Monte Carlo model checking

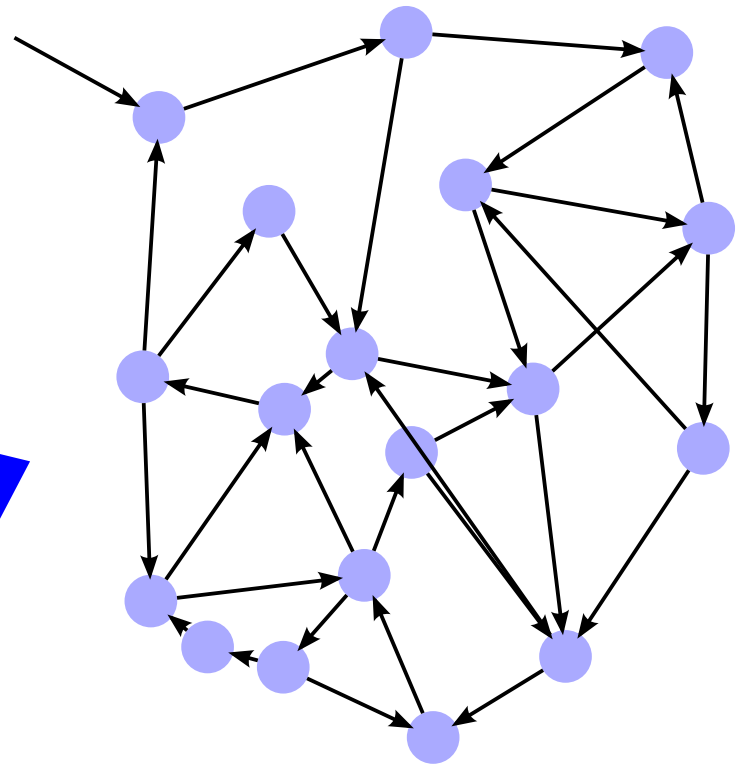
Marcin Sulikowski Maciej Pazurkiewicz

MIMUW

25 maja 2011

Weryfikacja modelowa

```
y := x;
z := 1;
while y > 0 do
    if y % 2 = 1 then
        z := z + 1;
    fi;
    z := z * y;
    y := y - 1;
od;
y := 0;
```



System S

Struktura Kripkego K

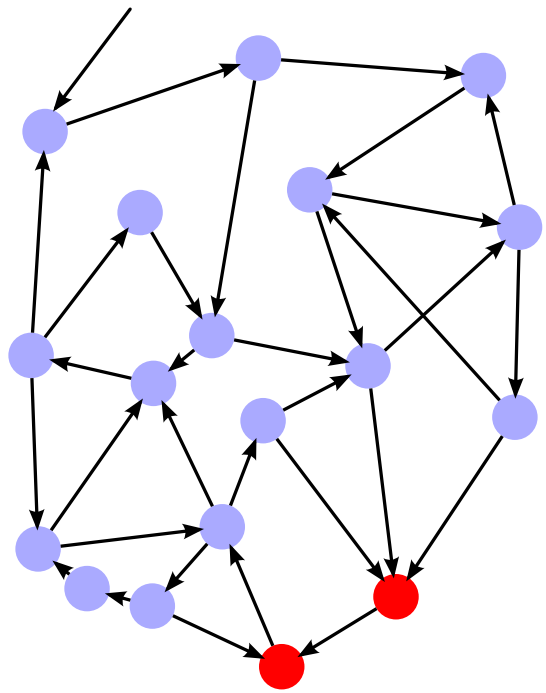
Weryfikacja modelowa

K – struktura Kripkego
φ – własność w LTL

Ogólna metoda:

$$\mathcal{L}\left(B_K \times B_{\neg\varphi}\right) \stackrel{?}{=} \emptyset$$

Algorytm DDFS:
Znaleźć lasso zawierające stan akceptujący
Czas i pamięć (dla $|\varphi| = \mathcal{O}(1)$): $\mathcal{O}(|K|)$



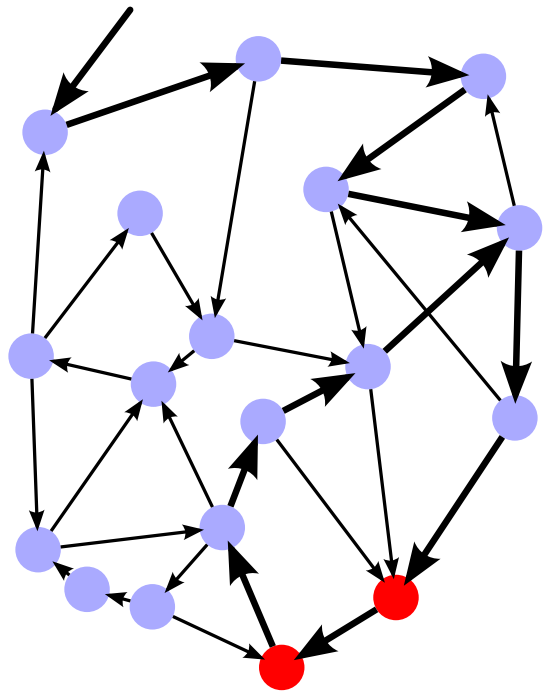
Weryfikacja modelowa

K – struktura Kripkego
φ – własność w LTL

Ogólna metoda:

$$\mathcal{L}\left(B_K \times B_{\neg\varphi}\right) \stackrel{?}{=} \emptyset$$

Algorytm DDFS:
Znaleźć lasso zawierające stan akceptujący
Czas i pamięć (dla $|\varphi| = \mathcal{O}(1)$): $\mathcal{O}(|K|)$



Weryfikacja modelowa

$|K| = \mathcal{O}\left(2^{|S|}\right)$, ale:

- redukcje częściowo-porządkowe (1990)
- OBDD – weryfikacja symboliczna (1992)
- BMC – weryfikacja ograniczona (1999)
- MC^2 – podejście probabilistyczne (2005)

Podejście probabilistyczne

Nie sprawdzamy wszystkich lass w $B_K \times B_{\neg\varphi}$, ale kilka losowych.

- Znaleźliśmy lasso akceptujące – φ nie zachodzi
- Nie znaleźliśmy lassa akceptującego – **?**

Podejście probabilistyczne – szacowanie

Z – zmienna losowa o rozkładzie dwupunktowym,
$P(Z = 1) = p_Z$, $P(Z = 0) = 1 - p_z = q_Z$

X – zmienna losowa równa liczbie prób do sukcesu, tj. do $Z = 1$.

$$p(N) = P(X = N) = q_Z^{N-1} p_Z$$

$$F(N) = P(X \leqslant N) = \sum_{n \leqslant N} p(n) = 1 - q_Z^N$$

$$F(N) \geqslant 1 - \delta \iff N \geqslant \ln(\delta)/\ln(1 - p_Z)$$

Przykład:

$$\frac{\ln(0{,}01)}{\ln\left(1-\frac{1}{6}\right)} = 25{,}26$$

Podejście probabilistyczne – szacowanie

$$H_0 : p_z \geqslant \epsilon$$

$$M = \ln(\delta)/\ln(1-\epsilon) \geqslant N = \ln(\delta)/\ln(1-p_z)$$

$$P(X \leqslant M \mid H_0) \geqslant 1-\delta$$

$$P(X > M \mid H_0) < \delta$$

Podejście probabilistyczne – szacowanie

$$H_0 : p_z \geqslant \epsilon$$

$$M = \ln(\delta)/\ln(1-\epsilon) \geqslant N = \ln(\delta)/\ln(1-p_z)$$

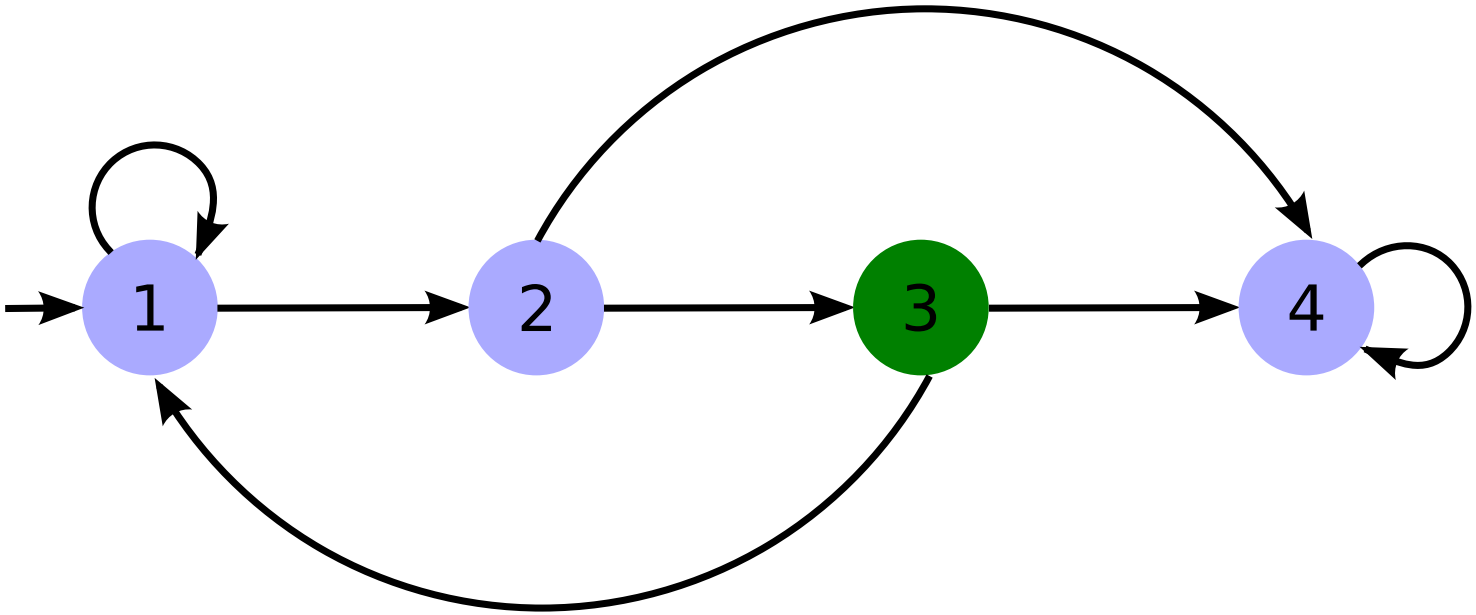
$$P(X \leqslant M \mid H_0) \geqslant 1 - \delta$$

$$P(X > M \mid H_0) < \delta$$

Wniosek: jeśli po $M = \frac{\ln(\delta)}{\ln(1-\epsilon)}$ próbach nie ma sukcesu, to z prawdopodobieństwem mniejszym niż δ mamy $p_z \geq \epsilon$.

Przestrzeń lass

Lassa tworzą przestrzeń probabilistyczną.



$$P[11] = \frac{1}{2}$$

$$P[1244] = \frac{1}{4}$$

$$P[1231] = \frac{1}{8}$$

$$P[12344] = \frac{1}{8}$$

Algorytm MC^2

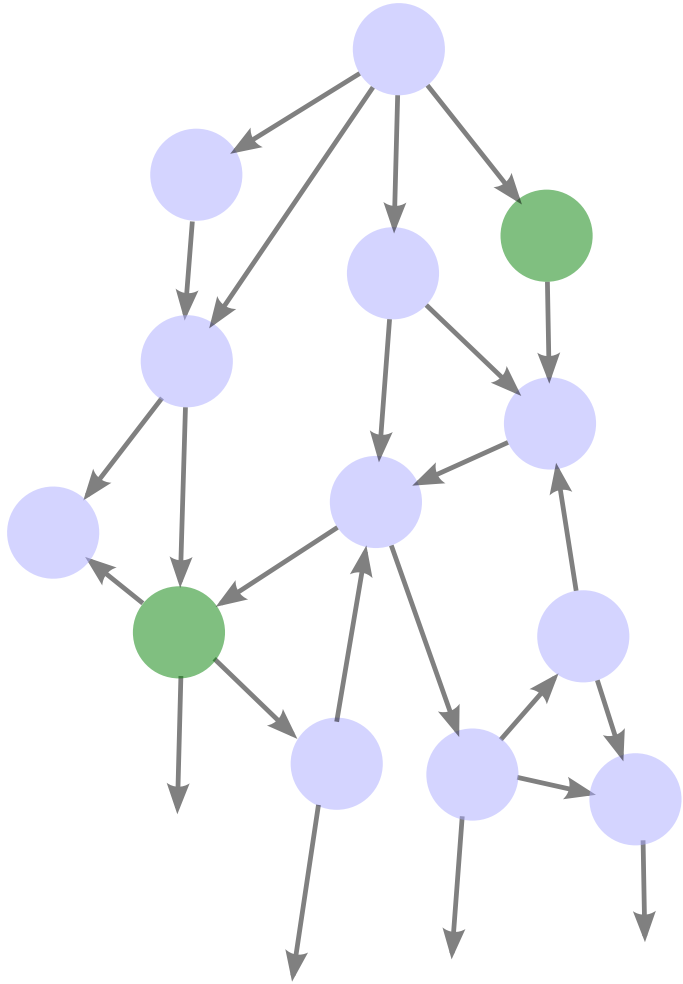
Algorytm polega na losowym przeszukiwaniu przestrzeni lass.

- jeśli znajdziemy lasso akceptujące, wiemy, że weryfikowana własność jest fałszywa i mamy stosowny kontrprzykład
- w przeciwnym wypadku, mamy pewne górne oszacowanie na prawdopodobieństwo wystąpienia takiego lassa

M := liczba_prob $// \frac{\ln \delta}{\ln 1-\epsilon}$
for i in 1..M
  if RandomLasso(B) == ⟨true, accepting_lasso⟩
    return false
return true

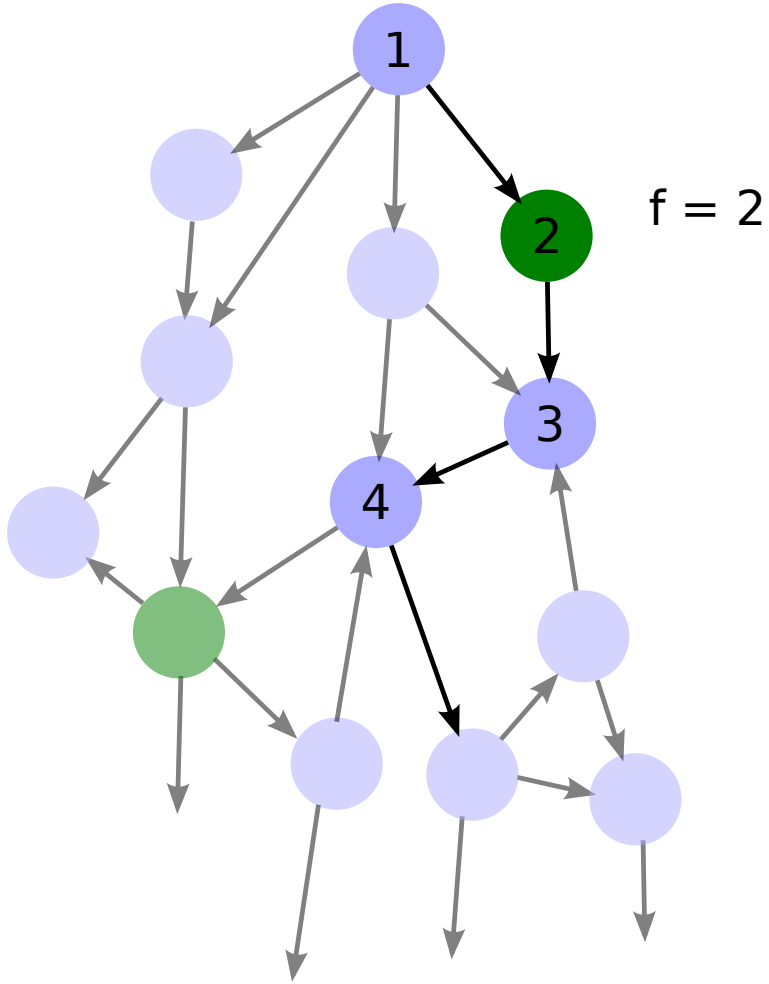
Znajdowanie lass

```
s := randomInit(B)
i := 0; f := 0
while(s ∉ HashTable)
   HashTable(s) := i++
   if accepting(s, B)
      f := i
   s := randomNext(s, B)

if (HashTable(s) ≤ f)
   return ⟨true, lasso(HashTable)⟩
else
   return ⟨false, null⟩
```



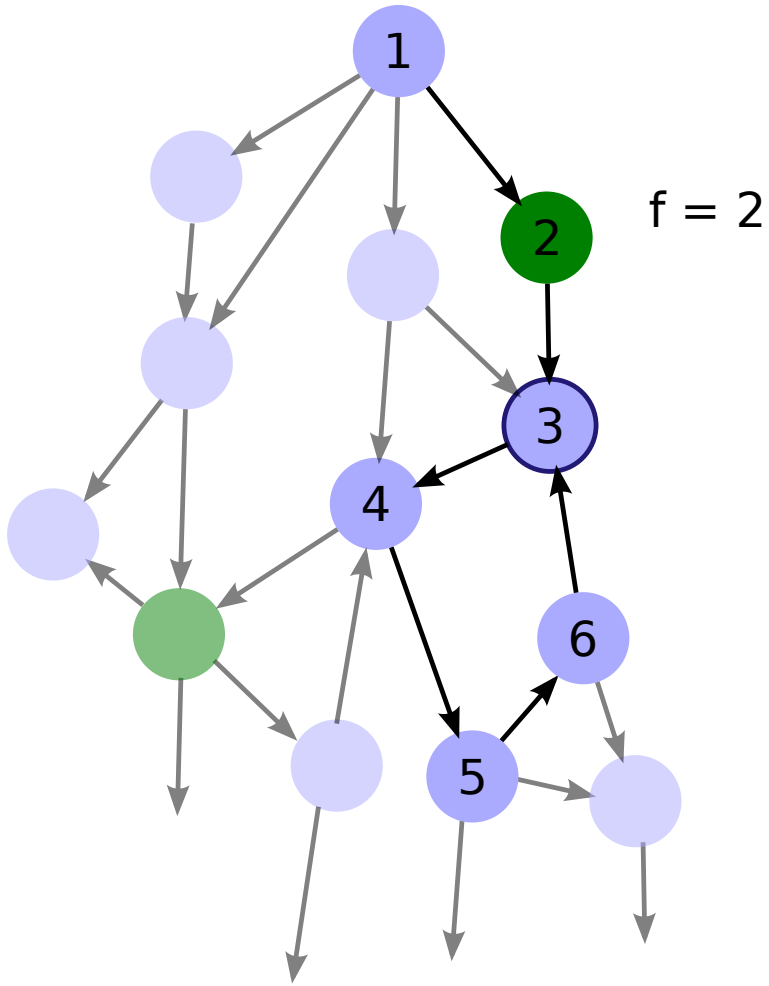
Znajdowanie lass

```
s := randomInit(B)
i := 0; f := 0
while(s ∉ HashTable)
   HashTable(s) := i++
   if accepting(s, B)
      f := i
   s := randomNext(s, B)

if (HashTable(s) ≤ f)
   return ⟨true, lasso(HashTable)⟩
else
   return ⟨false, null⟩
```



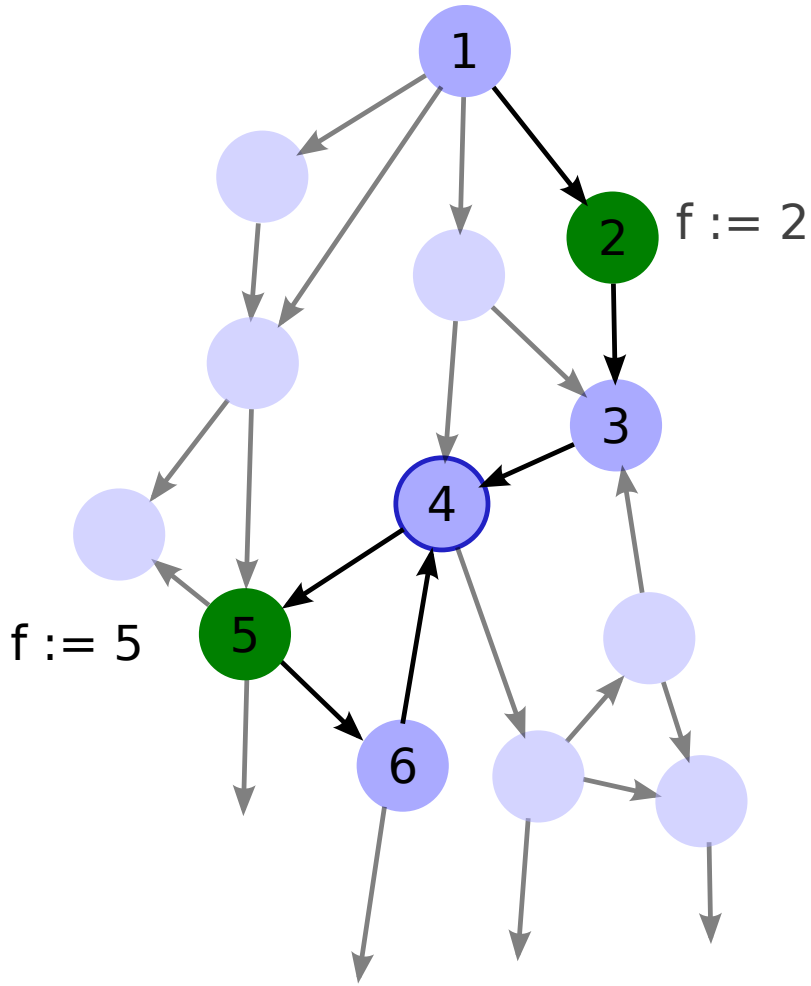
Znajdowanie lass

```
s := randomInit(B)
i := 0; f := 0
while(s ∉ HashTable)
   HashTable(s) := i++
   if accepting(s, B)
      f := i
   s := randomNext(s, B)

if (HashTable(s) ≤ f)
   return ⟨true, lasso(HashTable)⟩
else
   return ⟨false, null⟩
```



Znajdowanie lass

```
s := randomInit(B)
i := 0; f := 0
while(s ∉ HashTable)
   HashTable(s) := i++
   if accepting(s, B)
      f := i
   s := randomNext(s, B)

if (HashTable(s) ≤ f)
   return ⟨true, lasso(HashTable)⟩
else
   return ⟨false, null⟩
```



Algorytm MC^2: złożoność

Złożoność czasowa algorytmu MC^2 wynosi $\mathcal{O}(MD)$, natomiast złożoność pamięciowa – $\mathcal{O}(D)$, gdzie

- M – liczba prób
- D – średnica grafu stanów (długość najdłuższej ścieżki prostej)

Algorytm MC^2: złożoność

Złożoność czasowa algorytmu MC^2 wynosi $\mathcal{O}(MD)$, natomiast złożoność pamięciowa – $\mathcal{O}(D)$, gdzie

- M – liczba prób
- D – średnica grafu stanów (długość najdłuższej ścieżki prostej)

W teorii D może być wykładnicze ze względu na $|S + \varphi|$, co sprawia, że asymptotyczne złożoności MC^2 i pełnego przeszukiwania DDFS są równe. W praktyce D jest o wiele mniejsze.

Potencjalne rozszerzenia

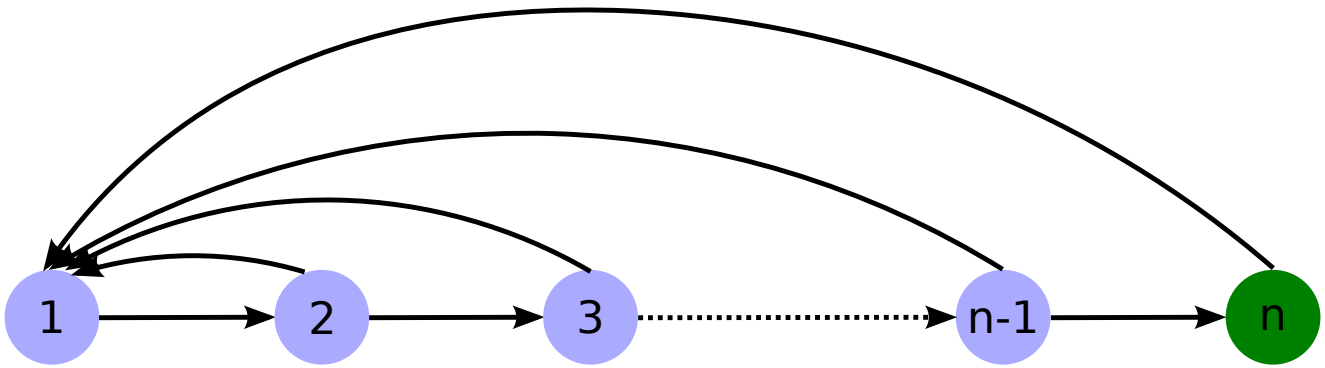
- Algorytm MC^2 jest niezależny od technik stosowanych w klasycznym *model checkingu* takich jak:
  - abstrakcja
  - redukcje częściowo-porządkowe
- Warto rozważać również inne strategie próbkowania przestrzeni lass.

Algorytm MC^2: niejednostajne próbkowanie

- Przedstawiony algorytm próbkowania przestrzeni lass prowadzi do tego, że krótkie lassa znajdowane są o wiele częściej niż długie.
- W zasadzie to dobrze:
  - często błędy nie leżą "głęboko",
  - dostajemy krótsze kontrprzykłady.

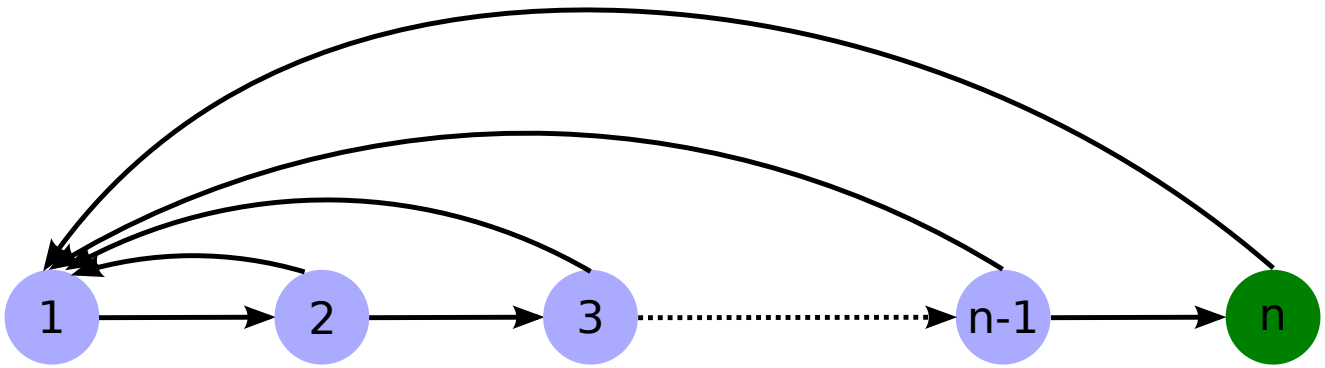
Algorytm MC^2: niejednostajne próbkowanie

Rozpatrzmy skrajny przypadek.



Prawdopodobieństwo wylosowania lassa akceptującego wynosi $\frac{1}{2^n}$.

Algorytm MC^2: niejednostajne próbkowanie



Zaproponowane rozwiązanie może polegać na odrzucaniu krawędzi powrotnych nietworzących lassa akceptującego, o ile nadal są nieodwiedzone krawędzie w przód.

Algorytm MC^2: wyniki

	DDFS		MC^2			
ph	time	entr	time	mxl	cxl	M
4	0.02	31	0.08	10	10	3
8	1.62	511	0.20	25	8	7
12	3:13	8191	0.25	37	11	11
16	>20:0:0	–	0.57	55	8	18
20	–	oom	3.16	484	9	20
30	–	oom	35.4	1478	11	100
40	–	oom	11:06	13486	10	209

	DDFS		MC^2			
ph	time	entr	time	mxl	cxl	M
4	0.17	29	0.02	8	8	2
8	0.71	77	0.01	7	7	1
12	1:08	125	0.02	9	9	1
16	7:47:0	173	0.11	18	18	1
20	–	oom	0.06	14	14	1
30	–	oom	1.12	223	223	1
40	–	oom	1.23	218	218	1

Table 1. Deadlock and starvation freedom for symmetric (unfair) version.

Algorytm MC^2: wyniki

	DDFS		MC^2		
ph	time	entr	time	mxl	avl
4	0:01	178	0:20	49	21
6	0:03	1772	0:45	116	42
8	0:58	18244	2:42	365	99
10	16:44	192476	7:20	720	234
12	–	oom	21:20	1665	564
16	–	oom	3:03:40	7358	3144
20	–	oom	19:02:00	34158	14923

	DDFS		MC^2		
ph	time	entr	time	mxl	avl
4	0:01	538	0:20	50	21
6	0:17	9106	0:46	123	42
8	7:56	161764	2:17	276	97
10	–	oom	7:37	760	240
12	–	oom	21:34	1682	570
16	–	oom	2:50:50	6124	2983
20	–	oom	22:59:10	44559	17949

Table 2. Deadlock and starvation freedom for fair asymmetric version.

Algorytm MC^2: wyniki

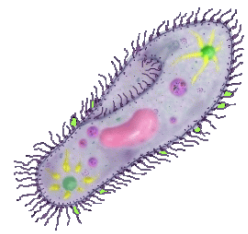
	DDFS		MC^2			
mr	time	entr	time	mxl	cxl	M
4	0.38	607	1.68	87	87	103
8	1.24	2527	11.3	208	65	697
16	5.87	13471	10.2	223	61	612
24	18.7	39007	3:06	280	44	12370
32	36.2	85279	2:54	269	63	11012

	DDFS		MC^2			
mr	time	entr	time	mxl	cxl	M
40	1:11	158431	1:46	325	117	7818
48	2:03	264607	1:45	232	25	6997
56	3:24	409951	6:54	278	133	28644
64	5:18	600607	7:12	347	32	29982
72	–	oom	11:53	336	63	43192

Table 3. Needham-Schroeder protocol.

Algorytm MC^2: podsumowanie

- Podejście randomizowane do weryfikacji LTL: przeglądamy tylko część lass w automacie Büchiego
- W przypadku porażki daje pewne informacje o prawdopodobieństwie ewentualnego sukcesu
- W porównaniu z DDFS ma o wiele lepszą złożoność pamięciową i czasową
- Można go zaimplementować używając OBDD dla dalszego zmniejszenia zużycia pamięci
- Łatwo się rozszerza na modele stochastyczne
- Niełatwo rozszerza się na weryfikację CTL



Bibliografia


R. Grosu, S. A. Smolka
Monte Carlo Model Checking
TACAS, tom 3440 LNCS, strony 271-286. Springer, 2005.