# Демонстрация
## на събитийни модели

Отчетна сесия на секция
„Алгебра и логика“

20 декември 2019

Димитър Добрев

d@dobrev.com

Институт по математика и информатика
Българска академия на науките

## Каква е целта на този доклад?

Искаме да покажем, че със събитийни модели можем да опишем много сложни светове по много прост начин.

## Защо ни трябва да описваме сложните светове по прост начин?

Целта е да създадем програма, която в произволен свят би се справила добре. Тоест, програма, която би разбрала произволен свят. Тази програма се нарича Изкуствен Интелект.

## Как работи програмата ИИ?

ИИ се опитва да разбере света като търси описание на света. Много важен е формата на това описание. Ако описанието е написано на някакъв формален език, то много важно е кой формален език сме избрали.

## Можем ли да опишем света чрез Машина на Тюринг?

Да, можем. Въпросът, който ИИ си задава е „Какво да правя?“ Отговорът е стратегия. Тази стратегия е изчислима, защото неизчислимите стратегии не са отговор. Тоест, стратегията може да се представи чрез компютърна програма или чрез Машина на Тюринг (което е същото).

## Можем ли да намерим Машината на Тюринг, която описва света?

Да, можем, ако имаме безкрайно бърз компютър, безкрайно дълго време за обучение и ако в този свят няма фатални грешки (т.е. ако агентът не може да допусне фатална грешка).

## Има ли по-добър начин от това да търсим Машина на Тюринг?

Да, ИИ може да не си задава директно въпроса „Какво да правя?“, а да започне първо с въпроса „Какво става?“. Когато разбере какво става, ще знае и какво да прави. Тоест, предлагаме да не се търси директно стратегия, а първо да се намери модел на света.

## Какъв модел ще търсим?

Може да търсим Markov decision process. Става дума за Partial Observability, защото случаят на Full Observability не е интересен.

## Можем ли да намерим модел под формата на MDP?

Да, всеки свят може да се представи като MDP. Може този модел да е безкраен, но ние ще предположим, че моделът е краен и в него няма фатални грешки (т.е., че между всеки две състояния има път).

При тези предположения ние можем ефективно да намерим MDP модела на света, но пак ще ни трябва безкрайно мощен компютър и безкрайно дълго време за обучение.

Това, че ни е нужен безкрайно мощен компютър не звучи чак толкова страшно. Можем да предположим, че един ден ще има достатъчно мощен компютър, който да намери съответния модел. По-лошо звучи изискването за безкрайно дълго време за обучение. Ако ИИ учи безкрайно бавно, то нашето ИИ ще е бавно развиващ се интелект.

## Можем ли да предложим модел намирането на който не изисква безкрайно мощен компютър и безкрайно дълго време за обучение?

Да това ще са събитийните модели. Ще направи демонстрация като покажем как правилата на играта шах могат да се опишат чрез тези модели.

## Може ли играта шах да бъде описана чрез Машина на Тюринг?

Да, има програми играещи шах, но автоматичното намиране на такава програма е на практика невъзможно.

## Може ли играта шах да бъде описана чрез MDP?

Да, съществува краен MDP, който описва шаха, но неговите състояния са колкото позициите в играта. Тоест, и този модел не може да бъде намерен.

## Как ще изглежда светът на играта шах?

Първо това ще е свят, в който агента не вижда всичко (Partial Observability, случаят на Full Observability не е интересен).

В този свят агента ще вижда само едно от квадратчетата на таблото. Това няма да е проблем, защото агента ще може да движи прозореца (видимото квадратче) и по този начин да огледа цялото табло.

Ще демонстрираме следната програма:

AI Unravels the Chess, step=21 (Pause)

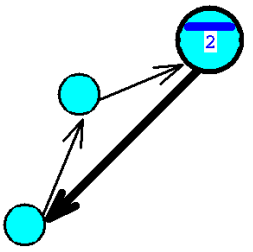
Restart Agent Time Show Shift Help

0 1 2 3 M M B W r k b Q K p L U

2 0 2 1 2 1 . . . . . .

*1.

1, 2, 3

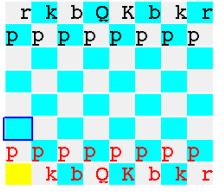



The trace

*1. => a0*b

r

| | | |
|---|---|---|
| views -> | .....xxxyyxxxyyx..yxyyxx. | |
| incorrect -> | .....224.24224..42262.62.4 | |
| actions -> | .....b..a..bb.aaa.b.....ba | |
| arrows -> | .....11111111111111111111 | |
| the trace -> | .....1..1..1..1..1..1..1 | |



## Какво представлява светът?

Входът и изходът на агента ще се състоят от три букви. Входът ще са буквите 0, x, y, а изходът ще се състои от 0, a, b. Освен входа агентът вижда още и кои са некоректните ходове:

1 – некоректно е $0$.
2 – некоректно е $a$.
4 – некоректно е $b$.

Агентът ще може да движи видимият прозорец наляво и надясно, както и нагоре и надолу. Също така ще може да вдигне наблюдаваната фигура, както и да сложи вдигната фигура на наблюдаваното квадратче.

## Как толкова много действия могат да се изразят с три букви?

Ходът на агента е разделен на три. Първо той казва какво ще е движението по хоризонтала – наляво, надясно или на място. Второ, ще е движението по вертикала – нагоре, надолу или на място. Трето агента ще каже дали вдига/спуска наблюдаваната/вдигната фигура или нищо не прави.

Това разделяне на действието на три е първата зависимост, която описва светът. Тази първа зависимост е отразена в първия събитиен модел. Това е моделът с три състояния, който брои: 1, 2, 3.

Тази зависимост има едно събитие (True). Това е събитието, което се случва на всяка стъпка. В едно от състоянията имаме следа, която казва, че ходът *b* ще е некоректен.

## Откриваем ли е този първи модел?

Да, ако погледнете редицата incorrect ще видите, че всеки трети член е 4 (или 6, което е 4+2). Тоест, в едно от състоянията се случва нещо специално и това ще ни помогне да намерим модела.

## Какъв ще е вторият модел?

Това ще е моделът, който ни каза в коя колона сме. Тоест, моделът, който ще ни даде X координатата на наблюдаващия прозорец.

Ето го и вторият модел:

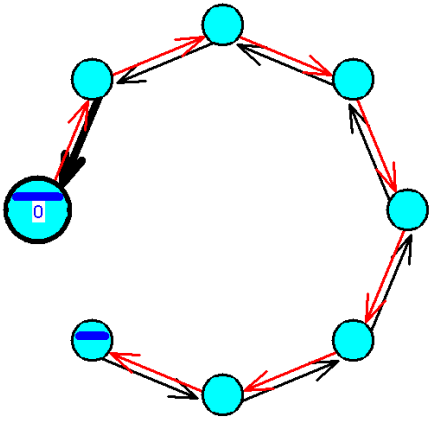
AI Unravels the Chess, step=21 (Pause)

Restart Agent World Time Show Shift Help

0 1 2 3 M M B W r k b Q K p L U

2 0 2 1 2 1 . . . . . .

Horizontal




1. m0=0 & a0=a

2. m0=0 & a0=b

The trace

1. m0=0 => a0*a

2. m0=0 => a0*b

r

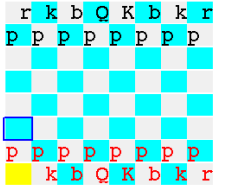
views -> .....xxxyyxxxyyx..yxyyxx.

incorrect -> .....224.24224..42262.62.4

actions -> .....b..a..bb.aaa.b.....ba

arrows -> .....2..1..2..1.....

the trace -> .....1.....1.....1..1..1..



## Какви са събитията?

Има две събития (наляво и надясно). Тези две събития се случват, когато първият модел е в състояние 0, а действието на агента е съответно a или b. Тоест, този модел е построен йерархично на базата на първия модел.

## Каква е следата?

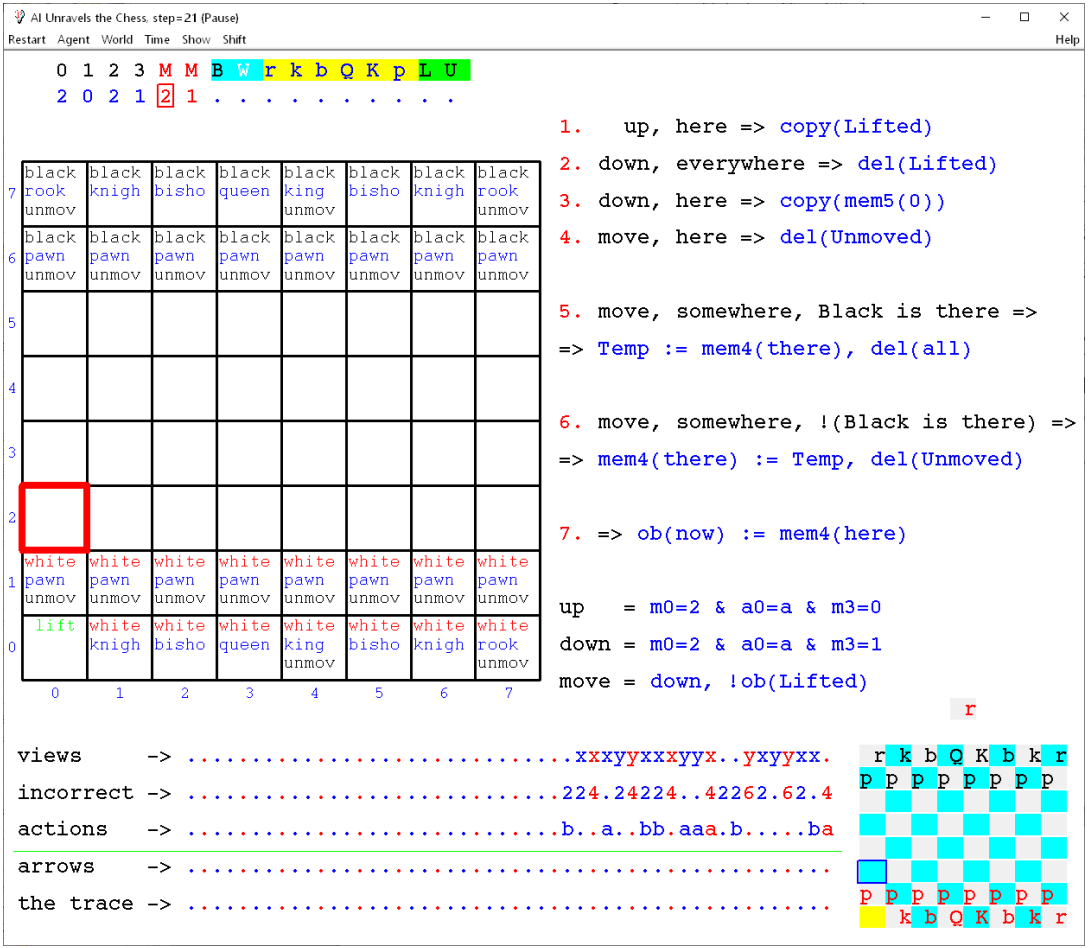
Следата е, че не може да се играе наляво. Такава е следата на първото състояние (което отговаря на най-лявата колона на таблото). Съответно, не може да се играе надясно в последното състояние.

## Откриваем ли е този модел?

Да, отново благодарение на следата.

## Следа с памет

Светът може да помни за всяка състояние на един модел какво последно се случва в това състояние. Това, което се случва постоянно е постоянната следа, но това което се е случило последно е нещо, което може да се запомни в паметта на следата. Тази памет е едномерен масив, в който на всяко състояние отговаря по една клетка от масива. Може да направим декартовото произведение на два модела и паметта на това декартово произведение. Например декартовото произведение на втория и третия модел има памет и това е таблото на играта. Може да се види в следния двумерен масив:


AI Unravels the Chess, step=21 (Pause)
Restart Agent World Time Show Shift
Help
0 1 2 3 M M B W r k b Q K p L U
2 0 2 1 2 1 . . . . . .
7 black rook unmov | black knigh unmov | black bisho unmov | black queen unmov | black king unmov | black bisho unmov | black knigh unmov | black rook unmov
6 black pawn unmov | black pawn unmov | black pawn unmov | black pawn unmov | black pawn unmov | black pawn unmov | black pawn unmov | black pawn unmov
5
4
3
2
1 white pawn unmov | white pawn unmov | white pawn unmov | white pawn unmov | white pawn unmov | white pawn unmov | white pawn unmov | white pawn unmov
0 lift | white knigh unmov | white bisho unmov | white queen unmov | white king unmov | white bisho unmov | white knigh unmov | white rook unmov
0 1 2 3 4 5 6 7
1. up, here => copy(Lifted)
2. down, everywhere => del(Lifted)
3. down, here => copy(mem5(0))
4. move, here => del(Unmoved)
5. move, somewhere, Black is there =>
=> Temp := mem4(there), del(all)
6. move, somewhere, !(Black is there) =>
=> mem4(there) := Temp, del(Unmoved)
7. => ob(now) := mem4(here)
up = m0=2 & a0=a & m3=0
down = m0=2 & a0=a & m3=1
move = down, !ob(Lifted)
r
views -> .....xxxyyxxxyyx..yxyyxx.
incorrect -> .....224.24224..42262.62.4
actions -> .....b..a..bb.aaa.b.....ba
arrows -> .....
the trace -> .....
r k b Q K b k r
p p p p p p p p
p p p p p p p p
k b Q K b k r

## Какво се случва в квадратчетата?

Там се появяват различни обекти. Обектите не се виждат директно, а се наблюдават техни свойства. Например, когато се появи черна пешка наблюдаваме свойството „черно“.

## Как изглажда свойството „черно“?

Свойствата са някакви събитийни модели. Ние нямаме идея как трябва да изглеждат тези модели, затова сме ги генерирали случайно. Например свойството „черно“ случайно се е получило да изглежда така:

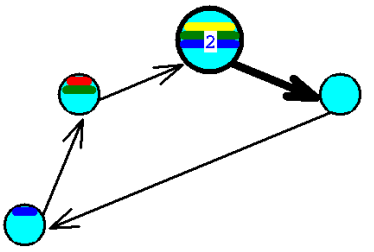
AI Unravels the Chess, step=47 (Pause)

Restart Agent World Time Show Shift Help

```
0 1 2 3 M M B W r k b Q K p L U
1 2 6 0 2 1 * . . . . . * . *
```

*1.

Black Colour



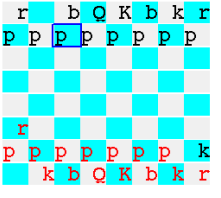

The trace

*1. v1=\=0 => v0=y

*2. v1=\=0 => v0=0

*4. => v0=0

```
views     -> ....xxxyyxxxyyx..yxyyxx.x.....................yy
incorrect -> ...224.24224..42262.62.42.6..6..6..6..6..6..6..
actions   -> ...b..a..bb.aaa.b.....babb..b.ba....bb.a..bb..b.ab
arrows    -> ...11111111111111111111111111111111111111111111111
the trace -> ...................................................7
```



Важното е, че когато наблюдаваме черен обект, това свойство е активно и влияе на входа, който агента вижда. Например, свойството бяло не е активно и то не влия на входа:

AI Unravels the Chess, step=47 (Pause)

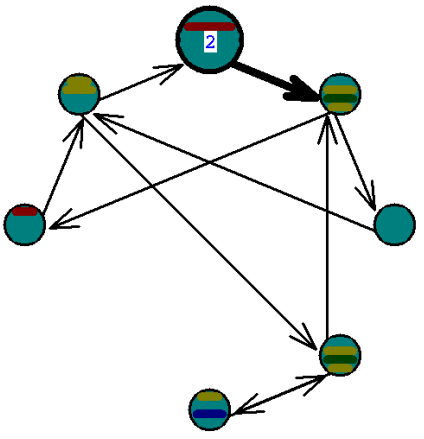
Restart Agent World Time Show Shift Help

0 1 2 3 M M B W r k b Q K p L U

1 2 6 0 2 1 * . . . . . * . *

*1.

White Colour




The trace

*1. v2=\=0 & v1=\=0 => v0=x

*2. => v0=x

*4. => v0=0

8. v2=0 & v1=\=0 => v0=y

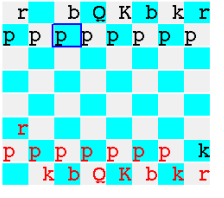
views -> ....xxxyyxxxyyx..yxyyxx.x.....................yy

incorrect -> ...224.24224..42262.62.42.6..6..6..6..6..6..6..

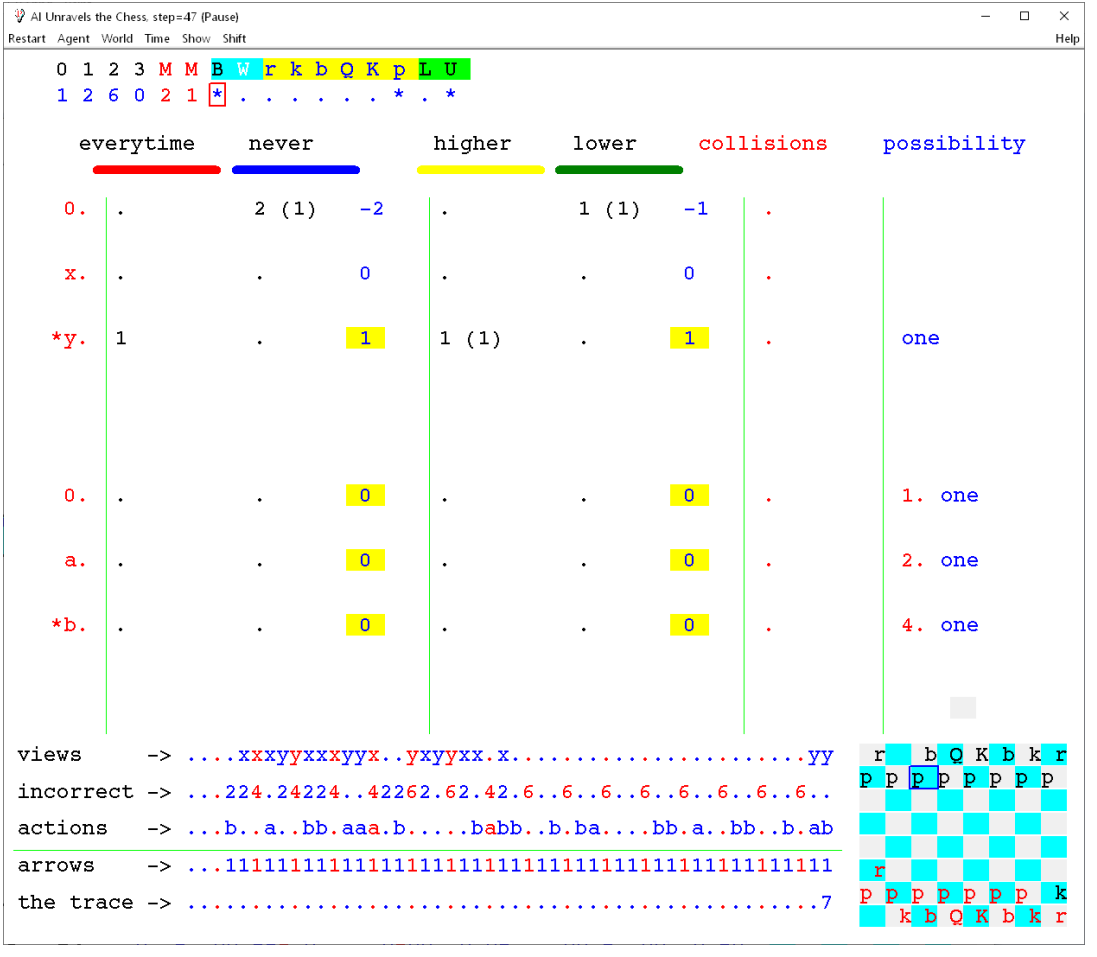
actions -> ...b..a..bb.aaa.b.....babb..b.ba....bb.a..bb..b.ab

arrows -> ...11111111111111111111111111111111111111111111111

the trace -> ....2e6.3161366...e.316.2.....................



Как света определя входа, при положение, че има много активни събитийни модели, които дават различни, понякога противоречиви насоки за това какво ще види агента. Следващото наблюдение на агента се определя, чрез гласуване от различните събитийни модели, които са активни в момента. Това може да се види в следващата таблица:


AI Unravels the Chess, step=47 (Pause)
Restart Agent World Time Show Shift
Help
0 1 2 3 M M B W r k b Q K p L U
1 2 6 0 2 1 * . . . . . * . *
everytime never higher lower collisions possibility
0. . 2 (1) -2 . 1 (1) -1 .
x. . . 0 . . 0 .
*y. 1 . 1 1 (1) . 1 . one
0. . . 0 . . 0 . 1. one
a. . . 0 . . 0 . 2. one
*b. . . 0 . . 0 . 4. one
views -> ....xxxyyxxxyyx..yxyyxx.x.....yy
incorrect -> ...224.24224..42262.62.42.6..6..6..6..6..6..6..6..
actions -> ...b..a..bb.aaa.b.....babb..b.ba....bb.a..bb..b.ab
arrows -> ...11111111111111111111111111111111111111111111111
the trace -> .....7

Тази програма ни показа как може да се опише играта шах, чрез 4 зависимости, 2 масива (т.е., паметта на следа на някоя зависимост) и 10 свойства. Свойствата също са събитийни модели, но те не са активни постоянно, а само когато свойството се наблюдава.

Тук липсва нещо. Ходовете на играчите са произволни и не са съобразени с правилата на играта шах. Например, агента игра непозволено с топа, а въображаемият играч, който е вътре в света, му отговори с коня, като скочи толкова далеч, че чак успя да вземе пешка.

## Как да забраним некоректните ходове?

За да кажем кой ход е коректен ни трябва още нещо. Нужно е да добавим алгоритми. Например, за да започне коня да се движи под формата на буквата „Г“ той трябва да изпълни определен алгоритъм на движение.

## Какво е алгоритъм?

След като зависимостите и свойствата са събитийни модели, да не би и алгоритмите да са същото? Точно така, алгоритмите също ще са събитийни модели.

Класическата представа за алгоритъм това е Машината на Тюринг. Можем ли да представим Машината на Тюринг чрез събитиен модел? Можем да представим главата, но не можем да представим лентата. Всъщност, какво е алгоритъм? Дали това е само главата или главата заедно с лентата?

## Каква е била идеята на самия Тюринг?

Той си е представял главата на машината като човек, който има крайна памет, а лентата е оприличавал с безкрайното количество хартия, с която този човек разполага, за да си води записки. Тоест, според Тюринг алгоритъма е в главата, а лентата е нещо външно, което се използва при изпълнението на алгоритъма.

Ако вие разполагате с алгоритъм за подреждане на буркани, дали бурканите са част от алгоритъма? Не, бурканите са част от света и вие можете да приложите този алгоритъм в този свят върху тези буркани.

Същото е с лентата, тя няма да е част от алгоритъма, а ще е част от света. В случая с играта шах, алгоритмите ще се изпълняват върху таблото. Тоест, таблото ще играе ролята на лента.

Главата на Машината на Тюринг можем да представим с краен събитиен модел. Какво ще стане, ако разрешим събитийния модел да има безкрайно много състояния? Тоест, ако разрешим Машина на Тюринг, чиято глава да има безкрайно много състояния. Такъв алгоритъм ще наречем неефективен. Ефективен алгоритъм ще бъде този, при който състоянията на главата (на събитийния модел) са крайно много.

## Какъв е изводът?

Изводът е, че чрез събитийни модели можем да опишем един сложен свят. При това можем да го опишем достатъчно просто, за да може това описание да бъде намерено автоматично. Тоест, събитийните модели са езика (формата) за описание на светове, който ИИ ще използва.