Markov Chains, MDPs and Topics in Probabilistic Verification

by
Adwait Godbole

Guide
Prof. S Akshay

A Study Report for the RnD Course

Computer Science and Engineering
Indian Institute of Technology Bombay
India
04 Apr 2018

Contents

1 Basic Markov Chains

1.1 Introduction

Markov chains can be intuitively thought of as a sequence of random variables where the probability distribution of the $k+1^{th}$ variable in the sequence is only dependent of the k^{th} variable and is independent of the variables prior to that (rewording of the Markov Property). Also this distribution is independent of the value of k. For finite Markov chains the number of possible values that the variables can take (support of the probability distribution) is finite. These values can be labelled with atomic propositions with a goal of model checking and hence can be viewed as states.

1.2 3 views of Markov Chains

1. **Infinite Sequence Form**: Markov chain is expressed as an infinite sequence of variables X_i. $pmf(X_{k+1}|X_k)$ is solely dependent on the value X_k and not on k itself.

2. **Matrix Form**: Markov chain can be expressed as an n-by-n (where n is the possible number of states or values that the variable can take) transition matrix M, where $M(i,j) = P(X_{s+1} = j|X_s = i)$ which can be called the single step transition probability. Multi-step transition probabilities can be easily computed by considering $M^s(i,j)$.

3. **Graph Form**: Graph with nodes (can be viewed as states labelled by atomic propositions) as the values that the Markov variables can take and edges labelled with the transition probabilities. This gives some more intuition along the lines of the state model. $Edge(i,j) = P(X_{k+1} = j|X_k = i)$.

1.3 Notation - A

1. Transition matrix M (is a row stochastic matrix) which gives one step transition probabilities

2. Transition matrix M^n gives n-step transition probabilities

3. $P_i(X_n = j)$ gives probability that starting from i the value attained after n-steps is j. In general sub-scripted probabilities are with the condition that the starting state is i.

4. H_i and T_i are the hitting times and first-passage times respectively.

5. V_i is the number of visits to i and f_i is the return probability for i.

6. m_i is the expected value of first passage time $m_i = E_i(T_i)$.

7. γ_i^k is the expected number of visits to i between successive visits to k.

1.4 Definitions - A

1. Two states communicate ($i \rightarrow j$) if $P_i(X_n = j) > 0$ for some value of n. This is also termed as reachability.

2. We have ($i \leftrightarrow j$) if $(i \rightarrow j) \wedge (j \rightarrow i)$

3. A set of states C form an SCC (strongly connected component and are said to be communicating) if $\forall i, j \in C$ we have ($i \leftrightarrow j$).

4. A set of states C is closed if $\forall i \in C$ we have $(i \leftrightarrow j) \Rightarrow j \in C$. Intuitively this set of states can be said to be absorbing.

5. A set of states S is said to form a BSCC (base-strongly connected component) if it is both strongly connected and closed. A Markov chain with a single BSCC is said to be irreducible.

6. A state is periodic if for some $N > 0$ we have $P_i(X_k = i) > 0 \quad \forall \quad k \geq N$. Alternatively period is the GCD of all k such that $p_{ii}^k > 0$. A state with period greater than 1 is said to be periodic.

7. A state is recurrent if the probability of there being infinitely many number of returns is 1 (and not zero which is the only other value that it can take).

8. A recurrent state is said to be null recurrent if m_i mean recurrence time is infinite else it is positive recurrent.

9. A distribution λ is said to be invariant if we have $\lambda M = \lambda$ along with $\sum_i \lambda_i = 1$.

2 Results and Properties of Markov Chains

2.1 Recurrence

1. **Equivalent conditions for Recurrence** Equivalence of recurrence, certainty of return and summability of n-step transition probabilities. Thus recurrence is equivalent to $f_i = 1$ and $\sum_n p_{ii}^n = \infty$.
   Proof Every new visit to i can be considered as a beginning of a new chain (and hence a further visit occurs with probability f_i), thus the probability of there being ∞-many further visits is 1 iff $f_i = 1$ else 0. This is formalized by the equation $P_i(V_i \geq k+1) = (f_i)^k$. We have $\sum_n p_{ii}^n = E_i(\sum_n 1_{\{X_n=i\}}) = E_i(V_i)$. Thus recurrence implies $\sum_n p_{ii}^n = \infty$. On the other hand if a state is transient, $E_i(V_i) = \sum_r P_i(V_i > r) = \sum_r f_i^r < \infty$.

2. **Recurrence as a class property** For a communicating class, states are either all recurrent or all transient.
   Proof Considering the sum of n-step transition probabilities, we can convert from the sum for a state i to sum for a state j with a constant multiplicative factor (dependent on the probability of reachability ($i \leftrightarrow j$)). Thus if one sum is infinite the other must be so too.

3. **Recurrence and closed classes** Every recurrent class is closed and conversely every finite closed class is recurrent
   Proof $\Rightarrow$ If possible the class is not closed and hence there exists i, j such that ($i \to j$) and $i \in C, j \notin C$. Thus from i if we go to j (with positive probability), the probability of returning, as also the probability of i being recurrent is 0.
   $\Leftarrow$ The probability of $V_i = \infty$ is either 0 or 1 for all i. Also in any run, there is at least one state which must occur infinitely many number of times. Thus the above probability for this state must be 1, and hence this state must be recurrent. Hence, this entire class must be recurrent.

4. **First passage time is $< \infty$ for a recurrent irreducible chain** For any initial distribution we have $P(T_j < \infty) = 1$ for any j for a recurrent and irreducible Markov chain.
   Proof All the states are recurrent, and hence we can show that $P_i(T_j < \infty) = 1$ for all i. Consider a chain starting with j and encountering i along the way ($p_{ji}^n > 0$). The probability that infinitely many j follow this i is 1, and hence we have $P_i(T_j < \infty) = 1$). Now $P(T_j < \infty)$ with a chain with an arbitrary initial distribution can be expressed as sum of probabilities $P_i(T_j < \infty)P(X_0 = i)$ (separating the initial state i) with $P_i(T_j < \infty) = 1$). Thus $P(T_j < \infty)$.

2.2 Invariance

1. **Invariant distribution as limit of n-step transition probabilities** For some i if $lim_{n\to\infty} p_{ij}^n = \pi_j$ then π is an invariant distribution.

Proof Firstly show that π is indeed a distribution (sums to 1). Now apply a Chapman-Kolmogorov $(1, n-1)$ split and utilize the fact that we are in a $n \to \infty$ limit. Thus we have $\pi_j = lim_{n\to\infty} p_{ij}^n = lim_{n\to\infty} p_{ij}^{n-1}$, which proves the statement.

2. **Every Irreducible and Recurrent stochastic matrix has a positive invariant measure** In effect we have to show that γ^k is this required invariant vector.
   Proof We can consider the idea of predecessor state (say i). Now γ_j^k can be split into cases that predecessor is i multiplied by p_{ij} as we know that the next state is going to be j. We note that we can take advantage of the fact that our sum can go from 0 to T_k as these border cases yield zeros. Thus we can 'shift' γ from j to its predecessor. Also we have $\gamma_k^k = 1$ which gives us that $0 < \gamma_i^k < \infty$.

3. **Uniqueness of Invariant measure** γ is the unique invariant measure given a recurrent and irreducible matrix.
   Proof We have been given that λ is invariant along with $\lambda_k = 1$. Thus we can expand λ_j with terms of sum over m of $P_k(X_m = j \quad and \quad T_k \geq m)$. Thus in the limit of m tending to ∞ we approach γ_j^k from below. Now given that P is recurrent we have we have γ_k is invariant and hence their difference is invariant as well. Also k^{th} component of difference must be zero, thus considering this extremal case we show that all must be zero for it to be invariant.

2.3 Periodicity

1. **Periodicity is a class property** For states i and j if $(i \leftrightarrow j)$ then i and j have the same period.
   Proof We can consider a looping argument. We can consider $i \to j \to i$ with looping at j to get a loop from i.

2. **Aperiodicity is equivalent to $p_{ii}^n > 0$ for all sufficiently large n** If period is greater than 1, we have $p_{ii}^n > 0$ only for those n such that period divides n. Converse is proved similarly (G.C.D. of consecutive numbers is 1).

2.4 Convergence

1. **Equivalence of positive Recurrence and Existence of an equilibrium distribution** For an irreducible chain P all states being positive recurrent and it having an invariant distribution are equivalent.
   Proof $\Rightarrow$ Existence of positive recurrent state implies that P is recurrent. Thus P must have an invariant measure γ. Moreover the elements of this invariant measure must be nonzero and finite as $\sum_i \gamma_i^k = m_i$ which is known to be finite and positive.
   $\Leftarrow$ Now given that there is an invariant distribution, we can construct γ^k

by considering $\gamma_i^k = \pi_i/\pi_k$. Now γ^k is lesser or equal to any invariant distribution. Thus, m_k which is sum of γ_i^k is less than ∞. Thus, P is positive recurrent.

2. **Fundamental Theorem of Markov chains** Given an ergodic (positive recurrent, irreducible and aperiodic) Markov chain with invariant distribution π. Then any initial distribution approaches π as $n \to \infty$.
Proof part I Consider the coupling argument with new Markov chain P' (X, Y) where $X = Markov(\lambda, P)$ and $Y = Markov(\pi, P)$. We will show that the first passage time for the state (b, b) for is finite. Now as P is aperiodic and irreducible $p_{ij}^n > 0$ for all n greater than some value (using property that $p_{ij}^m > 0$ for some m and aperiodicity). Hence we can show that P' is aperiodic and irreducible. Thus P' is positive recurrent and hence the first passage time (say T_p) for any state is $< \infty$.
Proof part II We need to show that $P(X_n = i)$ tends to π_i for $n \to \infty$. We consider cases separated by T_p. For $n > T_p$ we have $X = Y$. S the only times when X and Y diverge is for the small (finite) times $n < T_p$. The probability that $n < T_p$ goes to 0 as increases as T_p is finite. Thus $P(X_n = i) \to \pi_i$ as $n \to \infty$.

3 Model Checking formalism for Probabilistic Systems

3.1 Notation - B

1. **Discrete Time Markov Chain** is represented by M is a tuple (S, P, i_{init}, AP, L) of states, transition probabilities, initial distribution, atomic propositions and labels for all the states. P must be stochastic.

2. **Paths** $Paths_{fin}(M)$ and $Paths(M)$ denote finite and infinite paths in the Markov chain. Similarly $Paths_{fin}(s)$ and $Paths(s)$ represent paths starting from state s. $Post(s)$ and $Pre(s)$ represent successor and predecessor states for state s. $Post^*(s)$ and $Pre^*(s)$ sequences of states succeeding and preceding s. A state s is absorbing if $Post(s) = s$.

3. **LTL/CTL-like Notation**

   (a) $\Box$ ... always

   (b) $\Diamond$ ... eventually

   (c) $\bigcup$ ... until

   (d) $\bigcirc$ ... next

   (e) $\models$ ... satisfaction

   (f) $\vee$ ... disjoint union

4. **Cylindrical Set and its σ-algebra** $Cyl(\hat{\pi})$ is set of paths with prefix $\hat{\pi}$. Probability measure of such a cylinder is the product of all the 1-step transition probabilities involved multiplied by the probability of s_0 in the initial distribution. Probability measure of the zero length path is 1.

3.2 Reachability

1. **Elementary Reachability** The problem is to determine the probability of reaching a state from a set B starting from a state s. Thus we need to find $Pr^M(s \models \Diamond B)$
   One method is summing up an infinite sequence of probabilities depending upon the length of the path of the Markov chain.
   The above method is not sufficiently general and hence may not be applicable most scenarios. In this case we take up an inductive approach. We can form cases that we reach B in just one step (so the probability is given by $\sum_{i \in B} P(s, i)$. Else we go to an intermediate state an then consider probability of reaching B from this state. Thus it is useful to think of the probability of reaching from state s as a vector $\mathbf{x}$. We then have $\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{b}$ where $\mathbf{A}$ is the transition probability matrix for states not in B and $\mathbf{b}$ is 1-step reaching probability from all the states. Thus the problem reduces to that of matrix computations.

2. **Constrained Reachability** We now have an additional constraint that the intermediate states must belong to another state-set C. Thus we need to find $Pr^M(s \models C \bigcup B)$.
   Intuitively we split S into 3 subsets depending upon whether this probability is 0 or 1 at first glance or whether we do not know this initially. We aptly call these 3 parts $S_0, S_1, S_?$. We omit rows corresponding to S_0, S_1 from P to get $\mathbf{A}$. Similarly we define $\mathbf{b}$ to be the vector such that $b_s = P(s, S_1)$ for $s \in S_?$.
   To arrive at the probabilities consider $\mathbf{Ay + b}$ as a transformation on $\mathbf{y}$. Also let $\mathbf{x}$ be the required invariant distribution. Also define a sequence of distributions $\mathbf{x}^{(i)}$ after repeated transformations on $\mathbf{x}^{(0)}$ which is the zero vector.
   x is limit of $\mathbf{x}^{(n)}$ Firstly we can show that $\mathbf{x}^{(i)}$ is the constrained reaching probability within i-steps (can be shown by induction). Thus $lim_{n\to\infty}$ $\mathbf{x}^{(n)}$ gives the countable union of constrained reachability within n-steps and gives the probability of reaching with any number of steps which is precisely $\mathbf{x}$
   x is the fixed point of the chain We apply the next state split (sum over possibilities of next state) to $\mathbf{x}_s$. Simplifying next states that belong to S_1 or S_0 we get $\mathbf{Ax + b}$ which shows that $\mathbf{x}$ is invariant.
   x is the least fixed point Using induction we can show that $\mathbf{x}^{(i)}$ is less that any fixed point $\mathbf{y}$. As $\mathbf{x}$ is the limit of these distributions we have $\mathbf{x}$ smaller than $\mathbf{y}$.

   This in fact provides us with a method to approximately compute the distribution $\mathbf{x}$ by iteratively computing the $\mathbf{x}^{(i)}$. S_1 and S_0 should be chosen to be as large as possible in order to reduce linear equations. This can be achieved with a linear time graph searches from B. The requirement of **least** fixed point is made clear with the following example. Considering the event $\Diamond s$, we get infinitely many fixed points in interval [0, 1] for start state t. However the true probabilities for $\Diamond s$ are given by the least fixed point $x_t = 0$. In cases such as this, unless we impose restrictions on the

Figure 1: A simple Markov Chain illustrating requirement of least fixed point characterization and also suggesting that for linear equation soving techniques to work we need constraints on S_0

   set S_0, we do not have a unique fixed point and hence linear algebraic methods cannot be applied.

Uniqueness of Fixed Point If S_0 consists of all states s such that $s \not\models \exists(C \bigcup B)$ then we have a unique fixed point for $\mathbf{Ax + b}$.
Proof We will show that $\mathbf{x} = 0$ is the only solution to $\mathbf{Ax} = \mathbf{x}$. This will imply that there must be a unique fixed point as if there are two we can show that their difference is $\mathbf{0}$. Considering the extremal value from $\mathbf{x}$, $x_{max} > 0$, we can show that for all s such that $|x_s| = x_{max}$ we have $\sum_{t \in S_?} \mathbf{P}(s,t) = 1$ (else the particular equations given $\mathbf{Ax = x}$ for those s turn into inequalities). Also $|x_t| = x_{max}$ for all states in $Post(s) \cap S_?$. Thus we get $Post^*(s) \subseteq S_?$. This in fact implies that $s \not\models \exists(C \bigcup B)$ and thus s should have been a member of S_0 in the first place. Thus we must have $x_{max} = 0$ so that the inequalities become trivial equalities. Hence $\mathbf{x = 0}$ is the only solution to $\mathbf{Ax = x}$.

n-step Transition Probabilities are a special case of constrained reachability where $B = \phi$ and $C = S$. So we have $\mathbf{A}$ the same as $\mathbf{P}$ (the step transition probability matrix itself).
Reachability to a set B **within** n **steps** can now be computed by suitably modifying the Markov Chain by introducing loops at all states in B. Now $\Diamond^{\leq n} B$ for original chain reduces to $\Diamond^n B$ for the new chain.

3.3 Qualitative Properties

1. **Repeated Reachability** We have the notions of $\Box\Diamond B$ (always eventually B). This can be represented using countable intersections of countable unions of cylinder sets such that specifying any value n there exists some $m \geq n$ such that there is a finite path segment of length m that ends in a B state.

$$\Box\Diamond B = \bigcap_{n \geq 0} \bigcup_{m \geq n} Cyl(\text{length m ends in } B)$$

Showing equivalence between these two definitions is easy (as, always eventually B, implies that following any state, B must occur again; exactly what is specified by the intersection of unions form).
This can be extended to measurability of paths having infinite occurrences of finite path fragments as well.

2. **Persistence** We also have the notion of $\Diamond\Box B$ (eventually always B). This is the same as $\neg\Box\Diamond S \setminus B$.

3. **Fairness in path selection** Given a state t such that $\Box\Diamond t$ we have almost surely that the transition $t \to u$ is taken infinitely often.

$$Pr(s \models \Box\Diamond t) = Pr(\bigwedge_{\hat{\pi} \in Paths_{fin}(t)} \Box\Diamond\hat{\pi})$$

Proof I For all $\hat{\pi}$ given a visit to state t we have some (non-zero) probability associated with it. Now the probability of this path never being taken

after n visits to t is $(1-p)^n$. Thus considering the countable intersection the probability reduces to 0.
II Now we can show that the probability of $\Box\Diamond t$ but $\hat{\pi}$ never being taken some moment onward is zero. Now given that the set of finite path fragments is countable, we can extend the above to the set of all paths to finish the proof.

4. **Limit Behaviour of Markov Chains** The set of infinitely occurring states (inf) in any path belongs to a BSCC of the Markov chain.
Proof The set inf($\hat{\pi}$) is strongly connected and thus is contained in some SCC. Now let $T = inf(\hat{\pi})$. By the the theorem on fairness above, we have $Post^*(inf(\hat{\pi}) \subseteq inf(\hat{\pi})$ (almost surely these states are visited infinitely often). Thus $Post^*(T) \subseteq T$, which shows that T is a BSCC.

5. **Almost Sure Reachability** For a set of **absorbing** states B, $Pr(s \models \Diamond B) = 1$, $Post^*(t) \cap B \neq \phi$ for all $t \in Post^*(s)$ and $s \in S \setminus Pre^*(S \setminus Pre^*(B))$ are equivalent.
Proof The key idea is to note that B is absorbing. Thus $(i \rightarrow ii)$ is evident. We get $(ii \leftrightarrow iii)$ by making successive use of $Post$ and Pre notation and set complementation. The third part uses the fact that eventually we end up in a BSCC. As all states in B are absorbing, any BSCC is either a state in B or has empty intersection with B. The latter is not possible as we can reach B from any state in $Post^*(s)$.
(iii) suggests a graph search algorithm to determine all s such that $Pr(s \models \Diamond B) = 1$.

6. **Qualitative Constrained Reachability** We have already looked at the problem of $s \models C \bigcup B$. We made use of a sort of heuristic by way of S_1 and S_0. S_0 can be found out by backward graph search starting from B. For S_1, we consider a modified Markov chain by making all states in B and $S \setminus (C \cup B)$ absorbing. Thus we have the problem from $C \bigcup B$ in the old chain to $\Diamond B$ in the new chain which is our old problem again.

7. **Qualitative Repeated Reachability** We have that $Pr(s \models \Box\Diamond B) = 1$, $T \cap B \neq \phi$ for any BSCC T reachable from s and $s \models \forall\Box\exists\Diamond B$ are equivalent.
Proof $(i \leftrightarrow ii)$ is evident as from any state we almost surely end up in a BSCC.
We can also find quantitative repeated reachability to B from a state s. Consider U as the union of all BSCCs T such that $T \cap B \neq \phi$. Thus the problem reduces to $s \models \Diamond U$.
Note that the above analysis is valid only for *finite* Markov chains as can be illustrated with an example of an infinite random walk.

4 Markov Decision Processes

4.1 Definitions - B

1. A Markov Decision Process is a tuple M where S is a set of states, Act is set of actions, $\mathbf{P}$ is the transition probability function between a state, an action, a next state and a probability value. Also there is a set of initial distributions and and the states are labelled by atomic propositions. There is a requirement that $\sum_{s' \in S} \mathbf{P}(s, \alpha, s') \in 0, 1$.

2. For those α for which the above sum is 1 are termed as enabled actions for state s.

3. Similarly successors and predecessors of a state s as those states r and t such that $\mathbf{P}(r, \alpha, s) > 0$ and $\mathbf{P}(s, \alpha, t) > 0$.

4. Notion of path is a sequence of alternate states and actions starting with a state such that for any consecutive triple of state action state the probability of that triple is greater than 0. Additionally any finite path must end in a state as well.

4.2 Scheduler as a means to remove non-determinism

1. The non-deterministic choice of action selection being made at any state is resolved by a function G such that $G(s_0, s_1, ...s_n) = \alpha$ with the condition that α is an enabled action for state s_n. This function is called a **scheduler**.

2. A scheduler G induces a Markov chain on M. This Markov chain has states corresponding to S^+ and edges given by the scheduler at every state.

3. The paths π generated by the scheduler have traces (words on the alphabet 2^{AP}) $L(s_0)L(s_1)...$ which belong to some ω-regular property on 2^{AP}. Thus a probability can be associated with the satisfaction of this property P. $Pr^G(s \models P) = Pr_s^{M_G}\{\pi \in Paths(s) | trace(\pi) \in P\}$.

4. **Memory-less scheduler** is a scheduler such that the action is dependent only on $last(\pi)$.

5. **Finite memory scheduler** is a scheduler with an internal DFA where the states of the DFA are called modes. These modes function as memory units. The function act takes as argument both current state s and the mode to return next state. The function Δ is a transition function for the mode of the scheduler. The finite memory scheduler can be converted to a general scheduler. Basically the mode is a way of compactly (?and possibly insufficiently representing the history of the scheduler).

4.3 Reachability

4.3.1 Linear Program

1. Given an MDP we wish to compute the reachability probabilities,

$$Pr^{max}(s \models \Diamond B) = sup_G Pr^G(s \models \Diamond B)$$

where $s \in S$ and $B \subseteq S$. We can reduce our search to memory-less schedulers as for a given maximum there exists a memory-less scheduler that achieves it. The equations for $\mathbf{x}$ (the vector of maximum probabilities) are specified as

$$x_s = 1 \text{ for } s \in B$$
$$x_s = 0 \text{ for } s \not\models \exists \Diamond B$$
$$x_s = max\{ \sum_{t \in S} P(s, \alpha, t) x_t | \alpha \in Act(s)\} \text{ for other } s$$

This is in fact similar to the Bellman backup equation in reinforcement learning. The vector $\mathbf{x}$ is in fact a stationary point produced value iteration by the RL agent.

2. **Specification of memory-less scheduler** We will look at only those actions α such that

$$Pr^{max}(s \models \Diamond B) = x_s = \sum_{t \in S} P(s, \alpha, t) x_t$$

Note that any such scheduler may not work (e.g. for a state $s \notin B$ with $P(s, \alpha, s) = 1$ non-reachability is assured). Thus we also need to impose the condition of reachability. This is done by selecting only actions α, such that for some t, $P(s, \alpha, t) > 0$ and $||t|| = ||s|| - 1$, where $||s||$ denotes shortest path distance from s to B. This along with the above condition of max-action guarantees a correct scheduler. For states $s \in B$ and $s \not\models \exists \Diamond B$ we select an arbitrary action.

3. **Linear time program** We can specify a linear system to compute the max reachability probabilities $\mathbf{x}$. Firstly we set x_s to 1 and 0 respectively for those s such that $s \in B$ and $s \not\models \exists \Diamond B$. For other s we specify the equation, for all α

$$x_s \geq \sum_{t \in S} P(s, \alpha, t) x_t$$

and with the condition that $\sum_{s \in S} x_s$ is minimum.
We can show that $\mathbf{x}$ is indeed a solution to this system. Also for a solution $\mathbf{y}$ to the system under the minimality condition it satisfies the max reachability equations (else we could apply value iteration and get a smaller limiting $\mathbf{y}$' which would equal $\mathbf{x}$ and hence satisfy the system contradicting the assumption that $\mathbf{y}$ was the minimal solution.

4.3.2 Qualitative Reachability

1. We consider finding s such that $Pr^{max}(s \models \Diamond B) = 1$. One straightforward approach is to consider memory-less schedulers and their induced Markov chains. Given such a Markov chain, we can make states in B absorbing and apply result 3.3.5 *Almost Sure Reachability* to find all s. But the space of all schedulers being exponential we need to find alternative methods.

2. **Graph algorithm for possibility of sure reachability in MDP** We work iteratively on the graph and do the following at every iteration:

   (a) Find $U = \{s \not\models \exists \Diamond B | s \in S\}$.

   (b) For all u in U remove all edges towards u and the corresponding actions in $Pre(u)$.

   (c) Remove states t in $Pre(u)$ if $Act(t) = \phi$ (no more actions are possible). Add t to U. (Note that we need to keep a tab of states u which have already been analysed before. This can be done by introducing a new set of states or more simply by maintaining a boolean vector.)

   (d) Return new MDP on $S \setminus U$ and recompute U for it. Stop if U is empty and then return all states in last MDP.

   This algorithm is correct due to the truths that for all states t introduced in U we have $t \not\models \exists \Diamond B$; and that for all actions α in $Pre(u)$ there can't be a scheduler G that selects α and gives almost sure reachability for the corresponding state t and hence this action might as well be removed from $Act(t)$. The time complexity is quadratic in size of M.

4.3.3 Min Reachability

1. We can construct a set of equations similar to the max-reachability case for min-reachability. Though in this case we need the additional constraints:

$$x_s = 1 \text{ for } s \in B$$
$$x_s = 0 \text{ for } Pr^{min}(s \models \Diamond B) = 0$$
$$x_s = min\{\sum_{t \in S} P(s, \alpha, t) x_t | \alpha \in Act(s)\} \text{ for other } s$$

   Thus once again we can use value iteration for computing s for the third category. In fact the sequence of $\mathbf{x}$ obtained by value iteration (using third equation) are the n-step min-reaching probabilities (over all schedulers), $min(Pr^G(s \models \Diamond^{\leq n} B))$. Note that x_s for s in the first and second categories remain 1 and 0 respectively. These probabilities can be achieved by using finite memory schedulers with n modes (specifying how many of the n steps they have taken).

2. **Possibility of surely not reaching** By a graph algorithm we can determine $S^{min}_{=0} = \{s \in S | Pr^{min}(s \models \Diamond B) = 0\}$. We start from B and backtrack to find states s for which for all action choices $P(s, \alpha, B) > 0$. This

can be done by maintaining a current set $T = \{s \in S | Pr^{min}(s \models \Diamond B) > 0\}$, and add to states s such that $\forall \alpha\ P(s, \alpha, T) > 0$. Now, $S^{min}_{=0} = S \setminus T$ as for any state s in $S \setminus T$ we can select G such that $Post(s, G(s)) \cap T$. The algorithm is similar to 4.3.2.2 above.

3. **Almost surely reaching** We also have almost surely reachability similar to 3.3.5 but for Markov chains induced by all possible schedulers; $S^{min}_{=1} = \{s \in S | Pr^{min}(s \models \Diamond B) = 1\}$. We can show that:

   (a) $P^{min}(s \models \Diamond B) < 1$

   (b) $Pr^{G}(s \models \Box \neg B) > 0$ for some scheduler G

   (c) $s \models \exists((\neg B) \bigcup t)$ with $Pr^{G}(t \models \Box \neg B) = 1$ for some memory-less scheduler G.

   We can show (1) $\rightarrow$ (2) easily. For (2) $\rightarrow$ (3) we have the fact that the Markov chain corresponding to G ends up in a BSCC (without passing through B) C such that $C \cap B = \phi$. Hence t can be any of the states in C. For (3) $\rightarrow$ (1) we have some path (in the Markov chain of G) up to t which occurs with a positive probability. Once C is reached the probability of reaching B is zero. Hence for this scheduler the probability of not reaching B is positive (and equal to taking the above mentioned path). Hence the minimum probability must be less than 1.
   Associated graph algorithm:

   (a) Start with a set T of all states in graph.

   (b) For $s \in B$ set $Act(s)$ to ϕ. Sequentially remove these states from T and remove edges towards them.

   (c) If some state in T is left with no edges due to above operation, apply that operation on it as well.

   (d) Repeat till there are states with $Act(s) = \phi$.

   We can contrast this with the algorithm in 4.2.3. In general whenever a max is to be taken, we have the strategy of dropping edges which will surely *not* lead to the maximum from the graph. Once all edges are gone, the state itself cannot be a part of the solution and hence can be dropped. Then we have reduced the size of the problem and can recur further.

4. Similar to the maximum case, we have a memoryless scheduler on the basis of a linear program.

5. **Constrained Reachability** For the constrained reachability problem $C \bigcup B$ or the extension $C \bigcup^{\leq n} B$ we can simply delete (make absorbing) the states $S \setminus (C \cup B)$ and on the remaining states apply the previous reachability results.

5 PCTL

5.1 Introduction

PCTL is based on CTL but allows for probabilistic operators to generate predicates in addition to the universally $\forall$ and existentially $\exists$ operators in CTL. State formulae may be generated using the grammar:

$$\Phi := true|a|\Phi_1 \wedge \Phi_2|\neg\Psi|P_J(\phi)$$

Where, ϕ is a path formula. The last rule states the the probability of satisfaction of ϕ lies in the interval $J \subseteq (0, 1]$. The path-formulae syntax is very similar to CTL:

1. $\bigcirc$ ... next state
2. $\bigcup$ ... until (unbounded)
3. $\bigcup^{\leq n}$ ... until (bounded by n steps)

The bounded until essentially means that the path is at most n steps in length. Based in these operators we can derive other operators such as:

1. $\Diamond$ ... true $\bigcup$
2. $\Box$ ... using the duality of eventually and always

We can define a sigma-algebra (using cylindrical sets) on the Markov Chain similar to the way that was discussed in the section on Markov chains. We can prove the measurability of PCTL events by recursively breaking down the formula into its sub-formulae. For $\bigcirc\Phi$ we need to consider the cylindrical sets $Cyl(st)$ such that $t \models \Phi$. For bounded until $\Phi \bigcup^{\leq n} \Psi$, we need to consider cylindrical sets of length atmost n starting at s such that Φ is satisfied at all states until Ψ is satisfied. Unbounded until is simply the union of all the satisfying cylindrical sets seen above as n ranges from 0 to ∞.

5.2 PCTL Model Checking

Given a formula Φ and a state $s \in M$, we need to verify whether $s \models \Phi$. When Φ is purely a propositional formula (this reduces to the CTL case), we traverse the parse tree of Φ and compute $Sat(\phi)$ for each subformula of Φ. Now we combine these Sat sets usig the corresponding propositional operators:

1. For $\Phi \wedge \Psi$ we take $Sat(\Phi) \bigcap Sat(\Psi)$
2. For $\neg\Phi$ we take the complement $(Sat(\Phi))'$

Other secondary operators can also be evaluated similarly. But for PCTL we also need to consider cases when we have the operators P_J as well as the PCTL path operators.

1. For P_J we compute $Pr(s \models \phi)$ and check whether it belongs to the interval specified by J

2. For $\bigcirc\Phi$ we generate the bit vector $\mathbf{b}$ corresponding to $Sat(\Phi)$ for all the states in M. Now we just compute the matrix multiplication of P (the transition-probability matrix of M with $\mathbf{b}$.

3. For $\Phi \bigcup \Psi$. The bounded and unbounded until operators are resolved similar to the constrained reachability for Markov Chains. Given the constrained reachability $C \bigcup B$ we set $C = Sat(\Phi)$ and $B = Sat(\Psi)$.

As all these operations are at most polynomial in the size of M we have a polynomial time algorithm to check the PCTL formula.

We have discussed above that any PCTL formula can be checked in time polynomial in the size of the Markov chain M. It is however desirable to optimize this in case of some special formulae. For these we intend to use lesser complexity graph-based techniques.

5.3 Qualitative Fragment of PCTL

We explore the qualitative fragment of PCTL which imposes certain restrictions on operators that can be used in formulae.

1. We disallow the use of general probability operator P_J and instead restrict to $P_{>0}$ and $P_{=1}$

2. For path formulae we restrict to only $\bigcirc$ and unbounded until $\bigcup$ operators

We note that the other bounds at 0 and 1 can be derived using appropriate negations of the above two.

5.3.1 Comparison of Qualitative PCTL and CTL

We consider a Markov Chain M for the qualitative PCTL formulae. For the CTL formulae we consider the induced transition system, i.e. $(s, s') \in R$ iff $P(s, s') > 0$ is M where, R is the transition relation for the CTL and P is the transition probability matrix for the qualitative PCTL part. A CTL formula Ψ satisfied in this 'induced' transition system is denoted by $Sat_{TS(M)}(\Psi)$. We now define the equivalence of a formula in qualitative PCTL and CTL. A formula Φ in qualitative PCTL and a formula Ψ is CTL are equivalent if $Sat_M(\Phi) \equiv Sat_{TS(M)}(\Psi)$.

We now compare formulae in Qualitative PTCL with similar formulae in CTL

1. PCTL formula $P_{=1}(\phi)$ and CTL formula $\forall\phi$. We can see the the CTL formula is stronger than the former, as we need strictly all paths for the CTL formula while for the PCTL case we need almost sure satisfaction of ϕ.

2. In some special cases of ϕ however, we have equivalence of the above two formulae. In the case of $\phi = \bigcirc a$, we reason about paths of length 1. This occurs as there is no decay of probability as in the case with infinitely long paths.

3. Similarly we have the PCTL formula $P_{>0}(\phi)$ and the CTL formula $\exists\phi$. These are also generally not equal. In the special case of $\phi = \Diamond a$ however they become equal as we naturally need a finite length path which implies a non-zero probability. This reinforces the idea that whenever there is no decay to 0 due to infinitely long paths, we have equivalence.

4. For the PCTL formulae $P_{=1}(\Diamond a)$ and $P_{>0}(\Box a)$ however there is no CTL analog.

6 Safety Problems for MDPs

6.1 Definitions and Problem Introduction

1. An MDP (or PFA) as discussed in the earlier section is a tuple (S, $\sum$, M_α). Here S is the set of states, $\sum$ is the alphabet of actions and M_α is the stochastic matrix that determines the weight redistribution on selection of action α.

2. A Polytope H is over probabiliy distributions over the states of the MDP/PFA. Note that this definition us different from the one in the previous section.

3. The main difference in the one step strategies followed in an MDP and in a PFA is the following:

$$\text{In an MDP } M_\tau = \sum \tau(\alpha, s_i) M_{(\alpha,i)}$$

$$\text{In a PFA } M_\tau = \sum \tau(\alpha) M_{(\alpha)}$$

$$\text{Here } M_{(\alpha,i)} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ M_\alpha(i,1) & M_\alpha(i,2) & M_\alpha(i,3) & \dots & M_\alpha(i,n) \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

4. Thus row independence is available for choosing strategy in MDPs as information about state is available. On the other hand in PFAs this information is not available and thus in PFAs there is no row independence. It is this flexibility that that causes many techniques to work for MDPs but not for PFAs.

5. H is the polytope and H_{win} is the subset of the polytope such that $\forall \Delta \in H_{win}$ we have $\exists \sigma \Delta_m^\sigma \in H$. Here Δ_m^σ is the distribution obtained after following strategy σ from Δ for m steps.

6. The above definition also implies that for all distributions in H_{win} there is a an action that gives us distribution which is still in H_{win}.

7. **Existential Safety** To find whether

$$\exists \Delta \in H \text{ s.t. } \exists \sigma \forall m \in \mathbb{N} \text{ we have } \Delta_m^\sigma \in H$$

   This is equivalent to showing that $H_{win} \neq \phi$.

8. **Universal Safety** To find whether

$$\forall \Delta \in H \exists \sigma \text{ s.t. } \forall m \in \mathbb{N} \text{ we have } \Delta_m^\sigma \in H$$

   This is equivalent to showing that $H_{win} = H$.

6.2 Existential Safety for MDPs

1. We begin by showing that H_{win} is is convex:
   (a) Let x and y be the distributions in H such that there exist one step strategies from Let λ_x and $1 - \lambda_x$ be the weights assigned to x and y.
   (b) Similarly let M_x and M_y be the one step strategies followed at x and y respectively.
   (c) As we have row independence while taking linear combination we can construct M_z by taking the linear combination of rows of M_x and M_y (as these rows themselves are linear combinations of the "basis" rows of M_α's.
   (d) We assign the weights for M_x and M_y as $\lambda_x \frac{x_i}{z_i}$ and $(1-\lambda_x)\frac{y_i}{z_i}$ for the i^{th} row of the matrices.
   (e) We thus conclude that set the of distributions in M with a one step strategy is convex.
   (f) Using induction this is immediately extended to the result that the set of distributions from which there exist $H - safe$ strategies is convex. As this set is precisely H_{win} we have that H_{win} is convex.
   (g) As an aside we note that the above results are not true for PFAs as we don't have row independence.

2. **Kakutani's Fixed Point Theorem** Let S be a non-empty and compact and convex subset of some Euclidean space R^n. Let $f : S \to 2^S$ be an upper hemicontinuous set-valued function on S with the property that $f(x)$ is non-empty, convex and closed for all $x \in S$. Then f has a fixed point, i.e. a point x such that $x \in f(x)$. We first will see how this theorem can be applied in this context:
   (a) Our subset S is simply H_{win}.
   (b) It is easy to check that H_{win} satisfies the requirements of non-empty, convex and compact (as H is closed and bounded, H_{win} also must be close dand bounded).
   (c) Our function f is $f : H_{win} \to 2^{H_{win}}$ which is $f(\Delta) = \{\Delta' \in H_{win} | \Delta' = \Delta \cdot M_\tau$ for some one step strategy $\tau\}$.
   (d) Again we can see that f is non-empty (as there does exist atleast one action).
   (e) f is hemi-continuous - whenever f represents the same τ for an interval f must be continuous due to linearity.

   Thus now we are in a position to use this theorem for our purpose:
   (a) We need to choose an appropriate function in Kakutani's fixed point theorem.

(b) Let X the set of initial distributions Δ such that there exists a one-step strategy for which Δ is a fixed point.

(c) Now we immediately have that $X \subseteq H_{win}$. Also by Kakutani's FPT we have that

$$X \neq \phi \iff H_{win} \neq \phi$$

(d) We have reduced the goal to showing that the set of fixed point distributions in H is non-empty.

3. To conclude we will reduce the problem to a set of linear inequalities.

(a) Let Im_i be the set of sub-distributions obtained by following some strategy from state i (weighted by some $\lambda \in [0, 1]$).

(b) We also define t_i^j as the distribution obtained by following strategy j from state i. It is easy to see that for $\delta \in Im_i$ we have $\delta = \sum \mu^j t_i^j$ where $\sum \mu^j \leq 1$.

(c) This is extended to accommodate transitions from all states by constructing a strategy for a fixed point distribution. Let $\Delta = \sum_i \lambda_i s_i$ be a fixed point distribution. We have a one-step strategy τ s.t. $\Delta = \Delta M_\tau$. The elements and constraints of this strategy are:

i. μ_i^j the weight given to action j while in state i (row i of M_{α_j}).

ii. $\sum_j \mu_i^j = \lambda_i$ (as we have weight λ_i in state s_i).

iii. The distribution generated is $\sum_{i,j} \mu_i^j t_i^j$. This must agree with initial distribution for all states.

iv. Thus we have the condition $\sum_{i,j} \mu_i^j t_i^j = \sum_i \lambda_i s_i$.

(d) Whenever the above conditions are satisfied it is easy to see that Δ is a fixed point distribution. Conversely, when we have a fixed point distribution μ_i^j are simply the weights assigned in this strategy.

(e) Hence we can conclude that $X \neq \phi$ reduces to solving a set of linear equalities (in PTIME).

6.3 Universal Safety for MDPs

We want to show that the universal safety problem for MDPs is decidable, moreover it is co-NP-complete. We start off by showing that this problem has a co-NP upper bound. Then we will show that indeed this problem is co-NP-complete by reducing the complement of 3-CNFSAT to it.

6.3.1 Showing the co-NP upper bound

We want to show that the problem has a co-NP upper bound. As with the previous section we consider the decomposition of M_τ with μ_i^j. Thus we translate the universal safety condition into a set of universally quantified linear inequalities.

This falls into the class of QLIs (quantified linear implications) with a single alternation between universally and existentially quantified variables. This has been shown to be in co-NP.

6.3.2 Showing the co-NP-complete lower bound

This is shown using a reduction from the complement of the 3-CNFSAT problem. The plan is to encode variables in the instance of 3-CNFSAT into gadgets with states in the corresponding MDP. Constraints in the 3-CNFSAT are represented by constraints in the polytope H. Some nice ideas used in this reduction are:

1. Use of (loopback) states in the MDP to represent the choice of values assigned to variable literals. These function as the memory.

2. A boolean constraint can be mapped down to an algebraic inequality. Suppose for example, a variable x takes on one of two values 0 or m. Suppose that when $x = 1$, another variable y must be l else it must be 0. This can be specified using the algebraic constraint, $lx - my \in [0, ml]$. This algebraic constraint is easily specified using the polytope H.

3. For each clause, we encode the fact that none of its literals take the true value by a state encoding.

4. Inclusion of a stable state with a constant mass which is used as a control for the mass additions to the other states.

5. Inclusion of a trash state, which takes in all the remaining probability mass.

Note that in the above reduction we set up states for encoding predicates. The strategy satisfies a set of constraints which are equivalent to the not 3-CNFSAT instance. The fact that only this strategy is taken up is achieved by the polytope H.

7 Probabilistic Programs

7.1 Dijkstra's Guarded Command Language

We start by considering Dijkstra's Guarded Command Language, used for reasoning about sequential programs. The constructs and semantics of GCL are:

1. `skip` ... which is essentially to skip the current command
2. `abort` ... an equivalent formulation for the command `while(True): skip`
3. `x := E` ... variable assignment
4. `prog1 ; prog2` ... sequential execution of two commands/programs
5. `if(G) prog1 else prog2` ... conditional evaluation based on guard G
6. `prog1 [] prog2` ... non-deterministic choice between two programs
7. `while(G)prog` ... while loop

These programs are reasoned about by using what are known as **weakest preconditions**. The weakest precondition WP(P, F) is defined as the minimal set of predicates that must be true at the start of the program such that the program always terminates with F as `true`. These conditions are loosened in what are known as weakest *liberal* preconditions. The WLP(P, F) specifies the set of minimal preconditions necessary such that either the program terminates with F as `true` or it doesn't terminate.

The weakest preconditions are determined by starting with the required conditions at termination F and backtracking through the individual statements in the program. The backtracking rules (also termed as predicate transformer semantics) are as follows:

1. `skip` $\Rightarrow$ F
2. `abort` $\Rightarrow$ `false`
3. `x := E` $\Rightarrow$ F[`x := E`] ... F with the variable valuation of `x` as `E`
4. `prog1 ; prog2` $\Rightarrow$ WP(PROG1,WP(PROG2,F)) ... the WP for `prog2` must match the predicate(s) which are `true` at the end of `prog1`
5. `if (G) prog1 else prog2` $\Rightarrow (\mathrm{G} \wedge \mathrm{WP}(\mathrm{PROG1},\mathrm{F})) \vee (\neg\, \mathrm{G} \wedge \mathrm{WP}(\mathrm{PROG2},\mathrm{F}))$
6. `prog1 [] prog2` $\Rightarrow \mathrm{WP}(\mathrm{PROG1},\mathrm{F}) \wedge \mathrm{WP}(\mathrm{PROG2},\mathrm{F})$
7. `while(G)prog` $\Rightarrow \mu X.((G \wedge \mathrm{WP}(\mathrm{PROG}, \mathrm{X})) \vee (\neg G \wedge F))$ where μ is the least fixed point operate

Thus by backtracking by repeatedly using the above rules we end up with the weakest precondition at the start of the program. We now expand this language to include probabilistic commands as well.

7.2 Probabilistic GCL

We add the following two commands to the language specified above:

1. `prog1 [p] prog2` ... a probabilistic choice between two commands
2. `observe(G)` ... conditions the paths that are accepted

NOTE: Apart from the `observe` acting similar to an assert statement like in imperative programming language it also normalizes the probability distribution along the accepted paths.

As we now consider probabilities, it does not suffice to just consider qualitative properties as in the case of Dijkstra's GCL. Hence, we now consider reachability properties to reason about programs in Probabilistic GCL by way of **weakest pre-expectations**. An expectation maps each program state (set of commands and variable evaluation) to reals. A weakest pre-expectation can be viewed a transformer of expectations from one program state to another. This can be viewed as rewards in the state of the MDP. Note that only terminating runs are assigned a reward corresponding to the variable valuation. Non-terminating runs are assigned a reward of 0. WP(P,F) gives us the least expectation at the initial state of program P that ensures that P terminates with expectation F. Similar to the case with weakest pre-conditions we can generate backtracking rules:

1. `skip` $\rightarrow$ F
2. `abort` $\rightarrow$ `0`
3. `x := E` $\rightarrow$ F[`x := E`] ... F with the variable valuation of `x` as `E`
4. `observe(G)` $\rightarrow$ [G]$\cdot$F
5. `prog1 ; prog2` $\rightarrow$ WP(PROG1,WP(PROG2,F)) ... the WP for `prog2` must match the expectations at the end of `prog1`
6. `if (G) prog1 else prog2` $\rightarrow$ [G]$\cdot$WP(PROG1, F) + [$\neg$G]$\cdot$WP(PROG2, F)
7. `prog1 [p] prog2` $\rightarrow$ $p\cdot$WP(PROG1,F) + $(1-p)\cdot$WP(PROG2,F)
8. `while(G)prog` $\rightarrow$ μX([G]$\cdot$WP(PROG,F) + [$\neg$G]$\cdot$F)

We illustrate with an example. Consider the following program. Note that there is no conditioning:

- ```
  x := 0 [2/3] x := 1 ;
  if [x = 1] (y := 1 [2/3] y := 0) else (y := 0 [2/3] y := 1) ;
  ```

Consider the expectation [x = y] (assign a reward of 1 for termination with [x = y]). We will backtrack with WP(P,[X=Y]).

$$\begin{aligned}
wp(P,[x=y]) &= wp(c1, wp(c2,[x=y])) \\
&= wp(c1,[x=1]\cdot wp(c3,[x=y]) + [x=0]\cdot wp(c4,[x=y])) \\
&= wp(c1,[x=1]\cdot(2/3\cdot[x=1]+1/3\cdot[x=0]) + [x=0]\cdot(2/3\cdot[x=0]+1/3\cdot[x=1])) \\
&= wp(c1,(2/3\cdot[x=1]) + (2/3\cdot[x=0])) \\
&= 2/3 wp(x:=0, 2/3\cdot[x=1]+2/3\cdot[x=0]) + 1/3 wp(x:=1, 2/3\cdot[x=1]+2/3\cdot[x=0]) \\
&= 2/3
\end{aligned}$$

We have skipped a couple of steps in between ($[x=0]\cdot[x=1]$ reduces to zero).

Similar to this we define a notion of weakest liberal precondition as the least expectation for P such that the program either does not terminate or if it does, establishes f. In the cases where there is no conditioning the above backtracking is adequate. In cases of conditioning however the `observe` statement normalizes the probability distribution for the runs that pass the condition. Hence in our expectation too, we need to do the same. We thus need to divide WP(P, F) by the probability that a run passes all the conditioning. This is simply given by WLP(P, 1) (assign a reward of 1 for each successful or non-terminating run). Thus for a probabilistic program with conditioning we are infact interested in two quantities, $wp(P,f)$ and $wlp(P,1)$. We define the conditional weakest pre-expectation CWP(P, F) as the the ratio of these two quantities. Let us consider another example, this time having an `observe` conditioning statement:

- ```
  x := 0 [1/2] x := 1 ;
  y := 0 [1/2] y := -1 ;
  observe(x + y = 0);
  ```

We want to find the expected value of x for successful (runs passing the conditioning) runs of the program. For the first part we get:

$$\begin{aligned}
wp(P,x) &= wp(P,[x+y=0]\cdot x) \\
&= 1/2 wp(P; y:=0, [x+y=0]\cdot x) + 1/2 wp(P; y:=-1, [x+y=0]\cdot x) \\
&= 1/2 wp(P,[x=0]\cdot x) + 1/2 wp(P,[x=1]\cdot x) \\
&= 1/2 wp(P,[x=1]\cdot x) \\
&= 1/4
\end{aligned}$$

For the second part we get:

$$\begin{aligned} wlp(P,1) &= wlp(P,[x+y=0]\cdot 1) \\ &= 1/2 wlp(P; y=0, [x+y=0]) + 1/2 wlp(P; y=-1, [x+y=0]) \\ &= 1/2 wlp(P,[x=0]) + 1/2 wlp(P,[x=1]) \\ &= 1/2 \cdot 1/2 + 1/2 \cdot 1/2 \\ &= 1/2 \end{aligned}$$

Hence we conclude with $cwp(P,x) = 1/2$.

7.3 Semantic Intricacies

We will discuss some issues caused due to non-determinism along with conditioning using the observe statement. In these cases we will bring up the notion of a demonic scheduler which will represent the worst case behaviour possible for our program.

1. Firstly we state the difference between certainly terminating and almost surely terminating program. A **certainly** terminating program is one which has no non-terminating paths. An **almost surely** terminating program is one in which the probability of termination is one (but there may exist a non-zero number of runs/paths which do not terminate).

   - `repeat (x := 0) until (x != 0)` ... this is a certainly terminating program
   - `repeat (x := 0 [1/7] x := 1) until (x != 1)` ... this is an almost surely terminating program as there is one run (where x is always 0) which doesn't terminate, though the probability of this run is 0, and hence probability of termination is 1. We do not distinguish between the convergence rates of two almost-surely terminating programs.

2. We have a non-deterministic uncertainty. This must be resolved by a suitable scheduler if we want to reason about probabilities. This is shown by the below example.

   - `repeat (x := 0 [] x := 1) until (x != 0)`

3. Now we consider cases of conditioning using the observe statement.

   - `repeat (x := 1 [1/2] x := 0) observe(x = 1) until (x != 0)`

   We block runs that assign $x = 0$. The probability of the runs that assign $x = 1$ tends to 0, but these runs are normalized w.r.t. probability by the `observe` statement. Thus we get a $(\frac{0}{0})$ undefined value.

4. We can also consider a case where there is non-determinism between certain non-termination and undefined behaviour.

7.4 Operational Semantics - MDP

We translate the probabilistic program into a MDP. Each state in the MDP encodes a pair of the program to be evaluated (P) and the current variable valuation (σ). Thus each state is represented as $\langle P, \sigma \rangle$.

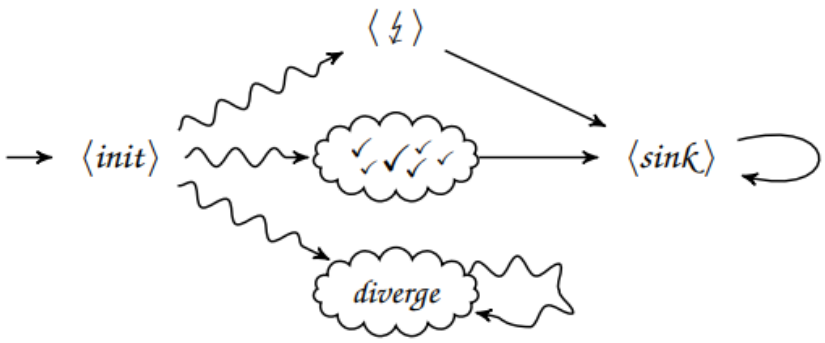

Figure 2: Characteristic MDP for Probabilistic Programs

The above image shows the general form of the MDP generated by a probabilistic program. For a probabilistic program P and set of variable valuations σ we represent this MDP by R^P_σ. The squiggly arrows represent possibly multiple paths over multiple states. It essentially has 5 types of noteworthy states:

1. *init* state which initializes all variables
2. *sink* state which indicates termination of the program
3. *diverge* type states from where there is no path that reaches a *sink* state
4. ✓ states from where we have guaranteed termination and hence these states have the *sink* state as their successor
5. *danger* type state where one or more conditioning statements have been violated

Reasoning about programs is equivalent to reachability or constrained reachability analysis on the MDP. Essentially we want to compute $E^{R^P_\sigma}(\Diamond S)$ the expectation of reaching a state belonging to the set S from the *init* by running the probabilistic program P. In the presence of `observe` commands, we refer to constrained reachability. We can define constrained reachability by $E^{R^P_\sigma}(\Diamond S | \neg\Diamond U)$ as the expectation of reaching states in S given that we do not pass through states in U. Note however that the above expectations only make sense once we have defined a scheduler to resolve all cases of non-determinism. A pessimistic viewpoint is to consider the probability distribution given by a demonic scheduler which was defined earlier.

For $wlp(P,1) > 0$ the $cwp(P,f)$ value obtained by backtracking in the previous section is equivalent to the constrained reachability on the MDP. This idea is formalized by the following:

1. We want to compute on the operational MDP the probability of reaching the *sink* state from the *init* state without passing through any of the *danger* states (these correspond to blocked runs of the program by the `observe` statements).

2. In the case of non-deterministic programs we will refer to the maximum and minimum expectations which arise from demonic and optimistic schedulers for the MDPs.

3. Thus essentially we want to compute $Exp^{R^f(P)}(\Diamond\langle sink\rangle|\neg\Diamond\langle danger\rangle)$, where $R^f(P)$ is the MDP representing the program. Also including a scheduler to resolve non-determinism, we convert our MDP to a Markov Chain. Let this Markov chain be $M^f(P)^G$, with G as the scheduler.

4. Thus we are left with $ExpR^{R^f(P)}(\Diamond\langle sink\rangle|\neg\Diamond\langle danger\rangle)$ which we can rewrite as follows:

$$\begin{aligned} & ExpR^{M^f(P)^G}(\Diamond\langle sink\rangle|\neg\Diamond\langle danger\rangle) \\ &= \frac{ExpR^{M^f(P)^G}(\Diamond\langle sink\rangle|\neg\Diamond\langle danger\rangle)}{Pr^{M^f(P)^G}(\neg\Diamond\langle danger\rangle)} \\ &= \frac{ExpR^{M^f(P)^G}(\Diamond\langle sink\rangle|\neg\Diamond\langle danger\rangle)}{1 - Pr^{M^f(P)}(\Diamond\langle danger\rangle)} \end{aligned}$$

   This is true as the paths that pass through *danger* states eventually are a subset of paths that reach the *sink* state.

7.5 Bounded Model Checking

We take up the discuss on the conversion of weakest pre-expectation for a probabilistic program P to a constrained reachability problem on the MDP corresponding to P. The problem with this approach however is that the MDP so generated may not be bounded in size. As an example let us take up the following case of probabilistic program:

- ```
  i := 1; flag := true;
  while(flag):
     flag := false [1/2] flag := true;
     i := i + 1;
  ```

Here though the program statements in P are finite, due to possibly infinite variable valuations for `i`, we get an infinite MDP corresponding to this program.
  ```

The above discussion motivates a need for developing a method to perform bounded model checking of a probabilistic program using MDPs. This requires a only a finitely large unrolling even if the actual MDP corresponding to the program is infinitely large. We refer to the equation in the last section for this purpose.

7.5.1 Partial Operational Semantics

We will formally define a notion of Partial Operational Semantics for a probabilistic program P. Let the (complete) MDP corresponding to P be $M^f(P)$. A partial operational semantics of P is a sub-MDP $M'^f(P)$ of $M^f(P)$ such that:

1. $S' \subseteq S$ .. where S and S' are state spaces of the MDPs M and M'
2. The set of actions is identical to that of M
3. The transition probalities for actions out of state s are defined to be:
   (a) $P'(s, \alpha, s') = 1$ if in the original MDP M, there was an action from s to a state not in S' (leaking out of current MDP).
   (b) $P'(s, \alpha, s') = P(s, \alpha, s')$ otherwise
4. Thus essentially we self-loop all 'expandable' states in S'.

This conversion to a subset MDP aids us in making a bounded modle check.

We now note that using our notion of partial operational semantics and sub-MDPs, all strict lower bounds for $ExpR^{M^f(P)^G}(\Diamond\langle sink\rangle)$ are computably enumerable. Essentially what this means is we can sequentially unroll the program (that is its MDP) for k steps with an increasing value of k. As the expectation value is monotonic we get a non-decreasing sequence of approximations to $ExpR^{M^f(P)^G}(\Diamond\langle sink\rangle)$. Thus given a query of the form, $ExpR^{M^f(P)^G}_{\leq\lambda}(\Diamond\langle sink\rangle)$, we can unroll the MDP until the value computed is greater than λ. This is formalized by the following statement:

$$M^f(P)' \not\models \phi \implies M^f(P) \not\models \phi \text{ ... where } \phi \text{ is of the kind } E_{\leq\lambda}(\Diamond S|\neg\Diamond U)$$

Dually, this will also work for satisfaction of lower bound properties.

Note that in the previous section with constrained reachability with $\frac{ExpR^{M^f(P)^G}(\Diamond\langle sink\rangle|\neg\Diamond\langle danger\rangle)}{1-Pr^{M^f(P)}(\Diamond\langle danger\rangle)}$, the numerator is monotonically increasing and the denominator is monotonically decreasing. Hence the fraction is monotonically increasing and the sequential finite unrolling technique will work.

Note that the above approach worked as we wanted to show the dis-satisfaction of ϕ. In case we wanted to show the satisfaction of ϕ it won't work as a finite unrolling is never conclusive. As the conditional expected reward will monotonically increase, $M^f(P)' \models \phi \implies P \models \phi$ only if there are no more expandable states. This is true even for checking the violation of lower bound properties.