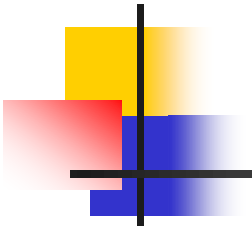
Лекция 13

Верификация параллельных программных и аппаратных систем

Курс лекций

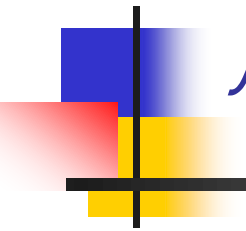
Шошмина Ирина Владимировна

Карпов Юрий Глебович



План курса

1. Введение
2. Метод Флойда-Хоара доказательства корректности программ
3. Исчисление взаимодействующих систем (CCS) Р.Милнера
4. Темпоральные логики
5. Алгоритм model checking для проверки формул CTL
6. Автоматный подход к проверке выполнения формул LTL
7. Система верификации Spin и язык Promela. Примеры верификации
8. Структура Крипке как модель реагирующих систем
9. Темпоральные свойства систем
10. Применения метода верификации model checking
11. BDD и их применение
12. Символьная проверка моделей
13. Количественный анализ дискретных систем
14. Верификация систем реального времени
15. Консультации по курсовой работе



Лекция 13

Количественный анализ дискретных систем. Probability and time

Требования к поведению систем. Model checking: проверка качественных свойств

Обычные алгоритмы Model checking - проверка ДОСТИЖИМОСТИ:

Примеры (LTL):

(a U b) – "*когда-то в будущем придем в состояние, из которого можно достичь b, а до этого во всех состояниях будет выполняться a*".

G(r $\rightarrow$ Fg) – "*всегда после того, как будет выставлен запрос r (request), разрешение g (grant) на доступ к ресурсу когда-нибудь в будущем будет получено*".

Никаких количественных гарантий эти свойства не имеют!

Расширения Model checking: количественный анализ

Последнее время разработано много расширений и различных приложений Model Checking.

Одна из групп – в Университете Бирмингема (а сейчас в Оксфордском Университете), исследует возможность комбинации **МС с вероятностным анализом и дискретным временем.**

"Количественные" методы верификации дают ответы типа:

Для протокола выбора лидера:

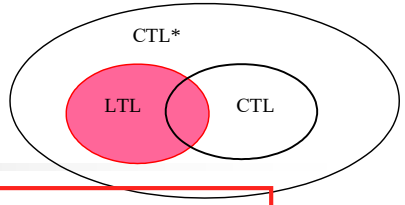
"С вероятностью p=0.9 процесс выбора лидера завершится в течение 25 шагов".

Для протокола передачи мультимедийной информации:

"Вероятность доставки кадра в течение 10 временных шагов ≥ 89%".

Для системы управления автомобиля: ***"максимальная вероятность того, что система открытия подушек безопасности не сработает в интервале 0.02 сек составляет 10^{-10}"***.

LTL



Некоторые свойства поведения дискретных систем удобно выражать LTL-формулами

Формулы LTL строятся как **A**α,

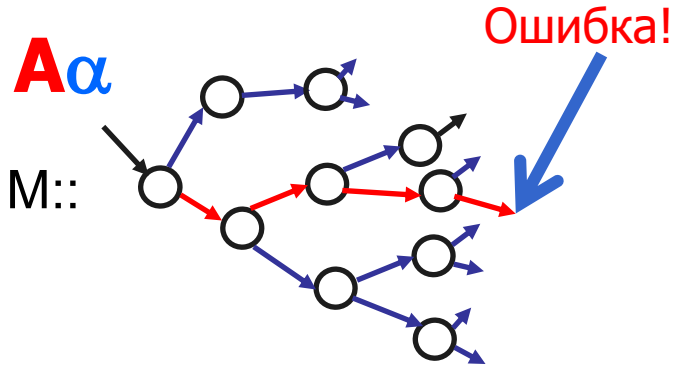
где α - формула пути.

α ::= p | ¬α | α ∨ α | α **U** α | **X**α

G и **F** выражаются через **U**ntil:

Fp = True **U** p; **G** p = ¬**F**¬p.

? M |= Aα

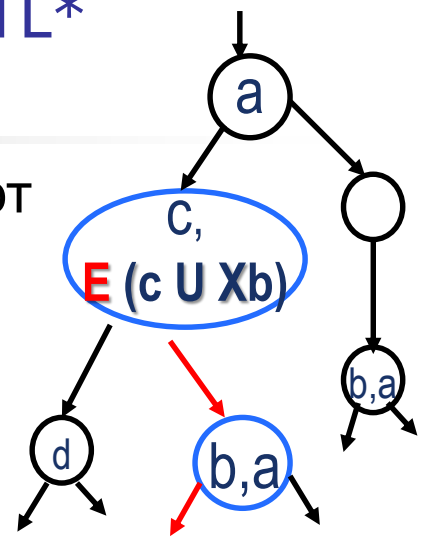


Формулы LTL понятны и удобны для выражения требований к поведению!

Формула LTL выполняется на структуре Крипке M, если она выполняется ***на любом пути (вычислении)***, начинающемся в начальном состоянии M

Проверка формулы LTL для M требует рассмотрения всех вычислений M, а таких вычислений *бесконечное* число, и каждое из них *бесконечно*.

Общая логика ветвящегося времени – CTL* Computational Tree Logic*



Темпоральные логики ветвящегося времени рассматривают ДЕРЕВЬЯ вычислений на развертке структуры Крипке.

Как построить темпоральную логику ветвящегося времени?

CTL* – одна из таких логик

Грамматика. Формула CTL* - это формула состояний ϕ:

- формулы состояний $\phi ::= p \mid \neg \phi \mid \phi \vee \phi \mid E\,\alpha \mid A\alpha$
- формулы путей $\alpha ::= \phi \mid \neg \alpha \mid \alpha \vee \alpha \mid \alpha U \alpha \mid X\alpha$

формула ϕ состояния s является формулой пути σ, если состояние s является начальным состоянием пути σ.

Формула пути – подобна формуле LTL!

Формула пути имеет смысл, только если зафиксирован путь!

В состояниях могут стоять только формулы состояний (state formulas)!

Вероятностная мера. Подход Ханссона - Джонссона

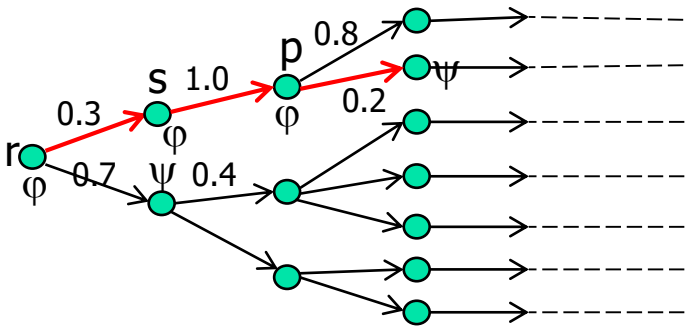
- Можно построить "простую" логику CTL, в которой вместо кванторов пути стоит *вероятность* того, что из данного состояния будет выбран путь, на котором данная LTL формула выполнится:

Грамматика. Формула PCTL - это формула состояний ϕ:

- Формулы состояний $\quad \phi ::= p \mid \neg\phi \mid \phi \vee \phi \mid \cancel{E}\alpha \mid \cancel{A}\alpha \mid P\alpha$
- Формулы путей $\quad \alpha ::= \phi\ U\ \phi \mid X\ \phi$

$P(\varphi U \psi) = ?$

$PX\varphi = ?$



в r: $PX\varphi = 0.3$

в p: $P(\varphi U \psi) = 0.2$

На вероятностной структуре Крипке в каждом ее состоянии можем подсчитать вероятности выполнения формул $X\varphi$ *и* $\varphi U \psi$

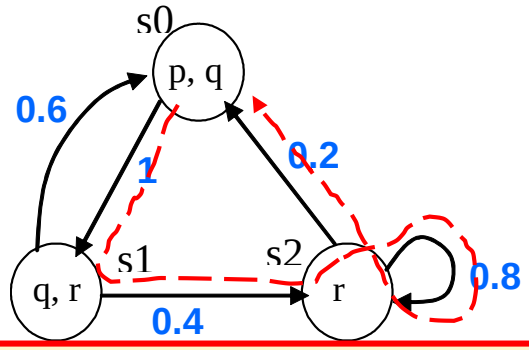
$P_{>0.1}$ = *вероятностная мера*

КАК СВЯЗАТЬ ЛОГИКУ и ВЕРОЯТНОСТЬ?

Ханссон и Джонссон (1994) ввели "*вероятностную меру*". Теперь можем говорить об истинности и ложности формул: если вероятностная мера выполняется, то формула истинна.

Model checking и вероятностный анализ

Модификация дискретной Марковской цепи (время считается неявно дискретными переходами из состояния в состояние, как и в структуре Крипке)



Помеченная дискретная Марковская цепь (S, s_0, P, L) – это:

S – конечное множество состояний,

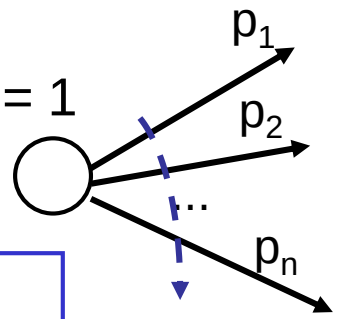
s_0 – начальное состояние;

$L: S \to 2^{AP}$ (с состоянием s из S связывается некоторое множество атомов, истинных в s).

$Pr: S \times S \to [0,1]$ – вероятностная матрица, такая, что $(\forall s \in S)\ \Sigma_{q \in S} Pr(s,q) = 1$

Можно считать эту модель расширением структуры Крипке – к структуре Крипке просто добавляются вероятности переходов Pr(s,q) из p в q. Переходы считаем независимыми событиями.

$\Sigma_i p_i = 1$



Оценка вероятности выбора вычисления σ :

Путь $\sigma = s^0 s^1 s^2 s^3 \ldots s^n$ – конечная цепочка состояний

$Pr(\sigma) = Pr(s^0 s^1 s^2 \ldots s^n) = Pr(s^0, s^1) \times Pr(s^1, s^2) \times Pr(s^2, s^3) \times \ldots \times Pr(s^{n-1}, s^n)$

Пример: $Pr(s^0 s^1 s^2\ s^2\ s^0) = 1 \times 0.4 \times 0.8 \times 0.2 = 0.064$

H. Hansson, B. Johnsson. A Logic for Reasoning about Time and Reliability // *Formal Aspects of Comput.* 6,1994)

Вероятностная CTL – PCTL (Hansson & Jonsson'94)

PCTL (Probabilistic CTL) заменяет кванторы E и A в CTL вероятностным оператором $Pr_{\sim p}(\alpha)$, где $p \in [0,1]$, $\sim \in \{\leq, <, =, \geq, >\}$, например, $P_{>0.3}(Xq)$

Формула состояния: $\varphi ::= q \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid P_{\sim p}(\alpha)$

где α -*формула пути:* $\alpha ::= X\varphi \mid \varphi_1 U \varphi_2$

~p вероятностная мера

Семантика вероятностного оператора:

(α - формула пути, s – состояние, $Paths_s$ – все пути из состояния s, σ – путь)

$s \models P_{\sim p}(\alpha)$ *iff* $Prob\{\sigma \in Paths_s \mid \sigma \models \alpha\} \sim p$

$P_{\sim p}(\alpha)$ – можно считать ***утверждением, истинным или ложным для состояния S:*** "*вероятностная мера ~p выполняется для формулы пути α, iff с мерой ~p из s может быть выбран путь, на котором выполняется формула α* ". Состояния можно ПОМЕЧАТЬ подформулами формулы φ.

$P_{>0.7}(\varphi U \neg\psi)$ – УТВЕРЖДЕНИЕ: *вероятность того, что на вычислениях из данного состояния выполнится формула пути* $(\varphi U \neg\psi)$, *больше* 0.7.

$P_{<0.1}((P_{>0.2} X\varphi) U \neg\psi)$ УТВЕРЖДЕНИЕ: *вероятность того, что на вычислениях из данного состояния выполнится формула пути* $(P_{>0.2} X\varphi) U \neg\psi$, *меньше* 0.1.

Как разбирать формулы логики PCTL?

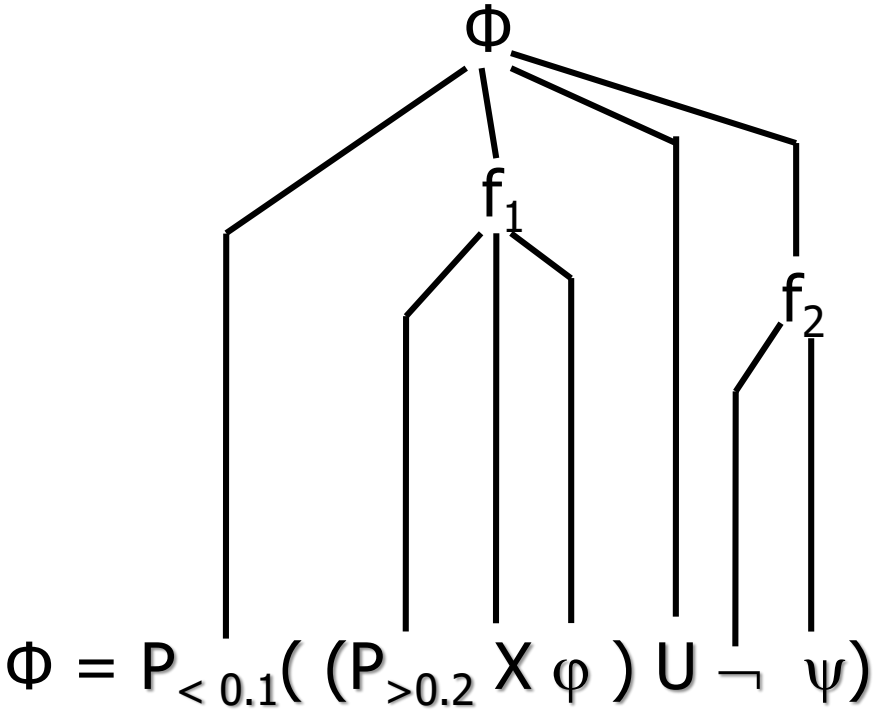
$\Phi = P_{< 0.1}((P_{>0.2}\, X\varphi)\, U \neg r)$

Формула состояния: $\varphi ::= q \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid P_{\sim p}(\alpha)$

где α -*формула пути:* $\alpha ::= X\,\varphi \mid \varphi_1\, U\, \varphi_2$

скобки можно опускать, если это не меняет смысла

1-й шаг всегда – построение синтаксического дерева, представляющего структуру формулы PCTL

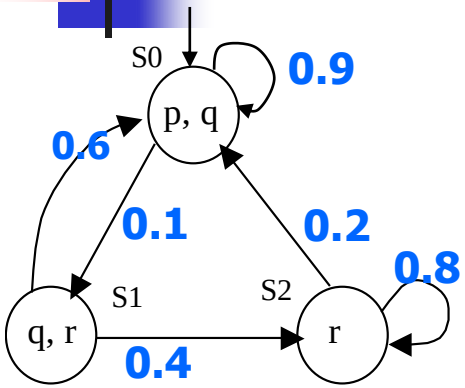


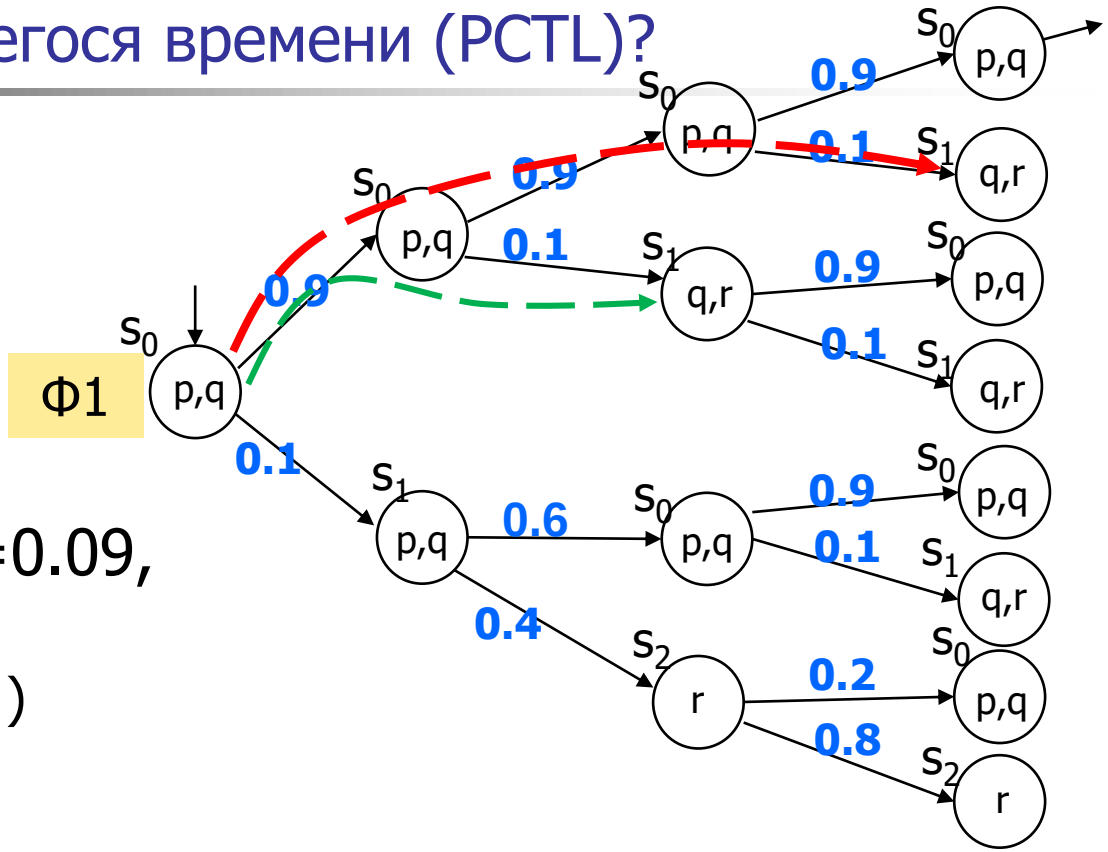
$f_1 = P_{> 0.2}\, X\, \varphi$

$f_2 = \neg\, \psi$

$\Phi = P_{< 0.1}(\, f_1\, U\, f_2)$

Как понимать формулы путей вероятностной логики ветвящегося времени (PCTL)?





P(- →)=0.081, P(- →)=0.09,

Φ1 = $P_{>0.4}$ Xq

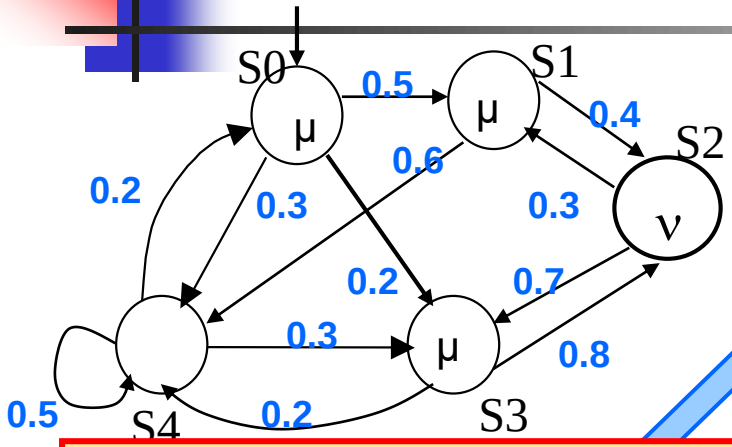
Φ2 = $P_{< 0.1}$ (($P_{>0.4}$ Xq) U¬p)

Φ1 выполняется в {s_0, s_1}
Φ2 не выполняется в s_0

Каждому отрезку пути из любого состояния S вероятностной структуры Крипке соответствует некоторая вероятность его выбора.

В логике PCTL формула состояний $P_{\sim p}(\alpha)$ ИСТИННА на тех состояниях, на которых *вероятность* выбора путей, на которых формула пути α выполняется, удовлетворяет указанной вероятностной мере ~p.

Вычисление истинности формулы $P_{\sim p}$ Xμ логики PCTL



Матрица вероятностей переходов

μ и ν - формулы состояний

Определим множество состояний, в которых выполняется формула $P_{\geq 0.6}$ Xμ (т.е. множество таких состояний, в которых вероятность формулы пути Xμ будет ≥0.6)

	s_0	s_1	s_2	s_3	s_4
s_0	--	0.5	–	0.2	0.3
s_1	--	--	0.4	--	0.6
s_2	--	0.3	--	0.7	–
s_3	--	--	0.8	--	0.2
s_4	0.2	--	--	0.3	0.5

X

1
1
0
1
0

Вектор- столбец: единицами отмечены те состояния, в которых удовлетворяется формула μ

= 0.7 0.0 1.0 0.0 0.5 Prob (Xμ) для всех s

Вероятность того, что из S_i за один шаг попадем в какое-нибудь состояние, в котором истинно μ

Мы получили вероятность выполнения формул. Надо перейти к утверждениям. Ведь вероятностная характеристика и логическое утверждение (утверждение о выполнимости) – ДВЕ разные сущности!

Алгоритм разметки для PCTL - формулы $P_{\sim p}X\mu$

- ИДЕЯ Hansson'a и Johnsson'a состоит в том, чтобы все свести к одной сущности – формулам состояний, **используя вероятностную меру.**
- Стандартный алгоритм разметки состояний для формул логики CTL *помечает* состояния, в которых выполняется формула состояний Ф этой логики.
- *Для формул логики PCTL* подсчитываем для каждого состояния вероятность того, что из него существует путь, удовлетворяющий формуле пути φ. Далее смотрим, в каких состояниях выполняется утверждение о **вероятностной мере**.

	s_0	s_1	s_2	s_3	s_4
s_0	--	0.5	–	0.2	0.3
s_1	--	--	0.4	--	0.6
s_2	--	0.3	--	0.7	–
s_3	--	--	0.8	--	0.2
s_4	0.2	--	--	0.3	0.5

x

1
1
0
1
0

Сумма вероятностей того, что из s_i за один шаг попадем в какое-нибудь состояние, в котором истинно μ

= 0.7 0.0 1.0 0.0 0.5

Prob (Xμ) для всех s

=>

$P_{\geq 0.6}$ Xμ

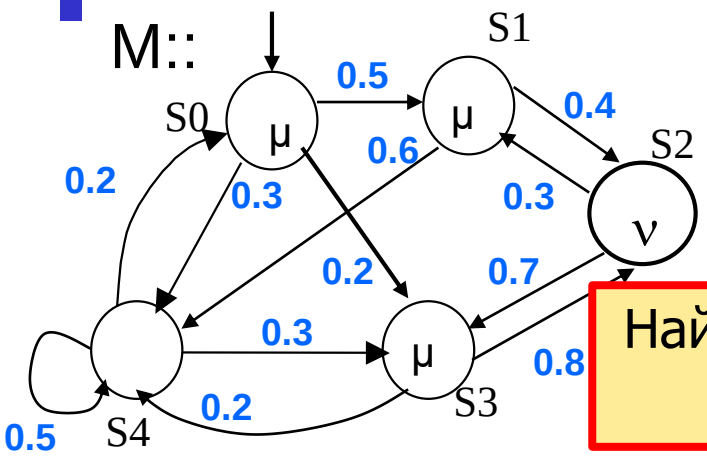
1
0
1
0
0

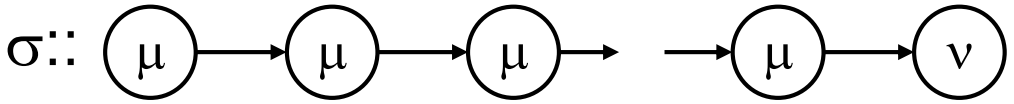
Единицами отмечены состояния, в которых удовлетворяется формула μ

формула состояний $P_{\geq \mathbf{0.6}}$ Xμ выполняется в состояниях S_0 и S_2

Вычисление истинности формулы $P_{\sim p}(\mu U \nu)$ логики PCTL

$(M, \sigma) \models \mu U \nu$ iff $\sigma = s_0 s_1 s_2 \ldots$ и
$\exists k \geq 0: (M, s_k) \models \nu$ и
$\forall j < k: (M, s_j) \models \mu$





Найдем состояния M, в которых с вероятностной мерой ~p выполняется формула $\mu U \nu$

S^{yes} – множество состояний, в которых выполняется ν.

S^{no} - множество состояний, в которых не выполняются ни ν, ни μ, и тех, из которых недостижимы состояния, помеченные ν.

$S^{?}$ - множество состояний, в которых не выполняется ν, но выполняется μ.

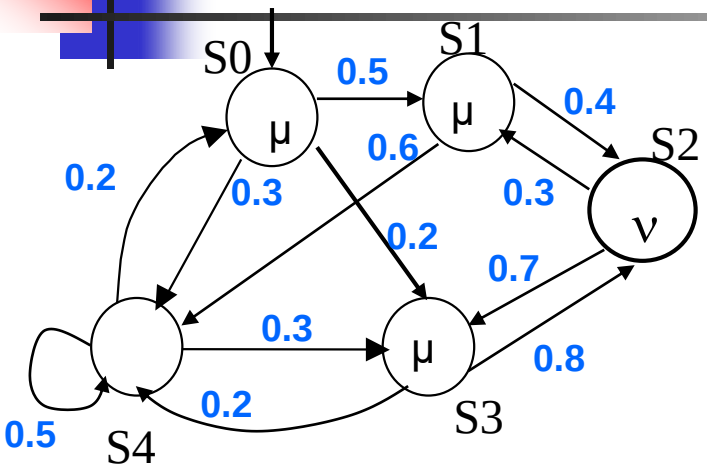
Определим: x_s – вероятность выполнения формулы $\mu U \nu$ в состоянии s.

$x_s = 1$ – если $s \in S^{yes}$

$x_s = 0$ – если $s \in S^{no}$

$x_s = \Sigma_{t \in S} P(s, t) * x_t$ – если $s \in S^{?}$, составляем линейное уравнение

Вычисление истинности PCTL формулы $P_{\sim p}(\mu U \nu)$



	s_0	s_1	s_2	s_3	s_4
s_0	--	0.5	–	0.2	0.3
s_1	--	--	0.4	--	0.6
s_2	--	0.3	--	0.7	–
s_3	--	--	0.8	--	0.2
s_4	0.2	--	--	0.3	0.5

μ и ν - формулы состояния

Матрица вероятностей переходов

Вычислим $P_{\geq 0.8}(\mu U \nu)$, т.е. в каких состояниях вероятность формулы Φ = μUν будет ≥0.8

x_s – это вероятность выполнения формулы Φ= μUν в состоянии s.

$x_2 \in S^{yes}$, $x_2=1$; $x_4 \in S^{no}$, $x_4=0$ (в s_2 выполнено ν, а в s_4 – не выполнены ни ν, ни μ).

Вероятности Φ в других состояниях нужно считать: $x_s = \Sigma\ Prob(s, s') * x_{s'}$

Система уравнений:

$x_0 = 0.5x_1+0.2x_3+0.3x_4$

$x_1 = 0.4x_2+0.6x_4$

$x_2 = 1$

$x_3 = 0.8x_2+0.2x_4$

$x_4 = 0$

⇒

Решаем:

$x_0 = 0.36$

$x_1 = 0.4$

$x_2 = 1$

$x_3 = 0.8$

$x_4 = 0$

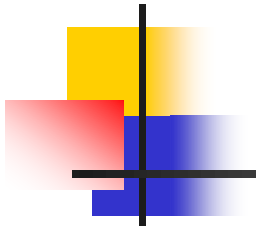
⇒

Формула Φ выполняется в тех состояниях, в которых вероятность выбора "нужного" пути удовл вероятностной мере (≥0.8)

⇒

0
0
1
1
0

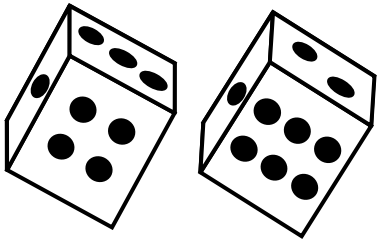
ИТАК, формула состояний $P_{\geq 0.8}$ μUν выполняется в состояниях s_2 и s_3



- ПРИМЕР применения PCTL: игра в кости

Пример: Моделирование игральной кости одной монетой

вместо



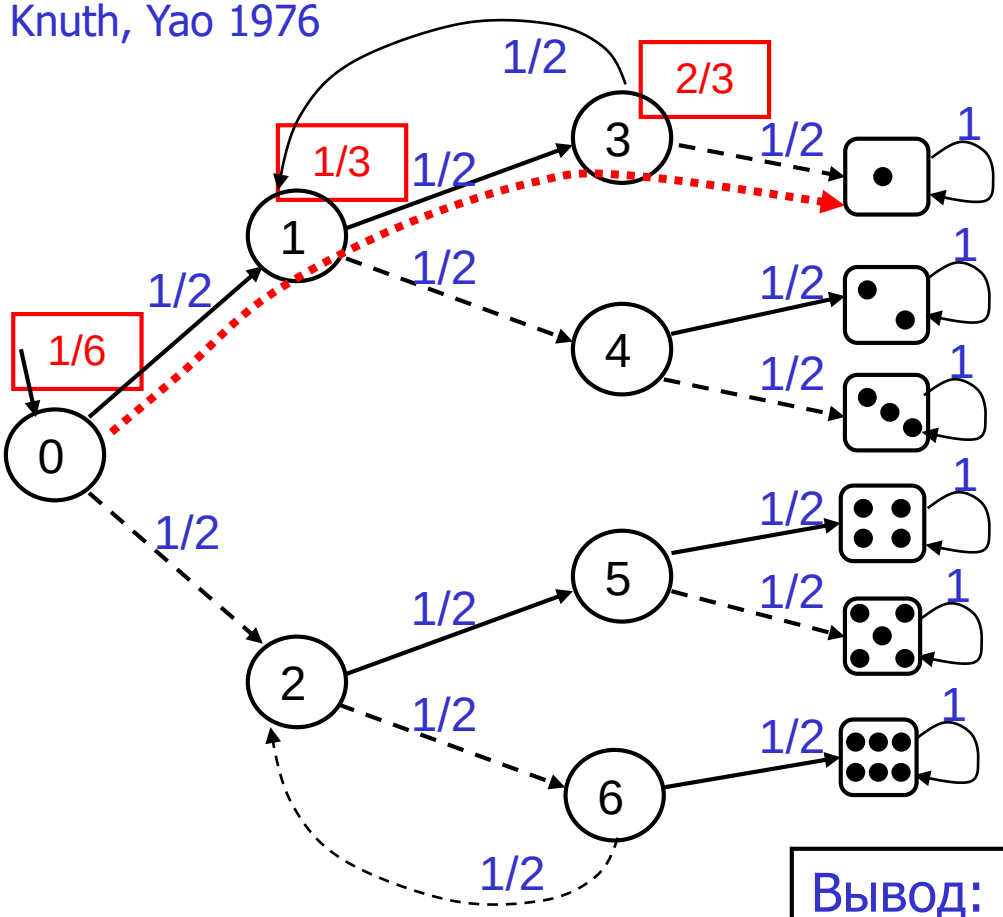
Проблема:

Смоделировать генератор случайных чисел, эквивалентный игральной кости, имея генератор случайных чисел, выдающий 0 и 1 с вероятностью 1/2

D. E. Knuth and A. C.-C. Yao. *The complexity of nonuniform random number generation* // In J. F. Traub, ed., Algorithms and Complexity: New Directions and Recent Results. Proc. of a Symposium, NY, 1976

Пример: Моделирование игральной кости одной монетой

Knuth, Yao 1976



орел

решка

подсчитаем:

Pr(F ” ⊙ ”) = Pr(*true* U “ ⊙ ”)

$x_0 = 1/2\ x_1 + 1/2\ x_2$

$x_1 = 1/2\ x_3 + 1/2\ x_4$

$x_2 = 0$ // Кость “ ⊙ ” недостижима

$x_3 = 1/2\ x1 + 1/2 \cdot 1$

$x4 = 0$ // Кость “ ⊙ ” недостижима

$x5 = 0$ // Кость “ ⊙ ” недостижима

$x6 = 0$ // Кость “ ⊙ ” недостижима

$x_0 = 1/6$ // Вероятность =1 в сост s_0

$x_1 = 1/3$

$x_3 = 2/3$

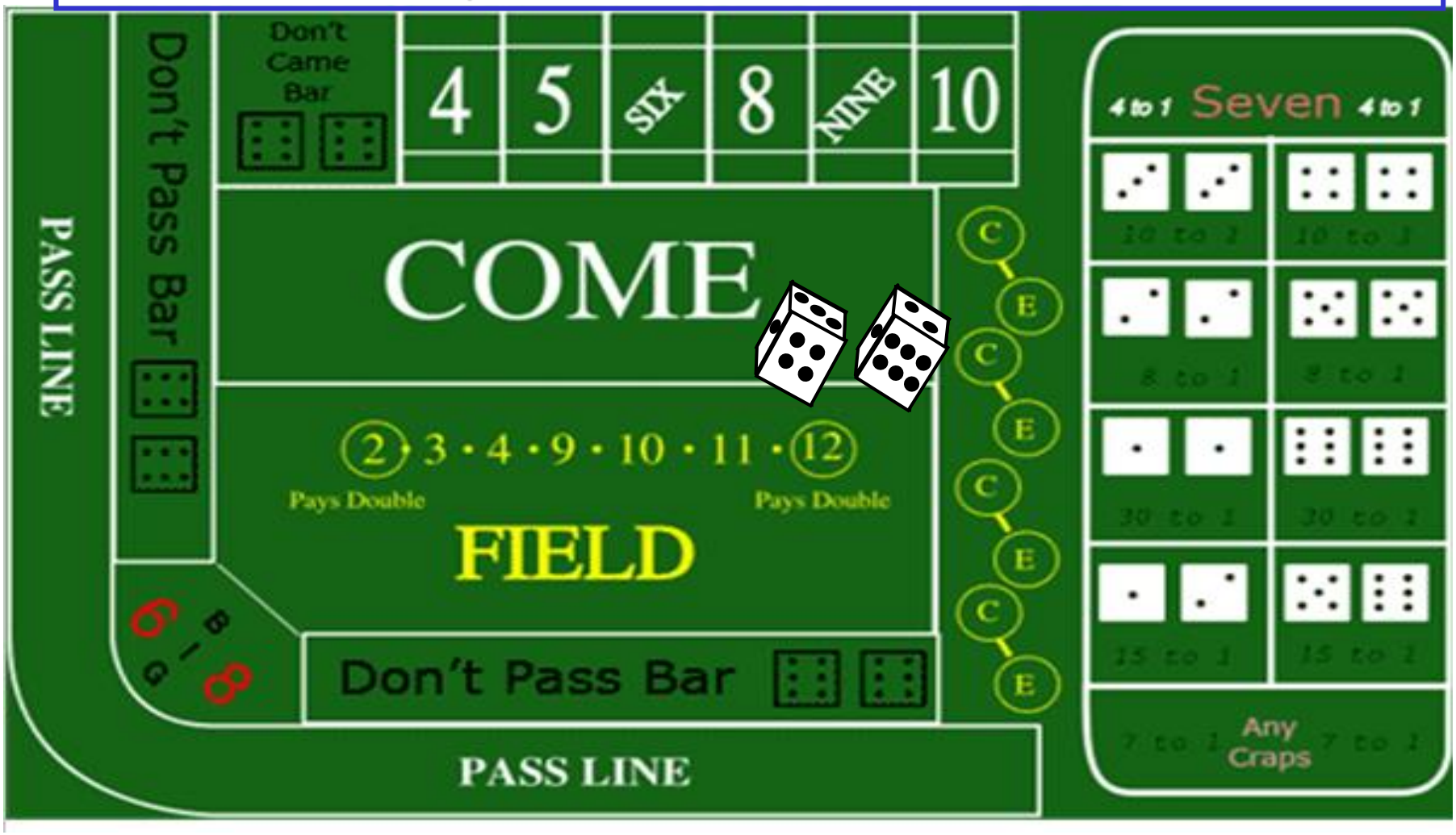
Вывод:
Вероятность достижения каждого из “терминальных” состояний = 1/6

Игра в кости

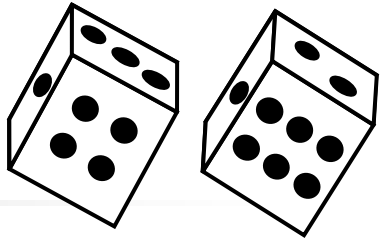
Игра в кости в центре порока - в Лас Вегасе, Невада (казино отеля Mandalay Bay)

Анализ игры в кости (Craps)

Игра - на стандартном поле, на котором определены места для ставок
Всего *около 40 различных ставок*, играть могут до 20 человек



Названия бросков в игре в Craps (по осям – выпадение очков на костях)



Names of Rolls in Craps

	1	2	3	4	5	6
1	Snake Eyes	Ace Deuce	Easy Four	Five (Fever Five)	Easy Six	Natural or Seven Out
2	Ace Deuce	Hard Four	Five (Fever Five)	Easy Six	Natural or Seven Out	Easy Eight
3	Easy Four	Five (Fever Five)	Hard Six	Natural or Seven Out	Easy Eight	Nine (Nina)
4	Five (Fever Five)	Easy Six	Natural or Seven Out	Hard Eight	Nine (Nina)	Easy Ten
5	Easy Six	Natural or Seven Out	Easy Eight	Nine (Nina)	Hard Ten	Yo (Yo-leven)
6	Natural or Seven Out	Easy Eight	Nine (Nina)	Easy Ten	Yo (Yo-leven)	Boxcars or Midnight

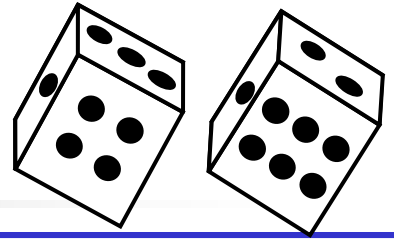
Число благоприятных исходов:

2 ⇔ 1,1 ⇒ 1/36
3 ⇔ 1,2; 2,1 ⇒ 2/36
4 ⇔ 1,3; 2,2; 3,1 ⇒ 3/36
5 ⇔ 1,4; 2,3; 3,2; 1,4 ⇒ 4/36
6 ⇔ 1,5; 2,4; 3,3; 4,2; 5,1 ⇒ 5/36
7 ⇔ 1,6; 2,5; 3,4; 4,3; 5,2;6,1 ⇒ 6/36
8 ⇔ 2,6; 3,5; 4,4; 5,3; 6,2 ⇒ 5/36
9 ⇔ 3,6; 4,5; 5,4; 6,3 ⇒ 4/36
10 ⇔ 4,6; 5,5; 6,4 ⇒ 3/36
11 ⇔ 5,6; 6,5 ⇒ 2/36
12 ⇔ 6,6 ⇒ 1/36

- Различных исходов бросания двух костей – 11.
- Каждый исход имеет свое устоявшееся имя. Имена напоминают крупье о конкретных ставках – выигрывают они или проигрывают
- Например, 2 –  (*snake eyes*) похожи на глаза змеи

  "Seven out" – выпадение 7 – важно для *конкретной* ставки "The Pass bet" и т.п.
- ***Каждый исход - результат разных наборов очков на костях, все они имеют разные вероятности.***

Анализ игры в кости (Craps). Ставка Snake Eyes



~ Сорок различных ставок. “Snake Eyes”, “Seven out”, …

Всего 36 вариантов

“Snake Eyes”: 1:1
Каждый бросок
выигрыш в игре
30:1

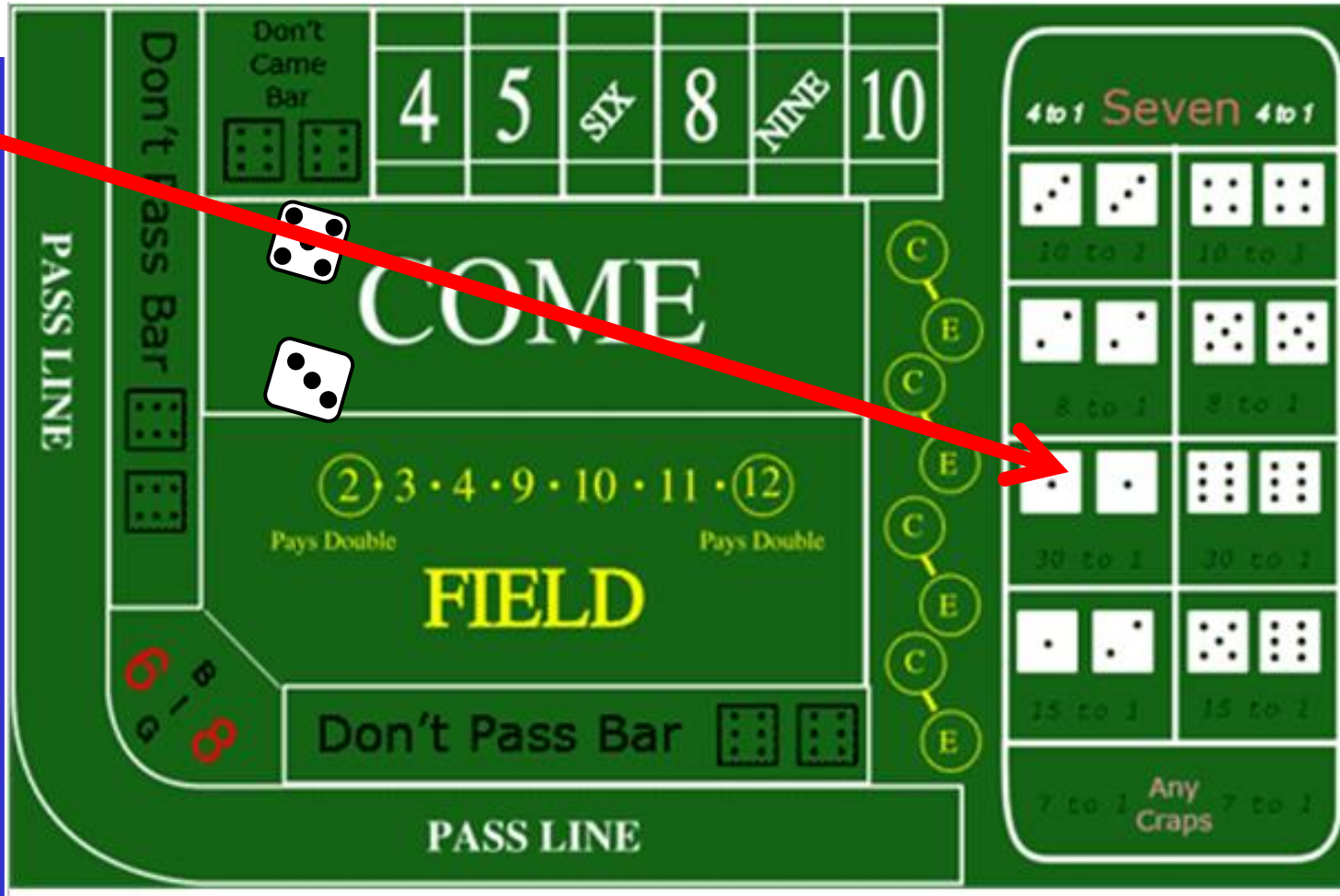
2 – выпадает в 1 комбинации из 36;

Казино имеет все остальные: 35 из 36

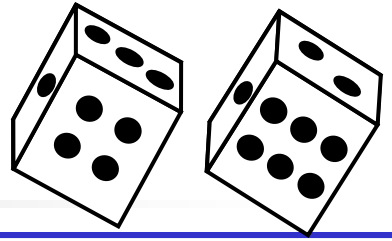
Общее число комбинаций 65:

игрок имеет - 30,
казино имеет - 35

Прибыль казино 5/65, т.е. 7.6%



Анализ игры в кости (Craps)



Seven: выигр. 7
Каждый бросок
выигрыш 4:1
7 – выпадает в 6 комбинациях из 36;

Казино имеет 30×1
Игрок: 4×6

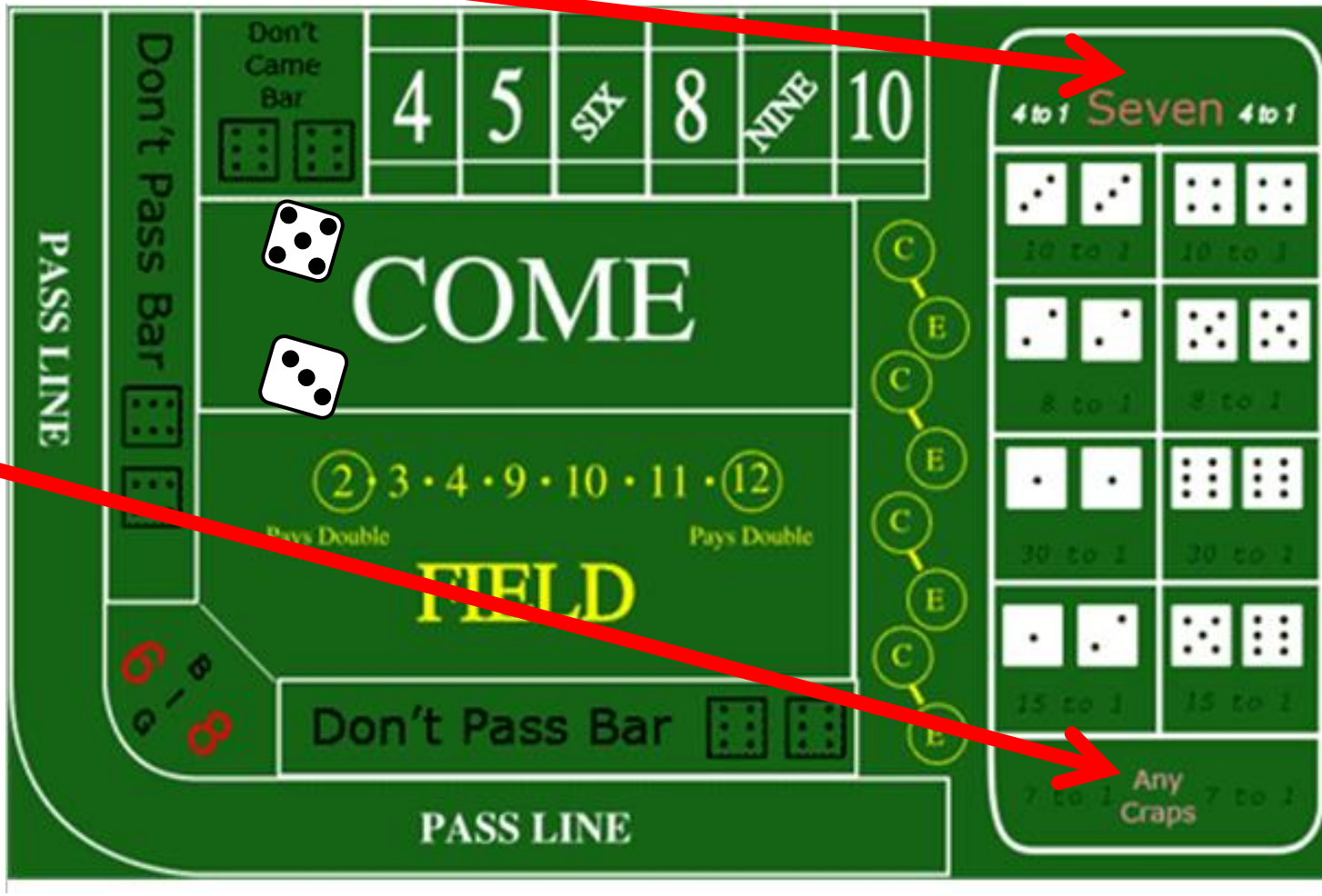
Прибыль казино 6/54, т.е. 11.1%

Any Craps: 2,3,12
Каждый бросок
выигрыш 7:1
Выигрыш в 4 комбинациях из 36;

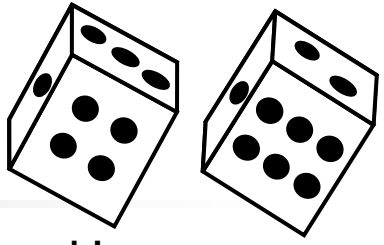
Казино имеет 32×1
Игрок имеет 7×4

Прибыль казино 4/60, т.е. 6.7%

All Seven; Any Craps, …



Craps: “*The Pass Bet*” – наиболее популярная ставка



Два этапа

I этап. *Come Out Roll*
Первый бросок

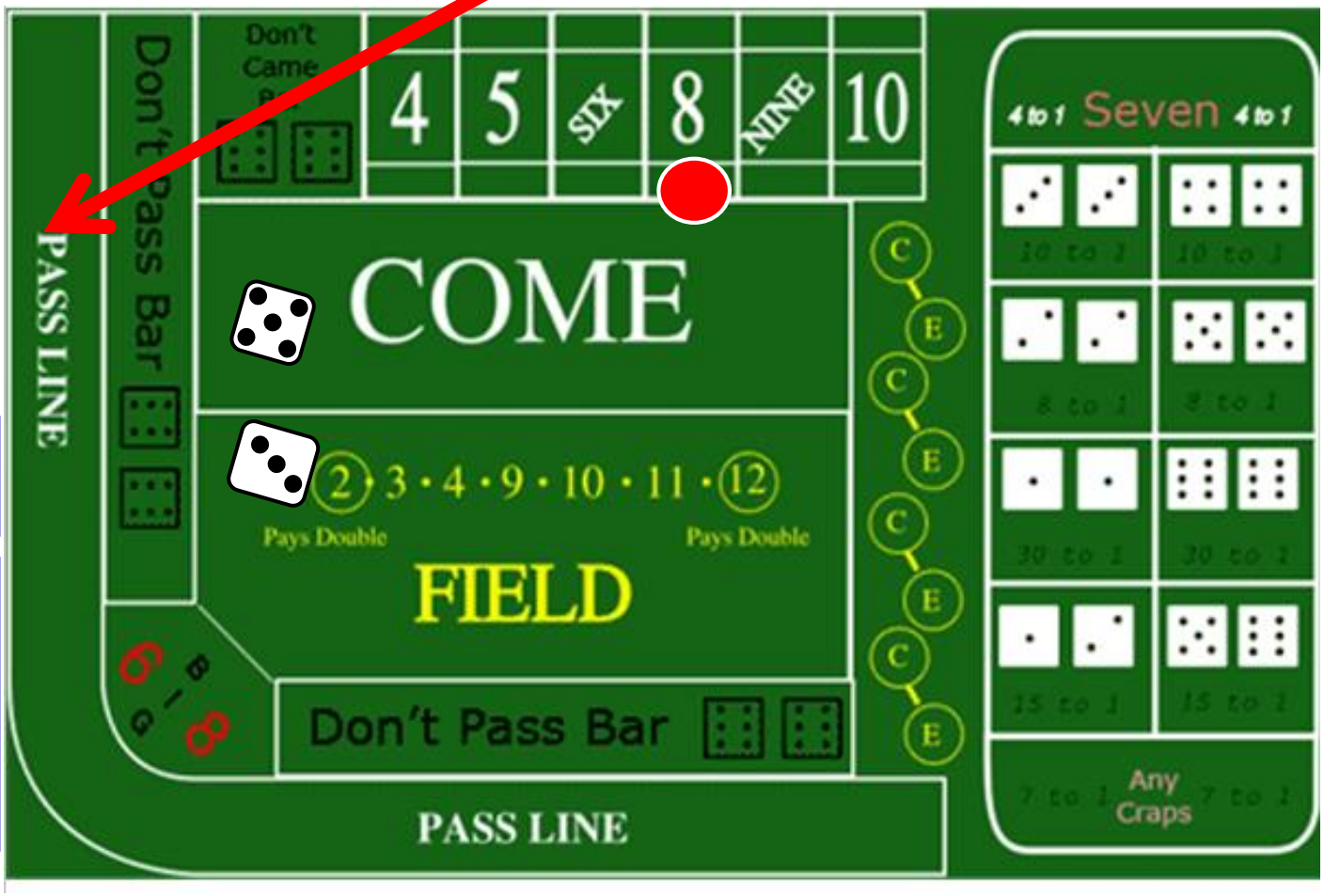
7,11 - *выигрыш.*
2,3,12 – проигрыш
4,5,6,8,9,10 =>Point,
и переход на II этап

Point запоминаем – ставится фишка

II этап. *Point Roll*
Набери Point

Нужно выбросить Point раньше, чем выпадет 7 (seven out)

Игрок ставит свои фишки на Pass Line



Как оценить вероятность выигрыша? Строим структуру Крипке и считаем вероятность того, что игрок из состояния start придет в состояние won

Игра в кости. Ставка “The Pass Bet”.
Анализ вероятностной структуры Крипке

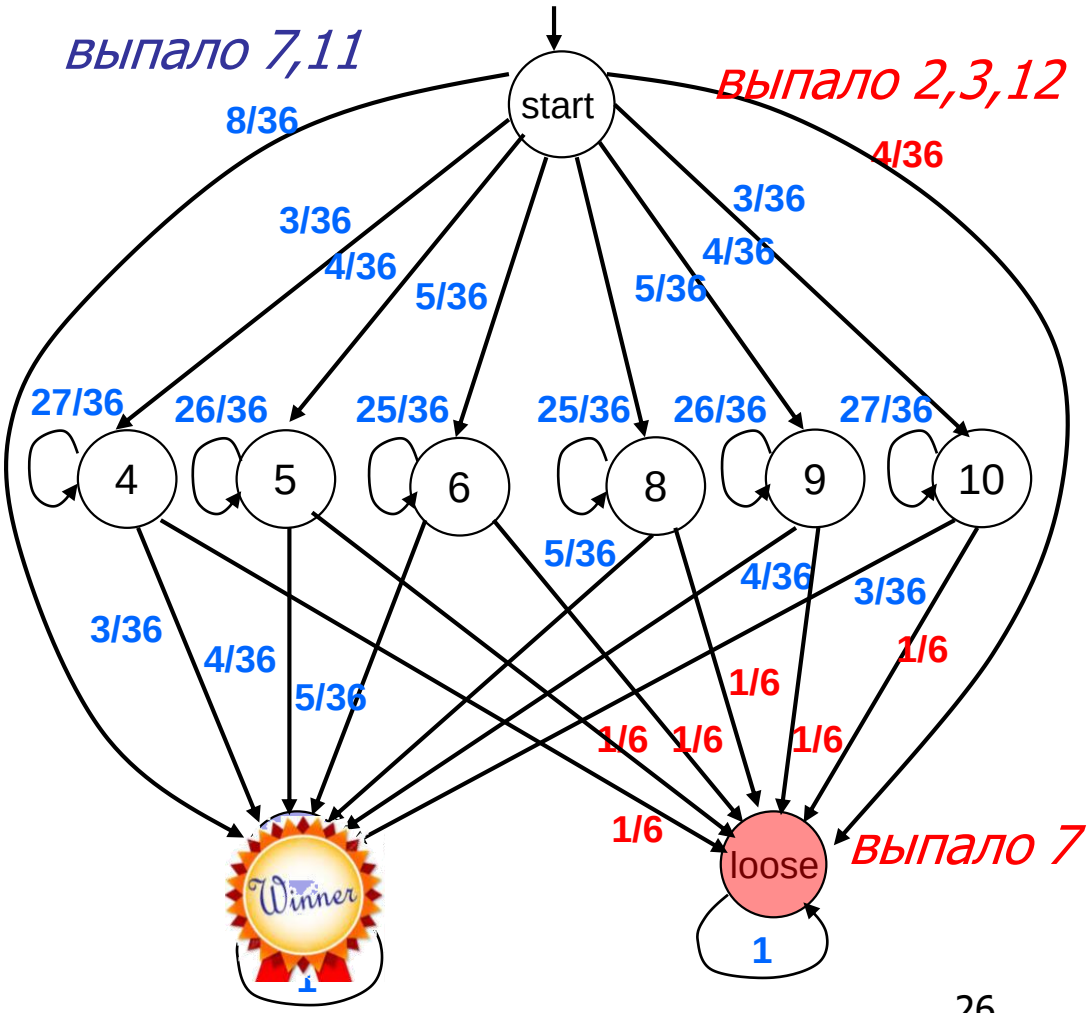
I этап. *Первый бросок*
7,11 - *выигрыш*.
2,3,12 – проигрыш
4,5,6,8,9,10 =Point, и на II этап

II этап. *Набери Point (seven out)*
Нужно выбросить Очко раньше 7

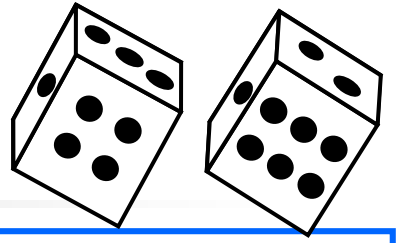
Число благоприятных исходов:

2 ⇔ 1,1 ⇒ 1/36
3 ⇔ 1,2; 2,1 ⇒ 2/36
4 ⇔ 1,3; 2,2; 3,1 ⇒ 3/36
5 ⇔ 1,4; 2,3; 3,2; 1,4 ⇒ 4/36
6 ⇔ 1,5; 2,4; 3,3; 4,2; 5,1 ⇒ 5/36
7 ⇔1,6; 2,5; 3,4; 4,3; 5,2; 6,1⇒ 6/36
8 ⇔ 2,6; 3,5; 4,4; 5,3; 6,2 ⇒ 5/36
9 ⇔ 3,6; 4,5; 5,4; 6,3 ⇒ 4/36
10 ⇔ 4,6; 5,5; 6,4 ⇒ 3/36
11 ⇔ 5,6; 6,5 ⇒ 2/36
12 ⇔ 6,6 ⇒ 1/36

Подсчитаем Pr(**F** ***Winner***)



Вероятность выигрыша ставки “The Pass Bet”



$Pr(F won) = Pr(true\ U\ won)$

Решение: $x_{start} = 0.493$ .... против 0.507

$x_{won} = 1$

$x_{loose} = 0$ – // из x_{loose} недостижимо x_{won}

$$x_{start} = 8/36\ x_{won} + 3/36\ x_4 + 4/36\ x_5 + 5/36\ x_6 + 3/36\ x_{10} + 4/36\ x_9 + 5/36\ x_8$$

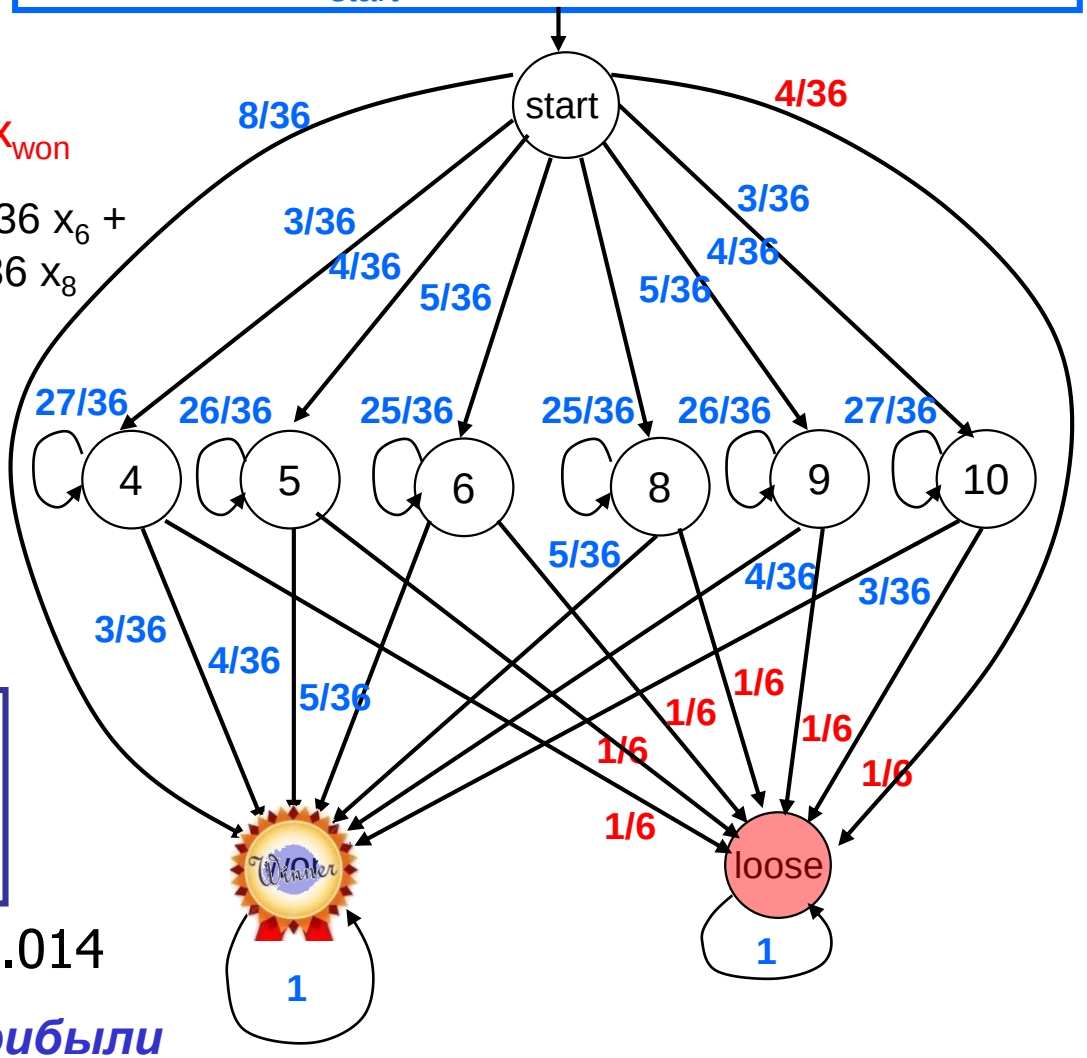
$$x_4 = 27/36\ x_4 + 3/36\ x_{won}$$
$$x_5 = 26/36\ x_5 + 4/36\ x_{won}$$
$$x_6 = 25/36\ x_6 + 5/36\ x_{won}$$
$$x_8 = 25/36\ x_8 + 5/36\ x_{won}$$
$$x_9 = 26/36\ x_9 + 4/36\ x_{won}$$
$$x_{10} = 27/36\ x_{10} + 3/36\ x_{won}$$

$x_4 = 1/3$
$x_5 = 2/5$
$x_6 = 5/11$
$x_8 = 5/11$
$x_9 = 2/5$
$x_{10} = 1/3$

$x_{start} = 0.493$

Казино 0.507

0.507-0.493 = 0.014



Казино имеет 1.4% прибыли

Craps: ставка"The Don't Pass Bet"

I этап.
Первый бросок

7,11 - *ПРОИГРЫШ*
3,12 – выигрыш
4,5,6,8,9,10 = Point,
и на II этап

Если выпала 2, то ставка возвращается игроку (ничья)

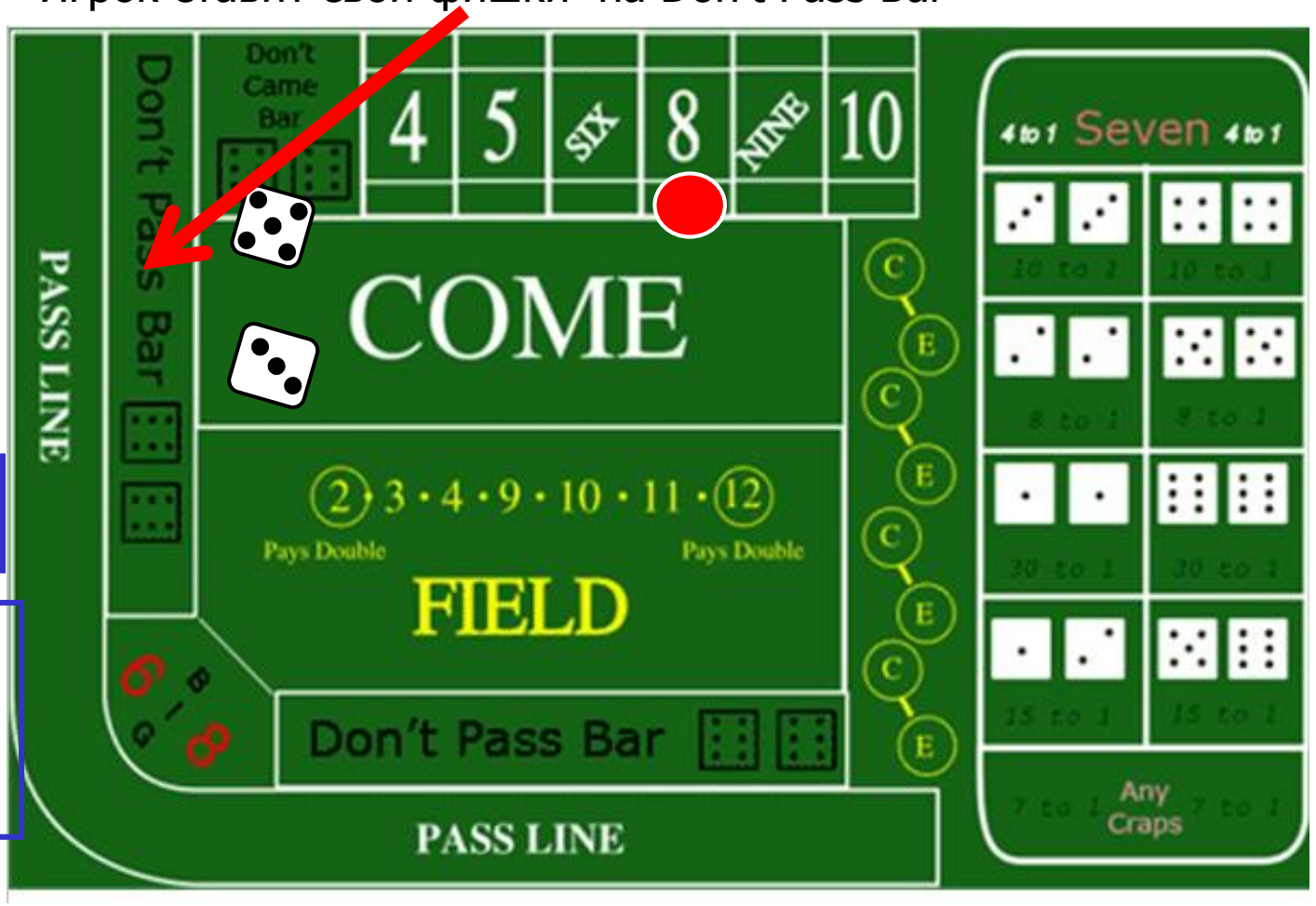
Point запоминается – ставится фишка

II этап.
НЕ *Набери Point*

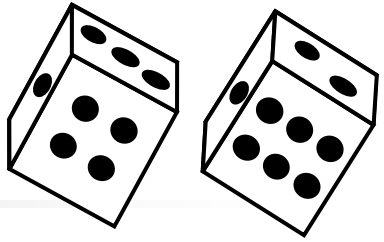
Нужно выбросить 7 раньше Point (seven out)

Ставка "The *Don't Pass Bet*"

Игрок ставит свои фишки на Don't Pass Bar



Craps: ставка"The Don't Pass Bet"



I этап.
Первый бросок

7,11 - *ПРОИГРЫШ*
3,12 – выигрыш
4,5,6,8,9,10 = Point,
и на II этап

Если выпала 2, то ставка возвращается игроку (ничья)

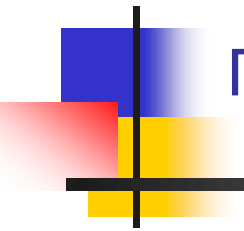
Point запоминается – ставится фишка

II этап.
НЕ *Набери Point*

Нужно выбросить 7 раньше Point (seven out)

Построение вероятностной структуры Крипке игры для *Don't Pass Bet* и вычисление на ней величины Pr(F won) - *самостоятельно*

Некоторые казино бесплатно награждают победителей футболками с напечатанным столом для ставок игры в Craps



Пример. Ненадежный канал

Model checking для вероятностной CTL

Формула PCTL–это формула состояния: $\varphi ::= q \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid P_{\sim p}(\alpha)$

где α -*формула пути:* $\alpha ::= X\,\varphi \mid \varphi_1\, U\, \varphi_2$

$P_{>0}(\alpha)$ соответствует квантору существования пути E - потому что с вероятностью >0 может быть выбран путь, на котором формула пути α выполняется. Но это соответствие **не абсолютное**.

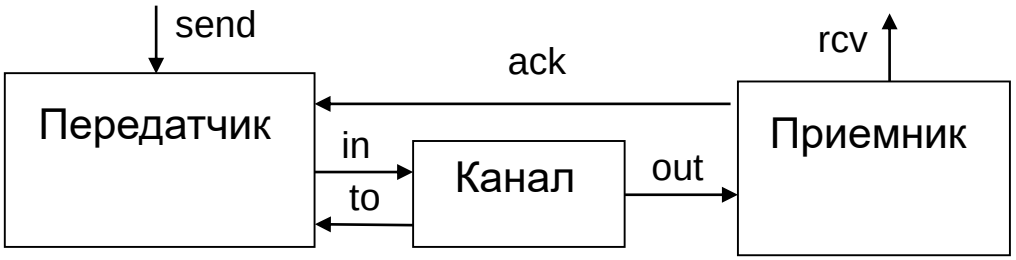
$P_{=1}(\alpha)$ соответствует универсальному квантору пути A: с единичной вероятностью будет выбран путь, на котором формула α выполняется (т.е. формула выполняется на любом пути). Но это соответствие не абсолютное.

Алгоритм верификации для формулы φ логики PCTL работает на основе алгоритма для CTL: индукцией по подформулам ψ_i формулы φ, определяя множества Sat(ψ_i) тех состояний, которые удовлетворяют формуле ψ_i - фактически, работает *"алгоритм разметки". **При вычислении алгоритма, все состояния, в которых истинны подформулы заданной формулы, должны быть уже определены.***

Пример: передача сообщений по ненадежному каналу

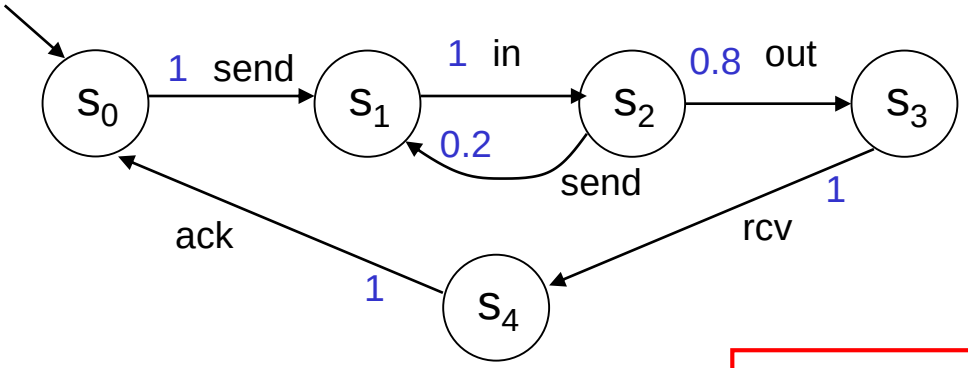
Вычислим
$Pr(F@s_4) = Pr(true\ U\ @s_4)$

Архитектура упрощенного протокола:



$$\begin{aligned} x_0 &= 1 \times x_1 \\ x_1 &= 1 \times x_2 \\ x_2 &= 0.2 \times x_1 + 0.8 \times x_3 \\ x_3 &= 1 \times x_4 \\ x_4 &= 1 \end{aligned}$$

$$\begin{aligned} x_0 &= 1 \\ x_1 &= 1 \\ x_2 &= 1 \\ x_3 &= 1 \\ x_4 &= 1 \end{aligned}$$



В s_0 выполняется формула $P_{=1}\, F@s_4$

Но $s_0 \not\models AF@s_4$

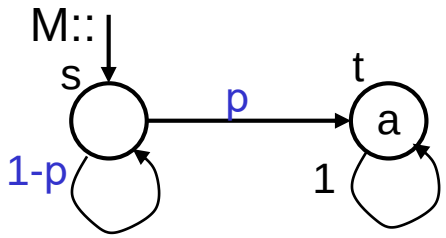
Для этого протокола A $F@s_4 \neq P_{=1} F\ @s_4$

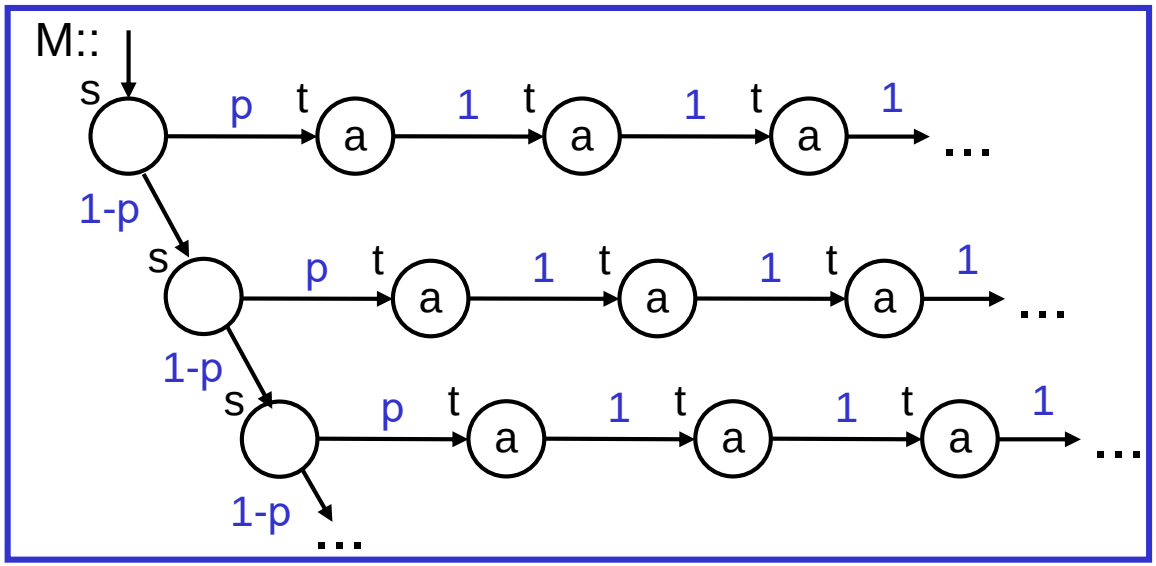
Существует путь, на котором из s_0 никогда не придем в s_3, но его вероятность 0

Кванторы пути A и E (сравнение с вероятностной CTL)

Оказывается, $A\varphi \neq P_{=1}\varphi$

Аналогично, $E\varphi \neq P_{>0}\varphi$





AFa НЕ выполняется на M

$P_{=1}Fa$ выполняется на M

EG¬a выполняется на M

$P_{>0}G\neg a$ НЕ выполняется на M

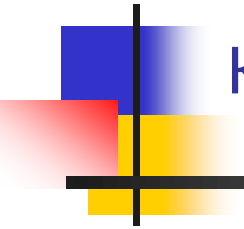
$x_s = (1-p)^* x_s + 1^* p_t$

$x_t = 1$

$x_s = 1$

$x_t = 1$

Путь σ = ssss..., его вероятность 0, но он ЕСТЬ!



Количественный анализ

Расширения Model checking: количественный анализ

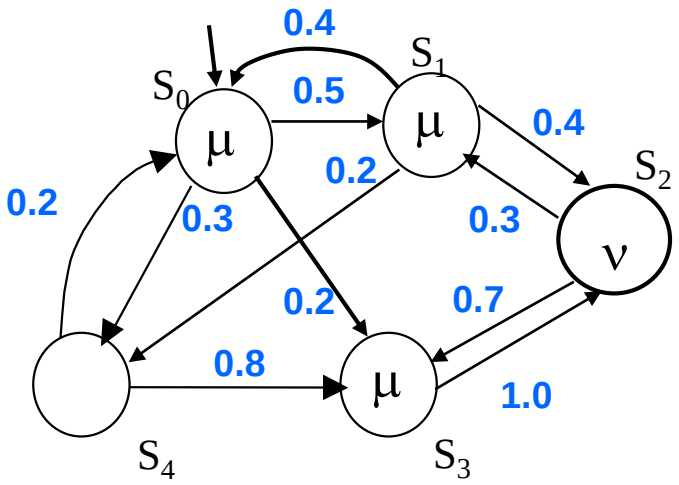
Этот подход позволяет подсчитать истинность или ложность того, что данная темпоральная формула станет истинной в течение **t** единиц времени (временных шагов) *с заданной вероятностью.*

В Uni. Birmingham разработана система ***PRISM***,
"*Probabilistic Symbolic Model Checker*"

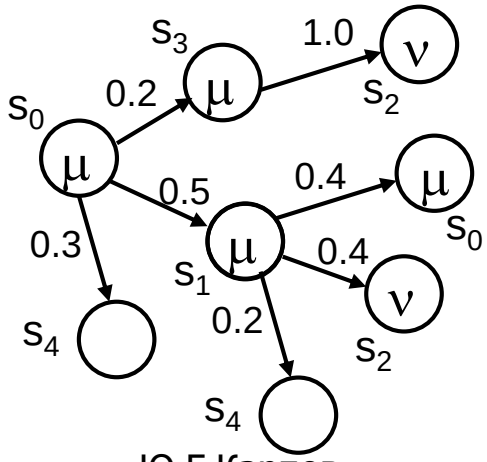
M. Kwiatkowska, D. Parker and H. Qu. Incremental Quantitative Verification for Markov Decision Processes. In *Proc. IEEE/IFIP Int. Conf. on Dependable Systems and Networks*, IEEE CS Press.June 2011

Учет времени (как числа шагов): PCTL формула $Pr_{\sim p}(\mu\, U^{\le n}\, \nu)$

Утверждение $Pr_{\sim p}(\mu\, U^{\le n}\, \nu)$*:* с вероятностной мерой ~p, не более, чем за n временных шагов, можем достичь состояния, в котором выполняется формула ν, *по пути, во всех состояниях которого* выполняется формула μ



P::	s_0	s_1	s_2	s_3	s_4
s_0	--	0.5	–	0.2	0.3
s_1	0.4	--	0.4	--	0.2
s_2	--	0.3	--	0.7	–
s_3	--	--	1.0	--	--
s_4	0.2	--	--	0.8	--

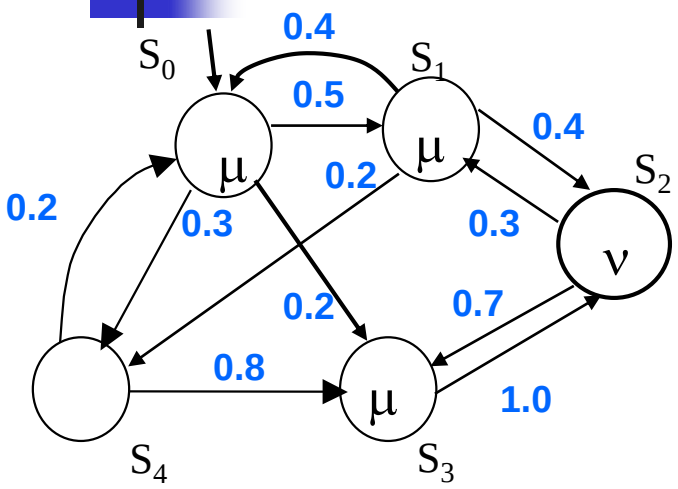


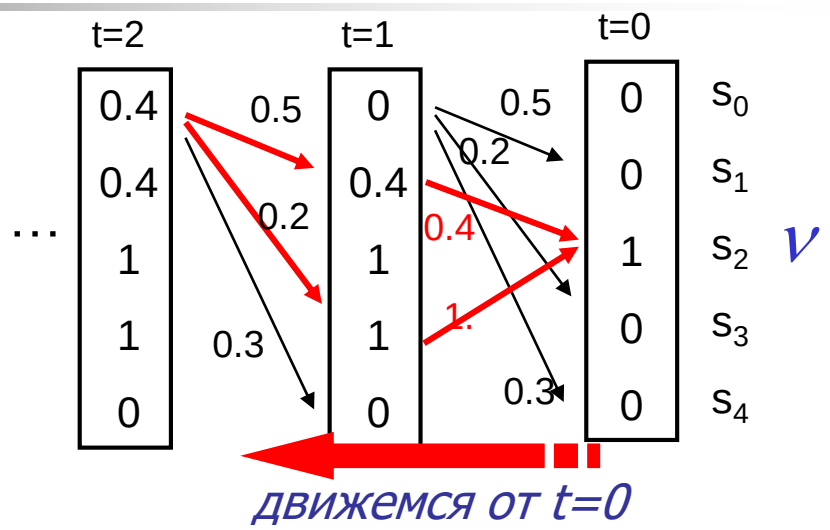
$\mu U^{\le n}\, \nu$: из текущего состояния формула $\mu U \nu$ выполнится не более, чем за n временных шагов

В s_0 вероятность выполнения $\mu\, U^{\le 2}\, \nu$ равна

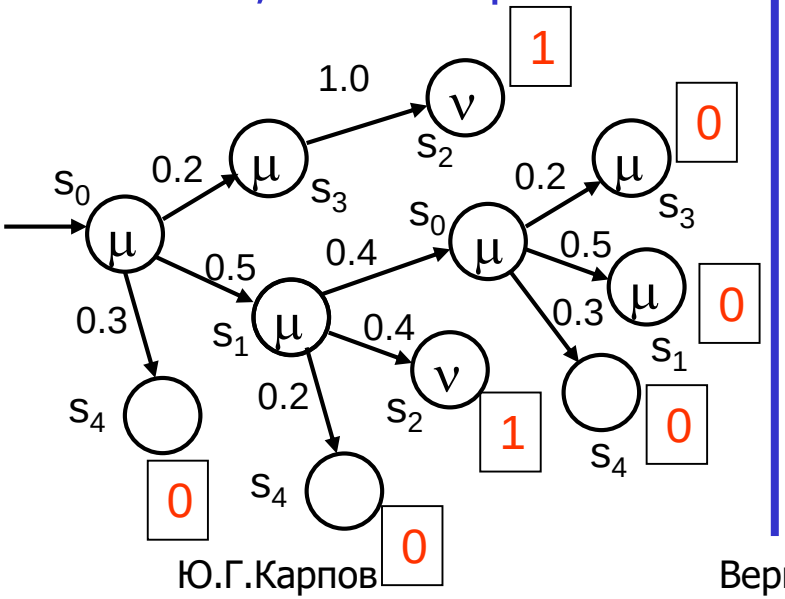
$Pr(\mu\, U^{\le 2}\, \nu) = 0.2 \times 1.0 + 0.5 \times 0.4 = 0.4$

Развертка вероятностной структуры по шагам времени $Pr_{\sim p}(\mu\, U^{\leq n}\, \nu)$





Не более, чем за три шага:



Если уже достигли состояния, в котором выполняется ν, останавливаемся с вероятностью 1

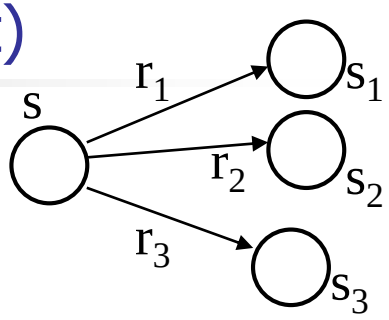
Если достигли состояния, в котором не выполняются ни μ, ни ν, останавливаемся с вероятностью 0

За время $t \leq 0$ - в s_2 с вер 1

За время $t \leq 1$ - в s_2 и s3 с вер 1, в s_1 с вер 0.4

За время $t \leq 2$ - в s2 и s3 с вер 1, в s_0 и s_1 с вер 0.5

Идея алгоритма вычисления Pr(s, μUν, t)



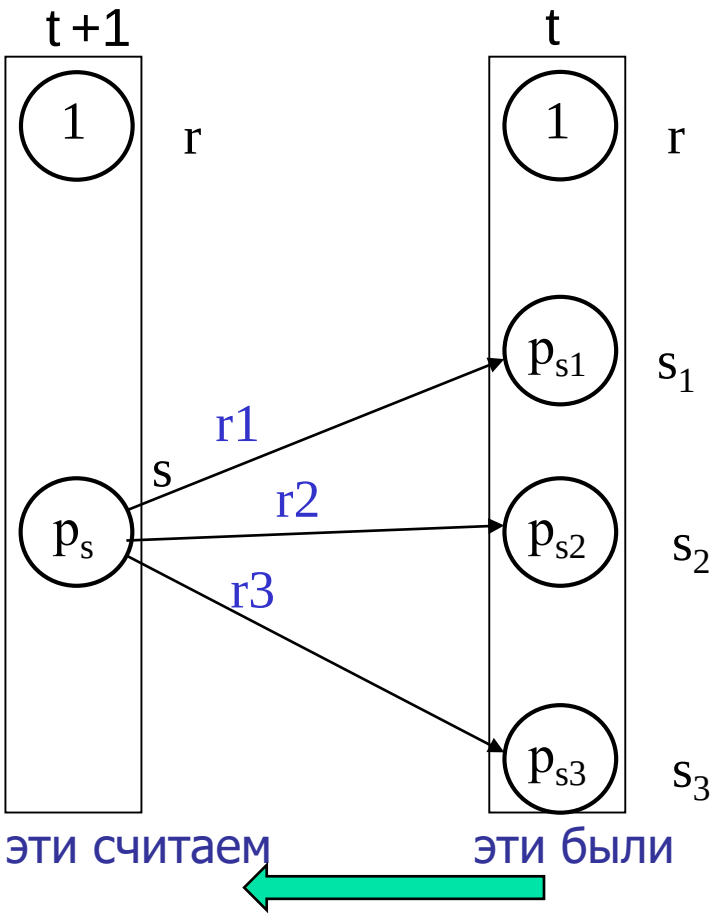
Pr(s, μUν, t) – Вероятность того, что из s не более, чем за t временных шагов придем в состояние, в котором выполняется ν, через состояния, в которых выполняется μ

Строим начальный вектор вероятностей p_s выполнения в состояниях s_0, s_1, … формулы μUν за не более, чем 0 шагов. В тех состояниях s, в которых выполняется ν, $p_s=1$, в остальных $p_s=0$

Вероятность p_s выполнения формулы μUν в состоянии s за t + 1 шагов равна сумме произведений вероятностей r_k переходов из s в состояния s_k на вероятности p_{sk} выполнения этой формулы в состояниях s_k за t шагов

$$p^{t+1}_s = r1 \times p^t_{s1} + r2 \times p^t_{s2} + r3 \times p^t_{s3}$$

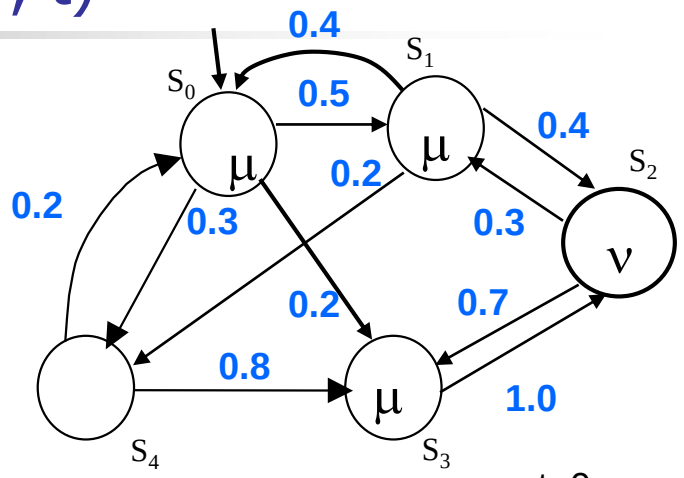
Если за t шагов вероятность выполнения формулы μUν в состоянии s равна 1, то за t+1 шагов эта вероятность в состоянии s также равна 1

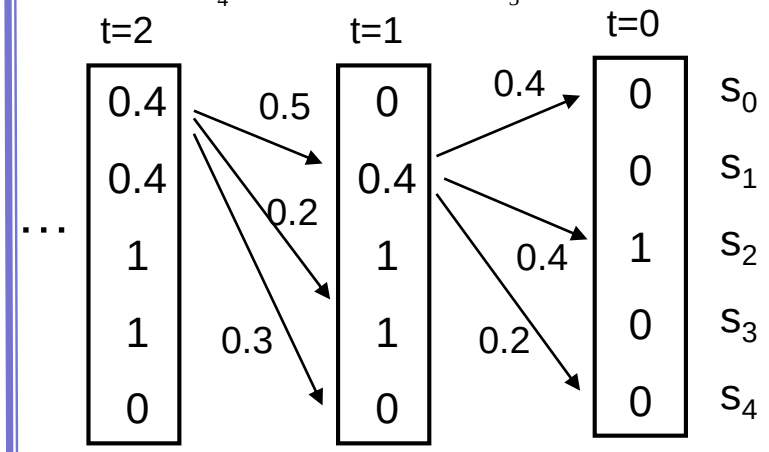


Алгоритм вычисления Pr(s, μUν, t)

```
begin
for all s∈S do
  if ν∈L(s) then Pr(s, μUν, 0) := 1   // ν   выполняется в s
        else Pr(s, μUν, 0) = 0;        // ν не выполняется в s
od;
for i=1 to t do
  for all s∈S do
    if ν∈L(s) then Pr(s, μUν, i) := 1; // ν выполняется в s
      else begin
        Pr(s, μUν, i) = 0;             // ν не выполняется в s
        if μ∈L(s) then                 // если μ выполняется в s
        for all s'∈S do
        Pr(s, μUν, i) = Pr(s, μUν, i-1) + Pr(s, s') × Pr(s', μUν, i-1)
         od
      end
  od
od
end
```

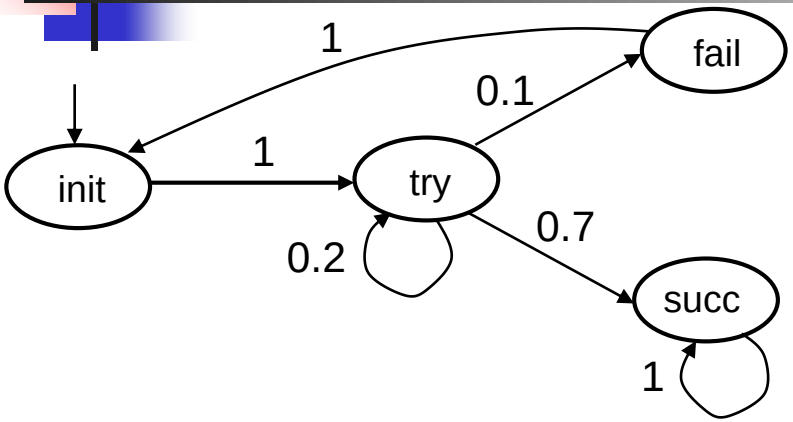
Pr(s, μUν, t) – Вероятность того, что из s не более, чем за t временных шагов придем в состояние, в котором выполняется ν, через состояния, в которых выполняется μ





Pr(s_0, μ**U**ν, 0) = 0.
Pr(s_0, μ**U**ν, 1) = 0.
Pr(s_0, μ**U**ν, 2) = 0.4

Пример: упрощенная модель протокола



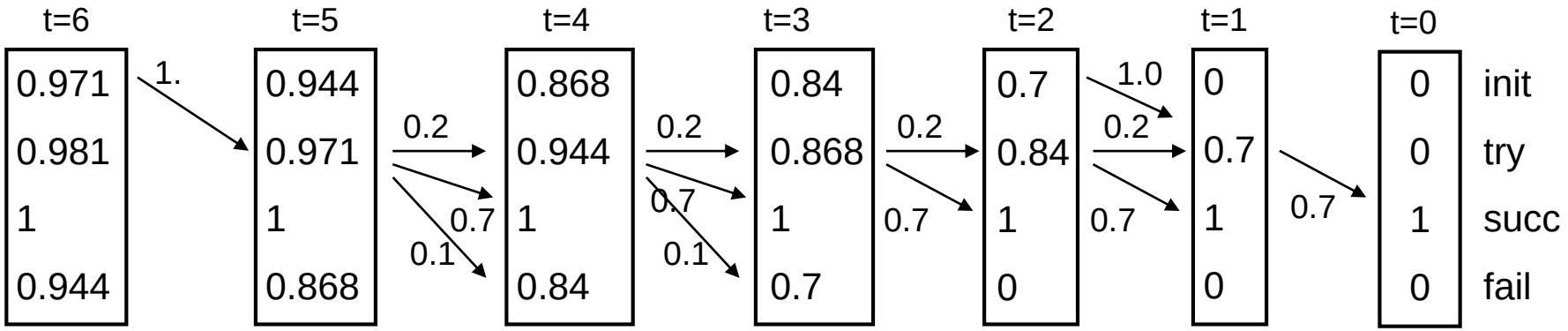
Проверим выполнение утверждения:

"С вероятностью, не меньшей 0.95, сообщение будет успешно доставлено в течение 6 единиц времени"

Формально: $init \models P_{\geq 0.95}(F^{\leq 6}\ succ)$

Вычислим вероятность выполнения формулы: $init \models F^{\leq 6}\ succ$ ***"вероятность того, что сообщение будет успешно доставлено в течение не более, чем 6 единиц времени",*** т.е. найдем Pr (init, true U succ, 6)

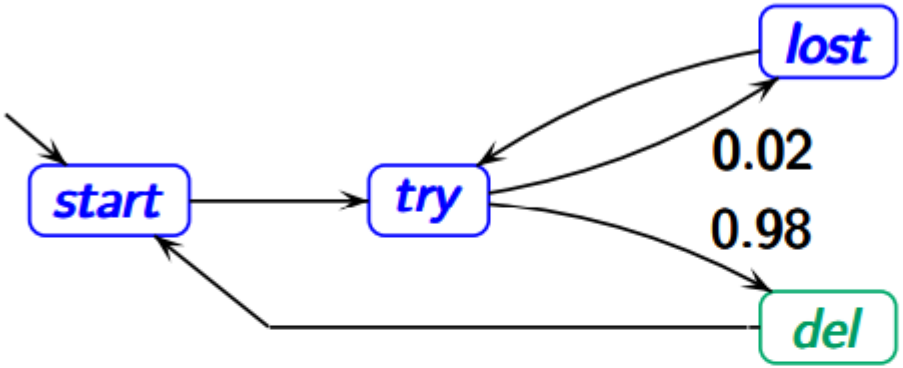
Подсчитаем Pr (s, true U succ, t) для всех состояний структуры и всех t от t=0 до t=6



Эта вероятность оказалась 0.971. Следовательно, $init \models P_{\geq 0.95}(F^{\leq 6}\ succ)$

Пример из руководства по вероятностной логике

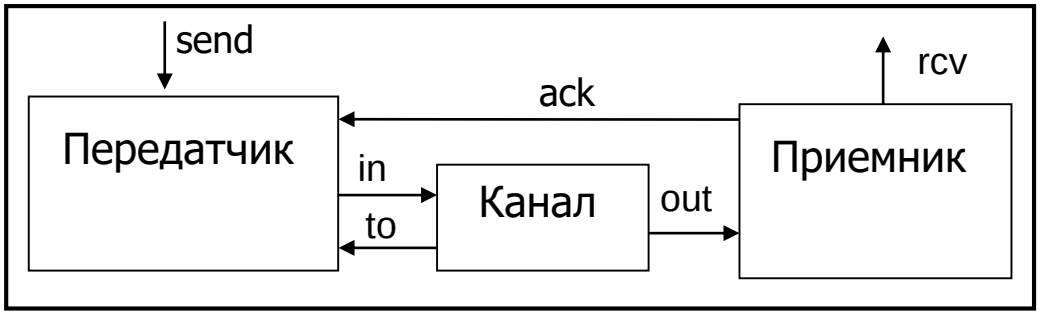
Example: Markov chain

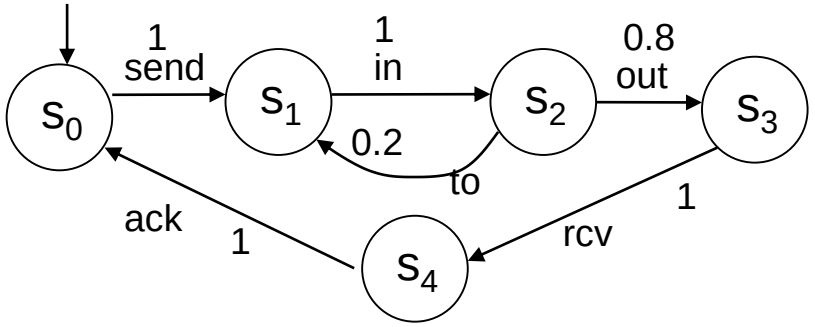


probability for delivering the message within **5** steps:

Пример оценки “мягкого дедлайна”

Упрощенный протокол “альтернирующего бита”





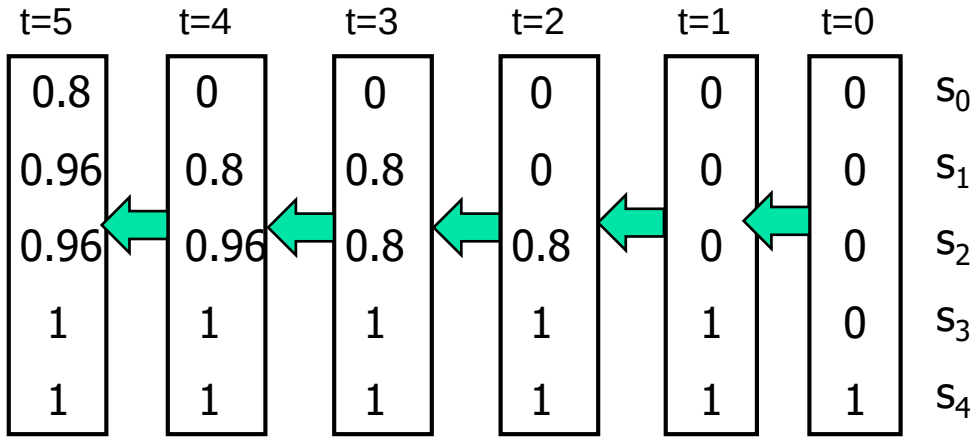
Проверим свойство:

$$\Phi = \mathbf{AG}(at\_s_0 \Rightarrow \mathbf{P}_{\geq 0.9}\ \mathbf{F}^{\leq 5}\ at\_s_4) =$$
$$\neg\mathbf{EF}\ \neg(at\_s_0 \Rightarrow \mathbf{P}_{\geq 0.9}\ \mathbf{F}^{\leq 5}\ at\_s_4)$$

Синтаксический анализ формулы Φ:

$f_1 = at\_s_0$
$f_2 = at\_s_4$
$f_3 = \mathbf{P}_{\geq 0.9}\ \mathbf{F}^{\leq 5}\ f_2$
$f_4 = f_1 \Rightarrow f_3$
$f_5 = \neg f_4$
$f_6 = \mathbf{EF}\ f_5$
$f_7 = \neg f_6 = \Phi$

$f_1 : \{ s_0 \}$
$f_2 : \{ s_4 \}$
$f_3 : \{ s_1, s_2, s_3, s_4 \}$
$f_4 : \{ s_1, s_2, s_3, s_4 \}$
$f_5 : \{ s_0 \}$
$f_6 : \{s_0, s_1, s_2, s_3, s_4\}$
$f_7 : \{\ \}$



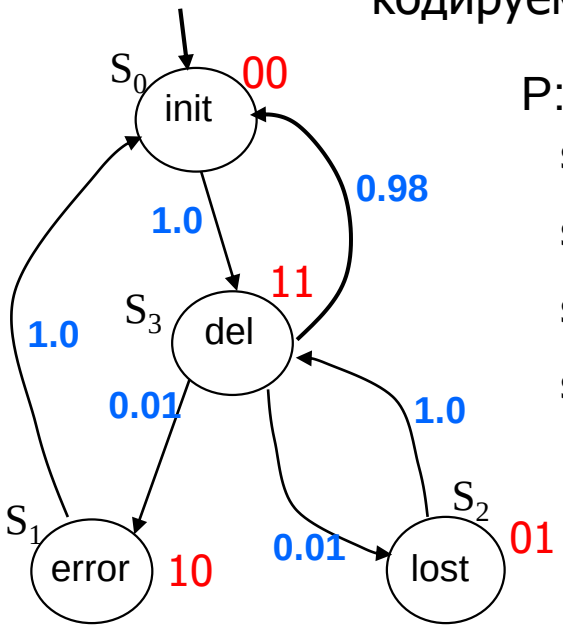
Свойство Φ НЕ выполняется для этого протокола

Символьный алгоритм Model checking для PTCL

- Вычисление вероятности выполнения формулы $\varphi U \psi$ основано на решении систем линейных алгебраических уравнений, заданных матрицей размера $n \times n$. Основная трудность с вероятностным model checking – интеграция пакетов линейной алгебры с существующими верификаторами. Эта трудность следует из необходимости явного представления пространства состояний системы.
- *Альтернатива: использование символьных вычислений, но использовать не BDD, а Multi-Terminal Binary Decision Diagrams (MTBDDs). Это обобщение BDD – не только 0 и 1 в качестве пометок листьев, а любые вещественные числа*
- Как и в случае применения BDD, точно оценить временную сложность алгоритмов model checking для MTBDDs трудно, но эксперименты показывают, что выигрыш существенный.

C.Baier, et.al. *Symbolic model checking for probabilistic processes.* In *ICALP '97*, pp. 430–440

Пример: символьный алгоритм вероятностного Model checking



кодируем состояния x_1x_2

P::

	s_0	s_1	s_2	s_3
s_0	0	0	0	1
s_1	0	0	0	1
s_2	1	0	0	0
s_3	.98	.01	.01	0

$F(x_1x'_1x_2x'_2)=$

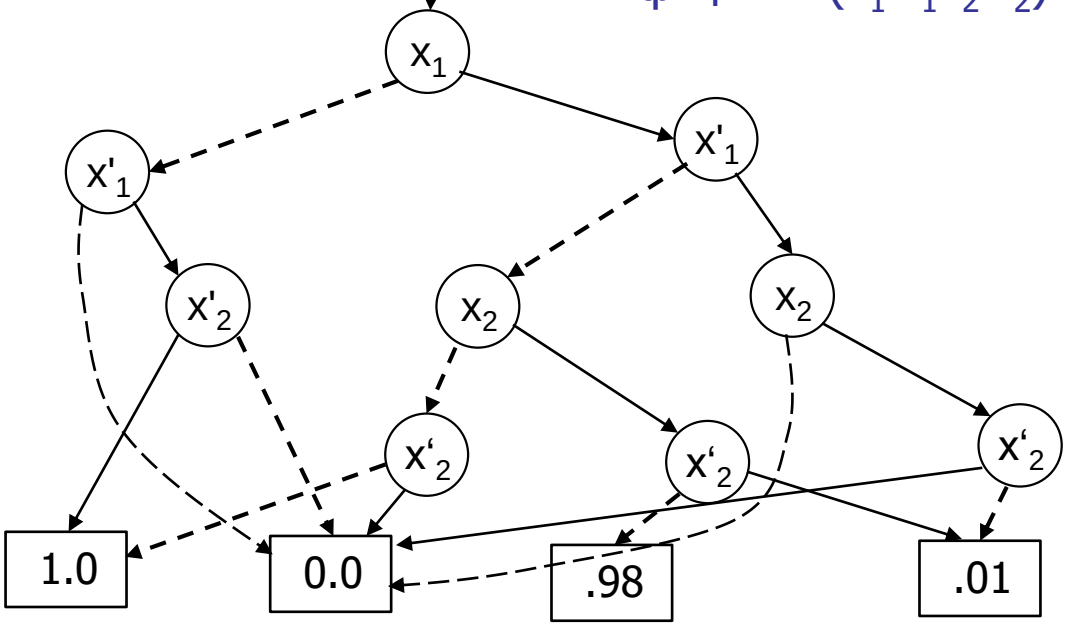
1, if $x_1x'_1x_2x'_2 \in \{0101, 0111, 1000\}$

0.01, if $x_1x'_1x_2x'_2 \in \{1011, 1110\}$

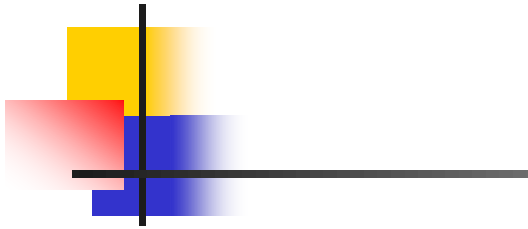
0.98, if $x_1x'_1x_2x'_2 \in \{1010\}$

0.0, otherwise

MTBDD ф-ции $F(x_1x'_1x_2x'_2)$



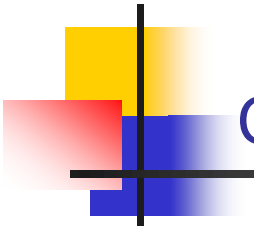
Пример курсового проекта:
Разработать самостоятельно алгоритмы оценки выполнимости формул PCTL с помощью символьных представлений



Символьная верификация для PCTL

Система верификации PRISM

M. Kwiatkowska, et.al. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. 23rd Int. Conf. Computer Aided Verification (CAV'11)*, vol. 6806 of *LNCS*, pages 585–591. Springer, 2011.



Система верификации PRISM

Разработана группой проф. Martha Kwiatkowska в Uni Birmingem в2001

- Позволяет выполнить анализ систем, включающих вероятность и время
- Распространяется свободно для исследований и обучения
- Десятки тысяч скачиваний
- Сотни статей, исследующих проблемы с помощью системы Prism
- Основана на символьных алгоритмах, BDD, MTBDD, алгоритмах анализа Марковских цепей
- Сайт www.cs.bham.ac.uk/~dxp/prism/ - методические материалы, алгоритмы, ...

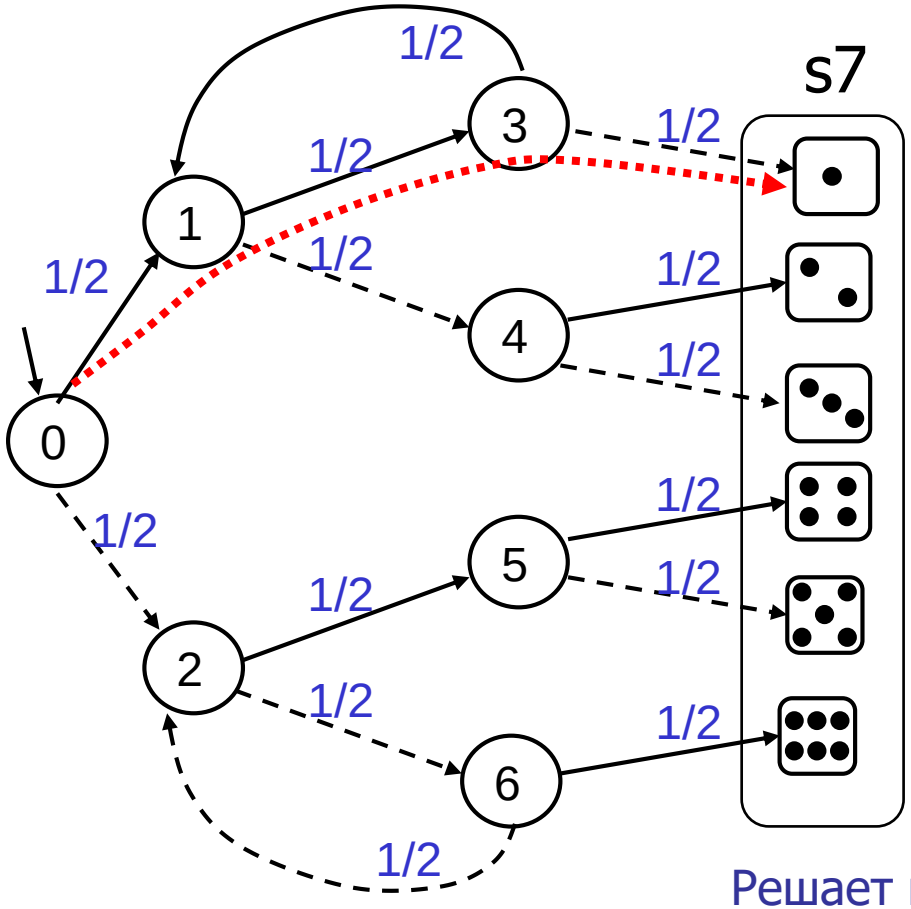
Система верификации PRISM (2)

- Функциональность
  - Реализован model checking для стохастических систем, Probabilistic temporal logic
  - Используются модели:
    - дискретные и непрерывные цепи Маркова,
    - Марковские решающие процессы
- Высокоуровневый язык представления моделей
  - Спецификации свойств вида:
    - P<0.01 [true U ≤100 error] – "*вероятность того, что система достигнет состояния error в течение не более 100 временных единиц, меньше, чем 0.01*"
    - P = ? [true U ≤50 terminate] – "*какова вероятность того, что система достигнет состояния terminate в течение не более 50 временных единиц?*"

Соотношения между формулами, используемые в PRISM

- $s \models P_{\geq p}(\alpha) \equiv s \models \neg P_{<1-p}(\alpha)$
- $s \models P_{>p}(\alpha) \equiv s \models \neg P_{\leq 1-p}(\alpha)$
- $F\alpha \equiv true\ U\ \alpha$
- $F^{\leq n}\alpha \equiv true\ U^{\leq n}\ \alpha$
- $G\alpha \equiv \neg F \neg \alpha$
- $P_{\leq p}(G\ \alpha) \equiv P_{\geq 1-p}(F\ \neg\alpha)$
- $P_{]p,q]}(G^{\leq n}\ \alpha) \equiv P_{[1-p,1-q[}(F^{\leq n}\ \neg\alpha)$

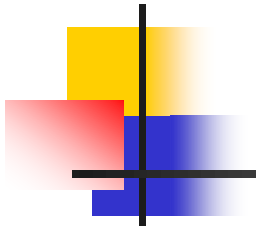
Пример. Моделирование игральной кости одной монетой: программа на входном языке системы PRISM




```
module die
// Discrete-time Markov chain DTMC
    s : [0..7] init 0;     // local state
    d : [1..6] init 0;     // value of the die

  [] s=0 -> 0.5 : (s'=1) + 0.5 : (s'=2);
  [] s=1 -> 0.5 : (s'=3) + 0.5 : (s'=4);
  [] s=2 -> 0.5 : (s'=5) + 0.5 : (s'=6);
  [] s=3 -> 0.5 : (s'=1) + 0.5 : (s'=7) & (d'=1);
  [] s=4 -> 0.5 : (s'=7)  & (d'=2) +
            0.5 : (s'=7)  & (d'=3);
  [] s=5 -> 0.5 : (s'=7)   & (d'=4) +
            0.5 : (s'=7)  &  (d'=5);
  [] s=6 -> 0.5 : (s'=2) +
            0.5 :(s'=7) & (d'=6);
  [] s=7 -> (s'=7);
endmodule
```

Решает нужные уравнения подсчета вероятностей
Знак [] – 'защита' (языка Дейкстры)



Спасибо за внимание