Как AI разбира какво се случва?

22 септември, 2017 г.

Повечето изследователи разглеждат AI като статична функция без памет. Това е една от малкото статии където AI се разглежда като устройство с памет. Когато имаме памет можем да си зададем въпросите: „Къде съм?“ и „Какво се случва?“ Когато нямаме памет се налага да предположим, че винаги сме на едно и също място и че света винаги е в едно и също състояние.

Keywords: Artificial Intelligence, Machine Learning, Reinforcement Learning, Partial Observability.

Въведение:

Стандартният подход в AI е да разгледаме едно множество от положителни примери и едно множество от отрицателни примери. Търсим функция, която за дадените положителни примери да казва ДА и за дадените отрицателни примери да казва НЕ. С помощта на намерената функция започваме да предсказваме какъв е правилният отговор за примери, за които не знаем дали са положителни или отрицателни.

По същество стандартният подход в AI представлява апроксимация. Търси се апроксимираща функция. Обикновено се търси в някакво определено множество от функции. Например при невронните мрежи функцията се търси в множеството на невронните мрежи.

Като типичен пример за стандартния подход в AI можем да посочим [1], където се търси една статична функция покриваща определени положителни и отрицателни примери. В статии като [2] нещата са малко по-различни, защото там на всяка стъпка се появяват нови примери и на всяка стъпка апроксимиращата функция се променя. Въпреки това, в [2] ние отново търсим статична функция без памет, макар че на всяка стъпка тази функция е различна. В [2] идеята е, че ние постоянно подобряваме апроксимационната функция, но във всеки конкретен момент ние имаме една конкретна статична функция.

Единствените изследвания, при които AI се разглежда като устройство с памет, са изследванията свързани с Reinforcement Learning. Дори и в тази област има два основни дяла, които се наричат Partial Observability и Full Observability. Нужна ли ни е памет когато виждаме всичко? Разбира се, че не! Ако виждаме какво е текущото състояние на света, то на нас не ни е нужно да помним каквото и да е от миналото, защото всичко което ни е нужно да знаем е кодирано в това, което виждаме. Тоест, единствения дял от AI, където AI се разглежда като устройство с памет е Reinforcement Learning with Partial Observability. Тази статия се отнася точно до този дял от Computer Science.

Защо казваме, че такива статии са малко? Има много статии в областта на Reinforcement Learning, но те в огромната си част изцяло са посветени на случая с

Full Observability. Например [3, 4]. Когато се спомене за Partial Observability обикновено се казва само, че такава възможност съществува, но не се казва какво правим в този случай.

В тази статия си задаваме въпроса „Какво се случва?“ Ако преведем този въпрос в термините на Reinforcement Learning, то той ще звучи по следния начин: „Кое е състоянието на света, в което се намираме в момента?“ Ако разглеждаме състоянията на света като граф, то въпросът „Какво се случва?“ може да се преведе като „Къде сме? В кой възел на графа сме?“ Ако си мислим за реалния свят, то „Какво се случва?“ означава „Къде се намираме физически в момента и какво е състоянието на света в този момент?“

За да кажем в кое състояние на света се намираме, първо трябва да опишем състоянията на света. Това никак няма да е лесно, защото ние не виждаме тези състояния напълно, а само частично (Partial Observability). По тази причина описанието на състоянията на света е свързано с това да си представим нещо невидимо.

За целта ние ще въведем понятието „свойство на света“, което ще представлява множество от състояния на света. С помощта на това понятие ние ще опишем текущото състояние на света. Това ще стане чрез свойствата на света, за които знаем дали са валидни. По този начин ние ще определим, че текущото състояние е елемент от сечението на група множества. Т.е. сечение между свойствата, които са валидни в този момент и допълненията на свойствата, които не са валидни. Ако тези свойства са достатъчно много, тогава сечението на всичките тези множества ще е достатъчно малко и описанието на текущото състояние на света ще е достатъчно точно. (Тоест, ние не определяме текущото състояние с точност до едно единствено състояние, за с точност до множество от състояния.)

За да опишем свойствата на света, първо ще видим с какви свойства разполагаме. Единственото, с което разполагаме, това са експерименталните свойства. Ще искаме да опишем някакво свойство чрез това, с което разполагаме. Това ще стане като определим за определено експериментално свойство, че от него следва нещо за свойството, което описваме. Например, че следва, че свойството с някаква вероятност е истина. Хубавите случай са когато следва че с вероятност 100% е истина или обратното, на 100% да е лъжа.

Как ще определяме свойствата? За да се образува капка дъжд е нужна една прашинка, която да послужи като кондензационно ядро. Аналогично, за да създадем свойство, което да е дефинирано за всички състояния на света, на нас ни е нужно да имаме свойството дефинирано за някакво сравнително малко подмножество. На базата на това подмножество ще направим екстраполация и ще предположим, че вероятността за цялото множество е същата каквато е за подмножеството.

Кои ще са тези кондензационни ядра, чрез които ще получим нужните свойства на света? Това ще са дефинираните в [8] тестове и съответстващите им функции на теста.

Тази статия по същество е продължение на [8], но, за да бъде разбрано написаното тук, не е задължително статия [8] да бъде прочетена (съкратени варианти на [8] са публикувани в [9] и в [10]). В [8] с понятието „Тестово състояние“ се описват пет различни неща: Тестове, Функция на теста, Прогноза за функцията, Тестово свойство и Тестово състояние. Това, че пет различни неща се обозначават с един термин може да се приеме за грешка, макар че между първото и второто има биекция. Прогнозата също е определена еднозначно, при условие, че сме фиксирали за кога е тя (т.е. след колко стъпки). Що се отнася до тестовото свойство то е някакво продължение на функцията на теста. Тази функция може да бъде продължена по много начини, но ние търсим едно продължение, което да е възможно най-естествено. Тестово състояние зависи от разбиването на групи на относителна устойчивост. Това разбиване може да се направи по много начини, но смислените разбивания са малко.

Тази статия започва с въвеждането на три нови дефиниции на Reinforcement Learning и доказателство за еквивалентността на тези нови дефиниции със стандартната дефиниция. Целта на новите дефиниции е да ни помогнат да въведем понятията случайност, тестово състояние и шум. Ще дадем един конкретен пример, който оправдава въвеждането на новите дефиниции. Ще въведем понятията събитие (експеримент), ще кажем какво е свойство на света и ще разгледаме свойствата определени от експерименти. Ще въведем понятието тест. От него ще получим тестовото свойство. Когато имаме не една, а много групи на относителна устойчивост, тогава вместо тестово свойство ще говорим за тестово състояние. Тест, ще говорим за тест, който определя не едно, а няколко свойства на света. Накрая ще кажем как се определят теориите, които ни описват свойствата на света.

По този начин ще отговорим на въпроса „Какво се случва в момента?“ Отговора е множество от свойствата, за които знаем дали са валидни в момента. Следващото действие на нашето устройство няма да зависи единствено от това какво вижда в момента. То ще зависи още от това, кои свойствата на света в момента са валидни (тоест, от това какво се случва в момента). Това означава, че нашето устройство ще е устройство с памет.

Постановка на задачата

Имаме една последователност *действие, наблюдение, действие, наблюдение* ... и искаме да разберем тази последователност. Целта ни е да предскажем как тази последователност ще продължи и да изберем такива действия, че да постигнем максимално добър резултат. Ние ще предполагаме, че тази последователност не е случайна, а че тя се определя от правилата на някакъв свят, в който ние се намираме.

Какъв е общия вид на света е съществено, защото ние ще се опитваме да разберем света, тоест ще се опитваме да му построим модел, а този модел ще го търсим във вид близък до избрания то нас общ вид.

Ще разгледаме четири възможни дефиниции за вида на света. Ще докажем, че тези четири дефиниции са еквивалентни. Ползата от трите нови дефиниции, които ще

предложим, е че те ще ни помогнат при построяването на модел на света. Втората дефиниция ще ни помогне да добавим понятието случайност към нашия модел, третата дефиниция естествено ще ни доведе до понятието Тестово свойство, а четвъртата ще ни помогне да добавим понятието шум.

Дефиниция 1. Това е обичайната дефиниция на свят при Reinforcement Learning. Тя е следната: Имаме едно множество S от вътрешни състояния на света и едно от тях s_0 е началното. Как се променя вътрешното състояние на света се определя от функцията World, а това какво виждаме на всяка стъпка се определя от функцията View. В сила е следното:

$$s_{i+1} = World(s_i, a_{i+1})$$
$$v_i = View(s_i)$$

Тук действията и наблюденията (a_i и v_i) са вектори от скалари с размерности n и m съответно. Всеки от тези скалари ще бъде крайна функция с k възможни стойности, където k ще е различно за различните координати на a и v.

Да си зададем въпроса дали функцията World е еднозначна (single-valued) или многозначна (multivalued)? При дефиниция едно тази функция е еднозначна, тоест света е детерминиран. При новите дефиниции това ще се промени.

Следващият въпрос е дали функцията World е тотална или частична (partial)? В [8] се аргументирахме, че трябва да предполагаме, че World е частична функция и случаите когато тя не е дефинирана, ще приемаме за некоректни ходове. Пак в [8] приехме, че на всяка стъпка ще можем да проверяваме за всеки ход дали е коректен или некоректен. Тоест, на всяка стъпката ние ще виждаме две неща. Първо ще виждаме това, което ни дава функцията View и второ ще виждаме това кои действия са коректни и кои некорекни в този момент.

Както казахме, това е обичайната дефиниция, която използват повечето автори (например [3, 4]) с тази разлика, че другите автори обикновено предполагат, че функцията World е тотална и че всички ходове са коректни. Няма да докажем, че дефинициите в случая с тотална и частична функцията World са еквивалентни, защото те не са. В случая когато допускаме некоректни ходове устройството получава повече информация. Това може частично да се емулира в случая, когото няма некоректни ходове, но тази емулация ще е частична, а не пълна.

Дефиниция 2. При еднозначната функцията World ни липсва понятието случайност. Нека да видим как можем да го добавим? Ако позволим функцията World да бъде многозначна, то следващото състояние на света няма да е определено точно, а ще бъде едно от няколко възможни. Добре, но ние бихме искали да кажем нещо за вероятностите на тези различни възможности. Нека имаме k различни възможности. Нека всяка от тях се случва с някаква вероятност p_i. Сега нещата заприличаха на Markov decision process или по-точно на Partially observable Markov decision process, защото тук се занимаваме със случая на Partial Observability.

Имаме два варианта на случайност. При първия вероятността въобще не е определена, а при втория тя е определена прекалено точно. И два варианта не ни харесват, затова ние ще изберем нещо по средата. При първия вариант вероятността е някъде в интервала *[0, 1]*. Във вторият вариант вероятността има фиксирана стойност, която е някакво си p (тоест тя е в интервала $[p, p]$). Ние ще изберем варианта, при който вероятността е в някакъв интервал $[a, b]$. Тоест, няма да знаем точно с каква вероятност ще се случи, но ще знаем, че вероятността е поне a и не повече от b.

Има два вида случайност. Прогнозируема случайност и непрогнозируема. Когато хвърляме зар то ще се падне шестица с вероятност 1/6. Това е прогнозируема случайност. Когато питаме шефа си за увеличение на заплатата отговора ще е ДА или НЕ, но не можем да кажем каква ще е вероятността да ни кажат ДА. Това е непрогнозируема случайност. Прогнозируемата случайност е доста определена, защото благодарение на Law of large numbers ние знаем доста точно колко ще са успешните опити. Варианта, който ние избрахме е едно съчетание между прогнозируема и непрогнозируема случайност. Разбира се, за да бъде дефиницията коректна трябва да се погрижим за това някои неравенства да бъдат изпълнени. Ако *World(s, a)* има k възможни стойности с вероятности в интервалите $[a_i, b_i]$, то трябва да са изпълнени неравенствата:

$$a_i \leq b_i \qquad \sum_{i=1}^{k} a_i \leq 1 \qquad \sum_{i=1}^{k} b_i \geq 1$$

Ако

$$Sum = \sum_{i=1}^{k} a_i$$

Тогава трябва да бъдат изпълнени и неравенствата:

$$b_i \leq 1 - Sum + a_i \qquad (1)$$

Ще предполагаме, че неравенството (1) е равенство поне за едно i. Можем да предположим дори, че е равенство за поне две i.

С това втора дефиниция на свят е завършена. Остава само да докажем, че тя е еквивалентна на първата.

Забележка: Ще предполагаме, че функцията World няма да е твърде недетерминирана, защото това би направило света твърде неразбираем. Например, ако предположим, че от всяко състояние и при всяко действие функцията World с еднаква вероятност може да премине към произволно друго състояние, то би се получил един напълно неразбираем свят. Много по-разбираемо би било, ако в повечето случаи функцията World връща една единствена възможност, а когато възможността не е единствена, то възможностите да са малко и една от тях да е много по-вероятна от останалите.

Теорема 1. Втора дефиниция е еквивалентна на първата.

Забележка: Следващото доказателство е техническо, затова съветваме читателя да го прескочи и да го приеме на доверие.

Доказателство: Едната посока е лесна, защото е очевидно, че първата дефиниция е частен случай на втората. За обратната посока е нужно за произволен свят описан по втората дефиниция да построим еквивалентен на него свят описан по първата дефиниция.

Идеята на доказателството е да скрием случайността в едно естествено число. Ще имаме една функция *F*, която от текущото случайно число ще изчислява следващото. По подобен начин се изчисляват псевдослучайните числа в компютрите. Там се започва от някакво число (ние ще започнем от нула) и всяко следващо псевдослучайно число се получава чрез функцията *F* от предишното (т.е. *F(x)*). Функцията *F* трябва да е достатъчно сложна, за да не можем да отгатнем кое ще бъде следващото число. Освен това *F* трябва да е добра, което означава че за някое Q всички остатъци по модул Q са равновероятни.

В нашия случай имаме две случайности и затова ще добавим две естествени числа и две функции *F_good* и *F_bad*. Тези две функции ще искаме да са достатъчно сложни (да не са изчислими и да не могат да бъдат апроксимирани с изчислима функция). Освен това за двете функции ще искаме редицата $F^i(0)$ да е без повторение. Само за функцията *F_good* ще искаме да е добра за някое конкретно Q, а за *F_bad* няма да искаме нищо повече. Ще използваме *F_good* за изчисляването на прогнозируемата случайност, а *F_bad* за изчисляването на непрогнозируемата.

Ще дефинираме функцията *F_good* по следния начин:
F_good(0)=0
$F\_good^{i+1}(0)$*= първото неизползвано число, от тези които дават остатък k при деление на Q, където* $k \in [0, Q-1]$ *и е избрано случайно.*

Забележка: Определихме функцията *F_good* чрез един безкраен процес, при който на всяка стъпка извършваме случаен избор (например като теглим от шапка, в която има *Q* топки). По този начин ние получаваме една неизчислима функция. В тази статия се опитваме да опишем един конкретен алгоритъм и всичко, което е част от този алгоритъм трябва да е изчислимо. Функциите *F_good* и *F_bad* не са част от този алгоритъм. Тяхната задача е единствено да се покаже еквивалентността на две дефиниции. Нужно ни е единствено да покажем, че тези две функции съществуват. Никога няма да строим тези функции на практика, нито ще ги изчисляваме.

Забележка: Ако по някаква причина поискаме да реализираме функциите *F_good* и *F_bad*, (например, ако искаме да построим изкуствен свят, в който да тестваме AI) тогава бихме използвали стандартната функция *Random*, която е вградена в повечето езици за програмиране. *Random* за разлика от *F_good* не е съвършена, но е достатъчно добра за целите на практиката. *Random* не работи с естествени числа, а с 32-битов integer. Тя не е съвършена в смисъл, че е сложна, но не е безкрайно

сложна и следващото случайно число теоретично може да бъде познато (но на практика не може). Освен това редицата от случайни числа не е безкрайна, а се повтаря (но след доста дълъг период). Тоест, за целите на практиката *F_good* може да бъде заменена с *Random*. *F_bad* също може да бъде построена с помощта на *Random*. Единствено трябва да се погрижим за това което функцията връща. Всички класове по модул *Q* да не са равно вероятни, а да имаме друго вероятностно разпределение. От време на време това вероятностно разпределение да се променя по случаен начин.

След тази подготовка сме готови да докажем еквивалентността на двете дефиниции. Нека имаме един свят според втората дефиниция съответно с *S*, s_0, и *World*. Новият свят, който ще построим ще има множество от състоянията $S\times\mathbb{N}\times\mathbb{N}$, начално състояние $(s_0, 0, 0)$ и функция *Big_World*, която се дефинира по следния начин:

$Big\_World((s, x, y), a)= (s', F\_good^{\,2}(x), F\_bad(y))$

Тук *F_good* е на квадрат, защото тази функция се използва два пъти при изчислението на s'.

$s'= World(s, a)$, когато $World(s, a)$ има една възможна стойност

Когато $World(s, a)$ има k възможни стойности, тогава избираме една от тях по следния начин:

Разделяме интервала $[0, 1]$ на $k+1$ подинтервала с дължини от a_1 до a_k, а последния с дължина колкото остане. Избираме случайна точка в интервала $[0, 1]$.

Ако точката е попаднала в някои от първите k подинтервала, то тогава s' ще бъде равно на i-тата от възможните стойности на $World(s, a)$. Тук i е номера на интервала, в който сме попаднали.

Ако точката е попаднала в последния подинтервал, тогава изчисляваме коефициентите:

$$c_i = \frac{b_i - a_i}{1 - \mathrm{Sum}}$$

Отново взимаме интервала $[0, 1]$ и нанасяме върху него точките c_i. По този начин разделяме интервала $[0, 1]$ на $k+1$ подинтервала (някои от които с нулева дължина). Отново избираме случайна точка в интервала $[0, 1]$ и гледаме в кой подинтервал тя е попаднала. На първия подинтервал съответстват k възможни стойности на $World(s, a)$, на втория съответстват $k-1$, а на последния съответстват 0 възможни стойности. В последния интервал няма как да попаднем, защото той е с дължина нула (защото предположихме, че поне едно от неравенствата (1) е равенство). Дори можем да предполагаме, че последните два интервала са с нулева дължина. След като разберем колко и кои са възможните стойности на $World(s, a)$ избираме една от тях с непрогнозируемата случайност. Как става това? Ако възможните стойности са R, то взимаме числото $(y \bmod R)+1$. Ако това число е 1, то взимаме първата от възможностите, ако е 2, взимаме втората и т.н.

Пропуснахме да кажем как избираме случайна точка в интервала $[0, 1]$. За целта разделяме интервала на Q равни части. Взимаме числото $(x \bmod Q)+1$ и това ще е номера на интервала, който ще изберем. Коя от точките на този интервал ще

вземем – няма значение, защото тези интервали ще се съдържат изцяло в интервалите, които ние разглеждаме. За да бъде вярно последното ще предположим следното:

Предполагаме, че числата a и b са рационални (ако са ирационални, то с леко закръгление можем да ги направим рационални). Дори ще предполагаме, че тези числа са от вида $x/100$ (тоест, че са стотни). Ако не са, пак със закръгление можем да ги направим от този тип. (В случая закръглението е допустимо, защото става дума за интервал на вероятност и ако разликата е малка, то това ще може да се осети чак след много голям брой стъпки.) Ще предполагаме още, че числото Q, което използвахме при построяването на F_good е равно на най-малкото общо кратно (the least common multiple) на числата от 1 до 100.

Забележка: При първия избор на случайна точка взимаме числото $(x \bmod Q)+1$, а при втория избор вместо x взимаме $F\_good(x)$, т.е. взимаме следващото случайно число.

С това еквивалентността на първата и втората дефиниция е доказана.

♦

Дефиниция 3. Зад тази дефиниция стои следната идея: Това, което виждаме може да се промени без да се променя мястото, на което се намираме. Например, вие сте в кухнята и виждате, че тя е боядисана в жълто. На следващия ден пак сте в кухнята, но този път виждате, че тя е боядисана в синьо.

Ще променим функцията View и състоянията на света. Ако функцията View връща вектор от скалари с размерност m, то тогава към всяко състояние ще добавим m видими променливи. Сега вече функцията View ще връща стойностите на тези m променливи. Освен видимите променливи ще добавим още u на брой невидими променливи. Идеята за невидимите променливи е, че не всичко може да се види. Например, ако някой ви е ядосан, това вие не го виждате, но то не е без значение, защото в последствие това ще се отрази на неговите действия.

Ще въведем понятието обобщено състояние и то ще се състои от състоянието на света и от стойността на всичките $|S|.(m+u)$ променливи. Когато искаме да подчертаем, че не става дума за обобщено състояние ще казваме стандартно състояние.

Функцията World ще бъде дефинирана за двойката от обобщено състояние и действие и ще връща обобщено състояние. Отново функцията World ще бъде многозначна и няма да е тотална. Тоест, отново ще допускаме случайност и некоректни ходове.

Забележка: Една променлива може да не е свързана със състоянието към което е прикачена. Това особено важи за невидимите променливи. Питаме се дали да не въведем глобални променливи? Отговора е, че глобални променливи не са ни нужни, защото всяка локална променлива може да считаме, че е една и съща за цяла група от състояния и дори за всички състояния.

Забележка: Ще предполагаме, че стойностите на променливите няма да се променят твърде необуздано, защото това би направило света твърде неразбираем. Например, ако предположим, че функцията World на всяка стъпка променя стойността на всички променливи по абсолютно произволен начин, то би се получил един напълно неразбираем свят. Много по-разбираемо би било, ако повечето променливи са константи и не се променят, а когато не са константи да се променят сравнително рядко и то по ясни и прости правила.

Теорема 2. Третата дефиниция е еквивалентна на втората.

Доказателство: Да докажем еквивалентността на втората и третата дефиниция е лесно. Ако допуснем, че всички променливи са константи (т.е. функцията World никога не ги променя), тогава ще видим, че дефиниция 2 е частен случай на дефиниция 3. Обратното, ако разглеждаме обобщените състояния като стандартни състояния, то от третата дефиниция получаваме втората. Единствената забележка е в това, че променливите може да са безбройно много (ако състоянията са безбройно много). Оттам получаваме, че обобщените състояния може да са прекалено много (continuum), но ако се ограничим само до достижимите състояния, то те пак ще са крайно или изброимо много.

♦

Дефиниция 4. Следващото нещо, което ще направим е да въведем понятието шум. Ще променим дефиниция 3 така, че новата функция View вече няма да връща чистата стойност на видимите променливи, а стойността замърсена с известно количество шум. За да опишем шума ще са ни нужни две неща, сила и спектър на шума. Силата ще бъде едно число Volume в интервала *[0, 1]*. Спектъра на шума ще е една k-tuple $<p_1, \ldots, p_k>$. За всяка видима променлива ще добавим още по k+1 невидими променливи, които ще съдържат силата и спектъра на шума на тази променлива (тоест, ще добавим още $|S|.m.(k+1)$ невидими променливи, ако k е едно и също за всички видими променливи). Функция View ще връща стойността на съответната видима променлива с вероятност $1 - \text{Volume}$. С вероятност *Volume* ще се връща шум, който ще бъде една от възможните *k* стойности, всяка от които с вероятност съответното p_i.

Обобщените състояния при дефиниция 4 нека да са същите като при дефиниция 3. Тоест, те ще зависят от всички видими и невидими променливи, но няма да зависят от това какво връща функцията View. Тоест, тук функцията View не се определя еднозначно от стойностите на видимите променливи на съответното състояние, а още от стойностите на невидимите променливите, които описваме шума. Освен това в определянето на функцията View участва и известна прогнозируема случайност.

Забележка: По този начин описваме ситуацията, когато на входа си устройството получава информацията замърсена с известно количество шум. Какво правим когато изходящата информация също е замърсена с шум? За второто сме се погрижили още при дефиниция 2, защото там вече е въведена възможността при едно и също действие да има различни възможни последствия за света, всяко от които с различна вероятност.

Забележка: Отново получихме нещо като Partially observable Markov decision process (POMDP) с тази разлика, че при нашия вариант променливите имат своята стойност, макар че тя е замърсена с шум, докато при POMDP има само шум (тоест за всички видими променливи шума е на 100%, което значи Volume=1). Единственото по което различаваме различните състояния в POMDP това е, че те имат различни спектри на шума.

Забележка: Ще предполагаме, че света не е прекалено шумен, защото ако навсякъде нивото на шума е максимално (т.е. едно) и навсякъде спектъра на шума е един и същи, то подобен свят би бил напълно неразбираем. Много по-разбираемо би било, ако нивото на шума е нула или близко до нула.

Теорема 3. Четвъртата дефиниция е еквивалентна на третата.

Доказателство: Ясно е, че дефиниция 3 е частен случай на дефиниция 4. Към всеки свят от дефиниция 3 можем да добавим шум, чието ниво навсякъде е нула и ще получим еквивалентен на него свят по дефиниция 4.

Да направим обратното. Нека вземем един свят по дефиниция 4. За всяко от обобщените състояния на този свят ще видим колко възможни изхода може да даде функцията View (по принцип възможният изход е един, но заради шума може да има много възможни изходи). За всеки от тези възможни изходи ще направим ново състояние, в което видимите променливи да имат стойността точно на възможния изход.

Забележка: От всяко обобщено състояние правим много нови стандартни състояния. Това, което ще се получи ще е свят по дефиниция 3 (и дори по дефиниция 2, защото видимите променливи ще са константи). Дали по този начин ние не губим информация? Дали не губим стойността на видимите и невидимите променливи? Не, защото тази информация е кодирана в новото състояние, което създаваме. То отразява стойността на всички променливи на всички състояния (а не само на променливите на текущото стандартно състояние).

Как ще дефинираме функцията World на новия свят? Ако между две обобщени състояния има връзка при действието *action* и тази връзка е с вероятност да се осъществи в интервала *[a, b]*, то тогава всяко от тези две обобщени състояния е заменено от много стандартни състояния и между всяко едно състояние от левите и всяко едно от десните пак ще има връзка при действието *action*. Разликата ще е само във вероятностния интервал. Той няма да е *[a, b]*, а ще бъде *[a.p, b.p]*, където *p* е вероятността точно този да е възможният изход, който да се е получил в дясно. Тоест, няма значение от кое точно състояние си тръгнал от левите, те всичките се държат по един и същи начин, защото те са еднакви от гледна точка на бъдещето. Те се различават само в настоящето (от функцията View) и то се различават само заради шума.

♦

Пример

Ще дадем един конкретен пример, който ще ни покаже предимствата на дефиниции 2, 3 и 4. Примерът ще е подобен на този, който сме дали в [6, 7]. Разликата ще е в това, че като основа на примера тук ще използваме играта шах, докато в [6, 7] използвахме играта Tic-Tac-Toe. Основните разлики са две:
1. Шаха има 64 квадратчета, докато Tic-Tac-Toe има само 9.
2. В шаха ще имаме команда „вдигни фигурата“ и „спусни фигурата“, докато в Tic-Tac-Toe вместо тези две команди имаме само „сложи кръстче“.

Нека имаме свят, в който играем шах срещу въображаем противник. Няма да виждаме цялото табло с всичките фигури. Вместо това, ще виждаме само едно квадратче и фигурата, която е в това квадратче. Нека да припомним, че тук се занимаваме със случая на Partial Observability. Ако виждахме цялото табло бихме имали случая на Full Observability. Въпреки всичко, това че виждаме само едно квадратче няма да е проблем, защото ще можем да местим поглед, т.е. да променяме квадратчето, което виждаме и по този начин да обходим цялата шахматна дъска.

Нашето действие ще бъде 3- tuple състояща се от <horizontal, vertical, command> където:
horizontal$\in${Left, Right, Nothing}
vertical$\in${Up, Down, Nothing}
command $\in${„вдигни фигурата“, „спусни фигурата“, New Game, Nothing}

Функцията View ще ни върне 3- tuple <chessman, color, immediate_reward>
chessman $\in${Pawn, Knight, Bishop, Rook, Queen, King, Nothing}
color$\in${Black, White, Nothing}
immediate_reward$\in${-1, 0, 1, Nothing}

Нашето действие ще се ни дава възможност да движим очи по таблото, по хоризонтала, по вертикала и дори по диагонал (т.е. едновременно по хоризонтала и по вертикала). Ще ни трябва търпение, защото ще се движим само с по едно квадратче на стъпка.

Освен да местим поглед по таблото, ще ни трябва и да можем да местим фигурите. За целта имаме две команди: „вдигни фигурата“, т.е. вдигни тази фигура, която виждаш в момента. Другата команда е „спусни фигурата“, т.е. спусни фигурата, която си вдигнал и я постави на квадратчето, което виждаш в момента. Разбира се, коя фигура си вдигнал ти не виждаш, но се надяваме, че това го помниш.

За наше улеснение ще предполагаме, че играем винаги с белите и че когато местим фигура (т.е. спускаме вдигната фигура) веднага (още в същата стъпка) въображаемият противник ще направи своя ход. Т.е. на следващата стъпка една от черните фигури вече ще е на друго място. Когато партията приключи immediate_reward ще стане 1, -1 или 0 в зависимост от това дали сме спечелили, загубили или партията е реми. В останалото време immediate_reward ще бъде Nothing. Ще предполагаме, че когато партията завърши няма веднага да започва

следващата, а ще имаме време да огледаме таблото и да разберем защо сме загубили. Когато сме готови за следващата партия извикваме командата New Game и фигурите се нареждат за нова партия.

Ще има ли некоректни ходове? Да, когато сме в лявата колона няма да можем да се движим наляво. Когато гледаме черна фигура или празно квадратче няма да имаме право да вдигаме фигура. Ако вече сме вдигнали фигура, няма да имаме право да вдигнем втора преди да спуснем вдигнатата.

Нека да опишем този свят в термините на дефиниция 1. Множеството на вътрешните състояния ще се състои от три неща (от наредени тройки). Първото ще бъде позицията на таблото (възможните позиции са много), второто ще бъде координатите на окото (64 възможности), третото ще бъде координатите на вдигнатата фигура (65 възможности – една допълнителна за случая когато нищо не сме вдигнали). За да бъдат нещата детерминирани ще предположим, че въображаемия противник е детерминиран. Тоест, че при една и съща позиция винаги играе един и същи ход. (Повечето програми, които играят шах са детерминиран противник.) При това положение е ясно как се дефинират функциите World и View.

Как да постъпим, ако искаме въображаемия противник да не е детерминиран. Например, той може да е човек или дори група хора (които се сменят и се редуват да играят срещу нас). Човекът и особено групата хора са недетерминиран противник. Естествено е при определена позиция някои ходове да са по-вероятни, а други по-малко вероятни, но да не може да се каже точно каква е вероятността за избирането на определен ход. В този случай най-естествена е дефиниция 2.

Състоянията на света пак ще са същите, но функцията World вече ще е многозначна. Ако искаме да се върнем към дефиниция 1, но да запазим недетерминираността на въображаемия противник, ще трябва да направим една сложна конструкция от типа на тази, която направихме при доказателството на Теорема 1. Това би увеличило броя на вътрешните състояния на света. Те и сега са много, но са крайно много, а така ще станат безкрайно много.

Как би изглеждал този пример при дефиниция 3? В този случай състоянията ще са само 64 (колкото са квадратчетата на таблото.) На всяко състояние ще има три видими променливи (chessman, color, immediate_reward). Третата видима променлива ще бъде обща за всичките 64 квадратчета. Това, че третата променлива е обща за всички квадратчета не е казано по никакъв начин. Устройството, което се опитва да разгадае света, ще трябва само да открие този факт. Разбира се, този факт не е труден за откриване. Много по-труден за откриване е факта, че първите две видими променливи зависят от състоянието на света и са различни за всяко квадратче.

В този случай няма да можем да минем само с видимите променливи. Трябва някъде да запомним коя фигура сме вдигнали. Този факт не се вижда в никое квадратче. Затова ще добавим към всяко квадратче една невидима променлива. Когато вдигнем фигурата, тя ще изчезне от видимите променливи и ще се появи в невидимата променлива на квадратчето, от което сме я вдигнали.

Така получихме един свят с 64 състояния и с по четири променливи към всяко състояние (три видими и една невидима). Броят на достижимите обобщени състояния на света при дефиниция 3 е точно толкова колкото са достижимите състояния при дефиниция 2. Въпреки това, света с 64 състояния изглежда много по-прост и по-разбираем, което оправдава въвеждането на дефиниция 3.

Нека сега представим света в термините на дефиниция 4. В този свят няма шум и няма смисъл да го представяме по дефиниция 4. За да стане подобно представяне необходимо хайде да добавим малко шум.

Ще предположим, че бялото е много тъмно, а черното е много светло и има вероятност да сбъркаме цвета на фигурата. Това ще го представим по следния начин. Към видимата променлива на цвета ще добавим шум с някаква стойност и спектър: Black 50%, White 50%, Nothing 0%. Когато квадратчето е празно, ще предположим, че шума е с *Volume=0.*

Ще предположим още, че фигурите Pawn и Bishop много си приличат и може да ги сбъркаме. За да представим това ще добавим към видимата променлива на фигурата шум, който ще имам някаква ненулева стойност когато фигурата е Pawn или Bishop. За останалите фигури шума нека да е нула. Спектъра ще бъде Pawn 50%, Bishop 50% и 0% за останалите случай.

Нека сега предположим, че кралят е твърде женствен и понякога го бъркаме с кралица, но не и обратното, тоест кралицата никога не я бъркаме с крал. Това ще го представим като добавим малко шум когато фигурата е King със спектър Queen 100%.

По този начин видяхме, че можем да опишем един доста сложен свят по един доста разбираем начин. Този пример оправдава въвеждането на дефиниции 2, 3 и 4.

Събитие или експеримент

Събитие ще бъде когато нещо се е случило, а експеримент, когато нещо сме направили. Разбира се, за всяко събитие ние може да сме се опитали да го предизвикаме или да го предотвратим. Затова ще считаме, че във всяко събитие ние имаме някакво участие. Затова няма да правим разлика между събитие и експеримент и ще приемаме тези две думи за синоними.

Искаме да дефинираме понятието експеримент (събитие). За целта първо ще кажем какво е история и какво е локална история около момента q.

Дефиниция: История ще наричаме редицата от действия и наблюдения a_1, v_1, ... , a_{t-1}, v_{t-1}, където t е текущия момент.

Забележка: За една история няма да казваме, в кой свят тя се е случила, защото ще считаме, че става дума за света, който трябва да разберем. По коя дефиниция е дефиниран този свят няма значение, защото четирите дефиниции са еквивалентни

(от тук нататък ще използваме предимно дефиниции 3 и 4, защото те са най-удобни за работа).

Забележка: Какво ще представлява една стъпка от историята? Дали да бъде <действие, наблюдение> или обратното? Решихме да бъде <действие, наблюдение>, защото момента, в който мислим е момента преди действието. В момента след действието не мислим, а само чакаме да се появи наблюдението. По тази причина историята започва с a_1. Първото действие нашето устройство ще трябва да го извърши на сляпо, защото още нищо няма да е видяло. Затова първото действие можем да го изберем произволно. Ще го изберем да бъде нулевият вектор (нулата я означаваме с Nothing – виж [6]). Казахме, че света започва от s_0, но ние не виждаме какво се случва в s_0. Затова света започва реално от s_1.

Дефиниция: Локална история около момента q ще наричаме една подредица получена от някоя история, където от номера на всеки индекс е извадено q.

Общия вид на локалната история е: a_{-k}, v_{-k}, ... , a_0, v_0, ... , a_s, v_s. Получаваме я от някоя история като вземем стъпката a_q, v_q и добавим последните k стъпки преди нея и следващите s стъпки след нея. Като извадим q от номера на всеки индекс стъпката a_q, v_q става a_0, v_0.

Ще разглеждаме локалната история като последователност от букви (тоест като дума). Ще представим тази дума като конкатенация от две думи *past.future*. Тук *past* завършва с a_0, v_0, а *future* е от там нататък. Тоест, настоящето е част от миналото, защото то вече се е случило.

Искаме да определим понятието експеримент (събитие) като булева функция, която да е монотонна (тоест, ако събитието се е случило в една локална история, като продължим локалната история, то пак да с е случило). Освен това искаме тази булева функция да бъде изчислима.

Дефиниция А: Експеримент ще наричаме булева функция дефинирана върху локалните истории *past.future*, която се определя от два изчислими езика L_1 и L_2 и е вярна точно когато $\exists u_1, u_2$ такива че $u_i \in L_i$ и u_1 е край на *past* и u_2 е начало на *future*.

Забележка: Не е задължително да разбираме за събитието в момента, когато то се е случило. Може да разберем по-късно (когато съобщят по новините). Това е причината, поради която събитието зависи не само от миналото, но и от бъдещето. Възможно е да не ни трябва да знаем бъдещето и дори да не ни трябва да знаем цялото минало. Тоест, може да разберем, че събитието ще се случи още преди то да се е случило (например, няколко стъпки преди да се случи).

При тази дефиниция на събитие изпускаме някои събития, за които се налага да броим от деня на раждането. Например събитието „Днес е понеделник“ е събитие от този вид. Ако не искаме да изпускаме тези събития ще трябва да променим дефиниция А по следния начин:

Дефиниция B: Същото като дефиниция A с тази разлика, че ще искаме локалната история да започва от началото (от a_1, v_1) и u_1 няма да е само край на *past*, а ще бъде равно на *past*.

В [8] разглеждаме зависимости без памет (т.е. събития по дефиниция A, при които дължините на u_1 и на u_2 са ограничени). В [8] стана дума за зависимости с памет (т.е. събития по дефиниция A, при които L_1 и L_2 са регулярни езици.)

Забележка: Регулярните езици могат да се опишат като думите започващи с нещо, съдържащи нещо, завършващи на нещо и в които нещо се е случило *m mod n* пъти. Дефиниция A ни казва, че *past* трябва да завършва на нещо или да съдържа нещо. Дефиниция B добавя още случаите когато *past* трябва да започва с нещо или нещо да се е случило *m mod n* пъти. Обаче, нас не ни интересуват случаите когато *past* започва с нещо или съдържа нещо. Тези случай не ни интересуват, защото макар че теоретично разглеждаме много възможни истории, на практика историята е само една (нашата история). Тази единствена история или е започнала с нещо или не е. В тази история нещо или се е случило или не се е случило. Интересно би било ако нещо се е случило наскоро (например, преди не повече от 3 стъпки). Така би се получило събитие, което променя стойността си през времето. Събития, които са константа, не са ни интересни.

Експериментални свойства

След като казахме какво е експеримент сме готови да дефинираме експериментално свойство. Нека първо да кажем какво е свойство и локална история около състояние.

Дефиниция: Свойство ще наричаме множество от обобщени състояния на света. В един конкретен момент едно свойство ще е валидно, ако съответното обобщено състояние е елемент на свойството (т.е. на множеството от обобщени състояния).

Забележка: При дефиниции 1 и 2 няма разлика между стандартно и обобщено състояние. В този случай свойството е просто множество от състояния.

Забележка: Когато говорим за свойство, обикновено ще имаме предвид не множеството, а неговата характеристична функция. Ще говорим за частични свойства (чиято характеристична функция е частична) и за продължението на частичното свойство до тотално.

Дефиниция: Локална история около състоянието s ще наричаме една локална история около някакъв момент q, която е получена от някаква история, в която съответното състояние s_q е точно състоянието s.

Тоест, локална история около s е история, която ни казва как сме минали през s.

Забележка: Около състоянието s може да имаме много различни истории, защото миналото и бъдещето не са определени еднозначно. Неопределеността на бъдещето идва от това, че не знаем кое от възможните действия ще изберем. При дефиниции

2, 3 и 4 към тази неопределеност се добавя и неопределеността на случайността. Що се отнася до миналото, то също е неопределено, защото може да има много различни състояния, които след някакво действие да доведат до състоянието s. Разбира се, ние бихме могли да предположим, че всяко следващо състояние е чисто ново. Тоест, да предположим, че състоянията не се повтарят. Ние обаче предпочитаме да предположим, че в нашия свят няма излишни състояния (т.е. че ако две състояния са еквивалентни спрямо бъдещето и настоящето, то сме слели тези две състояния в едно). Тоест, ще предполагаме, че състоянията могат да се повтарят. Дори и обобщените състояния могат да се повтарят, а стандартните се повтарят често.

Всеки експеримент ни дефинира едно свойство по следни начин:

Дефиниция: Експериментално свойство ще наричаме множеството от обобщени състояния s, такива че за s има локална история около s, такава че в тази локана история експеримента се е случил (т.е. в тази локална история експеримента е проведен).

Всеки експеримент ни дефинира по едно свойство, но това свойство не е разпознаваемо, а е полу-разпознаваемо. Тоест, ако експеримента е проведен, то свойството е валидно (в съответния момент). Не можем да кажем обратното. Ако експеримента не е проведен, не можем да кажем че свойството не е валидно, защото при друго развитие на миналото и на бъдещето, може би експеримента би бил възможен. Ако имаме шум (дефиниция 4) тогава дори и настоящето може да има друго развитие, заради шума.

Всеки експеримент разделя множеството на обобщените състояния на две (такива около които експеримента може да се извърши и такива, за които това не може да се случи). Когато експеримента е проведен около едно обобщено състояние, това значи, че то е някое от състоянията на свойството, но не всички състояния на свойството са равновероятни. Някои са по-вероятни, а други са почти невероятни. Каква е вероятността на съответното обобщено състояние не можем да кажем, но се надяваме, че събирайки статистика за един конкретен експеримент, индиректно ще отчетем тези вероятности.

Експерименталните свойства са най-доброто, с което разполагаме. Ако искаме да определим някакво свойство, ще трябва да го опишем чрез експерименталните свойства. Това описание ще става с помощта на статистиката, която сме събрали. В [8] разказахме за това как ще събираме статистика за зависимостите без памет. Пак в [8] стана дума и за събитията с памет.

Какво е тест?

Лошото на експерименталните свойства е, че са полу-разпознаваеми. Иска ни се да добавим свойства, които да са разпознаваеми (може да не са разпознаваеми във всеки момент, но все пак да има моменти, в които да можем да кажем дали свойството е валидно или не е).

За целта ще въведем понятието „тест“.

Дефиниция: Тест ще наричаме експеримент с резултат. Резултата е булева функция, която е дефинирана винаги, когато експеримента е извършен и която не зависи от начина, по който е извършен експеримента. Самия експеримент ще наричаме условието на теста.

Идеята е, всеки път когато експеримента е проведен да имаме резултат и този резултат да е ДА или НЕ. Не искаме резултата да зависи от миналото и от бъдещето, защото има много възможни развития за миналото и за бъдещето. Затова нека резултата зависи само от настоящето.

Настояще ще наричаме това, което виждаме в момента (момента, който ни интересува, а не текущия момент, защото текущият момент е един, но ние се интересуваме от всички моменти). Тоест, настоящето е v_q. Ние виждаме това, но виждаме и още нещо. Виждаме кои от действията са коректни в този момент и кои не са. Към вектора v_q можем да добавим булеви променливи, по една за всяко действие. Тези променливи ще са видими. Стойността на всяка една от тях ще ни казва дали съответното действие е коректно в този момент.

Забележка: Когато пишем теоретична статия избираме тази конструкция, която е най-лесна за описване. Тук статията е практична и затова ще изберем конструкцията, която е най-подходяща за реализация. Затова вместо да описваме възможните ходове, ще опишем обобщените ходове (виж [8]). Идеята е да не си играем на дребно с отделните ходове, а да оперираме с цели групи от ходове. Ще виждаме по две булеви променливи за всеки обобщен ход. Тази променливи ще се казват *all* и *nobody*.

В [8] написахме някои аргументи, които показват, че можем да приемем, че резултата от експеримента има вида x_i=*constant*. Тук x_i е някоя от видимите променливи, а *constant* е някоя от възможните стойности на тази променлива.

Забележка: В [8] решихме, че няма да пробваме задължително всички възможни ходове, за да видим кои от тях са коректни. В примера, който дадохме, възможните ходове са 36. Ако на всяка стъпка ги пробваме всичките, би било досадно. Затова, ако съответната видима променлива е някоя, от тези които се казват *all* и *nobody*, то тогава ще предполагаме, че условието на теста ни гарантира, че нужните ходове са пробвани. Ако стойността на променливата е true, то тогава трябва да са пробвани всички ходове от групата. Ако стойността е false, тогава трябва да е пробван поне един ход от групата, но такъв ход, който да показва, че не е true.

Тестови функции

Всеки тест ни определя една функция върху локалните истории.

Дефиниция: Функция на теста ще наричаме функцията дефинирана в моментите, в които условието се е случило. Стойността на тази функция ще бъде равна на резултата на теста.

Искаме да продължим тестовата функция, така че да имаме някаква прогноза за моментите, когато тя не е дефинирана.

Дефиниция: Теория на теста ще наричаме функция, която за всяка локална история връща две числа (прогноза и увереност). Ще предполагаме, че когато функцията на теста е дефинирана, теорията връща като прогноза стойността на функцията с увереност единица. В останалите случай увереността ще е по-малка от единица.

Прогнозата обикновено е нула или едно, но тя може да бъде и между тези две стойности. В този случай имаме прогноза за резултат с някаква вероятност.

Всяка тестова функция определя по едно частично свойство. Ще го наречем най-малкото свойство на теста. Множеството, в което това свойство е дефинирано, е експерименталното свойство (тук експеримента е условието на теста). Стойността на частичното свойство е стойността на резултата от теста.

Забележка: Най-малкото свойство на теста е едно обобщение на функция на теста, защото то е дефинирано за някои моменти, за които функцията не е дефинирана.

Можем ли да продължим най-малкото свойство на теста до някое тотално свойство? Да, можем, но ние можем да го продължим по много различни начини. Целта е да го продължим по естествен начин така, че полученото свойство реално да описва света.

Дефиниция: Свойство на теста ще наричаме всяко продължение на най-малкото свойство на теста.

В [8] разгледахме примера, където теста беше „Заключена ли е вратата?“. В моментите, когато сме провели експеримента, знаем отговора. Тоест, знаем тестовата функция. За състоянията, около които теста може да бъде направен, може да дефинираме коректно какъв би бил правилния отговор (макар че ние няма да знаем този отговор, ако не сме провели теста). Има ли свойство на света, което да описва резултата от провеждането на този тест? Нека имаме свойството „Вратата е заключена“ и това свойство да се променя по някакви правила. Изглежда сякаш само трябва да определим това свойство и така ще можем да прогнозираме резултата от този тест. В действителност, в нашия свят може да имаме много различни врати и тогава няма да е подходящо да опишем състоянието на света само с едно свойство. Разбира се, бихме могли да кажем: „Вратата до която се намираме е заключена“. Това е едно свойство, което е дефинирано винаги, освен когато не сме до никоя врата. Въпреки това, по-доброто описание на света би било, ако въведем много свойства (по едно за всяка врата). Така стигаме до дефиницията на тестово състояние:

Дефиниция: Нека сме разбили множеството от обобщените състояния на света на групи, които ще наричаме групи на относителна устойчивост. Тестовото състояние ще бъде една булева функция, която е дефинирана за всеки момент и за всяка една от тези групи.

Грубо казано, тестовото състояние ни казва кои врати са заключени и кои отключени в момента.

Дефиниция: Теория на тестовото състояние ще наричаме функция, която за всяка локална история и за всяка група на относителна устойчивост връща две числа (прогноза и увереност).

Забележка: Може да имаме група на относителна устойчивост, в която теста да е невъзможен. Тогава се предполага, че теория на тестовото състояние ще даде прогноза за тази група с увереност нула.

Как от теорията на тестовото състояние можем да получим теория на теста. Тоест, как от това, че имаме идея за това кои врати са заключени и кои отключени ще получим идея за това дали вратата, пред която сме, е заключена. Първо трябва да си отговорим на въпроса пред коя врата сме (тоест в коя от групите на относителна устойчивост се намираме). След това ще вземем предсказанието за тази врата, което теорията на тестовото състояние ни дава.

Групите на относителна устойчивост ще ги представяме като състоянията на краен автомат. Ако автомата е детерминиран, ще знаем точно в коя група сме. В противен случай ще знаем приблизително. Ако автомата е детерминиран, то тестовото състояние може да бъде изразено, чрез няколко теста и свойствата на тези тестове. Условията на тези тестове ще бъдат условието на теста плюс това, че сме в съответното състояния на автомата. Затова предпочитаме автомата да е детерминиран. В противен случай условието, че сме в съответното състояние на автомата няма де е изчислима функция.

Тестови състояния

Дадохме формална дефиниция на това какво е тестово състояние. Нека кажем каква е връзката между тестовите състояния и дефиниция 3.

Ако условието на теста е общовалидното условие, тогава тестовите състояния са точно видимите променливи. Това последното не е съвсем точно, защото тестовете са булеви, а видимите променливи имат k възможни стойности. За да се отстрани тази неточност, временно ще предполагаме, че видимите променливи също са булеви.

Нека имаме един тест, чието условие е общовалидното и който връща стойността на първата видима променлива. Тогава състоянието на този тест ще бъде първата видима променлива. Разбира се, това не е една променлива, защото всяко стандартно състояние си има първа видима променлива. Става дума за много променливи (може би дори за безбройно много). Много от тези променливи може да са равни. Може дори те всичките да са равни. Тогава тестовото състояние няма да се състои от всичките тези променливи, а ще вземем по една променлива от всяка група равни променливи.

Тоест видяхме, че ако имаме свят по дефиниция 3 неговите видими променливи представляват тестови състояния. Нека да направим обратното. Нека да вземем свят по дефиниция 2 и някакъв тест, чийто резултат е първата видима променлива. Нека да построим свят по дефиниция 3, чиято първа видима променлива да е точно тестовото състояние на този тест.

За целта ще разбием състоянията на света на групи на относителна устойчивост. Кое точно разбиване да изберем? Което и разбиване да вземем, ще ни свърши работа, но хубаво е разбиването да е такова, че действително да отговаря на света и това действително да са групи на относителна устойчивост.

Нека сега, всяка група на относителна устойчивост да бъде едно стандартно състояние. Ще добавим още невидими променливи, ако е нужно, за да запаметим в тях цялата информация, която имаме за обобщеното състояние. Ще дефинираме функциите View и World така, че получения свят да е еквивалентен на този, от който тръгнахме.

Забележка 1: Можем да представим една група на относителна устойчивост, чрез няколко стандартни състояния. Това разбиване е полезно макар да не е задължително. С това разбиване света може да стане много по-прост. Например, вместо да си мислим, че всички постоянно заключени врати са в едно стандартно състояние, може би ще е по-просто, ако имаме различни състояния отговарящи на различни постоянно заключени врати.

Какво би се случило, ако изберем неподходящо разбиване? Нека си представим, че сме разбили множеството на вратите на метални и на дървени. Предполагаме, че ако една от металните врати е заключена, то тогава всички метални врати са заключени. Аналогично и за дървените врати. Тогава виждаме заключена метална врата и заключаваме, че всички метални врати в момента са заключени. В следващият момент виждаме отключена метална врата. Решаваме, че сега всичките са отключени. Възможно ли е това наистина да е така? Възможно е и няма как тази възможност да бъде потвърдена или отхвърлена експериментално. Въпреки това, ако разбиването не е адекватно, групите на относителна устойчивост ще са доста неустойчиви.

Примери за тестови състояния

Нека вземем света, който описахме в примера (играта шах). Нека предположим, че описанието на този свят е по дефиниция 2. Ще разгледаме два теста и ще видим как чрез техните тестови състояния света може да се разбие на групи на относителна устойчивост и по този начин да представим света като свят по дефиниция 3.

Първият тест ще бъде „Виждам бяла фигура“. Условието на теста ще бъде общовалидното условие (Тоест, кога проверявам дали виждам бяла фигура. Винаги проверявам.) Резултата на теста ще бъде color=white.

Кои ще са групите на относителна устойчивост? Това ще са квадратчетата на таблото. Тестовото свойство ще бъде „Има бяла фигура в квадратчето, което

гледаме“. Тестовото състояние ще бъде позицията на таблото. Е, не точно позицията, а само това на кои квадратчета има бяла фигура. Тестовото състояние няма да съдържа информация за това къде са черните фигури, нито за това коя точно е бялата фигура в квадратчето.

Получихме свят с 64 стандартни състояния. Видимите променливи са ясни. Трябва да добавим невидими променливи, в които да запишем информацията за обобщеното състояние, която я няма във видимите променливи. Това може да го направим по същия начин, както вече го направихме.

Втория тест ще бъде „Ако виждам бяла фигура, то мога да я вдигна.“

Резултата на теста ще бъде „Вдигни фигурата е коректен ход.“ Да отбележим, че „вдигни фигурата“ не е един ход, а е цяла група от ходове (т.е. обобщен ход), защото вдигайки фигурата може едновременно с това да се преместим по хоризонтала или по вертикала. Тоест, резултата на теста е „В групата от ходове <*, *, вдигни фигурата> има поне един коректен ход.“ Няма как да искаме всичките ходове от групата да са коректни, защото ход може да е некоректен заради движението (например ако сме в лявата колона и се опитваме да се преместим на ляво). Тази група си има една видима променлива *nobody*, която трябва да е лъжа. Формално записано ще изглежда така: *nobody(<*, *, „вдигни фигурата“>)=false*. Тук е записано, за кой обобщен ход е *nobody*, защото различните обобщени ходове си имат различни променливи *nobody*.

Условието на теста ще бъде „Виждам бяла фигура“ (т.е. color=white). Към условието на теста трябва да добавим това, че сме пробвали ходовете от групата. Ако *nobody* е *true* сме пробвали всичките ходове, ако *nobody* е *false* сме пробвали поне един ход, но такъв, който е коректен.

Тук групите на относителна устойчивост са две. Такива състояния, за които теста винаги ще върне истина и такива, за които винаги ще върне лъжа. Ще имаме лъжа, когато сме вдигнали друга фигура и още не сме я пуснали и когато играта е свършила и още не сме играли „New Game“, за да започнем нова.

Как ще представим света по дефиниция 3? Както отбелязахме в забележка 1 една група на относителна устойчивост може да се представи с повече от едно стандартно състояние. Тук групата, в която теста връща винаги лъжа ще я представим с две стандартни състояния (когато сме вдигнали фигура и когато играта е свършила). Да отбележим, че няма как хем да сме вдигнали фигура, хем играта да е свършила.

Получихме свят с три стандартни състояния. Видимите променливи са ясни. Трябва да добавим невидими променливи, в които да запишем цялата информация за обобщеното състояние. Тоест, трябва да запишем позицията на таблото (за това ще ни трябват 64 променливи или 2.64, ако сме по-разточителни). Трябва да запишем още координатите на квадратчето, в което се намираме и още, ако сме вдигнали фигура, от къде сме я вдигнали и коя е тази фигура.

Как намираме теориите?

Каква е разликата между теорията на теста и теорията на тестовото състояние? В първия случай предполагаме, че имаме само една група на относителна устойчивост и че теста ни определя едно свойство. Във втория случай предполагаме, че имаме много групи на относителна устойчивост и че теста ни определя много свойства на света (по едно за всяка група).

Ще кажем как определяме теорията на теста. (Теорията на тестовото състояние се определя по аналогичен начин.)

Събрали сме статистика за определени експерименти. Когато експеримента и теста едновременно са проведени броим колко пъти теста е ни е върнал ДА и колко пъти е върнал НЕ. Нека това да са числата n и m. Тогава прогнозата, която ни дава този експеримент е $\frac{n}{n+m}$, а увереността зависи от $n+m$. В даден момент много експерименти са проведени, всеки от които ни дава някаква прогноза с някаква увереност. Тук няма да обсъждаме как се изчислява каква е общата прогноза и общата увереност.

Забележка: Когато едно тестово състояние е замърсено с шум (дефиниция 4), тогава няма как да намерим правила, които да дават точна прогноза (т.е. прогнозата да бъде цяло число). Всяка прогноза, която е с достатъчно голяма увереност, ще е приблизителна (т.е. тя ще е някаква вероятност p такава, че $0<p<1$). Обратното, ако прогнозата е приблизителна, няма как да знаем дали това е защото резултата на теста е замърсен с шум или защото условието на теста просто е такова, че дава приблизителна прогноза. Ако всички прогнози са приблизителни, може да предположим, че имаме някакъв шум или че още не сме открили правилото, което ще ни даде точна прогноза.

Освен чрез експериментите ще предсказваме свойството на теста още на базата на предположението, че това свойство е устойчиво. Ще предполагаме, че сме събрали статистика за това, колко устойчиво е това свойство. На базата на това ще предполагаме, че след като сме проверили свойството (направили сме теста), стойността на свойството продължава да е същата още известно време. Увереността на това предположение ще намалява с времето (т.е. колкото повече стъпки са изминали от проверката, толкова по-малко ще разчитаме, че стойността е същата).

Следващото ниво е да предполагаме, че в света живеят и други агенти, които променят свойството по свое усмотрение (виж [8]).

Как определяме групите на относителна устойчивост, за да дефинираме такова тестово състояние, което да е смислено и адекватно. Отново при помощта на статистиката. Всъщност това е задачата да намерим краен автомат, който смислено да разделя състоянията на света на групи. Тази задача също ще остане извън темите разгледани в тази статия.

Заключение

Тази статия започна с претенцията да бъде различна и да предложи за намирането на AI нещо повече от апроксимация. Въпреки това, тук отново се използва метода на апроксимацията. Когато имаме тестова функция и искаме да я продължим до някакво тестово свойство, тогава това което правим на практика е апроксимация. Въпреки всичко, има разлики. Повечето автори търсят една апроксимационна функция, която трябва да е решението на проблема (т.е. тя трябва да е AI). Ние тук търсим не една, а много апроксимационни функции (по една за всяко свойство). Тези функции не са самото AI. Тяхната задача е само да помагат на устройството да разбере какво се случва в момента.

Забележка: Дето се вика, ако имахме Full Observability и виждахме всичко, то тези апроксимации въобще нямаше да ни са нужни. Въпреки, че търсенето на тестови свойства е насочено към случая с Partial Observability, това би ни било полезно дори и в случая с Full Observability. Проблемът при Full Observability е, че в този случай ние имаме твърде много информация и е много трудно да отличим съществените от несъществените. Нека да забравим какво виждаме и да се съсредоточим само на тестовите свойства, които сме намерили. Ние ще отделим тези тестови свойства, които са интересни и точно това ще е съществената информация, която ще ни е нужна.

Както казахме, при нас апроксимацията не е цялото решение, а само част от решението. Преди това трябва да съберем статистика, за да има на базата на какво да апроксимираме. Тук трябва да спомена зависимостите със и без памет, за които се говори в [8]. След това правим апроксимация на базата на експериментите и на статистиката, която сме събрали за тях. Също така използваме предположението за устойчивост на тестовите състояния (това също може да бъде прието за някаква апроксимация). Следващият метод, който използваме за определянето на стойността на тестовите състояния е допускането на предположението, че в същия свят освен нас има и други агенти и те променят тези тестови състояния с цел да ни помогнат или да ни прецакат. Този подход вече не може да бъде наречен апроксимация, най-малкото защото няма формула, с която да бъде изчислен. Тоест, описаното в тази статия е нещо повече от алгоритъм за апроксимация.

Дори след намирането на тестовите състояния задачата още не е решена, защото остава да се планират бъдещите действия, за да бъде получен максимален резултат.

Забележка: Тук си поставяме задачата да опишем един конкретен алгоритъм и това е алгоритъма на мислещата машина. Тук не си задаваме въпроса какво е AI, нито въпроса каква е дефиницията на AI. Тези въпроси вече са разгледани в [5] както и в други по-ранни статии. Затова в [5] използваме класическата дефиниция на Reinforcement Learning, докато в тази статия сме предложили три други еквивалентни дефиниции. Когато искаме да дадем теоретична дефиниция на AI, тогава класическата дефиниция на Reinforcement Learning ни е достатъчна, но когато искаме да опишем AI достатъчно подробно, за да доведе това до конкретна реализация, тогава ни е нужно да променим дефиницията, така че по-лесно да работим с нея.

Защо тестовите свойства и тестовите състояния са толкова важни? Именно тези свойства и тези състояния ни дават разбирането на света. Да разберем света означава да имаме идея за неща, които не виждаме директно. Например, да знаем, че вратата на третия етаж е отключена. За да формулираме това твърдение на нас ни е нужен съответния тест. За да решим дали твърдението е вярно ще ни трябва теория на тестовото състояние на този тест.

Много изследователи в областта на AI споделят мнението, че AI трябва да може да прави логически изводи на базата на някоя система за автоматично доказателство на теореми. Например съждителното или предикатното смятане (propositional or predicate calculus). Ние също споделяме това мнение, но въпросът е как от последователността (действие, наблюдение) да се стигне до съждително или до предикатно смятане? За да имаме съждително смятане, на нас са ни нужни съждения. За да направим предикатно смятане, ще са ни нужни предикати. Ако не можем да разберем последователността (действие, наблюдение), то за нас тя би била просто шум. Как от тази последователност да изведем съждения и предикати? Свързващото звено, което ни е нужно са тестовите свойства и тестовите състояния. Например тестовото свойство „Вратата е отключена“ може да бъде търсеното от нас съждение. Тестовото състояние „Вратата X е отключена“ ще бъде предиката, от който можем да направим твърдения от вида „Всички врати са отключени“.

References

[1] Boris Kraychev and Ivan Koychev. Computationally Effective Algorithm for Information Extraction and Online Review Mining. In Proc. of International Conference on Web Intelligence, Mining and Semantics June 13-15, 2012, Craiova, Romania, ACM.

[2] Tatiana Kosovskaya. Self-Modificated Predicate Networks. International Journal “Information Theories and Applications”, Vol. 22, Number 3, 2015, *pp.245-257*.

[3] Richard Sutton and Andrew Barto. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 2017. ISBN 0-262-19398-1.

[4] Dimitri Bertsekas and John Tsitsiklis. Neuro-Dynamic Programming. Nashua, NH: Athena Scientific, 1996. ISBN 1-886529-10-8.

[5] Dimiter Dobrev. Comparison between the two definitions of AI. *arXiv:1302.0216, January, 2013.* http://www.dobrev.com/AI/

[6] Dimiter Dobrev. Giving the AI definition a form suitable for engineers. *April, 2013.* http://www.dobrev.com/AI/

[7] Dimiter Dobrev. The algorithm of the Thinking machine. *viXra:1605.0190, May, 2016.* http://www.dobrev.com/AI/

[8] Dimiter Dobrev. Incorrect Moves and Testable States. *viXra:1705.0108, May, 2017.* http://www.dobrev.com/AI/

[9] Dimiter Dobrev. Incorrect Moves and Testable States. International Journal "Information Theories and Applications", Vol. 24, Number 1, 2017, *pp.85-90*. http://www.foibg.com/ijita/vol24/ijita-fv24.htm

[10] Dimiter Dobrev. Incorrect Moves and Testable States. In Proc. of 11th Panhellenic Logic Symposium, July 12-16, 2017, Delphi, Greece, *pp.65-70.* http://pls11.cs.ntua.gr/