

Recent results on Howard's algorithm

Peter Bro Miltersen

Aarhus University

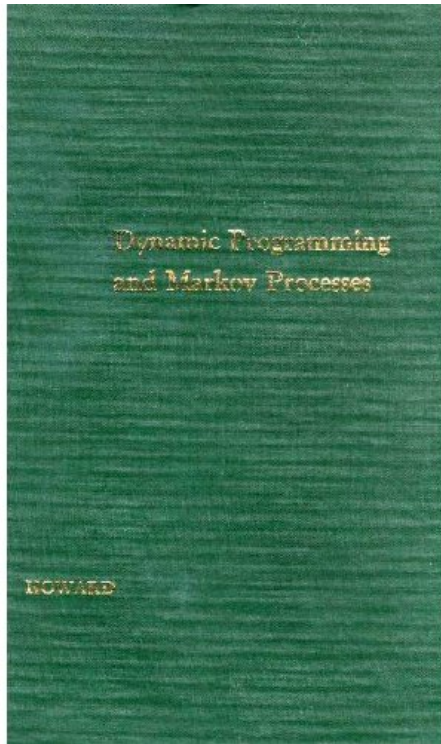
Computer Science Department

Ingredients of CFEM

- Game theory and mechanism design
- Cryptography
- Algorithms and Operations Research

Howard's algorithm (1960)

(aka policy iteration, policy improvement,
strategy iteration/improvement)



Basic algorithm for online, sequential
decision making in face of uncertainty

Optimal Replacement Policies for Dairy Cows Based on Daily Yield Measurements

Lars Relund Nielsen*

Department of Business Studies, Aarhus University, Fuglesangs Allé 4, DK-8210 Aarhus V, Denmark.

Erik Jørgensen

Department of Genetics and Biotechnology, Aarhus University, PO Box 50, DK-8830 Tjele, Denmark.

Anders Ringgaard Kristensen

Department of Large Animal Sciences, University of Copenhagen, Grønnegårdsvej 2, DK-1870 Frederiksberg C, Denmark.

Søren Østergaard

Department of Animal Health and Bioscience, Aarhus University, PO Box 50, DK-8830 Tjele, Denmark.

January 27, 2010



An MDP-Based Approach to Online Mechanism Design

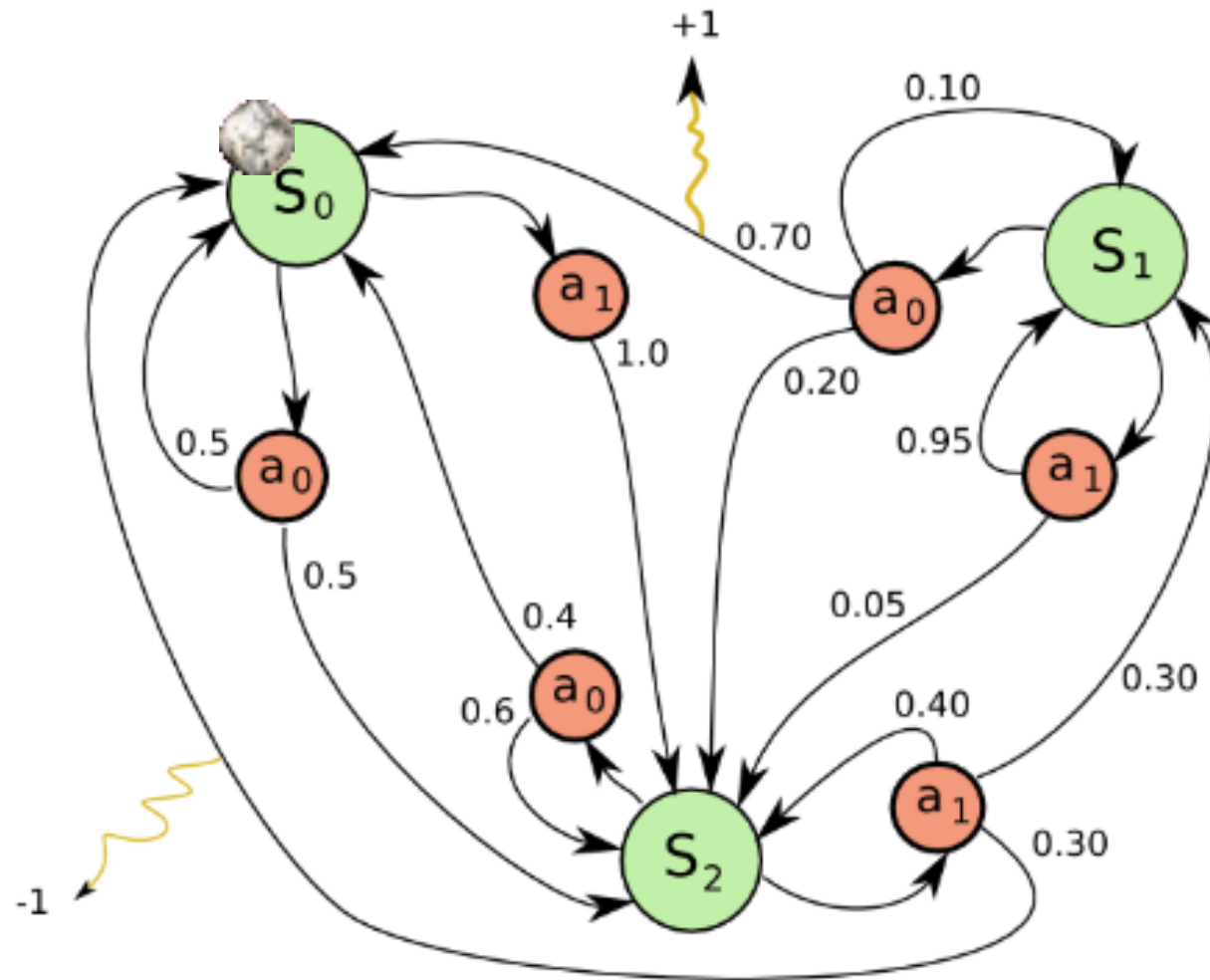
David C. Parkes
Division of Engineering and Applied Sciences
Harvard University
`parkes@eecs.harvard.edu`

Satinder Singh
Computer Science and Engineering
University of Michigan
`baveja@umich.edu`

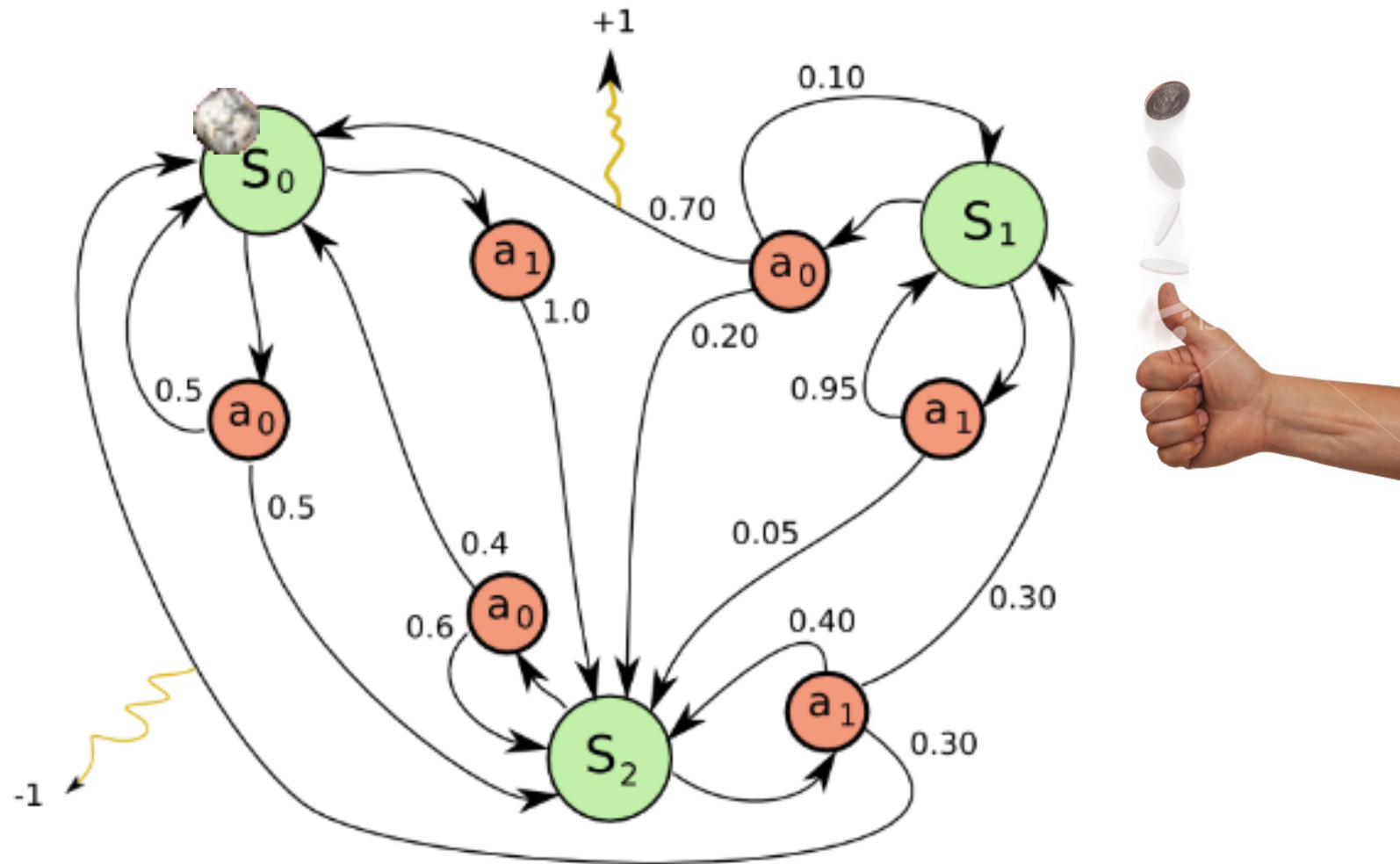
Recent results on the time complexity of Howard's algorithm

- O. Friedman, LICS'09
- J. Fearnley, ICALP'10
- Y. Ye, 2010
- T.D. Hansen, U. Zwick, ISAAC'10
- T.D. Hansen, P.B. Miltersen, U. Zwick, ICS'11
- K.A. Hansen, R. Ibsen-Jensen, P.B. Miltersen, 2011

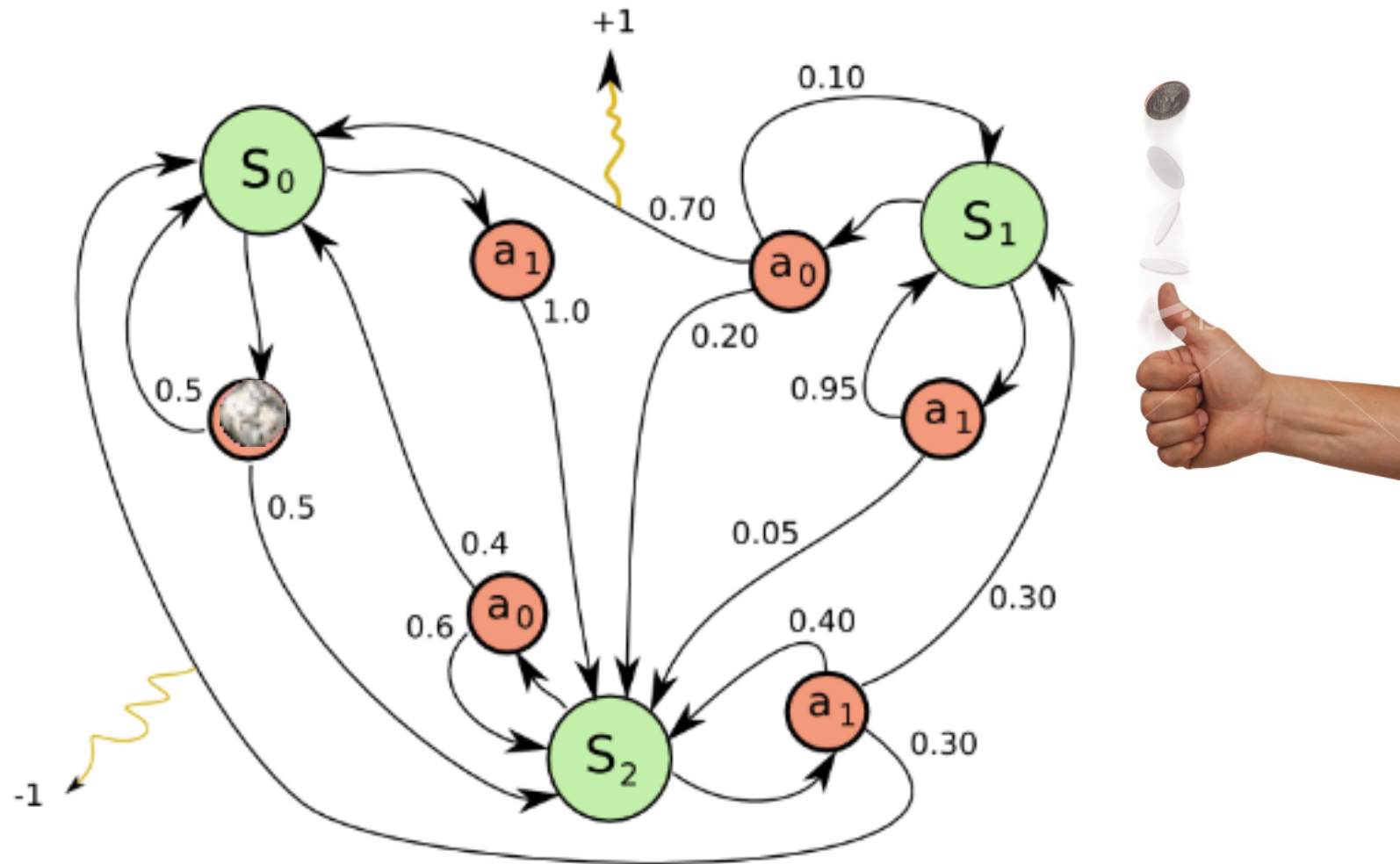
Markov Decision Process (MDP)



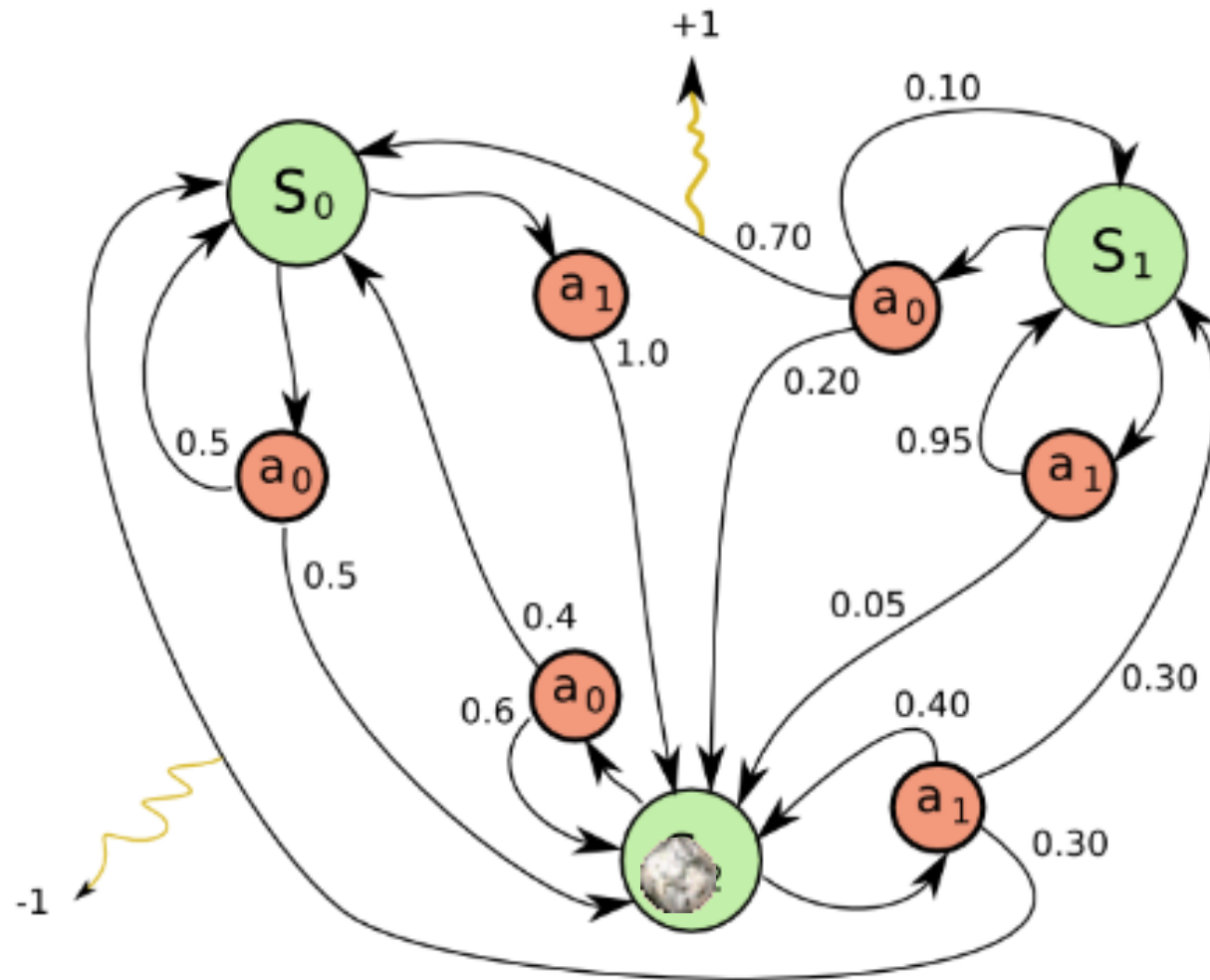
Markov Decision Process (MDP)



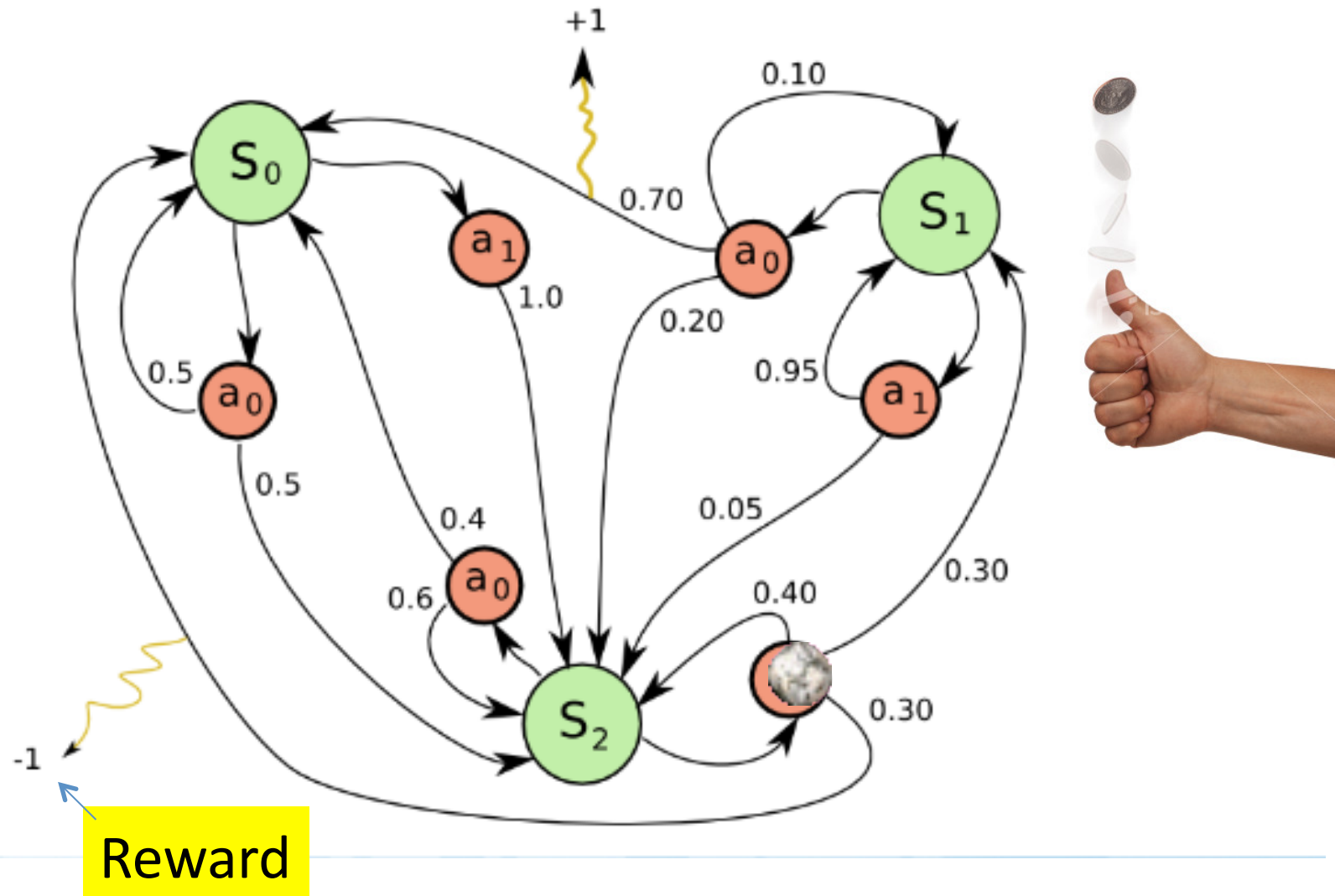
Markov Decision Process (MDP)



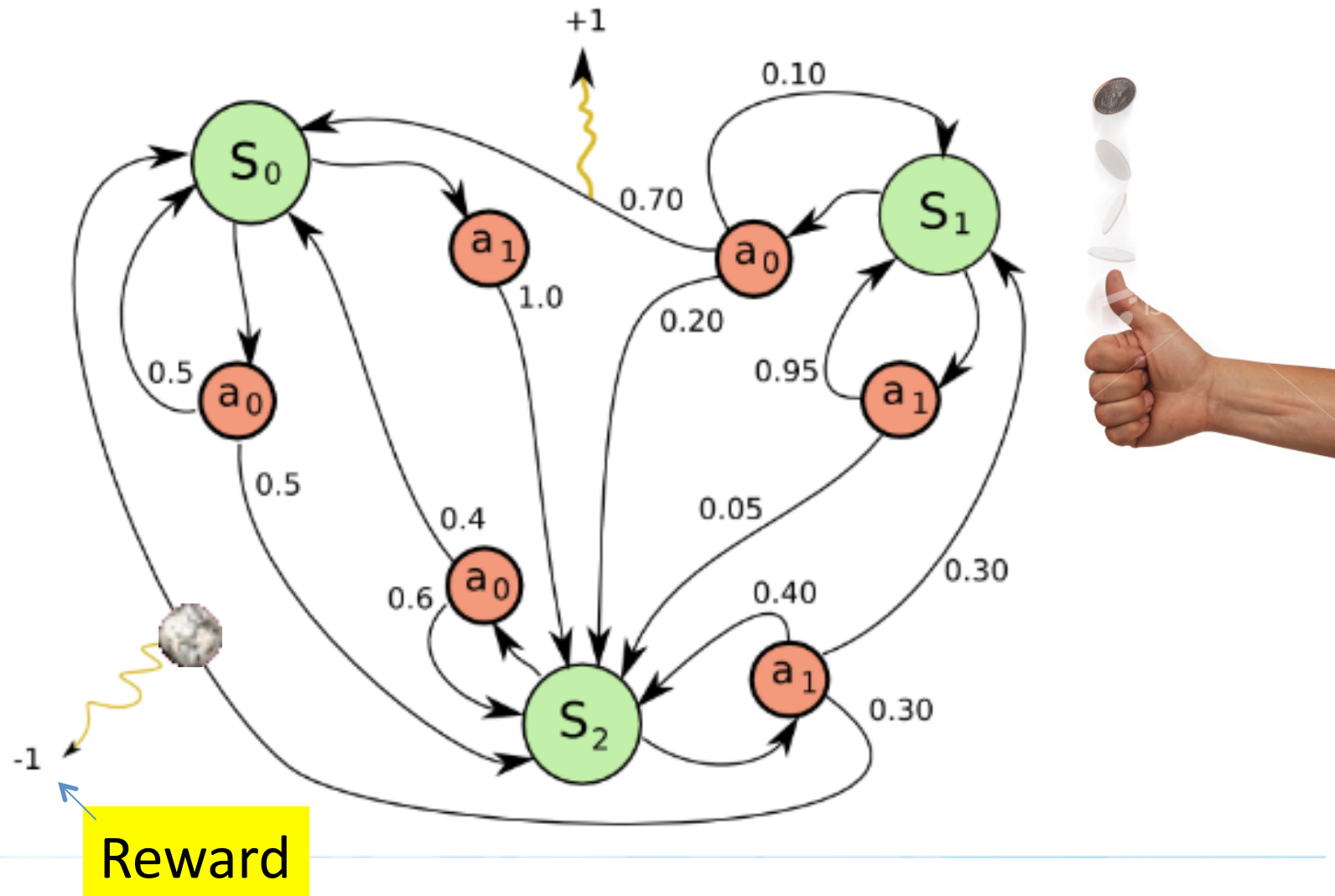
Markov Decision Process (MDP)



Markov Decision Process (MDP)



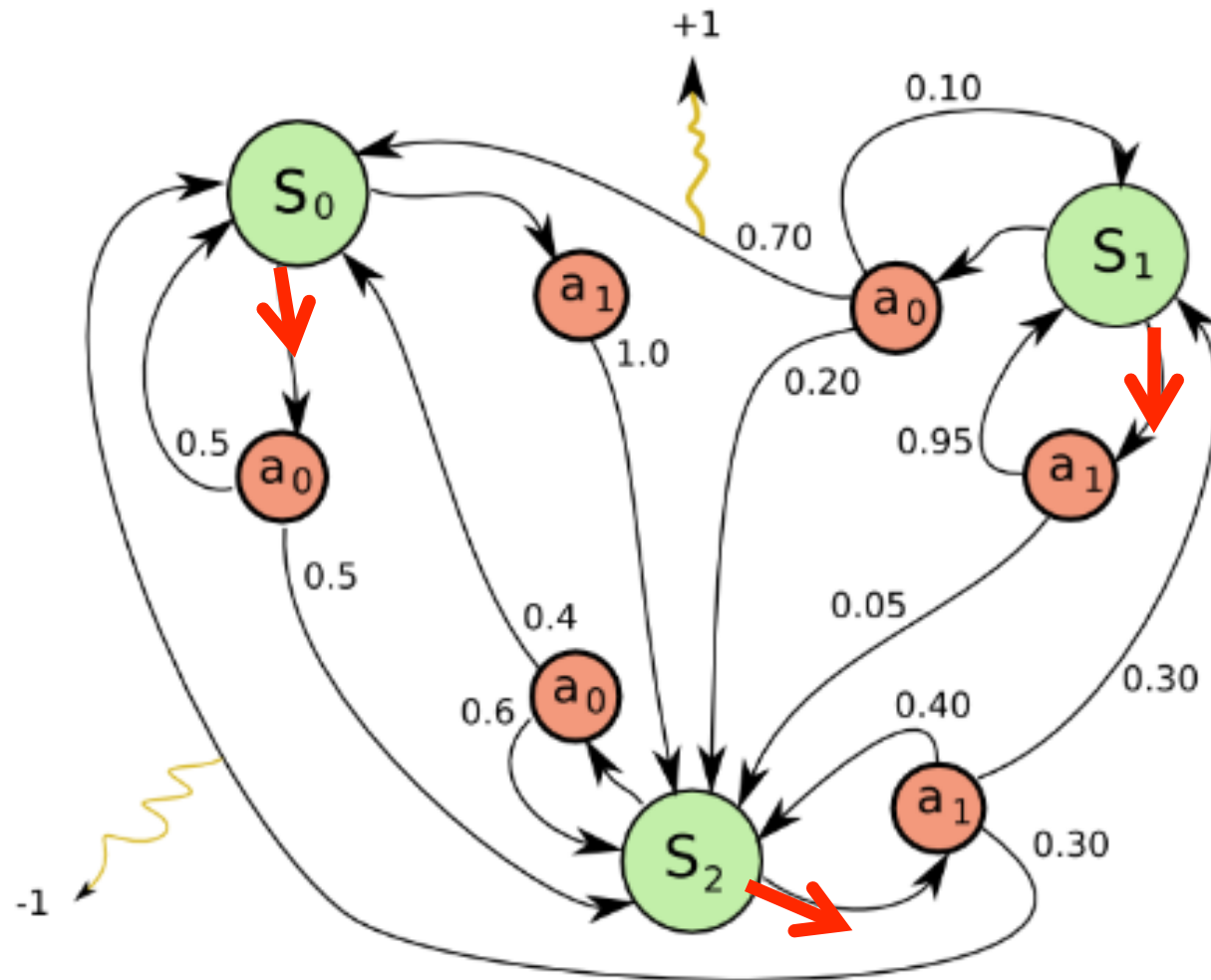
Markov Decision Process (MDP)



Rewards and Payoffs

- We play forever and keep collecting rewards.
- Our ***payoff*** is an accumulated reward that starts at 0.
- If reward r is collected at time t , our payoff is increased by $r(1-\lambda)^t$, where λ is the ***discount factor***.
- We want to find a ***policy*** that maximizes our expected total payoff.

A policy



Howard's algorithm (1960)

- Start with an arbitrary policy (a choice of action in each position)
- Compute expected total payoff for each position
- For all positions, if a chosen action picks a lower expected payoff over a higher expected payoff, ***switch***.
- Iterate!

Howard's algorithm

- *The* method of choice in practice for solving MDPs.
- Complexity analysis?

Letter to the Editor:

How Good is Howard's Policy Improvement Algorithm?

By *N. Schmitz*¹

Some standard algorithms of Operations Research are known to be very useful in practice, although they may show an extremely bad ("exponentially bad") worst case behaviour. An important example is Dantzig's simplex algorithm for linear programming (see e.g. Klee/Minty [1972]).

From our experience the first property (good behaviour / small number of iterations for real life problems) also applies to Howard's policy improvement algorithm for Markovian decision processes (see Mine/Osaki [1970]). However, we could not find any information as to the second aspect (bad worst cases). Therefore I would like to ask the following

Question 1: What is the worst case behaviour of Howard's policy improvement algorithm?

If the considered decision process concerns a system with N "states" $1, \dots, N$, where in state i one has to choose among m_i "actions", the usual proof of convergence for the policy improvement algorithm yields the (trivial) bound

$$(*) \quad \prod_{i=1}^N m_i$$

A Polynomial Time Bound for Howard's Policy Improvement Algorithm

U. Meister and U. Holzbaur

Universität Ulm, Abteilung Mathematik VII (OR), Oberer Eselsberg, D-7900 Ulm

Received March 18, 1985 / Accepted in revised form October 16, 1985

Summary. We consider a discounted Markovian Decision Process (MDP) with finite state and action space. For a fixed discount factor we derive a bound for the number of steps, taken by Howard's policy improvement algorithm (PIA) to determine an optimal policy for the MDP, that is essentially polynomial in the number of states and actions of the MDP. The main tools are the contraction properties of the PIA and a lower bound for the difference of the value functions of a MDP with

(cf. e.g. Klee and Minty [4]). A trivial bound for the number of iterations in the PIA would be the number of policies of the MDP. We give a bound for the number of iterations in Howard's PIA, that is essentially polynomial, i.e. the bound for the number of iterations in the PIA is linear in the number of states, but depends on the logarithms of the size of the data and of the discount factor as well.

How many iterations before termination?

(assuming fixed discount factor)

- Meister and Holzbaaur, 1986: **$O(n L)$**
 - n = number of states, L = largest *bitsize* of reward.
 - Not a *strongly* polynomial bound.
- Ye, 2010: **$O(m n \log n)$**
 - m = number of actions
 - first strongly polynomial bound
- Hansen, Miltersen, Zwick, 2011: **$O(m \log n)$**

About the proof

- Identify an action that
 - is played in the current policy
 - will ***never*** be played again in any policy that occurs after another $O(\log n)$ iterations.

How to find the action?

Primal:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & (J - \gamma P)^T \mathbf{x} = \mathbf{e} \\ & \mathbf{x} \geq 0 \end{aligned}$$

Dual:

$$\begin{aligned} \max_{\mathbf{v}} \quad & \mathbf{e}^T \mathbf{v} \\ \text{s.t.} \quad & (J - \gamma P) \mathbf{v} + \mathbf{s} = \mathbf{c} \\ & \mathbf{s} \geq 0 \end{aligned}$$

Complementary slackness: $\mathbf{c}^T \mathbf{x} - \mathbf{e}^T \mathbf{v}^* = (\mathbf{s}^*)^T \mathbf{x}$



Take action so that
corresponding term makes
big contribution to this sum

How many iterations before termination?

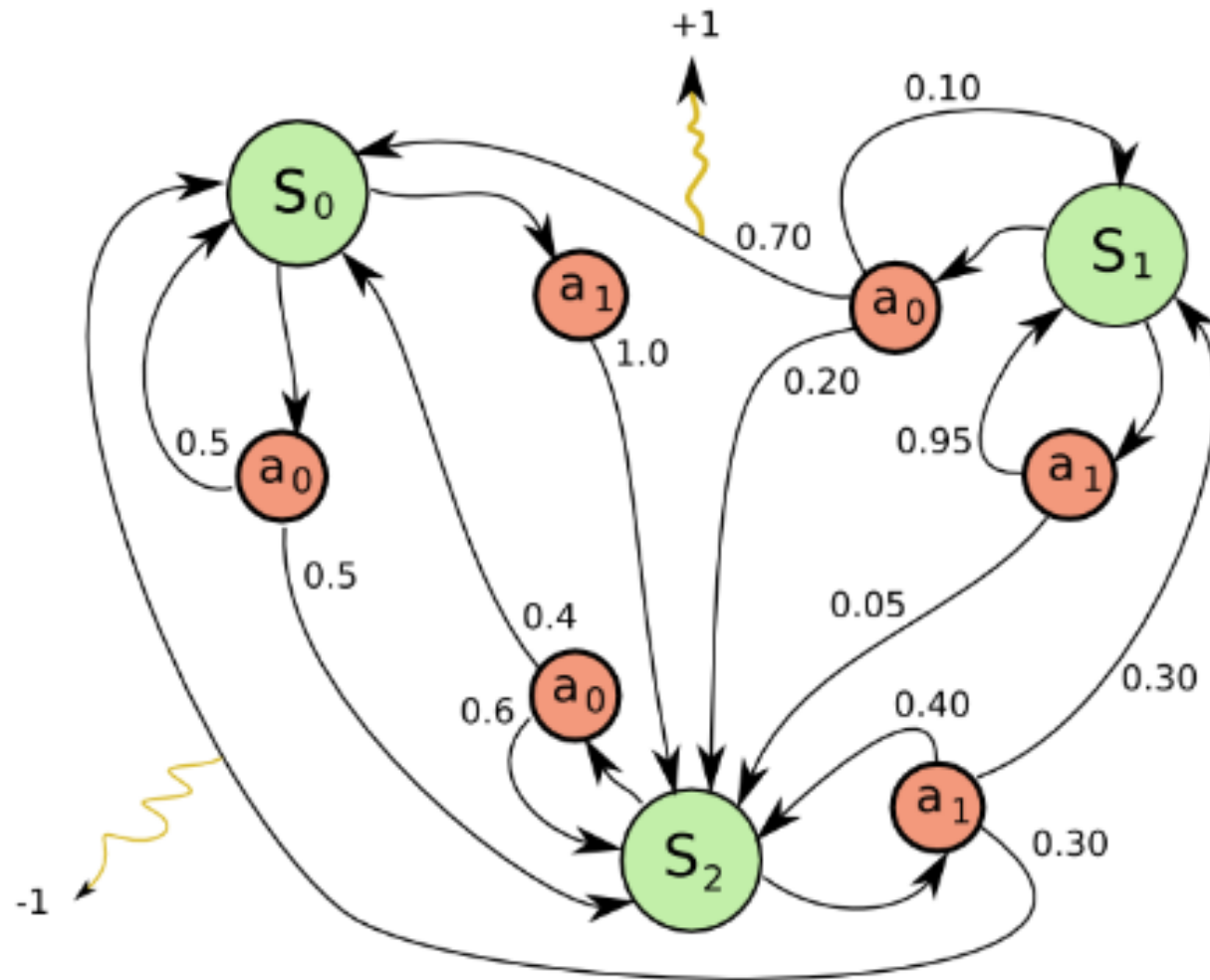
(assuming fixed discount factor)

- Meister and Holzbaaur, 1986: **$O(n L)$**
 - n = number of states, L = largest ***bitsize*** of reward.
 - Not a ***strongly*** polynomial bound.
- Ye, 2010: **$O(m n \log n)$**
 - m = number of actions
 - first strongly polynomial bound
- Hansen, Miltersen, Zwick, 2011: **$O(m \log n)$**

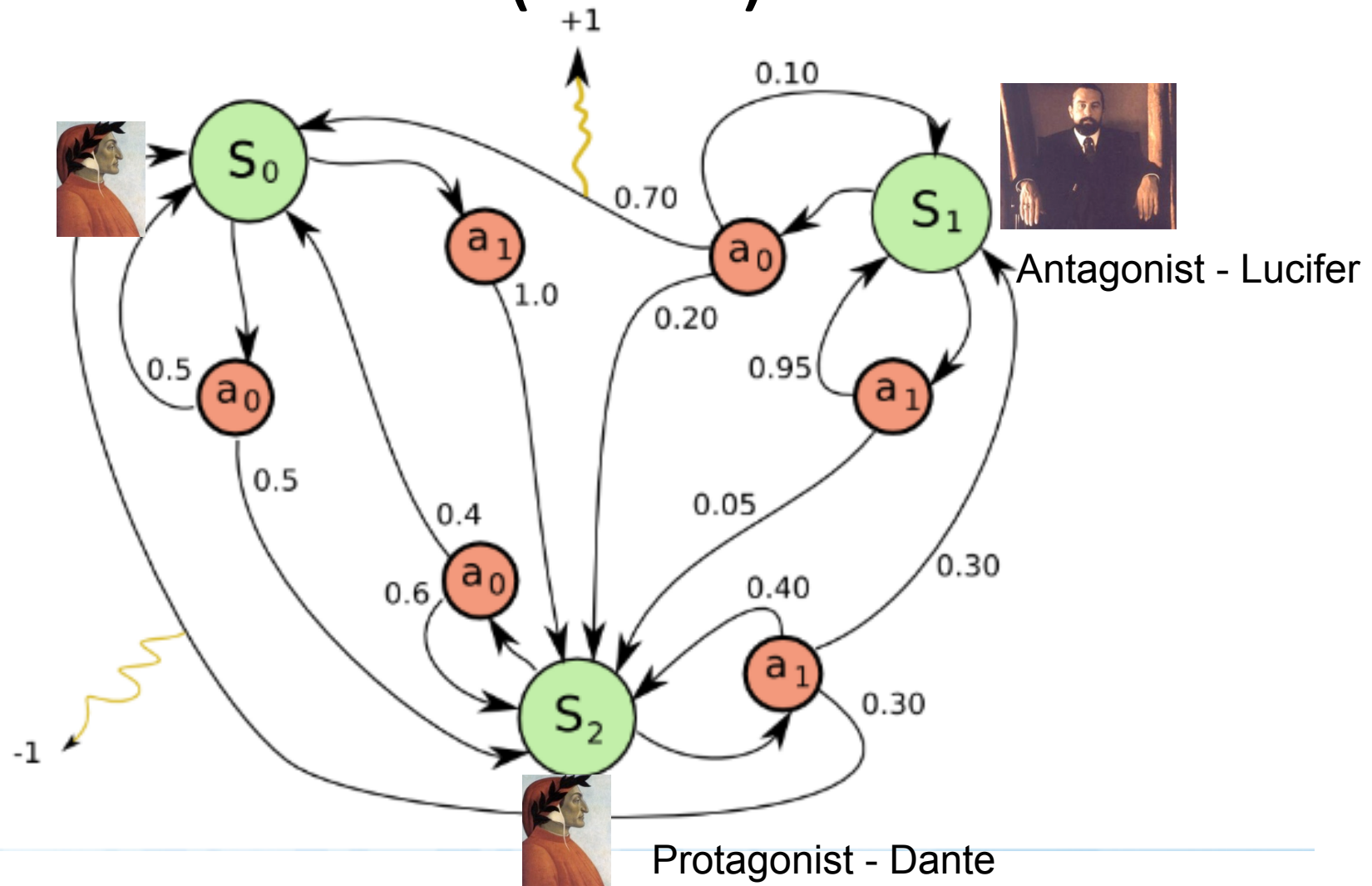
Even better bounds?

- $O(m)$ iterations?
- $O(n)$ iterations?
 - No! Hansen and Zwick, ISAAC 2010.

Markov Decision Process (MDP)



Turn-based Stochastic Game (TBSG)



Howard's algorithm (1960)

- Start with an arbitrary policy (a choice of action in each position)
- *Compute expected total payoff for each position.*
- For all positions, if a chosen action picks a lower expected payoff over a higher expected payoff, **switch**.
- Iterate!



Strategy iteration

- Start with an arbitrary strategy for the protagonist
- *Solve the resulting one-player game (MDP) for the antagonist. Compute expected total payoff for each position*
- For all positions, if a chosen action picks a lower expected payoff over a higher expected payoff, **switch**.
- Iterate!

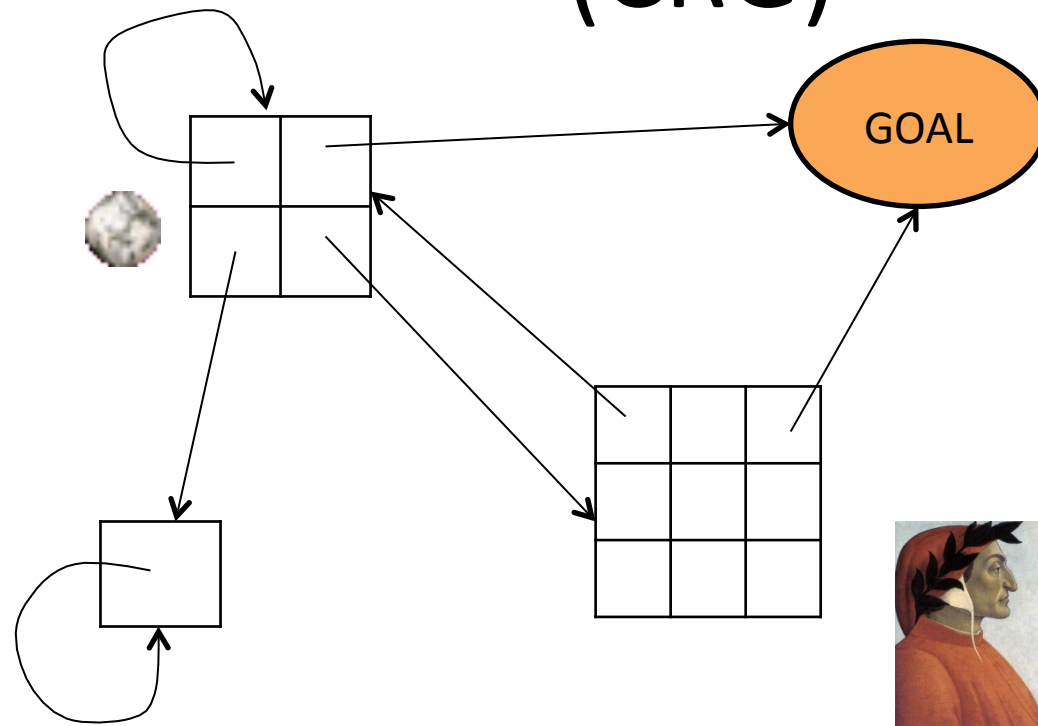
How many iterations?

- Hansen, Miltersen and Zwick, 2011:
 - $O(m \log n)$ iterations suffices, as for the case of MDPs!
 - This is the first strongly polynomial bound for **any** algorithm solving turn-based stochastic games.
 - “De-LPfication” of Ye’s analysis.

What if the discount factor is not fixed (or if the MDP is undiscounted)?

- Friedman, 2009:  (2^n) iterations for TBSGs.
- Fearnley, 2010:  (2^n) for MDPs
- Technique recently generalized by Friedman, Hansen and Zwick to show tight lower bounds for the ***Random Facet Algorithm***
 - for turn-based stochastic games (SODA'11)
 - **for linear programming! (upcoming)**

Concurrent Reachability Game (CRG)

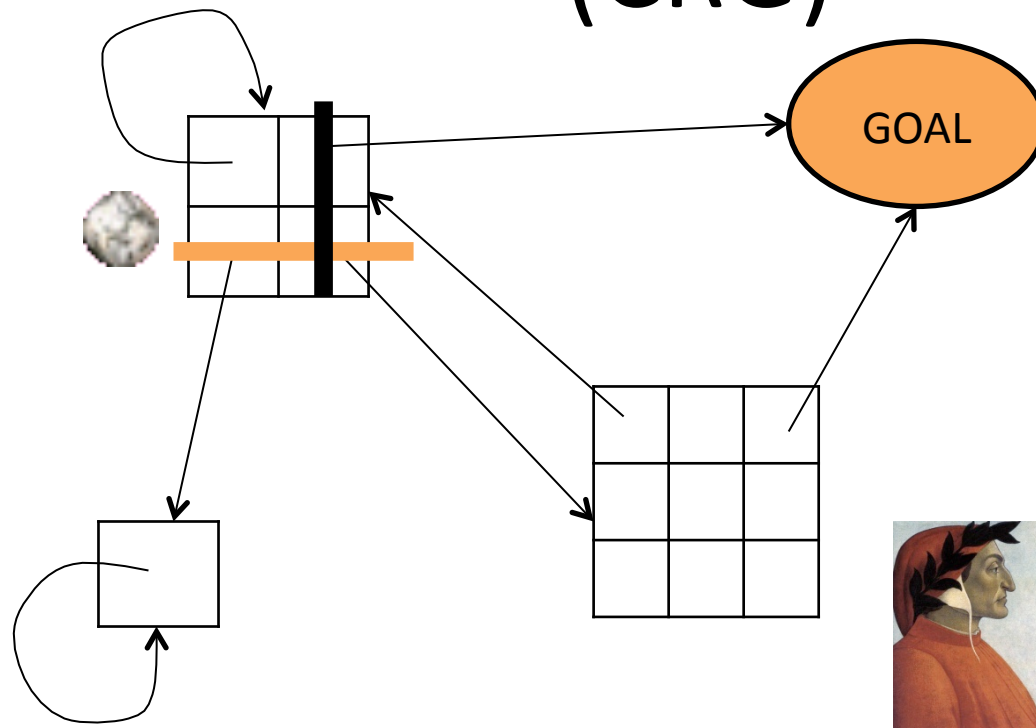


Dante - Row player
Wants to reach GOAL



Lucifer – Column player
Wants to prevent Dante
from reaching GOAL

Concurrent Reachability Game (CRG)

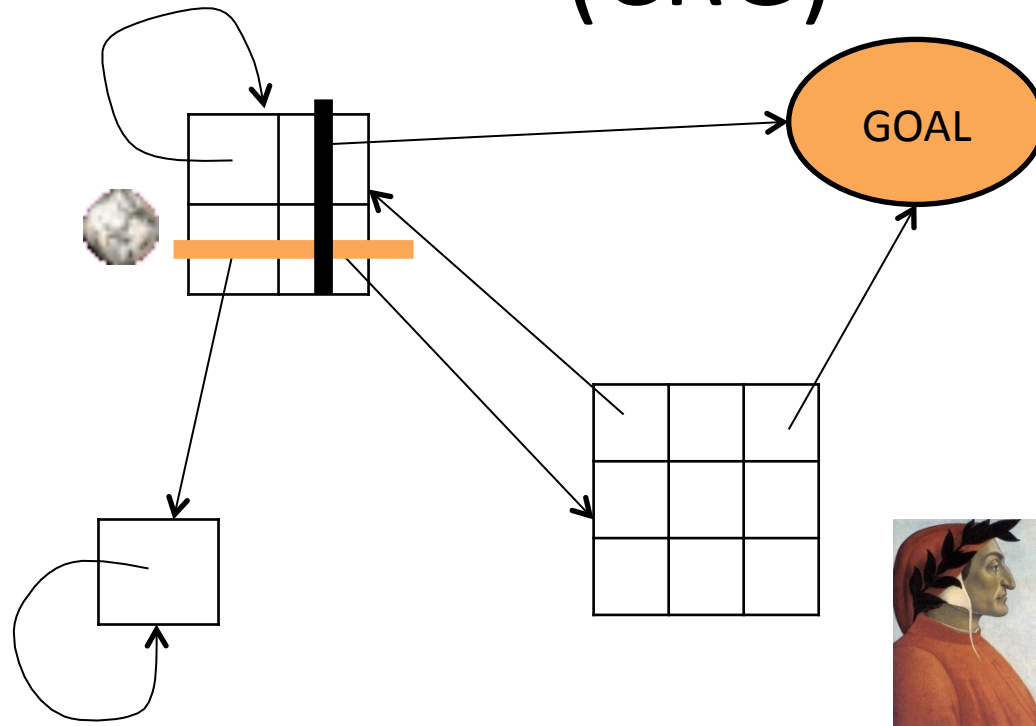


Dante - Row player
Wants to reach GOAL



Lucifer – Column player
Wants to prevent Dante
from reaching GOAL

Concurrent Reachability Game (CRG)

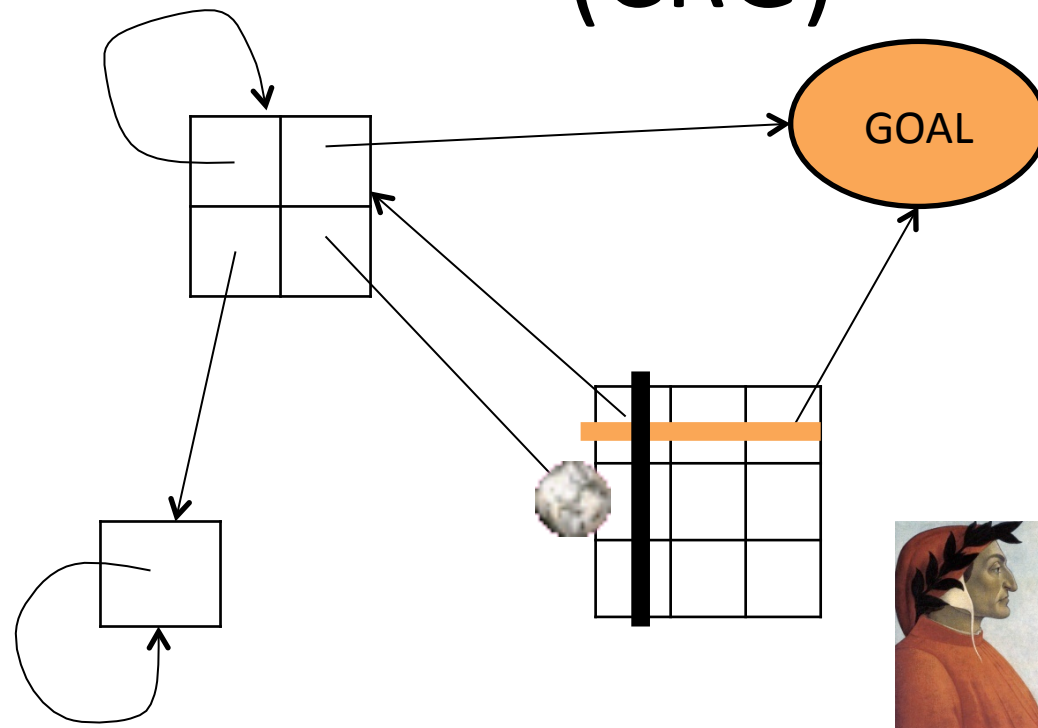


Dante - Row player
Wants to reach GOAL



Lucifer – Column player
Wants to prevent Dante
from reaching GOAL

Concurrent Reachability Game (CRG)

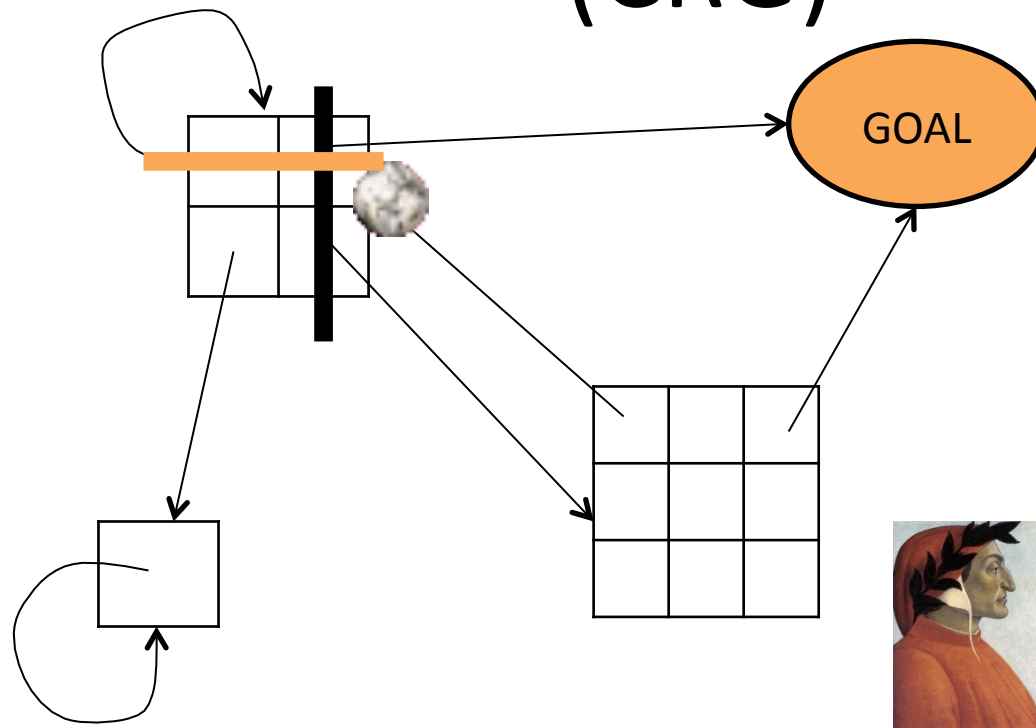


Dante - Row player
Wants to reach GOAL



Lucifer – Column player
Wants to prevent Dante
from reaching GOAL

Concurrent Reachability Game (CRG)



Dante - Row player
Wants to reach GOAL



Lucifer – Column player
Wants to prevent Dante
from reaching GOAL

Concurrent Reachability Game

- Arena:
 - Finite directed graph.
 - One terminal **GOAL** node, a terminal trap node, **N** non-terminal nodes.
 - Each non-terminal node contains an **m** x **m** *matrix* of outgoing arcs.
- Play:
 - A pebble moves from position to position.
 - In each step, Dante chooses a row and Lucifer *simultaneously* chooses a column of the matrix.
 - The pebble moves along the appropriate arc.
 - If Dante reaches the **GOAL** position he wins
 - If this *never* happens, Lucifer wins.

Why?

- Generalizes all problems we saw earlier
 - Not obvious – how to model rewards?
 - “generalizes” can be rigorously defined by the notion of polynomial reductions.
- The simplest case of two-player undiscounted imperfect information stochastic games.
- Models poker tournaments...

Values and Near-Optimal Strategies

- Each position i in a CRG has a *value* v_i so that

$$\begin{aligned} v_i &= \min_{\text{stationary } \mathbf{y}} \max_{\text{general } \mathbf{x}} \mu_i(\mathbf{x}, \mathbf{y}) \\ &= \text{sup}_{\text{stationary } \mathbf{x}} \min_{\text{general } \mathbf{y}} \mu_i(\mathbf{x}, \mathbf{y}) \end{aligned}$$

where $\mu_i(\mathbf{x}, \mathbf{y})$ is the probability of reaching GOAL when Dante plays by strategy \mathbf{x} and Lucifer plays by strategy \mathbf{y} .

Algorithmic problems

- ***Quantitatively*** solving a CRG.
 - Approximating the values of the nodes.
- ***Strategically*** solving a CRG.
 - Computing an ϵ -optimal stationary strategy for a given ϵ .

Howard's algorithm for CRGs

Chatterjee, de Alfaro, Henzinger '06

```
1:  $t := 1$  .
2:  $x^1 :=$  the uniform distribution at each position .
3: while true do
4:    $y^t :=$  an optimal best reply to  $x^t$ ; ← Solve Markov
5:   for  $i \in \{0, 1, 2, \dots, N, N + 1\}$  do Decision Process
6:      $v_i^t := \mu_i(x^t, y^t)$ 
7:   end for
8:    $t := t + 1$ 
9:   for  $i \in \{1, 2, \dots, N\}$  do
10:    if  $\text{val}(A_i(v^{t-1})) > v_i^{t-1}$  then
11:       $x_i^t := \text{maximin}(A_i(v^{t-1}))$  ← Solve matrix game
12:    else
13:       $x_i^t := x_i^{t-1}$ 
14:    end if
15:  end for
16: end while
```

Properties

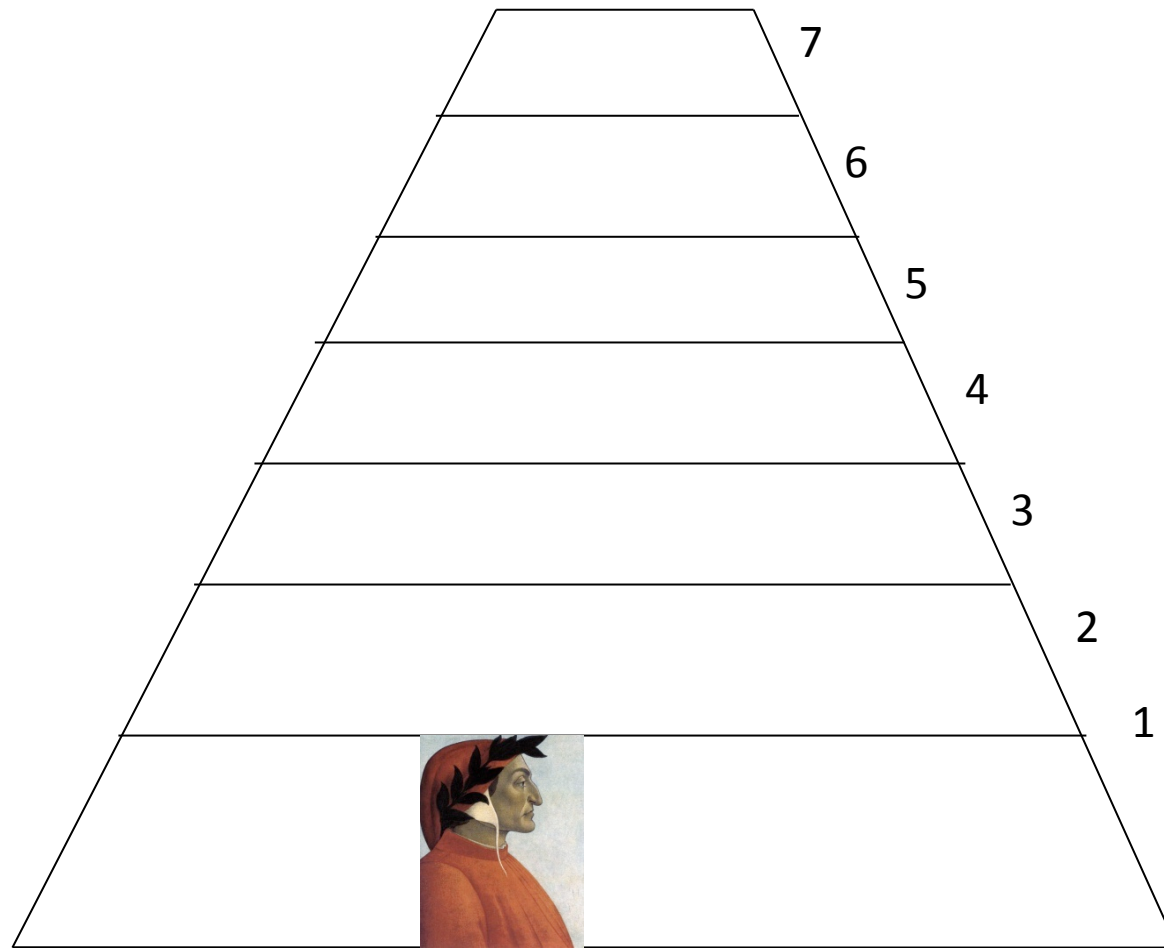
- The valuations v_i^t converge to the values v_i (from below).
- The strategies x^t guarantee the valuations v_i^t for Dante.
- What is the number of iterations required to guarantee a good approximation?

Hansen, Ibsen-Jensen, M., 2011

- Solving Concurrent Reachability Games using strategy iteration has worst case time complexity ***doubly exponential*** in size of the input.
- This is an upper and a lower bound. For games with N positions and m actions for each player in each position:
 - $(1/\epsilon)^{m^{N/4}}$ iterations are (sometimes) necessary to get ϵ -approximation of value.
 - $(1/\epsilon)^{2^{31} m N}$ iterations are always sufficient.

Dante in Purgatory

(Hansen, Koucky, Miltersen, LICS'09)

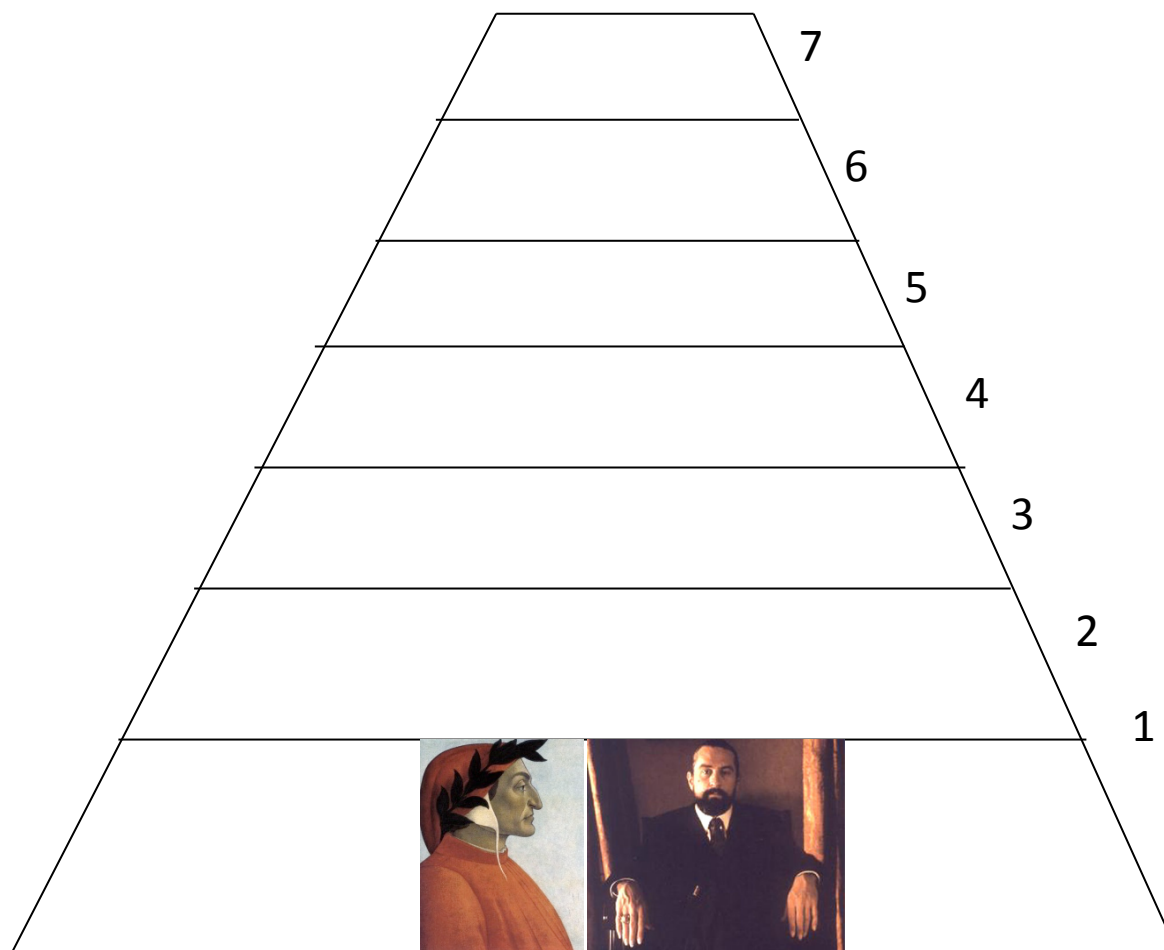


Purgatory has 7 terraces.

Dante enters Purgatory
at terrace 1.

Dante in Purgatory

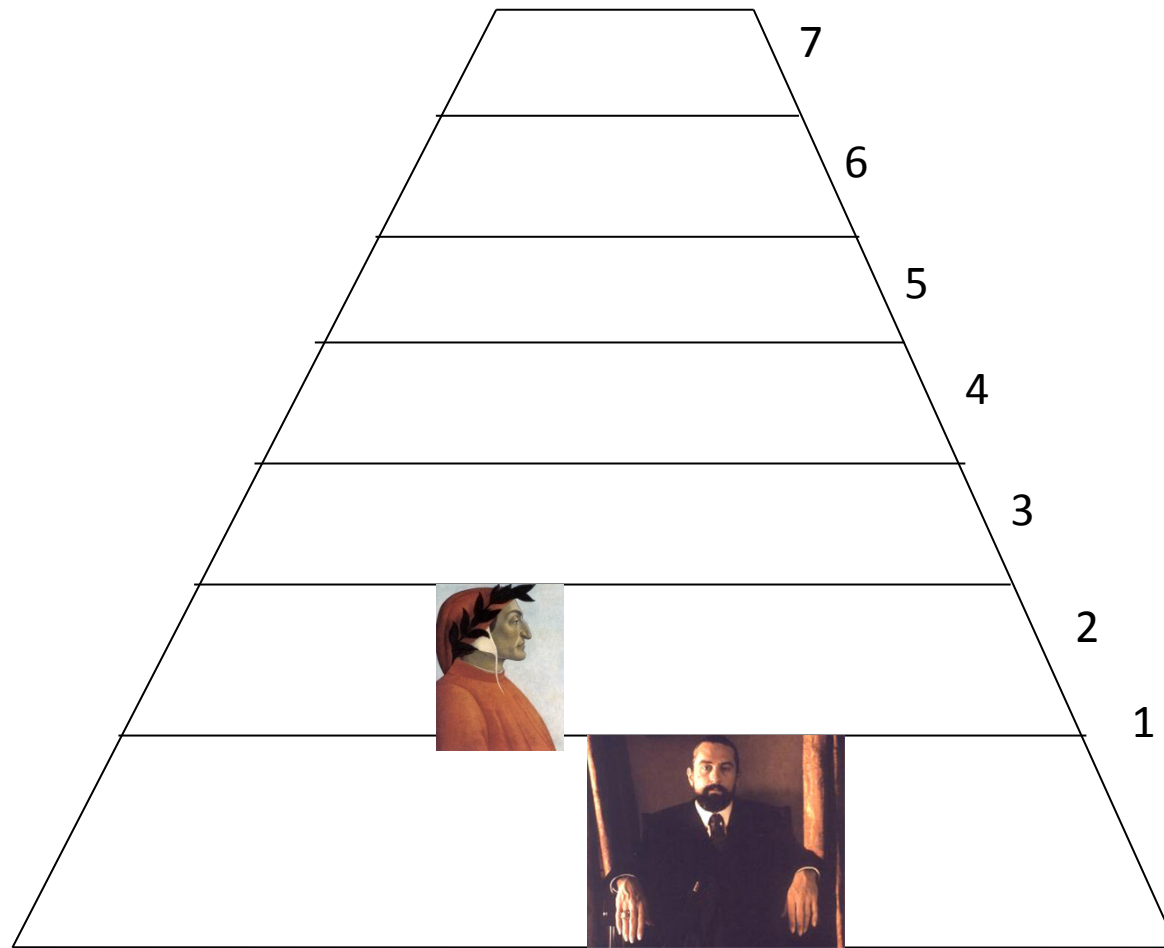
(Hansen, Koucky, Miltersen, LICS'09)



While in Purgatory, once a second, Dante must play Matching Pennies with Lucifer

Dante in Purgatory

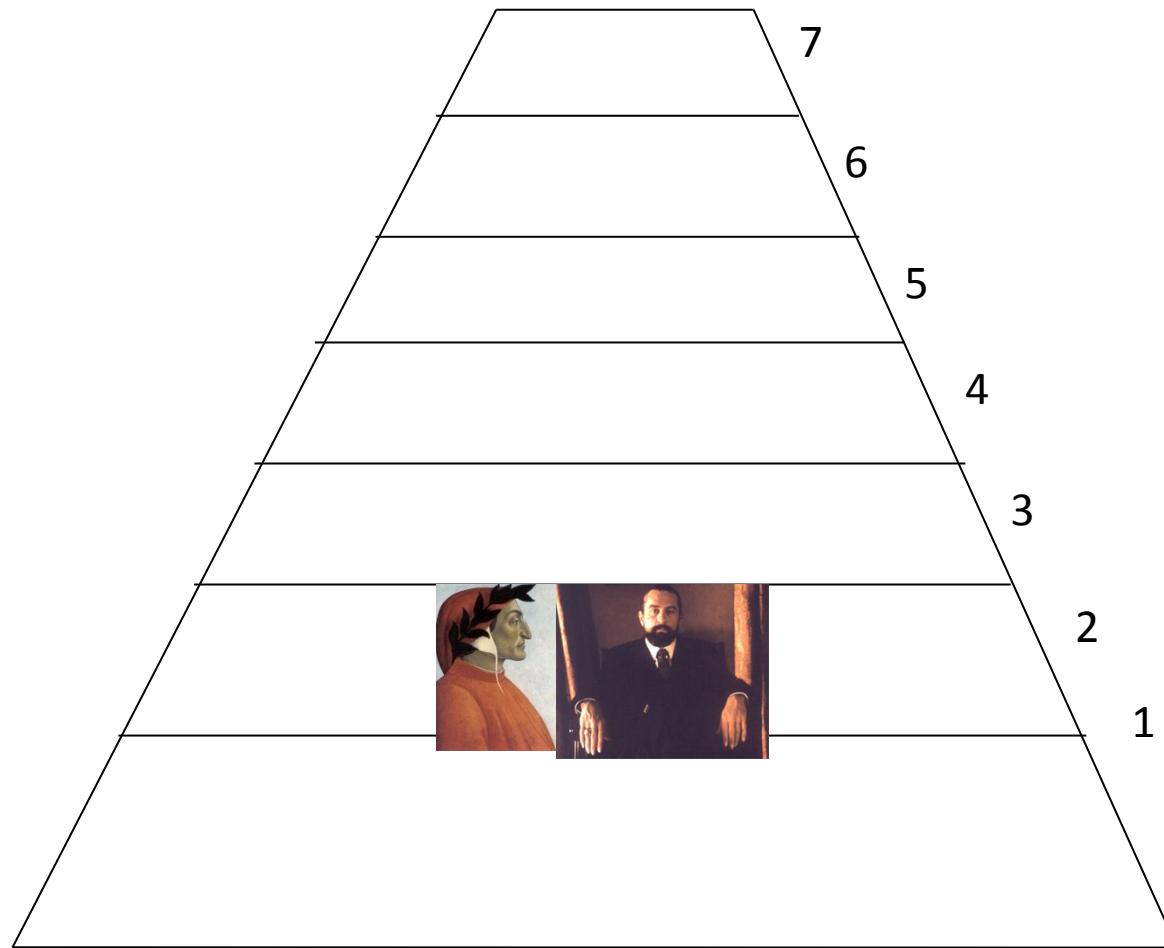
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds
to the next terrace

Dante in Purgatory

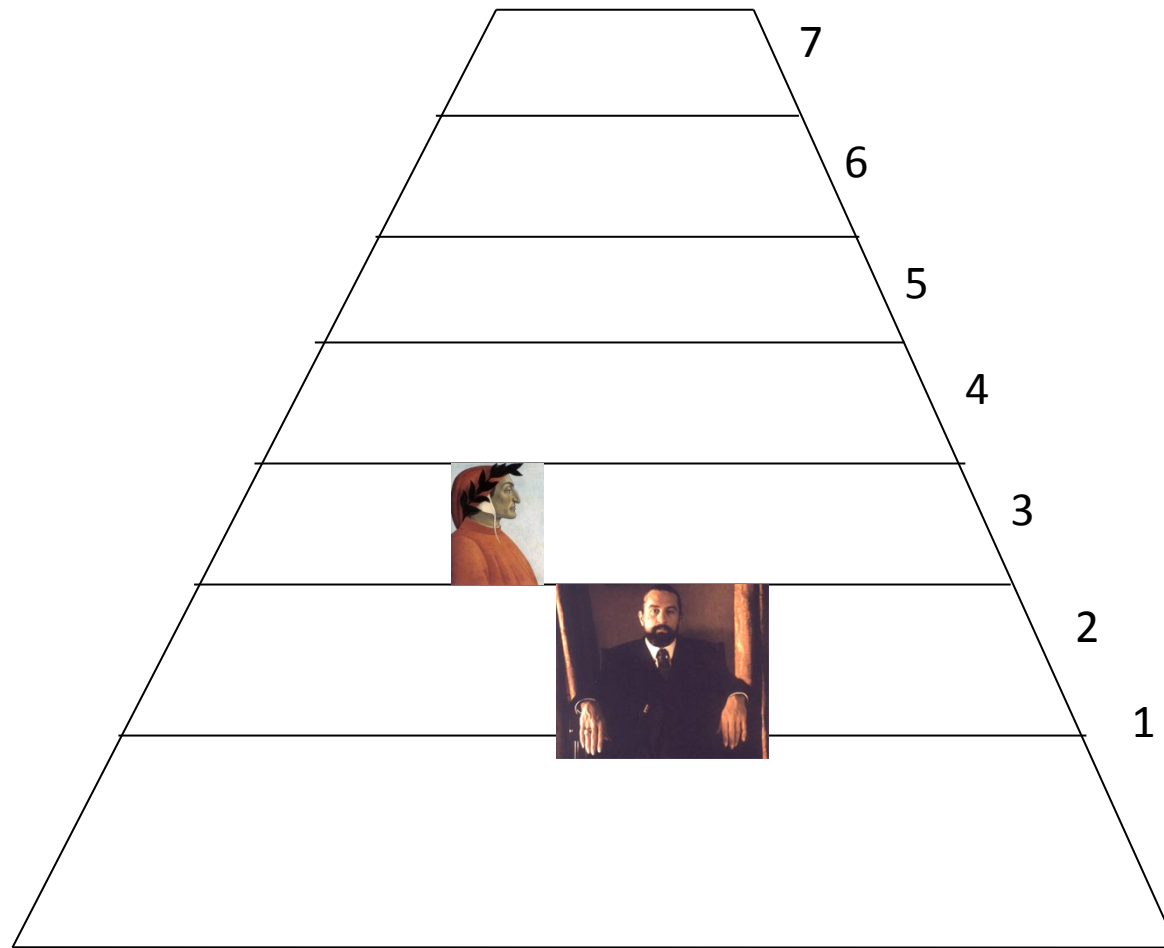
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds
to the next terrace

Dante in Purgatory

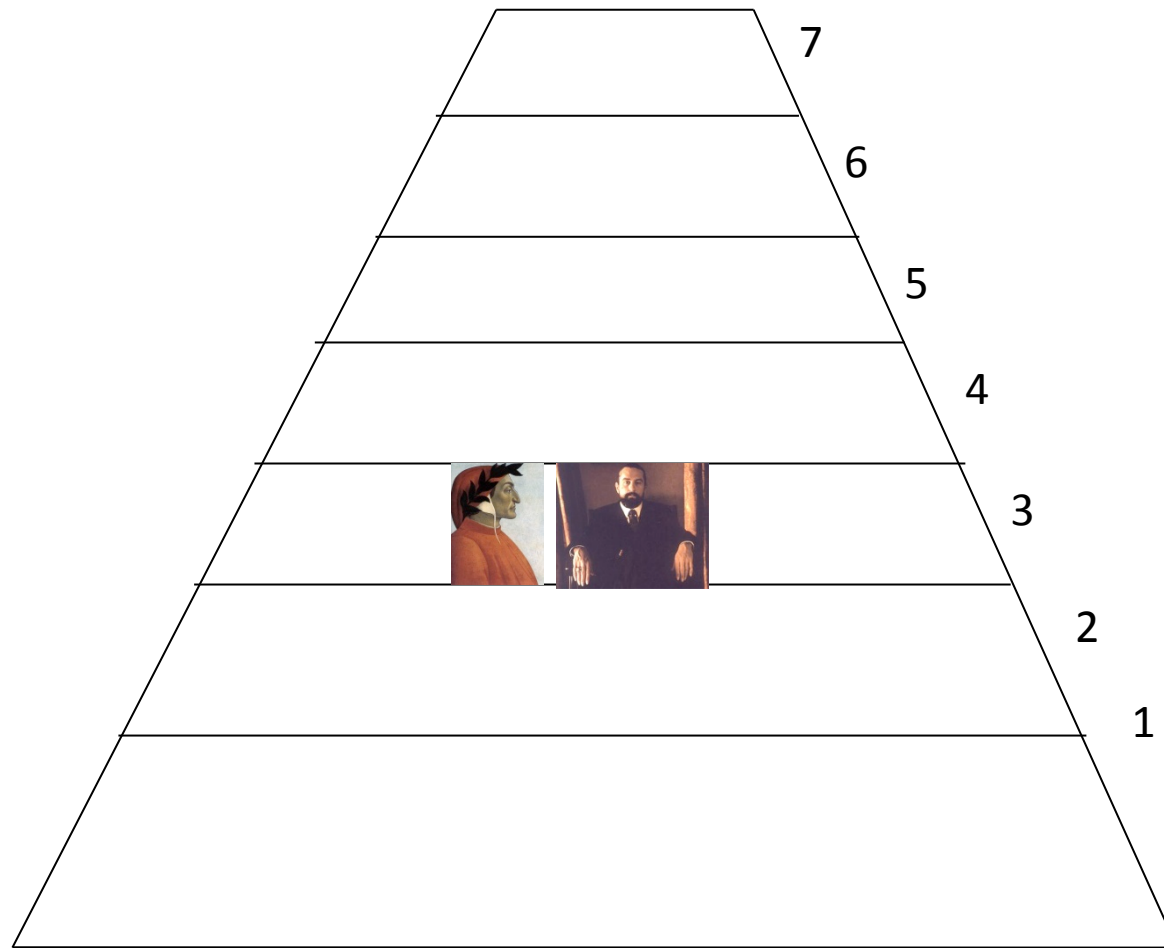
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds
to the next terrace

Dante in Purgatory

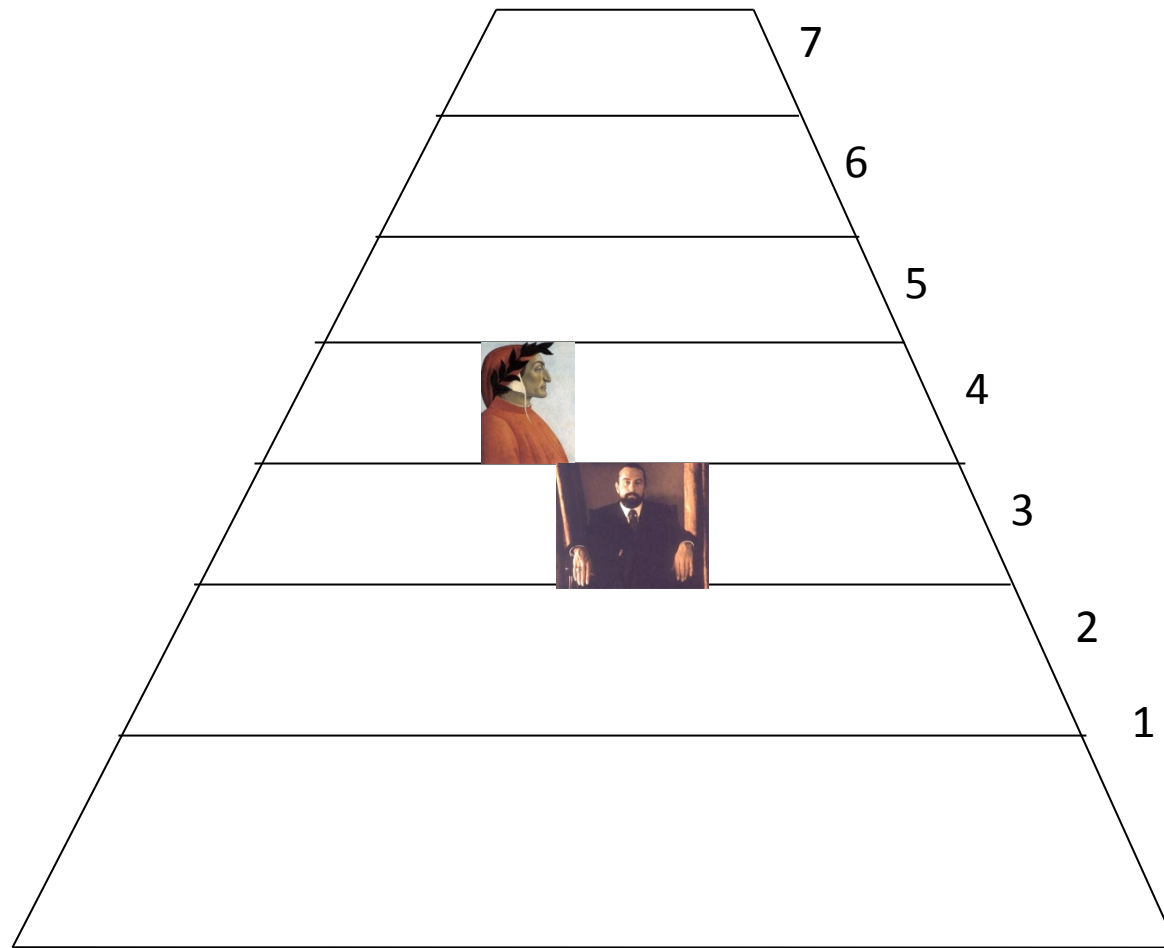
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds
to the next terrace

Dante in Purgatory

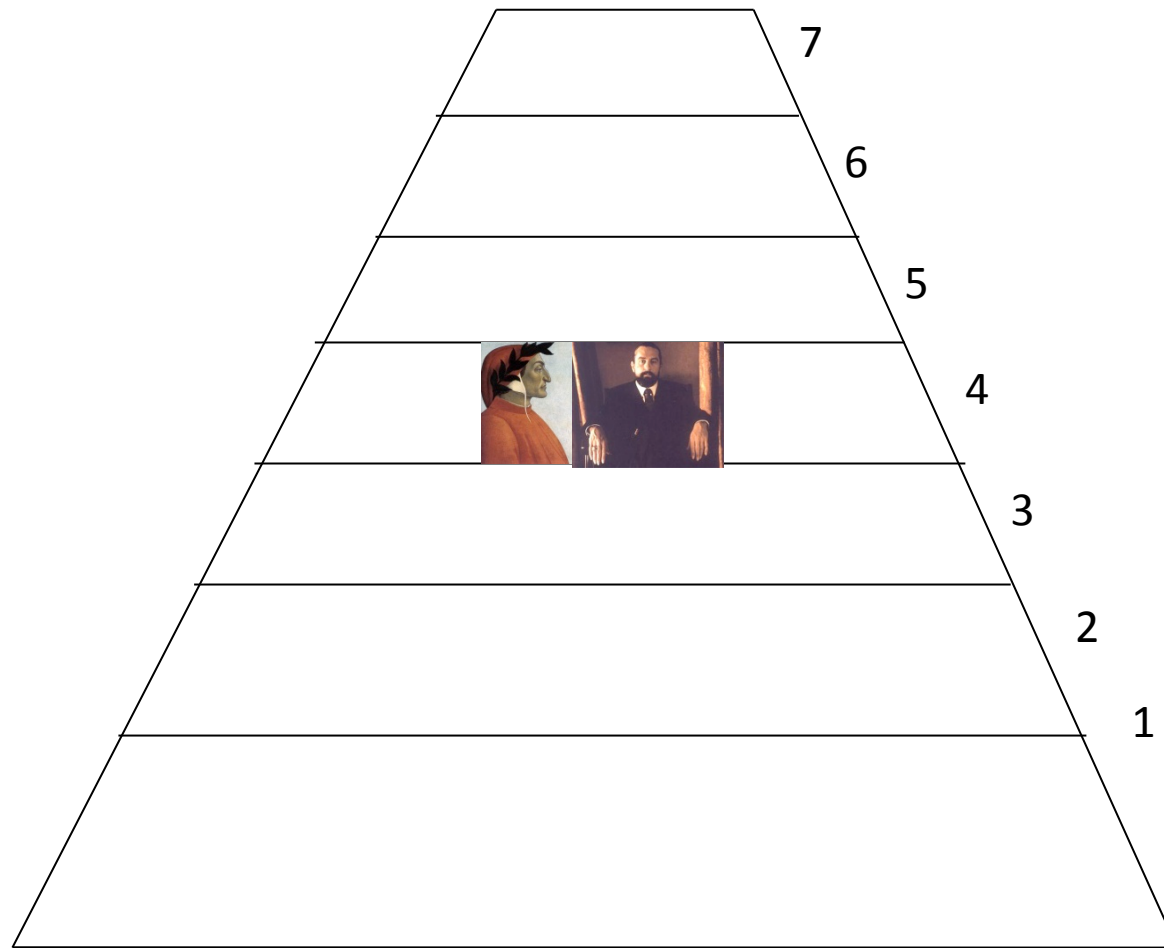
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds
to the next terrace

Dante in Purgatory

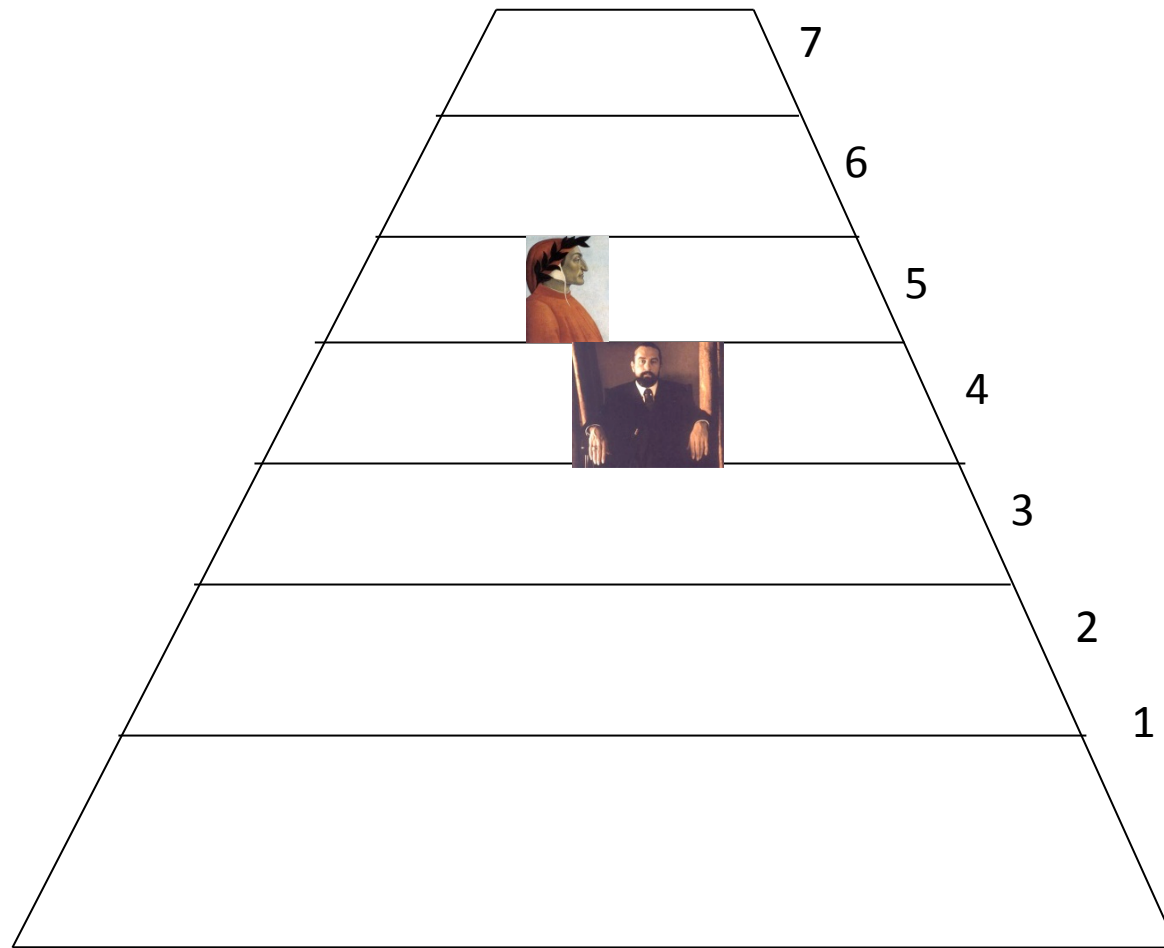
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds
to the next terrace

Dante in Purgatory

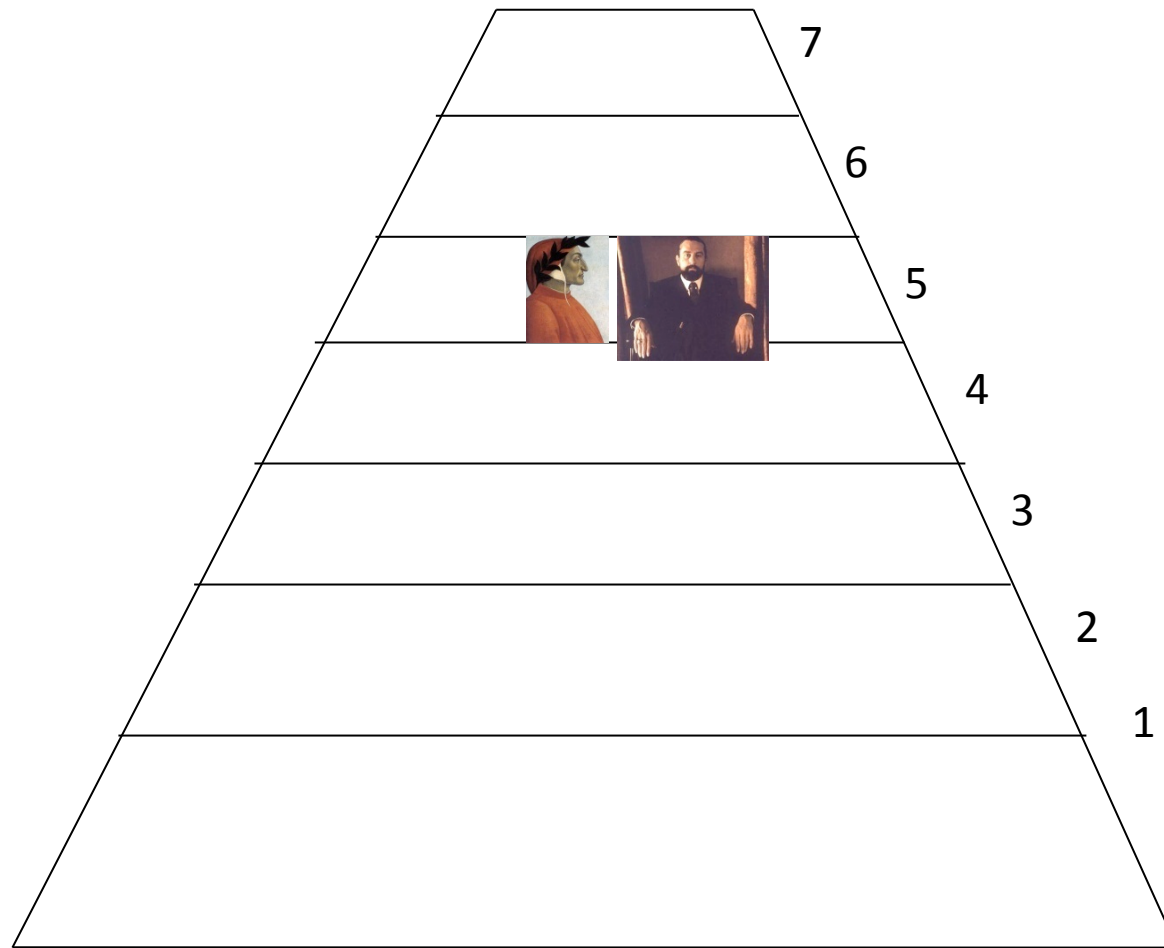
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds
to the next terrace

Dante in Purgatory

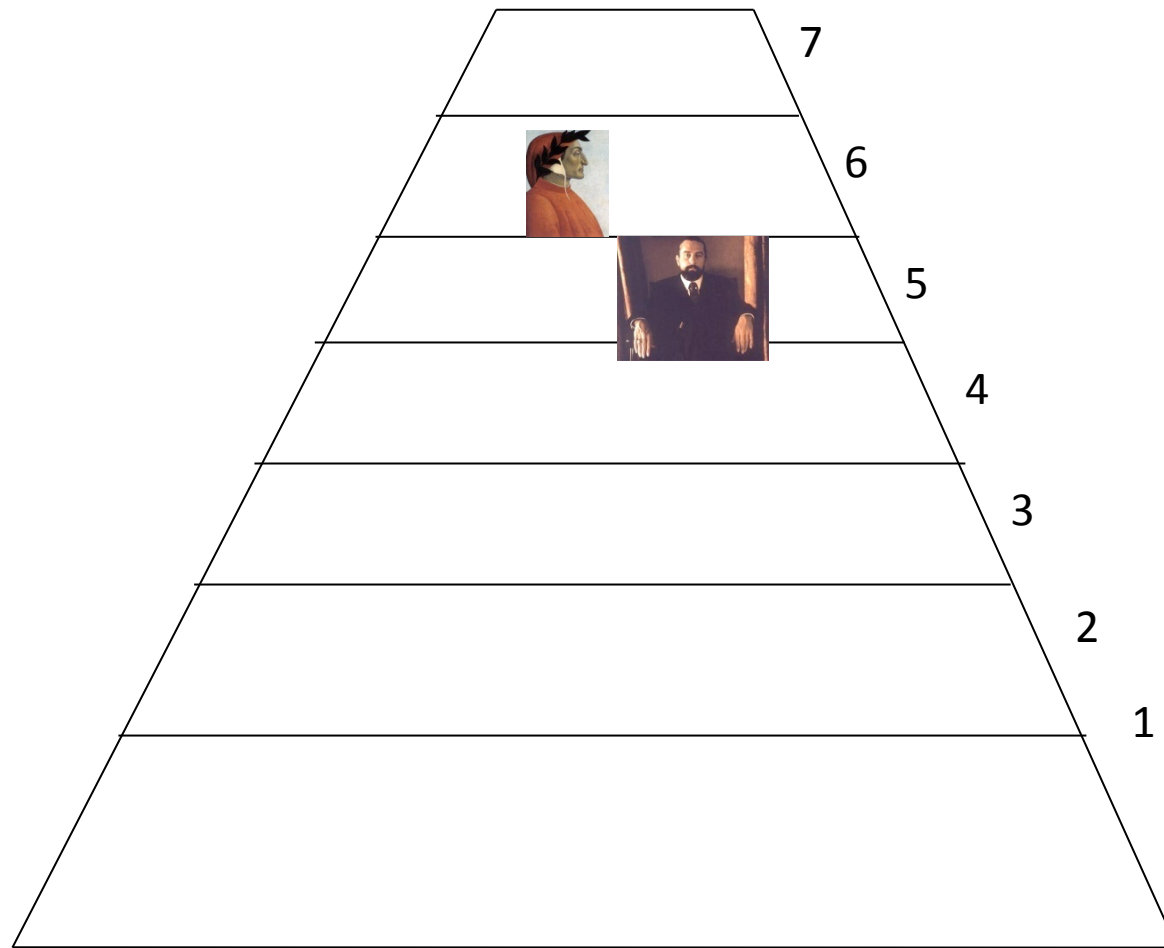
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds
to the next terrace

Dante in Purgatory

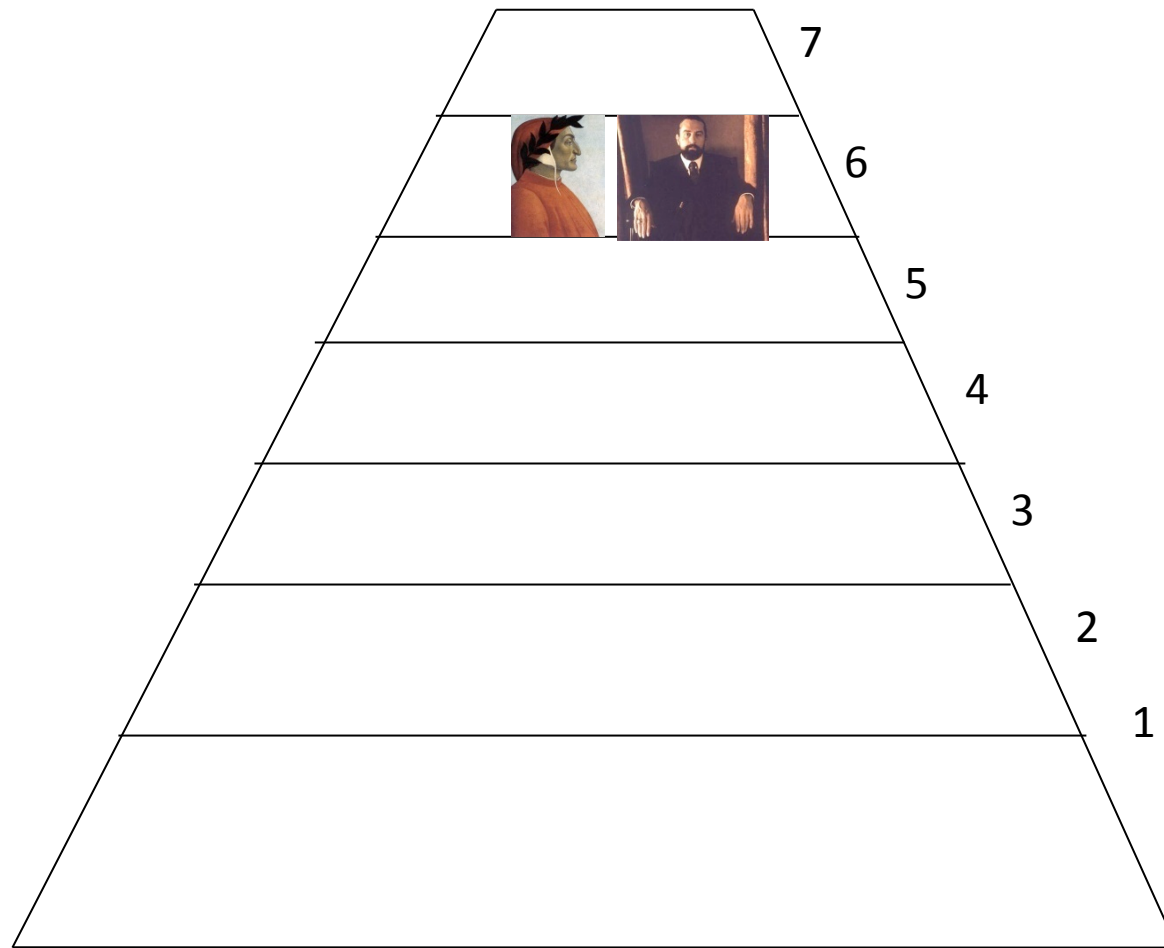
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds
to the next terrace

Dante in Purgatory

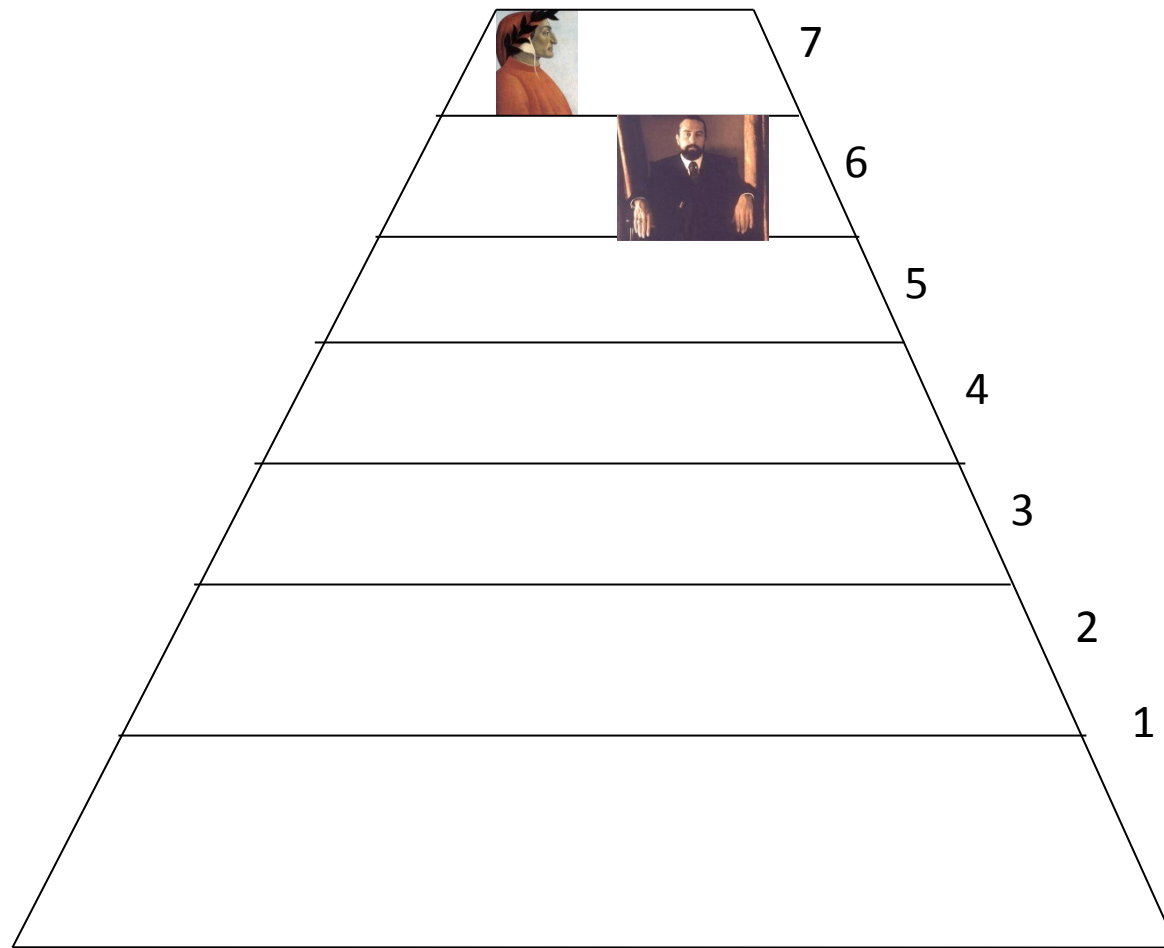
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds
to the next terrace

Dante in Purgatory

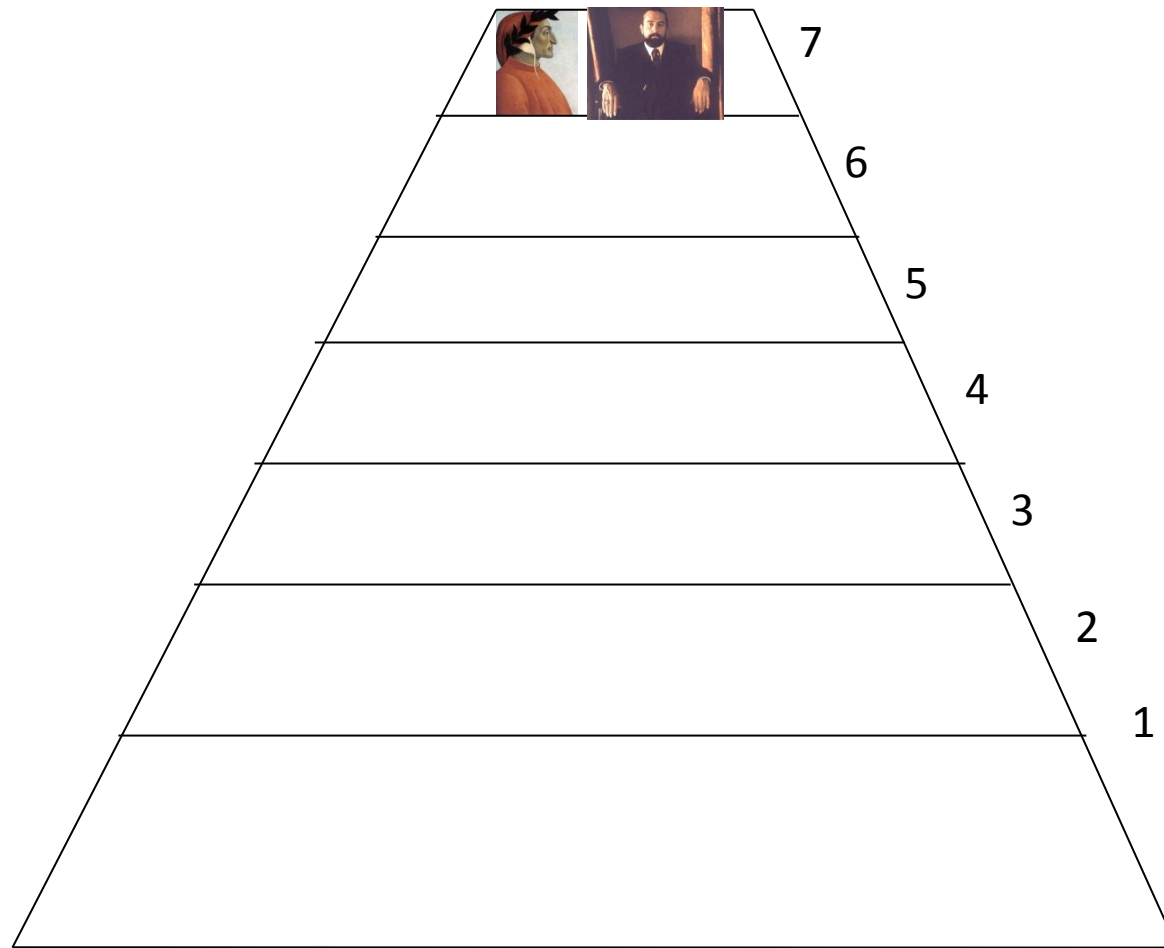
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds
to the next terrace

Dante in Purgatory

(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds
to the next terrace

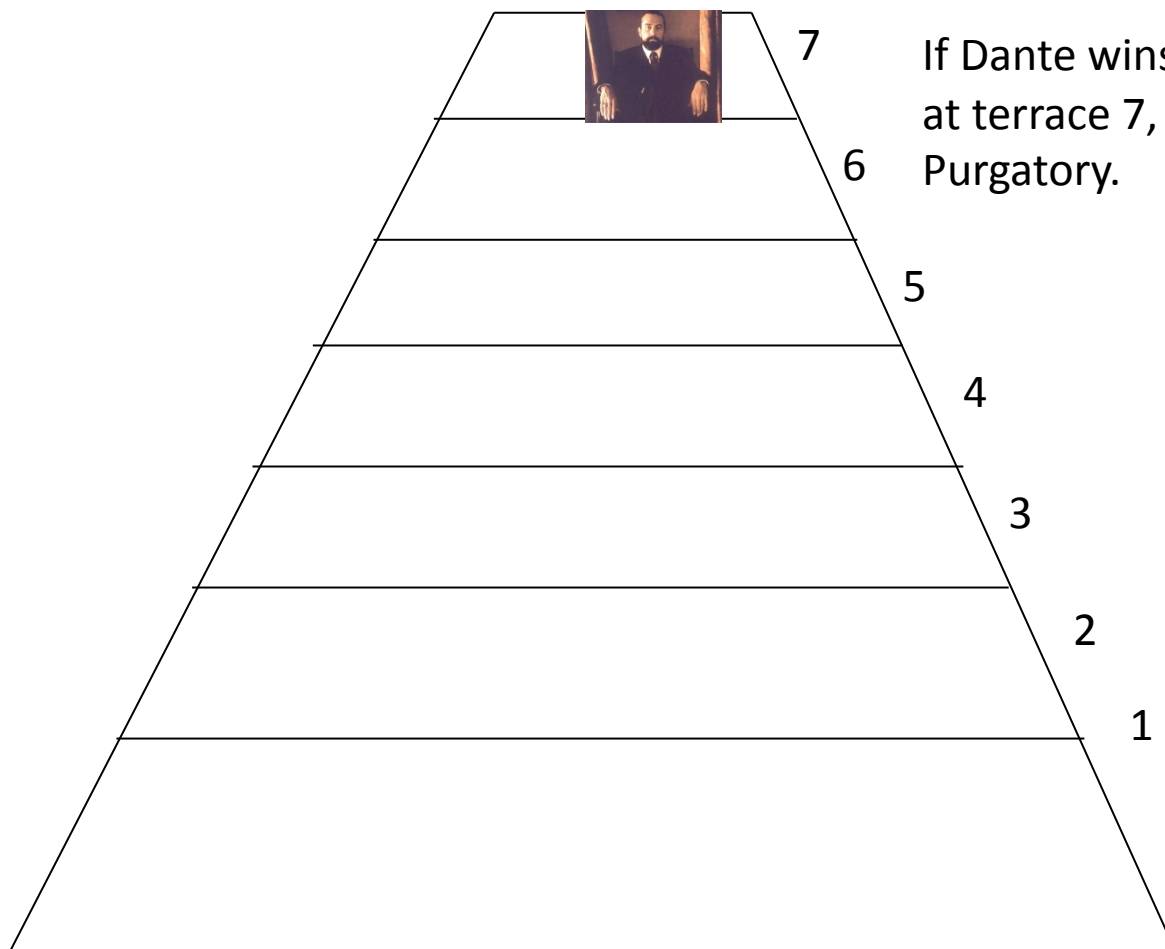
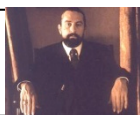
Dante in Purgatory

(Hansen, Koucky, Miltersen, LICS'09)



Dante in Purgatory

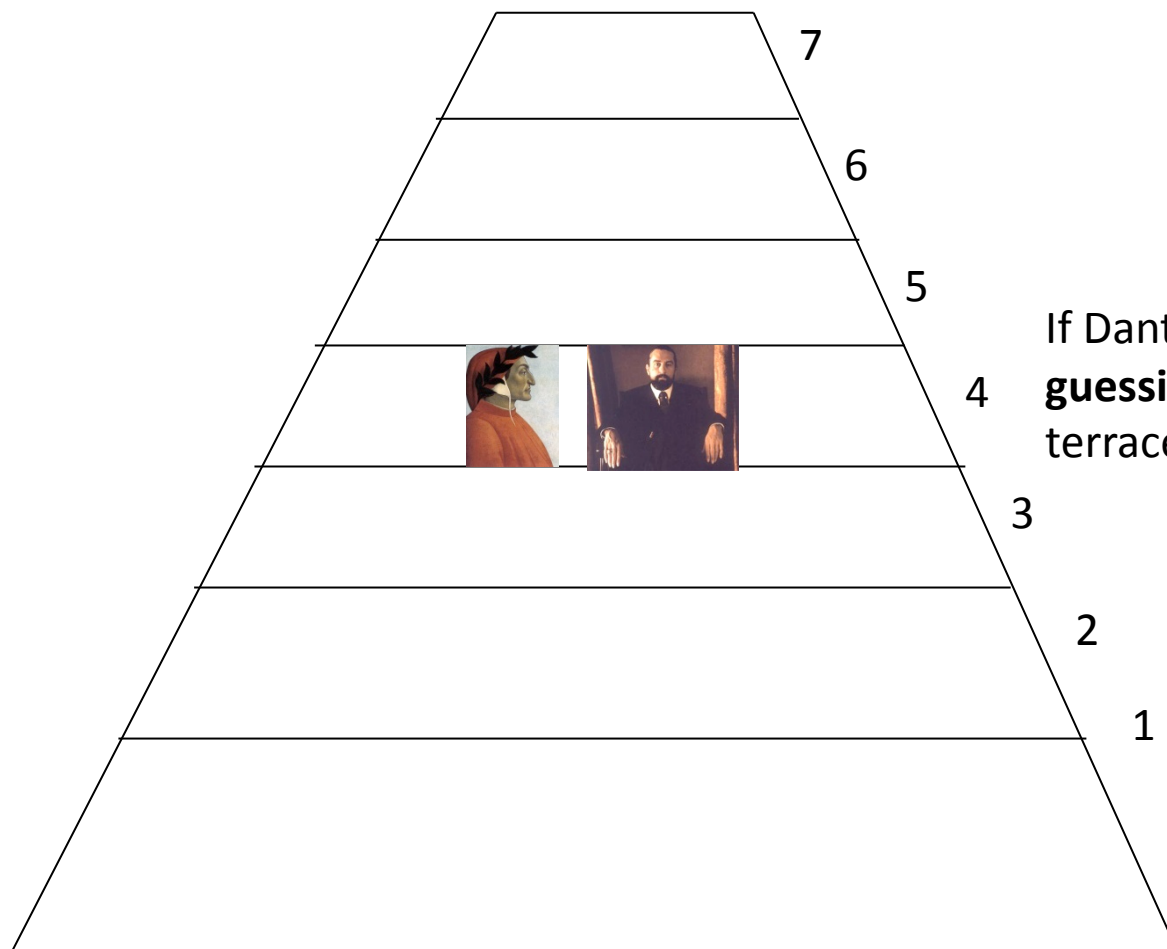
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins Matching Pennies at terrace 7, he wins the game of Purgatory.

Dante in Purgatory

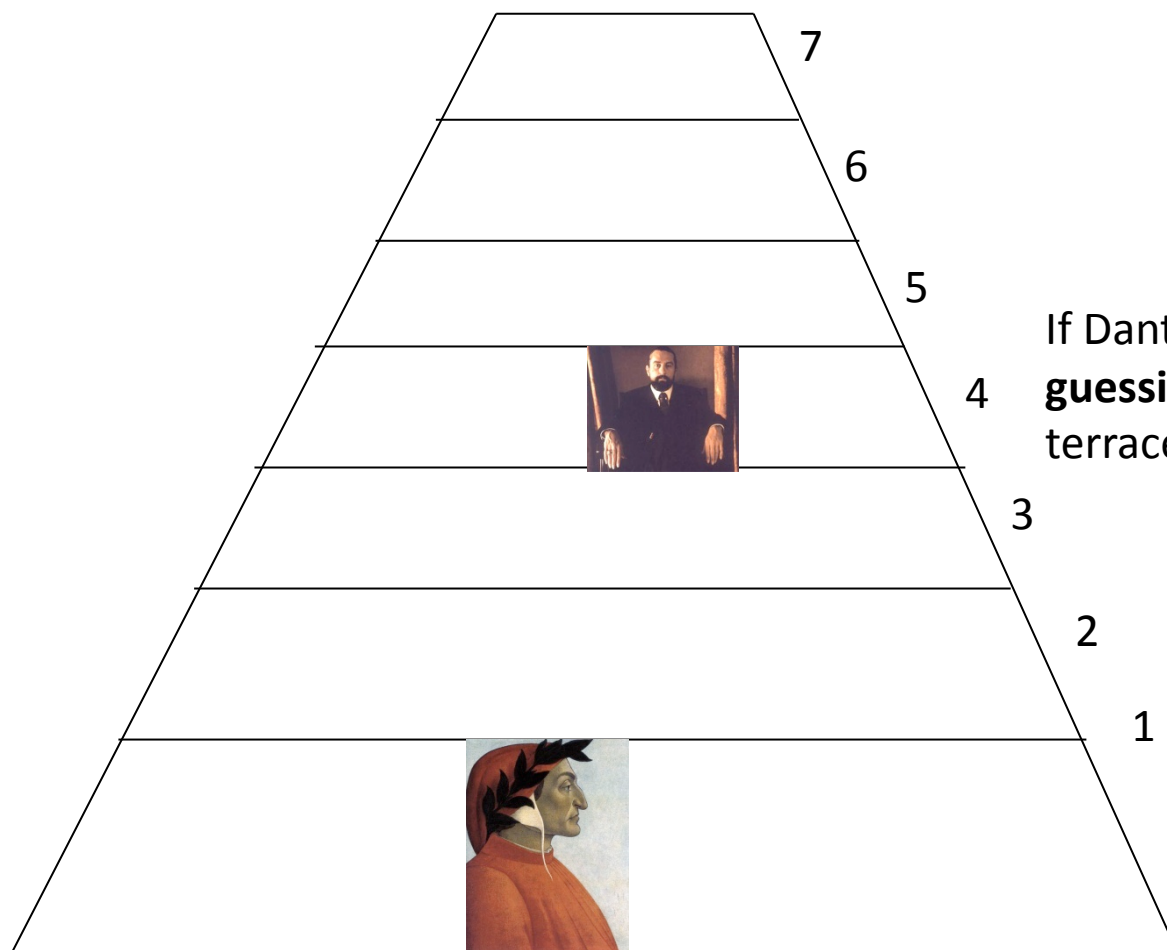
(Hansen, Koucky, Miltersen, LICS'09)



If Dante loses Matching Pennies
guessing Heads, he goes back to
terrace 1.

Dante in Purgatory

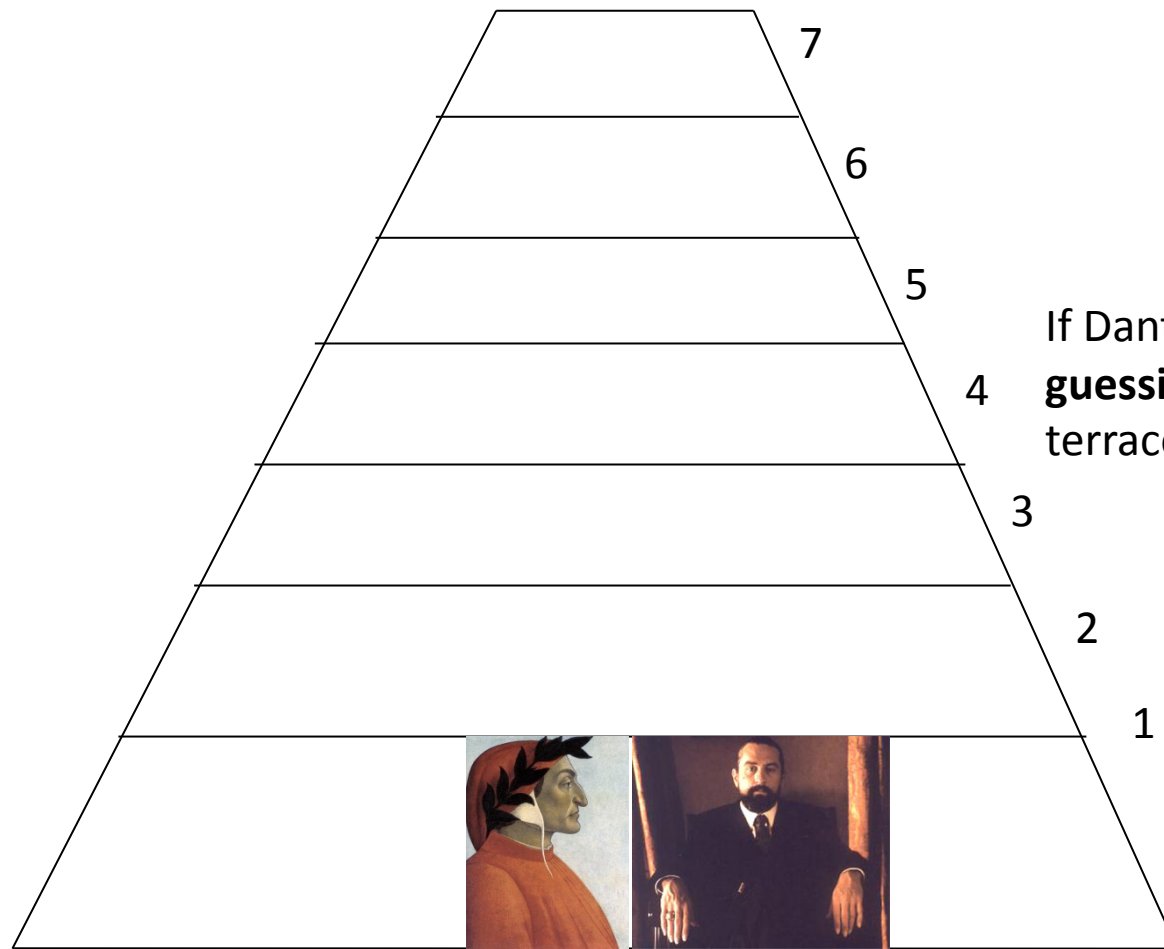
(Hansen, Koucky, Miltersen, LICS'09)



If Dante loses Matching Pennies
guessing Heads, he goes back to
terrace 1.

Dante in Purgatory

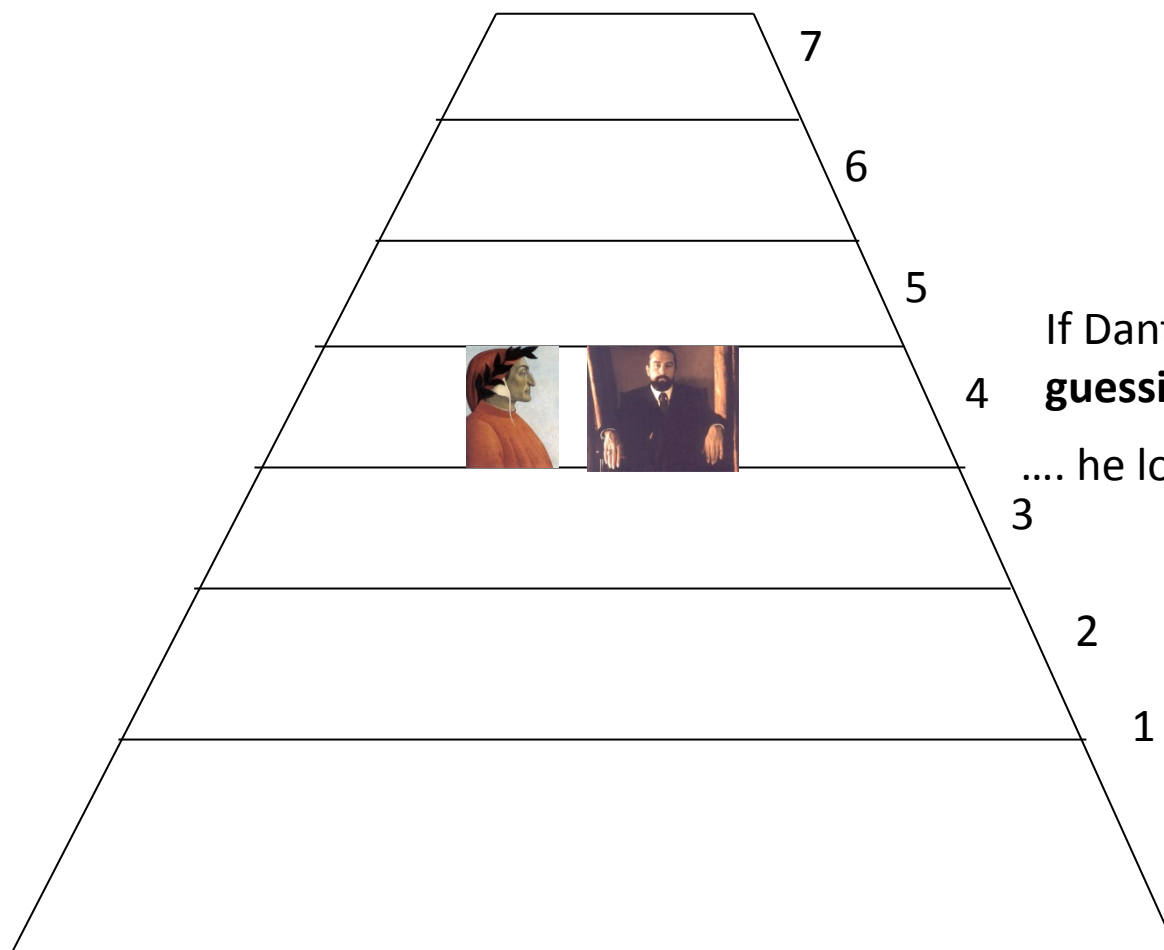
(Hansen, Koucky, Miltersen, LICS'09)



If Dante loses Matching Pennies
guessing Heads, he goes back to
terrace 1.

Dante in Purgatory

(Hansen, Koucky, Miltersen, LICS'09)



If Dante loses Matching Pennies
guessing Tails.....

.... he loses the game of Purgatory!!!!

Dante in Purgatory

- Is there is a strategy for Dante so that he is guaranteed to win the game of Purgatory with probability at least 90%?
 - Yes! A bit surprising – when Dante wins, he has guessed correctly seven times in a row!

Purgatory is a game of *doubly exponential patience* (Hansen, Koucky, M., LICS'09).

- The **patience** of a stationary strategy is $1/p$ where p is the smallest non-zero probability used by the strategy (Everett, 1953).
- To win with probability $1-\epsilon$, Dante must choose “Heads” at terrace i with probability greater than (roughly)
 $1 - \epsilon^{2^{7-i}}$
- On the other hand, choosing “Heads” with probability 1 is no good!
- To win with probability $9/10$, he must choose “Heads” at terrace 1 with probability greater than $1-(1/10)^{64} =$
0.999
99999999999.
- Note that Lucifer can respond by always choosing “Tails” at terrace 1, making the play take **very** long time.

New: Strategy iteration is slow on Purgatory

#iterations:	Valuation of lowest terrace:
1	0.01347
10	0.03542
100	0.06879
1000	0.10207
10000	0.13396
100000	0.16461
1000000	0.19415
10000000	0.22263
100000000	0.24828
$> 2 \cdot 10^{65}$	0.9
$> 10^{128}$	0.99

Main result

- For games with N positions and m actions for each player in each position:
 - $(1/\varepsilon)^{m^{N/4}}$ iterations are (sometimes) necessary to get ε -approximation of value.
 - $(1/\varepsilon)^{2^{31} m N}$ iterations are always sufficient.
- For the lower bound, we generalize Purgatory to more than 2 actions..

Generalized Purgatory $P(N,m)$

- Lucifer repeatedly hides a number between 1 and m .
- Dante must try to guess the number.
- If he guesses correctly N times in a row, he wins the game.
- If he ever guesses incorrectly ***overshooting*** Lucifer's number, he loses the game.

A bit about the proof

A bit about the proof

- As for the case of MDPs we can relate the valuations computed by strategy iteration to the valuations computed by ***value iteration***.

The diagram illustrates the relationship between three types of valuations for a state i at time t . It features the inequality $\tilde{v}_i^t \leq v_i^t \leq v_i$. Three yellow boxes with arrows point to the terms: 'Valuations computed by value iteration' points to \tilde{v}_i^t , 'Valuations computed by strategy iteration' points to v_i^t , and 'Actual values' points to v_i . Blue arrows also point from the boxes to the terms they describe.

$$\tilde{v}_i^t \leq v_i^t \leq v_i$$

Valuations computed by value iteration

Valuations computed by strategy iteration

Actual values

Value iteration (dynamic programming)

```
1:  $t := 0$ 
2:  $\tilde{v}^0 := (0, 0, \dots, 1)$  {the vector  $\tilde{v}^0$  is indexed  $0, 1, \dots, N, N + 1$ }
3: while true do
4:    $t := t + 1$ 
5:    $\tilde{v}_0^t := 0$ 
6:    $\tilde{v}_{N+1}^t := 1$ 
7:   for  $i \in \{1, 2, \dots, N\}$  do
8:      $\tilde{v}_i^t := \text{val}(A_i(\tilde{v}^{t-1}))$ 
9:   end for
10: end while
```

Value iteration computes the value of the time bounded game,
for larger and larger values of the time bound, by **backward induction**.

Why value iteration is slow on Purgatory (sketch!)

- The valuations computed after t iterations are the actual values of the game with time bound t .
- We know that to win Purgatory with any significant probability, Dante must be very patient (use very small probabilities).
- This means that Lucifer can make play take a very long time.
- This means that Dante cannot win the time-bounded game with any good probability.
- This means the valuations computed by value iteration are far from the correct values. **QED!**

(the above "proof" cheats slightly by blurring the distinction between stationary strategies and arbitrary ones)

Why strategy iteration is slow on Purgatory (sketch!)

- Strategy iteration on Purgatory with n terraces ***compute the same*** sequence of strategies for the lowest terrace as strategy iteration on Purgatory with ***one*** terrace only.
- Strategy iteration and value iteration are ***in synch*** when applied to Purgatory with one terrace.
- This allows us to conclude that the patience of the strategy computed after few iterations is low.
- We already know that strategies of low patience do not do well for Purgatory, so the strategies computed are not very good. **QED!**

Upper bound

- For any CRG with n positions and m actions for each player in each position, $(1/2)^{2^{31mn}}$ iterations are sufficient to achieve 2 -optimal strategy.
- Proof sketch: Express that the value of the time bounded game approximates the unrestriced game in the first order theory of the reals and appeal to the model theory of that logic.

Conclusion

- Howard's algorithm solves **discounted MDPs and turn based stochastic games** really fast – number of iterations close to linear!
- Howard's algorithm solves **undiscounted MDPs** relatively slow – worst case number of iterations exponential.
- Howard's algorithm solves **Concurrent Reachability Games** really slowly – number of iterations doubly exponential!