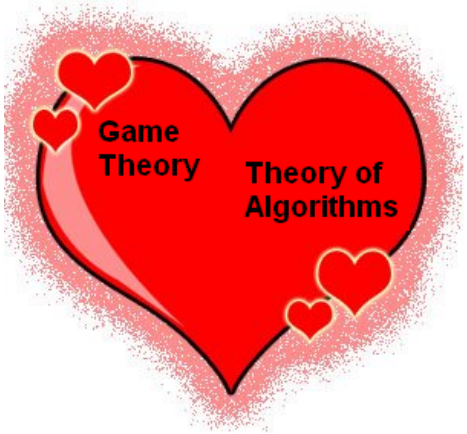
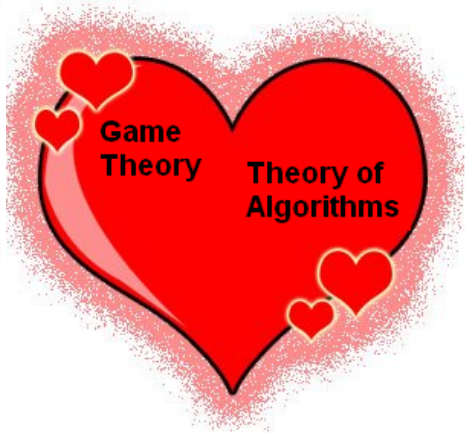
Day 1:



Center for
Algorithmic Game Theory

Day 2:



Center for
Algorithmic Game Theory

Shapley's stochastic games

STOCHASTIC GAMES*

BY L. S. SHAPLEY

PRINCETON UNIVERSITY

Communicated by J. von Neumann, July 17, 1953

Introduction.—In a stochastic game the play proceeds by steps from position to position, according to transition probabilities controlled jointly by the two players. We shall assume a finite number, N, of positions, and finite numbers m_k, n_k of choices at each position; nevertheless, the game may not be bounded in length. If, when at position k, the players choose their ith and jth alternatives, respectively, then with probability $s_{ij}^k > 0$ the game stops, while with probability p_{ij}^{kl} the game moves to position l. Define

$$s = \min_{k,i,j} s_{ij}^k.$$

Since s is positive, the game ends with probability 1 after a finite number of steps, because, for any number t, the probability that it has *not* stopped after t steps is not more than $(1 - s)^t$.

Payments accumulate throughout the course of play: the first player takes a_{ij}^k from the second whenever the pair i, j is chosen at position k. If we define the bound M:

$$M = \max_{k,i,j} |a_{ij}^k|,$$

then we see that the expected total gain or loss is bounded by

$$M + (1 - s)M + (1 - s)^2M + \ldots = M/s. \qquad (1)$$

The process therefore depends on $N^2 + N$ matrices

$$P^{kl} = (p_{ij}^{kl} | i = 1, 2, \ldots, m_k; j = 1, 2, \ldots, n_k)$$

$$A^k = (a_{ij}^k | i = 1, 2, \ldots, m_k; j = 1, 2, \ldots, n_k),$$

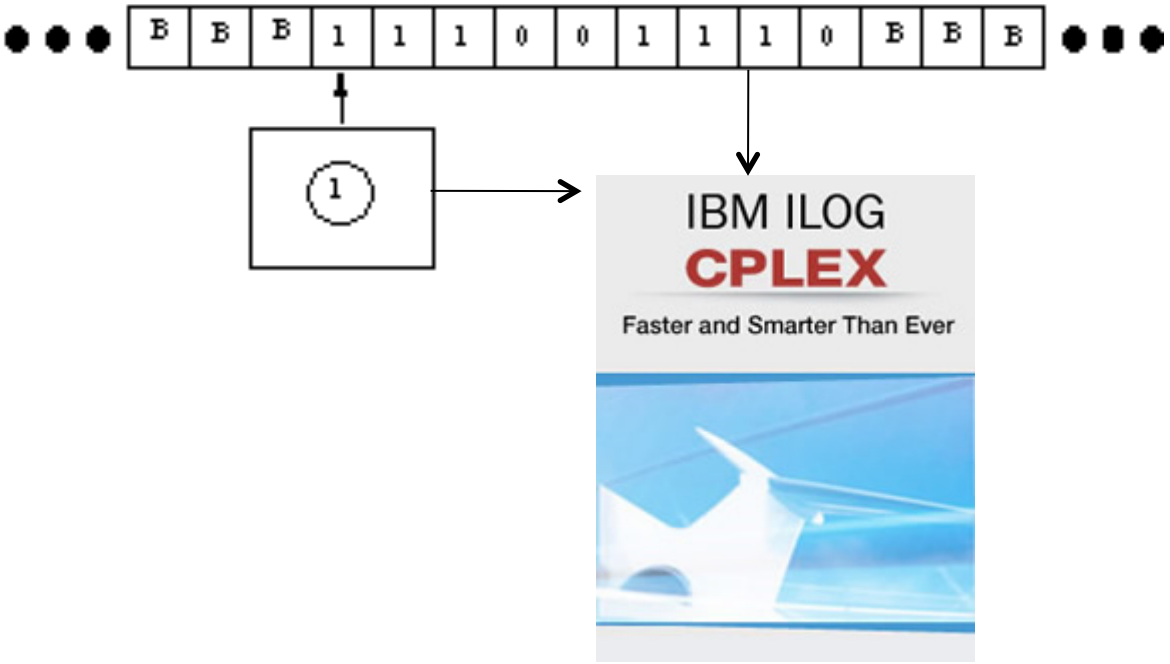
with $k, l = 1, 2, \ldots, N$, with elements satisfying

$$p_{ij}^{kl} \geq 0, \ |a_{ij}^k| \leq M, \ \sum_{l=1}^{N} p_{ij}^{kl} = 1 - s_{ij}^k \leq 1 - s < 1.$$

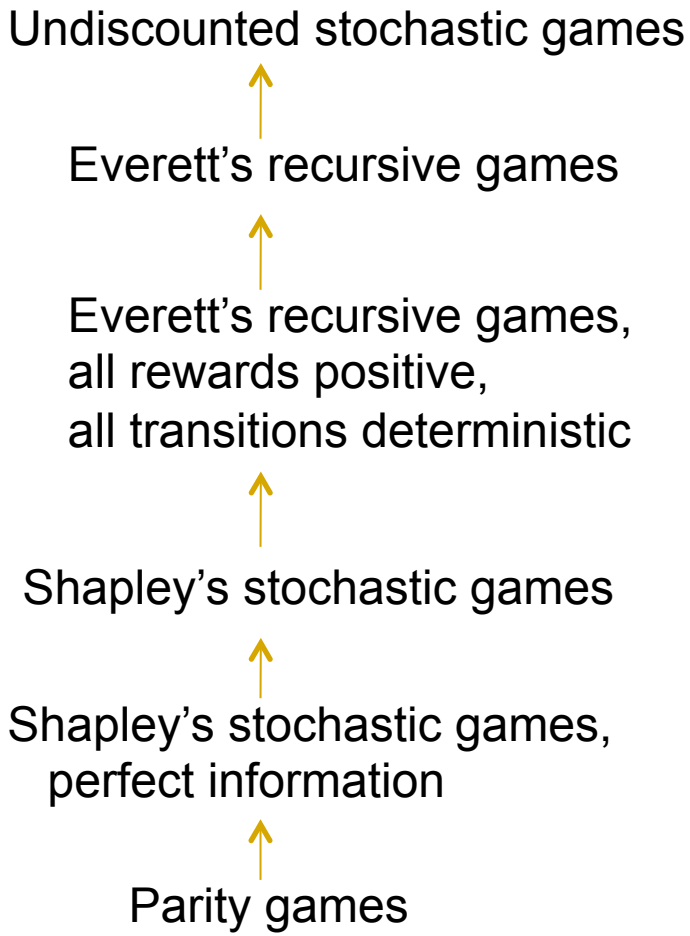
By specifying a starting position we obtain a particular game Γ^k. The term "stochastic game" will refer to the collection $\Gamma = \{\Gamma^k | k = 1, 2, \ldots, N\}$.

The full sets of pure and mixed strategies in these games are rather cumbersome, since they take account of much information that turns out to be irrelevant. However, we shall have to introduce a notation only

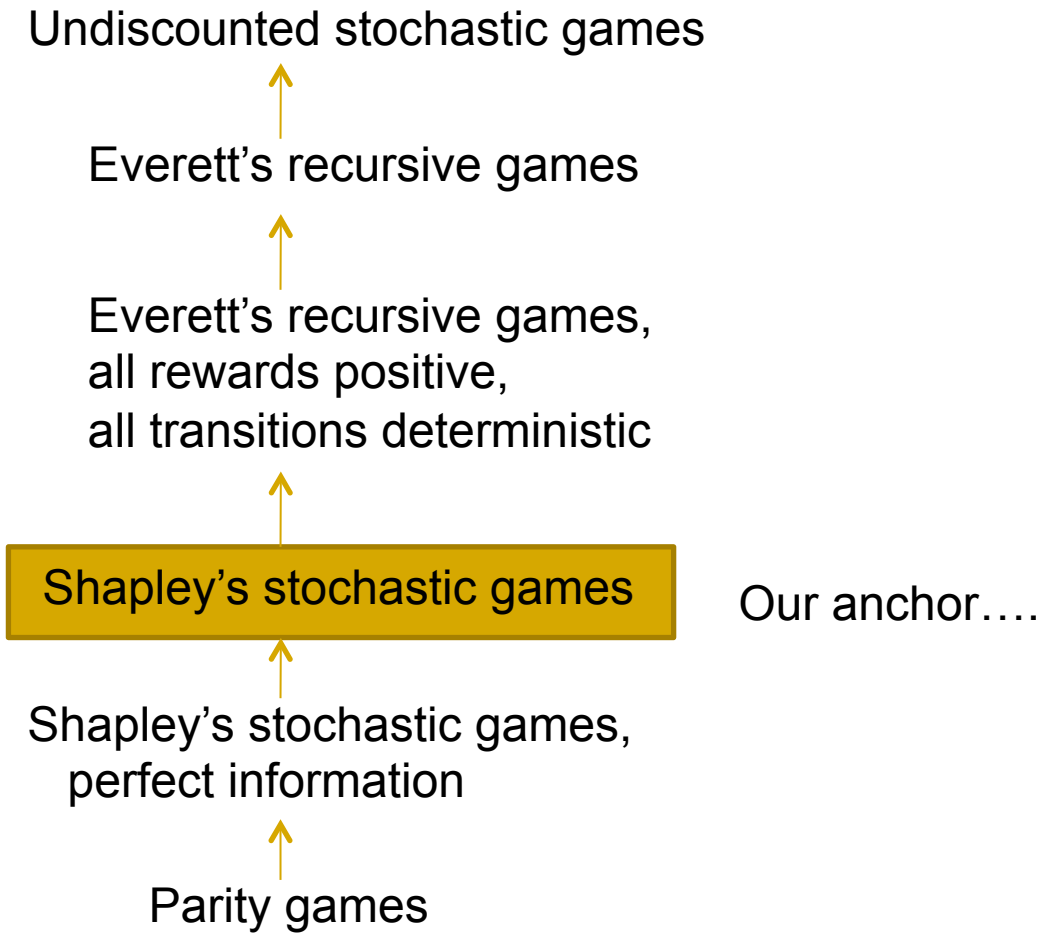
The algorithmic lens: Polynomial time reductions



Finite stochastic games - the algorithmic lens



Finite stochastic games - the algorithmic lens



Our anchor….

Finite stochastic games - the algorithmic lens

Undiscounted stochastic games *The Big match*

↑

Everett's recursive games

↑

Everett's recursive games,
all rewards positive,
all transitions deterministic *Purgatory*

↑

Shapley's stochastic games

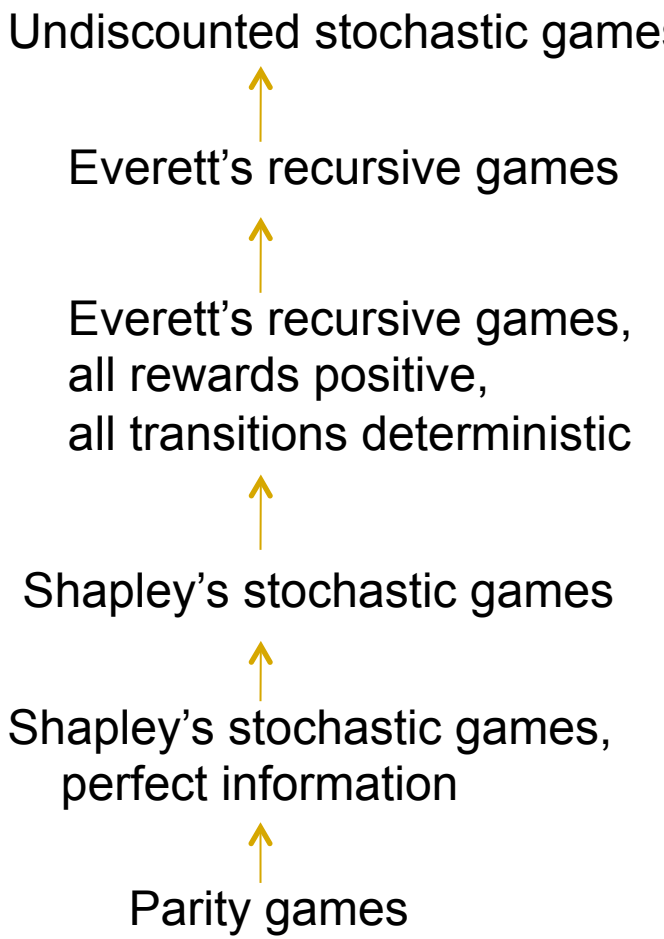
↑

Shapley's stochastic games,
perfect information *Backgammon*

↑

Parity games

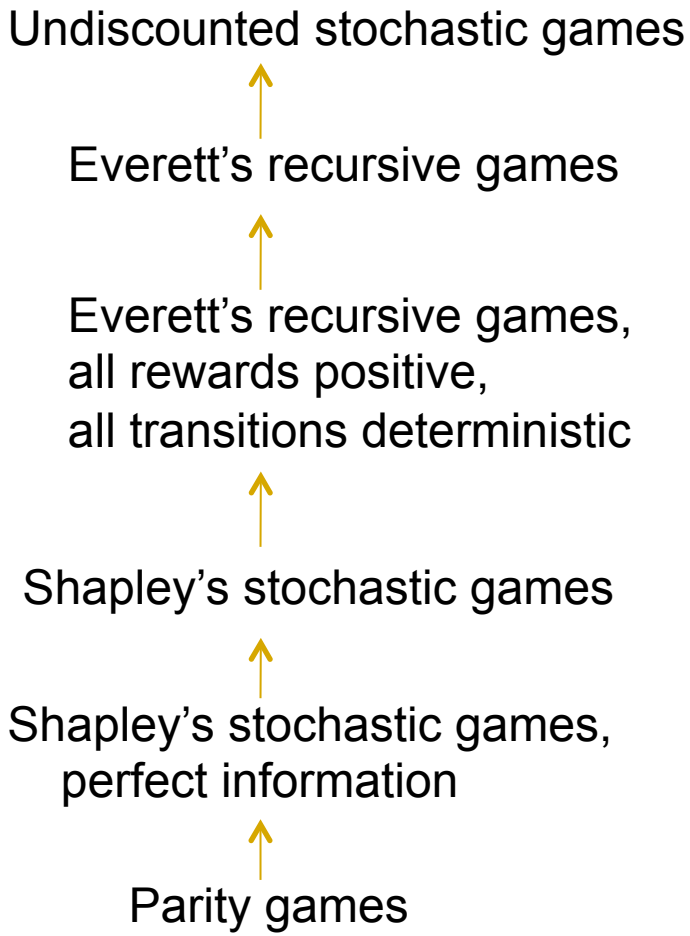
Finite stochastic games - the algorithmic lens



B
↑
A

means
"solving A
polynomial time reduces to
solving B"

Finite stochastic games - the algorithmic lens



B

A

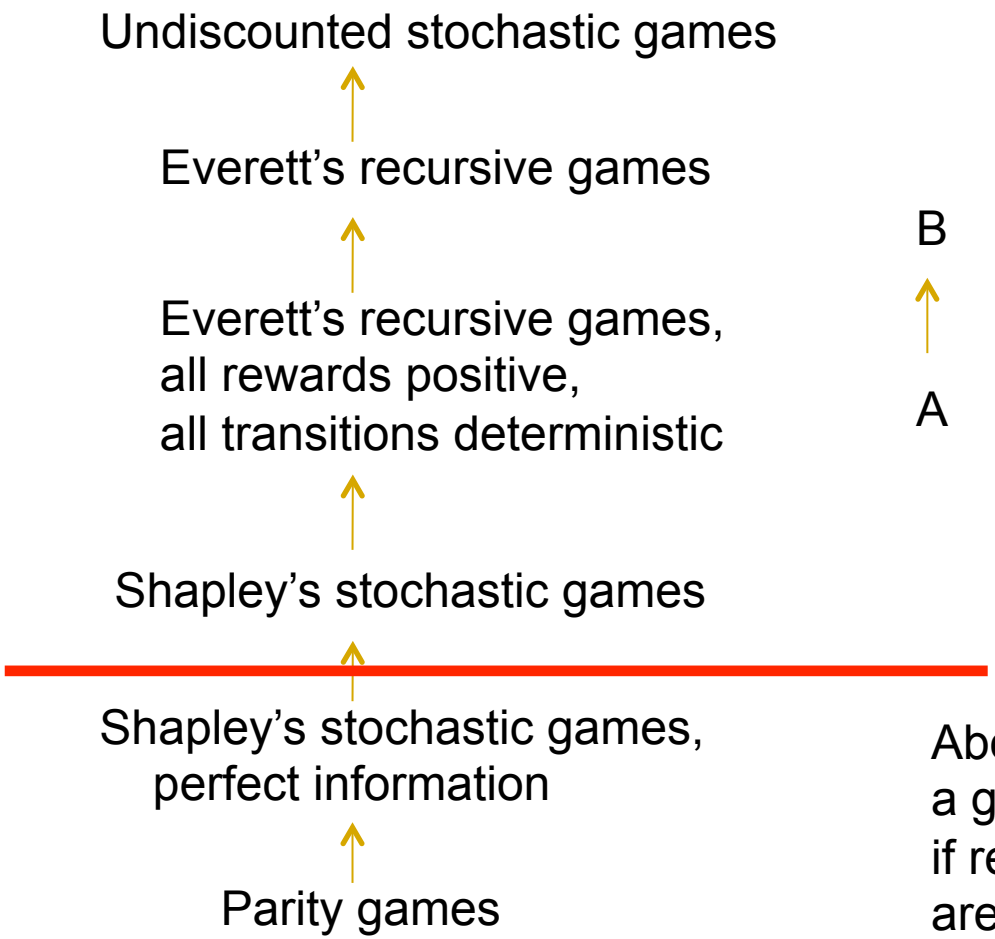
actually means
"comparing the value of instance of A to given rational number polynomial time reduces to comparing the value of instance of B to given rational number"

Why so cumbersome?

Why not just compute values and optimal strategies?

Three reasons….

Finite stochastic games - the algorithmic lens



B

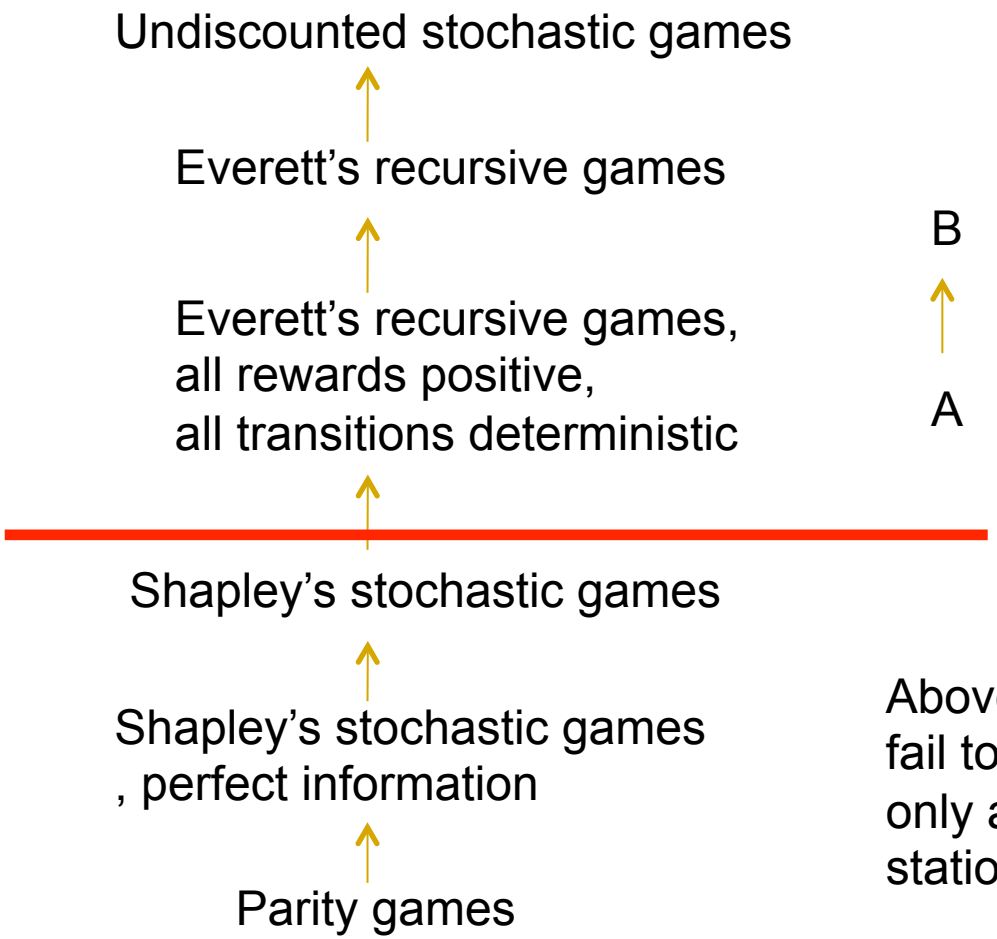
↑

A

actually means
"comparing the value
of instance of A to given
rational number
polynomial time reduces to
comparing the value of
Instance of B to given
rational number"

Above this line, the value of
a game can be irrational, even
if rewards and transition probabilities
are rational.

Finite stochastic games - the algorithmic lens



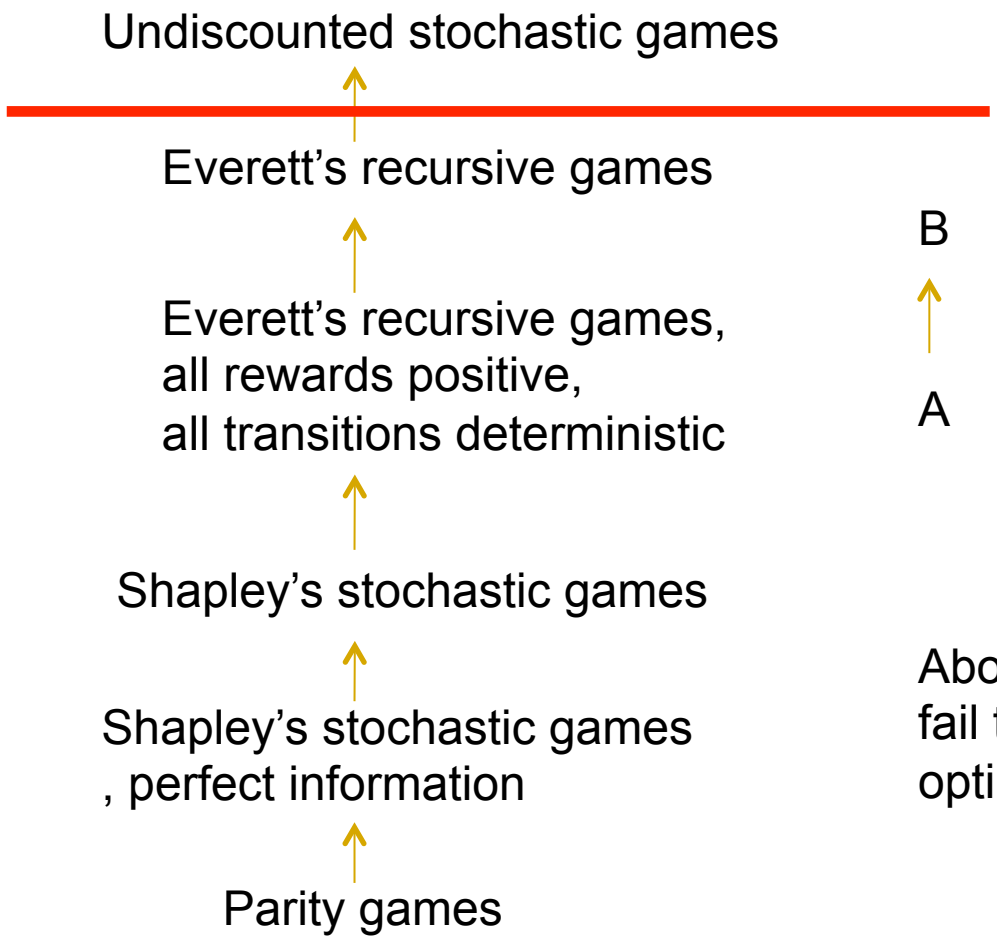
B

A

actually means
"comparing the value
of instance of A to given
rational number
polynomial time reduces to
comparing the value of
Instance of B to given
rational number"

Above this line, games may
fail to have optimal strategies, so
only approximately optimal
stationary strategies can be computed.

Finite stochastic games - the algorithmic lens



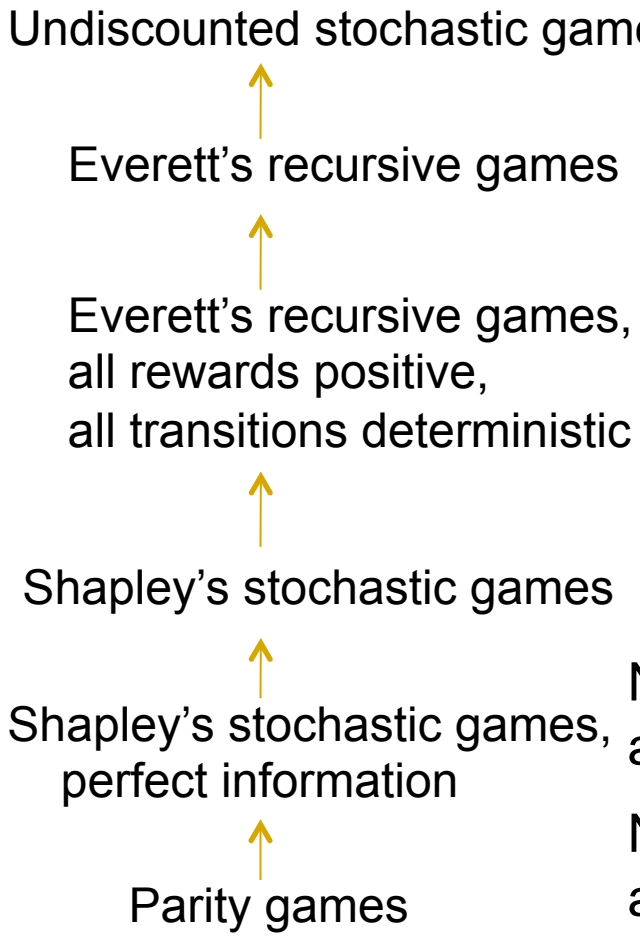
B

A

actually means
"comparing the value of instance of A to given rational number polynomial time reduces to comparing the value of Instance of B to given rational number"

Above this line, games may fail to have even approximately optimal stationary strategies.

Finite stochastic games - the algorithmic lens



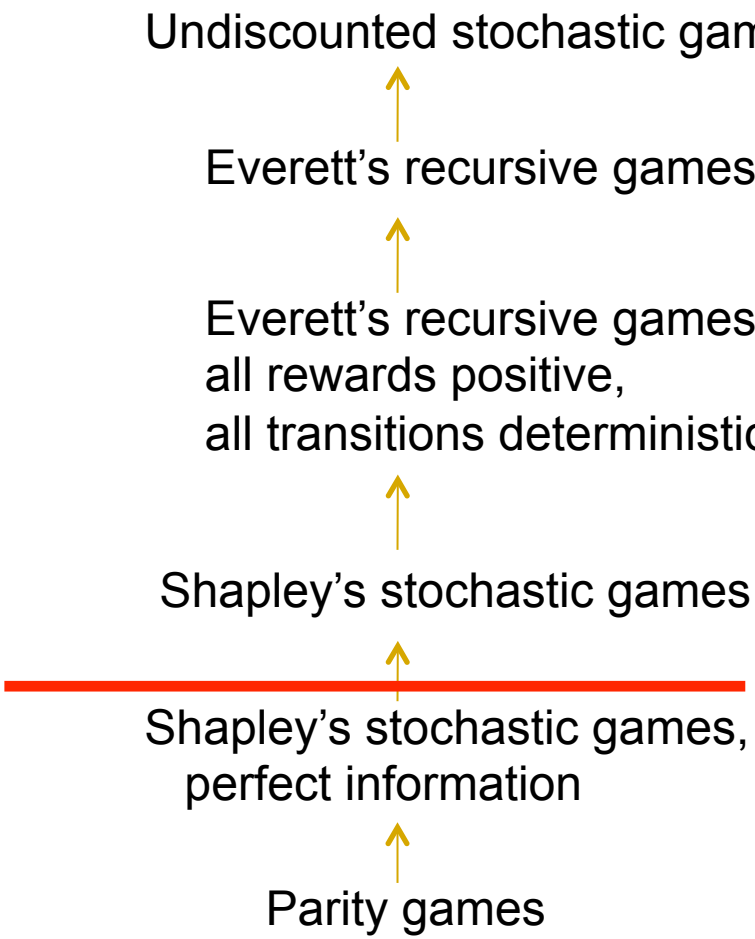
B
↑
A

actually means
"comparing the value
of instance of A to given
rational number
polynomial time reduces to
comparing the value of
Instance of B to given
rational number"

None of these computational tasks are known to be polynomial time solvable.

None of these computational tasks are known to be NP-hard or PPAD-hard.

Finite stochastic games - the algorithmic lens



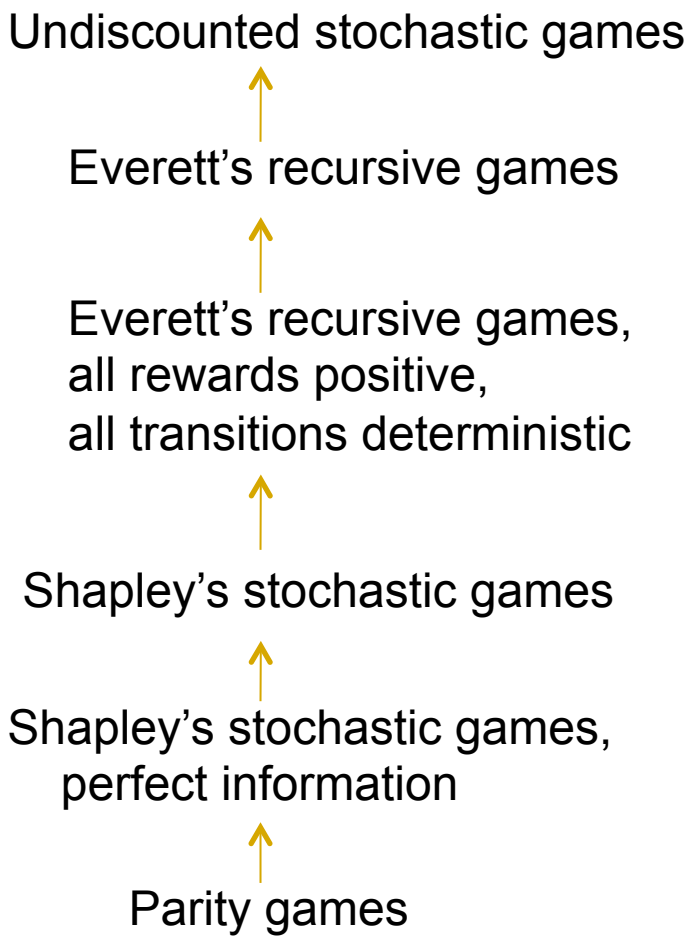
Tasks above the red line are "SQRT-SUM hard".

B ↑ A means "comparing the value of instance of A to given rational number polynomial time reduces to comparing the value of Instance of B to given rational number"

None of these computational tasks are known to be polynomial time solvable.

None of these computational tasks are known to be NP-hard or PPAD-hard.

Finite stochastic games - the algorithmic lens



Also, we don't know how to reverse any of the arrows!

Finite stochastic games - the ~~algorithmic~~ engineering lens

Undiscounted stochastic games

↑

Everett's recursive games

↑

Everett's recursive games,
all rewards positive,
all transitions deterministic

↑

Shapley's stochastic games

↑

Shapley's stochastic games,
perfect information

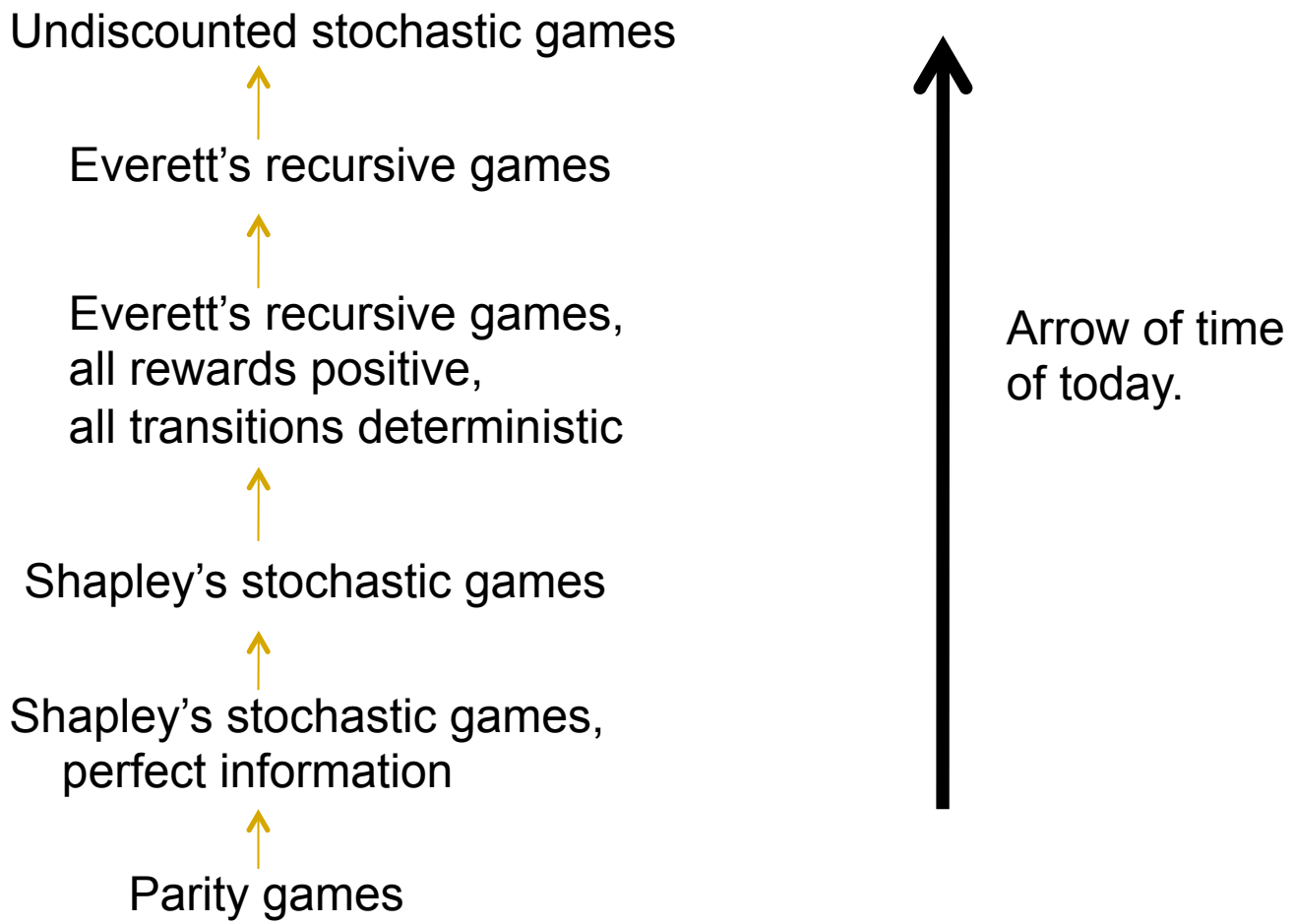
↑

Parity games

Below this line there are algorithms that solve these games efficiently except on ***extremely*** carefully constructed instances

Even for the microwave oven, the best such algorithms came from above
(were devised first for Shapley's model)

Finite stochastic games - the algorithmic lens



Computer aided design and analysis of embedded systems

- Embedded System =
  - Hardware/software combination =
  - Thing with microchips and bottons

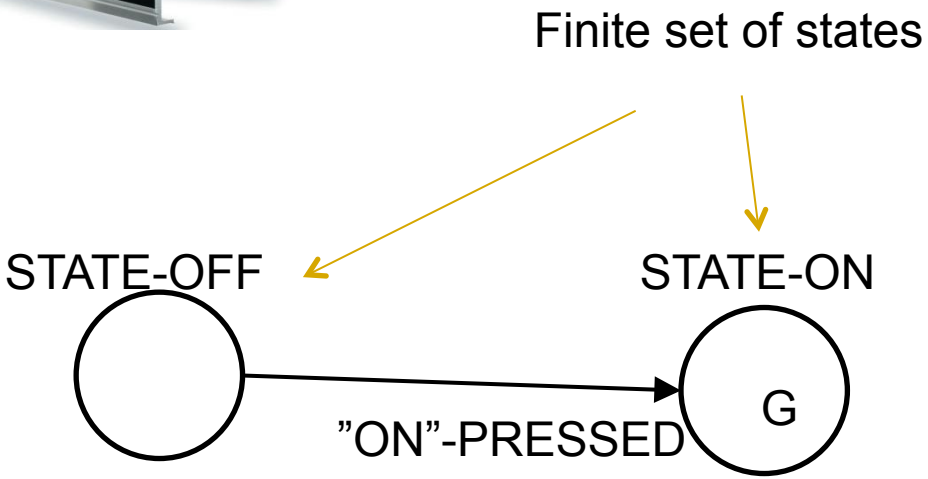
- Methodology: Design formal models of these before production. Analyse the models for desired properties.

Why?

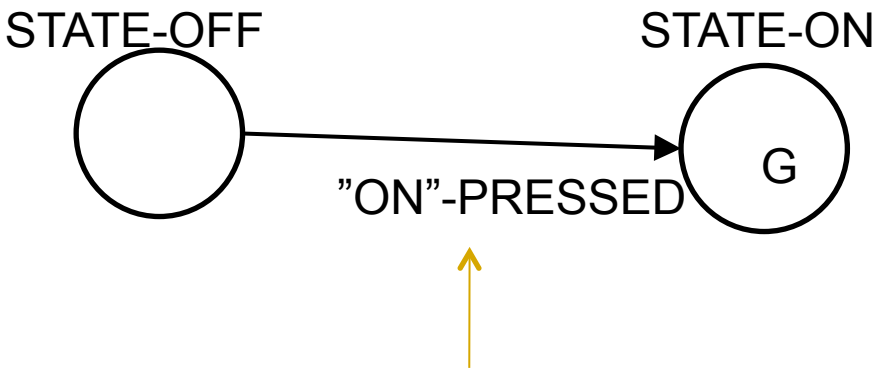
European Space Agency's Ariane 5 rocket exploding due to software error.

Standard Model: Transition system

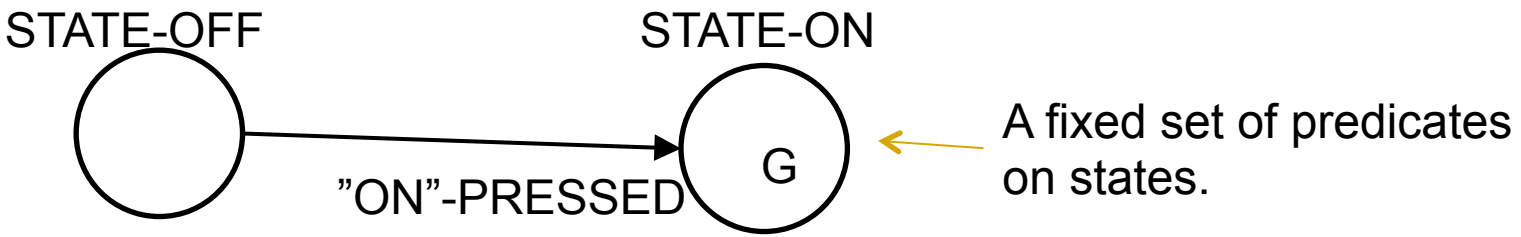
Standard Model: Transition system



Standard Model: Transition system

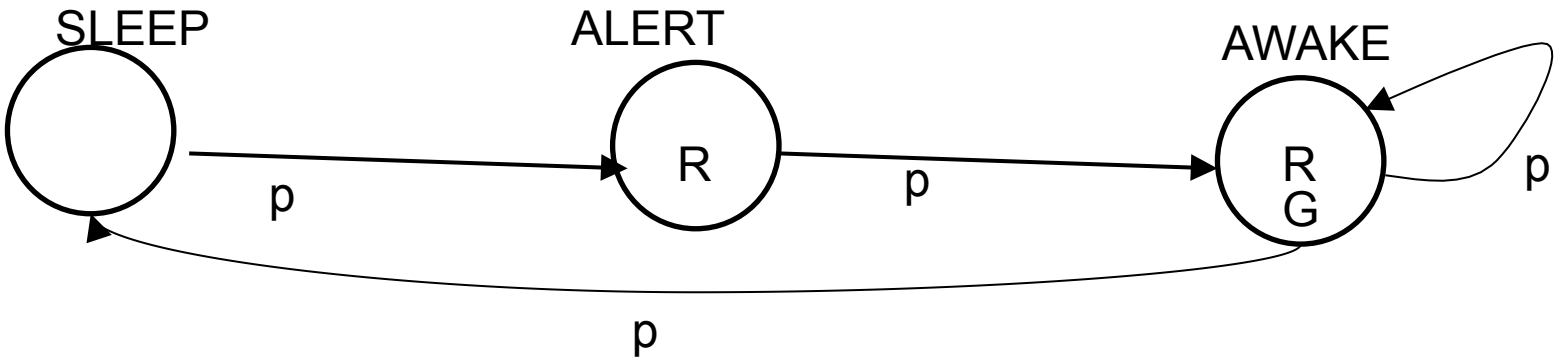


Standard Model: Transition system

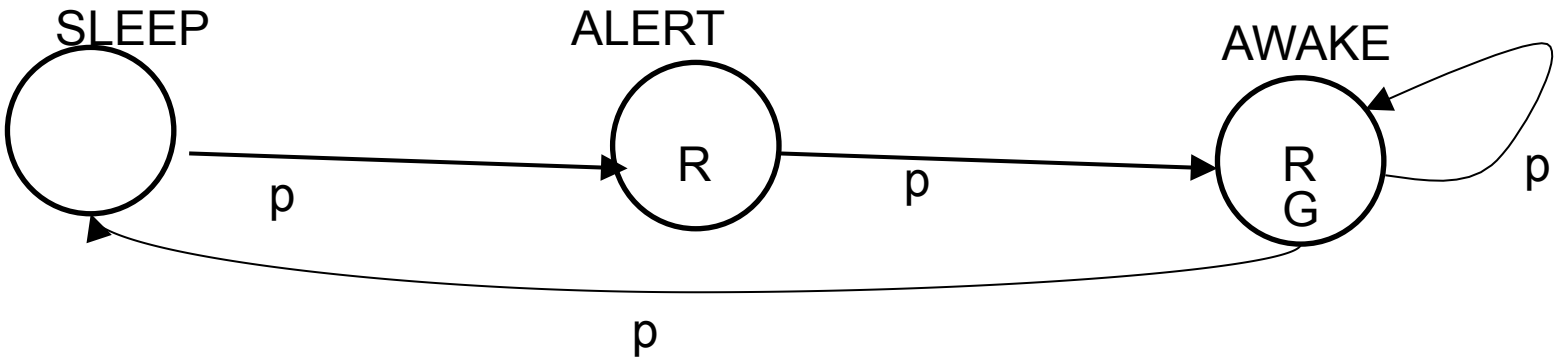


The μ-calculus

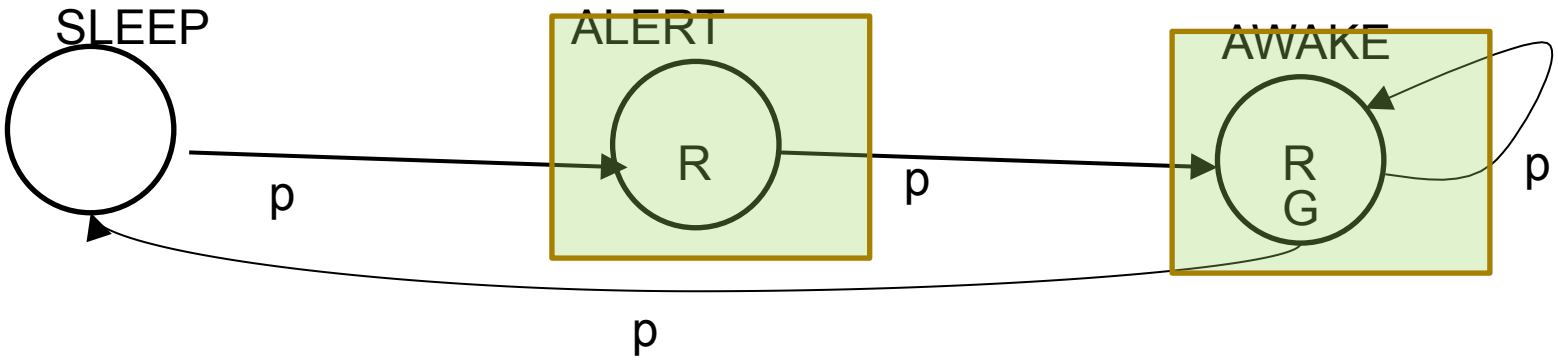
- A modal logic for expressing properties of transition systems.
  - *The green light is on.*
  - *If the user presses ”On”, the green light turns on.*
  - *If the user keeps pressing ”On” over and over, the green light will eventually turn on.*
  - *The Ariane rocket could explode….*
- Given a mu-calculus formula and a transition system, the formula may be true in some states and not in others.
  - This is what makes the logic ”modal”.


SLEEP
ALERT
AWAKE
R
R
G
p
p
p
p

Mu-calculus formulae



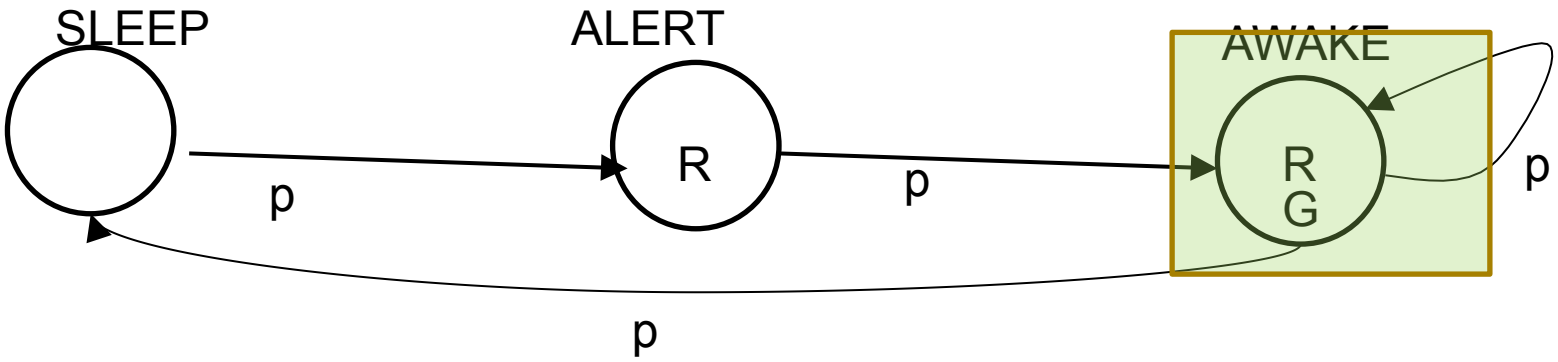
Mu-calculus formulae



Atomic formulae

$$R$$

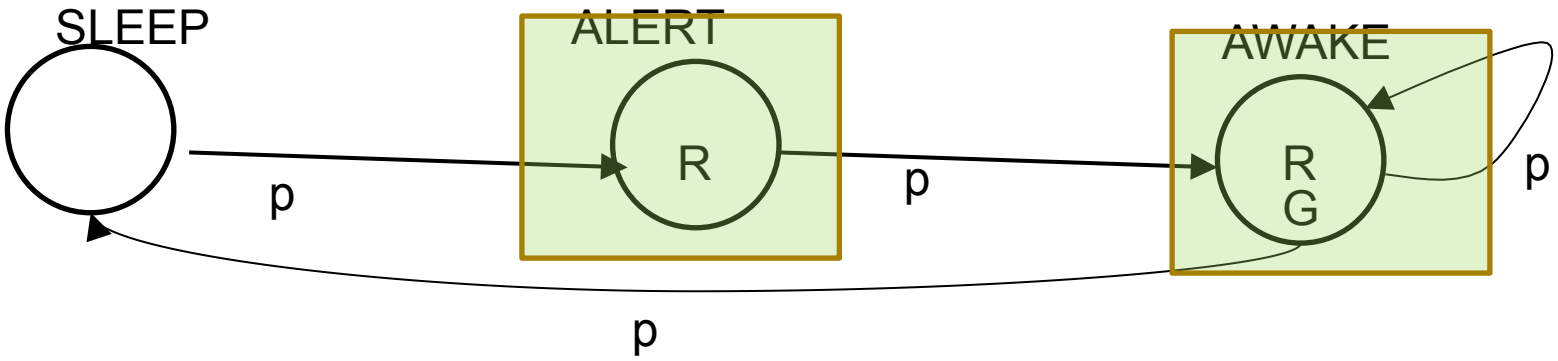
Mu-calculus formulae



Conjunctions and disjunctions

$$R \wedge G$$

Mu-calculus formulae

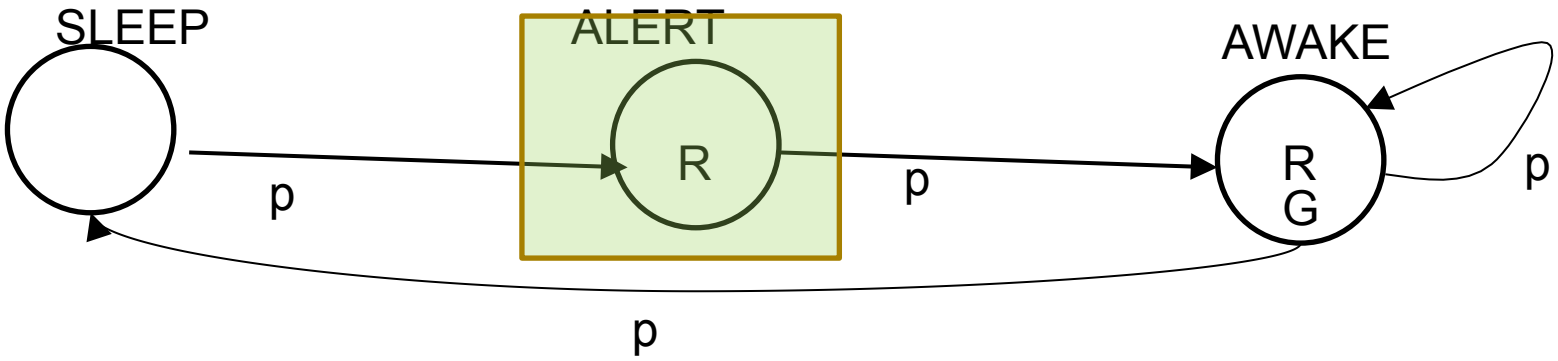


Modalities

$$\langle p \rangle G$$

”p kan happen in a way so that G becomes true”

Mu-calculus formulae

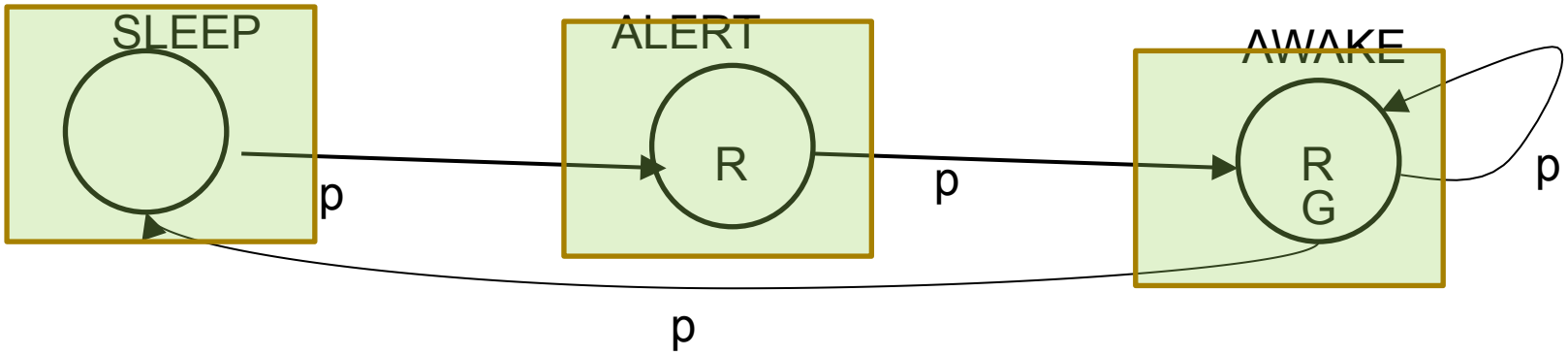


Modalities

$$[p]G$$

”No matter how p happens
G becomes true”

Mu-calculus formulae



”If the user keeps on pressing the button, it is not impossible that the green light will *eventually* turn on”

$$G \vee \langle p \rangle G \vee \langle p \rangle \langle p \rangle G \vee \ldots$$

$$\mu X.G \vee \langle p \rangle X$$

μ-formulae

$$G \vee [p]G \vee [p][p]G \vee \ldots$$

$$\mu X.G \vee [p]X$$

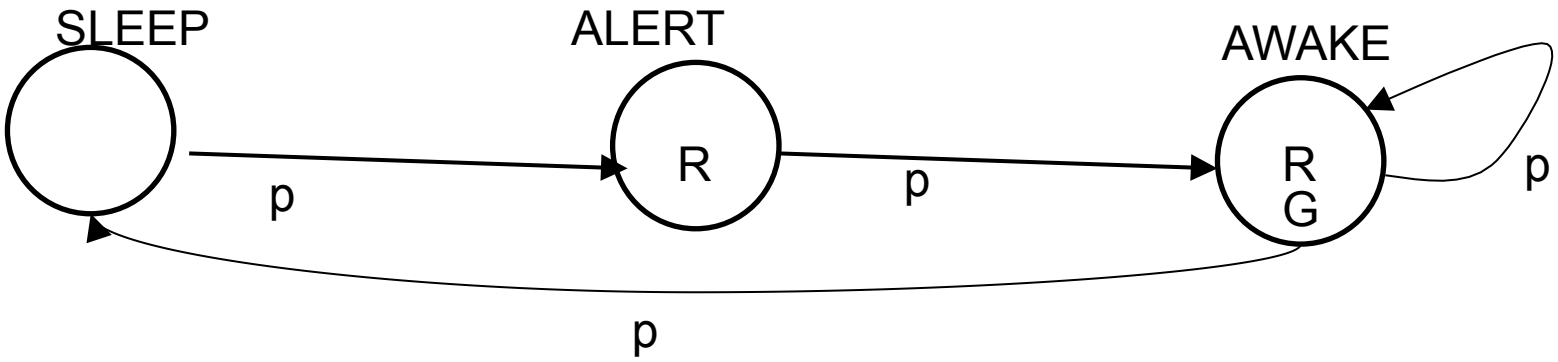
- Blurring the destinction between predicates and sets of states, the set of states that makes the formula true is by definition the ***least fixed point*** of

$$X \rightarrow G \vee [p]X$$

μ-formulae vs. ν-formulae

- μ-formulae are ***least*** fixed points.
  - Interesting μ-formulae are typically ”infinite ORs” expressing ”Something eventually has to happen”
- ν-formulae are ***greatest*** fixed points.
  - Interesting ν-formulae are typically ”infinite ANDs” expressing ”Something is never allowed to happen”.

Mu-calculus formulae



”The green light never goes off

$$G \wedge [p]G \wedge [p][p]G \wedge \ldots$$

$$\mu X.G \vee [p]X$$

Nested fixed points!

- The light is only green finitely many times in any infinite run of the system.

$$\mu Y.\nu Z.(G \wedge [p]Y) \vee (\neg G \wedge [p]Z)$$

- ”a” is treated *fairly*: It is not *enabled* infinitely often but only *invoked* finitely often.

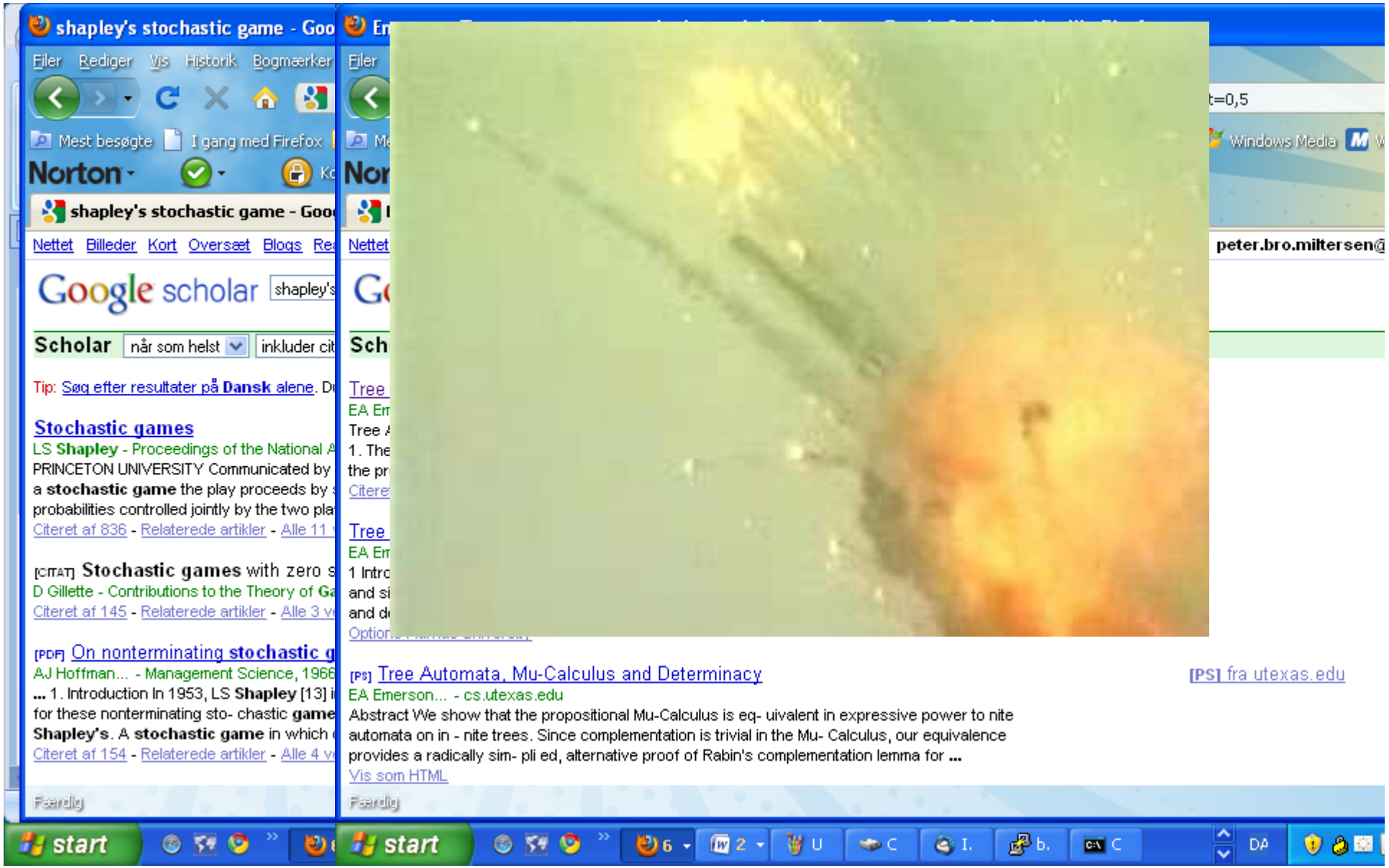
$$\nu X.\mu Y.\nu Z.[a]X \wedge (\langle a\rangle \mathrm{tt} \Rightarrow [-a]Y) \wedge [-a]Z.$$

Computational problem

- MODEL-CHECKING-MU-CALCULUS
  - Input: A transition system and a mu-calculus formula
  - Ouput: For which states are the formula true?

- How important is this problem?

A bibliographical experiment



Moving to game theory….

- MODEL-CHECKING-MU-CALCULUS
  - Input: A transition system and a mu-calculus formula
  - Ouput: For which states are the formula true?

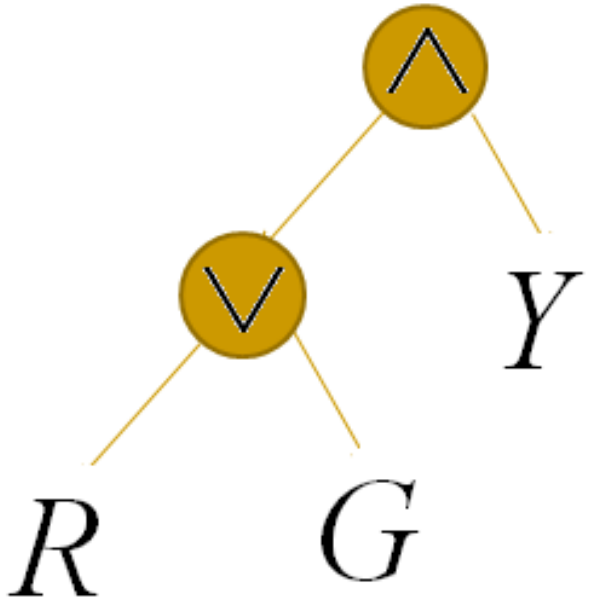
- This problem can be captured by two-player zero-sum game.

Intuition

- ***Propositional*** logic (Atomic formulas, conjunctions and disjunctions) can be captured by a game.

Capturing propositional logic by a game

$$(R \wedge G) \vee Y$$



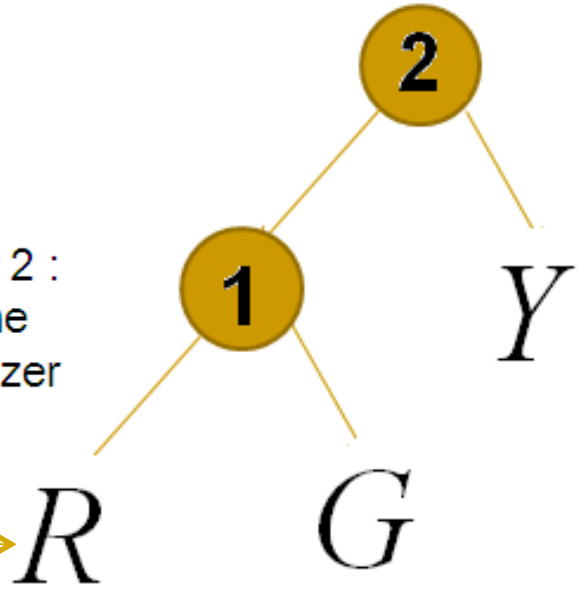
Capturing propositional logic by a game

$$(R \wedge G) \vee Y$$

Perfect information extensive form game!

Player 1: Max, the Maximizer

Player 2 : Min, the Minimizer



Min pays Max a dollar if R true
Max pays Min a dollar if R is false

Intuition

- ***Propositional*** logic (Atomic formulas, conjunctions and disjunctions) can be captured by a game.

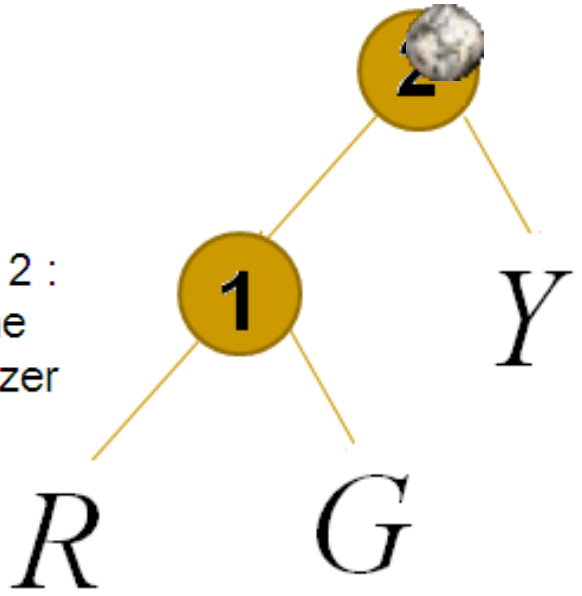
- The mu-calculus can be captured by an extension of exactly the same game!

Capturing propositional logic by a game

$$(R \wedge G) \vee Y$$

Player 1:
Max, the
Maximizer

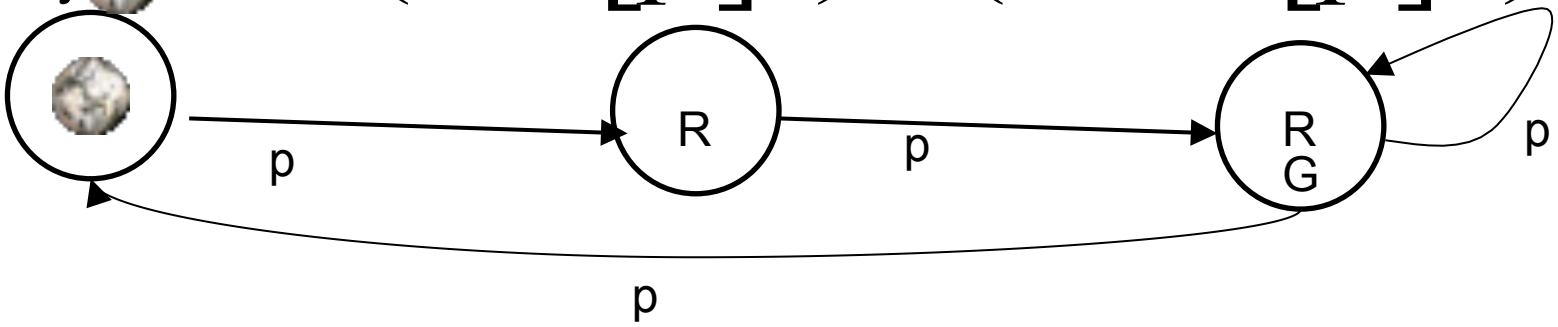
Player 2 :
Min, the
Minimizer



The game capturing model checking the mu-calculus

- State space of the game =
  - States of the transition system x Subformula of the formula
- Graphically, *two* pebbles
  - one pebble on the system
  - one in the formula.

$$\mu Y.\nu Z.(G \wedge [p]Y) \vee (\neg G \wedge [p]Z)$$



- When formula-pebble is on a greek letter it is moved inwards.
- When formula-pebble is on [a]F, it is moved to F and Min moves system-pebble along a,
- When formula-peblle is on <a>F, it is moved to F and Max moves system-pebble along a.
- If play is infinite, Max wins *if the outermost greek letter visited infinitely often is v.*
- *Max has a winning strategy if and only if the formula is true for starting state of system-pebble.*

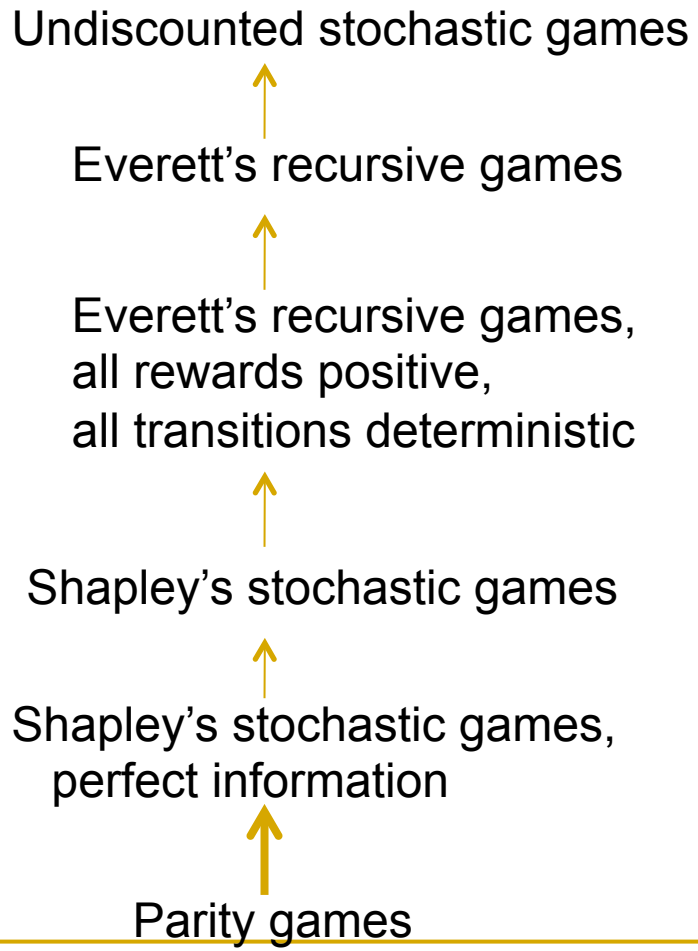
Parity games (the cleaned up model)

- Finite state space 1,2,…,N, start state is 1.
- Game is presented by directed graph on state space. Arcs are also called actions.
- Each state belongs to either Player 1 or Player 2. That player determines which action to take.
- Actions have integer ***priorities***.
- Play continues forever.
- If the largest priority seen infinitely often is odd, Player 1 wins (payoff 1), otherwise Player 2 wins (payoff -1).

Facts about parity games (Emerson and Jutla 1991)

- Model checking the mu-calculus is **polynomial time equivalent** to computing the value of a given parity game.
- Any parity game has a value.
- The value is either -1 or 1.
- The players can guarantee the value with ***positional*** strategies = deterministic stationary strategies.
- Last three facts can also been obtained as corollaries of Martin's theorem on Borel determinacy (1975).

Finite stochastic games - the algorithmic lens



B
↑
A

means
"solving A
polynomial time reduces to
solving B"

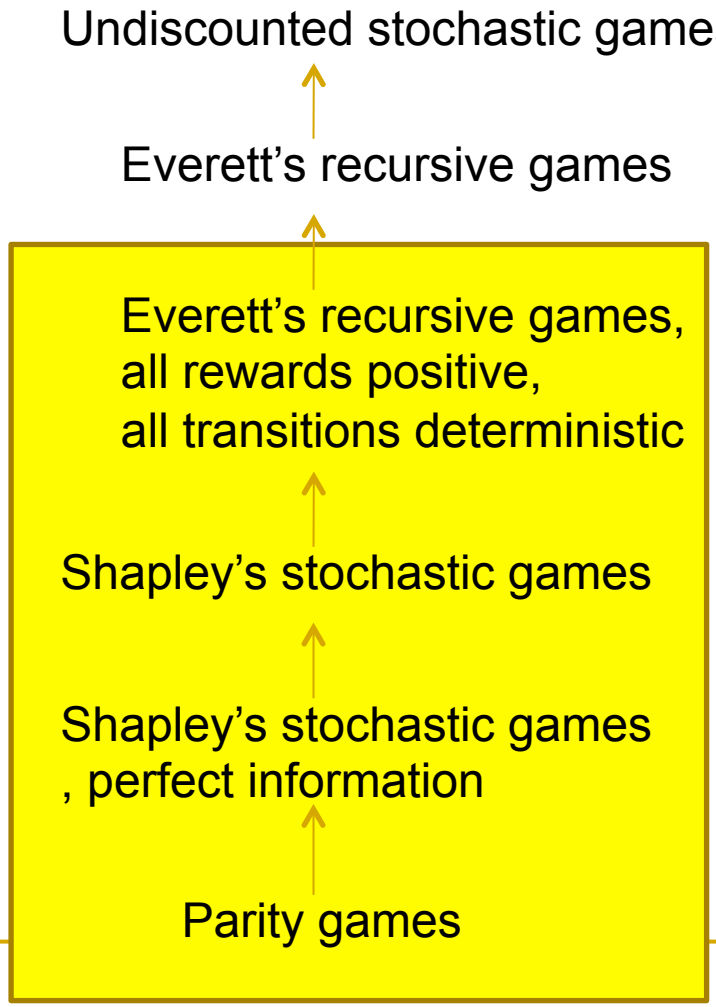
Solving Shapley's stochastic games

- No polynomial time algorithm known.
- The engineer does not mind because of

  Howard's algorithm!

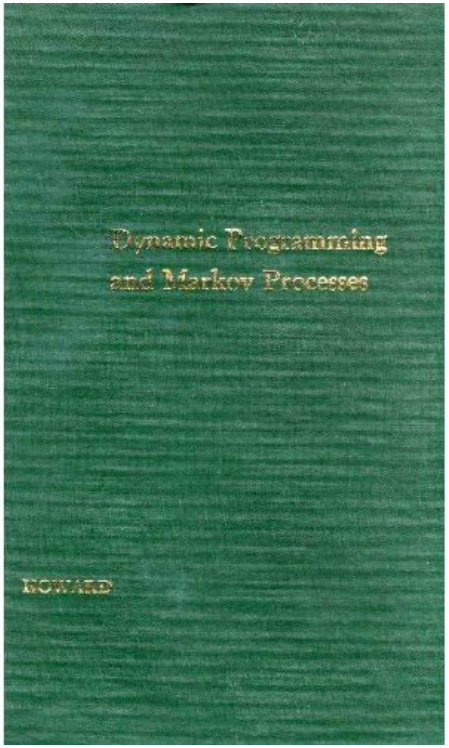
  - a.k.a. ***policy improvement, policy iteration, strategy improvement, stategy iteration.***
  - Howard is to Shapley's stochastic games what the simplex algorithm is to linear programming, "polynomial time in practice"
  - Preferred algorithm for solving parity games is adaptation of Howard due to Jurdzinski and Voge (2000).
  - ***Conjectured until 2009 that to be polynomial time for the perfect information case.*** Oliver Friedmann in 2009 found examples showing otherwise.
  - The examples were obtained by looking at the parity game case!

Howard – A panacea for stochastic game woes!



While Howard only described his algorithm for Markov Decision Processes (1-player stochastic games) it applies to entire yellow area.

Howard's algorithm (1960)



Recent results on the time complexity of Howard's algorithm

- O. Friedmann, 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.

Howard (1960)

- Description of the algorithm for the case of 1-player stochastic games = Markov Decision Process (MDP).

- We will describe it more generally.

Howard's algorithm for Shapley's stochastic games.

- x := some stationary strategy for Player 1.
- Repeat forever
  - Let y be a best reply to x by Player 2.
  - Let w be the vector of expected payoffs when Player 1 plays x and Player 2 plays y.
  - For each position k, do
    - Let x at k be an optimal strategy to $A^k(w)$

Correctness

Howard's algorithm

- *The* method of choice in practice for solving MDPs and perfect information stochastic games.
- For those, the strategies computed will ***stabilize*** and the algorithm runs in finite time.
- Complexity analysis?

Zeitschrift für Operations Research, Volume 29, page 315 – 316

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" $1, \ldots N$, 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

$$N$$

OR Spektrum (1986) 8:37–40

OR Spektrum

© Springer-Verlag 1986

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 Holzbaur, 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 strategy that occurs after another O(log n) iterations.
- First derived for the MDP case using complementary slackness of the LP formulation.
- Then generalized to the 2-player case.

How to find the action?

Primal:

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

Dual:

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

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 Holzbaur, 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.

What if the discount factor is not fixed?

- Friedman, 2009: $\Omega(2^n)$ iterations for TBSGs.
- Fearnley, 2010: $\Omega(2^n)$ 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 programing! (upcoming)**

Random Facet Algorithm

- Randomized variant of Howard's algorithm, defined for the perfect information case.

- Best known worst case (expected) complexity for solving these games: Exponential in $N^{1/2}$, when Player 1 has a binary choice in each position.

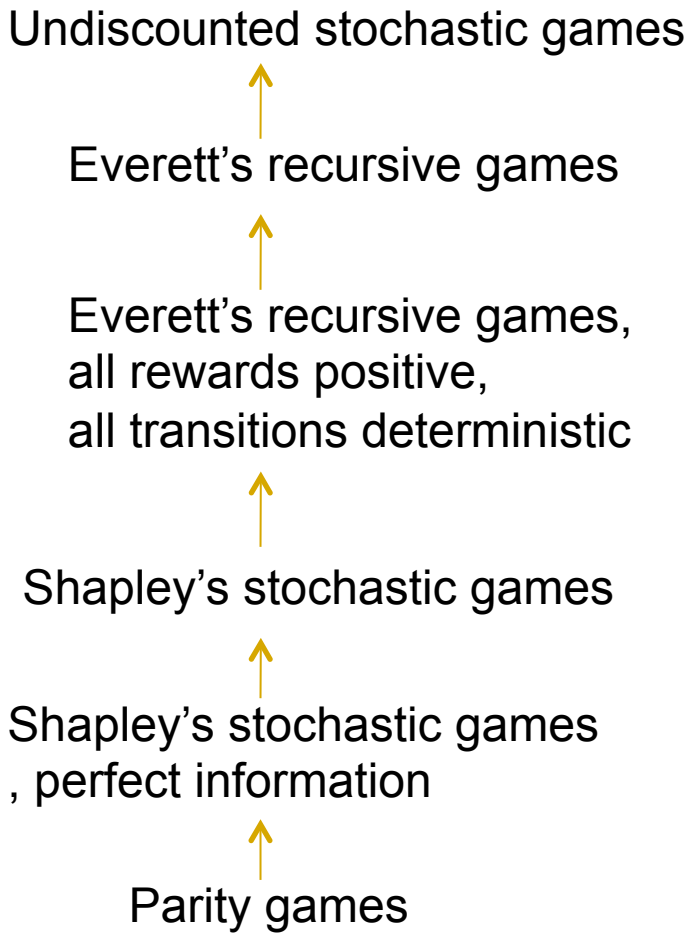
RandomFacet Algorithm (Ludwig 1995)

- RandomFacet (G, π):
  - If $m_k < 2$ for all positions, return π.
  - Pick at random a position k so that $m_k = 2$
  - G' := G with choice of π at k frozen
  - π' := RandomFacet(G', π)
  - If π' is optimal in G return π'
  - Switch choice of π' at k
  - Return(RandomFacet(G, π')

Howard's algorithm for Shapley's games (not perfect information)

- Exponential time is needed in worst case before non-trivial approximation is achieved, even for games with only one non-absorbing position.

Finite stochastic games - the algorithmic lens



a.k.a. Concurrent Reachability Games

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: $\quad y^t :=$ an optimal *best reply* to x^t;
5: $\quad$ **for** $i \in \{0, 1, 2, \ldots, N, N+1\}$ **do**
6: $\quad\quad v_i^t := \mu_i(x^t, y^t)$
7: $\quad$ **end for**
8: $\quad t := t + 1$
9: $\quad$ **for** $i \in \{1, 2, \ldots, N\}$ **do**
10: $\quad\quad$ **if** $\mathrm{val}(A_i(v^{t-1})) > v_i^{t-1}$ **then**
11: $\quad\quad\quad x_i^t := \mathrm{maximin}(A_i(v^{t-1}))$
12: $\quad\quad$ **else**
13: $\quad\quad\quad x_i^t := x_i^{t-1}$
14: $\quad\quad$ **end if**
15: $\quad$ **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?

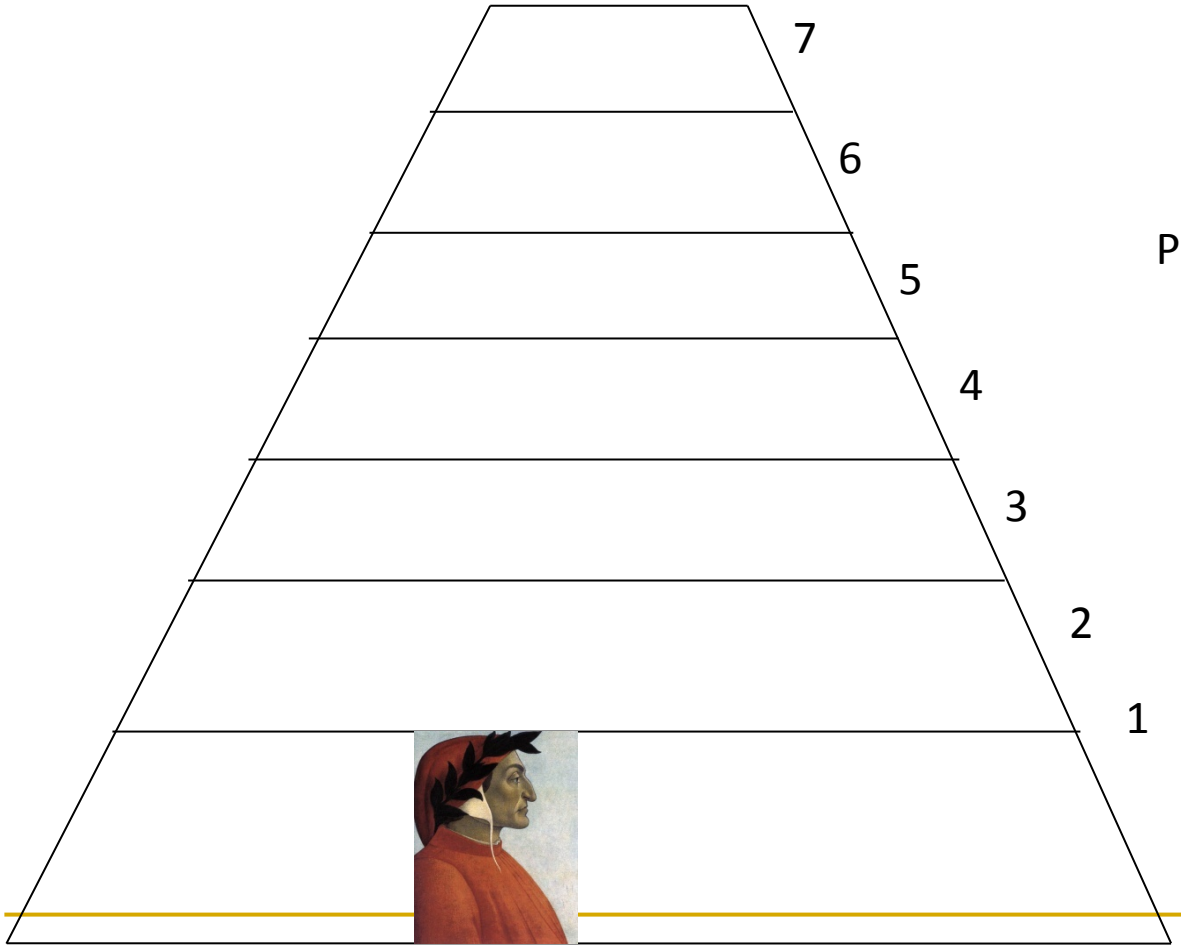
Hansen, Ibsen-Jensen, M., 2011

- 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:
  - $(1/\varepsilon)^{m^{N/4}}$ iterations are (sometimes) necessary to get ε-approximation of value.
  - $(1/\varepsilon)^{2^{31\ m\ N}}$ iterations are always sufficient.

Dante in Purgatory

(Hansen, Koucky, Miltersen, LICS' 09)

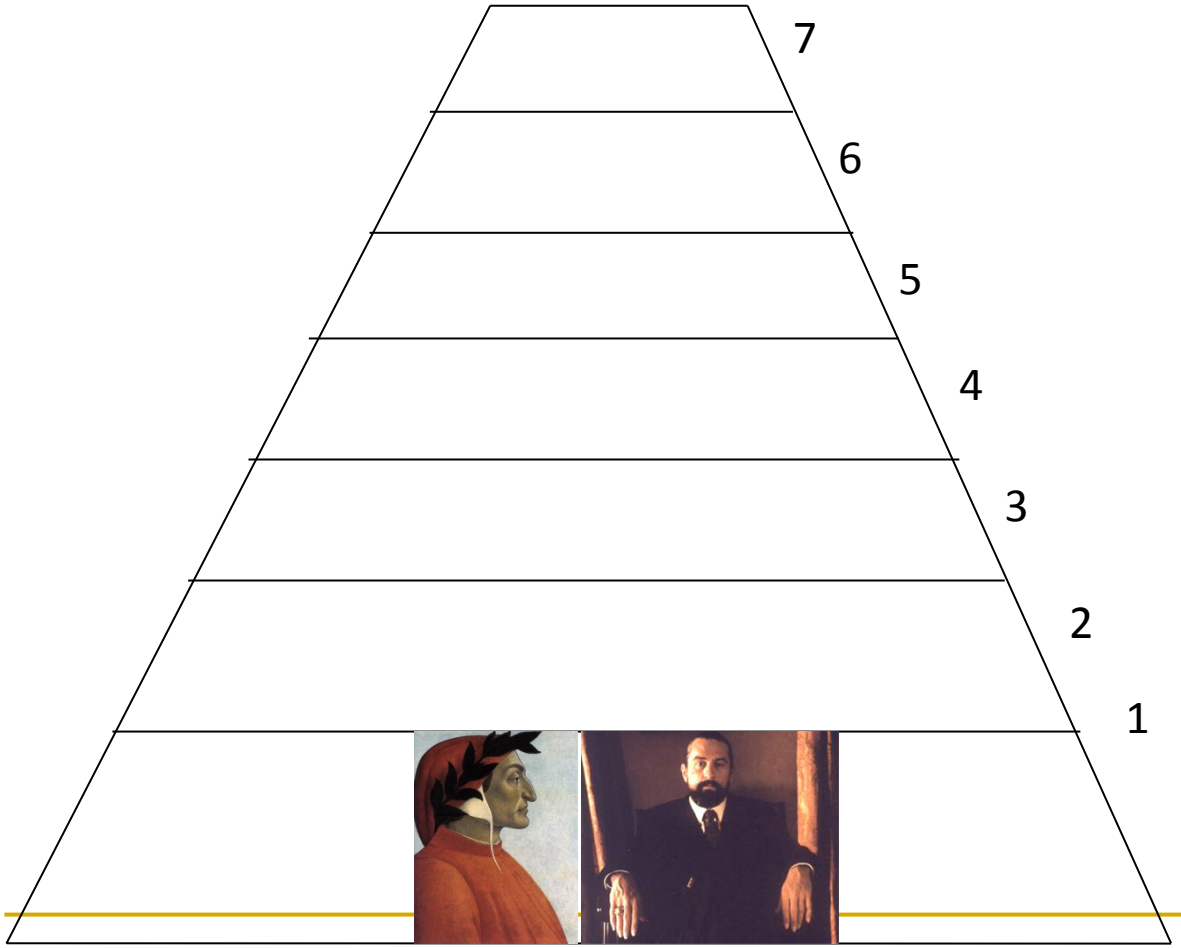


Purgatory has 7 terraces.

Dante enters Purgatory at terrace 1.

Dante in Purgatory

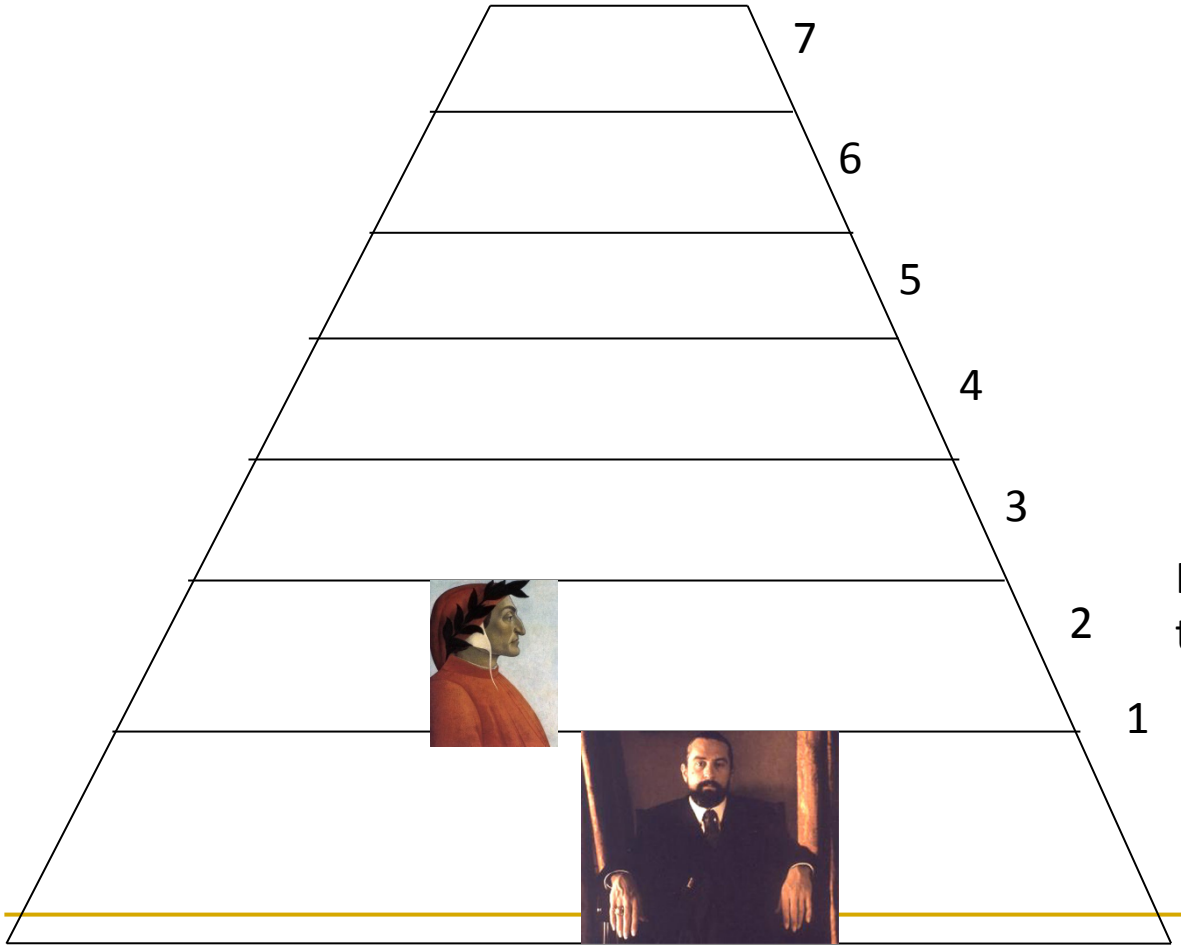
(Hansen, Koucky, Miltersen, LICS' 09)



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

Dante in Purgatory

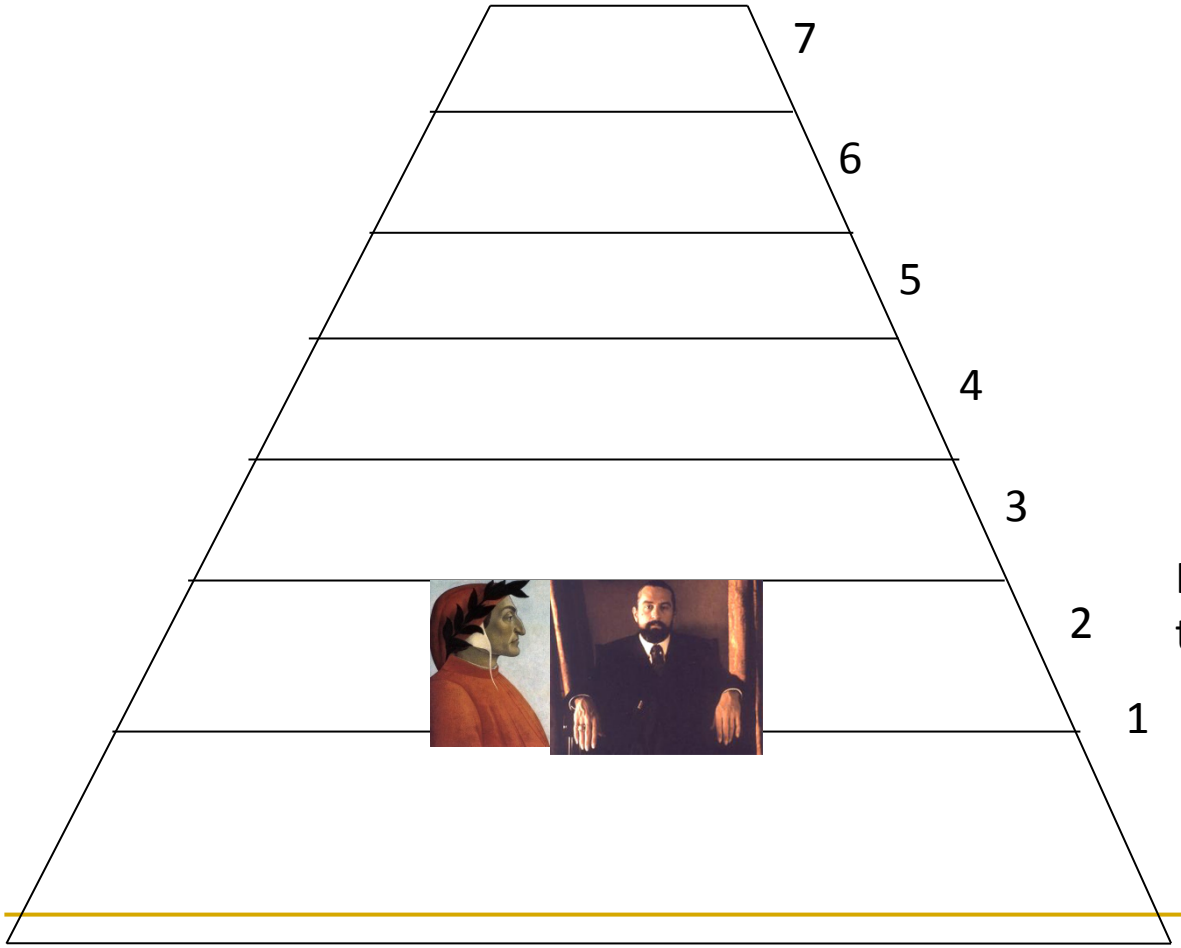
(Hansen, Koucky, Miltersen, LICS' 09)



If Dante wins, he proceeds to the next terrace

Dante in Purgatory

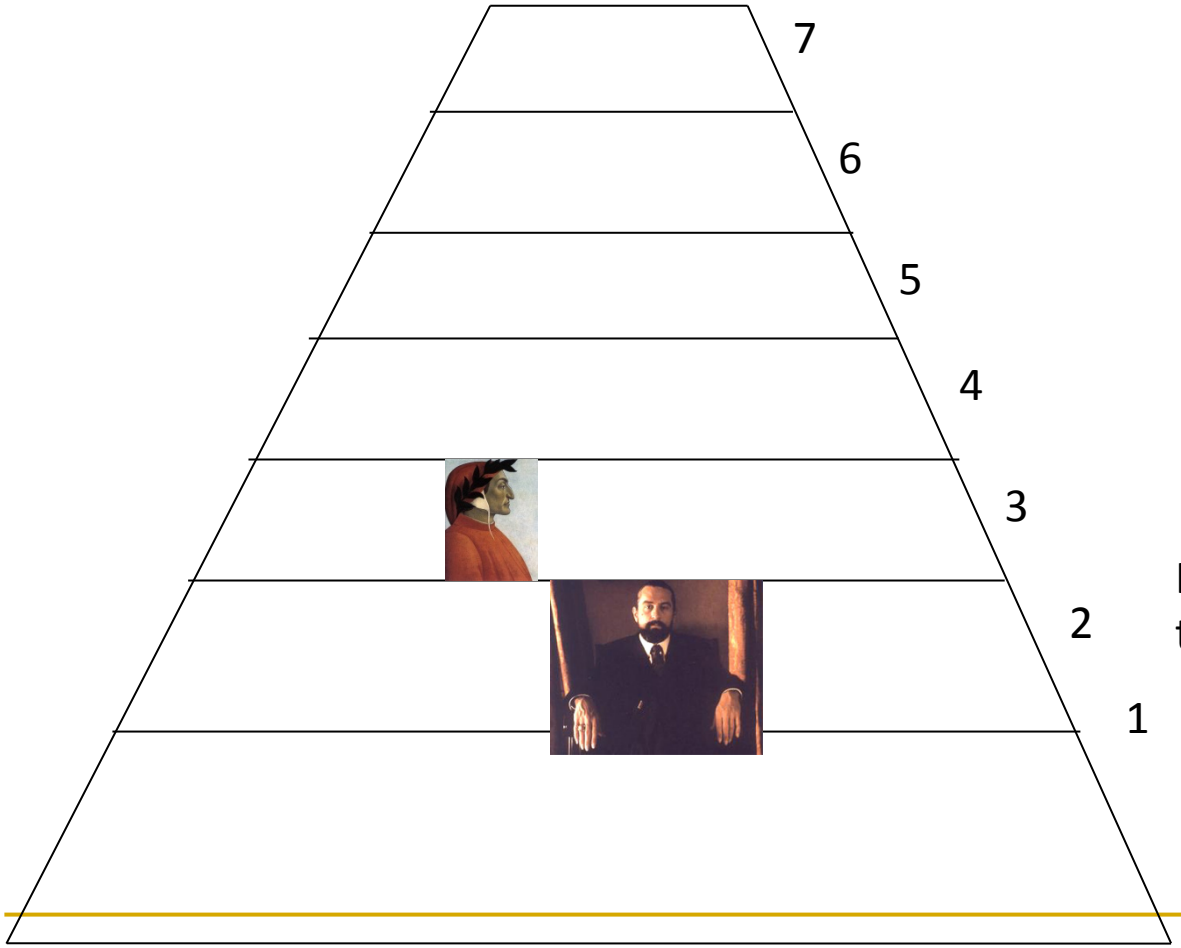
(Hansen, Koucky, Miltersen, LICS' 09)



If Dante wins, he proceeds to the next terrace

Dante in Purgatory

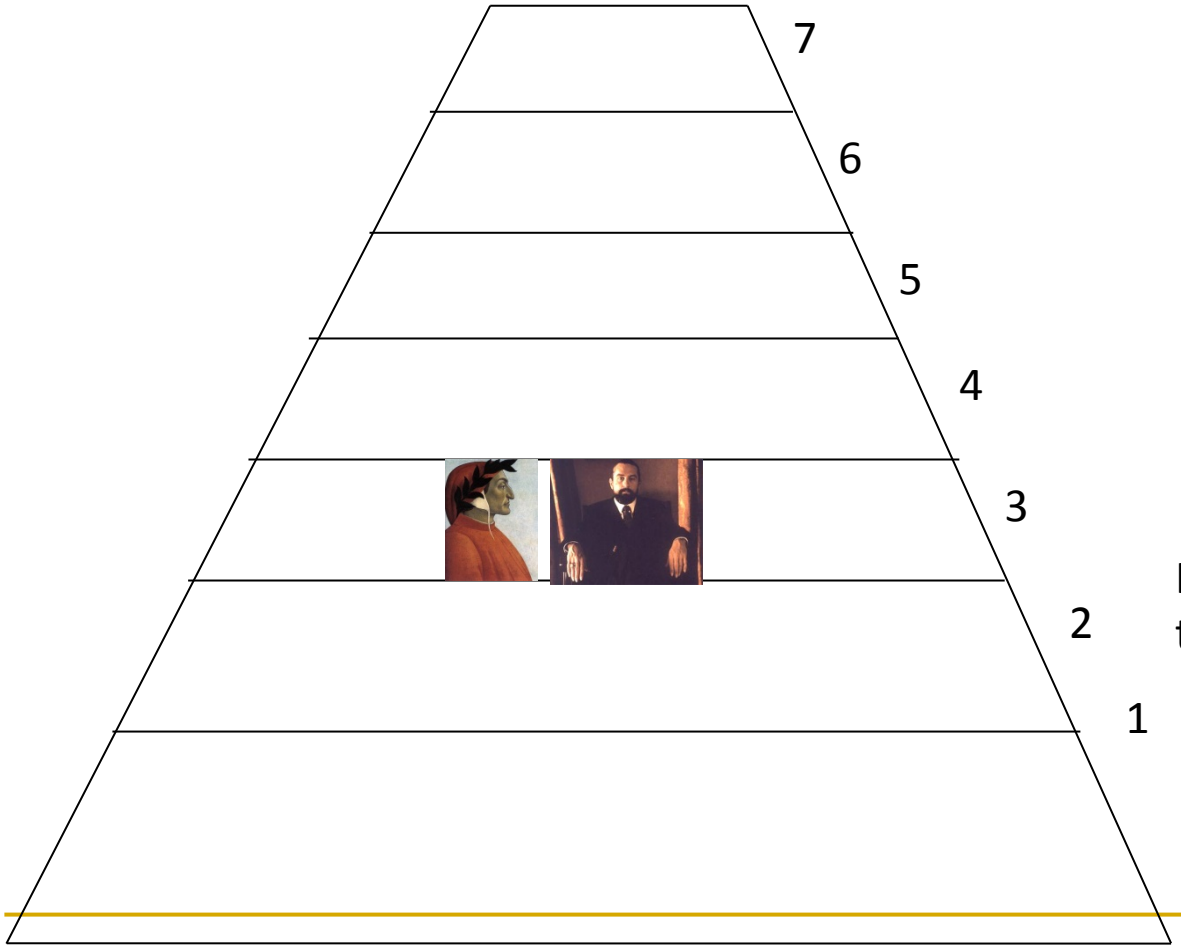
(Hansen, Koucky, Miltersen, LICS' 09)



If Dante wins, he proceeds to the next terrace

Dante in Purgatory

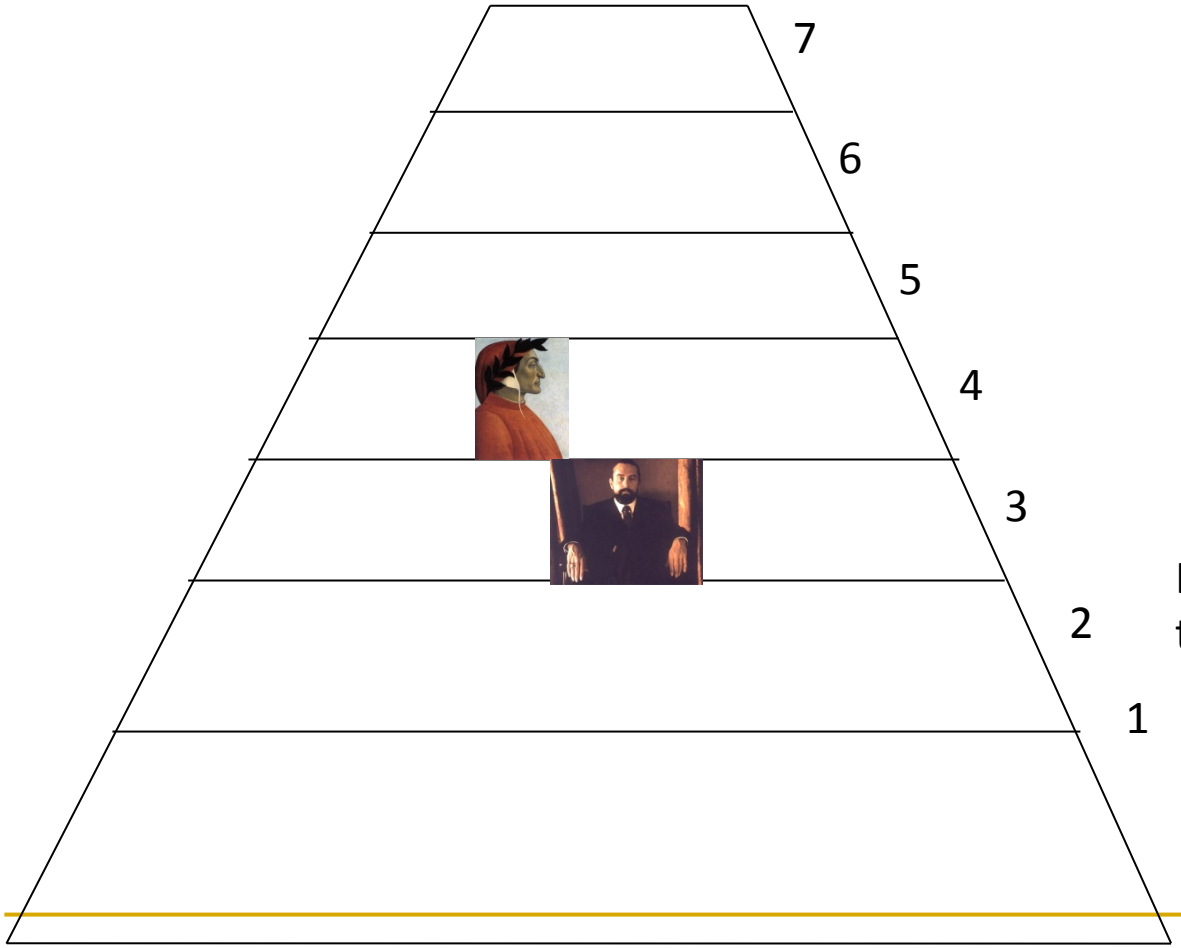
(Hansen, Koucky, Miltersen, LICS' 09)



If Dante wins, he proceeds to the next terrace

Dante in Purgatory

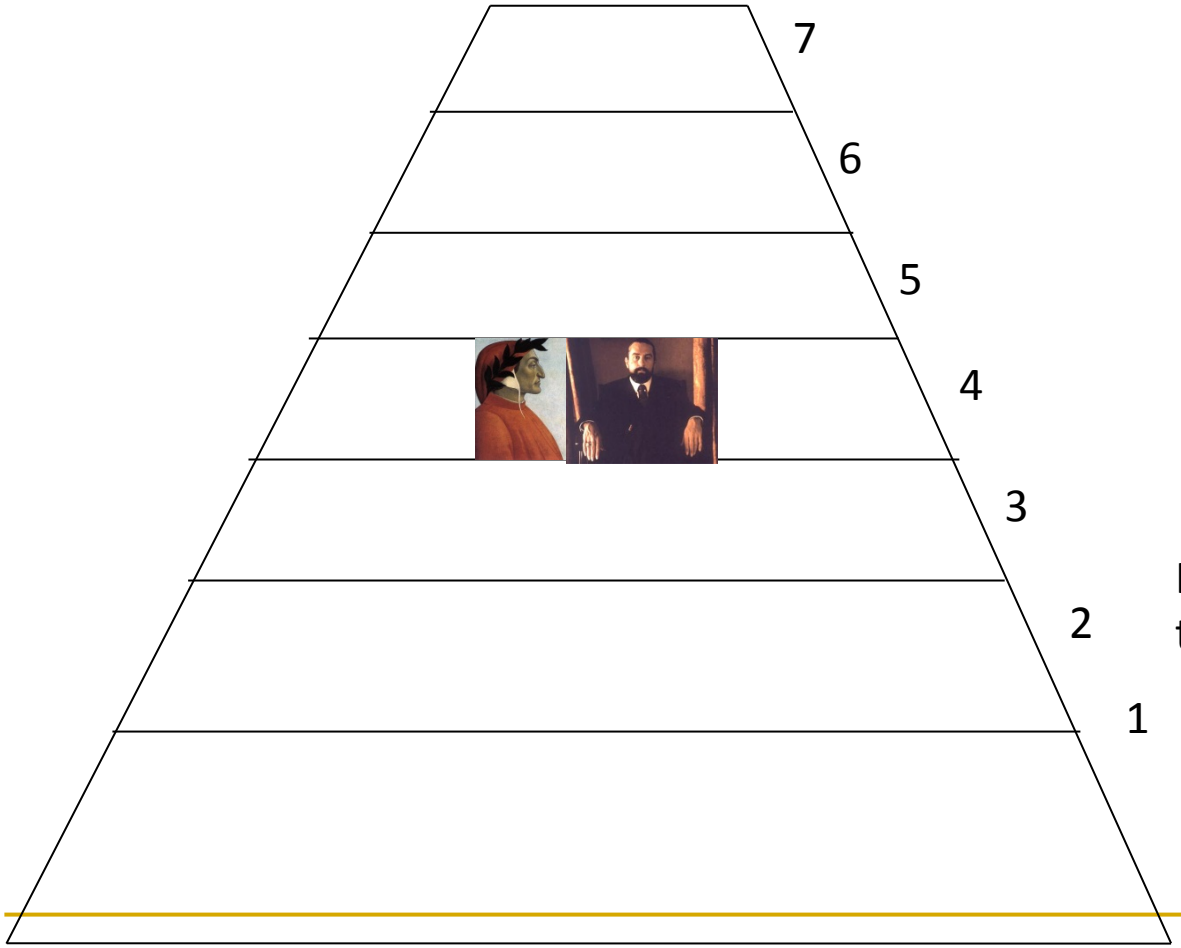
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds to the next terrace

Dante in Purgatory

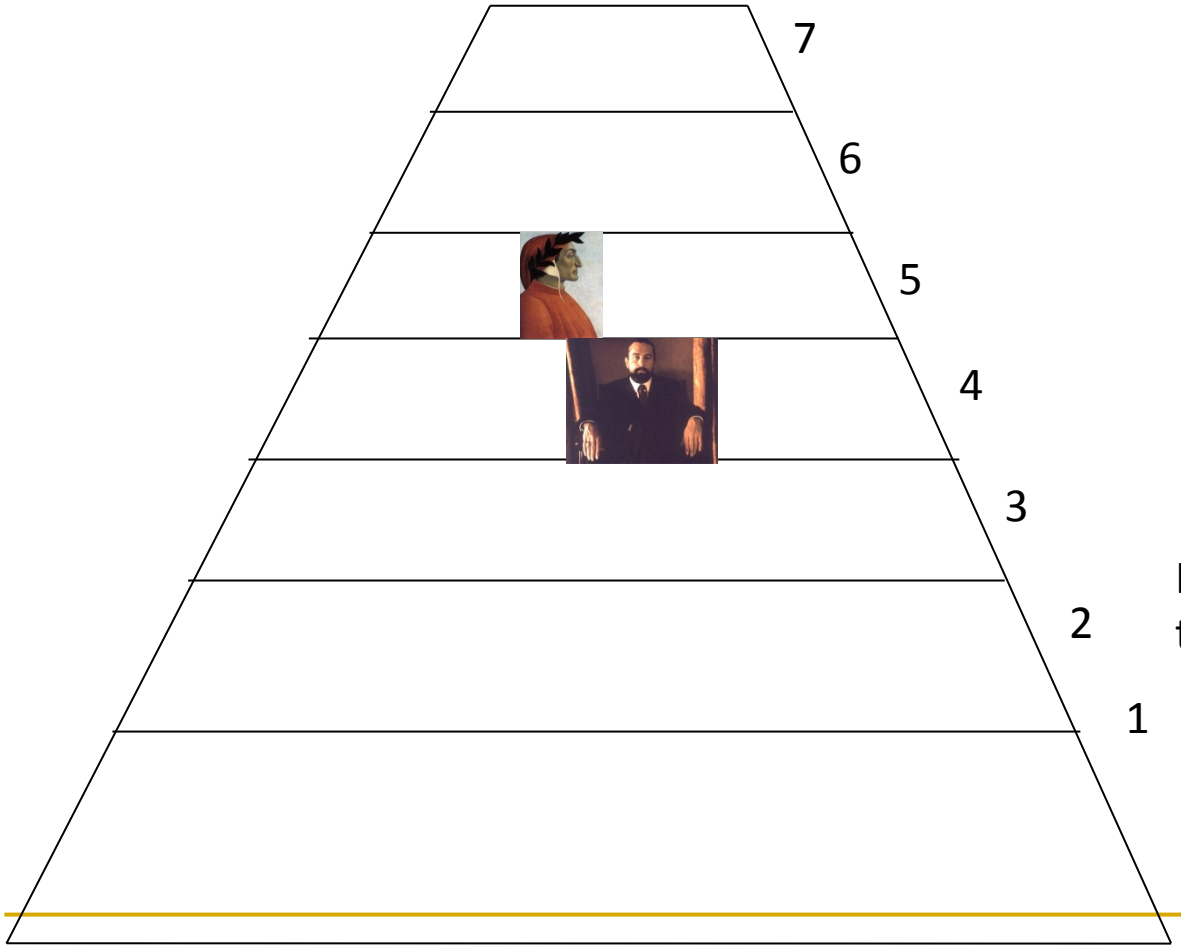
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds to the next terrace

Dante in Purgatory

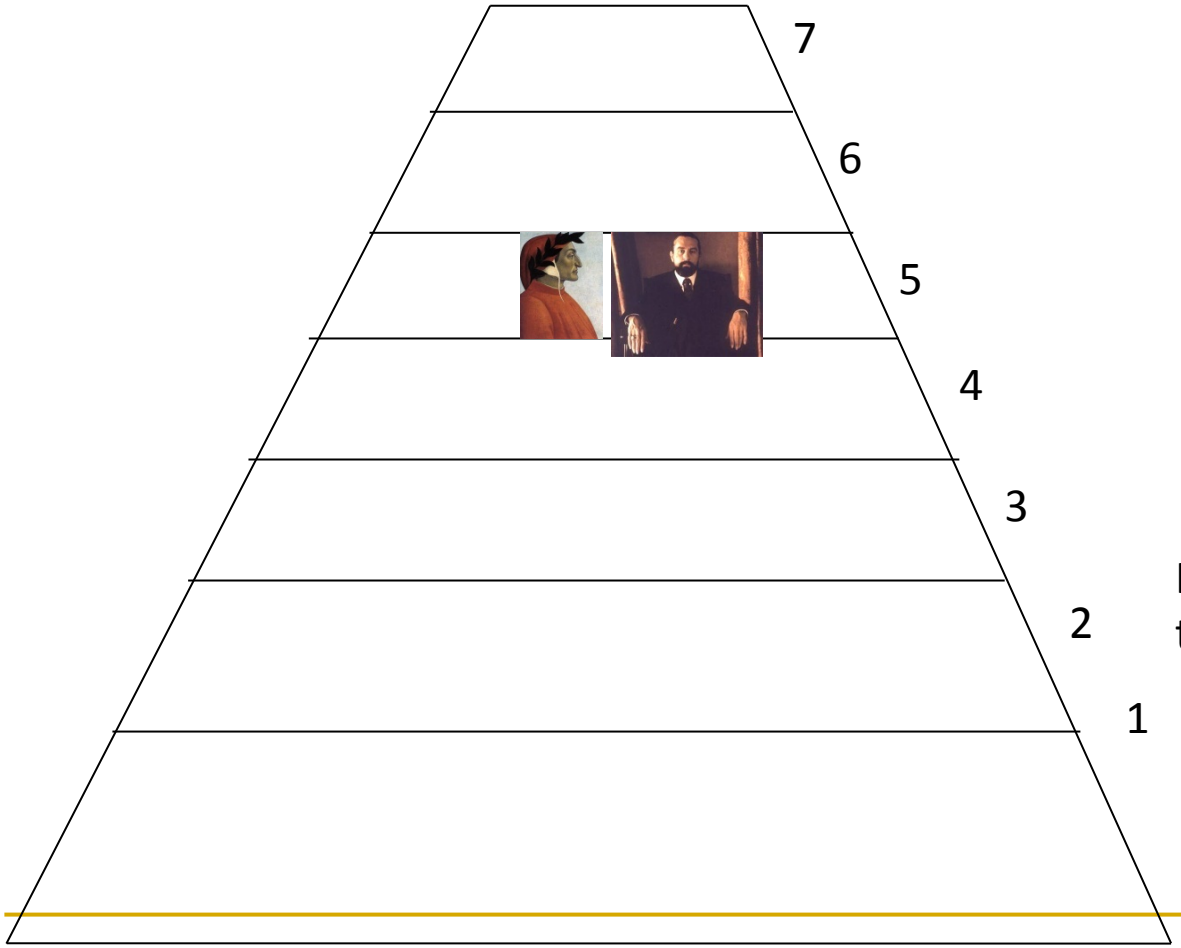
(Hansen, Koucky, Miltersen, LICS' 09)



If Dante wins, he proceeds to the next terrace

Dante in Purgatory

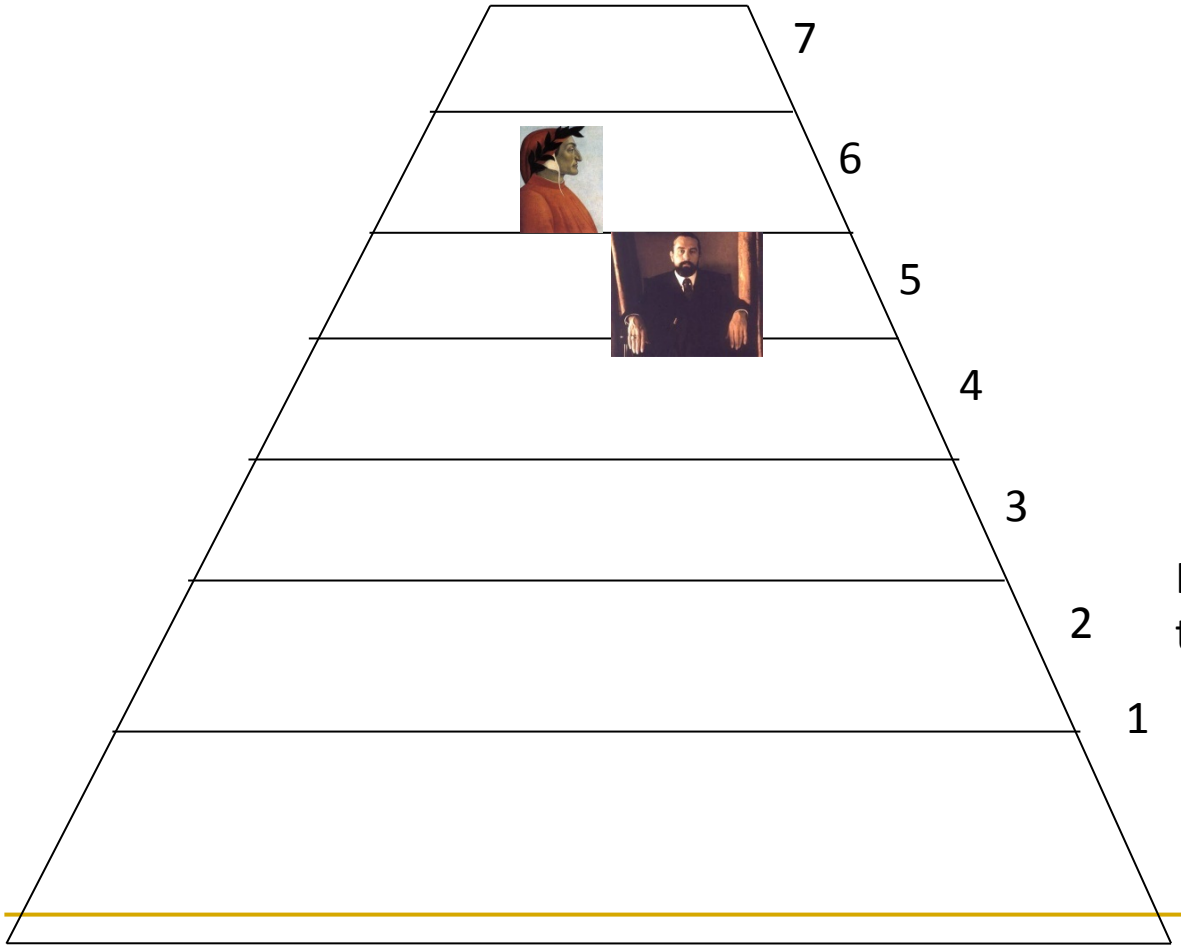
(Hansen, Koucky, Miltersen, LICS' 09)



If Dante wins, he proceeds to the next terrace

Dante in Purgatory

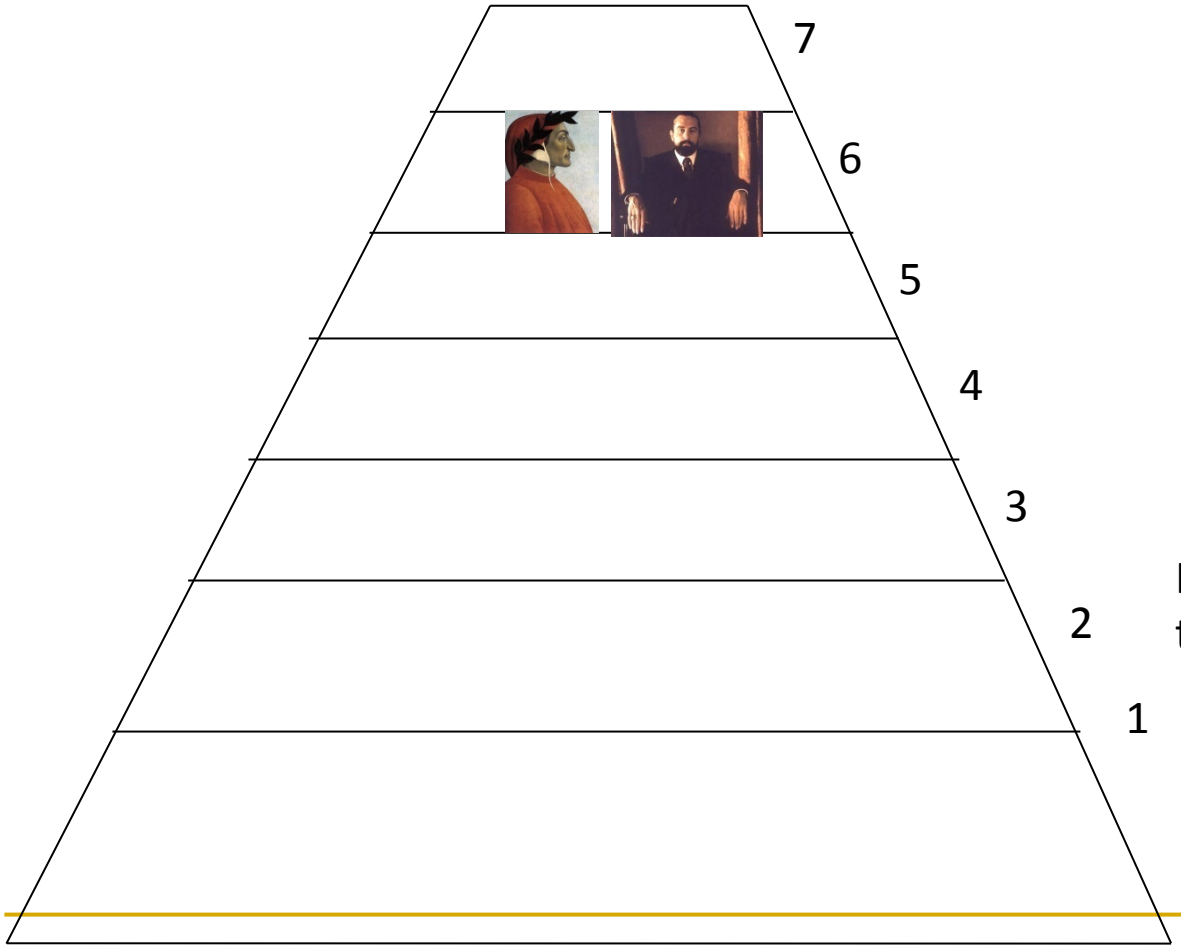
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds to the next terrace

Dante in Purgatory

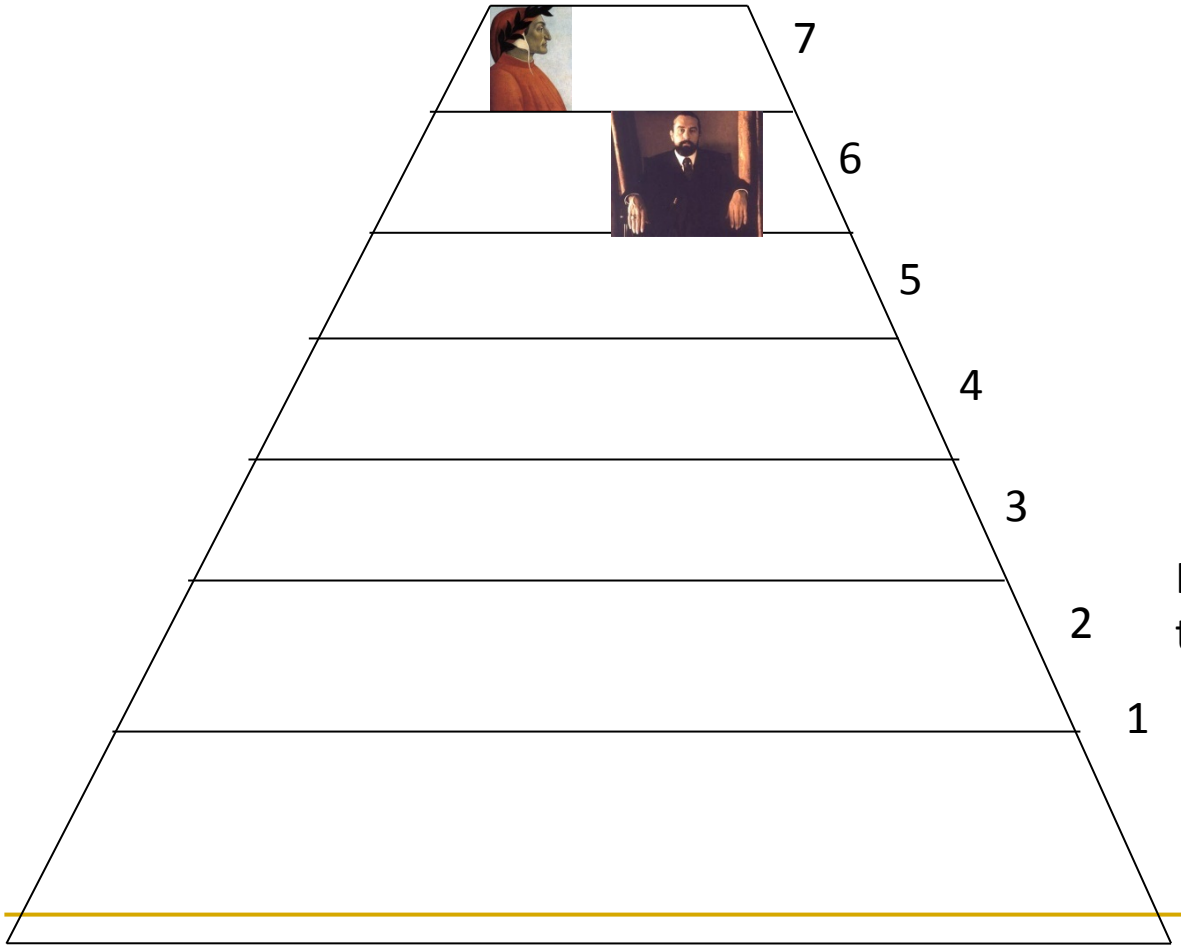
(Hansen, Koucky, Miltersen, LICS'09)



If Dante wins, he proceeds to the next terrace

Dante in Purgatory

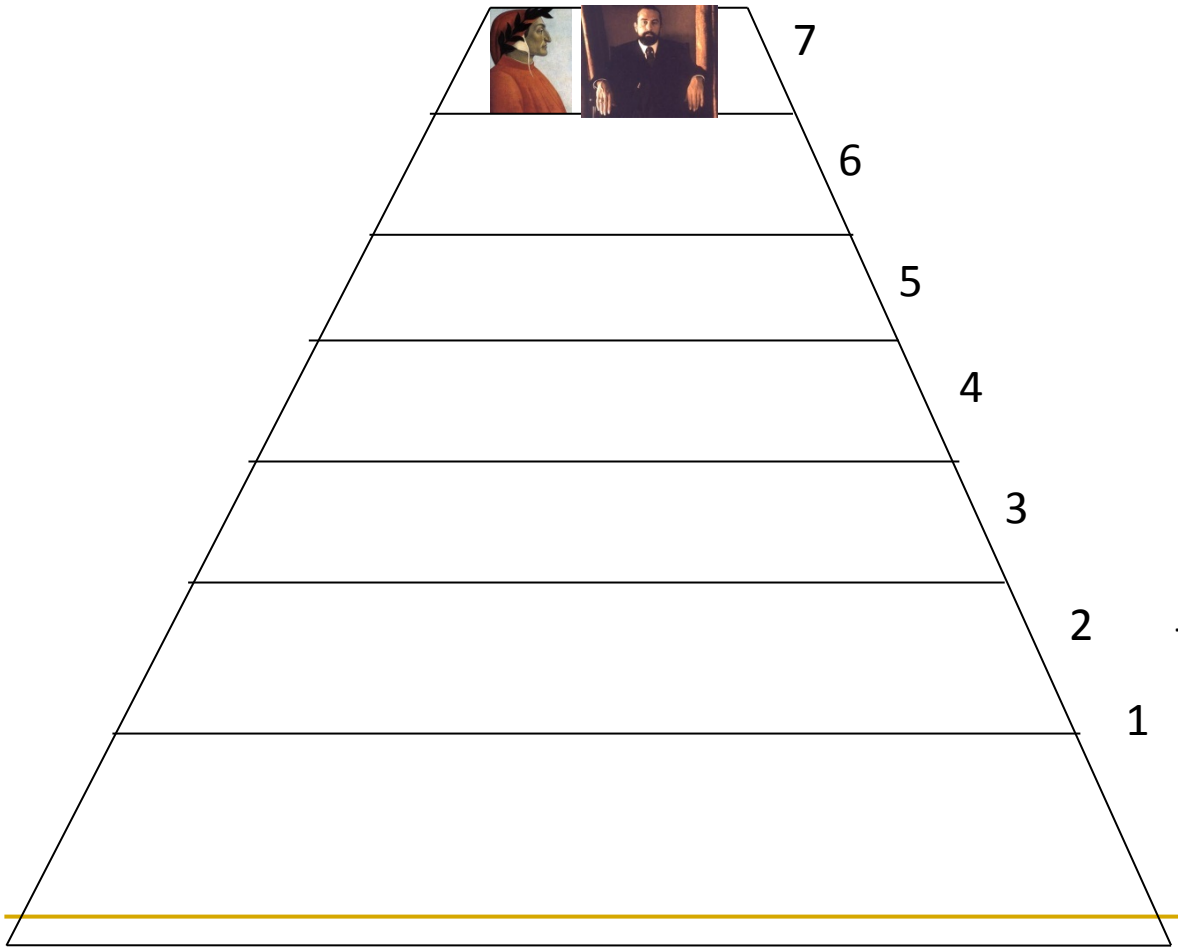
(Hansen, Koucky, Miltersen, LICS' 09)



If Dante wins, he proceeds to the next terrace

Dante in Purgatory

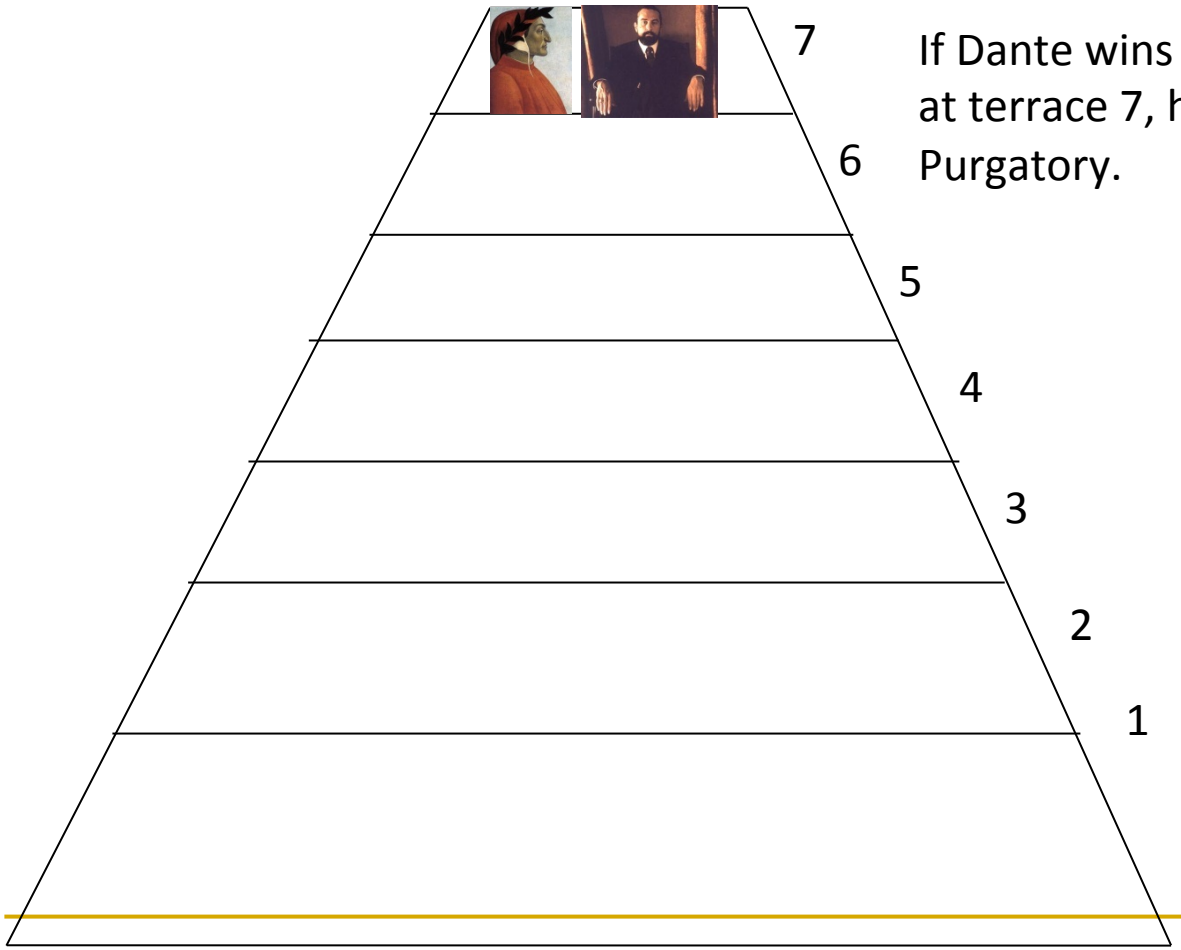
(Hansen, Koucky, Miltersen, LICS' 09)



If Dante wins, he proceeds to the next terrace

Dante in Purgatory

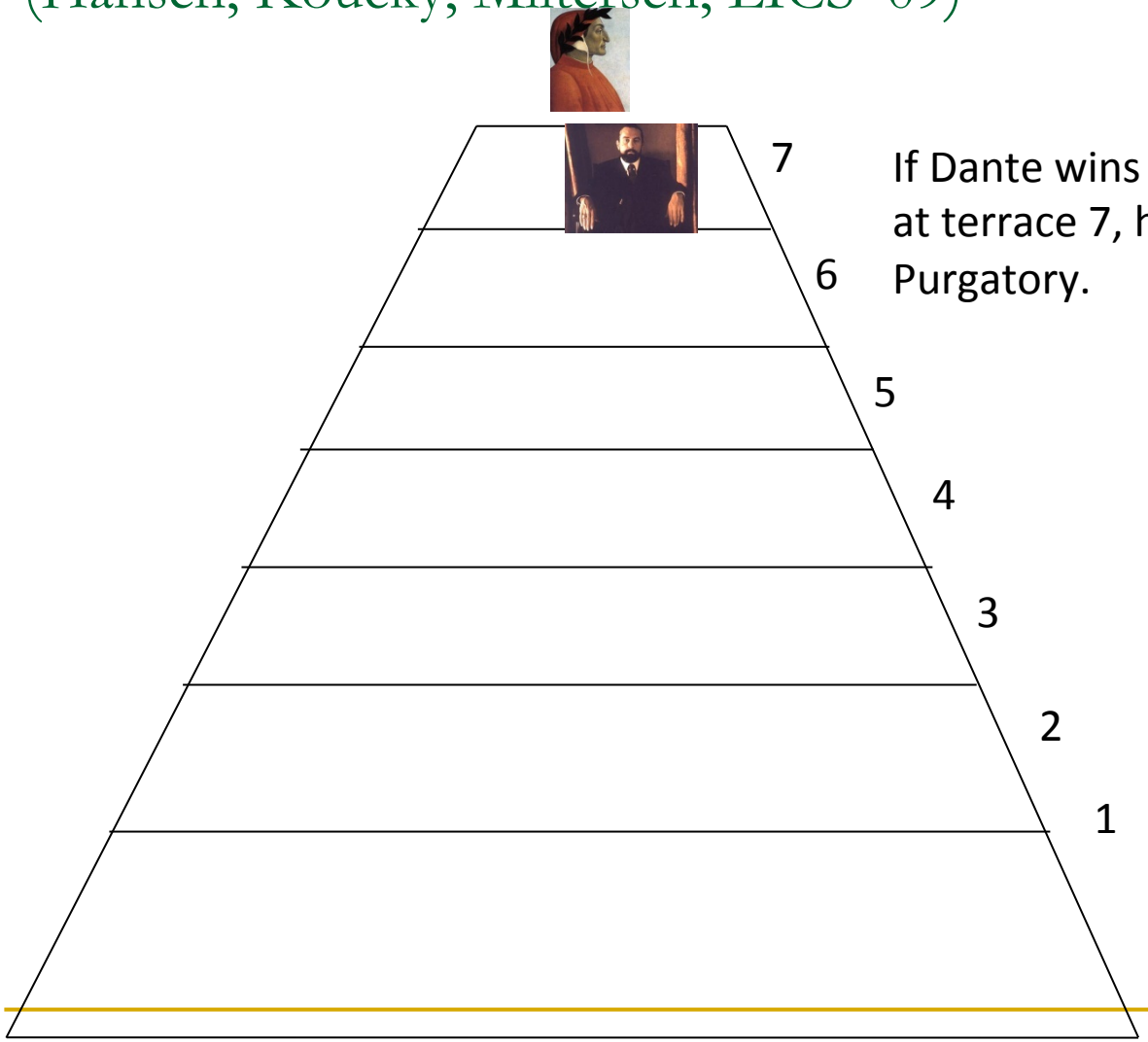
(Hansen, Koucky, Miltersen, LICS'09)



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

Dante in Purgatory

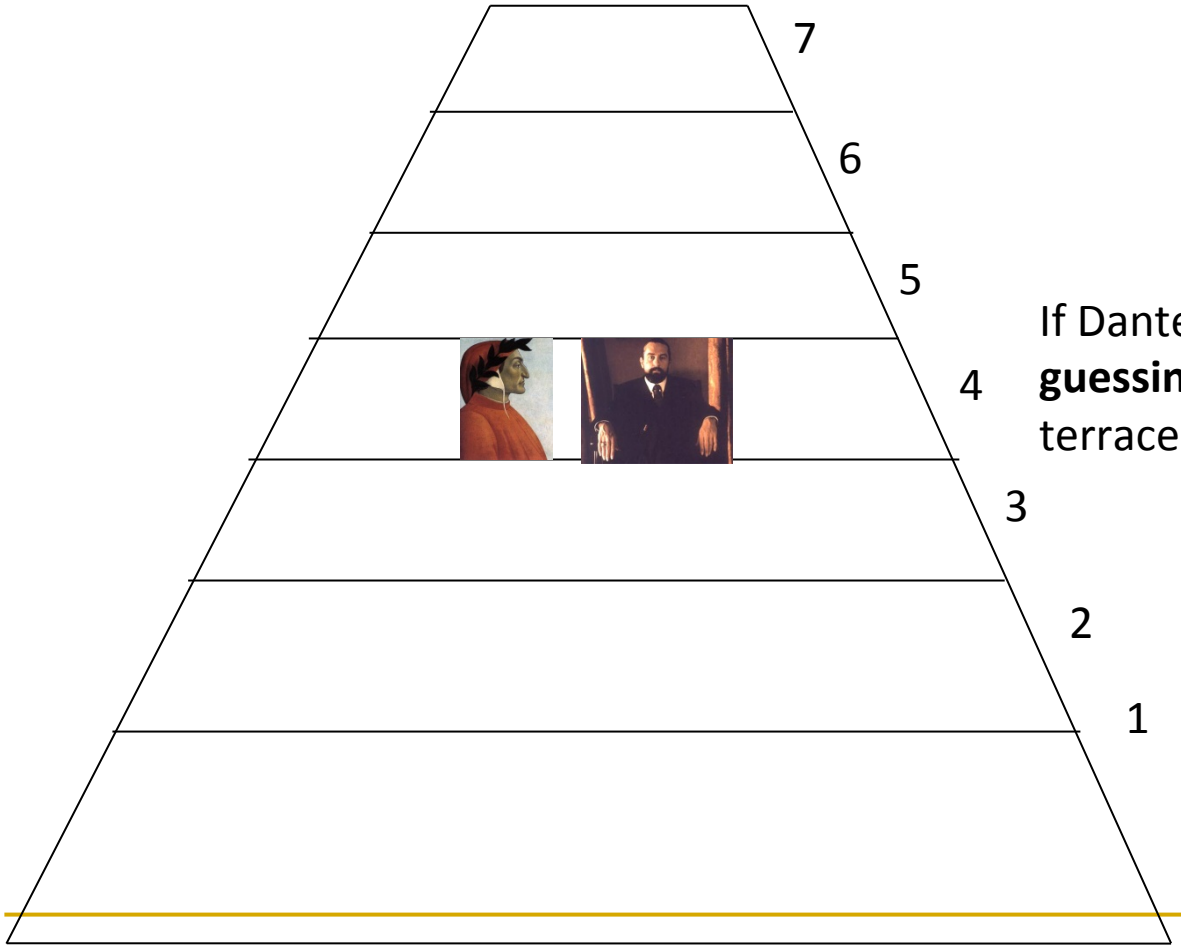
(Hansen, Koucky, Miltersen, LICS'09)



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

Dante in Purgatory

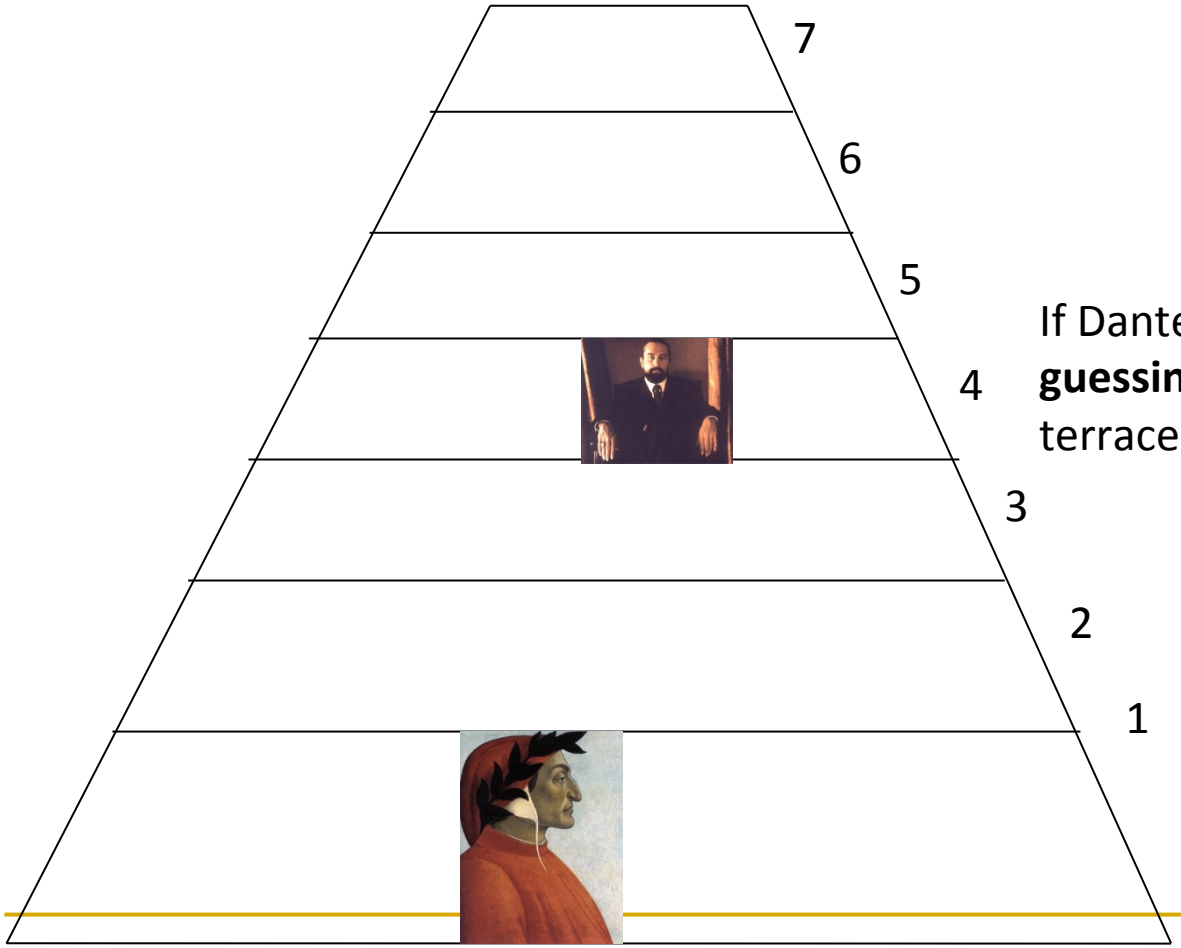
(Hansen, Koucky, Miltersen, LICS'09)



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

Dante in Purgatory

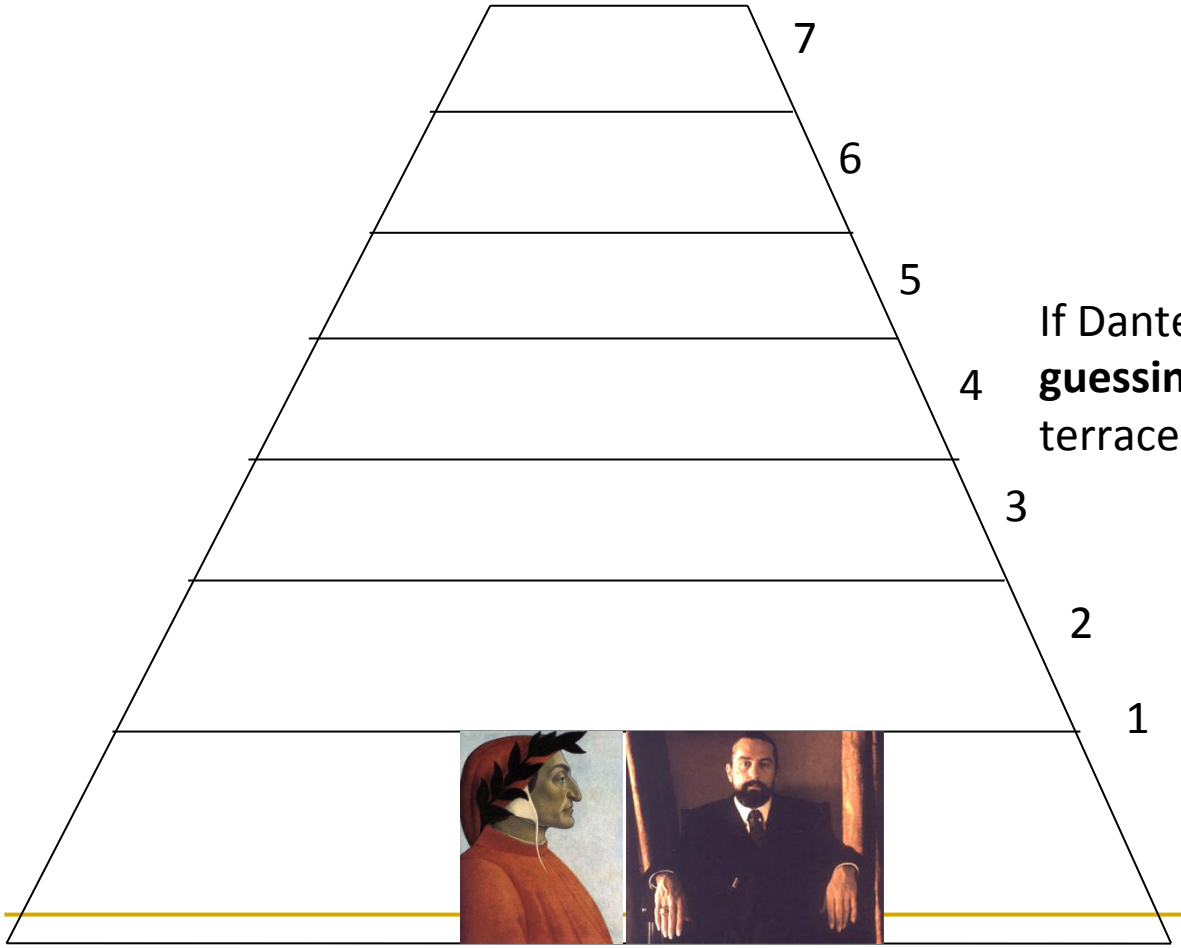
(Hansen, Koucky, Miltersen, LICS' 09)



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

Dante in Purgatory

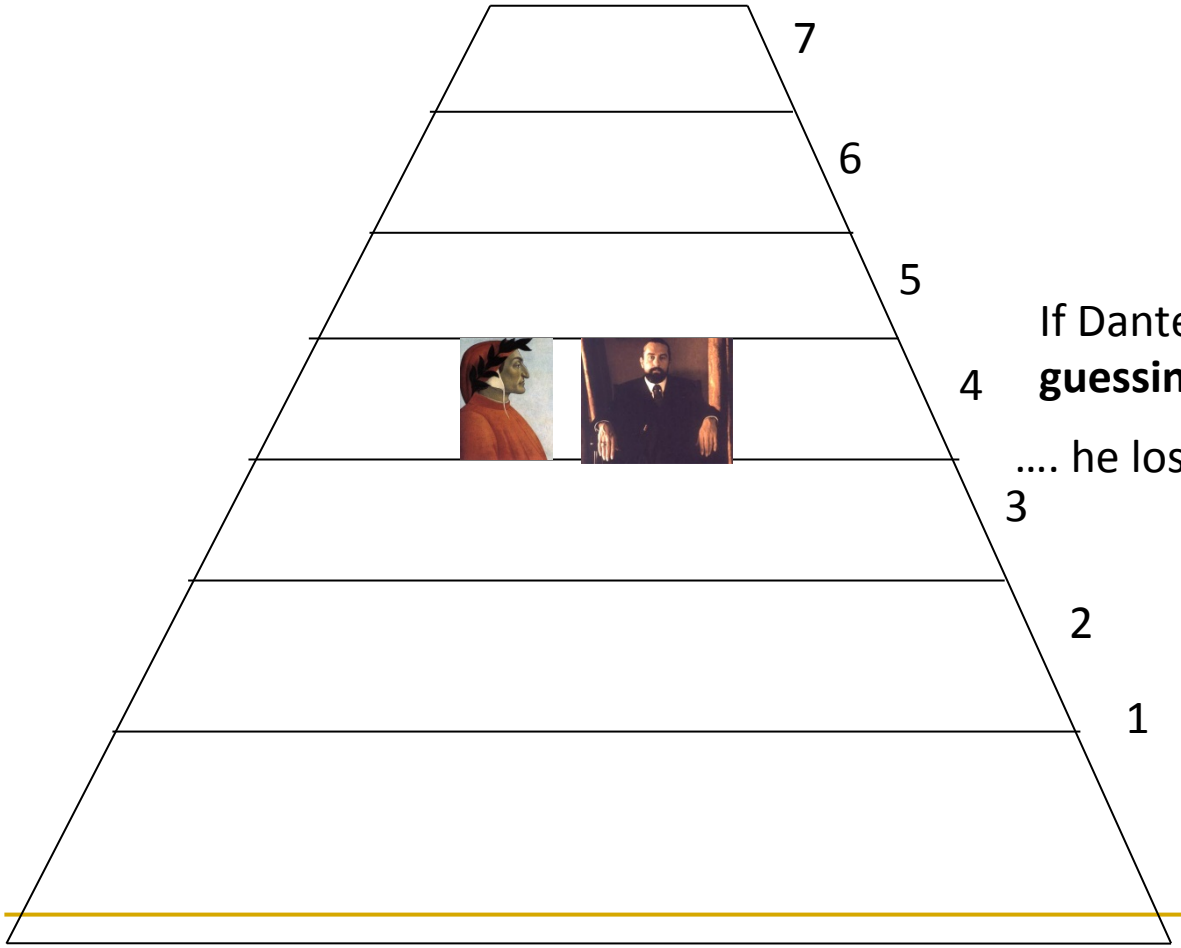
(Hansen, Koucky, Miltersen, LICS' 09)



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!!!!

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
$> 2*10^{65}$	0.9
$> 10^{128}$	0.99

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.