Monte Carlo w model checkingu

Sławomir Rudnicki

Niezawodność systemów współbieżnych i obiektowych

28 kwietnia 2010

Cel

Weryfikacja systemów komputerowych

- sprawdzenie, czy formuła LTL zachodzi dla danego systemu

Problemy:

- Problem weryfikacji LTL jest PSPACE-zupełny
- Eksplozja stanów
  - nie dysponujemy zasobami wystarczającymi do sprawdzenia całego systemu przez proste wyliczenie stanów.

Techniki i heurystyki

- weryfikacja symboliczna (SMC)
- weryfikacja z ograniczoną głębokością (BMC)
- redukcje częściowo-porządkowe

Monte Carlo

Sprawdzenie własności systemu bez wyliczania całego modelu.

- losowe próbkowanie ścieżek w celu znalezienia błędu
  - symulacja losowego wykonania programu
- ograniczenie prawdopodobieństwa wystąpienia błędu

Wprowadzenie

Dane:

- automat Büchiego B_S reprezentujący system
- formuła LTL φ:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi U \varphi \mid X\varphi$$

Automat Büchiego – przypomnienie

$$B = (\Sigma, Q, Q_0, \delta, F)$$

- Σ – alfabet wejściowy
- Q – skończony zbiór stanów
- Q_0 – stan początkowy
- $\delta \subset Q \times \Sigma \times Q$ – relacja przejścia
- F – stany akceptujące

W skrócie:

- skończony automat dla nieskończonych słów
- akceptuje język ω-regularny

Podejście tradycyjne

- dla zaprzeczenia φ tworzymy automat Büchiego $B_{\neg\varphi}$
- Mnożymy oba automaty:

$$B = B_S \times B_{\neg\varphi}$$

- sprawdzamy pustość języka $L(B)$

Iloczyn automatów Büchiego – przypomnienie

$$B_1 = (\Sigma, Q_1, Q_0^1, \delta_1, F_1)$$
$$B_2 = (\Sigma, Q_2, Q_0^2, \delta_2, F_2)$$
$$B = B_1 \times B_2 = (\Sigma, Q, Q_0, \delta, F)$$

- $Q = Q_1 \times Q_2 \times \{0, 1, 2\}$
- $Q_0 = Q_0^1 \times Q_0^2 \times \{0\}$
- $F = Q_1 \times Q_2 \times \{2\}$
- $(s_1, s_2, x)\vec{a}(t_1, t_2, y) \iff s_1 \vec{a} t_1 \wedge s_2 \vec{a} t_2$

x		y	
0	$\rightarrow$	1	jeżeli $t_1 \in F_1$
1	$\rightarrow$	2	jeżeli $t_2 \in F_2$

Podejście tradycyjne – DDFS

DFS1(s):

$T \leftarrow T \cup (s, 0)$;
S.push(s);
for $t \in next(s)$ **do**
  if $(t, 0) \notin T \wedge$ DFS1(t) **then**
    return *true*;
  end if
end for
if $s \in F \wedge (s, 1) \notin T \wedge$ DFS2(s) **then**
  return *true*;
end if
S.pop;
return *false*;

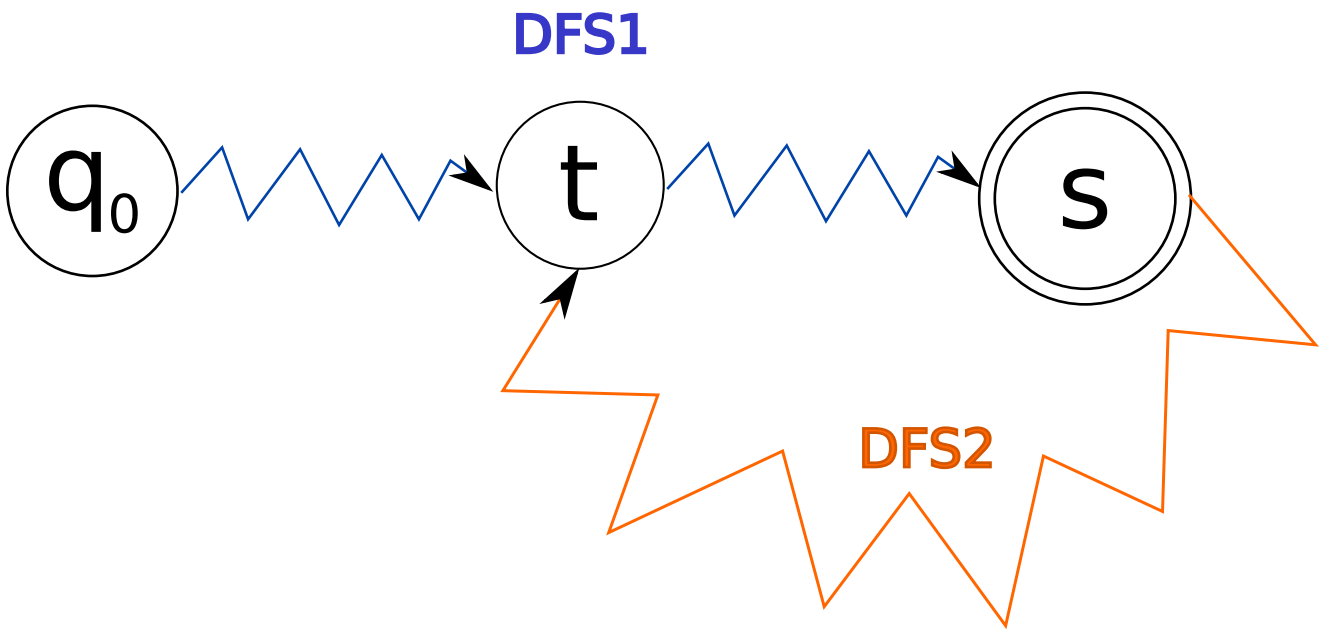
Podejście tradycyjne – DDFS

DFS2(s):

$T \leftarrow T \cup (s, 1)$;
for $t \in next(s)$ **do**
  if $t \in S$ **then**
    return *true*;
  end if
  if $(t, 1) \notin T \wedge$ DFS2(t) **then**
    return *true*;
  end if
end for
return *false*;

Podejście tradycyjne – DDFS

Poszukiwanie lassa ze stanem akceptującym w cyklu.



Monte Carlo

Zamiast przeglądać kolejne krawędzie wychodzące z każdego stanu, próbkujemy losowe “wykonania” programu (ścieżki w automacie B) w poszukiwaniu akceptującego cyklu.

Przestrzeń lass

- Prawdopodobieństwo przebiegu skończonego $\sigma = s_0 s_1 \dots s_n$

$$P(s_0) = \frac{1}{|Q_0|}$$

$$P(s_0 s_1 \dots s_n) = P(s_0 \dots s_{n-1}) \cdot \frac{1}{m},$$

gdzie $m = |\{t \mid \exists a\ (s, a, t) \in \delta\}|$

Uwaga

- Każdą tranzycję z danego stanu uważamy za równie prawdopodobną.

Przestrzeń lass

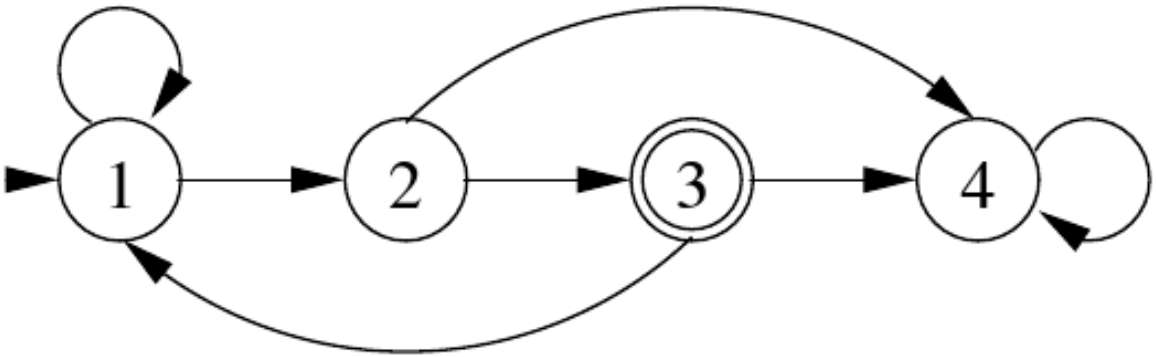
- Zmienna losowa akceptacji Z
- Wartość oczekiwana Z:

$$p_Z = P(Z = 1) = \sum_{\sigma \in L_a} P(\sigma)$$

Interpretacja

- p_Z jest prawdopodobieństwem, że losowe lasso w B jest akceptujące.

Przestrzeń lass – przykład



- cztery lassa: 1-1, 1-2-4-4, 1-2-3-1, 1-2-3-4-4
- prawdopodobieństwa lass: $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}$
- $p_Z = \frac{1}{8}$

Algorytm MC^2 – pojedyncza próbka

$T = \emptyset, i = 0, f = 0;$
$s \leftarrow$ losowy stan z Q_0;
while $(s, *) \notin T$ **do**
  $i \leftarrow i + 1;$
  $T \leftarrow T \cup (s, i);$
  if $s \in F$ **then**
    $f \leftarrow i;$
  end if
  $s \leftarrow$ losowy następnik s;
end while
if $(s, i) \in T \wedge i < f$ **then**
  return kontrprzykład T;
end if

Liczba próbek

Jak wielu próbek potrzebujemy?

- Schemat Bernoulliego:
  - N niezależnych prób
  - prawdopodobieństwo sukcesu p_Z
  - prawdopodobieństwo porażki $q_Z = 1 - p_Z$
- X - liczba prób do sukcesu
  - zmienna losowa z rozkładem geometrycznym

Liczba próbek

X - liczba prób do sukcesu

- zmienna losowa z rozkładem geometrycznym
- $P(X = k) = q_Z^{k-1} p_Z$
- $P(X \leq k) = \sum_{i=0}^{k} P(X = i) = 1 - q_Z^k$

Liczba próbek

Żądamy, aby $P(X \leq N) \geq 1 - \delta$:

$$1 - q_Z^N \geq 1 - \delta,$$

$$N \geq \frac{\ln(\delta)}{\ln(1 - p_Z)}$$

Zakładamy ponadto, że $p_Z \geq \varepsilon$:

$$N' \geq \frac{\ln(\delta)}{\ln(1 - \varepsilon)} \geq \frac{\ln(\delta)}{\ln(1 - p_Z)}$$

Złożoność algorytmu MC^2

Aby stwierdzić, że

$$P(p_Z \geq \varepsilon) \leq \delta$$

należy przeanalizować $N = \frac{ln(\delta)}{\ln(1-\varepsilon)}$ próbek.

Jeżeli przez D oznaczymy długość najdłuższej ścieżki prostej w B, to algorytm MC^2 działa w czasie

$$\mathcal{O}(N \cdot D)$$

i pamięci

$$\mathcal{O}(D)$$

Porównanie: k filozofów

zakleszczenie

	DDFS		MC^2		
ph	time	entr	time	max	avg
4	0:01	178	0:20	49	21
5	0:01	500	0:27	77	28
6	0:03	1772	0:45	116	42
7	0:11	5344	1:23	188	66
8	0:58	18244	2:42	365	99
9	3:54	57334	4:30	527	151
10	16:44	192476	7:20	720	234
12	–	oom	21:20	1665	564
14	–	oom	1:09:52	2994	1442
16	–	oom	3:03:40	7358	3144
18	–	oom	6:41:30	13426	5896
20	–	oom	19:02:00	34158	14923

zagłodzenie

	DDFS		MC^2		
ph	time	entr	time	max	avg
4	0:01	538	0:20	50	21
5	0:03	1986	0:28	79	30
6	0:17	9106	0:46	123	42
7	1:24	36031	1:19	182	64
8	7:56	161764	2:17	276	97
9	43:39	667221	4:47	474	155
10	–	oom	7:37	760	240
12	–	oom	21:34	1682	570
14	–	oom	1:09:45	3001	1363
16	–	oom	2:50:50	6124	2983
18	–	oom	8:24:10	17962	7390
20	–	oom	22:59:10	44559	17949

$$\delta = 0.1, \varepsilon = 1.8 \cdot 10^{-4}, N = 1257$$

Porównanie: protokół Needhama-Schrödera

	DDFS		MC^2				
mr	time	entr	time	mxl	cxl	N	$\widetilde{p}_z$
4	0.38	607	1.68	87	87	103	6.4E-3
8	1.24	2527	11.3	208	65	697	0.9E-3
16	5.87	13471	10.2	223	61	612	1.1E-3
24	18.7	39007	3:06	280	44	12370	5.5E-4
32	36.2	85279	2:54	269	63	11012	6.2E-4
40	1:11	158431	1:46	325	117	7818	8.8E-4
48	2:03	264607	1:45	232	25	6997	9.8E-4
56	3:24	409951	6:54	278	133	28644	2.4E-4
64	5:18	600607	7:12	347	32	29982	2.3E-4
72	–	oom	11:53	336	63	43192	1.6E-4

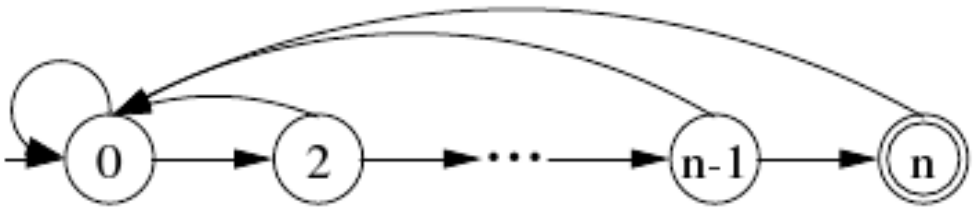
$$\delta = 10^{-3}$$

Warianty

- Oszacowanie p_Z z obu stron:

$$P(p_Z(1-\varepsilon) \leq \tilde{p_Z} \leq p_Z(1+\varepsilon)) \geq 1-\delta$$

- Multi-lasso: ignorowanie lass nieakceptujących



Podsumowanie

Zalety

- Niewielki koszt obliczeniowy i pamięciowy
- Świetna do szybkiego wyszukiwania błędów

Wady

- Brak pewności!

Randomizacja DDFS

```
DFS1(s):
  ...
  for t ∈ next(s) do
    if (t, 0) ∉ T ∧ DFS1(t) then
      return true;
    end if
  end for
  ...
```

```
DFS2(s):
  ...
  for t ∈ next(s) do
    if t ∈ S then
      return true;
    end if
    ...
  end for
```

Randomizacja DDFS

```
DFS1(s):
  ...
  for t ∈ next(s) do
    if (t, 0) ∉ T ∧ DFS1(t) then
      return true;
    end if
  end for
  ...
```

```
DFS2(s):
  ...
  for t ∈ next(s) do
    if t ∈ S then
      return true;
    end if
    ...
  end for
```

Przykład

```
c := 0; i := 0;
while i < 1000 do
  if
    true -> c := c + 1
    true -> c := c + 2
  fi
  i := i + 1
od
```

- $G(c < 2000 - d)$
- $G(i < 1000 \vee c \neq 1500)$

Randomizacja spamiętywania stanów

Motywacja:

- dla zatrzymania się i poprawności algorytmu DDFS nie potrzeba tablicy haszującej odwiedzonych stanów
- w tablicy można utrzymywać tylko niektóre stany

Problem: które stany zapamiętać?

- nie wiemy, które będą często odwiedzane
- wybierzmy losowo!

Spamiętywanie stanów: strategia dynamiczna

- przy każdym wycofaniu się DDFS ze stanu jest on usuwany z tablicy z prawdopodobieństwem P_{Del}, w przeciwnym wypadku jest zapamiętywany na zawsze.
- po k odwiedzinach prawdopodobieństwo spamiętania wynosi

$$P_{ESt} = 1 - P_{Del}^{k}$$

Wada

- odwlekanie spamiętywania często odwiedzanych stanów

Spamiętywanie stanów: strategia dynamiczna

Prawdopodobieństwo, że stan zostanie spamiętany przy i-tym odwiedzeniu:

$$P_{Del}^{i-1} P_{Sto}$$

Średnia liczba odwiedzin do spamiętania:

$$\frac{1}{P_{Sto}}$$

Redukcja pamięci, o ile każdy stan jest odwiedzany k razy:

$$P_{Del}^{k} \cdot M$$

Wniosek

- kompromis między szybkością działania a wielkością użytej pamięci

Spamiętywanie stanów: strategia statyczna

- Stany do spamiętania są znane z góry
- Każdy stan zostaje albo spamiętany z prawdopodobieństwem P_{Sto} albo zapomniany na zawsze

Wada

- Niewrażliwość na liczbę odwiedzin

Strategie spamiętywania: wyniki

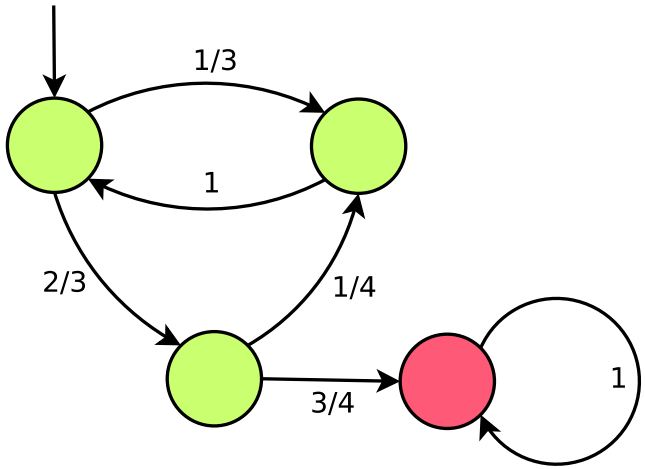
	States	Peak	Saving	Transitions	Overhead
SPIN	17068	17068	0%	32077	1.00
Dynamic Strategy					
$P_{sto} = 0.50$	10998	11421	33%	46074	1.44
$P_{sto} = 0.10$	6724	8263	52%	136344	4.25
$P_{sto} = 0.01$	5559	7407	57%	1110526	34.62
Static Strategy					
$P_{sto} = 0.75$	12807	12812	25%	38761	1.21
$P_{sto} = 0.50$	8568	9661	43%	63662	1.98
$P_{sto} = 0.40$	6852	8417	51%	390737	12.18

Algorytm Petersona

Systemy probabilistyczne

Łańcuch Markowa z czasem dyskretnym

$$\mathcal{M} = (S, s_0, P)$$



- S – zbiór stanów,
- s_0 – stan początkowy,
- $P : S^2 \to [0; 1]$ – rozkład prawdopodobieństw przejścia taki, że:

$$\forall_{s \in S} \sum_{t \in S} P(s, t) = 1$$

PLTL

Rozszerzenie LTL o więzy dla prawdopodobieństwa spełnienia formuły w losowym przebiegu:

$$\psi ::= P_{\trianglelefteq b}[\phi] \mid P_{=?}[\phi],$$

- ϕ – formuła LTL,
- $\trianglelefteq \in \{<, >, \leq, \geq\}$
- b – ograniczenie na prawdopodobieństwo

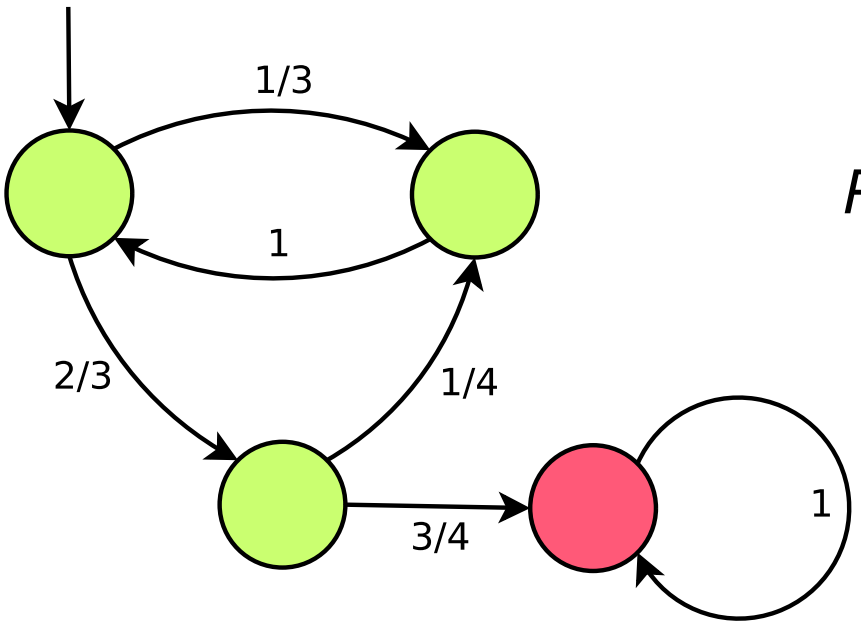
Macierze rzadkie

- Interpretujemy P jako macierz (stochastyczną) prawdopodobieństw przejść
- Badamy zachowanie ciągu:

$$P, P \cdot P, P \cdot P \cdot P, \ldots, P^k, \ldots$$

w celu weryfikacji własności ψ

Macierze rzadkie: przykład

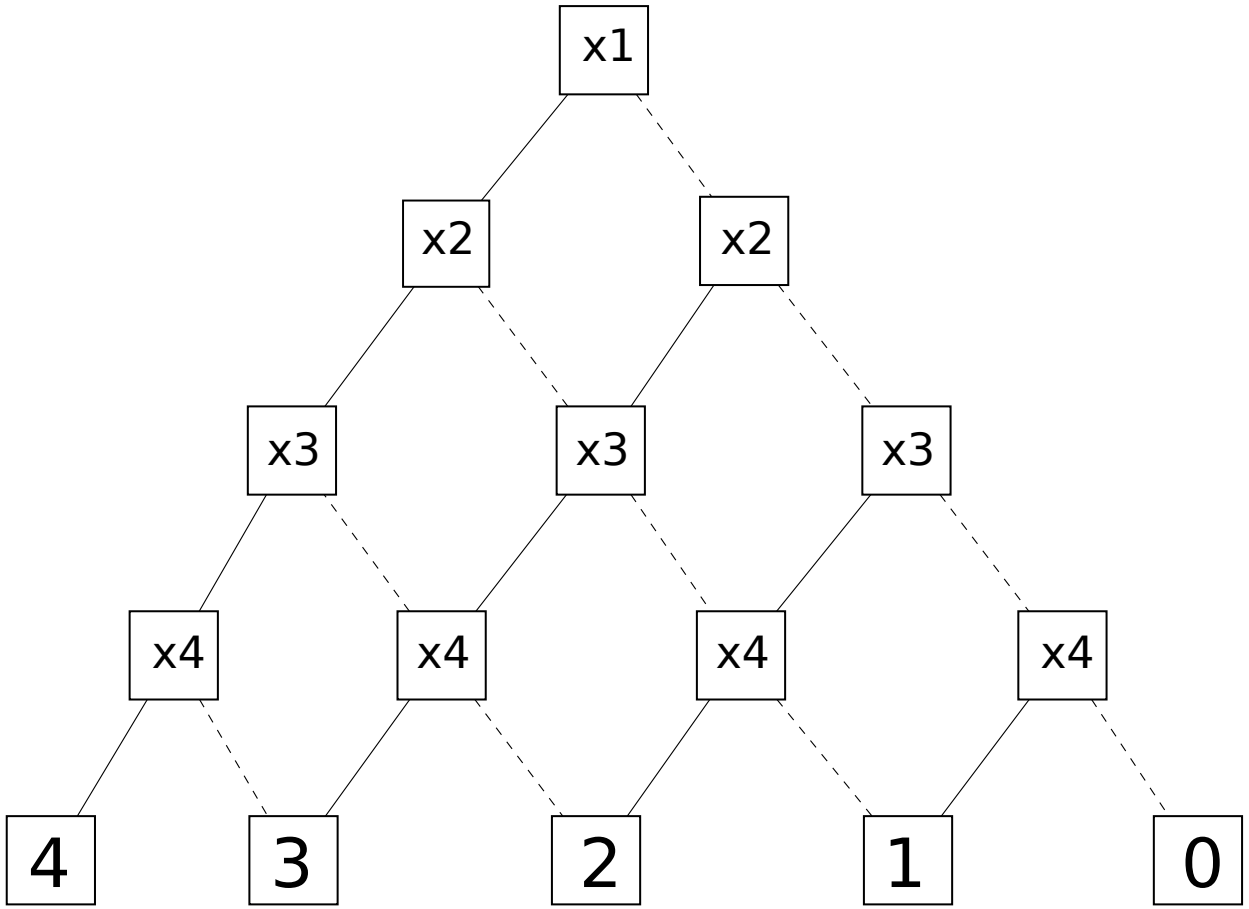


$$P = \begin{bmatrix} & 1/3 & 2/3 & \\ 1 & & & \\ & 1/4 & & 3/4 \\ & & & 1 \end{bmatrix}$$

Weryfikacja symboliczna

- Analogicznie do SMC, reprezentujemy formułę i zachowanie modelu jako BDD
- W wierzchołkach terminalnych zamiast $\{0, 1\}$ znajdują się prawdopodobieństwa
  - Multi-Terminal Binary Decision Diagram – reprezentuje funkcję $\{0, 1\}^m \rightarrow \mathbb{R}$

MTBDD – przykład



Aproksymacja Monte Carlo

Będziemy rozpatrywać tylko podzbiór EPF $\subset$ LTL:

- zmienne zdaniowe i ich negacje,
- alternatywa, koniunkcja,
- operatory temporalne U i X.

Monotoniczność

Formuły EPF są *monotoniczne*:

$$\forall_{k>0}\forall_{\pi_k} \quad \pi \models_{\mathcal{M}} \phi \;\Rightarrow\; \pi^{+} \models_{\mathcal{M}} \phi,$$

gdzie π^{+} jest dowolną ścieżką o prefiksie π

Aproksymacja Monte Carlo

Sprawdzanie własności przez próbkowanie ścieżek w łańcuchu Markowa.

- połączenie BMC i aproksymacji Monte Carlo:
  - próbkowanie tylko ścieżek długości k

Aproksymacja Monte Carlo

Weryfikujemy własność $P_{\geq b}[\phi]$:

- dobieramy $k \sim log(|S|)$
- próbkujemy N ścieżek długości k obliczając oszacowanie prawdopodobieństwa spełnienia formuły:

$$\frac{A}{N} \approx P_k[\phi]$$

- jeżeli $P_k[\phi] \geq b$, to z monotoniczności również $P[\phi] \geq b$, w przeciwnym wypadku zwiększamy k

Poprawność aproksymacji

Chcemy uzyskać prawdopodobnie dokładną aproksymację $\tilde{p} \approx P_k[\phi] \stackrel{ozn.}{=} p$:

$$P\left(p - \varepsilon \leq \tilde{p} \leq p + \varepsilon\right) \geq 1 - \delta$$

Aby tak było, należy wziąć $\mathcal{O}\left(\ln \frac{1}{\delta} \cdot \frac{1}{\varepsilon}\right)$ próbek (ścieżek).

Poprawność aproksymacji

Nierówność Chernoffa-Hoeffdinga

Dla m niezależnych zmiennych losowych o tym samym rozkładzie Bernoulliego (z prawdopodobieństwem sukcesu p), dla dowolnego $\varepsilon \in [0; 1]$:

$$P_\varepsilon = P\left(\frac{1}{m}\sum_i X_i > p + \varepsilon\right) \leq e^{-m\Theta(\varepsilon)}$$

Podsumowanie

Zalety

- niskie zużycie pamięci
- możliwe zrównoleglenie obliczeń

Wady

- stosunkowo powolne dla niewielkich systemów
- z natury niedokładne

Zastosowania

Model checking systemów deterministycznych:

- szybkie odnajdywanie błędów
- niemożliwa pełna weryfikacja

Systemy probabilistyczne:

- efektywne znajdowanie dobrych oszacowań na prawdopodobieństwo
- zastosowania w bioinformatyce, m.in.:
  - symulacje reakcji chemicznych
  - kaskady sygnałowe w błonach komórkowych

Bibliografia


Sistla A. P., Clarke E. M.,
The complexity of propositional linear temporal logics,
Journal of the ACM, vol. 32 n. 3, pp. 733–749, 1985


Vardi, M. Y., Wolper, P.,
An automata-theoretic approach for automatic program verification,
Proc. IEEE Symposium on Logic in Computer Science, pp. 332–344, 1986.


Brim L., Černá I., Nečesal M.,
Randomization helps in LTL model checking,
Proc. Joint International Workshop, PAPM-PROBMIV 2001, pp. 105–119, Springer 2001.

Bibliografia


Guirado G., Herault T., Lassaigne R., Peyronnet S.,
Distribution, Approximation and Probabilistic Model Checking,
ENTCS, Vol. 135, n. 2, pp. 19–30, Elsevier 2006


Donaldson R., Gilbert D.,
A Monte Carlo model checker for probabilistic LTL with numerical constraints,
DCS Technical Report, Uniwersytet w Glasgow, 2008.