

Finite Horizon Analysis of Markov Chains with the Mur φ Verifier

Giuseppe Della Penna Benedetto Intrigila Igor Melatti

Dip. di Informatica, Università di L'Aquila

Enrico Tronci Marisa Venturini Zilli

Dip. di Informatica, Università di Roma "La Sapienza"

- Markov Chain analysis

- Markov Chain analysis
- Given the description of a Markov Chain, it verifies a PCTL property

- Markov Chain analysis
- Given the description of a Markov Chain, it verifies a PCTL property
- PCTL: Probabilistic CTL
 - $A \rightarrow P_{\geq \alpha}[\text{true } UB]$
 - $A \rightarrow P_{> \alpha}[\text{true } U^{\leq k} B]$

- Markov Chain analysis
- Given the description of a Markov Chain, it verifies a PCTL property
- PCTL: Probabilistic CTL
 - $A \rightarrow P_{\geq \alpha}[\text{true } UB]$
 - $A \rightarrow P_{> \alpha}[\text{true } U^{\leq k} B]$
- Very few available probabilistic model checkers

- Markov Chain analysis
- Given the description of a Markov Chain, it verifies a PCTL property
- PCTL: Probabilistic CTL
 - $A \rightarrow P_{\geq \alpha}[\text{true } UB]$
 - $A \rightarrow P_{> \alpha}[\text{true } U^{\leq k} B]$
- Very few available probabilistic model checkers
 - PRISM

- Markov Chain analysis
- Given the description of a Markov Chain, it verifies a PCTL property
- PCTL: Probabilistic CTL
 - $A \rightarrow P_{\geq \alpha}[\text{true } UB]$
 - $A \rightarrow P_{> \alpha}[\text{true } U^{\leq k} B]$
- Very few available probabilistic model checkers
 - PRISM
 - Two Towers

- Markov Chain analysis
- Given the description of a Markov Chain, it verifies a PCTL property
- PCTL: Probabilistic CTL
 - $A \rightarrow P_{\geq \alpha}[\text{true } UB]$
 - $A \rightarrow P_{> \alpha}[\text{true } U^{\leq k} B]$
- Very few available probabilistic model checkers
 - PRISM
 - Two Towers
 - FHP-Mur φ (new)

PRISM Probabilistic Symbolic Model Checker

PRISM Probabilistic Symbolic Model Checker

- State-of-the-art probabilistic model checker

PRISM Probabilistic Symbolic Model Checker

- State-of-the-art probabilistic model checker
- Implicit verification algorithm (MTBDD-based)

PRISM Probabilistic Symbolic Model Checker

- State-of-the-art probabilistic model checker
- Implicit verification algorithm (MTBDD-based)
- It allows to verify three types of Markov Chains:

DTMC, with PCTL are the “classic” ones, here we will deal with these only

MDP, with PCTL non-determinism added

CTMC, with CSL continuous time managed

PRISM Probabilistic Symbolic Model Checker

- State-of-the-art probabilistic model checker
- Implicit verification algorithm (MTBDD-based)
- It allows to verify three types of Markov Chains:
 - DTMC, with PCTL** are the “classic” ones, here we will deal with these only
 - MDP, with PCTL** non-determinism added
 - CTMC, with CSL** continuous time managed
- Three verification modalities:
 - totally MTBDD-based (calculating fix points)
 - algebraic (on the Markov Chain transition matrix)
 - an hybrid modality between the two previous ones

- Finite Horizon Probabilistic-Mur φ

- Finite Horizon Probabilistic-Mur φ
- Explicit probabilistic model checker

- Finite Horizon Probabilistic-Mur φ
- Explicit probabilistic model checker
 - symbolic and explicit verification are not comparable in non-probabilistic model checking

- Finite Horizon Probabilistic-Mur φ
- Explicit probabilistic model checker
 - symbolic and explicit verification are not comparable in non-probabilistic model checking
 - we will show that this holds also for probabilistic model checking

- Finite Horizon Probabilistic-Mur φ
- Explicit probabilistic model checker
 - symbolic and explicit verification are not comparable in non-probabilistic model checking
 - we will show that this holds also for probabilistic model checking
- Mur φ modified in the input language and in the verification algorithm

- Finite Horizon Probabilistic-Mur φ
- Explicit probabilistic model checker
 - symbolic and explicit verification are not comparable in non-probabilistic model checking
 - we will show that this holds also for probabilistic model checking
- Mur φ modified in the input language and in the verification algorithm
- Specialized in verifying a particular type of PCTL properties
 - $P_{\leq \alpha}[\text{true } U^{\leq k} \phi] \equiv Pr((\exists i \leq k \phi(\pi(i))) \mid \pi \in \text{Path}(\mathcal{M})) \leq \alpha$
 - ϕ is a boolean function defined on states

- Finite Horizon Probabilistic-Mur φ
- Explicit probabilistic model checker
 - symbolic and explicit verification are not comparable in non-probabilistic model checking
 - we will show that this holds also for probabilistic model checking
- Mur φ modified in the input language and in the verification algorithm
- Specialized in verifying a particular type of PCTL properties
 - $P_{\leq \alpha}[\text{true } U^{\leq k} \phi] \equiv Pr((\exists i \leq k \phi(\pi(i))) \mid \pi \in \text{Path}(\mathcal{M})) \leq \alpha$
 - ϕ is a boolean function defined on states
 - If ϕ models an error, we are asking if the error probability is acceptable

- We added finite precision real numbers and probabilities:

- We added finite precision real numbers and probabilities:
 - on the initial states (initial probability distribution)
 - * n initial states with probability p_1, \dots, p_n
 - * $\sum_{i=1}^n p_i = 1$ has always to hold

- We added finite precision real numbers and probabilities:
 - on the initial states (initial probability distribution)
 - * n initial states with probability p_1, \dots, p_n
 - * $\sum_{i=1}^n p_i = 1$ has always to hold
 - on the rules (they now define a Markov Chain transition function)
 - * s_1, \dots, s_n successor states of s with probability p_1, \dots, p_n
 - * $\sum_{i=1}^n p_i = 1$ has always to hold

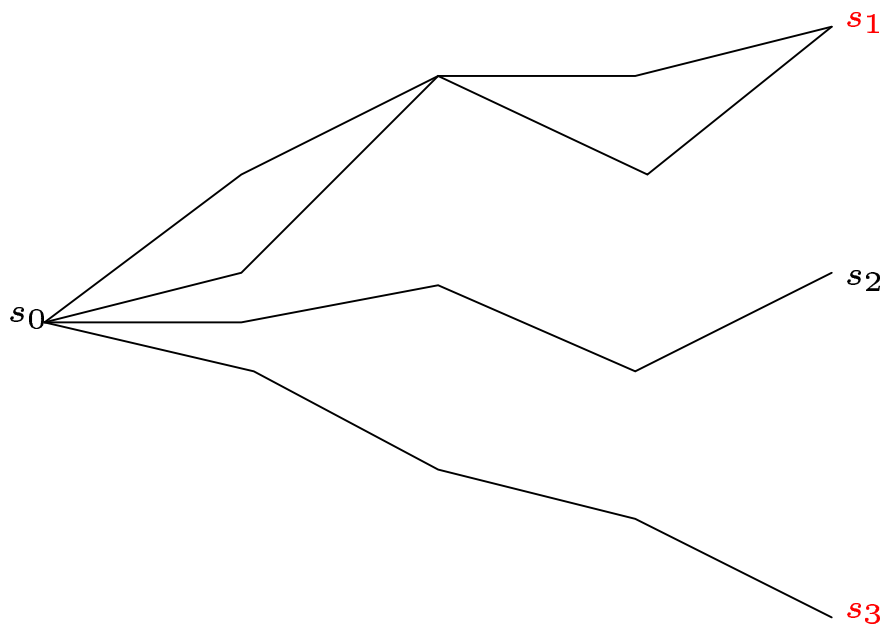
- We added finite precision real numbers and probabilities:
 - on the initial states (initial probability distribution)
 - * n initial states with probability p_1, \dots, p_n
 - * $\sum_{i=1}^n p_i = 1$ has always to hold
 - on the rules (they now define a Markov Chain transition function)
 - * s_1, \dots, s_n successor states of s with probability p_1, \dots, p_n
 - * $\sum_{i=1}^n p_i = 1$ has always to hold
 - on the invariant to be verified
 - * property to be verified: is the probability of the event “an error state (i.e., not satisfying the invariant) is reachable within a given number of steps” less than a given α ?
 - * i.e., does $Pr(\exists i \leq k : \phi(\pi(i)) \mid \pi \text{ is a Markov Chain path}) \leq \alpha$ hold?
 - * equivalent to the PCTL formula $P_{\leq \alpha}[\text{true } U^{\leq k} \phi]$

- Let ErrProb be the probability of $[\text{true } U^{\leq k} \phi]$

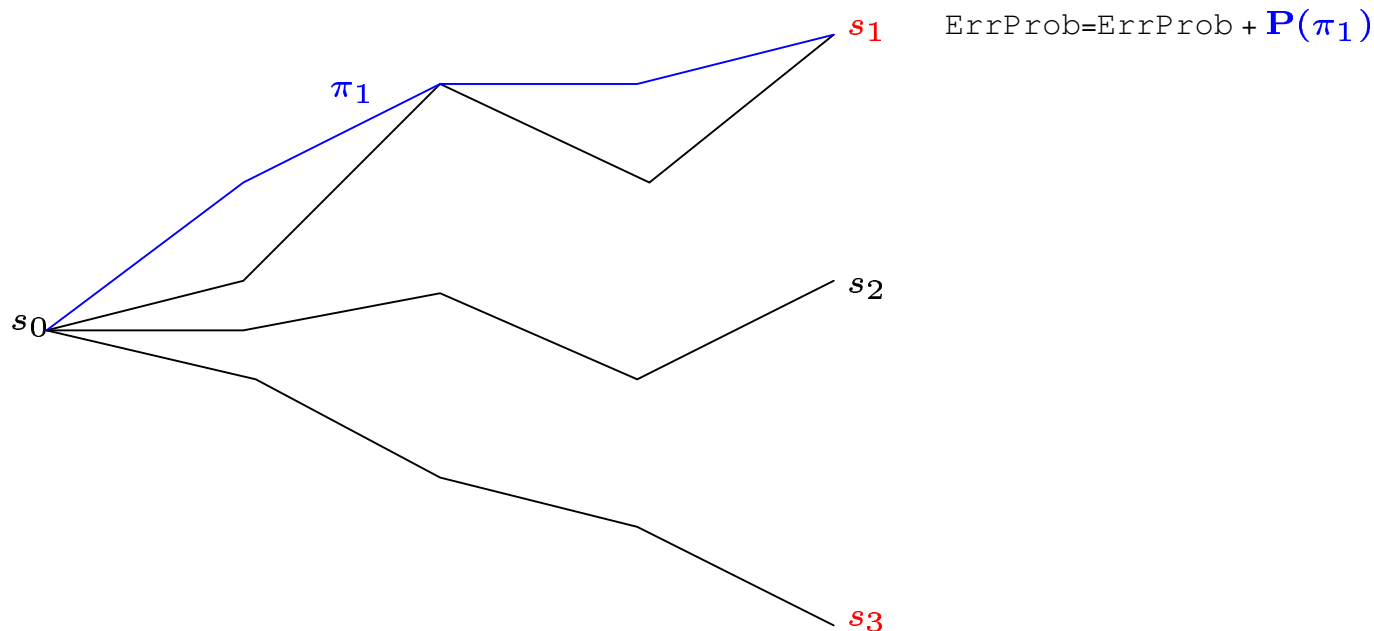
- Let ErrProb be the probability of $[\text{true } U^{\leq k} \phi]$
- Initially, ErrProb = 0

- Let ErrProb be the probability of $[\text{true } U^{\leq k} \phi]$
- Initially, ErrProb = 0
- ErrProb is incremented whenever a state s is reached such that ϕ holds in s

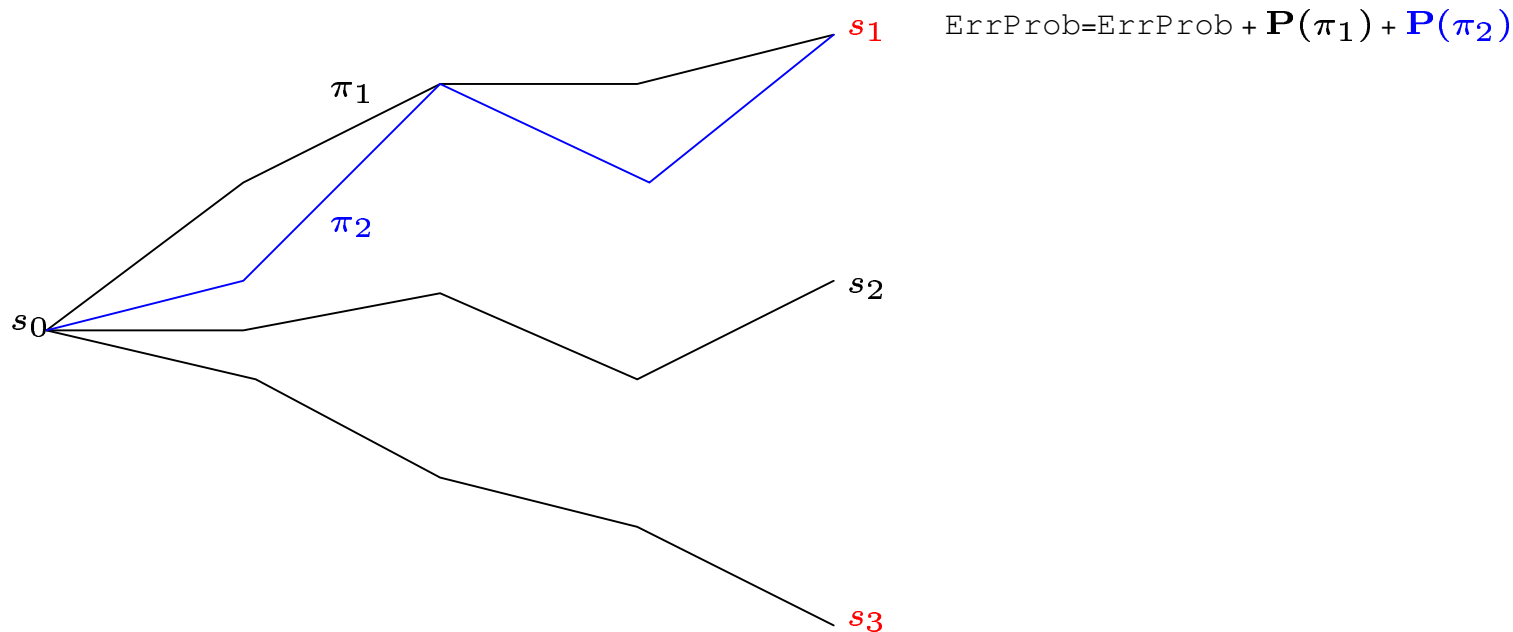
- Let ErrProb be the probability of $[\text{true } U^{\leq k} \phi]$
- Initially, ErrProb = 0
- ErrProb is incremented whenever a state s is reached such that ϕ holds in s



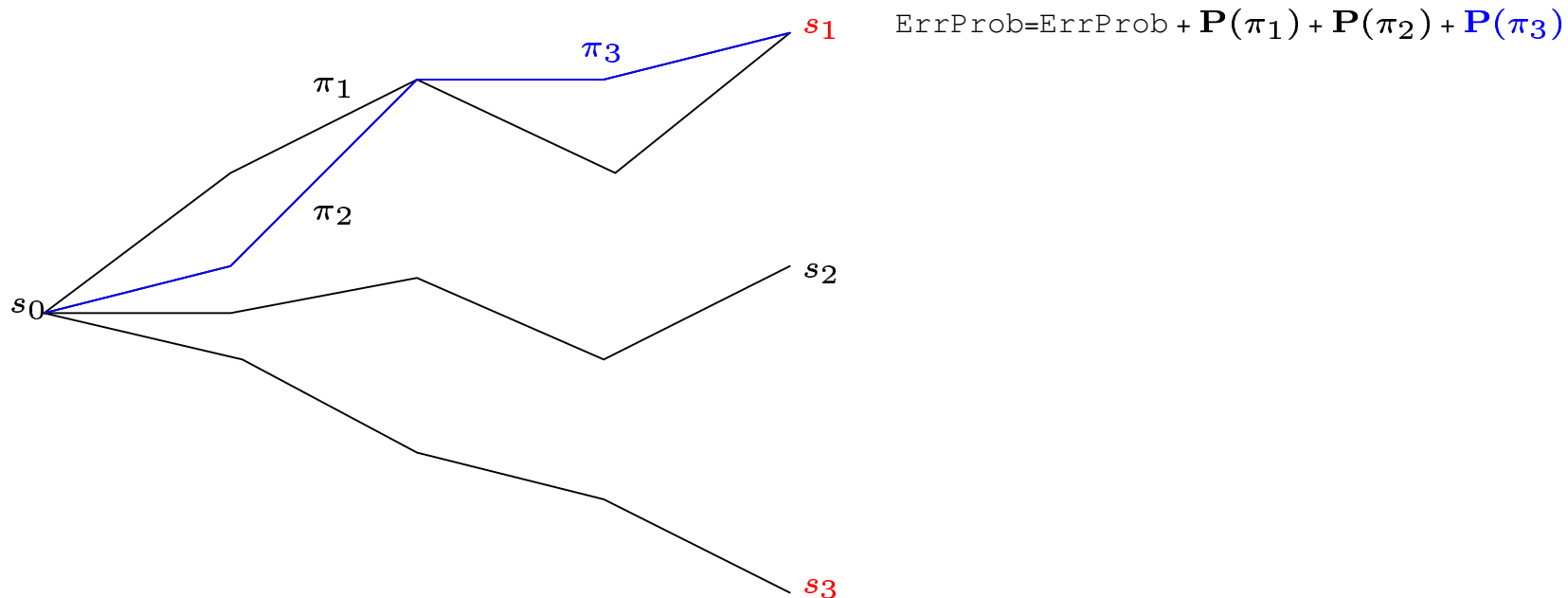
- Let ErrProb be the probability of $[\text{true } U^{\leq k} \phi]$
- Initially, ErrProb = 0
- ErrProb is incremented whenever a state s is reached such that ϕ holds in s



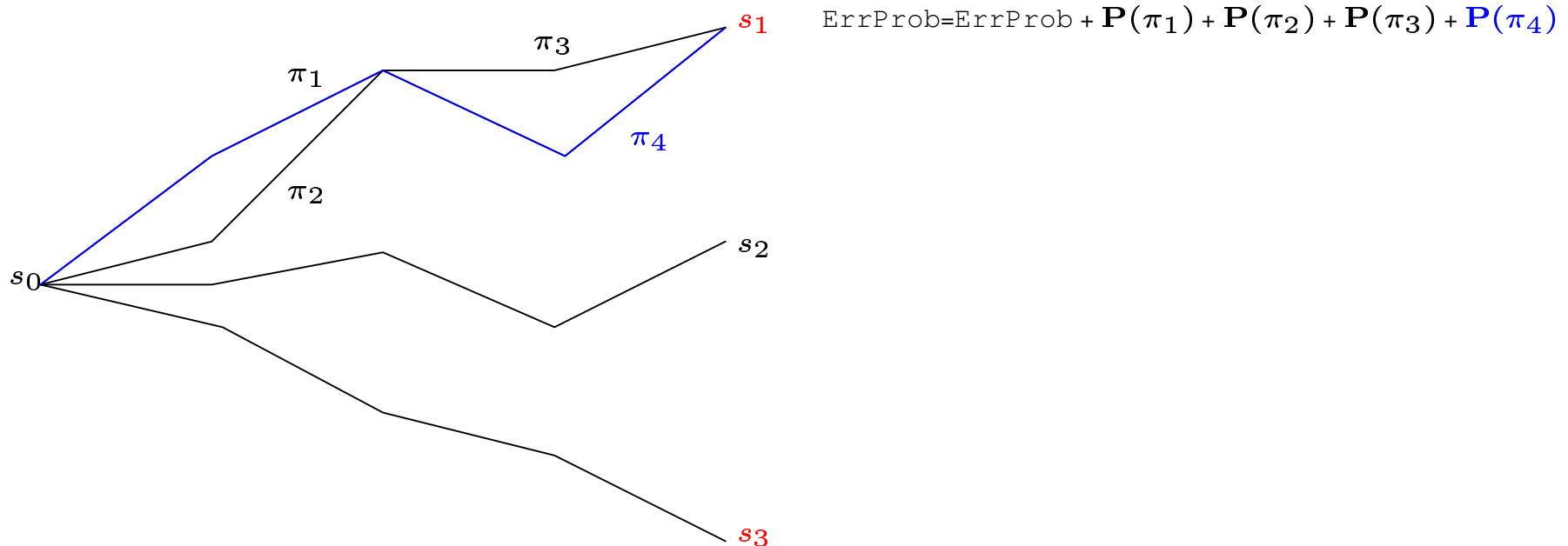
- Let ErrProb be the probability of $[\text{true } U^{\leq k} \phi]$
- Initially, ErrProb = 0
- ErrProb is incremented whenever a state s is reached such that ϕ holds in s



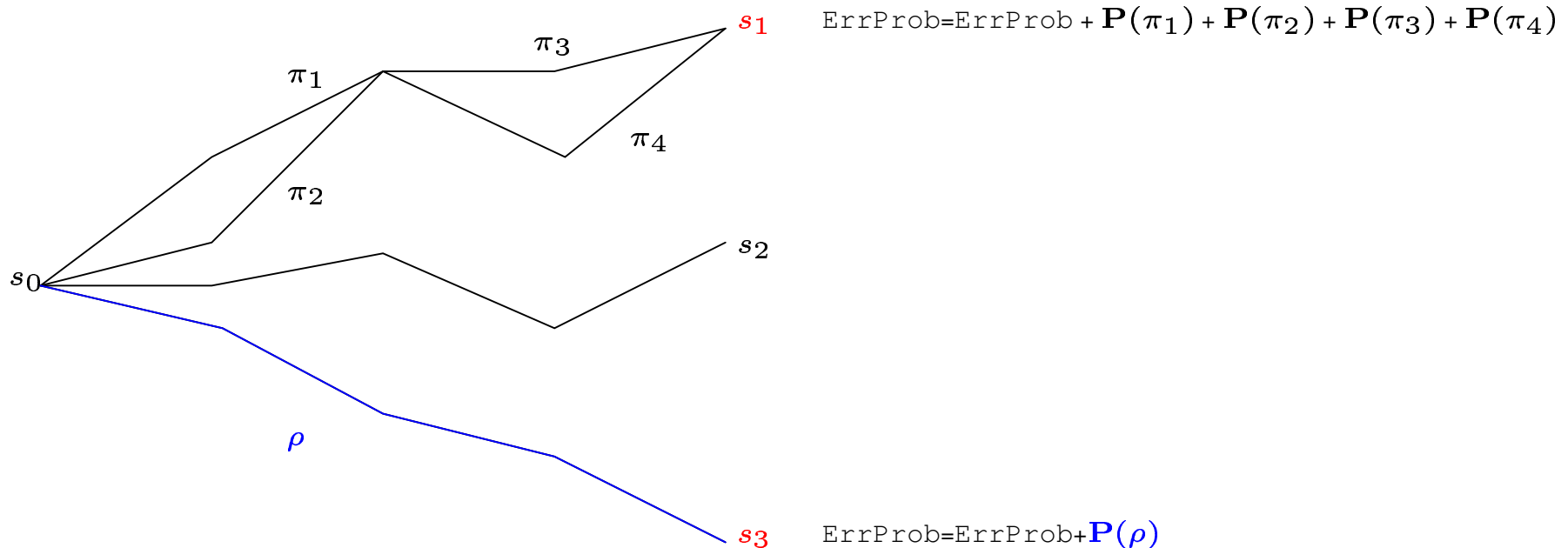
- Let ErrProb be the probability of $[\text{true } U^{\leq k} \phi]$
- Initially, ErrProb = 0
- ErrProb is incremented whenever a state s is reached such that ϕ holds in s



- Let ErrProb be the probability of $[\text{true } U^{\leq k} \phi]$
- Initially, ErrProb = 0
- ErrProb is incremented whenever a state s is reached such that ϕ holds in s



- Let ErrProb be the probability of $[\text{true } U^{\leq k} \phi]$
- Initially, ErrProb = 0
- ErrProb is incremented whenever a state s is reached such that ϕ holds in s



- $\text{ErrProb} = \text{ErrProb} + p$ where p is the probability to reach s from the initial states
 - If there are m paths to s , p is the sum of the probabilities of these m paths

- $\text{ErrProb} = \text{ErrProb} + p$ where p is the probability to reach s from the initial states
 - If there are m paths to s , p is the sum of the probabilities of these m paths
- Already visited states are not to be discarded, since they can be reached via different paths

- $\text{ErrProb} = \text{ErrProb} + p$ where p is the probability to reach s from the initial states
 - If there are m paths to s , p is the sum of the probabilities of these m paths
- Already visited states are not to be discarded, since they can be reached via different paths
- It is necessary to compute paths probabilities

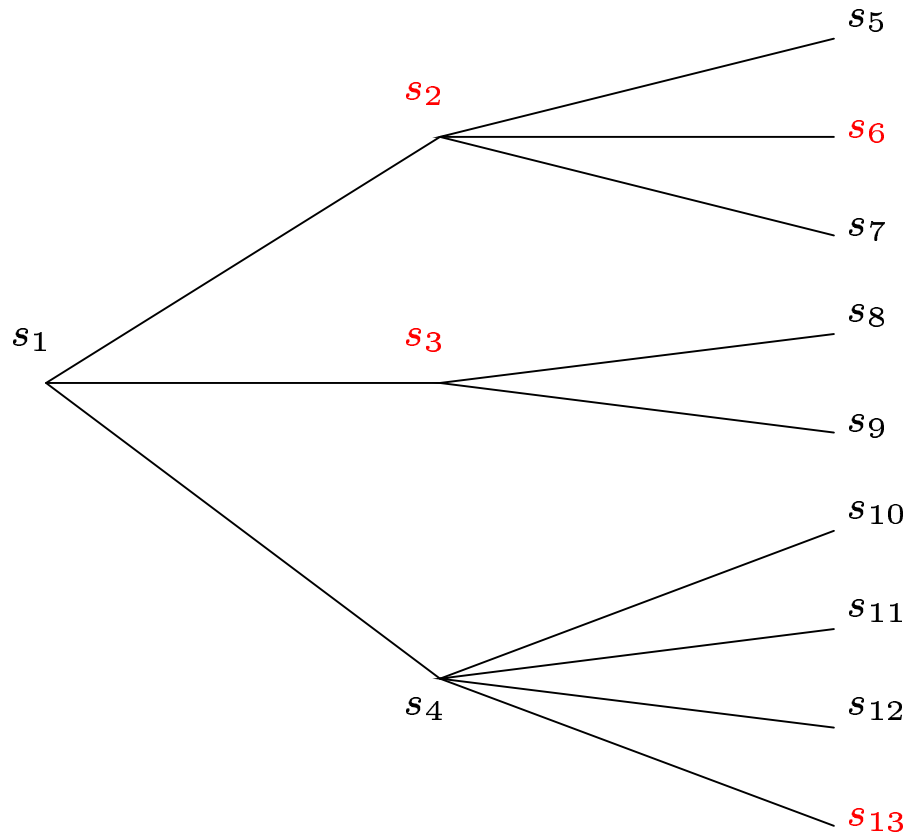
- $\text{ErrProb} = \text{ErrProb} + p$ where p is the probability to reach s from the initial states
 - If there are m paths to s , p is the sum of the probabilities of these m paths
- Already visited states are not to be discarded, since they can be reached via different paths
- It is necessary to compute paths probabilities
 - The initial states are reached with a given probability

- $\text{ErrProb} = \text{ErrProb} + p$ where p is the probability to reach s from the initial states
 - If there are m paths to s , p is the sum of the probabilities of these m paths
- Already visited states are not to be discarded, since they can be reached via different paths
- It is necessary to compute paths probabilities
 - The initial states are reached with a given probability
 - If s is reached with probability p , and s goes to t with probability q , then t is reached with probability pq

- $\text{ErrProb} = \text{ErrProb} + p$ where p is the probability to reach s from the initial states
 - If there are m paths to s , p is the sum of the probabilities of these m paths
- Already visited states are not to be discarded, since they can be reached via different paths
- It is necessary to compute paths probabilities
 - The initial states are reached with a given probability
 - If s is reached with probability p , and s goes to t with probability q , then t is reached with probability pq
 - The additive property for ErrProb holds for every reachable state

- $\text{ErrProb} = \text{ErrProb} + p$ where p is the probability to reach s from the initial states
 - If there are m paths to s , p is the sum of the probabilities of these m paths
- Already visited states are not to be discarded, since they can be reached via different paths
- It is necessary to compute paths probabilities
 - The initial states are reached with a given probability
 - If s is reached with probability p , and s goes to t with probability q , then t is reached with probability pq
 - The additive property for ErrProb holds for every reachable state
- The reachability analysis is stopped after the k -th step

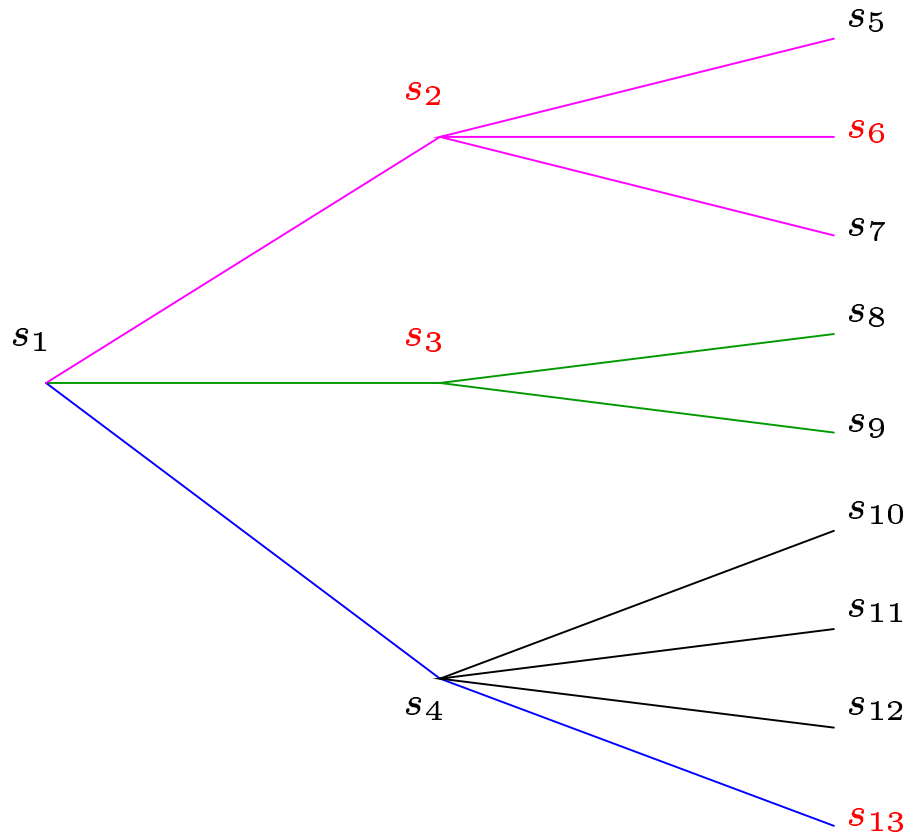
- $\text{ErrProb} = \text{ErrProb} + p$ where p is the probability to reach s from the initial states
 - If there are m paths to s , p is the sum of the probabilities of these m paths
- Already visited states are not to be discarded, since they can be reached via different paths
- It is necessary to compute paths probabilities
 - The initial states are reached with a given probability
 - If s is reached with probability p , and s goes to t with probability q , then t is reached with probability pq
 - The additive property for ErrProb holds for every reachable state
- The reachability analysis is stopped after the k -th step
- States that satisfy ϕ are not expanded



Uniform probability

s_2, s_6, s_3, s_{13}

are the states in which ϕ holds (error states)

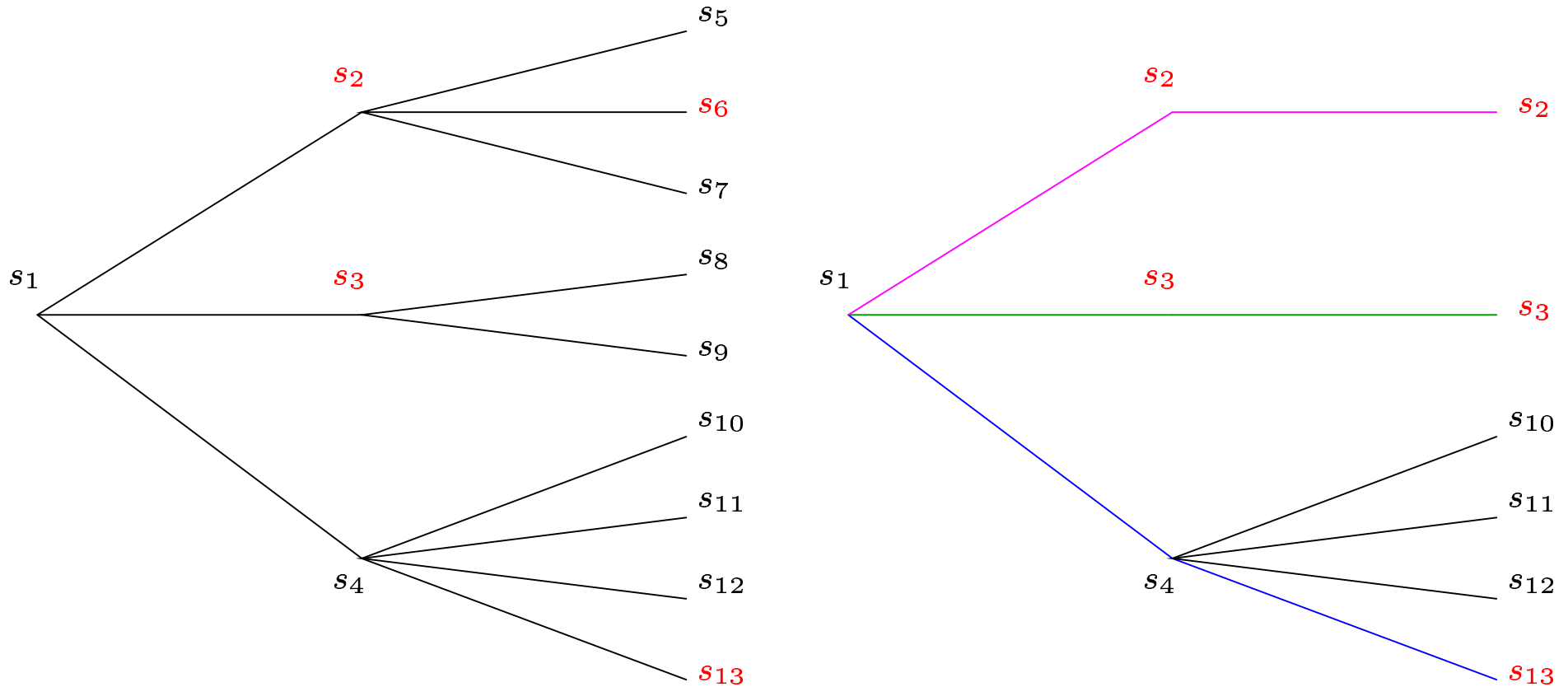


Uniform probability

s_2, s_6, s_3, s_{13}

are the states in which ϕ holds (error states)

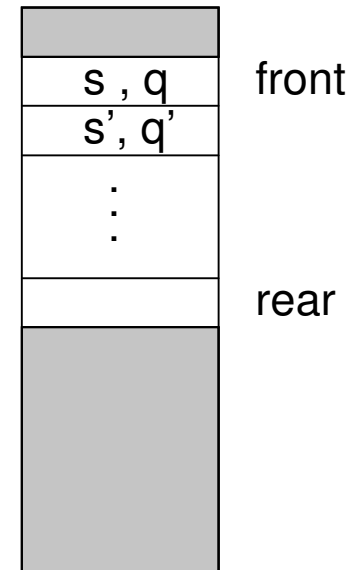
$$P[\text{true } U^{\leq 2} \phi] = \frac{1}{3} \frac{1}{3} + \frac{1}{3} \frac{1}{3} + \frac{1}{3} \frac{1}{3} + \frac{1}{3} \frac{1}{2} + \frac{1}{3} \frac{1}{2} + \frac{1}{3} \frac{1}{4} = \frac{3}{4}$$



If s is such that $\phi(s)$ holds then the Markov Chain starting from s is forced to cycle on s

$$P[\text{true } U^{\leq 2} \phi] = \frac{1}{3} \mathbf{1} + \frac{1}{3} \mathbf{1} + \frac{1}{3} \frac{1}{4} = \frac{3}{4} \text{ again}$$

Queue

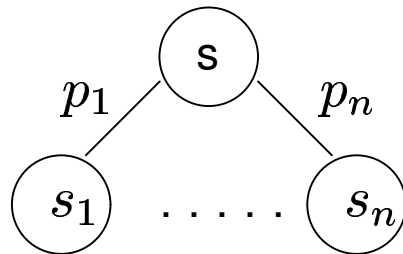


s : state to be expanded

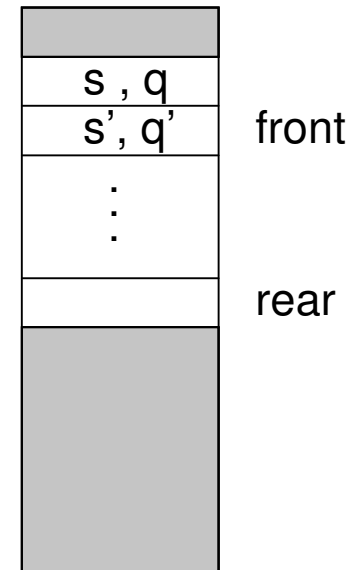
q : probability of reaching s in $l - 1$ levels

s' next state to be expanded

State explosion virtually never occurs: if the queue grows too much, disk storage is used



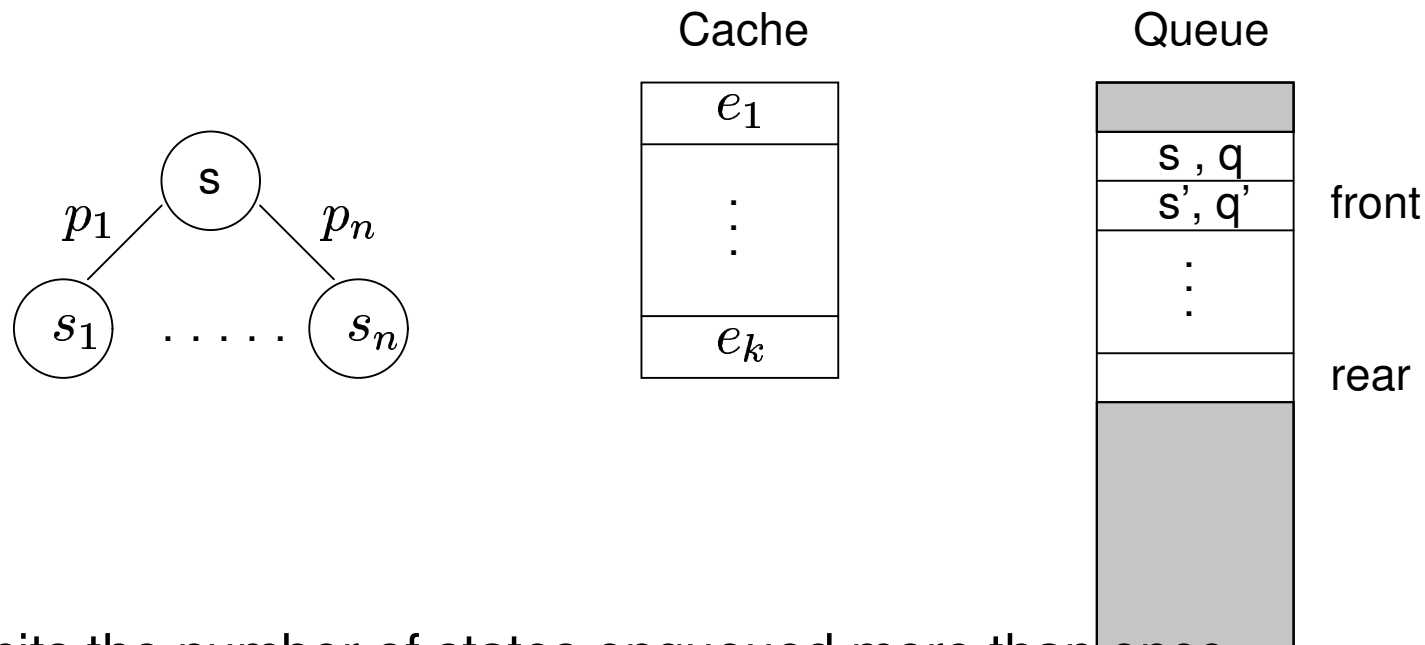
Queue



s : state to be expanded

p_1, \dots, p_n : rules whose probability is strictly positive in s

$$\left. \begin{array}{l} \forall i. p_i \in [0, 1] \\ \sum_{i=1}^n p_i = 1 \end{array} \right\} \text{Conditions to have a Markov Chain}$$

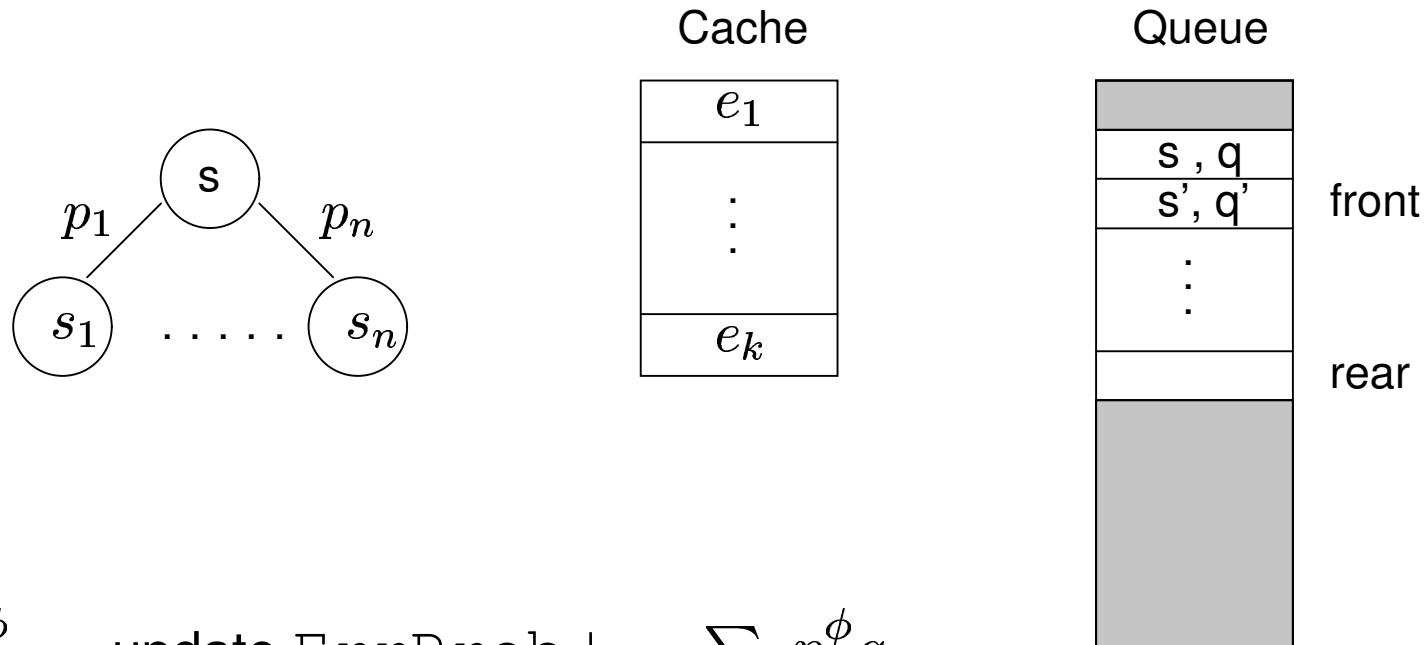


Cache: limits the number of states enqueued more than once

$\forall i. e_i$ is empty or stores a pair (state, probability)

$s_1^\phi, \dots, s_m^\phi, p_1^\phi, \dots, p_m^\phi$ states among the s_i in which ϕ holds (error states)
and their transition probabilities

$s_1^{\neg\phi}, \dots, s_{n-m}^{\neg\phi}, p_1^{\neg\phi}, \dots, p_{n-m}^{\neg\phi}$ “correct” states (all the other ones) and their
transition probabilities

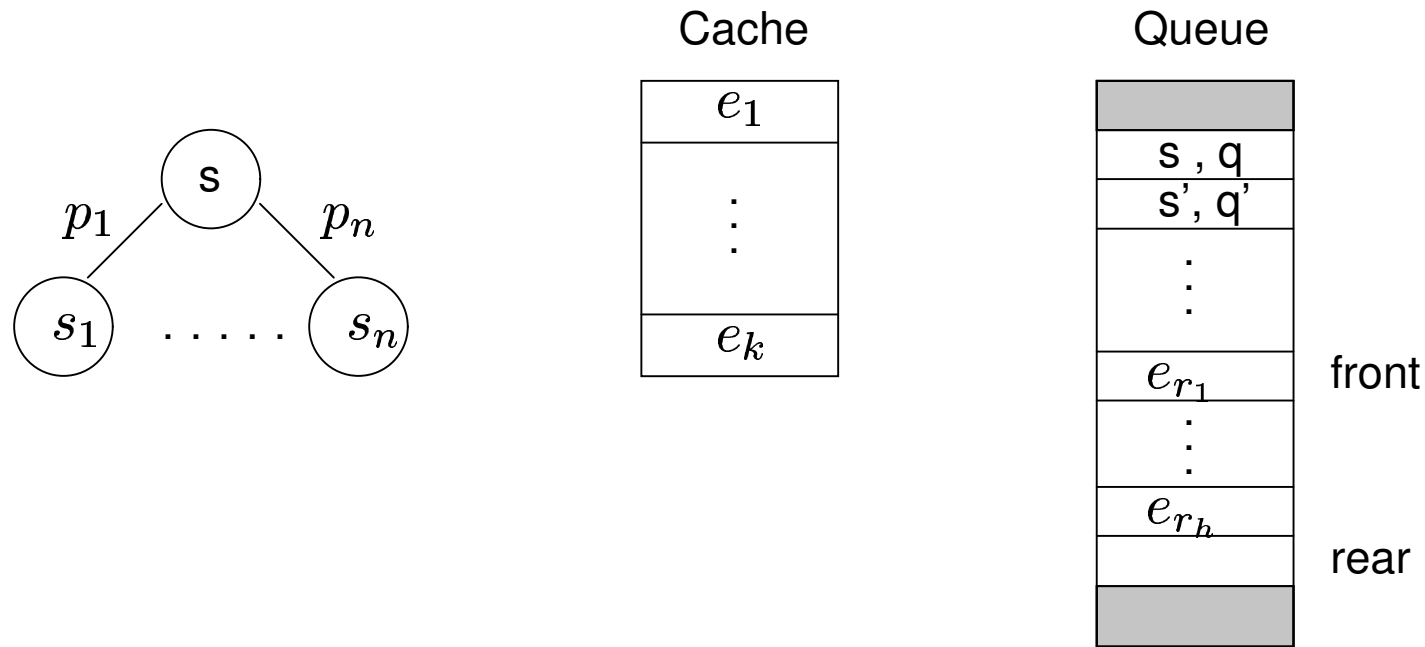


$$s_1^\phi, \dots, s_{n-m}^\phi \text{ update ErrProb} += \sum_i p_i^\phi q$$

At the end of the s expansion $\forall i. \exists j_i : e_{j_i} = (s_i^{-\phi}, q_i)$

$$\forall i. q_i =$$

$$\begin{cases} p_i^{-\phi} q & \text{if } s_i^{-\phi} \text{ was not in the Cache} \\ p_i^{-\phi} q + \text{Cache}[h(s_i^{-\phi})] \cdot \text{prob} & \text{otherwise} \end{cases}$$

**swap**All non-empty cache entries (e_{r_i}) are enqueued

All cache entries will now result empty

 s' next state to be expanded after the enqueue of $s_m^{-\neg\phi}$ **BFS levels**

as before; each level changing is always preceded by a swap

Probabilistic dining philosophers Pnueli-Zuck (PZ) and Lehmann-Rabin (LR) protocols

Probabilistic dining philosophers Pnueli-Zuck (PZ) and Lehmann-Rabin (LR) protocols

PZ is there a positive probability that a philosopher

- become hungry
- choose the left fork first

Probabilistic dining philosophers Pnueli-Zuck (PZ) and Lehmann-Rabin (LR) protocols

PZ is there a positive probability that a philosopher

- become hungry
- choose the left fork first

LR the same as PZ, but

- is there a positive probability that a philosopher puts down the left fork first
- no philosopher will never wait more than a fixed number (N) of actions made by the other philosopher before making an action himself

Probabilistic dining philosophers Pnueli-Zuck (PZ) and Lehmann-Rabin (LR) protocols

PZ is there a positive probability that a philosopher

- become hungry
- choose the left fork first

LR the same as PZ, but

- is there a positive probability that a philosopher puts down the left fork first
- no philosopher will never wait more than a fixed number (N) of actions made by the other philosopher before making an action himself

Hybrid systems Verification of a turbogas control system, assuming a probability distribution on the user demand

NPHIL	MAX_WAIT	Probability	Mur ϕ Memory (MB)	PRISM Memory (MB)	Mur ϕ Time	PRISM Time
3	3	7.335194164e-05	200	0.9057	51.970 s	1.487 s
3	4	6.883132778e-10	200	1.6844	52.610 s	2.507 s
4	3	1.88985976e-06	200	28.1066	4 min	28.72 s
4	4	2.910383046e-12	200	66.2659	4 min	1 min
5	3	9.164495139e-08	200	916.8246	23 min	17 min
5	4	4.194304e-14	200	N/A	23 min	N/A
8	3	1.210429649e-10	1000	N/A	2 $\frac{1}{2}$ days	N/A

$P_{\leq 1.0}[\text{true } U^{\leq 20} \text{ a philosopher has waited for MAX_WAIT transitions}]$

Results on a 2-processors (both INTEL Pentium III 500Mhz) computer with 2GB of RAM

NPHIL: number of philosophers

MAX_WAIT: max waiting time for every philosopher

Implicit vs Explicit sometimes the former performs better than the latter,
sometimes not

Implicit vs Explicit sometimes the former performs better than the latter,
sometimes not

Probabilistic verification We showed that this holds for probabilistic
verification

Implicit vs Explicit sometimes the former performs better than the latter,
sometimes not

Probabilistic verification We showed that this holds for probabilistic
verification

Termination is not all, also time is important

Implicit vs Explicit sometimes the former performs better than the latter,
sometimes not

Probabilistic verification We showed that this holds for probabilistic
verification

Termination is not all, also time is important

- PRISM, if terminates, terminates faster than FHP-Mur φ

Implicit vs Explicit sometimes the former performs better than the latter,
sometimes not

Probabilistic verification We showed that this holds for probabilistic
verification

Termination is not all, also time is important

- PRISM, if terminates, terminates faster than FHP-Mur φ
- FHP-Mur φ virtually always terminates (thanks to the disk storage of the queue), but it could require too much time

Implicit vs Explicit sometimes the former performs better than the latter, sometimes not

Probabilistic verification We showed that this holds for probabilistic verification

Termination is not all, also time is important

- PRISM, if terminates, terminates faster than FHP-Mur φ
- FHP-Mur φ virtually always terminates (thanks to the disk storage of the queue), but it could require too much time
 - if the horizon is too much long, the verification will take a great amount of time

Implicit vs Explicit sometimes the former performs better than the latter, sometimes not

Probabilistic verification We showed that this holds for probabilistic verification

Termination is not all, also time is important

- PRISM, if terminates, terminates faster than FHP-Mur φ
- FHP-Mur φ virtually always terminates (thanks to the disk storage of the queue), but it could require too much time
 - if the horizon is too much long, the verification will take a great amount of time
 - PRISM execution time is not dependent from the horizon

Not comparable There are cases in which Mur φ is better, other in which PRISM is

Not comparable There are cases in which Mur φ is better, other in which PRISM is

PCTL formulas Only of a certain type in FHP-Mur φ

Not comparable There are cases in which Mur φ is better, other in which PRISM is

PCTL formulas Only of a certain type in FHP-Mur φ

FHP-Mur φ better when

- the transition function is based on (complex) mathematical operations
- the horizon is not too long

Not comparable There are cases in which Mur φ is better, other in which PRISM is

PCTL formulas Only of a certain type in FHP-Mur φ

FHP-Mur φ better when

- the transition function is based on (complex) mathematical operations
- the horizon is not too long

PRISM better in the other cases

Not comparable There are cases in which Mur φ is better, other in which PRISM is

PCTL formulas Only of a certain type in FHP-Mur φ

FHP-Mur φ better when

- the transition function is based on (complex) mathematical operations
- the horizon is not too long

PRISM better in the other cases

FHP-Mur φ is however a probabilistic model checker to be taken into account

More features for FHP-Mur φ and then comparison with PRISM

More features for FHP-Mur φ and then comparison with PRISM

- Handling of PCTL formulas like $A \rightarrow P_{\bowtie\alpha}[\text{true } U^{\leq k} \phi]$

More features for FHP-Mur φ and then comparison with PRISM

- Handling of PCTL formulas like $A \rightarrow P_{\bowtie\alpha}[\text{true } U^{\leq k} \phi]$
- Infinite horizon
 - Some precomputations will be necessary in these two cases

More features for FHP-Mur φ and then comparison with PRISM

- Handling of PCTL formulas like $A \rightarrow P_{\bowtie\alpha}[\text{true } U^{\leq k} \phi]$
- Infinite horizon
 - Some precomputations will be necessary in these two cases
- Continuous Markov Chains
 - Approximable to Discrete Time Markov Chain with an exponential distribution
 - The smaller the sampling step
 - * the lowest the approximation error
 - * the higher the execution time

- G. Della Penna, B. Intrigila, I. Melatti, E. Tronci, and M. V. Zilli *Finite Horizon Verification of Markov Chains with the Mur φ Verifier*, CHARME, L'Aquila, 2003
- G. Della Penna, B. Intrigila, I. Melatti, E. Tronci, and M. V. Zilli *Integrating RAM and Disk based Verification within the Mur φ Verifier*, CHARME, L'Aquila, 2003
- G. Della Penna, B. Intrigila, I. Melatti, E. Tronci, and M. V. Zilli *Finite Horizon Verification of Stochastic Process with the Mur φ Verifier*, ICTCS, Bertinoro (FC), 2003
- G. Della Penna, B. Intrigila, I. Melatti, M. Minichino, E. Ciancamerla, A. Parisse, E. Tronci, and M. V. Zilli *Automatic Verification of a Turbogas Control System with the Mur φ Verifier*, HSCC, Prague, 2003
- G. Della Penna, B. Intrigila, E. Tronci, and M. Venturini Zilli *Exploiting Transition Locality in the Disk based Mur φ Verifier*, FMCAD, Portland 2002
- E. Tronci, G. Della Penna, B. Intrigila, and M. Venturini Zilli *Exploiting Transition Locality in Automatic Verification*, CHARME, Edinburgh 2001
- <http://www.dsi.uniroma1.it/~tronci/cached.murphi.html>
- <http://vv.cs.byu.edu/mug> (Mur φ users group)