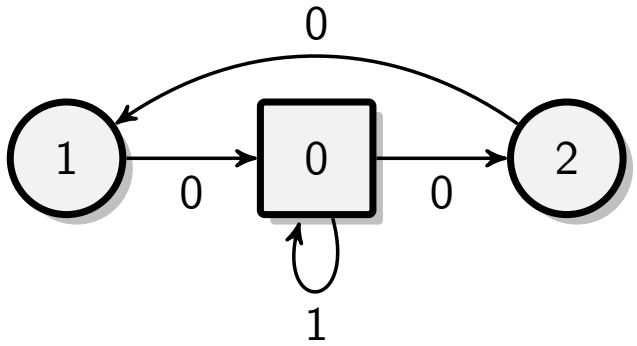
- Parity game: Player 0 wins from everywhere, but it takes arbitrarily long to "answer" 1 with 2.



- Add edge-costs: Player 0 wins if there is a bound b and a position n such that every odd color after n is followed by a larger even color with cost $\leq b$ in between $\Rightarrow$

Quantitative Games

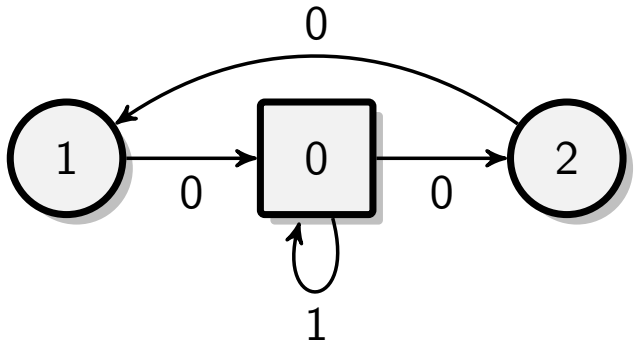
- Parity game: Player 0 wins from everywhere, but it takes arbitrarily long to "answer" 1 with 2.



- Add edge-costs: Player 0 wins if there is a bound b and a position n such that every odd color after n is followed by a larger even color with cost $\leq b$ in between $\Rightarrow$ Player 1 wins example from everywhere (stay longer and longer in 0).

Quantitative Games

- Parity game: Player 0 wins from everywhere, but it takes arbitrarily long to "answer" 1 with 2.



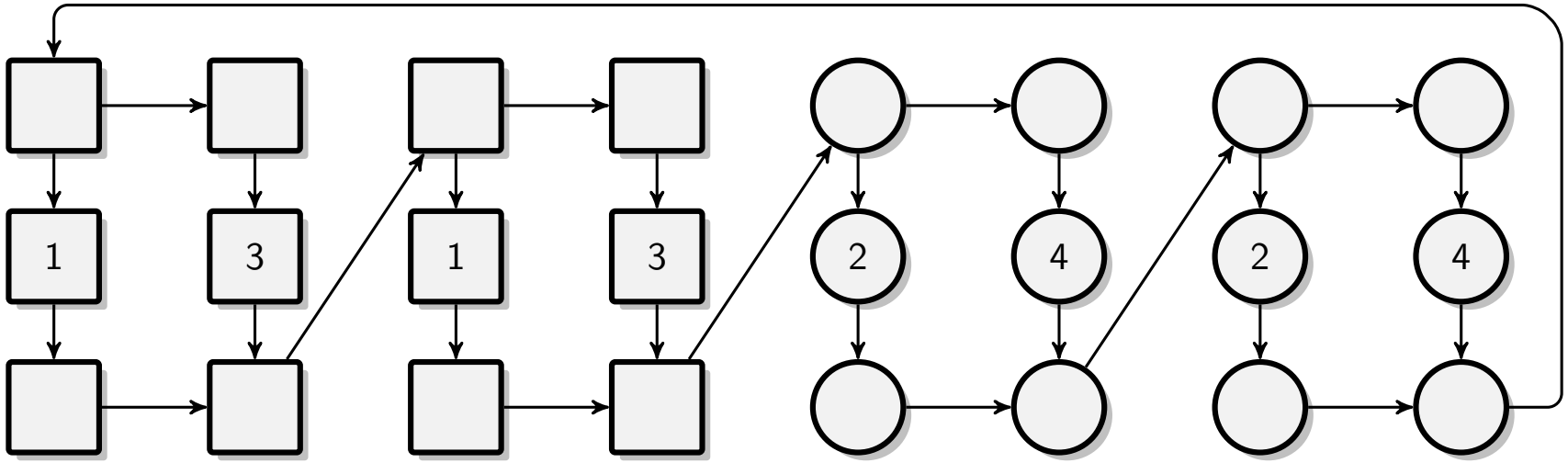
- Add edge-costs: Player 0 wins if there is a bound b and a position n such that every odd color after n is followed by a larger even color with cost $\leq b$ in between $\Rightarrow$ Player 1 wins example from everywhere (stay longer and longer in 0).

Theorem

Parity games with costs are determined, Player 0 *has positional winning strategies, and they can be solved in quasi-polynomial time.*

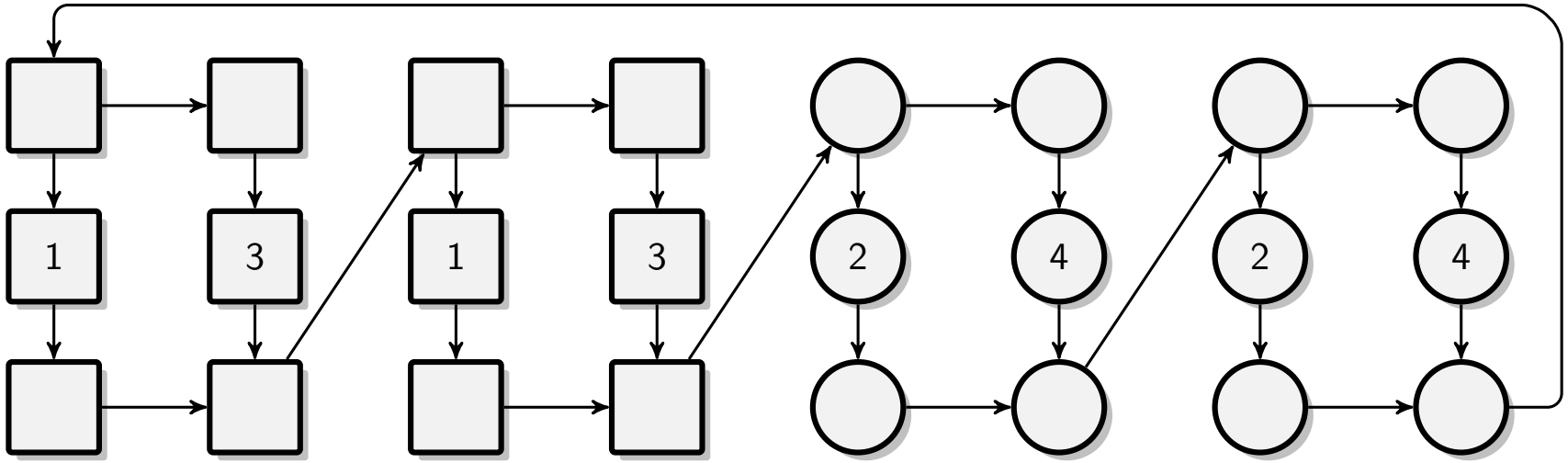
Tradeoffs

Every edge has cost 1



Tradeoffs

Every edge has cost 1

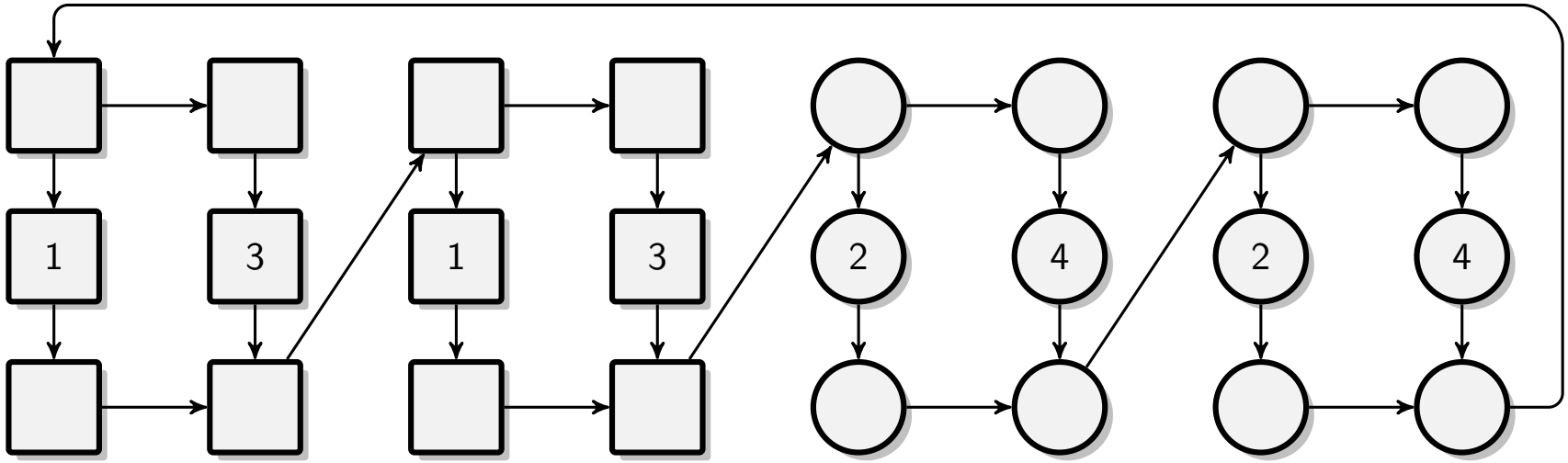


Player 0 has:

- Positional winning strategy with bound 9.
- Finite-state strategy with 2 states and bound 8.

Tradeoffs

Every edge has cost 1



Player 0 has:

- Positional winning strategy with bound 9.
- Finite-state strategy with 2 states and bound 8.

With d odd colors and d gadgets for each player: Player 0 has:

- Positional winning strategy with bound $d^2 + 3d - 1$.
- Finite-state strategy with 2^{d-1} states and bound $d^2 + 2d$.

Many other variants

- More winning conditions: various quantitative conditions

Many other variants

- More winning conditions: various quantitative conditions
- Games on timed automata $\Rightarrow$ uncountable arenas

Many other variants

- More winning conditions: various quantitative conditions
- Games on timed automata $\Rightarrow$ uncountable arenas
- Play even longer: games of ordinal length

Many other variants

- More winning conditions: various quantitative conditions
- Games on timed automata $\Rightarrow$ uncountable arenas
- Play even longer: games of ordinal length
- Games with delay: Player 0 is allowed to skip some moves to obtain lookahead on Player 1's moves. Basic question: what kind of lookahead is necessary to win.

Many other variants

- More winning conditions: various quantitative conditions
- Games on timed automata $\Rightarrow$ uncountable arenas
- Play even longer: games of ordinal length
- Games with delay: Player 0 is allowed to skip some moves to obtain lookahead on Player 1's moves. Basic question: what kind of lookahead is necessary to win.
- More than two players $\Rightarrow$ no longer zero-sum games. Requires whole new theory (equilibria).

Many other variants

- More winning conditions: various quantitative conditions
- Games on timed automata $\Rightarrow$ uncountable arenas
- Play even longer: games of ordinal length
- Games with delay: Player 0 is allowed to skip some moves to obtain lookahead on Player 1's moves. Basic question: what kind of lookahead is necessary to win.
- More than two players $\Rightarrow$ no longer zero-sum games. Requires whole new theory (equilibria).
- etc.

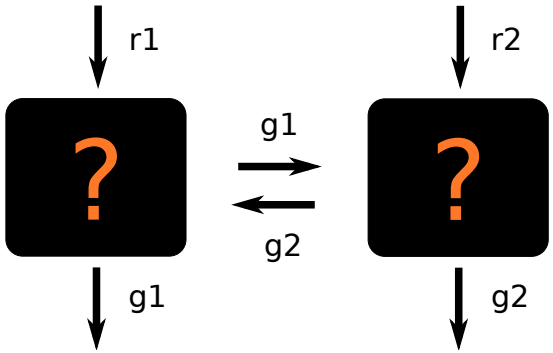
Many other variants

- More winning conditions: various quantitative conditions
- Games on timed automata $\Rightarrow$ uncountable arenas
- Play even longer: games of ordinal length
- Games with delay: Player 0 is allowed to skip some moves to obtain lookahead on Player 1's moves. Basic question: what kind of lookahead is necessary to win.
- More than two players $\Rightarrow$ no longer zero-sum games. Requires whole new theory (equilibria).
- etc.

And: any combination of extensions discussed above.

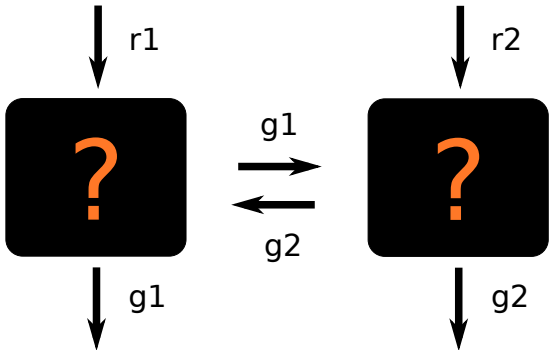
Distributed Synthesis

- Synthesize multiple components of a distributed system at once



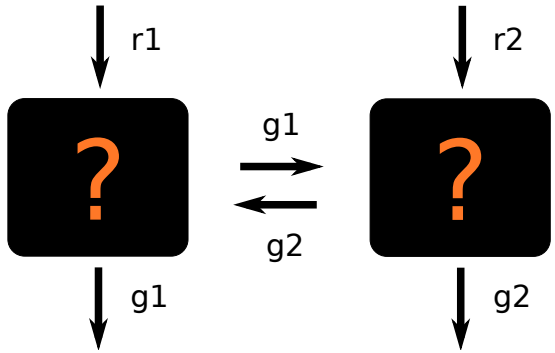
Distributed Synthesis

- Synthesize multiple components of a distributed system at once
- Each component can only act on local information and observations
  $\Rightarrow$ partial information



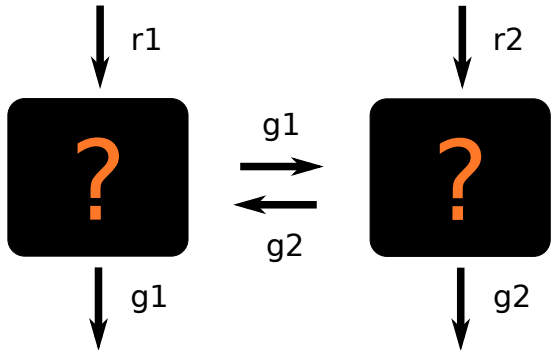
Distributed Synthesis

- Synthesize multiple components of a distributed system at once
- Each component can only act on local information and observations
  $\Rightarrow$ partial information
- In general undecidable
  - $\Rightarrow$ try to identify conditions under which decidability is regained
  - $\Rightarrow$ semi-decision procedures (Bounded Synthesis)



Distributed Synthesis

- Synthesize multiple components of a distributed system at once
- Each component can only act on local information and observations
  $\Rightarrow$ partial information
- In general undecidable
  - $\Rightarrow$ try to identify conditions under which decidability is regained
  - $\Rightarrow$ semi-decision procedures (Bounded Synthesis)



And: combinations with previously mentioned extensions, as well as special properties in distributed systems.

Fault-tolerant Distributed Synthesis

- In (large) distributed systems, need to model the possibility of (hardware) faults

Fault-tolerant Distributed Synthesis

- In (large) distributed systems, need to model the possibility of (hardware) faults
- We can take the possibility of such faults into consideration at synthesis time

Fault-tolerant Distributed Synthesis

- In (large) distributed systems, need to model the possibility of (hardware) faults
- We can take the possibility of such faults into consideration at synthesis time
- Self-stabilization: system can go to an arbitrary state and must stabilize to a "good" state

Fault-tolerant Distributed Synthesis

- In (large) distributed systems, need to model the possibility of (hardware) faults
- We can take the possibility of such faults into consideration at synthesis time
- Self-stabilization: system can go to an arbitrary state and must stabilize to a "good" state
  $\Rightarrow$ can be modeled by making all states initial (and possibly fixing a stabilization time)

Fault-tolerant Distributed Synthesis

- In (large) distributed systems, need to model the possibility of (hardware) faults
- We can take the possibility of such faults into consideration at synthesis time
- Self-stabilization: system can go to an arbitrary state and must stabilize to a "good" state
  $\Rightarrow$ can be modeled by making all states initial (and possibly fixing a stabilization time)
- Byzantine faults: a subset of the components could be controlled by an adversary

Fault-tolerant Distributed Synthesis

- In (large) distributed systems, need to model the possibility of (hardware) faults
- We can take the possibility of such faults into consideration at synthesis time
- Self-stabilization: system can go to an arbitrary state and must stabilize to a "good" state
  $\Rightarrow$ can be modeled by making all states initial (and possibly fixing a stabilization time)
- Byzantine faults: a subset of the components could be controlled by an adversary
  $\Rightarrow$ can be modeled by making this component part of the environment (and by considering all possible combinations of Byzantine components)

Synthesis and Verification of Parameterized Systems

- What if we don't know the number of components in the system?

Synthesis and Verification of Parameterized Systems

- What if we don't know the number of components in the system?

$\Rightarrow$ Synthesize or verify for an arbitrary number of components

Synthesis and Verification of Parameterized Systems

- What if we don't know the number of components in the system?

$\Rightarrow$ Synthesize or verify for an arbitrary number of components

- already verification undecidable in general

Synthesis and Verification of Parameterized Systems

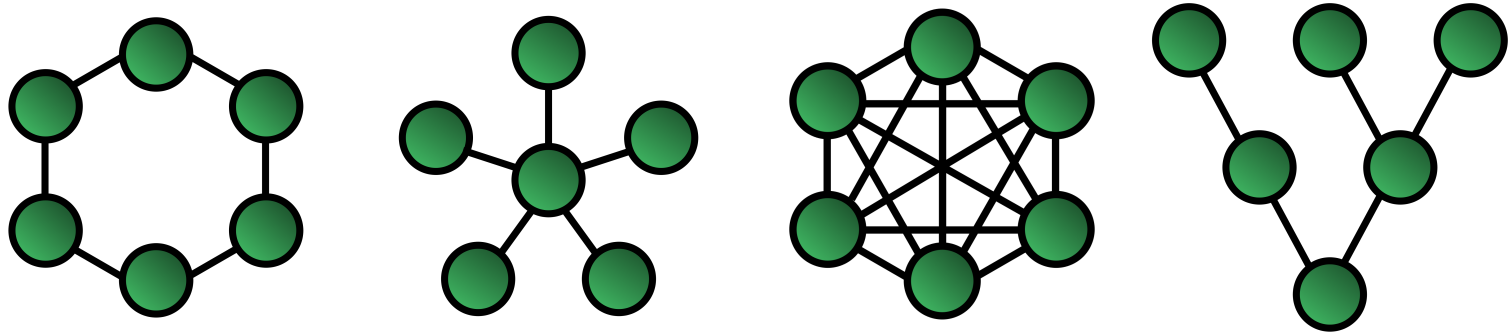
- What if we don't know the number of components in the system?

$\Rightarrow$ Synthesize or verify for an arbitrary number of components

- already verification undecidable in general

$\Rightarrow$ try to identify conditions under which decidability is regained (network topology, communication primitives)

$\Rightarrow$ semi-decision procedures



Thank you
&
Good luck for the exam!