Българска академия на науките
Институт по математика и информатика
Секция „Алгебра и логика“

Изкуствен Интелект – дефиниция, реализация и последствия

Какво е това, как да го направим и какво ще правим
след като го направим?

Димитър Димитров Добрев

основната част на дисертационния труд
за зачисляване на свободна аспирантура

по професионално направление „Информатика“ (4.x)
научна специалност „Математическа логика“

Научен ръководител: доц. д-р Любомир Иванов

София 2018

Дисертацията съдържа 131 страници, от които 6 страници уводни раздели, 120 страници основен текст и 5 страници заключителен материал, включително библиография с 42 заглавия.

Съдържание

Увод

В този дисертационен труд ще разгледаме три фундаментални въпроса. Нямаме претенции да сме решили втория и третия въпрос, но претендираме, че сме отговорили на първия. Тоест, твърдим, че в тази дисертация се съдържа ясна, точна и конкретна формална дефиниция на понятието Изкуствен Интелект.

По отношение на въпроса „Как да направим ИИ?“, макар тази дисертация да не дава отговор на този въпрос, тук ще намерите много резултати, които дават частичен отговор. Много важно е това, че преди да си отговорим на въпроса „Как да го направим“ ние сме отговорили на въпроса „Какво искаме да направим“. Повечето изследователи в областта на ИИ се опитват да създадат ИИ без да са имат ясна представа какво е това ИИ. Това е все едно да се опитваш да направиш палачинка с шоколад без да знаеш какво е палачинка и какво е шоколад.

Що се отнася до третия въпрос, неговия отговор ще ни е нужен чак когато вече сме отговорили на втория въпрос. Това не значи, че трябва да чакаме появата на ИИ и чак тогава да си зададем въпроса „И сега какво ще правим?“. Ако чакаме до момента, в който този въпрос стане наложителен, ще допуснем огромна грешка. Когато ИИ се появи, вече ще е много късно да се питаме: „И сега накъде?“ Този въпрос ние трябва да сме си го задали много преди реалната поява на тази машина (т.е. на тази програма, защото ИИ не е машина, а е програма, както ще видите по-долу).

Тази дисертация съдържа три части, които са посветени на трите въпроса, които тази дисертация разглежда.

Първата част отговаря на въпроса „Какво е ИИ?“ Този част се състои от две глави озаглавени съответно „неформална дефиниция“ и „формална дефиниция“. По същество тези две глави са статиите [1, 2] и [3].

Втората част разглежда въпроса „Как да направим ИИ?“. Този част се състои от пет глави, които по същество са статиите [4 - 8].

Третата част се състои от две глави. Първата е статията [9], а втората глава предстои да бъде допълнена и издадена като отделна статия.

1 Какво е Изкуствен Интелект?

1.1 Неформална дефиниция

Трябва ли ни да знаем какво е AI? На този въпрос ще отговорим просто: Да, ако искаме да го открием, то определено ще е по-лесно да го намерим, ако знаем какво търсим. В противен случай ще се окажем в положението на Алхимиците, които са търсили Философския камък, но почти не са имали представа какво е това.

Най-известната дефиниция на AI е така наречения Тюрингов тест. Тюринг е английски математик известен както с Машините на Тюринг, така и с разбиването на немските кодове по време на Втората Световна Воина.

Тюринговия тест е доста прост. Поставяме нещо зад една завеса и то разговаря с нас. Ако не можем да го различим от човек, то това е AI. Тази дефиниция е по-стара от петдесет години и затова ще се опитаме да направим нова, по-съвременна.

Дефиницията на Тюринг предполага, че Интелект е човек с натрупаните през годините знания. А Интелект ли е бебето, което току що се е появило на бял свят? Нашият отговор ще е: Да. Тоест нашата дефиниция за Интелект ще е това, което не знае нищо, но може да се научи. Тук се различаваме от повечето хора, които като чуят Интелект, си представят професор от университета.

Преди да дадем формална дефиниция на AI ще направим уговорката, че приемаме тезиса на Чърч, който гласи, че всяко изчислително устройство може да се моделира с програма. Т.е. ние ще търсим AI в множеството на програмите. Ще предположим, че AI е стъпково устройство, което живее в някакъв свят, на всяка стъпка получава информация (от света) и въздейства (на света), чрез информацията, която извежда. Ще допуснем още, че информацията, която получава и предава на всяка стъпка е крайно много. Например получава *n* бита и предава *m* бита.

След тези уговорки можем да изкажем неформално нашата дефиниция. AI ще наричаме такава програма, която в произволен свят би се справила не по-зле от човек.

Следващата задача ще е да формализираме тази дефиниция, за да можем да я използваме и с нея да търсим AI. Първо, какво за нас ще е свят? Това ще са две функции **World(s, d)** и **View(s)**. Първата ще взима като аргументи състоянието на света и въздействието, което оказва на света нашето устройство. Като резултат тази функция ще връща новото състояние на света (което той ще има на следващата стъпка). Втората функция ще ни казва, какво вижда нашето устройство. Аргумент на тази функция ще е състоянието на света, а получената стойност ще е информацията, която ще получава устройството (на дадена стъпка). Трябва да добавим и един елемент $\mathbf{s_0}$, това ще е състоянието на света, когато нашето устройство се е родило. По време на живота му света ще премине през състоянията $\mathbf{s_0}$, $\mathbf{s_1}$, $\mathbf{s_2}$, ... Устройството ще въздейства на света с информацията, която извежда на всяка стъпка $\mathbf{d_0}$, $\mathbf{d_1}$, $\mathbf{d_2}$, ... Също така, AI ще получава информация от света $\mathbf{v_0}$, $\mathbf{v_1}$, $\mathbf{v_2}$, ... Ясно е, че $\mathbf{s_{i+1}=World(s_i, d_i)}$ и че $\mathbf{v_i=View(s_i)}$.

До тук имаме всичко. Имаме свят и устройство, което живее в него. Липсва само едно, смисъла на живота. Какво е живота без болка и радост, ще кажат философите. За това ще въведем смисъл на живота. Това ще е оценка, която ни казва дали една редица $\mathbf{v_0}$, $\mathbf{v_1}$, $\mathbf{v_2}$, ... е по-добра от друга.

За повечето хора живота е протекъл по-добре, ако са видели повече швейцарски курорти и по-малко каменовъглени мини. Горе-долу и ние така ще дефинираме смисъла на живота. Ще отделим от $\mathbf{v_i}$ два бита, които ще наречем победа и загуба и смисълът ще е да се получат повече победи и по-малко загуби.

Остана последната стъпка, да се даде алгоритъм, който ще открие AI. Идеята ще е да пуснем всички програми върху всички светове и да отделим тези, които се справят най-добре. Това не звучи като алгоритъм, все едно да проведем безкраен изпит с безброй много кандидати. Това че кандидатите са безброй много не е проблем. На нас не ни трябват всички, а само някои от издържалите изпита. Обаче това, че изпита е безкраен и никога няма да свърши за нито един кандидат е проблем.

За да направим изпита краен, ще ограничим световете до краен брой. Но дори само за един свят изпита би бил безкраен, затова ще добавим изисквания за ефективност. Ще кажем за всеки от световете колко време (такта) даваме на устройството за обучение и какъв успех трябва да постигне след като се е обучило (например в следващите сто такта съотношението на победи към загуби да е поне 9 към 1). Още ще ограничим времето и паметта, които устройството ще може да използва на една стъпка. Тези изисквания не

трябва да са твърде крути, защото нашия AI няма да ги удовлетвори и ще пропадне на изпита.

Ще предполагаме още, че в световете, които сме подбрали за изпита няма фатални грешки. Ако имаше, то търсеният AI би могъл да направи такава грешка докато се обучава и да пропадне на изпита. Разбира се, ако този AI живее в този свят много пъти, то средно ще постигне една добра оценка, но ние искаме да го пускаме само по веднъж на свят.

Реалният свят не отговаря на това изискване, защото може да си Айнщайн, но да бъдеш изяден от някоя мечка, още преди да си осъзнал теорията на относителността. Т.е. да допуснеш фатална грешка преди да си се обучил. Пример за свят без фатални грешки е, ако нашето устройство играе шах срещу някакъв противник. След края на всяка партия започва следваща. В този свят AI може да загуби много партии, но това няма да се отрази на следващите.

След като направихме изпита краен, то сега нашият алгоритъм може да започне да генерира програмите една по една, да подлага всяка на изпита и да извежда тези, които са го издържали. Ще считаме, че нашият алгоритъм изрежда програмите по сложност (т.е. по дължина). Т.е. първо ще изведе най-простата (късата) програма, от тези които издържат изпита. Дали тази програма ще е AI или нашия AI ще се генерира по-късно? Тук трябва да отбележим, че не всички програми издържали изпита са AI. Например, ако напишем програма специално за световете, върху които тестваме, то тя ще мине, но няма да е AI. Този проблем го имаме и с кандидат студентските изпити. На тях минават много хора, които са зазубрили всички типове задачи, но това не са интелекти, а зубрачи.

Ще считаме, че световете които сме включили в изпита са достатъчно много и достатъчно разнообразни (по-големия брои задачи не утежнява изпита за преподавателя, защото повечето кандидати ще пропаднат още в началото). При това положение ще се отсеят почти всички програми, които не са AI. Следователно, първата (най-простата) която ще се изведе ще е AI, а тази която е написана само за световете от изпита, ще е по-сложна и ще се изведе по-нататък.

Значи вече разполагаме с алгоритъм за откриването на AI! Значи ли това, че всеки сносен програмист може да напише програма по него, да я пусне, да почака малко и тя ще му изведе търсения AI. Отговорът е да, така е, но времето което ще се наложи да почакате ще е доста (да кажем сто хиляди години). Причината за това се крие в явлението наречено комбинаторен взрив. Не е трудно да се напише програма, която се очаква да спре след смъртта на вселената (например, прибавяйте единица към десет байтов брояч докато не се препълни).

Горното означава, че описаният алгоритъм е напълно безполезен, но не така стои въпросът с дефиницията на AI. Сега след като знаем какво е това AI можем да се опитаме да го построим директно и да използваме алгоритъма, за да се убедим, че това наистина е AI.

След като нашето устройство е издържало изпита във всички тестови светове, можем да го пуснем и в нашия (реалния) свят. Разбира се, то няма да е готово веднага да премине през теста на Тюринг, защото в най-добрият случаи зад завесата ще се чуе само едно гукане. Нашето устройство ще се нуждае първо от възпитателки и гувернантки, които да го научат на добри обноски. Учителите ще трябва да го поощряват като подават единица на шината му - победа и да го наказват чрез шината загуба. Така нашия AI ще се труди прилежно стараейки се да получи максимално много поощрения и минимално наказания. Може би тогава няма да програмираме компютрите, а ще ги възпитаваме и обучаваме. И може би един ден, след като ги възпитаме и обучим, ще станем излишни.

1.2 Формална дефиниция

Определяме какво е AI чрез изпит. Като резултат от изпита ще получим оценка. Тази оценка ще наречем IQ. Приемаме, че всички програми, чието IQ е над определено ниво удовлетворяват дефиницията за AI.

За да обясним идеята ще използваме аналогията с кандидатстудентските изпити. Задачите за изпита избираме случайно, но на всички студенти даваме едни и същи задачи. При това, задачите трябва да са логични, защото искаме да приемем студентите, които мислят логично, а не тези които случайно са уцелили правилния отговор. Оценката определяме на базата на това, колко от задачите кандидат-студента е решил. Не можем да кажем колко задачи трябва да бъдат решени, за да бъде приет студента, защото не знаем колко хора ще се явят на изпита и как ще се представят. Бихме могли да фиксираме една оценка (например 5.50) и да кажем, че възнамеряваме да приемем всички кандидат-студенти получили повече от 5.50. По-добре е, вместо да фиксираме минималната оценка, да я определим в последствие, когато изпита е приключил. Тогава взимаме оценката на този, който е на определена позиция (например, който е на позиция 100 по успех и така приемаме първите 100).

Тази аналогия добре описва изпита за AI, но когато провеждаме изпит между програми не можем да отделим първите 100, които са се представили най-добре, защото в този случай кандидатите са безбройно много.Може би по-добра е аналогията, че провеждаме конкурс за директор и изпита се проточва във времето. Спираме когато някой кандидат събере достатъчно точки. Колко точки са достатъчно? Може предварително да сме избрали определено ниво, но може да променим нивото в последствие, ако предварително избраното ниво се окаже прекалено ниско или прекалено високо.

В [14] вече направихме подобна конструкция, където на различните програми се дават различни оценки ($IQ \in [0,1]$). Тук ще повторим тази конструкция, за да я подобрим. Причините, поради които статията [14] се нуждае от подобрение, са следните:

1. В [14] се разглеждат едновременно въпросите „Какво е AI?“ и „Как можем да създадем AI?“ Не е добра идея да се объркват въпросите „какво“ и „как“. Тук ще се ограничим само с въпроса „Какво е AI?“ и няма да се занимаваме с въпроса как да намерим тази програма.

2. В [14] дефинираме AI като програма, а тук ще дефинираме AI като стратегия. В [14] AI е програма и света, в който AI живее, също е програма. Така имаме две програми, които играят една срещу друга, което е малко объркващо. По-добре е да дефинираме AI като стратегия и да имаме програма играеща срещу стратегия. Разбира се, тази стратегия ще е изчислима, защото е крайна. Ще наречем AI програма, всяка програма, която реализира AI стратегия за първите хиляда игри (партии). Какво ще прави AI програмата след хиляда игри няма да е определено, но се надяваме, че тя ще продължи да се държи интелигентно и по-нататък.

3. В [14] за представянето на света се използват недетерминирани Тюринг машини. Това е едно излишно усложняване. То би имало смисъл, ако нямаше връзка между отделните игри (партии) и ако след всяка игра лентата на машината се изчиства (нулира). Тъй като лентата не се изчиства, всяка следваща игра зависи от това, което е станало в предишната игра (това което е останало върху лентата). Затова ще използваме детерминирани Тюринг

машини, те са по-прости, а това че зависят от това което е останало върху лентата прави поведението им различно (недетерминирано) в различните игри.

4. На всяка стъпка имаме действие и наблюдение. В [14] действието и наблюдението са по една буква, а тук действието ще е *n* букви и наблюдението ще е *m* букви. Разбира се, ние бихме могли да кодираме няколко букви с една, но това противоречи на идеята, че трябва да избягваме излишното кодиране [4]. Света е достатъчно сложен и е трудно да го разберем. Ако добавим едно излишно кодиране, света ще стане още по-неразбираем.

5. В [14] се предполага, че всички ходове са коректни, докато тук ще добавим понятието некоректен ход. От една страна, важно е да предполагаме съществуването на некоректни ходове. От друга страна,така ще се отървем от произволното прекъсване на работата на машината на Тюринг, което правим в [14], за да избегнем зацикляне.

6. Машината на Тюринг е теоретичен можел за представяне на изчисление, който не се нуждае от ефективност. Тук ще използваме този модел за реална работа и затова ще го променим, като го направим много по-ефективен. Цената за тази по-добра ефективност ще бъде усложняването на модела.

7. В [14] използваме машината на Тюринг, за да опишем един логичен свят. Щом е изчислим, значи е логичен. Въпреки това света, който описва машината на Тюринг, не е много логичен. Всичко се записва върху една лента и програмата въобще не е структурирана. Произволно се скача от команда на команда (това програмистите го наричат „спагети код“). Работата на подобна програма е доста нелогична, затова ще я променим като добавим подпрограми и ще направим машината многолентова.

8. В [14] дефинираме IQ като едно средноаритметично, което не може да бъде изчислено точно, поради комбинаторната експлозия. Там казваме, че то може да бъде изчислено приблизително, чрез статистическа извадка. Тук ще въведем термините глобално IQ, което не може да бъде изчислено точно и локално IQ, което ще бъде лесно изчислимо и ще се изчислява чрез конкретна статистическа извадка. Локалното IQ ще клони към глобалното IQ, когато размерът на статистическата извадка клони към безкрайност. Множеството на световете, които използваме за изчисляването на глобалното IQ е крайно (огромно, но крайно). Въпреки това размерът на статистическата извадка може да клони към безкрайност, защото в тази извадка може да има повторения (макар че повторение е малко вероятно поради огромния размер на множеството, от което се избира тази извадка).

1.2.1 Литературен обзор

Тюринг в [20] предлага своята дефиниция за AI (теста на Тюринг). Идеята е, че ако една машина успешно може да имитира човек, то тази машина е AI. При теста на Тюринг, както и в нашата статия, имаме изпит и за да бъде призната една машина за AI, тя трябва да издържи изпита. Една разлика е, че там има изпитващ, докато при нас имаме тест с фиксирани въпроси. Тоест, теста на Тюринг е субективен, поради което той е една неформална дефиниция на AI. Все пак, основният проблем на дефиницията на Тюринг не е в нейната субективност и неформалност, а в това, че тя не дефинира интелект, а нещо повече. Тя дефинира образован интелект. За да издържи един интелект теста на Тюринг,

той трябва да е образован. Дори може да предположим, че той има англосаксонско образование, защото ако не владее английски той не би се представил добре на теста.

Интелект и образован интелект са две различни неща, както различни неща са компютър без софтуер и компютър със софтуер. Ако попитате един математик, какво е компютър, той ще ви отговори: „Машина на Тюринг“. Ако зададете същия въпрос на едно дете, то то ще ви каже: „Компютърът е нещо, с което могат да се играят игри, да се гледат филми и т.н.“ Тоест, математика възприема компютъра само като хардуер, а детето възприема компютъра като неделима система от софтуер и хардуер. Когато Тюринг дава дефиниция на компютър (машината на Тюринг), той описва само хардуера, но когато дефинира интелект, той описва една неделима система от интелект и образование.

Макар теста на Тюринг да описва образован интелект, самият Тюринг много добре разбира разликата между образован и необразован интелект. В края на [20] той задава въпроса „Instead of trying to produce a program to simulate the adult mind, why not rather try to produce one which simulates the child's?“

Дефиницията на AI, която ще дадем в тази статия отговаря на този въпрос. Тя не включва образованието и затова задачите от теста не предполагат никакви предварителни знания. По-точно, във всяка задача ще предполагаме, че започваме на чисто и по време на решаването на задачата се образоваме, т.е. откриваме зависимости и се учим от грешките си.

McCarthy в [21] казва, че отличителната черта на AI е: „ability to achieve goals in the world“. Ние няма да спорим с McCarthy, а само ще уточним това, което е казал. Ще уточним какво е свят и какво е произволен свят и какви са целите, които AI трябва да постигне. На тази база ще създадем IQ тест, при който програмите, които постигат повече цели, имат по-високо IQ.

McCarthy казва още, че няма „definition of intelligence that doesn't depend on relating it to human intelligence“. Действително, в тази статия ние изчисляваме стойността на IQ без това да е свързано с човешката интелигентност, но, за да кажем дали една програма е AI, трябва да кажем колко е минималното за целта IQ. За този минимум ние избрахме числото 0.7. Това число е избрано произволно и ние ще го променим, ако това ниво се окаже много по-ниско или много по-високо от човешкия интелект. Тоест, при нашата дефиниция човешкия интелект също се използва, но това е само за сравнение и целта е единствено определянето на една константа.

Друг въпрос зададен от McCarthy: Can we ask a yes or no question “Is this machine intelligent?” Отговорът е „не“ и ние сме напълно съгласни, защото не се знае колко е минималното за целта IQ.

Обаче, ние няма да се съгласим с отговора на следващият въпрос на McCarthy. Той пита „Do computer programs have IQs?“ и отговаря с „не“. В тази статия, както и в [14] ние показахме, че може да се дефинира IQ за програми. В отговорът си McCarthy по-скоро е искал да каже, че IQ тестовете за хора не са подходящи за компютърни програми. Теста, който ние предлагаме не е за хора, а е за програми.

В някои статии (например в [22]) се разглежда въпроса как да се създаде програма, която може да решава IQ тестове създадени за хора. В нашата статия въпросът, който разглеждаме, е обратният. Тук ние създаваме IQ тестове предназначени за програми (за стратегии). Тестовете, които ще предложим, няма да са подходящи за хора, макар да е възможно човек след известно обучение да се научи да ги решава. Обученият човек ще си записва какво се е случило и ще анализира, за да открие зависимости. Необучения човек няма да си записва и няма да успее да забележи зависимостите, освен ако те не бъдат представени във визуален вид или по друг начин, който да е удобен за възприемане от човек.

Много трудно е да си мислим за човека като за стратегия по следните причини. Първо, човека е недетерминиран, тоест, той не осъществява една детерминирана стратегия. (Човека можем да го разглеждаме като недетерминирана стратегия или като множество от стратегии, от които случайно се избира една.) Второ, не е ясно, колко време сме дали на човека, за да направи един ход. Трето, не е ясно колко сериозно ще подходи човека към задачата (ако подходи по-сериозно, ще получи по-добър резултат). Четвърто, човека винаги е обучен и никога не започва от нулата. Дори и новороденото бебе е преминало през някакво обучение в корема на майка си. Затова, когато един човек осъществява една стратегия, резултата до голяма степен зависи от неговото образование, обучение, опит.

В [23] Detterman заплашва, че ще създаде система от тестове (develop a unique battery of intelligence tests), които ще могат да измерят IQ-то на компютърните програми. В тази статия, както и в [14] ние не заплашваме, а направо създаваме такъв тест.

Малко объркващо е това, че Detterman казва „компютър“, когато иска да каже „системата от компютър и програма“. По-добре е тази система за по-кратко да се нарече програма, защото когато знаем коя е програмата, не е много важно, на кой компютър ще я пуснем, защото разликата между два компютъра е единствено в бързодействието им. Обратното, ако на един компютър пуснем две различни програми, то разликата в поведението ще е огромна.

Detterman възнамерява да тества компютърните програми с IQ тестове за хора. Тоест, и той, подобно на Тюринг, не прави разлика между интелект и образован интелект. Затова теста на Detterman няма да дава време за обучение, а ще предполага, че компютърната програма, която се явява на теста, е предварително обучена.

Detterman разчита на това, че компютрите са по-дори от хората в намирането на фактологична информация. Това е една способност на компютрите, която не е директно свързана с интелекта. По подобен начин компютрите са много силни в аритметичните операции, но това не ги прави интелигентни.

Има едно нещо, с което сме съгласни с Detterman. Той отбелязва, че с предварително запрограмирани рецепти (hoc algorithms) могат да се решат много задачи, но истинският интелект трябва да може сам да намери решението.

В [24] авторите си поставят амбициозната задача да създадат IQ тест, който да е подходящ и за AI, и за хора, и за програми като Siri и AlphaGo, които не са AI. Не можем да сравняваме AI с програми, които не са AI, защото първото е програма, която може да бъде обучена за произволна задача, а второто е програма написана за конкретна задача.

Например, как да сравним програма играеща шах с AI? Единственото което може да прави програмата играеща шах е да играе шах, докато AI може всичко, но не веднага, а след като бъде обучен. Тоест, единствения начин да сравним тези две програми е да ги пуснем да играят шах, то там програмата играеща шах ще има предимство, защото AI ще загуби време, за да се обучи, докато неговия опонент няма да има нужда да се учи как се играе шах, защото той това го може. Ако сравним програмата играеща шах с AI, който е обучен да играе шах, то тогава първата програма пак би имала известно предимство пред AI, както специализирания хардуер (т.е. хардуер направен специално за определена задача) има предимство пред компютърна програма работеща на компютър. Тоест, идеята да сравняваме AI с програми, които не са AI, не е добра.

Авторите на [24] се обръщат към понятия като: the abilities to acquire, master, create, and feedback knowledge. Това е все едно да обясняваме термин, чието значение не знаем, с други термини, чието значение на знаем.

В [25] се прави много сериозно обсъждане на въпроса за това как се мери IQ на AI. Също така, в [25] е направен много добър обзор на различните работи посветени на тази тема. Известен недостатък на [25] е това, че там има известна противоречивост. От една страна статията се казва „IQ tests are not for machines, yet“. От друга страна в по-ранни статии на Orallo [26, 27] се дава формална дефиниция на AI на базата на IQ тест. Изглежда сякаш Orallo прави стъпка назад и се отказва от предишните си резултати. Друга противоречивост в [25] е това, че от една страна се казва: „human IQ tests are not for machines“. Това е нещо, с което ние сме напълно съгласни, както сме съгласни и с доводите, които съпътстват това твърдение. От друга страна в същата статия авторите казват, че са съгласни с Detterman, че “there is a better alternative: test computers on human intelligence tests”

В [25], както и в [24], се търси универсално IQ, което да е приложимо към всички програми и дори и към хората. Тук се разминаваме с авторите на [24, 25]. Вече обяснихме защо не е добра идея да сравняваме AI с програми, които не са AI. Също така обяснихме защо не е добра идея да използваме едни и същи тестове за AI и за хора.

Много важна е статията [26], защото това е първата статия, в която се говори за формална дефиниция на AI и това е и първата статия, в която се въвежда понятието IQ-test за AI. Действително в [26] този тест се нарича C-test, но изрично се казва, че това е IQ-test за AI. Трябва да се извиним, че сме пропуснали да цитираме [26] в [14]. Това е един пропуск, който сега коригираме.

Въпреки сериозността и задълбочеността на [26], там са допуснати някои неточности, които не можем да подминем. Това, че обръщаме внимание на някои пропуски и неточности на [26] по никакъв начин не означава, че подценяваме значимостта на тази статия. Няма съвършени статии и във всяка статия могат да се намерят неточности. Обикновено първата статия, която се появява в някоя нова област, е леко объркана и неясна. Обикновено в по-късните статии нещата се избистрят и изясняват.

Най-сериозната неточност на [26] е това, че там не се дефинира AI, а нещо друго. Това друго нещо ще го наречем „наблюдател“. Бихме могли да кажем, че наблюдател е програма, която има само вход и няма изход, но това би било твърде рестриктивно. Затова

ще кажем, че наблюдател е програма, чийто изход не влияе на състоянието на света, но може да повлияе на оценката, която програмата ще получи.

Пример за наблюдател е, когато играем на борсата с виртуален портфейл. Ние не влияем на борсовите цени, защото не играем реално, а само виртуално. (Дори и да играехме реално, пак може да се предположи, че нашите действия не влияят на борсата, защото когато играем с малки суми нашето влияние е пренебрежимо малко.) Когато играем с виртуален портфейл, ние на всяка стъпка променяме портфейла си и след всяка стъпка стойността на портфейла се променя в зависимост от промяната на цените на акциите. Оценката, която ще получим, ще бъде нашата виртуална печалба или загуба.

Това, че програмата дефинирана в [26] не може да влияе на света е съществен проблем, защото както казват хората: „За да се научиш, трябва да пипнеш“. Има и други поговорки, които казват, че само с гледане не може да се научим. Лишавайки AI от възможността да експериментира, ние сериозно го ограничаваме.

В [16] се отбелязва, че има два проблема, които AI трябва да реши. Първия е да разбере света (да построи модел на света), а втория проблем е да планира действията си на базата на намерения модел. Тоест, „наблюдателят“ решава само първия проблем, без да реши втория.

В [26] дефиницията се ограничава само до „наблюдатели“, но да се даде дефиниция на наблюдател, е достатъчно значима задача, защото това е половината от това, което AI трябва да свърши. За съжаление тази задача не е решена напълно, защото наблюдателя, който се дефинира в [26] е малко по-специален. Той или разбира света изцяло или въобще не го разбира. Тоест, наблюдател, който работи на принципа „всичко или нищо“.

В [26] са дадени случайни стрингове, които могат да се продължат по един единствен начин. Тоест, предполага се, че има една единствена най-проста зависимост и тази зависимост или ще бъде открита или няма да бъде. Този подход е твърде рестриктивен, защото предполага само наблюдатели, които разбират света напълно. Това е възможно само при много прости светове. Всеки по-сложен свят не може да бъде разбран напълно и се налага той да бъде разбран частично.

Авторите на [26] полагат сериозни усилия да направят зависимостите exception-free. Много интересно е определението за exception-free зависимости, което се дава в [26]. Въпреки всичко ние не бихме тръгнали по този път, защото ако зависимостите включваха изключения, това би бил един начин да се допусне частично разбиране на света. Търсенето на една единствена exception-free зависимост, която да опише всичко е причината, поради която дефинирания в [26] наблюдател залага на принципа „всичко или нищо“.

Имаме още няколко дребни препоръки към [26].

Даваме някакво време на AI, за да намери зависимостта. Въпросът е дали ще го дадем това време на веднъж или ще го разпределим на много стъпки. По принцип AI търси зависимостите през целия си живот. Тоест, за много стъпки. В [26] зависимостта се търси само за една стъпка. В нашата статия предполагаме, че в живота стъпките са около един милион (хиляда игри по хиляда стъпки). Това означава, че трябва да дадем на програмата,

която се дефинира в [26], милион пъти повече време. Нашата препоръка е зависимостите да се търсят след всяка стъпка, а не само след последната.

В [26] не ни харесва още това, че програмата, която генерира теста не работи поради комбинаторна експлозия. Това е програмата наречена „The Generator“. Тоест, в [26] не се дава тест, а само се показва, че такъв тест теоретично съществува.

Друг проблем е това, че са наложени две ограничения. Когато „наблюдателят“ прави предсказание той трябва да заложи всичко на един резултат и винаги трябва да залага една и съща сума. По-добре би било да има свободата да залага на повече от един резултат, както и да може да реши каква сума да заложи. Когато сме по-уверени залагаме повече, а когато се колебаем залагаме по-малко или пасуваме. Тези две ограничения не променят съществено нещата, защото умния ще докаже, че е умен дори и с тези ограничения, но така се замъглява картината. Ако ги нямаше тези ограничения разликата между IQ-то на умните и глупавите би била по-голяма.

Не ни харесва и това, че по-сложните задачи в [26] имат по-голяма тежест от по-простите, а би трябвало да е обратното. (Има един коефициент *e*, който се предполага че е неотрицателен, а би било по-добре да е отрицателен.) Вярно е, че когато правим контролно даваме повече точки на по-трудните задачи, но това е защото предполагаме, че студентите ще загубят повече време с по-трудната задача. Тук не е така. Тук на всяка задача даваме еднакво време. Затова, ако някоя проста задача не може да бъде решена, това е сериозен проблем и това трябва да се отрази в оценката. Освен това, понякога ще уцелваме решенията на трудните задачи случайно и затова те трябва да са по-малко и с по-малка тежест, в противен случай ще дадат незаслужено покачване на IQ-то.

Имаме понятията глобално и локално IQ (тези две понятия са дефинирани в нашата статия). Глобалното IQ е нещо точно, което не може да се сметне точно (заради комбинаторна експлозия). Локално IQ не е нещо точно, защото зависи от избора на конкретните въпроси в теста, но при конкретни въпроси, то може да се сметне лесно и точно. Това, което се дефинира в [26] е локалното IQ, а не глобалното. Все пак, да не забравяме, че локалното IQ клони към глобалното, но в [26] нищо не се казва за програмата „зубрач“, която е основния проблем на локалното IQ.

Как да коригираме дефиницията от [26], така че да получим дефиниция на наблюдател, която да не е на принципа „всичко или нищо“?

Вместо да взимаме специална k-разбираема редица, ние ще вземем съвсем произволна програма и редицата, която тази произволна програма генерира. Вместо да предсказваме продължението еднократно, ние ще го предсказваме на всяка стъпка. Няма да залагаме всичко на едно предсказание, а ще разрешим залога да бъде разпределен между няколко предсказания. Ще разрешим и залога да бъде различен. (Например, ще предположим, че сумата от залозите за последните пет стъпки е ограничен от някаква константа, но че имаме свободата да изберем кога да залагаме.) Успеха за една редица ще бъде сумата от успехите на различните стъпки. Локалното IQ ще бъде средното аритметично на успехите на редиците, които сме включили в теста.

По този начин няма да искаме от „наблюдателя“ да разбере света напълно, защото той може да хване някакви зависимости и по този начин, чрез частично разбиране на света, да получи високо IQ.

Проблем на [26] е, че програмата, която дефинира е всъщност програмата [10]. Това е една проста програмка, която предсказва продължението на редицата на базата на най-простата зависимост, която може да генерира началото й.

Аз дори твърдя нещо повече: Никоя програма удовлетворяваща дефиницията в [26] не е по-добра от [10]. Това, че никоя не е по-добра като резултат, е ясно, но аз твърдя че дори и като бързодействие никоя не е по-добра, защото дефиницията в [26] не ни дава никакви възможности за хитруване и за поетапно откриване на зависимости и единствената възможност остава глупавото изброяване на всички възможни зависимости.

Интересното е, че в по-новите статии на Orallo (например в [27]) той явно е разбрал основния пропуск на [26] и това, което дефинира вече не е „наблюдател“, а е програма, която може да влияе на света. За съжаление тази програма отново не е AI. В [27] се определя програма, която играе някаква игра, която се състои в обикаляне на един лабиринт (граф) като се гони нещо добро и се бяга от нещо лошо. За да се играе тази игра, определено има нужда от интелигентност, но програмата играеща тази игра не е AI, както и програмата играеща шах не е AI.

Пак в [27] авторите говорят за reinforcement learning. Тоест, ясно е, че те много добре знаят какъв е общия вид на AI. Защо тогава те не използват този общ вид, а се ограничават със световете на някаква определена игра? Според мен причината е, че те се опитват да избегнат световете, в които са възможни фатални грешки. За проблема с фаталните грешки се споменава още в [2], но фаталните грешки всъщност не са проблем. Ние хората също живеем в свят, в който има фатални грешки, но това не ни пречи да се изявим и да покажем кой е по-добър и кой е по-лош. Е да, при хората е проблем, защото ние живеем само един живот, но теста за IQ се състои от много задачи, всяка от които е един отделен живот. Дори и да се получат фатални грешки в няколко живота, това няма да повлияе съществено на средната оценка. Дори и при хората живота не е един. От гледна точка на отделния човек, живота действително е един, но от гледна точка на еволюцията, животите са много. Някои от нашите наследници ще загинат поради допускане на фатални грешки, но други ще оцелеят. По този начин средния успех на нашите наследници няма съществено да се повлияе от това, че някои от тях са допуснали фатални грешки.

Вярно е, че в тази наша статия, както и в [14], ние също не използваме произволен свят, а взимаме само изчислимите светове, но това ограничение не е съществено, защото всеки ограничен във времето свят е изчислим, а ние можем спокойно да приемем, че всички светове са ограничени във времето. Тоест, ограничавайки се с изчислимите светове ние въобще не се ограничаваме, а само даваме по-голямо тежест на световете, които са по-прости (по Колмогоров).

В [27] има още едно нещо, с което ние не сме съгласни. Това нещо се нарича „коефициент на обезценка“. Разбира се, това не идва на авторите на [27], а е нещо широко разпространено сред хората работещи в областта на reinforcement learning. Идеята на „коефициента на обезценка“ е че миналото е по-важно от бъдещето. Живота е потенциално безкраен, а за да оценим един безкраен живот трябва да обезценим бъдещето. Все пак, не е

добра идея миналото да е по-важно от бъдещето. Би било по-добре да е обратното, защото в миналото ние още не сме се обучили, а в бъдещето вече сме обучени. При хората ние не броим колко пъти се е напикавал човек докато е бил бебе. Вместо това ние гледаме какви постижения е постигнал човека в зрялата си възраст. Затова подхода, който ние приехме в [14] и в тази статия е да няма „коефициент на обезценка“, но живота да е ограничен. Казано с други думи, подхода тук и в [14] е коефициента на обезценка да е едно до един момент (края на живота) и да бъде нула от този момент нататък.

1.2.2 Постановка на задачата

Имаме устройство, което живее в някакъв свят. На всяка стъпка устройството извежда *n* букви (това е действието), след което получава *m* букви от външния свят (първата от които ще наречем оценка (reward), а останалите *m-1* букви ще наречем наблюдение). Оценката ще има 5 възможни стойности: *{nothing, victory, loss, draw, incorrect_move}*.

Думите „ход“ и „действие“ ще ги използваме като синоними. Ако гледаме на живота като на игра е по-естествено да кажем ход вместо действие. Думите „история“ и „живот“ също ще ги използваме като синоними.

Стъпка на устройството ще наречем тройка от вида <действие, оценка, наблюдение>. Живот на устройството ще наречем последователността от стъпки, която се е получила когато устройството е взаимодействало с определен свят.

Истински живот ще наричаме живота без некоректните ходове. Тоест, всички тройки <действие, оценка, наблюдение>, в които оценката е „*incorrect_move*“ трябва да се премахнат от живота, за да остане истинския живот.

Момент ще наречем поредица от стъпки такава, че последната стъпка преди поредицата и последната стъпка от поредицата да са коректни и всички стъпки между тези двете да са некоректни. Тоест, в живота стъпките може да са повече от моментите, но в истинския живот броя на стъпките е равен на броя на моментите.

Ще предполагаме, че устройството и света се държат детерминистично. Тест, ще предполагаме, че ако знаем кое е устройството и кой е света, то знаем и коя е историята.

Поведението на устройството можем да го представим като стратегия, тоест като функция, която за всяко начало на живота дава следващия ход на устройството. Аналогично можем да представим поведението на света като стратегия, която на всяко начало на живота и всяко действие на устройството дава оценката и наблюдението, които устройството ще получи на следващата стъпка. Трябва да отбележим, че стратегията на света не зависи от некоректните ходове. Тоест, стратегията на света можем да си я мислим като функция на истинския живот. Обратно, стратегията на устройството ще зависи от некоректните ходове (тези ходове ще представляват допълнителна информация, която устройството ще използва).

Можем да си мислим че устройството и света са две стратегии играещи една срещу друга, но това не е точно, защото устройството има цел, докато света няма цел. Тоест, света не

играе срещу устройството. Предполагаме, че света просто съществува и че той не се интересува от това дали на устройството му е добре или зле.

Представянето на устройството и на света като стратегии не е много добра идея, защото стратегията помни всичко (т.е. тя зависи от живота до момента). Нормално е да предполагаме, че устройството може и да не помни всичко. Аналогично нещо може да се каже и за света. Може да имаме свят от чието вътрешно състояние ние да можем да възстановим цялото минало (тоест живота до момента). Възможно е обаче света да не помни всичко и да имаме две различни истории, които да водят до едно и също вътрешно състояние на света. Затова ние ще представим света и устройството като функции.

Нека имаме две множества Q и S. Това ще са множествата на вътрешните състояния на устройството и на света. Тези множества ще са крайни или най-много изброими. Нека q_0 и s_0 са началните състояния на устройството и на света. Ще предположим, че тези начални състояния са фиксирани, защото живота ще зависи от това от кои начални състояния сме тръгнали, а ние искаме живота да зависи само от устройството и от света.

Устройството и света ще бъдат функциите:

$$Device: Q \times Rewards \times Observations \times 2^{Actions} \to Actions \times Q$$
$$World: S \times Actions \to Rewards \times Observations \times S$$

Функцията *Device* за всяко вътрешно състояние на устройството, оценка, наблюдение и множество от доказано некоректни в този момент ходове ще върне действие и ново вътрешно състояние на устройството. Ще предполагаме, че *Device* никога не връща действие, което е доказано некоректен в този момент ход.

Вътрешното състояние на устройството ще отразява това, което то е запомнило. Какво може да е запомнило? То може да помни всичко, което се е случило до момента, плюс последното си действие. Затова написахме $Actions \times Q$, а не $Q \times Actions$. Искахме да подчертаем, че новото вътрешно състояние на устройството може да помни последното действие.

Функцията *World* за всяко вътрешно състояние на света и действие ще върне оценка, наблюдение и ново вътрешно състояние на света. Естествено е да предполагаме, че има моменти (вътрешни състояния на света), в които определено действие е невъзможно или некоректно. Тоест, естествено е да предполагаме, че функцията *World* е частична. Ние ще додефинираме функцията и за тези моменти като по този начин ще я продължим до тотална. В тези моменти *Reward* ще бъде равно на *incorrect_move*, стойността на наблюдението ще е без значение, а новото вътрешно състояние ще бъде същото като старото, макар че и то е без значение.

Вътрешното състояние на света ще отразява това, което света е запомнил. Какво може той да е запомнил? Той може да помни всичко, което се е случило до момента, плюс последните оценка и наблюдение. Затова написахме $Rewards \times Observations \times S$, а не $S \times Rewards \times Observations$. Искахме да подчертаем, че новото вътрешно състояние може да помни последните оценка и наблюдение.

В [2] и в [14] дефинирахме новите <Reward, Observation> като функция на новото вътрешно състояние. Тоест там предполагахме, че те задължително се помнят, докато сега това изискване отпада. Да вземем като пример свят, в които играем шах. Партията завършва и новото вътрешно състояние на света е шахматна дъска с началната позиция. В този случай не е нужно да помним кой е победил в последната партия. Подобно изискване само би ни затруднило.

Живота на устройството ще изглежда така:
$<a_1, r_1, o_1>, <a_2, r_2, o_2>, \ldots, <a_{t-1}, r_{t-1}, o_{t-1}>$

Да видим как функциите *Device* и *World* ни определят живота.

$<a_{i+1}, q_{i+1}> = Device(q_{i-j}, r_{i-j}, o_{i-j}, incorrect\_actions_i)$
$<r_{i+1}, o_{i+1}, s_{i+1}> = World(s_{i-j}, a_{i+1})$

Тук *i-j* е последната коректна стъпка преди *i+1*. Множеството $incorrect\_actions_i$ съдържа доказано некоректните действия в този момент. Това множество има *j* елемента. Тоест $incorrect\_actions_i = \{a_{i-j+1}, \ldots, a_i\}$. Множеството $incorrect\_actions_0$ ще бъде празното множество, защото в първия момент още преди първата стъпка няма да има доказано некоректни действия.

За да определим живота ще трябва да фиксираме първата оценка и първото наблюдение (r_0, o_0). Не бихме искали живота да зависи от това кои ще са първата оценка и първото наблюдение и затова решаваме всичките букви на тези два вектора да имат стойността *nothing*. Това е едно логично решение, защото естествено е в първия момент ние да не получавам никаква оценка и да не виждаме нищо съществено (т.е. в първия момент ние виждаме нулевата стъпка). Няма да определяме нулевото действие a_0, защото не го използваме.

След като сме фиксирали ($q_0, s_0, r_0, o_0, incorrect\_actions_0$) можем да построим живота до стъпка *t* и този живот ще зависи само от функциите *Device* и *World*.
$<a_1, r_1, o_1>, <a_2, r_2, o_2>, \ldots, <a_{t-1}, r_{t-1}, o_{t-1}>$

Заедно с живота, ние ще построим и редиците с вътрешните състояния на устройството и на света, както и редицата $incorrect\_actions_i$ от доказано некоректните ходове в съответния момент. Да отбележим, че некоректните действия в даден момент може да са много, но доказано некоректните са само тези които вече сме пробвали и вече сме видели, че действително са некоректни в този момент.

Ще предполагаме, че функцията *Device* връща винаги ход, който не е доказано некоректен. В противен случай ще се получи зацикляне. Какво ще правим, ако всички ходове са доказано некоректни? (Тоест, ако $incorrect\_actions_i$ съвпада с цялото множество *Actions*.) В този случай ще предполагаме, че функцията *Device* не е дефинирана. Това е случая когато влизаме в тупик и няма никакъв възможен следващ ход.

Забележка: Дефинициите на *Device* като функция и като стратегия са еквивалентни с тази разлика, че ако разглеждаме *Device* като стратегия, то може да има значение в даден момент в какъв ред сме пробвали некоректните ходове, а при дефиницията като функция това е без значение. Тази разлика може да се отстрани по два начина. Първият е при

дефиницията на функция вместо множеството на доказано некоректните ходове да вземем списъка на тези ходове. Вторият начин, е да се ограничим със стратегии, при които да няма значение реда, в който сме пробвали некоректните ходове. В тази статия няма да има значение какво би се случило, ако стратегията пробва некоректните ходове в друг ред, защото предполагаме че стратегията е детерминирана и реда, в който тя ще пробва некоректните ходове е фиксиран.

Забележка: Тук предполагаме, че когато се опитаме да играем некоректен ход, нищо не се случва, а само получаваме информация, че хода е бил некоректен. Можем да разрешим на устройството да пробва дали хода е коректен или некоректен без задължително да го играе (както сме направили в някои предишни наши статии). В този случай ще трябва да променим постановката на задачата и да добавим още една оценка *correct_move*. Функцията *Device* ще трябва да има още един аргумент съдържаш доказано коректните ходове. Когато пробваме ход, който е доказано коректен ще приемем, че играем този ход. Когато ходът не е в множеството на доказано коректните, тогава ще приемем че само го пробваме. Функцията *World* ще има още един булев параметър, който ще казва дали хода се играе действително или само се пробва. Когато хода само се пробва ще върне една от двете оценки *correct_move* или *incorrect_move*. Тогава хода няма да се играе, а само ще се добави към следващото множество на доказано коректните или на доказано некоректните ходове.

Игра (пария) ще наричаме част от живота, която се намира между две последователни заключителни оценки. Заключителни оценки ще наричаме стойностите *{victory, loss, draw}*. Ще предполагаме, че всяка игра е не по-дълга от хиляда хода (т.е. хиляда момента, броя на стъпките може да е и по-голям заради некоректните ходове). Ще предполагаме, че в живота имаме не повече от хиляда игри.

Ще дадем дефиниция на това коя стратегия е AI и нашата дефиниция ще зависи от редица параметри. Повечето параметри ще ги фиксираме да бъдат числото хиляда, защото това е едно хубаво кръгло число. Друго подобно кръгло число е милион. Ако заменим числото хиляда с милион, то ще получим друга дефиниция на AI, която няма да се отличава съществено от дадената.

1.2.3 Параметри

Брой на буквите на действието	n
Брой на буквите на наблюдението	m
Брой на възможните символи за всяка от буквите на действието	$k_1, \ldots, k_n$, $k_i \geq 2$.
Брой на възможните символи за всяка от буквите на оценката и наблюдението	$k_{n+1}, \ldots, k_{n+m}$, $k_i \geq 2$, $k_{n+1}=5$.
Брой символи на лентите	$MaxSymbols = 10 + \max_{i \in [1,\ n+m]} k_i$
Брой на глобалните ленти	7 (от 3 до 9)
Брой на вътрешните състояния	1000
Брой тестови светове	1000
Максимален брой игри в един живот	1000
Максимален брой ходове в една игра	1000

Максимален брой стъпки на машината на Тюринг за една стъпка от живота	1000
Вероятност, с помощта на която се генерира машината	$\frac{1}{10}$ = 10%
Минималното IQ, за да може да бъде призната стратегията за AI	0.7 = 70%

Първите четири реда на таблицата ни дават параметрите, които описват входа и изхода на AI. Тези параметри ни казват какъв е формата на търсения AI. Затова тези параметри не можем да променяме произволно. Следващите осем параметъра влияят на избора на световете избрани за изпита и затова влияят на IQ-то, което ще получим и от там влияят на дефиницията на AI. Последния параметър също влияе на дефиницията. Тоест, променяйки последните девет параметъра ние бихме променили дефиницията на AI.

Тук не се включват някои параметри, от които зависи дефиницията на AI. Например не се казва точно кой е генератора на псевдо-случайни числа, който използваме, за да изберем световете от изпита. Разбира се, зависимостта на дефиницията от този параметър е незначителна.

Възможните символи за *i*-тата буква на действието ще са от 0 до k_i*-1*. Аналогично възможните букви за *i*-тата буква на наблюдението ще са от 0 до k_{n+1+i}*-1*. Символът 0 ще го наречем nothing. Първата буква на наблюдението ще бъде оценката. Когато става дума за оценката, символите 1, 2, 3 ще ги наречем victory, loss, draw и те ще са заключителните оценки. Оценката 4 (incorrect move) няма да се извежда от машината на Тюринг в резултат на извикване на командата q_1. Тази оценка ще се извежда само когато машината на Тюринг зацикли (тоест направи повече от 1000 стъпки без да достигне до заключителното състояние) или когато изгърми (например да извика командата **return** при празен стек).

Символите на лентата ще бъдат колкото е нужно, за да се кодира с тях действието и наблюдението. Тоест, максималното k_i за i от 1 до $n+m$. Към това ще добавим още 10 служебни символа, първият от които ще бъде празния символ λ.

1.2.4 Какъв е изпита (теста)

За изпита ще изберем 1000 свята. Във всеки от тези 1000 свята кандидата ще изживее по един живот, състоящ се от не повече от 1000 игри. Накрая ще изчислим броя на победите, загубите и ремитата. Като резултат ще получим IQ, което е равно на средното аритметично, където победата е едно, загубата е нула, а ремито е 1/2.

Световете ще ги изберем случайно, но искаме избраните светове да са фиксирани и затова ще ги изберем псевдо-случайно като преди да започнем избора ще инициализираме генератора на псевдо-случайни числа с числото 1. По този начин ще провеждаме изпита винаги с едни и същи светове.

Много от случайно генерираните светове ще са такива, в които задължително се печели или задължително се губи. Включването на подобни светове в изпита е безсмислено и затова ние ще изхвърлим тези светове от изпита. Така ще останат 1000 смислени свята.

1.2.5 Какво е свят

Във вестника можете да намерите задачи от вида „Кое е следващото число в редицата?“. Ако всички възможни редици са равновероятни, то следващото число може да бъде което си поиска. Когато търсим следващото число в редицата ние предполагаме, че по-простите редици са по-вероятни от по-сложните. Тоест, ние използваме принципа наречен „Бръснача на Окам“.

Аналогично е и положението със световете. Ако разглеждаме света като стратегия и ако всички стратегии са равновероятни, то няма как да предскажем бъдещето и няма на базата на какво да изберем един ход пред друг ход. За световете ние също ще използваме „Бръснача на Окам“ и ще предположим, че по-простите светове са по-вероятни. Кога един свят е по-прост от друг? Ще използваме сложността по Колмогоров. Тоест, ако света е стратегия, то по-простата стратегия е тази, която се генерира от машина на Тюринг с по-малко състояния.

Ние сме се ограничили само до крайните стратегии. Следователно всички стратегии са изчислими. Затова ние ще приемем, че света е някаква машина на Тюринг, която изчислява някаква стратегия.

1.2.6 Кои ще са възможните светове

Ще се ограничим до машините на Тюринг с 1000 състояния. Това множество включва и машините с по-малко състояния, защото всяка машина може да бъде допълнена с недостижими състояния. Машините на Тюринг, които използват съществено повече от 1000 състояния няма да са част от това множество. Тези светове ще ги приемем за твърде сложни и затова те няма да участват в дефиницията.

Получаваме едно огромно множество от стратегии (светове). Тук по-простите стратегии ще имат по-голяма тежест (те ще са по-вероятни), защото ще се генерират от повече машини на Тюринг.

Допълнително, на всяка машина на Тюринг ще дадем някаква тежест. Тоест, ще предпочитаме някои машини на Тюринг пред други. Например, ако машината използва повече състоянията с по-малък номер, ще я предпочетем пред тази, която използва повече от тези, които са с по-голям номер.

Причините, поради които даваме различна тежест на различните машини на Тюринг, са две. Първата е, че по този начин даваме по-голяма тежест на по-простите машини (например, тези, при които достижимите състояния са по-малко, са по-прости). Втората причина е, че искаме по случаен начин да генерираме работеща машина, което е много трудно. Тези машини, които имат по-голям шанс да са работещи са с по-голяма тежест, следователно избираме ги с по-голяма вероятност и по този начин увеличаваме шанса си да уцелим работеща машина.

Тежестта на машината е равна на вероятността, с която бихме я избрали. Ние няма да изчисляваме тази вероятност. Това би било едно доста трудно изчисление. Просто избираме машината на Тюринг случайно и чрез този избор вкарваме вероятността като параметър във формулата. Тоест, при изчисляването на локалното IQ, тази вероятност няма да се изчислява. Ако изчисляваме глобалното IQ, то тогава би трябвало да вземем всички машини на Тюринг, за всяка да сметнем успеха на устройството при тази машина и вероятността тази машина да бъде избрана. Трябва да умножим тези две числа и да сумираме по всички машини. Подобно изчисление е невъзможно поради комбинаторната експлозия. Не е проблем това, че усложняваме това изчисление, като добавяме изчисляване на вероятности. По този начин ние нищо не променяме. Глобалното IQ остава на теория изчислимо, а на практика пак е не изчислимо.

1.2.7 Как да сметнем IQ-то на конкретна програма

Бихме могли да кажем, че IQ-то ще е равно на средното аритметично на успеха във всички светове. (Тук трябва да отчетем, че световете не са равновероятни и трябва да умножим успеха по вероятността (теглото) на съответния свят.)

Това IQ ще го наречем глобално. Дефиницията на глобалното IQ е много хубава. Единствената забележка е, че това IQ не може да бъде изчислено. По-точно, то може да бъде изчислено на теория, но на практика не може, поради огромния брой светове които допуснахме за възможни.

Все пак, глобалното IQ можем да го изчислим приблизително по методите на статистиката. Ще изберем случайно 1000 свята и ще сметнем средното аритметично за тези светове. Полученият резултат би бил близък до глобалното IQ.

Тук проблемът е, че при различен избор на тестовите светове ще получим различно приближение на глобалното IQ. Ние искаме да имаме програма, която за всеки кандидат да дава неговото IQ и това да е една определена стойност, а не някакво приближение на нещо друго. Затова ние ще фиксираме произволно избраните 1000 свята и ще кажем, че локалното IQ е средния успех върху тези 1000 свята. (Тук различните светове няма да имат различно тегло, защото това е отчетено при избора на тестовите светове. По-тежките се избират с по-голяма вероятност.)

Идеята да фиксираме произволно избраните светове е същата като да дадем на всички кандидат-студенти едни и същи задачи.

Локалното IQ е една лесно изчислима функция и то добре описва идеята ни за това какво е IQ. Има само един проблем. Има една програма, която ще наречем програмата зубрач. Тази програма е подготвена специално за тестовите 1000 свята и нейното локално IQ е много високо, но глобалното ѝ IQ е ниско. Как ще решим този проблем? Когато търсим AI ще използваме локалното IQ. Когато намерим програма, която има много високо локално IQ, за която подозираме, че е програмата зубрач, ще ѝ дадем допълнителни задачи. Тоест, ще изчислим второто локално IQ. Това означава, че ще вземем следващите 1000 произволни свята и върху тях ще сметнем друго средно аритметично. Може да продължим и с третото и с четвъртото локално IQ.

1.2.8 Как машина на Тюринг прави ход

Светът го представяме като машина на Тюринг. Тоест, тя трябва да приема действията като вход и да извежда наблюденията като изход.

Първите $m+1$ състояния на машината ще бъдат специални. Състоянието q_{m+1} ще бъде началното и заключителното състояние на машината. Състоянията от q_1 до q_m ще бъдат състоянията, при които се извеждат буквите на наблюдението.

При първия ход на машината всички ленти ще са празни (т.е. ще са покрити със символа λ). Първата текуща лента ще бъде с номер 3 (номера 0, 1 и 2 се използват служебно).

Ход машината ще прави като започне от началното състояние q_{m+1} и завърши в същото състояние (то е и заключително). В началото на всеки ход върху текущата лента под главата на машината ще се запише n буквена дума, която ще е действието. (Това, което е било на първите n букви на лентата преди да се запише тази дума ще се изтрие.)

При всяка стъпка ще се следи за извикването на състоянията от q_1 до q_m. При извикването на тези състояния ще се извеждат буквите на наблюдението. Ако състоянието q_i се извика в рамките на един ход няколко пъти, то ще се гледа само първото му извикване. Ако не се извика нито веднъж, то i-тата буква на наблюдението ще бъде символа nothing. Ако се извика поне веднъж, то i-тата буква на наблюдението ще бъде стойността на „паметта на главата“ в момента след първото извикване на q_i. Ако i-тата буква на наблюдението е по-голяма или равна от съответното k_{n+i}, то тогава машината ще изгърми и ще се случи същото, което се случва при зацикляне.

1.2.9 Некоректни ходове

Когато машината на Тюринг не успее да направи ход, защото зацикля или изгърмява по някаква причина, тогава ще считаме, че действието, което е въведено в началото на хода е некоректно и невъзможно. В този случай ще се върнем назад и ще пробваме да въведем друго действие. По-точно, няма да се връщаме ние, а ще върнем назад света (машината на Тюринг). Ще дадем на устройството оценка *incorrect _move* и ще вземем следващия ход, който то ще ни даде. Ще продължим с този нов ход, все едно че устройството е изиграло него вместо некоректния ход. (Ако устройството повтори същия некоректен ход, ще го дисквалифицираме, защото то няма право да пробва два пъти един и същи ход в един и същи момент.)

Връщането назад на машината на Тюринг е съвсем естествена операция. Трябва само да сме запомнили конфигурацията на машината преди началото на хода. Ако възстановим тази конфигурация , можем да въведем друго действие и да продължим все едно, че това е първото действие, което сме пробвали.

В [14] се прилага друг подход. Там се предполага, че всички ходове са коректни и проблемът със зацикляне се решава като се прекъсва изпълнението на програмата,

присъжда се служебно равенство и програмата се рестартира от началното състояние. При това рестартиране лентата е останала в състоянието, в което е била в момента на прекъсването. Това е една много лоша практика. Вие знаете, че когато изключвате компютъра си, трябва да дадете командата „Shut Down“. Другият вариант е да издърпате щепсела от контакта, но тогава хард диска ще остане в състоянието, в което е бил когато сте дръпнали щепсела. Подобно отношение би накарало вашия компютър да се държи странно и нелогично. Същото може да се каже за машина на Тюринг, която се прекъсва в произволен момент и след това се рестартира от началното състояние. Ние искаме света да е колкото се може по-логичен и затова ще се погрижим да няма подобни прекъсвания.

При положение, че допускаме някой от ходовете да са некоректни, трябва да кажем какво правим когато всички ходове са некоректни. Да предположим, че имаме стотина възможни действия. Пробваме ги всичките и те всичките се оказват некоректни. Тогава ще считаме че сме попаднали в тупик и че няма път наникъде.

Ако живота е последователността от 1000 игри, то живота свършва естествено след 1000 игри или с внезапна смърт (попадане в тупик). Как да оценим един живот, ако той е завършил преждевременно? Можем да сметнем само изиграните до момента игри. Тогава нашата AI стратегия ще предпочете да се самоубие (да влезе в тупик), когато разбере че в текущия живот нещата са се объркали и оттук нататък се очакват само загуби.

Ние не искаме нашата дефиниция да ни даде AI стратегия със суицидни наклонности и затова ще изберем друго решение. Когато стратегията попадне в тупик, ще смятаме че всички останали игри до 1000 са загуби. По този начин, ще сме сигурни, че стратегията няма да влезе доброволно в тупик, а ще се бори до последно.

Ще разбере ли стратегията, че влиза в тупик? Подобно нещо няма как да се научи по метода на проба и грешка, защото в тупик се влиза само веднъж. Въпреки това, ако стратегията е много умна, то тя може да предскаже някои от влизанията в тупик. Например, ако броя на възможните ходове намалява, то това не е добре, защото може накрая да не остане нито един възможен ход. Друг пример е човека. Да вземем един конкретен човек, който никога не е умирал. Той няма личен опит, но въпреки това той може да предвиди някои от ситуациите, които биха довели до смъртта му.

1.2.10 Наказваме мотаенето

Казахме, че една игра продължава не повече от 1000 хода. Какво ще направим, ако играта продължи повече от 1000 хода? Тогава ще отсъдим служебно реми. Забележете, че тук не се намесваме в работата на машината на Тюринг и тя продължава да играе същата партия. Намесата е само към стратегията, защото тя ще получи оценка реми, въпреки че света (машината) е дал оценка nothing. Това, че не прекъсваме работата на машината гарантира, че тя ще запази логичното си поведение.

Ако изминат още 1000 хода без заключителна оценка ще присъдим служебна загуба. Тоест, ще накажем стратегията за мотаенето. Искаме да имаме AI стратегия, която се стреми бързо да завърши партията и да започне нова.

Наказвайки преждевременната смърт и мотаенето ние променяме IQ-то на случайната стратегия. Ако играем случайно, то очакваното IQ е 1/2. За да бъде повече, трябва нарочно да се стремим да печелим. За да бъде по-малко, трябва нарочно да се стремим да губим. Обявявайки всички игри след внезапната смърт за загуби, ние намаляваме IQ-то на всички стратегии. Аналогично, добавянето на служебни загуби също ще даде такова намаление. С това решение IQ-то на случайната стратегия няма да е 1/2, а ще е по-малко.

Колко точно е IQ-то на случайната стратегия ще можем да изчислим когато напишем програмата изчисляваща локалното IQ. Разбира се, случайната стратегия не е детерминирана и затова ще трябва да я изпитаме няколко пъти и да вземем средното. Тоест, IQ-то на случайната стратегия ще е приблизително. Точно локално и глобално IQ ще имат само детерминираните стратегии.

1.2.11 По-логична и по-ефективна машина на Тюринг

Както казахме, ще променим дефиницията на машината на Тюринг, за да я направим по-логична и по-ефективна. За целта ще направим машината на Тюринг многолентова и ще ѝ дадем възможност да вика подпрограми.

Защо тази машина ще ни даде по-логичен свят?

Първото е това, че машината на Тюринг ще е многолентова. Състоянието на света е естественото да се представи като декартово произведение на много параметри, които слабо си взаимодействат. Затова многолентовата машина на Тюринг дава по-логичен модел на света от еднолентовата.

Второто е това, че в тази машина ще има подпрограми, които се извикват от много места. Това ще е по-логично отколкото всеки път да се вика различна подпрограма. Когато нямаме стек се налага да помним къде да се върнем след изпълнението на подпрограмата. За да стане това, трябва всяка подпрограма да се извиква само от едно място (иначе няма да знае къде да се върне).

Когато викаме подпрограма ще ѝ дадем една чиста лента, на която тя да записва междините си резултати. Ако не ѝ дадем такава чиста лента, то тя ще трябва да използва някоя от общите ленти и това ще направи работата ѝ доста по-нелогична, защото ще се получат странни взаимодействия между различните извиквания на една подпрограма.

Защо тази машина ще прави по-малко стъпки и ще е с по-малко вътрешни състояния?

По-добра ефективност, ще означава по-малко стъпки и най-вече по-малко вътрешни състояния. Важно е стъпките да не са прекалено много, защото ако машината направи повече от 1000 стъпки за един ход ние приемаме, че е зациклила и я спираме. Важно е и вътрешните състояния да не са прекалено много, защото ние се ограничихме до машините с не повече от 1000 състояния. Затова нашата машина е добре да използва по-малко състояния.

Това, че машината е многолентова ще намали броя на стъпките, защото когато лентата е една ще се наложи на главата да се движи много, за да записва междините резултати. По-лесно ще бъде тези резултати да бъдат записани на друга лента.

По-съществено ще е това, че ще намалим броя на вътрешните състояния на машината. Класическата машина на Тюринг използва огромен брой вътрешни състояния (тя помни всичко, което трябва да се помни, във вътрешното си състояние). Например, когато се вика подпрограма трябва да се запомни от къде тази подпрограма е извикана и къде трябва да се върнем. Когато искаме да преместим символ от едно място на друго трябва да помним кои символ сме взели.

По тази причина усложняваме машината на Тюринг и тя освен вътрешното си състояние ще помни още коя е текущата лента, показалеца на стека (за подпрограми) и един символ „паметта на главата“.

1.2.12 Машина на Тюринг със стек

Как ще изглежда програмата на тази машина? Тя ще бъде една таблица с размери 1000 на MaxSymbols. Тук 1000 са възможните команди, а MaxSymbols са възможните символи. Във всяко от полетата на тази таблица ще има пет команди.

Първата команда е „Пишем върху лентата“
MaxSymbols+2 възможни стойности:
(без промяна, старата стойност на паметта на главата, конкретен символ)

Втората команда е „Променяме паметта на главата“
MaxSymbols+2 възможни стойности:
(без промяна, старата стойност на символа от лентата, конкретен символ)

Третата команда е „Движение на главата“
Три възможни стойности:
(ляво, дясно, на място)

Четвъртата команда е „Подпрограма“
Има две полета:
„Състояние“ в интервала [0, 1000] (тук 0 означава командата NULL).
„Нова текуща лента“ в интервала [0, 9] (тук 0 означава текущата лента, 1 текущата лента на бащината подпрограма, 2 временната лента, която е създадена специално за това извикване на тази подпрограма, от 3 до 9 са глобалните ленти).

Петата команда е „Следващо състояние“
Стойността е в интервала [0, 1000] (тук 0 означава командата return).

Когато се извиква една подпрограма, какво записваме в стека? Записваме три неща: Къде трябва да се върнем след return (това е петата команда), коя е старата текуща лента (за да я възстановим при return) и коя е временната лента създадена специално за това извикване на тази подпрограма. Новата лента също трябва някъде да се запише. Нека това да не е в

стека, а някъде другаде. При изпълнение на return съответната временна лента се унищожава.

1.2.13 Как попълваме таблицата

За да създадем произволна машина на Тюринг, ще трябва да запълним таблицата с размери 1000 на MaxSymbols със случайни команди. За целта първо ще кажем как избираме една случайна команда. Трябва да генерираме 6 числа (четвъртата команда има две полета). Биха могли тези числа да са равно вероятни, но ние предпочитаме, по-малките да са по-вероятни от по големите. Защо е това наше предпочитание? Защото, ако състоянията са равновероятни, то програмата ще се пръсне по много различни състояния, а ние искаме някои състояния да се използват по-често от други. Аналогично с лентите, искаме някои ленти да се използват по-често от други. Това важи и за служебните символи (за неслужебните не важи).

Как да изберем число от 0 до k с намаляваща вероятност. Например нека хвърлим монета и ако се паднеези избираме с вероятност 1/2 числото 0, в противен случай отново хвърляме монета и ако се паднеези избираме с вероятност 1/2 числото 1 и т.н. Когато стигнем до k, ако не се е падналоези започваме отново от 0.

Вероятността от 1/2 ни дава прекалено стръмно намаляване на вероятността на следващото число. Затова ние ще използваме вероятността 1/10. Така с тази вероятност от 1/10 ще генерираме всичките тези 6 числа. Всъщност, това което използваме е геометричното разпределение.

Забележка: Само за номера на подпрограмата ще подходим различно. Там 0 ще го вземем с вероятност 9/10 вместо с вероятност 1/10. (Нататък ще продължим пак с 1/10.) Това е така защото не искаме да се викат прекалено често подпрограми и да се пълни излишно стека. Така вероятността на командата return ще е равна на вероятността да се извика подпрограма.

Казахме как се генерира една команда (едно квадратче в таблицата). Да кажем как ще се генерира един стълб състоящ се от MaxSymbols квадратчета. Ще разглеждаме командата на тази машина като switch, който има MaxSymbols случая (case:). Когато програмираме и използваме switch ние не описваме всичките случай, а само няколко. Останалите случай ги описваме с default (тоест, останалите случай са еднакви). По-логична би била една програма, ако при голяма част от случаите командата е една и съща. Затова ние първо ще изберем случайно колко ще са различните команди в тази колона. Ще изберем по описания по-горе начин с намаляваща вероятност. След като сме избрали колко от позициите ще са различни ще изберем случайно кои ще са различните позиции (пак по-малките номера ще са по-вероятни). Накрая ще запълним позициите, които трябва да са различни различно, а останалите позиции ще запълним с една и съща команда.

Вече имаме алгоритъм за попълването на една колона и можем да попълним 1000 колони. Така ще получим първата случайна машина на Тюринг. Тази процедура е твърде тежка и затова втората машина ще я получим от първата като променим първите $m+1$ състояния (които са специални) и още 10 случайни състояния. Тази промяна е достатъчна, защото

огромната част от състоянията не се използват и променяйки специалните състояния ние ще започнем да използваме други състояния. Оттам новата машина на Тюринг ще е много различна в състоянията, които използва, макар че в недостижимите състояния двете машини да са почти еднакви.

1.2.14 Изхвърляне на шлаката

Вече имаме процедура, с която бързо и лесно можем да генерираме 1000 тестови свята. Проблемът е, че повечето от тези светове не са интересни. От 1000 свята интересните може да се окажат само два-три. Затова ние ще искаме да изхвърлим тези светове, които не са интересни и да проведем тест с 1000 интересни свята.

Пример за свят, който не е интересен е, ако още на първия ход се влиза в тупик.

Затова, за да докажем, че един свят е интересен ще пуснем случайната стратегия да изживее един живот в него. Ще искаме през този живот стратегията да не влиза в тупик. Ще искаме да имаме поне една победа и поне една загуба. Ще искаме служебните ремита и служебните загуби да не са повече от 10. Ако това е изпълнено при този случаен живот ще приемем, че света е интересен.

Как ще направим теста от 1000 интересни свята. Първо ще инициализираме генератора на псевдо-случайни числа с числото 0. После ще генерираме случайно нулевият свят. После ще инициализираме генератора с 1 и ще получим първия свят от нулевия чрез малка промяна. Ако първия свят е интересен ще инициализираме генератора с 2 и ще създадем втория свят. Ако първия свят не е интересен ще инициализираме с 2, 3, 4, 5 и т.н. докато не получим от нулевия свят интересен първи. По този начин ще създадем 1000 интересни свята. Няма да ги помним всичките, а ще помним само масив от 1000 числа. Това ще са стойностите, с които трябва да инициализираме генератора, за да получим от предишния интересен свят нов интересен свят.

Забележка: Трябва да отбележим, че различните машини на Тюринг ние ги избираме с различна вероятност. Тази различна вероятност е различната тежест на различните машини. Това уточнение ни трябва, ако искаме да дадем точна дефиниция на глобалното IQ. Дефиницията е:

$$\text{Global IQ}(Strategy) = \sum_{TM}^{\text{TM} \in Interesting} \text{P}(\text{TM} \mid Interesting).\text{Success}(Strategy, TM)$$

Тук P(TM | Interesting) е условната вероятност машината TM да бъде избрана при условие, че света на TM е интересен. Success(Strategy, TM) е средното аритметично получено след като стратегията Strategy е изживяла един живот в света определен от машината TM. Сумата е по всички машини с 1000 състояния, чиито светове са интересни.

Това глобално IQ не може да бъде изчислено, но това е теоретичната стойност, която се опитваме да доближим с локалното IQ.

Съответно локалното IQ ще бъде равно на:

$$\text{Local IQ}(Strategy) = \sum_{i=1}^{1000} \text{Success}(Strategy, TM_i)$$

Тук TM_i е i-тата от предварително избраните тестови светове (машини на Тюринг).

1.2.15 Окончателна дефиниция

Дефиниция: Ще кажем, че една стратегия е AI, ако нейното локално IQ е повече от 0.7.

Тук избрахме същата стойност, която избрахме и в [14]. Тази стойност и тук и в [14] я избираме съвсем произволно. Това е все едно предварително да кажем, че ще назначим за директор на нашата фирма всеки, който реши 70% от задачите в теста. Тази летва може да се окаже твърде ниска или твърде висока и в последствие може да се наложи тази стойност да бъде променена.

Дефиниция: Ще кажем, че една програма е AI, ако стратегията, която тя играе в първите 1000 игри е AI стратегия.

1.2.16 Аналогия с дефиницията на гросмайстор

За да обясним още веднъж дефиницията на AI ще повторим същата конструкция, но този път ще дефинираме какво е програма играеща шах. Това вече го направихме в [15], но тъй като настоящата статия повтаря и подобрява конструкцията описана в [14], тук ще повторим и конструкцията от [15], като отразим съответните изменения.

Програма играеща шах ще наричаме такава програма, която играе като гросмайстор. За да бъде един човек гросмайстор трябва неговият ЕЛО коефициент (Elo rating system) да бъде най-малко 2500 точки. Лошото е, че ЕЛО коефициента се изчислява на базата на играта на човека с други хора. За да получим обективна оценка на това, колко добър е играча, ще заменим другите играчи с крайно множество от детерминирани компютърни програми. Ще изиграем по една партия с всеки от тези играчи и резултата ще бъде средното аритметично от резултатите на отделните партии. Ако някой играч увисне (зацикли) и не успее да довърши играта ще присъдим служебна победа в полза на неговия опонент. Аналогично, ако някой изиграе некоректен ход. Тоест програмата, която кандидатства за гросмайстор, трябва да играе само коректни ходове и да не зацикля, защото, ако го направи, ще я накажем със служебна загуба. Аналогично, това важи и за компютърните програми от крайното множеството, които сме избрали, за да оценим чрез тях нашата програма.

Кое ще е крайното множество от детерминирани компютърни програми, които ще използваме за да проведем изпита за гросмайстор? Бихме могли да вземем всички програми не по-дълги от определена дължина. Повечето от тези програми ще играят случайно, често ще зациклят и ще играят много некоректни ходове. Разумно би било да ги отсеем и да оставим само тези програми, които са интересни (програмите, които играят

твърде безумно са баласт, който само утежнява теста.) Интересни ще са тези програми, които не зациклят, не играят некоректни ходове и който, освен това, играят сравнително добре.

Когато търсихме интересни светове, за да направим теста за AI, тогава използвахме метода на пълното изброяване (Brute-force search). Тук няма как да използваме този метод, защото много малко вероятно е случайно да попаднем на програма, която не зацикля и играе само коректни ходове. Затова вместо множеството на всички програми не по-дълги от определена дължина, ние ще вземем едно определено множество от програми такива, че всичките да са интересни (да не зациклят, да играят само коректни ходове и да играят сравнително добре)

Ще вземем една конкретна програма, която смята пет хода напред и разделя позициите на три вида: печеливши (такива, при които се печели до пет хода), губещи (такива, при които се губи до пет хода) и неопределени (останалите позиции). Кой ход ще изиграе тази програма? Тя ще избере случайно една от печелившите позиции. Ако няма такава ще избере случайно една от неопределените позиции. Ако и такава няма, то тогава ще избере случайно една от губещите позиции.

Това, което описахме е една недетерминирана програма. Ако вземем детерминираните стратегии, които тази програма може да реализира, те са огромен брой (все пак крайно много, защото дължината на играта е ограничена). Всяка от тези стратегии се изчислява от безбройно много програми, но ние ще приемем, че за всяка стратегия сме избрали по една програма, която я изчислява.

Като резултат получихме едно огромно множество от програми и можем да кажем, че ЕЛО коефициента ще го сметнем на базата на играта с всичките тези програми. За съжаление тези програми са твърде много и ние не можем да изчислим този коефициент поради комбинаторната експлозия. Вместо това, ние ще изберем 1000 от тези програми и ще сметнем коефициента след изиграването на 1000 игри (по една игра с всяка от тях). Ще изберем тези програми случайно, но ще ги изберем еднократно и всеки път ще определяме ЕЛО коефициента на базата на едни и същи тестови програми.

Как от недетерминираната програма ще направим детерминирана? Много просто, вместо да избираме случаен ход, ние ще избираме псевдо-случаен. Преди да започнем играта ще инициализираме генератора на случайни числа с числото едно. Така получихме една детерминирана програма. Трябват ни 1000 такива програми. Ще инициализираме с числата от 1 до 1000 и така ще получим 1000 различни детерминирани програми (между тях може да има и еднакви, особено ако генератора на псевдо-случайни числа не е много добър). Това ще са програмите, с които ще смятаме ЕЛО коефициента.

Ще наречем една програма гросмайстор, ако получения по този начин коефициент е повече от 90%. Тази стойност я избрахме произволно. Може да се окаже, че стойността трябва да е по-голяма. Може да се наложи дори да променим тестовите светове и вместо 5 хода напред да ги накараме да изчисляват например 10 хода.

Отново ще имаме проблема със зубрачите. Може да се назубри как да се победят някой от тези 1000 програми. Става дума за детерминирани програми, а една детерминирана

програма, ако я победим веднъж, можем да я побеждаваме колкото пъти си искаме като повтаряме същата партия.

Получаваме доста добра аналогия между дефиницията на програмата гросмайстор и дефиницията на AI. В единия случай говорим за ЕЛО коефициент, а в другия случай говорим за IQ. В първия случай ще играем шах срещу хиляда опонента, а във втория случай ще живеем хиляда живота в хиляда свята. Разликата е, че в първия случай срещу всеки опонент ние ще изиграем по една партия, а във втория във всеки живот ще направим по хиляда партии. Това е така, защото в първия случай правилата на играта са определени и се предполага, че програмата гросмайстор знае правилата и знае да играе (т.е. не се учи докато играе). Във втория случай AI не знае правилата на живота и има нужда от хиляда партии, за да разбере тези правила и да се научи да живее успешно.

Заключение

Тук описахме един изпит (тест), с който можем да изчислим IQ-то на произволна програма. По-точно писахме програмата, която ще проведе този изпит и ще ни каже какво е IQ-то на кандидата. Тази програма, беше описана толкова детайлно, че спокойно можем да поверим написването ѝ на някой студент, като му я дадем като курсова работа.

С помощта на тази програма бихме могли да проведем изпита за AI в рамките само на няколко минути. Толкова време ще е нужно на изпитващата програма да провери резултата от теста. Към това време трябва да добавим времето, което ще дадем на кандидата за мислене. Ако разглеждаме AI като стратегия, то тогава не си задаваме въпросът за това колко време ще мисли кандидата. Ако разглеждаме AI като програма, която изчислява AI стратегия, тогава трябва да кажем колко време даваме на тази програма, за пресмятането на една стъпка.

Тоест, времето за теста ще е няколко минути за проверката му, още известно време за мислене на проверяваната програма, плюс още време за създаването на теста (генерирането на масива от 1000 числа). Последното време не го броим, защото теста ще се генерира еднократно.

Нека си зададем въпроса, каква би била ползата от подобен тест. Ако някой ни даде конкретна програма, ни бихме могли да я тестваме и да кажем колко е IQ-то на тази програма. Но ние нямаме програми, които да са кандидати за AI. Тоест, ние нямаме кого да тестваме.

Едно възможно приложение на описания в тази статия тест е да го използваме за намирането на AI. Бихме могли да търсим по метода на пълното изчерпване. Разбира се, по този начин бихме могли да търсим, но не и да намерим. Поради комбинаторната експлозия, с този метод няма да стигнем далеч. Има и по интелигентен начин за търсене. Можем да направим един генетичен алгоритъм. В някой голям компютър ще създадем популация от програми кандидати за AI. За всеки от тези кандидати ще изчислим неговото IQ. Ще съчетаваме кандидатите с високо IQ, за да получим потомство с още по-високо IQ. Тези кандидати, чието IQ е много ниско ще ги убиваме, за да освободим място за по-

перспективните. По този начин, по пътя на естествения подбор, ще получим програми с много високо IQ.

Генетичния алгоритъм е една възможност за намирането на AI, но по този начин ние ще получим програма, за която не знаем как работи. Ако искаме да контролираме една програма, е по-добре сами да сме си я написали вместо да сме я генерирали автоматично. Затова аз съм привърженик на директния подход при създаването на AI. Тоест, аз съм привърженик на това, че сами трябва да напишем тази програма.

2 Как да го направим?

2.1 Представяне на дефиницията на AI във вид удобен за инженера

Ако искате да отговорите на въпроса: "Какво е това AI?" то първо трябва да се съобразите, на кого отговаряте и каква е целта на вашия отговор. Дали искате да убедите читателя, че AI съществува и че притежава едни или други интересни от теоретична гледна точка качества или искате да напишете инструкция за създаването му, в която вашият читател ще открие редица полезни практически съвети и технически трикове, които биха му били полезни при реален опит за написване на програмата, която наричаме AI.

Нека вземем като пример едно друго устройство наречено "компютър". Ако искаме да кажем какво е "компютър", първо трябва да решим за кого е предназначен нашият отговор. Може нашите читатели да са математици и да се интересуват от въпросите: "Съществува ли такова устройство?", "Какви свойства притежава то?", "Можем ли да го сведем до друго познато ни устройство? (т.е. да сведем нещата до предишния случай)". Математиците биха се зарадвали на дефиниция от вида "Машина на Тюринг" или "Машина с неограничени регистри", защото на тях им е нужно едно максимално просто описание, с което може лесно да се работи. Ако искате да проведете едно доказателство по индукция ще трябва да проверите за всяка от командите на Машината на Тюринг дали запазва индукционното предположение. Хубавото е, че всичките команди на Машината на Тюринг са от един и същи вид. Ако решите да проведете подобно доказателство използвайки модела на реален компютър, то ще се сблъсквате с проблема, че там възможните команди са стотици и е почти невъзможно да проверите за всяка една от тях дали запазва индукционното предположение.

Ако искате да отговорите на въпросът: "Какво е компютър?" и ако отговора ви е предназначен за инженери, то ще трябва да кажете нещо за процесор и памет, за шини по които се предават данните, за бройна система, по която тези данни се кодират и т.н.

Разликата между математиците и инженерите е първо в начина им на мислене и второ в целта, която те преследват. За математика понятието е интересно от теоретична гледна точка, а инженера иска да създаде реален продукт и затова него го вълнуват редица технически детайли, които за математика са несъществени.

Ето едно типично математическо разсъждение: "Искам да звънна на моя приятел Пешо, но съм забравил какъв беше телефонният му номер. Няма проблем, множеството на всички

телефони номера е крайно, ще звънна на всички и един от тях е номера на Пешо." Подобно разсъждение много често се среща в математическите доказателства. За математика въпросът е: "Мога ли да звънна на Пешо?" и отговорът е: "Да, може. Съществува алгоритъм, който след краен брой стъпки ще даде желания резултат". Разбира се, инженерите не използват подобни доказателства, защото тях не ги интересува въпросът: "Това теоретично възможно ли е?". Инженерите търсят реално работещо решение. Дори и да не знаят защо това работи, не се притесняват, защото важното за тях е решението да работи, а защо работи не е чак толкова важно.

В статиите [1, 2, 14] ние се опитахме да се харесаме на математиците и предложихме дефиниция на Изкуствен Интелект, която е удобна от теоретична гледна точка, но за целите на практиката е нужно да влезем в детайли, които са нужни за създаването на реална програма удовлетворяваща дефиницията.

Тоест, целта на тази статия е да се хареса на инженерите и да им каже какво е Изкуствен Интелект по начин, който би им бил полезен за създаването на реална работеща програма.

В [14] ние описахме конкретна програма, която удовлетворява дефиницията на AI, но тази програма беше толкова неефективна, че за да работи й трябва безкрайно бърз компютър. В [2] дадохме алгоритъм, който след краен брой стъпки, ще намери ефективна програма удовлетворяваща дефиницията, но този алгоритъм е толкова безполезен, колкото алгоритъмът позволяващ ви да звъннете на Пешо, защото крайният брой стъпки е толкова голям, че на практика е безкраен. В тази статия няма да се занимаваме с теоретичния въпрос – съществува ли AI, а ще се насочим към практически въпроси свързани с това как изглежда тази програма.

2.1.1 Формат на данните

Както казахме в статиите [1, 2, 12, 14], Изкуственият Интелект е стъпково устройство, което на всяка стъпка въвежда и извежда някаква информация. Естествено изниква въпросът какъв е формата на тази информация.

В [2] входните и изходните данни бяха булеви вектори. Там оценката беше изразена чрез два специални бита от входа, които нарекохме "победа" и "загуба". Тоест, в [2] данните имат някаква структура, докато в [1] входните данни са просто букви от някаква крайна азбука (аналогично и изходните). Победата и загубата са просто някакви подмножества на множеството на входните символи. Тоест, в [1] данните нямат никаква структура.

Статията [2] беше публикувана в научно-популярно списание и по тази причина тя е написана за по-широката публика, докато статията [1] е предназначена за математическо списание и там техническите детайли са изчистени. Сметнато е, че за математиците формата на данните е без значение.

В [12] се разглежда един конкретен свят. Това е играта Tic-tac-toe. Формата на данните е същия като в [2], но е добавен един бит, който нарекохме "некоректен ход". В [12] се вижда, че булевият вектор не е най-подходящият възможен формат, защото там входа показва какво има в текущото квадратче от таблото на играта. Възможностите са три:

празно, кръгче и кръстче. Тези три възможности се кодират в два бита, като една от четирите комбинации на двата бита просто не се използва, т.е. такъв вход никога не идва. Подобно е и положението с изхода. Там възможните действия са шест, което се кодира в три бита, като две от осемте комбинации на трите бита просто не се използват, т.е. когато устройството пробва да играе такава комбинация, винаги получава отговор "некоректен ход".

Това, че с помощта на кодиране можем да прехвърлим данните от един формат в друг означава, че от теоретична гледна точка формата на данните е без значение, но от гледна точка на практиката е важно да сме избрали правилния формат, за да се избегне нуждата от кодиране. Целта на AI е да разбере света и за да бъде тази цел лесно достижима е добре този свят да е максимално прост и лесен за разбиране. Ако сложим едно кодиране на входа и на изхода, по този начин ние може така да усложним света, че нашето устройство да не успее да го разбере.

Да вземем като пример цифрите от 0 до 9. Какво ще стане, ако вземем да ги разбъркаме? Ами, нищо особено. Щом сме успели да ги научим в този ред, ще успеем да ги научим и по новия начин. Да вземе сега числата от 0 до 99 и да ги разбъркаме. Нека започнем с едно леко разбъркване, като сменим само местата на двете цифри. Това разбъркване също няма да е проблем, защото щом сме успели да научим да пишем цифрите от ляво на дясно, можем да се научим да ги пишем и на обратно. Нека сега към числата от 0 до 99 да приложим една напълно произволна пермутация. Това вече ще е сериозен проблем, защото ще нарушим естествената логика, по която тези числа са подредени. Нека новият ред да изглежда приблизително така: 38, 12, 76 и т.н. В този нов ред няма никаква логика и ако искаме да се научим да броим от 0 до 99, то това би било едно много сериозно предизвикателство.

Аналогично е положението с булевите вектори. Предполагаме, че в тях света е представен по естествен начин и едно евентуално кодиране само ще усложни света и ще затрудни задачата този свят да бъде разбран. Ако разбъркаме нулата и единицата, то няма да е проблем, ако това са два символа, които произволно са кодирани с 0 и с 1, но ако предполагаме, че между тези символи има някаква наредба (т.е. че 0 е по-малко от 1), тогава разбъркването може да усложни света. Ако разменим местата на координатите във вектора, това също не би трябвало да е проблем, ако няма логика в подреждането на тези координати, но ако близките координати са свързани повече от далечните, то подобно разбъркване също може да е усложни света. Ако приложим произволна пермутация върху векторите, то това почти сигурно би било проблем, защото това ще скрие логиката, по която този формат е избран (стига да има такава логика разбира се).

Както казахме в [18], ще искаме светът да бъде сума от действието на различни фактори, които могат да бъдат независими, но могат и да си влияят един на друг. За подобен свят представянето на данните чрез вектори е особено подходящо. С вектори ще представяме и вътрешното състояние на света. (Светът е нещо външно за нашето устройство и за нас създателите на устройството няма значение какъв е формата на вътрешните състояния на света, но устройството ще търси можел на света и в този модел вътрешните състояния на света трябва да се опишат. За целите на това описание векторният формат е много подходящ, защото ако имаме два независими модела на света, много лесно можем да ги обединим в един, като новите състояния ще бъдат конкатенация от векторите, които отговарят на състоянията на двата модела.)

2.1.2 Сигнали

Дефиниция: Функция на един аргумент (времето), която връща скалар ще наричаме сигнал.

Тоест, координатите на векторите са сигнали. Ще разглеждаме четири вида сигнали: входящи, изходящи, оценъчни и вътрешни. Изходящите сигнали, това са координатите на изходния вектор. Входящите и оценъчните сигнали, това да координатите на входящия вектор.

Забележка: Тук изрично сме разделили входа на две: оценъчна част, която ни дава смисъла и чисто информационна част. По същия начин е направено в [2]. Там оценката беше изразена чрез два сигнала, които нарекохме "победа" и "загуба". Така е и в [12], но там има още един оценъчен сигнал наречен "некоректен ход". По-различно е в [1], където сме избрали да има само един входен сигнал, в който да са кодирани и информационната и оценъчната част на входа.

Освен сигналите участващи във входния и изходния вектор, имаме и много вътрешни сигнали, с които устройството ще работи и които ще използва, за да построи модел на света. Ще предполагаме, че устройството търси представяне на вътрешното състояние на света с вектор, чиито координати са вътрешни сигнали. Множеството на всевъзможните вътрешни сигнали е безкрайно, но във всеки конкретен момент, устройството е съсредоточило вниманието си върху краен брой вътрешни сигнали, които му се струват интересни и адекватни за света, в който е попаднало.

2.1.3 Небулеви вектори

Добре, след като решихме, че формата на данните и на вътрешните състояния на света ще е векторен, да си зададем въпросът дали можем да се ограничим в множеството на булевите вектори? Отговорът е, не. В [12] видяхме, че ако използваме само булеви вектори това налага допълнително кодиране.

Ако се ограничим само до булевите вектори, то тогава входните и изходните сигнали ще са булеви функции. Нека да допуснем и други по-сложни сигнали. Ще разрешим сигналът да връща k възможни стойности от множеството {0, 1, ... , k-1} вместо да има само две възможни стойности от множеството {0, 1}. Ще предполагаме още, че редът на тези стойности не е случаен (тоест, че наредбата "по-голямо" в множеството {0, 1, ... , k-1} кореспондира на някаква естествена наредба характерна за света, ако такава наредба има, разбира се). Предполагаме, че редът на възможните стойности на сигнала не е случаен, защото предполагаме, че света ни е предоставен по максимално естествения начин без да бъдем обременявани с никакво ненужно кодиране.

По този начин разширихме множеството на сигналите до множеството на крайните функции. Ще го разширим допълнително като разрешим сигналът да връща и безкрайни скалари като цяло, естествено или реално число. Отново ще предполагаме, че

представянето на данните като цяло или като реално число не е случайно, а е свързано с вътрешната структура на света. Следователно ще очакваме при реалните числа да имаме свойството непрекъснатост (т.е. малките промени да не са съществени). Ще очакваме още релацията "по-голямо" при естествените и реалните числа да не е случайна, а да е свързана с естествената структура на света.

2.1.4 Представяне на безкрайните обекти като крайни

Ще разглеждаме устройство, което на входа и изхода си въвежда (съответно извежда) вектори от скалари, където някои от скаларите са крайни, но може да има и безкрайни скалари. При това положение излизаме от практическото решение, при което имаме програма, която на всяка стъпка въвежда и извежда крайно количество информация. Допускането, че можем да имаме изброими скалари, води до резултата, че информацията въведена на входа и изведена на изхода няма да е крайна. Щом допускаме и неизброими скалари (като реалните числа) това води до резултата, че входните и изходните вектори въобще не могат да се кодират като крайна последователност от битове.

Тоест, допускането на безкрайни обекти като част от входа и изхода води до теоретичен модел, който не съответства напълно на практиката. Все пак, можем да предположим, че естествените и реалните числа участващи във входните и изходните вектори не са истински, а са компютърно представени (например, че са представени в 64 бита, кодирани със стандартното компютърно кодиране). Така получаваме два модела. Първият модел е теоретичен, където устройството работи с вход и изход съдържащи реални и естествени числа. При втория модел, тези числа не са реални и естествени, а са псевдо-реалните и псевдо-естествените числа, с които работи компютъра.

С кой от тези два модела ще работим? Ще работим с теоретичния, като резултатите ще ги приложим в практиката използвайки практическия модел. По същия начин постъпват хората, които се занимават с машини на Тюринг. Там те работят с теоретичния модел на компютър с безкрайна памет (защото машината на Тюринг има безкрайна лента). След това те прилагат получените резултати върху реалните компютри, чиято памет е крайна.

Единствения случай, когато вместо теоретичния модел ще използваме практическия, това е когато искаме да използваме това, че множествата на възможните входове и изходи са крайни. Това ще ни трябва, за да направим някое безсмислено доказателство за съществуване от типа "Телефонните номера са крайно много, следователно мога да звънна на Пешо."

2.1.5 Символите Undef и Nothing

В много езици за програмиране се въвежда един специален символ за случая, когато нямаме стойност или не знаем каква е стойността. Този символ ще е полезен и в нашия случай. Ще го наречем символа Undef.

Друг специален символ ще бъде Nothing. Той ще ни е нужен за случая, когато нищо не се случва. Няма да даваме точно определение на това, кога в света нищо не се случва, а ще

считаме, че един от входните символи е натоварен с допълнителен смисъл и че това е съвсем неформално и ориентировъчно без да означава нещо конкретно. Например, в реалния свят тишината може да бъде приета за символа Nothing. Интересно е, че когато човек спи на шум, той се събужда, ако шума рязко спре. Тоест, внезапно появилата се тишина се възприема като интересно събитие. Това означава, че символа Nothing е символ като всички останали и неговия смисъл като "нищо" е съвсем ориентировъчен.

Необходимо ли е, един определен символ да бъде натоварен с особен смисъл. От теоретична гледан точка не е нужно, но тук в тази статия се занимаваме с въпроса за практическото решение на задачата и искаме максимално да улесним устройството да разбере света и затова предполагаме, че сме му дали някаква предварителна информация. Такава информация е особения смисъл на символа Nothing.

Кога ще използваме символа Nothing? Например при входа, когато няма вход. Разбира се, нямането на вход, също е вход, но това е един по-специален вход.

Кога ще използваме символа Undef? Това ще бъде, когато входа е неизвестен (например някой датчик не е сработил или е закъснял с подаването на информацията си). Дори можем да предположим, че има възможност липсващата информация да се появи със закъснение (тоест, след още една стъпка да постъпи информация, че на предишната стъпка стойността на Undef е трябвало да бъде еди-коя-си.) Разбира се, нашето устройство трябва да може да приеме и обработи подобна информация, иначе тя не би имала никакъв смисъл.

От всеки сигнал може да получим нов като кажем "този сигнал преди k стъпки". Тоест, това е памет за сигнала. Естествено възниква въпроса, как да дефинираме тази памет за моментите когато t-k е отрицателно. Тоест, какво да си спомняме за момент преди раждането. Естествената стойност подходяща за този случай е символа Undef.

Нека вземем друг вътрешен сигнал. Нека имаме модел на света състоящ се от краен недетерминиран автомат. Нека имаме сигнал, който ни връща състоянието на този автомат в момента t. Ако автомата беше детерминиран, то и съответния му сигнал щеше да е детерминиран. Но при недетерминирания автомат може в даден момент да не знаем в кое състояние той се намира. Тогава естествено е сигнала да връща символа Undef. След още няколко стъпки, може да разберем в кое състояние сме били. Тогава може да постъпи допълнителна информация, която да ни каже, че стойността на сигнала на онази стъпка, е била еди-коя-си, а не Undef.

Както виждате, допускането на символите Undef и Nothing е много подходящо при входните сигнали и още по-подходящо при вътрешните сигнали.

При изходните сигнали също ще използваме символа Nothing. Представете си, че в даден момент, не знаете какво да правите. В този случай най-подходящо е нищо да не правите, но вие трябва нещо да направите, защото вашето устройство трябва да изведе някакъв изход, защото в противен случай то би увиснало безмълвно, а това предполагаме, че не трябва да се случва.

Кой да бъде изхода съответстващ на "не правя нищо". Нека това да бъде символа Nothing. Всеки друг символ може да играе тази роля, но ние предполагаме, че сме опростили и стандартизирали света в известна степен така, че да е по-лесно разбираем.

Естествено е символа Nothing да бъде котвата, за която устройството ще се захване, когато се появява (ражда) в един нов непознат свят. Първата му стъпка ще бъде да пробва, какво ще се случи когато играе Nothing по всичките координати на изходния вектор. После ще опита да даде стойност на една от координатите, а останалите да остави да бъдат Nothing.

Можем да допуснем, че при изходящите сигнали имаме право да използваме и символа Undef като предполагаме, че със закъснение от няколко стъпки устройството ще има право да уточни стойността на изхода и да каже какво е трябвало да стои на мястото на символа Undef. Това допускане много би усложнило нещата и затова няма да го правим. Тоест, символа Undef няма да участва в изходящите сигнали.

Нужен ли е символа Nothing при оценъчните сигнали? Отговорът е да. В статиите [2,12,14] предполагаме, че оценката се дава от два булеви сигнала наречени "победа" и "загуба". Когато тези два сигнала едновременно са единица, това го приемаме за "реми", а когато двата сигнала са едновременно нула, тогава приемаме, че нямаме никаква оценка. Както виждате, добавено е едно излишно кодиране. В [14] се изчислява успеха на устройството, като се смята средното аритметично от победите, загубите и ремитата, но в това средно аритметично не участват случаите когато не сме получили никаква оценка.

Не е логично да предполагаме, че на всяка стъпка нашето устройство ще получава някаква оценка. По-добре е да предположим, че на повечето стъпки никаква оценка не се получава и за тези случаи нека използваме символа Nothing.

За да бъдат нещата прости и логични нека предположим, че оценката се дава от един сигнал, който има стойност 1, 0 и 1/2 съответно при случаите на победа, загуба и реми, а когато няма оценка сигнала да има стойност Nothing. Тогава успеха на устройството ще бъде средното аритметично на тези стойности на този сигнал, които са различни от Nothing.

За да опростим нещата ще предположим, че за случаите на входящи, изходящи и вътрешни сигнали символа Nothing съвпада със символа нула. Решихме един от символите да бъде натоварен с особен смисъл и най-подходящия за целта е символа нула. Другото предимство на нулата е, че тя присъства във всички формати. Има я и в булевия формат и в крайния и в естествените числа и в реалните и т.н.

Единствени случай, когато ще предполагаме, че нулата и Nothing са различни символи, това ще е в оценъчните сигнали. Там правим средно аритметично между всички стойности различни от Nothing, а нулата е нормално число, което спокойно може да участва в средното аритметично. Ние бихме искали специално да подчертаем, че Nothing не участва в изчисляването на средното аритметично и затова ще приемен, че то не е нула, а е нещо различно.

2.1.6 Възможни оценки

В [2] казахме, че за Изкуствен Интелект признаваме такава програма, която в произволен свят би се справила не по-зле от човек. За да можем да сравняваме и да казваме кой се е

справил по-добре и кой по-зле, трябва да имаме някаква наредба на животите, която да ни казва кой живот е по-добър и кой е по-лош.

Тази наредба ще наричаме "смисъла на живота". За да дефинираме тази наредба, първо ще дефинираме смисъла на живота за случаите на краен живот, а след това ще разширим наредбата и за случаите на безкраен живот.

Нека имаме произволна линейна наредба на крайните животи. (Да припомним, че живот наричаме последователността от входните и изходните вектори от момента на раждането, до определен момент или до безкрайност)

За всяка линейна наредба между крайните животи има съответстваща на тази наредба, оценъчна функция (тази функция ще я наречем Succes). Тоест, ако един живот е по-добър от друг, то функцията Succes за по-добрия живот връща по-голяма стойност, отколкото за по-лошия.

Как по естествен начин да продължим дефиницията на смисъла на живота, така че тя да включи и безкрайните животи. Нека вземем редицата от началата на един безкраен живот. Нека за всяко едно начало да изчислим стойността на функцията Succes. Получаваме една редица от реални числа и ако тази редица е сходяща, то естествено е да дефинираме функцията Succes за този безкраен живот да бъде равна на тази граница (тук плюс и минус безкрайност също са възможни граници). Ако редицата не е сходяща, то ще считаме, че функцията Succes за този безкраен живот не е определена точно, но че нейната стойност е някъде между точната долна и точната горна граница на тази редица. По този начин получаваме една нова функция Succes, която за някои животи връща точна стойност, а за други връща интервал от възможно най-малката до възможно най-голямата стойност.

По този начин новата функция Succes определя една частична наредба в множеството на всички животи (тази наредба няма да е линейна). Един живот е по-добър от друг, ако стойността на функцията Succes за този живот е по-голяма от стойността на Succes за другия или ако интервала, в който се намира тази стойност е в дясно (т.е. е по-голям) от интервала, в който се намира другата стойност.

Изкуствения Интелект е устройство, което сме го оставили само да разбере света, но смисъла на живота трябва предварително да сме му го задали. Няма как да искаме от него да се справя добре, при положение, че той не знае кой резултат е добър и кой е лош. Устройството трябва да може във всеки момент само да изчисли функцията Succes и тя трябва да зависи само от оценъчните сигнали, защото входните и изходните сигнали са само информативни и не ни дават директен смисъл.

Най-простото решение е да предположим, че имаме един оценъчен сигнал, който ни връща стойността на функцията Succes. Това решение не е много добро, защото по този начин натоварваме света да помни какво се е случило с устройството до момента, за да може да даде цялостна оценка за целия му живот. По-добре е да имаме един оценъчен сигнал и функцията Succes да бъде средното аритметично от всичките стойности на този сигнал. По този начин света ще оценява устройството за последната му стъпка, а не за целия му живот до момента. Когато на поредната стъпка нищо интересно не се е случило и функцията Succes няма нужда да се променя, то можем да предположим, че сигнала връща предишната стойност на Succes и така средното аритметично ще се запази, но пак

обременяваме света да помни какво е станало до момента и затова в този случай ще върнем символа Nothing. Това е един по-прост начин, за да запазим средното аритметично.

Каквато и да е функцията Succes можем да намерим оценъчен сигнал, чието средно аритметично да е точно тази функция. Този оценъчен сигнал не е определен еднозначно, защото на моменти имаме избор дали сигнала да върне символа Nothing или да върне своето средното аритметично получено до момента. Трябва да отбележим още, че предполагаме, че функцията Succes връща нула за празния живот (за живота с дължина нула). Това последното не е проблем, защото ако добавим константа към функцията Succes или ако я умножим с положителна константа няма да променим наредбата, която тя определя.

Следователно, произволен смисъл на живота може да се представи като средното аритметично на оценъчен сигнал, който връща реално число. Въпреки всичко, ние не харесваме това решение, защото бихме искали светът да е устойчив и оценката (функцията Succes) да не може да скача неконтролируемо. Затова ще предполагаме, че оценъчния сигнал е крайна функция и връща стойност от множеството {Nothing, 0, 1, ... , k}. Тоест, ще предполагаме, че има k+2 възможни стойности. Следователно функцията Succes ще бъде в интервала [0, k].

Това решение също не е перфектно, защото може да искаме да имаме различни нива на приоритет. Например, важно е да не закъснееш за училище, но много по-важно е да не загинеш в автомобилна катастрофа. Тук под много по-важно разбираме безкрайно по-важно. Искаме нашата дефиниция да разрешава светът да има N нива на приоритет. За целта ще предполагаме, че имаме N оценъчни сигнала и че функцията Succes връща не число, а вектор. Този вектор се получава, като се сметне средното аритметично покоординатно. (Като за всяка координата сумата се дели на броя пъти когато тази координата е била различна от символа Nothing.) Сравнението на два такива вектора ще става покоординатно. Тоест, гледа се координатата на най-високото ниво на приоритет. Ако за тази координата стойностите на двата вектора са равни, се гледа следващата координата и така нататък.

Можем ли да емулираме свят с две нива на приоритет, когато оценъчният сигнал не е ограничен? Да, нека малкия приоритет да връща 0 или 1, а големия приоритет да връща 0 или 2 умножено по броя пъти до момента, когато стойността на сигнала е била различна от символа Nothing. Допълнително света трябва да запомни момента, когато е дал 2 умножено по нещо-си и от този момент нататък към всяка оценка да прибавя 2. По този начин, ако е получена оценка само по малкия приоритет, то функцията Succes ще бъде в интервала [0, 1], но ако имаме оценка от високия приоритет, то функцията Succes ще бъде по-голяма или равна на 2. Тук при тази емулация се отказахме от ограничението на оценъчния сигнал и обременихме света да помни какво е станало до момента.

Има ли смисъл на живота, който не може да бъде представен с N нива на приоритет? Отговорът е да. Нека вземем смисъл на живота, който има изброимо много нива на приоритет. Това може да се емулира с един неограничен оценъчен сигнал, но не може да се представи с N ограничени оценъчни сигнала за никое N. Тоест, избирайки тази дефиниция на оценката ние се ограничаваме и по този начин не всеки смисъл на живота ще е възможен, но считаме, че световете с N нива на приоритет са достатъчни за

практиката и че не е нужно да разглеждаме светове с изброимо много нива на приоритет. Дори, за практиката, в повечето случаи, достатъчно е нивото на приоритет да е едно.

Освен N-те оценъчни сигнала, от които се изчислява функцията Succes ще имаме още един булев оценъчен сигнал, който ще наречем "некоректен ход". Този сигнал ще разгледаме в следващата секция.

2.1.7 Некоректен ход

Какво е "некоректен ход"? Например в шаха, ако се опиташ да играеш с коня като с царица, то това е некоректен ход. Също така в шаха нямаш право да дадеш на противника да ти вземе царя, тоест некоректен е всеки твой ход, след който ти си шах. Има много игри в които взимането е задължително. В тези игри некоректен ход е когато можеш да вземеш, но не взимаш.

Ясно е, че в повечето светове има некоректни ходове. При положение, че сме фиксирали множеството на изходящите вектори, то естествено е да предполагаме, че не всички техни стойности са коректен ход. Нормално е в един конкретен момент един конкретен изходящ вектор да е коректен ход, а в друг момент същият вектор вече да е некоректен ход.

За да позволим в света да има некоректни ходове, трябва да си отговорим на два въпроса: "Какво става със света когато устройството играе некоректен ход?" и "Как света връща към устройството информация за това, че последния ход е бил некоректен?"

В статиите [1, 2] въобще не се говори за некоректни ходове. Там се предполага, че светът наказва устройството за всеки негов некоректен ход, като го пляска през ръцете (т.е. като му дава лоша оценка). Например в света, където играете шах, какво ще стане, когато се опитате да играете некоректен ход? Едно възможно решение е да дефинираме света така, че при некоректен ход да губите партията, която играете в момента и да автоматично да започвате нова партия.

В статията [12] е въведен отделен сигнал наречен "некоректен ход". Там се предполага, че опитите на устройството да играе некоректен ход не водят до промяна на света. Резултата е, че светът си остава в същото вътрешно състояние, но връща сигнала некоректен ход към устройството. Грешката в [12] е, че некоректния ход се приема за наказание и се предполага, че устройството ще се научи да играе само коректни ходове и ще избягва некоректните, за да не бъде наказано.

Информацията за това кой ход е коректен и кой не е, е много важна за разбирането на света. Да вземем като пример случая, когато в тъмното намираме пътя си като опипваме с ръце стените. Опипването на стените може да бъде прието за некоректен ход, защото ние се опитваме да прекараме ръката си през пространство през което тя не може да мине, защото там има стена. Въпреки това, ние съзнателно правим този некоректен ход, за да разберем къде е стената.

Може би е добре да променим дефиницията на свят дадена в статиите [1, 2] и към функциите World и View, които описват света да добавим още една функция наречена

Correct, която за всяко вътрешно състояние на света да връща множеството на възможните ходове.

Ние искаме да направим задачата на устройството максимално проста и така да дефинираме света, че той да е колкото се може по-лесен за разбиране. Затова, разумно е да предполагаме, че на всяка стъпка, устройството получава като вход не само стойността на функцията View, но също така, то да получава и стойността на функцията Correct.

По този начин изкачат два проблема:

Първият проблем е, че информацията, която ще ни върне функцията Correct може да е прекалено много. Ако възможните изходи са k на брой, то възможните стойности на функцията Correct са 2^k. Съответно, ако изходите са изброимо много, то стойности на функцията Correct са континиум много и т.н.

Вторият проблем е в това, че по този начин ние ще усложним света, като наложим изискването на всяка стъпка той да изчислява функцията Correct, да кодира резултата в подходящ формат и да го подава към устройството.

От тези проблеми ще се отървем, ако предположим, че света не казва на устройството експлицитно кои ходове са коректни, а при всеки некоректен ход той връща информация за това, че ходът е некоректен.

Тоест, предполагаме, че имаме един булев сигнал наречен "некоректен ход". Ще разрешим на устройството да прави некоректни ходове и когато се получи единица на този сигнал, това няма да го разглеждаме като наказание за устройството, а просто като полезна информация.

Все пак ще направим четири предположения:

1. Ще предполагаме, че некоректния ход не променя вътрешното състояние на света. Тоест, играейки некоректен ход, устройството не губи нищо. Може да се каже, че и не печели нищо. Единствената печалба е информацията, която получава. Тоест, играейки един ход, ако този ход се окаже некоректен, устройството получава информацията, че този ход е некоректен, което може да се окаже полезна информация.

2. Ще предполагаме, че ако сме опитали един ход и света ни е казал, че той е некоректен, то няма нужда да го пробваме повторно докато състоянието на света е същото. Разбира се, ако предполагаме, че функцията Correct е фиксирана и се знае кои са коректните ходове още преди да сме опитали, то горното е така, но ние може да искаме да допуснем и нещо по-слабо. Например, представете си свят с вграден в него часовник, който отчита колко време устройството е мислило. В този свят функцията Correct зависи не само от състоянието на света, но и от това колко сме се забавили преди да играем даден ход. Е, ще предполагаме, че дори и функцията Correct да се променя в зависимост от забавянето, то некоректните ходове само се увеличават. Т.е. ще предполагаме, че ако един ход е некоректен, то и да го повторим, пак ще е некоректен.

3. Ще предполагаме, че устройството няма право да играе едни и същи некоректни ходове до безкрайност. Тест, то ще е длъжно да помни какви некоректни ходове е играло и да не

ги повтаря поне до получаването на коректен ход. След получаването на коректен ход устройството може да изчисти паметта си и да пробва ходове, които са били некоректни, защото това че един ход е бил некоректен на предишна стъпка, не значи че ще бъде некоректен и на следващата.

Горното предположение го направихме, за да не разрешим на устройството да зацикля. Разбира се, възможните некоректни ходове може да са много и дори безкрайно много, което пак да доведе до забавяне или зацикляне, но припомняйки, че възможните изходи са крайно или псевдо безкрайно (което също е крайно), ще стигнем до извода, че, поне на теория, подобно зацикляне не може да се случи.

4. Ще предполагаме, че функцията Correct никога не връща празното множество. Тоест, ще предполагаме, че винаги има поне един коректен ход. Ако допуснем съществуването на тупици, в които няма изход (т.е. функцията Correct връща празното множество), то тези моменти може да ги асоцииираме със смъртта. Може да смятаме смъртта за грешка, която се опитваме да избегнем, но тази грешка винаги е фатална и затова не можем да се учим от нея. Затова по-удобно е да считаме, че в нашия свят няма такива моменти.

Все пак може, когато програмираме устройството наречено AI, да заложим в него принципа да се стреми към състояние, в което възможните ходове са повече. В играта шах ние се стремим да развием фигурите си така, че възможните ходове да са възможно най-много. Загубата на царицата силно намалява броя на възможните ходове, което прави тази загуба нежелана. В живота хората се стремят към свободата. Тоест, искат да имат колкото се може повече възможни ходове. Всеки човек затворен в прекалено малко и тясно пространство се чувства дискомфортно. Също така, човек се чувства дискомфортно когато е заключен или когато е с белезници. По този начин можем да обясним и стремежът на хората към власт и пари, защото това дава допълнителна свобода. Когато имаш пари можеш да си купиш яхта, а можеш и да не си купиш, а когато нямаш, нямаш този избор. Следователно, в човека инстинктивно е заложено да се стреми към състояние, в което възможностите са повече. Логично е този принцип да бъде заложен и в Изкуствения Интелект. Ако нашето устройство избягва случаите, когато възможните ходове са малко, то то би се опитало да избегне и смъртта, защото това е случая когато нямаме никакви възможни ходове.

2.1.8 Как ще използваме некоректните ходове

Добре, нашето устройство разбира света и има представа кой ход е коректен и кой не е. Как ще очакваме от него да постъпи? Когато един ход със сигурност е некоректен, то устройството няма да го пробва, за да не губи излишно процесорно време (тук със сигурност означава с много голяма вероятност, защото нищо не е абсолютно сигурно). Когато един ход почти сигурно е некоректен, то тогава устройството ще го пробва, защото нищо не губи от това, а само получава информация. Ако ходът действително е некоректен, то на следващи път това ще се знае с още по-голяма сигурност, а ако вземе случайно да се окаже коректен, то устройството може да загуби, но може и да спечели като открие нови неподозирани възможности. Когато не се знае дали хода е коректен или не е коректен устройството може да го пробва, а може и да не го пробва. От една страна ще иска да провери дали хода е коректен, но от друга ще се страхува от евентуални неприятни

последици. Например, вие не се опитвате да скочите през прозореца, за да проверите дали ще успеете, защото ако случайно успеете, последствията могат да се окажат много лоши.

Въпрос: Дали некоректните ходове са част от живота? Когато опитваме некоректен ход дали се увеличава брояча на стъпките (т.е. параметъра време)? Отговорът е не. Ако разгледаме булевият сигнал "некоректен ход" ще видим, че той е нула за всяко t. Тоест, всички ходове записани в историята са коректни. Ще предполагаме, че некоректните ходове просто не се записват в историята (в живота).

Живота е последователност от входни и изходни вектори. Ако сигнала "некоректен ход" е една от координатите на входящия вектор, то тази координата винаги е нула. Ще считаме, че сигнала "некоректен ход" не е част от историята, защото е безсмислено да включваме сигнал, който е константно нула.

Все пак казахме, че искаме информацията, която се получава от некоректните ходове да може да бъде използвана. Затова, ще променим дефиницията на живота като вмъкнем множеството от векторите на некоректните ходове, които сме опитали, между входящия вектор и коректния изходящ вектор. Тоест, живота ще стане последователност от входен вектор, множество некоректни изходни вектори, коректен изходен вектор и т.н.

В следващата статия ще разгледаме зависимостите без памет. Това са зависимости от типа: "Ако виждам това и правя онова, ще се получи еди-какво-си." Тези зависимости се представят като импликации от вида: **a(t-1)=1, b(t)=0, do(t)=1 => bad_move(t+1)=1**. Тази импликация трябва да бъде прочетена така: Ако сигнала **b** на тази стъпка е нула и ако сигнала **a** на предишната стъпка е бил едно и ако изберем сигнала **do** на тази стъпка да бъде едно, то това е некоректен ход. Сигнала **do** го избираме какъв да бъде, защото това е изходящ сигнал, а сигналите **a** и **b** са каквито са, защото това са входящи сигнали, които не ги избираме, а ни ги дава светът.

Импликацията води към **bad_move=1**, но това няма да се запише в историята, защото там се записват само коректните ходове.

Горната импликация ни казва, че при някакви обстоятелства определен ход ще е некоректен. Тоест, тук виждаме как на базата на информацията събрана от некоректните ходове можем да се научим да познаваме такива ходове. В следващата статия ще видим как от това, че даден ход е некоректен може да се извлече и повече информация за това какво е състоянието на света и какво ще се случи на следващата стъпка.

2.1.9 Добавяне на некоректните ходове към дефиницията на AI

В [14] ние дефинираме AI, като устройството, чието IQ е достатъчно високо. Изчисляването на IQ става на базата на множество от тестови светове, като се взима средния успех на устройството за световете от това множество. Световете, които се използват в [14] са генерираните от произволна машина на Тюринг (чиято сложност не надвишава стойността на един параметър, който наричаме ниво на интелигентност).

Множеството от тестовите светове, което сме използвали в [14] е така избрано, че световете да бъдат максимално естествени и разбираеми. Целта не е да затрудним устройството като го накараме да разбира неразбираеми светове, а напротив да го улесним като направим тестовите светове максимално естествени и разбираеми.

Един от проблемите в [14] е това, че там всички ходове са коректни. Тоест, устройството, което дефинираме в [14] не знае какво е това некоректен ход и няма да може да се справи в свят, в който част от ходовете са некоректни. Хубаво би било да променим дефиницията от [14], така че тя да разрешава светове, в които има некоректни ходове. От друга страна добавянето на некоректните ходове ще направи световете, които сме избрали за тестови, да бъдат по-естествени и по-разбираеми.

Проблем в [14] са световете, които в даден момент зациклят. Щом света се генерира от произволна машина на Тюринг, то тази машина може в определен момент да зацикли. Тук проблемите са два: как да разберем, че машината е зациклила и какво да направим след като разберем, че е зациклила. Първият проблем в [14] го решаваме простичко: Ако за 800 стъпки машината не върне резултат, ще считаме, че е зациклила. Решението на първия проблем няма да променяме, но ще променим решението на втория.

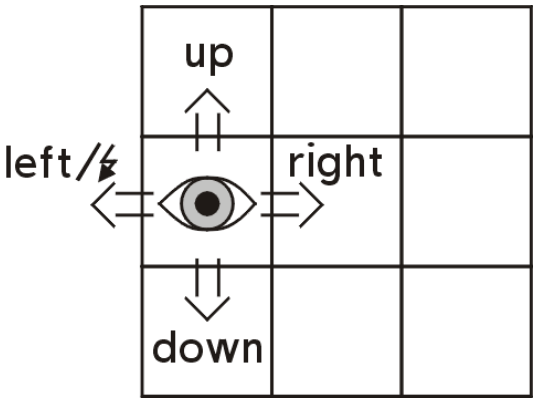
В [14], когато машината на Тюринг зацикля, ние я прекъсваме и рестартираме, тоест запазваме това което е записано върху лентата, но променяме вътрешното й състояние, така че следващото й състояние да бъде началното. Това решение не е добро. Това е все едно да изключваме компютъра си като дръпваме щепсела, без да се вълнуваме какво ще остане записано на харддиска. Подобно отношение към компютъра може да направи неговото поведение твърде сложно и непрогнозируемо. Прекъсвайки зациклящата програма ние излишно усложняваме света, който тя генерира. По-добре би било след като я спрем, да кажем, че този ход е некоректен и да върнем машината назад като възстановим информацията записана върху лентата й до тази, която е била когато машината е започнала работа върху този некоректен ход. По този начин това, че сме направили некоректен ход няма да има никакво отражение на бъдещото, което ще се случи в съответния свят. По този начин светът генериран от машината ще бъде много по-естествен и по-разбираем.

Остана още един проблем. Възможно е произволната машина на Тюринг да попадне в състояние (на лентата), при което всеки ход води до зацикляне. Тоест, ще трябва да се откажем от изискването винаги да съществува поне един коректен ход. Последното означава, светът да е такъв, че устройството да не може „да умре“. В [14] ние вече се оказахме от изискването в света да няма фатални грешки. По същата логика и по същите причини можем да се откажем и от изискването устройството да не може „да умре“. Понятията „смърт“ и „фатална грешка“ звучат като синоними, но ако погледнете как сме дефинирали тези понятия, ще видите, че те са различни.

В [14] има още един случай, когато програмата се прекъсва. Там живота се състои от 100 игри, всяка от които е не по-дълга от 1000 хода. Затова, ако една игра продължи повече от 1000 хода, тя се прекъсва. Това обаче не е истинско прекъсване, защото само се вмъква едно „реми“ без да се променя хода на работа на машина на Тюринг.

2.1.10 Пример

Ще използваме същия пример, който разгледахме в [12]. Това е света на играта Морски Шах, където устройството не вижда цялото табло, а само едно квадратче от него (Фигура 1).



Окото на устройството се намира върху квадратчето, което се вижда. Възможните ходове са шест. Окото може да се предвижва в четирите възможни посоки, можем да сложим кръстче в квадратчето, върху което окото се намира в момента и шестата команда е да поискаме нова игра (т.е. да се изчистят всички квадратчета и да започнем отначало).

В [12] светът използваше само булеви вектори, а тук ще се откажем от това ограничение, което ще направи дефиницията на света по-проста и съответно светът ще стане по-разбираем.

Вместо два булеви сигнала, които да кодират какво вижда окото, ще имаме един входящ сигнал с три възможни стойности {0, 1, 2}, които ще съответстват на празно, кръстче и кръгче. Вместо двата оценъчни сигнала "победа" и "загуба" ще имаме един с четири възможни стойности: {Nothing, 0, 1, 2}, които ще съответстват на "няма оценка", "загуба", "реми" и "победа". Вместо трите булеви изходни сигнала, които кодираха шестте възможни изхода, сега ще имаме четири изходни сигнала. Първите два ще дават посоката на движение на окото. Ще ги наречем vertical и horizontal. Техните възможни стойности ще бъдат в множеството {0, 1, 2}, което ще отговаря на "не мърда", "нагоре" и "надолу" или съответно на "не мърда", "наляво" и "надясно". Другите два изходящи сигнала ще са булеви и ще ги наречем put_cross и new_game. Какво правят последните два сигнала е ясно.

В [12] имахме шест възможни действия и на всеки ход можехме да извършим само едно от тях. Сега можем да извършим четири действия едновременно. Ако искаме, можем да не извършим и нито едно действие (като поставим нула на четирите координати на изходящия вектор). Бихме могли да предполагаме, че имаме право да извършим само едно действие и всеки друг изход се приема от света за некоректен ход, но по-интересно е да предполагаме, че можем да извършим до четири действия на един ход. Например, можем да поставим кръстче, да мръднем на горе и наляво и да поискаме нова игра. Разбира се, и четирите действия трябва да са коректни, защото в противен случай ще получим "некоректен ход" и нищо няма да стане.

Когато разрешаваме няколко действия в един ход трябва да уточним последователността им. Все едно е дали първо ще мръднем на горе и после на ляво или обратното. Все едно е дали първо ще се движим и после ще поскаме нова игра или обратното. Единственото действие, което не може да комутира с останалите е поставянето на кръстче, затова винаги ще предполагаме, че първо сме сложили кръстчето и после сме извършили другите действия.

По този начин представяме играта Морски Шах от [12] като свят с едно ниво на приоритет, с един входящ сигнал, четири изходящи и два оценъчни. (Вторият оценъчен сигнал е "некоректен ход", който си остава същия, както беше дефиниран в [12].)

Защо тук описаното представяне на света е по-добро от направеното в [12]? Защото имаме по-малко кодиране, което прави света по-прост и по-разбираем. Да вземем като пример правилото: "Ако клетката, която виждаш не е празна и ако се опиташ да сложиш кръстче, то това е некоректен ход." Сега това правило може да се запише като импликация съдържаща само три атома:
cell(t)=/=0, put_cross(t)=1 => bad_move(t+1)=1

В [12] тази импликация щеше да се състои от шест атома, защото сигнала **cell** там се кодира с два булеви сигнала и изхода там се кодира с три сигнала. Когато се опитваме да намерим зависимост без памет, каквато е горната, на нас ни се налага да намалим броя на импликациите като вземем само по-късите от тях. По тази причина, колкото по-къса е една импликация, толкова по-голям е шанса нашето устройство да я открие.

2.2 Как AI разбира какво се случва?

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 [28], където се търси една статична функция покриваща определени положителни и отрицателни примери. В статии като [29] 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 [29] ние отново търсим статична функция без памет, макар че на всяка стъпка тази функция е различна. В [29] 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 [30, 31]. 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 [19] тестове и съответстващите им функции на теста.

Тази статия по същество е продължение на [19], но, за да бъде разбрано написаното тук, не е задължително статия [19] да бъде прочетена. В [19] 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.

2.2.1 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 [19] 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 [19] 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 [30, 31]) 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 mod 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.

♦

2.2.2 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 [4]. Разликата ще е в това, че като основа на примера тук ще използваме играта шах, докато в [4] 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∈{Left, Right, Nothing}
vertical∈{Up, Down, Nothing}
command ∈{„вдигни фигурата“, „спусни фигурата“, New Game, Nothing}

Функцията View ще ни върне 3- tuple <chessman, color, immediate_reward>
chessman ∈{Pawn, Knight, Bishop, Rook, Queen, King, Nothing}

color∈{Black, White, Nothing}
immediate_reward∈{-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.

2.2.3 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 $a_1, v_1, \dots, a_{t-1}, v_{t-1}$, 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 [4]). 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: $a_{-k}, v_{-k}, \dots, a_0, v_0, \dots, a_s, v_s$. 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 В: Същото като дефиниция А 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 [19] разглеждаме зависимости без памет (т.е. събития по дефиниция А, при които дължините на u_1 и на u_2 са ограничени). В [19] стана дума за зависимости с памет (т.е. събития по дефиниция А, 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 А ни казва, че *past* трябва да завършва на нещо или да съдържа нещо. Дефиниция В 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.

2.2.4 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 [19] разказахме за това как ще събираме статистика за зависимостите без памет. Пак в [19] стана дума и за събитията с памет.

2.2.5 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 [19]). 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 [19] 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 [19] 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.

2.2.6 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 [19] 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 какво да направим? Първата възможност е да предположим, че теория на тестовото състояние ще даде прогноза за тази група с увереност нула. Втората възможност е да направим прогноза въпреки всичко. Погледнете отворения въпрос, който е по-долу.

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 състояние на автомата. Затова предпочитаме автомата да е детерминиран. В противен случай условието, че сме в съответното състояние на автомата няма де е изчислима функция.

2.2.7 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.

2.2.8 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.

2.2.9 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 [19]).

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.

Отворен въпрос: Понякога тестовото свойство не може да бъде тествано. Има два такива случая. Първият е, когато тестово свойство няма смисъл. Вторият случай е, когато има смисъл, но ние не можем директно да го тестваме и затова трябва да оценим стойността му индиректно. Въпросът е как да различим тези два случая? Например случая „къщата ни изгоря“ и случая „ръцете ми са вързани“. В първия случай тестово свойство „врата е

отключена“ въобще няма смисъл. Във вторият случай свойството има смисъл, но не можем да тестваме директно, защото ръцете ни са вързани. Можем да оценим свойството индиректно при помощта на правило от вида „Днес е понеделник, а в понеделник вратата винаги е отключена“. Това би било коректно разсъждение във втория случай, но не и в първия.

2.2.10 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 същественото от несъщественото. 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 споменем зависимостите със и без памет, за които се говори в [19]. 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 [18] както и в други по-ранни статии. Затова в [18] 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“.

2.3 Минимален и максимален модел при Reinforcement Learning

Всеки тест ни дава едно свойство, което ще наречем резултата на теста. Продължението на това свойство ще наречем тестовото свойство. Въпросът е, какво представлява това свойство? Дали това не е свойство на състоянието на света? Отговорът е и да и не. Ако вземем произволен модел на света, то отговорът е не, но ако изберем максималния модел на света, то тогава отговорът е да. Имаме различни модели. Минималният модел е този, при който света знае за миналото и за бъдещето минималното, което му е нужно. При максималния модел света знае всичко за миналото и за бъдещето. При този модел, ако хвърлите зар света знае какво ще се падне и дори знае вие какво ще направите. Например, знае дали ще хвърлите зара.

2.3.1 Въведение

Ние се опитваме да разберем света. За да го разберем, ще използваме тестове. Пример за тест е:

Ако натисна дръжката на врата $\Rightarrow$ вратата ще се отвори.

Тестовете съдържат някакво условие (предпоставка). В случая условието е, че съм натиснал дръжката на вратата. Когато условието е изпълнено, тогава теста е проведен и тогава получаваме резултата на теста, който е истина или лъжа. В случая врата се отваря или не се отваря.

Всеки тест ни дава едно свойство (резултата на теста). В случая свойството е „врата е заключена“. Ние не знаем каква е стойността на това свойство във всеки един момент, а само в моментите когато теста е проведен.

Ние ще предполагаме, че това свойство има смисъл във всеки един момент и ще се опитаме да продължим характеристичната функция на това свойство извън множество на моментите когато сме провели теста. Възможно е това свойство да не е тотално и да не е дефинирано във всеки един момент. Например, ако някой открадне врата, то въпросът дали тя е заключена губи своя смисъл. Тоест, ще се опитаме да продължим характеристичната функция на свойството, но не е задължително да получим тотална функция.

Каква е идеята в продължаването на резултата на теста до тестово свойство? Ако ви дам едно резенче (slice) от краставица, дали бихте могли от резенчето да възстановите цялата краставица? Разбира се, става дума за мислено възстановяване, а не за физическо.
Това възстановяване не е единствено и при него може да се получат различни обекти. Например, от резенчето от краставица можете да получите носорог, като си представяте, че това е резен от рога му. Разбира се, ние ще търсим продължения на резултата, които са максимално прости, естествени и достоверни.

Имайте предвид, че резенчето от краставица е нещо реално, докато самата краставица е нещо въображаемо и измислено. Ако ви дам да видите цялата краставица, то на вас ще е ви е по-лесно да си я представите, но вие може да си представите здрава краставица, а тя да се окаже изгнила отвътре. Тоест, никога вие не получавате цялата информация. Винаги получавате само една част (един резен информация), по който трябва да си представите целия обект.

Нека си зададем въпроса какво представлява това свойство (резултата на теста и неговото продължение). В различни моменти от времето свойството е истина или лъжа. Въпреки всичко, това не е свойство на времето, защото свойството не зависи толкова от конкретния момент, колкото от развитието на историята около този момент. Може би това е свойство характеризиращо състоянието на света (т.е. това е множеството от състоянията на света, при които свойството е истина).

Дали зависи свойството от миналото и от бъдещето? Обикновено тестът не се провежда в един конкретен момент, а в продължение на известно време. Тоест, резултата от теста зависи от времевия контекст (близкото минало и бъдеще в рамките на провеждането на теста). Ако вземем тестовото свойство, то то зависи от по-широк времеви контекст и може да ни казва нещо за далечното минало и бъдеще на света.

Например свойството: „В писмото има добра новина“. Съответния тест е: „Отварям писмото и проверявам какво пише в него“. Това свойство ни казва нещо за бъдещето. По точно, казва ни какво ще прочетем в писмото, когато го отворим. Дали това е свойство на света? Дали света знае какво пише в писмото още преди да сме го отворили. Обикновено си мислим, че знае, но би могъл е да не знае. Например, повечето компютърни игри не си дават труда да изчисляват целия свят, а се грижат само за тази част от света, която играчът вижда в момента. Ако света е такава игра, то той ще реши какво пише в писмото чак когато го отворите. Друг пример, нека живота е телевизионен сериал. Ако в 1354 серия вие получите писмо и десет серии по-късно го отворите, то кога сценариста ще реши какво

пише в писмото? Когато пише сценария на серията когато получавате писмото или когато пише серията когато го отваряте? Тоест, виждате, че света може предварително да знае, а може и да не знае предварително какво ще се случи в бъдеще.

Подобен е и въпросът с миналото. Например свойството: „Върнал съм се от почивка“ ни дава информация за миналото. Съответния тест е: Проверявам дали съм на почивка или в командировка и след това се връщам. Предполагаме, че ако сте се върнали от почивка и още сте си в къщи (никъде другаде не сте ходили), то продължавате да сте се върнали от почивка. Тоест, продължихме свойството за моменти, в които то не е тествано.

Нека предположим, че за бъдещето на света няма никакво значение дали сте се върнали от почивка или от командировка. Тогава има ли смисъл света да помни този факт? Въпросът, който вълнува историците е дали света помни миналото? Дали едно историческо събитие е оставило някакви документи или други следи за това, че се е случило? Отговорът е, че миналото може да се помни, а може и да не се помни.

В тази статия ще разгледаме два модела на света – минимален и максимален. При минималния светът помни минималното от миналото и знае минималното за бъдещето. При максималния е обратното, там светът помни всичко от миналото и знае всичко за бъдещето. Там светът знае точно какво ще се случи и дори знае вие (агента) какво ще направите.

2.3.2 Какво търсим?

Какво е дадено и какво се търси? В тази статия ще търсим обяснение на света. При Reinforcement Learning [30] имаме един агент, който живее в някакъв свят. Света е един ориентиран граф подобен на този на фигура 1. Агентът се движи от състояние на състояние по стрелките и събира някакви награди (rewards). За него ще е много важно да разбере света, за да може да намери наградите. Има много статии, където се предполага, че света е даден и се търси стратегия, която би била успешна в този свят (например в [33]). В тази статия ще предполагаме, че света е неизвестен.

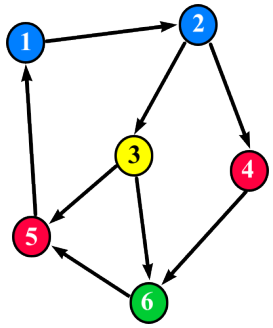


Figure 1

На фигура 1 възможните действия на агента са отбелязани със стрелки, а възможните наблюдения са отбелязани с различни номера и цветове. На всяка стрелка би трябвало да има етикет (label), който да казва на кое действие съответства тази стрелка. На тази фигура етикетите са изпуснати, но се вижда, че понякога има една възможна стрелка, а понякога възможностите са няколко (при състояния 2 и 3). Ще предполагаме, че не всички действия са възможни, както и че имаме недетерминирани преходи. Тоест, ще предполагаме, че при

конкретно състояние и конкретно действие може и да няма стрелка с този етикет излизаща от това състояние, както и че може такива стрелки да има повече от една.

Това, че допускаме недетерминирани преходи означава, че допускаме случайност. В [5] показахме, че има два вида случайност – прогнозируема и непрогнозируема. Тук използваме непрогнозируемата случайност (нещо се случва с вероятност в интервала [0, 1]). Тази случайност се използва още и при NFA (nondeterministic finite automaton). Там нещо може да се случи, а може и да не се случи, но не знаем с каква вероятност ще се случи. При POMDP (partially observable Markov decision process) се използва прогнозируемата случайност (нещо се случва с точно определена вероятност). В [5] доказахме еквивалентност между четирите вида модели (детерминирания, моделите с двете случайности и модела с комбинацията от двете случайности). Тоест, можем да работим, с който модел ни е най-удобно и ние сме избрали този.

Казваме, че имаме Full Observability когато можем по това, което виждаме да познаем кое е състоянието. Съответно, казваме че имаме Partial Observability в противния случай. Например, ако виждаме номера на състоянието, тогава имаме Full Observability, но ако виждаме само цвета, тогава не виждаме всичко. Например, ако виждаме „червено“ не можем да кажем дали сме в състояние 4 или в 5.

Ако имаме модела на света, то чрез него бихме могли да предскажем бъдещето. Например ако сме в състоянието 4 можем да предскажем, че следващото състояние ще е 6. Ако сме в червено състояние, можем да предскажем, че следващото ще е зелено или синьо. По същият начин както предсказваме бъдещето можем да предвидим и миналото. Трябва само да обърнем посоката на стрелките и милото става бъдеще. Единственият проблем е, че при обръщане на стрелките от детерминиран граф може да се получи недетерминиран, но ние избрахме да използваме като модели недетерминирани графи.

Какво ни е дадено? Дадена ни е историята до текущия момент. Тоест дадена ни е редицата от действия (outputs) и наблюдения (inputs или view).

$$a_1, v_2, a_3, v_4, \dots, a_{t-1}, v_t$$

Тук номерацията не показва номера на стъпката, а номера на момента. Във всяка стъпка има два момента. В първия извеждаме информация (това е нашето действие), а във втория въвеждаме какво виждаме. Тоест, номера на стъпката ще е номера на момента разделен на две.

Предпочитаме да разделим времето не на стъпки, а на моменти, заради събитийните модели където състоянията ще се променят в определени моменти. За една стъпка състоянието ще може да се промени два пъти, защото в стъпката има два момента.

Моментите ще са два вида – входящи и изходящи, които ще наричаме още четни и нечетни моменти.

Нека отбележим, че входовете и изходите ще са вектори от скалари. Ще предполагаме, че тези скалари са крайни. Бихме могли да се ограничим до булеви вектори, но няма да го направим за да избегнем излишното кодиране (виж [4]).

Към историята ще добавим още и некоректните ходове, които сме пробвали преди да изиграем поредния си ход.

$$bad_1, a_1, v_2, bad_3, a_3, v_4, \ldots, bad_{t-1}, a_{t-1}, v_t$$

Тук елемента ***bad*** можем да си го мислим като списък от некоректни ходове или като множество, защото реда, в който сме пробвали некоректните ходове, е без значение. Ще считаме, че некоректните ходове са пробвани в същия момент, когато е изигран съответния коректен ход (затова списъка ***bad*** и следващия след него коректен ход ***a*** имат еднакъв индекс).

Дефиниция: Живот ще наричаме история, която не може да бъде продължена.

Една история не може да бъде продължена, ако е безкрайна или ако завършва със състояние, от което не излизат никакви стрелки. В [3] такива състояния нарекохме „внезапна смърт“.

2.3.3 Двоен модел

Фигура 1 е направена на базата на стандартния модел, който е базиран на стъпки. Ние ще използваме двойния модел, който е базиран на моменти.

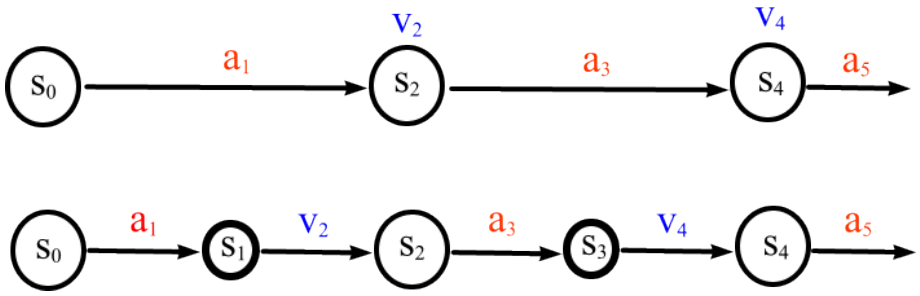


Figure 2

Разликата между стандартния и двойния модел е илюстрирана на фигура 2. Двойния модел се получава от стандартния като всяко състояние бъде заменено от две състояния и стрелка помежду им. Всички стрелки, които са влизали в състоянието, сега ще влизат в първото, а стрелките които са излизали от състоянието, сега ще излизат от второто. Етикета, който е бил на състоянието сега е етикет на стрелката между новите състояния. При двойния модел само стрелките имат етикети, а състоянията нямат. Тук сме отбелязали нечетните състояния с по-малки кръгове, за да подчертаем, че имаме два вида състояния.

Забележка: Тук ще считаме, че на състоянието съответства времеви период, а на стелката съответства само един миг, който е началото на следващия времеви период. При събитийните модели ще считаме, че на състоянията съответстват дълги периоди от време, а на стрелките съответстват съвсем къси периоди (от един момент или малко повече).

Изглежда сякаш в двойния модел състоянията са два пъти повече, но това не е така, защото ако две състояния са еквивалентни от гледна точка на бъдещето, то те мога да бъдат обединени (слети) в едно. В стандартния модел могат да бъдат обединени само състояния, които са еквивалентни от гледна точка на настоящето и на бъдещето. Тоест, в стандартния

модел състоянието трябва да помни настоящето, докато в двойния няма такава нужда. Това е една от причините, поради която въвеждаме този модел. В тази статия за нас ще е много важно каква е информацията, която можем да извлечем от състоянието. Тоест, какво може да каже състоянието за миналото и за бъдещето. Настоящето е част от миналото, защото то вече се е случило.

В двойния модел състоянията ще са два вида. Състояния след вход и състояния след изход. Ще ги наричаме още четни и нечетни състояния. По-важни са четните състояния, защото това са състоянията, в които мислим. В нечетните състояния ние не мислим, а само чакаме да видим каква информация ще ни даде света. (Може да предполагаме, че в нечетните моменти мисли светът. Така се редуваме, в един момент мислим ние, а в следващия мисли светът.)

Да вземем като пример света, в който играем шах срещу въображаем противник. Нашите действия ще са вектори от вида (x_1, y_1, x_2, y_2). Тези вектори описват нашия ход. Ще виждаме хода на противника и оценката (reward). Тоест, входа ще бъде вектор от вида (x_1, y_1, x_2, y_2, R). При двойния модел състоянията ще бъдат позициите на дъската. Четните състояния ще са тези, при които белите са на ход (т.е. ние сме на ход). Когато нашия ход завършва играта (например мат), тогава противника ще трябва да играе някакъв празен ход и да ни даде само оценката на играта. Т.е. вектор от вида (0, 0, 0, 0, R). Този празен ход трябва да премине към началната позиция, когато играта започва отначало.

При стандартния модел нещата ще са много по-сложни. Тогава състоянията ще са само позициите, при които белите са на ход, но ще трябва да се помни още и хода на противника довел до тази позиция. Тоест, състоянията ще са повече, защото позициите когато белите са на ход са два пъти по-малко, но ако всяка от тях може да се получи по средно 100 начина (от 100 различни позиции с 100 различни хода на противника), тогава, като теглим чертата получаваме, че стандартният модел ще има 50 пъти повече състояния.

Както казахме, при стандартния модел настоящето се налага да се помни, а в двойния не се налага. Да видим как стои въпросът с бъдещето. Ако играем шах срещу детерминиран противник, то тогава и двата модела ни дават детерминиран граф. Нека допуснем, че противника е недетерминиран и че за всяка позиция има по няколко възможни хода, които той може да изиграе. Тогава стандартния модел ни дава недетерминиран граф (един наш ход води до няколко различни ответни хода и съответните им позиции). Двойния модел пак си е детерминиран, защото нашия ход води до една определена позиция. От тази позиция излизат няколко възможни стрелки, които съответстват на възможните отговори на противника (ако противникът беше детерминиран щеше да излиза само една стрелка.)

Винаги ли двойният модел ни дава детерминиран граф? Не винаги, но винаги можем да го сведем до детерминиран. Да си представим същата игра, но сега няма да виждаме хода на противника, а само ще виждаме коя фигура е преместил. Тоест, няма да виждаме (x_2, y_2). Сега, когато знаем коя е фигурата, но не и къде е преместена, ще имаме няколко възможни позиции. Естествено би било да представим тази игра с недетерминиран граф, но можем да го направим и с детерминиран граф. Ако състоянието на света не е конкретна позиция на дъската, а е множество от възможни позиции, тогава на конкретния ход ще отговаря една единствена стрелка, която ще води до множество от възможни позиции. Тоест, при първия (недетерминирания) вариант света знае нещо повече за бъдещето, отколкото при втория. При детерминирания вариант света не знае коя точно е позицията. Знае кое е множеството

от възможните позиции, а коя е точно позицията ще разбере по-късно, когато това проличи по входа (по наблюденията). Това е като с писмото. В единия случай света знае какво пише в писмо, а във втория случай ще реши какво пише, чак когато ти отвориш писмото.

2.3.4 Минимален модел

Един двоен модел на света ще го наричаме минимален, ако света знае за миналото и бъдещето само толкова колкото му е необходимо. При минималния модел, ако две състояния са еквивалентни спрямо бъдещето, то те съвпадат. Тоест, не се помни нищо от минало, което не ни е нужно, за да определим бъдещето.

Освен това, минималния модел е детерминиран граф. Това означава, че разклоненията са избутани максимално напред към бъдещето. Тоест, всяко нещо за бъдещето се разбира чак когато му дойде времето (когато то повлияе на наблюдението, но не по-рано).

Минимален модел не значи с най-малко състояния. Спрямо миналото минималността действително намалява състоянията, но спрямо бъдещето по-скоро ги увеличава (защото преминаваме от конкретни възможности към множества от конкретни възможности).

Процедурата по детерминизация винаги може да се приложи и винаги можем да получим детерминиран граф. Това, че графа е детерминиран не означава, че агента е детерминиран или че света е детерминиран. За да бъде агента детерминиран, трябва да няма разклонения в четните състояния. (Тук под детерминиран агент имахме предвид, че е принуден да играе детерминирано, защото има само един коректен ход. Обикновено под детерминиран агент разбираме такъв, който играе детерминирано без да е принуден да го прави.) За да бъде света детерминиран трябва да няма разклонения в нечетните състояния. Това, че графа е детерминиран означава, че състоянията знаят за бъдещето минималното. (Ако две състояния са различни, то те имат различно минало. Т.е. различни са защото имат различно минало, а не защото знаят нещо повече за бъдещето.)

2.3.5 Тотален модел

Когато един ход е некоректен стрелката по този ход просто липса и този ход просто не може да бъде направен. Бихме искали агента да може да пробва некоректните ходови и като ги пробва нищо да не се случи. Т.е. той да получи информация, че хода е некоректен, но да остане в същото състояние.

Тази цел ще я постигнем като добавим към всяко четно състояние (при което има некоректни ходове), по едно допълнително нечетно състояние (виж фигура 3). Всички некоректни ходове от четното състояние ще ги насочим към нечетното. От там ще се върнем обратно с една стрелка с етикет „bad“. Този етикет ще е един специален нов вектор, който сме добавили за целта и който ще получаваме като вход само когато опитаме некоректен ход.

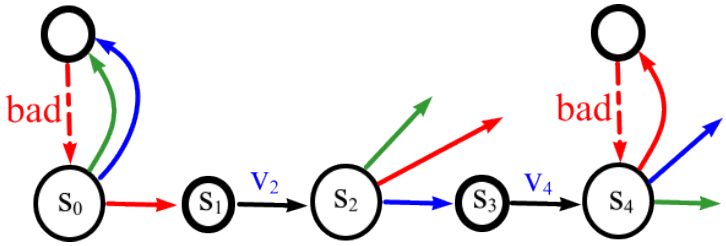


Figure 3

На фигура 3 възможните ходове са отбелязани със стрелки с червен, син и зелен цвят. В състоянието s_0 има два некоректни хода, а в състоянието s_4 има само един. В състоянието s_2 няма некоректни ходове и затова за него не сме добавили допълнително нечетно състояние.

По този начин ще получим един тотален модел, при който всички ходове могат да се пробват, но коректните ходове реално променят състоянието на света, докато некоректните само дават информация, че са некоректни. Тоест, получаваме един нов тотален модел, който описва същия свят като предишния модел, с тази разлика, че некоректните ходове вече могат да се пробват.

2.3.6 Максимален модел

След като имаме минимален модел, може би има и максимален. Това трябва да е модела, при който състоянието знае всичко за миналото, всичко за това кои са коректните ходове и всичко за бъдещето.

Да знаем всичко за миналото е лесно, просто когато вървим назад (срещу стрелките) не трябва да има разклонения. Тоест, ако модела има формата на дърво, тогава състоянието ще знае всичко за миналото (тръгвайки от него назад ще можем да възстановим цялата история).

Да знае състоянието кои са некоректните ходове също не е трудно, защото това е крайна информация, която лесно ще добавим.

За да знаем всичко за бъдещето, трябва да няма недетерминираност. Тоест, ако хвърлим зар, трябва да знаем какво ще се падне. Ще построим модел, при който цялата недетерминираност ще е концентрирана в началното състояние. След като изберем началното състояние (по недетерминиран начин) нататък всичко ще е детерминирано.

Нека вземем дървото на всички достижими състояния. (Това е дърво, ако има еквивалентни състояния не ги сливаме в едно.) Ще детерминираме това дърво, макар това последното да не е задължително. От така полученото дърво ще получим всички стратегии (policy) на света. Това са поддървета, в които няма разклонения по наблюдението, а разклоненията по действията се запазват. Тези дървета са много (the cardinality of the continuum). От всичките тези дървета правим модел, където началните състояния ще са корените на всичките тези дървета.

Тук думата стратегия не е много подходящо да се използва, защото казваме стратегия, когато имаме някаква цел. Тук предполагаме, че агента има цел, а света няма цел. Ако предположим, че целия свят се опитва да ни помогне или да ни попречи, то това би било

твърде егоцентрично. Въпреки това, ще предполагаме, че в света има агенти, които имат своите цели. Тоест, света няма цел, но въпреки това различните поведения на света ще наричаме стратегии.

Следователно направихме модел, който се състои от всички стратегии на света. Още преди да започне живота света избира по случаен начин една от своите стратегиите и я следва до края на живота на агента. Идеята е, още преди да започне играта да намислим как ще играем. Можем да намислим, че ако той ни играе с коня, ние ще отговорим с офицера и т.н. Едно такова намисляне представлява стратегия и се изразява с едно безкрайно дърво. Тези дървета са неизброимо много. Може да си намислим, че ще играем използвайки определена детерминирана програма, но по този начин ние може да си намислим само някоя изчислима стратегия, а те са много по-малко (само изброимо много).

Полученият модел е еквивалентен на модела, от който тръгнахме, защото всяка история, която е възможна при единия модел е възможна и при другия. При новия модел света се държи детерминирано, с изключение на първия момент, когато избираме началното състояние.

В този свят, ако вземем произволно състояние, то то знае почти всичко за бъдещето, с изключение на това, че не знае какво действие ще избере агента. Бихме искали да направим модел, при който дори и това да се знае от състоянието на света.

Тук има един проблем. Обикновено предполагаме, че света е даден, а агента е произволен. Тоест, няма как света да знае какво ще направи агента, защото той си няма идея кой агент ще му се падне. Сега ще предположим, че агента е фиксиран и че света би могъл да знае нещо за агента. Например, света би могъл да знае, че в определена ситуация агента няма да изиграе определен ход, макар че този ход е коректен и би могъл да бъде изигран. Това, че определен ход няма да бъде изигран от агента ще го отбележим с липсваща стрелка в графа на модела.

Сега ще предположим, че още преди да започне живота света и агента са решили как да играят. Тоест, избрали са своята стратегия, която ще следват до края на живота на агента. Това може да се опише с наредена двойка от две безкрайни дървета или чрез един живот (безкраен път в дървото). Това е така, защото резултата от прилагането на две фиксирани стратегии е един фиксиран път.

Така получихме максималния модел на света. Той се състои от всички възможни животи (това са пътища в дървото на достижимите състояния). Единственото, което ни липсва, това е информацията за некоректните ходове. За целта ще добавим примки (loops), както направихме, когато правехме модела да е тотален. Тук няма да получим тотален модел, защото ще добавим примки само по некоректните ходове, а не по всички липсващи стрелки. Така получаваме модела изобразен на фигура 4. (При s_2 няма примка, защото от това състояние не излизат некоректни ходове)

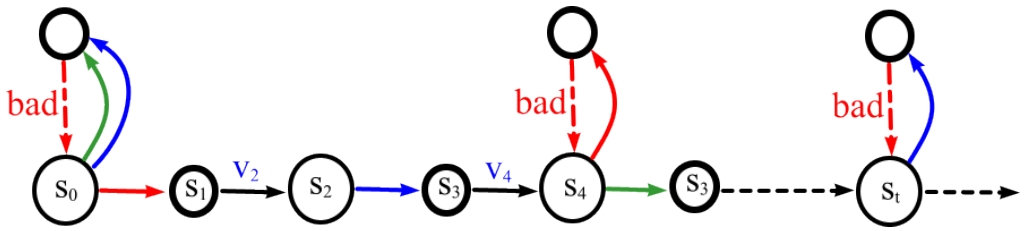


Figure 4

На тази фигура е изобразен само един живот, а не всички възможни животи, които са част от максималния модел. Нас ни интересува само един живот и това е живота, който живеем. Затова може да предполагаме, че максималния модел съдържа само един живот и това е живота, който ни интересува. По този начин ще загубим еквивалентността с първоначалния модел, но получения модел е това, което ни трябва, защото другите възможни животи не са важни.

В така получения максимален модел от всяко състояние може да се възстанови цялата история и дори целия живот. Пътищата напред и назад са без разклонения. Примките, които добавихме, няма да ги броим за разклонения, защото символа *bad* не се среща при наблюденията след коректен ход. Тук състоянието знае кои ходове са некоректни, но не знае кои от тях агента е пробвал. Тази информация не е важна за света, защото от нея не зависи нито миналото нито бъдещето. Разбира се, тази информация е важна за агента, защото той може да не знае кои ходове са некоректни и затова му е полезно да знае кои ходове вече е пробвал.

2.3.7 Заключение

Опитваме се да разберем света, тоест да намерим неговия модел. Проблемът е, че този модел въобще не е единствен. Може да имаме недостижими и еквивалентни състояния, но това не е проблем. Може да има части на света, които никога не сме посещавали и които никога няма да посетим. Например, има ли живот на Марс? След като никога не сме ходили там и никога няма да отидем, то това е без значение (тоест, това няма да се отрази на нашия живот).

Най-съществения проблем е, че имаме модели, в които света знае повече и модели при които света знае по-малко. Кой модел търсим? Отговорът е, че търсим максималния модел. Тоест, ние ще се стремим максимално да разберем миналото и максимално да предскажем бъдещето. Ще се опитаме да предскажем дори и собственото си поведение. Ние сме част от света и за да го разберем трябва да се опитаме да предскажем дори и собственото си поведение.

За да опишем максималния модел ние ще използваме така наречения разширен модел. В този модел ще представим състоянието на света като вектор с огромен брой координати (променливи). Те ще са хиляди или дори безбройно много. От теоретична гледна точка, те ще са безбройно много, но на практика ще изберем само най-интересните от тях. Може би това е модела, за който говори Sutton в [32] (state representation which contains many state variables).

Първите координати (променливи), които ще сложим във вектора описващ разширеното състояние, това ще е това, което виждаме в момента. При двойния модел наблюдението не

е функция на състоянието на света, защото, ако състоянието е четно, то може да има много стрелки с различни наблюдения, които да влизат в него. Ако състоянието е нечетно, то съответно може да има много стрелки, които да излизат от него. Тук обаче говорим за максималния модел и при него винаги има само една влизаща и една излизаща стрелка. Тоест, при максималния модел какво виждаме в момента се определя от състоянието на света.

Към това, което виждаме в момента, ще добавим още тестовите свойства на различни тестове. Това не са резенчетата от краставица, които са нещо реално. Тава са измислените краставици, които ние сме измислили на базата на реалните резенчета. Тоест, разширения модел няма да е нещо реално, а ще е нещо измислено.

Въпрос: Ако имаме Full Observability, тогава има ли нужда да добавяме тестови свойства към вектора на състоянието на разширения модел? Отговор: Да има нужда, защото при Full Observability ние знаем кое е състоянието на света, но това е състоянието при някой модел, който не е максималният. Ако имаме Full Observability с максимален модел, то света е толкова елементарен, че чак не е интересен.

Как да измислим едно тестово свойство? Помислете си за свойството „заключена ли е вратата“. Пробваме и получаваме ту „заключено“, ту „отключено“. Много трудно е да направим модел и да разберем кога е заключено и кога отключено. Много по-голям успех ще имаме, ако допуснем, че вратите са повече от една (особено, ако те действително са повече). В новият модел трябва да имаме идея пред коя врата сме застанали в момента. Тогава може някои врати да са постоянно отключени, други постоянно заключени, а трети да променят състоянието си по някакви правила. В този случай няма да добавим към разширения модел една променлива, която да отразява едно тестото свойство. Ще добавим много променливи, по една за всяка врата (която да отразява свойството на вратата) и една променлива, която да отразява пред коя врата сме в момента. Това представяне в [5] беше наречено тестово състояние.

Пред коя врата сме застанали в момента се определя от събитийните модели, които ще разгледаме в следващата статия.

2.4 Събитийни модели

При Reinforcement Learning търсим смисъл в потока входно-изходна информация. Ако не намерим смисъл, този поток за нас ще бъде просто шум. За да намерим смисъл трябва да се научим да откриваме и разпознаваме обекти. Какво е обект? В тази статия ще покажем, че обекта е event-driven модел. Тези модели са обобщение на action-driven моделите. При Markov decision process имаме action-driven модел и там състоянието на модела се променя на всяка стъпка. Предимството на event-driven моделите е, че те са по-устойчиви и променят състоянието си само при настъпването на някакви определени събития. Тези събития може да се случват много рядко и затова състоянието на event-driven моделите е много по-предвидимо.

2.4.1 Въведение

Какво е обект? Пример за обект може да бъде вашата любима песен, определена дума или синтактична категория (като глагол), това може да е човек, предмет или животно, това може да е вашата къща или квартала, в който живеете.

Вие можете да разпознавате обекти. Вашата любима песен ще я познаете дори и да чуете само част от нея. Дори и певеца да е друг, дори и качеството да е лошо, вие пак ще разпознаете песента.

Обектите в главата ви са подредени йерархично. Пример за йерархия: животно, куче, пудел, вашето куче. Някой обекти притежават свойството единственост. Например вашето куче е единствено и ако го подстрижете, то то ще е подстригано. Ако подстрижете произволно куче, от там няма да следва, че всички кучета ще бъдат подстригани.

Свойството единственост няма да е характеристика, която да не може да се промени. Например смятате един човек за единствен, но установявате, че той има брат близнак. Мислите, че сте говорили с един и същи човек, но се оказва че сте говорили с двама различни. Друг пример. Във вашата къща имате стол. Смятате, че този стол е единствен. Знаете, че са произведени хиляди такива столове, но този е единствения такъв стол, който имате вкъщи. Боядисвате стола и очаквате следващия път като го видите той да е боядисан. Може да се окаже, че във вашата къща има два такива стола, а вие сте боядисали само единия.

Как разпознаваме обектите? Вашата любима песен може да я чуете от начало, от средата или от края. Човека можете да го видите в профил или в анфас. Квартала може да го видите тръгвайки от много различни кръстовища. Тоест, за да разпознаем един обект, на нас не ние е нужно да го видим по съвсем същия начин, по който вече сме го виждали.

Ориентираният граф също е обект и ние го разпознаваме когато минем определен път в този граф. Разбира се, пътя трябва да е достатъчно характерен, за да сме сигурни, че това е точно този ориентиран граф. Ако не сме сигурни, може да направим експеримент и да завием в тази посока, която ще ни даде повече информация. Например, вие сте на едно кръстовище, но не знаете дали сте във вашия квартал. Тръгвате към главната улица, за да се ориентирате. Друг пример. Виждате предмет, който не можете да разпознаете. Започвате да го въртите в ръцете си и да го оглеждате. Виждате познат човек, но не сигурни, че е той. Започвате да го оглеждате, викате го по име, за да видите дали ще се обърне.

Не винаги имаме контрола и не винаги можем да направим експеримент. Когато чуем една дума понякога не можем да разберем какво сме чули. Може да имаме възможност да помолим да ни повторят думата или ако е на запис може да върнем записа и да я чуем отново, но в повечето случай се получава: който разбрал – разбрал.

В тази статия обектите ще ги представяме като ориентирани графи (event-driven модели). Повечето обекти не са ни постоянно пред очите. Тоест, обектите се появяват и изчезват от нашето полезрение. Има ли обекти, които наблюдаваме постоянно. Има ли ориентирани графи, в които ние се намираме постоянно и никога не ги напускаме. Има такива обекти (модели). Например дните от седмицата. Имаме един модел със седем състояния (дните от седмицата) и ние постоянно сме в едно от тези седем състояния. Друг пример е нашия

квартал. Ако цял живот никога не излезем от нашия квартал, то това ще е обект, който винаги е в полезрението ни.

Повечето event-driven модели ще имат едно специално състояние, което ще наречем *outside*. Ще считаме, че обекта е извън нашето полезрение, когато сме в състоянието *outside*.

При event-driven моделите основният въпрос ще бъде „Къде съм?“ Тоест, в кое състояние съм? Ако event-driven модела е обект, тогава основният въпрос е „Виждам ли обекта?“ Тоест, дали съм в състоянието *outside* или в някое от другите състояния.

Тази статия ще я започнем представяйки интуитивна идея за event-driven моделите. После ще направим сравнение между тези и action-driven моделите. Ще опишем задачата неформално и ще кажем защо няма да търсим началното състояние. Ще дадем дефиниция на история и ще кажем каква е целта на агента. Ще покажем, че можем да имаме много различни критерии и че Sutton в [32] не е прав като изказва, че има само един критерии. Ще покажем, че понятието discount factor не трябва да участва в дефиницията на Reinforcement Learning (RL), защото той е свързан със стратегията, а не със смисъла на живота. Ще обсъдим дали модела на света трябва да е детерминиран или да не е. За некоректните ходове ще покажем, че можем да минем и без тях, но е по-добре да предполагаме, че има такива. Ще дадем дефиниция на RL. После ще дефинираме съвършения модел, който е action-driven. Ще усложним този модел като добавим случайност. Ще покажем, че трябва да има отпечатък, тоест трябва да се случва нещо специално, което да отличава различните състояния. Ще въведем пълните модели и ще покажем, че те са също така непостижими като съвършените. Ще кажем какво е събитие и ще направим следващото усложняване на модела, което ще го превърне от action-driven в event-driven. Тоест, ще заменим действията с произволни събития. Ще добавим към модела и променливи и ще направим декартовото произведение от всички адекватни модели, които сме открили. Това декартово произведение ще бъде модела, който търсим и който възможно най-добре описва света.

2.4.2 Интуитивна идея

Преди да сме объркали читателя с много приказки ще се опитаме да дадем интуитивна идея за това какво представлява event-driven модела. Това е ориентиран граф подобен на фигури 1 и 2.

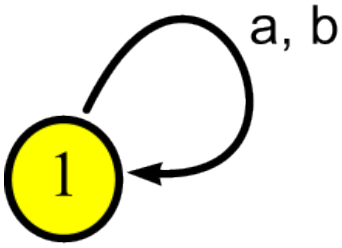


Figure 5

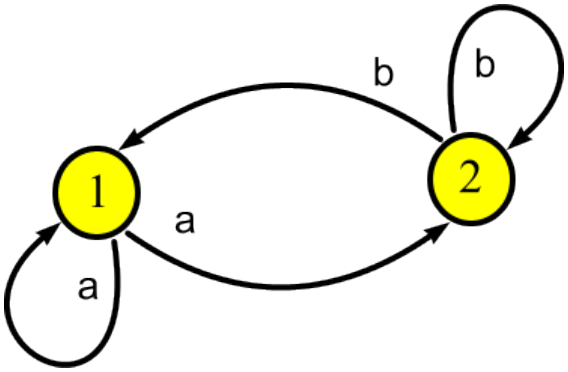


Figure 6

Тук *a* и *b* са събития. Ако *a* и *b* бяха действия, то тогава фигури 1 и 2 щяха да изобразяват action-driven модели. Разбира се, действието е събитие и затова action-driven моделите са частен случай на event-driven моделите.

Модела на фигура 1 е много елементарен и ако направим статистика по него, единственото което ще отчетем е колко пъти се е случило събитието *a* и колко пъти събитието *b*. Тоест, ще имаме идея кое събитие е по-вероятно да се случи.

Модела на фигура 2 е по-интересен. Тук, ако сме в състоянието 1 събитието *b* не може да се случи. Същото можем да кажем за състоянието 2 и събитието *a*. Тоест, този модел ни предсказва кое ще е следващото събитие (*a* или *b*). Ако знаем в кое състояние сме, ще знаем кое ще е следващото събитие.

Ако обърнем стрелките (фигура 3), ще получим модел, който помни кое е последното събитие (*a* или *b*). Ако знаем в кое състояние сме, ще знаем кое събитие последно се е случило.

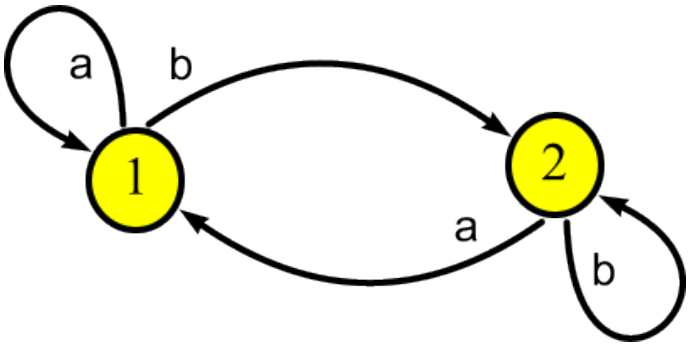


Figure 7

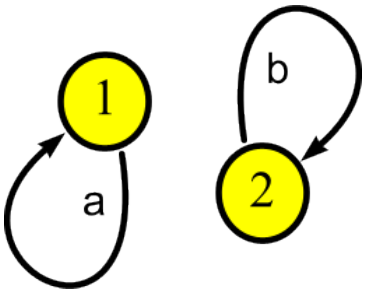


Figure 8

Без значение дали разглеждаме фигура 2 или 3, и в двата случая, трябва да има разлика между състоянията 1 и 2. Трябва нещо специално да се случва в някое от тези състояния. Това нещо ще го наречем отпечатък. Ако няма отпечатък, т.е. ако няма нищо специално, то тогава двата модела са безсмислени.

При фигура 2 има разлика между състояния 1 и 2 и тя е, че в 1 излизат стрелки само по *a* (и в 2 само по *b*). Ако няма никаква друга разлика между 1 и 2, тогава няма да можем да кажем в кое състояние сме, защото тези състояния ще са напълно неотличими спрямо миналото.

При фигура 3 знаем точно в кое състояние сме, но това няма никакъв смисъл, ако тези състояния са напълно неотличими спрямо бъдещето. Тоест, няма да има никакъв смисъл да помним дали последно се е случило *a* или *b*, ако от това не следва нищо за бъдещето.

Като пример нека да разгледаме следната криминална история (фигура 4). Търсим кой е убиеца. Дали е пощальона или скитника? Ако убиеца е пощальона, тогава сме в състояние 1. Ако убиеца е скитника, тогава сме в състояние 2. Събитието *a* е „доказахме, че убиеца е пощальона“. Съответно събитието *b* е „доказахме, че убиеца е скитника“. Въпросът е, в кое

състояние сме? Тоест, кой е убиеца? Когато се появи доказателството (събитието a или b) ще разберем. За съжаление, доказателството може и никога да не се появи. Въпросът е, как да разберем кой е убиеца преди да се е появило доказателството. Може да предположим, че когато убиецът е пощальона е по-вероятно да се появи улика потвърждаваща това, отколкото улика отричаща това. Тоест, уликите не доказват кой е убиеца, но ни дават кое от двете е с по-голяма вероятност.

Модела на фигура 4 на нас не ни харесва защото ние искаме да имаме повторяемост и по всяка стрелка да сме минали многократно. В този модел, ако се е случило a, то тогава b няма как да се случи.

Повторяемостта е много важна. Хераклит е казал, че човек не може да влезе два пъти в една и съща река. Реката всеки път е различна. По тази логика не можете да влезете два пъти в една и съща стая и не можете да срещнете два пъти един и същи човек. Стаята всеки път е различна. Някой е боядисал стените, друг е разместил мебелите. Един е влязъл, друг е излязъл. Ако всичко е по старому, ще се появи някоя муха, която ще попречи на повторяемостта.

Въпреки всичко, ние искаме да влизаме много пъти в една и съща река, и в една и съща стая и да срещаме един и същи човек. Те може да не са съвсем едни и същи, но ние сме готови да пренебрегнем малките разлики и дори и големите разлики, но да направим света по-прост и по-разбираем.

Това е идеята която стои зад event-driven моделите. Това са модели, при които имаме малко състояния, които посещаваме многократно и всяко наше посещение е дълго (състои се от много стъпки). Обратното е при action-driven моделите, особено ако модела е съвършен (виж по-долу). Съвършен модел винаги съществува и най-лесния начин да направим такъв модел е да наредим състоянията в редица и да минем през тях еднократно. В съвършеният модел състоянието описва всичко и затова е почти невъзможно едно и също състояние да се повтори два пъти (както не можем да влезем два пъти в една и съща река).

2.4.3 Event-Driven vs Action-Driven

Каква е разликата между event-driven и action-driven моделите. При Markov decision process (MDP) модела е един ориентиран граф, който променя състоянието си след всяко действие (т.е. на всяка стъпка). Такъв модел е като машина, която щрака твърде бързо. Много трудно е при такъв модел да си отговорим на въпроса „Къде съм?“. Тоест, много трудно е да кажем кое е текущото състояние. Да си представим, че човекът е стъпково устройство, което прави 24 стъпки в секунда. (В киното кадъра се сменя 24 пъти в секунда и ние приемаме действието като непрекъснато. Тоест, 24 стъпки в секунда е едно добро предположение.) Ако текущото ви състояние се променя 24 пъти в секунда, едва ли бихте ли могли да кажете кое е текущото ви състояние. Нека вземем модел, който не се променя при всяко действие, а при появата на някакви по-интересни събития. Пример за такива събития са изгрева и залеза. Те се случват по веднъж в денонощието. След изгрева от нощ преминаваме към ден, а след залеза обратното. Ако ви попитам дали в момента е ден или нощ вие вероятно ще успеете да ми отговорите. Тоест, текущото състояние на event-driven модела е много по-предвидимо (защото е много по-стабилно и по-рядко се променя).

Друга разлика е, че при action-driven моделите може да се опитаме да опишем поведението на света без това да включва поведението на агента докато при event-driven моделите се налага да описваме света и агента живеещ в него като една единна система.

Например при MDP изчисляваме каква е вероятността за определен преход при определено състояние и определено действие. Ние обаче не изчисляваме каква е вероятността агента да извърши определеното действие, защото се опитваме да опишем само света без да описваме агента. При event-driven моделите се налага да опишем света и агента като единна система, защото когато се е случило едно събитие ние не можем да кажем, дали това е само по волята на агента или това се е случило, защото светът е такъв какъвто е. При MDP ние не предсказваме миналото, защото по коя стрелка сме влезли в едно състояние зависи не само от света, но и от агента. Например (фигура 5), ако сме в състоянието 2 и последното действие е червена стрелка, тогава може да сме дошли от състоянията 1 и 5. Ако знаем, че в състоянието 1 агента никога не би избрал червена стрелка, тогава предишното състояние трябва е било 5.

При event-driven моделите ние не правим разлика между миналото и бъдещето. Събираме статистика и за едното и за другото, опитваме се да предскажем и едното и другото.

Много по-лесно е да опишем света и агента като единна система, отколкото да опишем само света без агента. Например, вие няма да си боядисате косата зелена и това е едно просто описание на вас и на света, в който вие живеете. Дали няма да го направите защото сте достатъчно разумен или не можете да го направите защото нямате зелена боя, това е без значение. Ако някой ви попита дали утре ще сте със зелена коса, можете спокойно да му кажете, че това не може да се случи. Дали това го знаете от анализа на вашето собствено поведение или от някакви ограничения, които са наложени от света, това е без значение. Важното е, че косата ви не може утре да е зелена.

2.4.4 Неформално описание

Първо ще опишем задачата неформално. Имаме един свят (environment) и един агент, който живее в този свят. Света е един ориентиран граф подобен на този на фигура 5. Агентът се движи от състояние на състояние по стрелките. Състоянията и стрелките са етикирани с някакви етикети. На фигура 5 вместо етикети сме използвали различни цветове. Докато се движи агента въвежда (observe) информация (това е етикета на състоянието, в което се намира) и действа (избира етикета на следващата стрелка, по която ще мине). Да отбележим, че агента избира само етикета (цвета) на стрелката, но не и конкретната стрелка. Например в състояние 6, ако избере да продължи със синя стрелка, то не се знае дали ще отиде в състояние 5 или в 7. Може някоя възможност да не съществува. Например от състояние 2 не излиза нито една червена стелка. Тоест, в това състояние агента няма право да избере да продължи по червена стрелка и задължително трябва да избере синя.

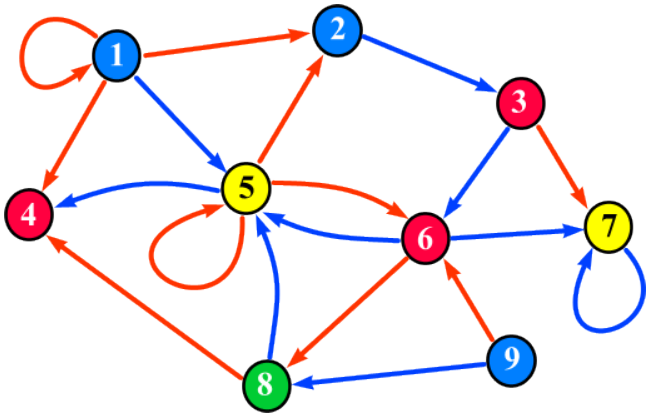


Figure 9

Пътя на агента в ориентирания граф ще наречем живота на агента. Всяко състояние може да бъде начало или край на живота, но има състояния, които, ако участват в пътя, задължително са неговото начало. Такива състояния ще наречем абсолютно начало. Това са състояния, в които не влизат стрелки. Тоест, преди тях не може да има минало. На фигура 5 такова състояние е 9. Състоянието 1 не е абсолютно начало, заради червената примка (loop). Аналогично, състоянията, от които не излизат стрелки, ще наречем внезапна смърт. Такова състояние, ако участва в пътя, то задължително е последното. След такова състояние няма бъдеще. На фигура 5 такова състояние е 4. Състоянието 7 не е внезапна смърт, заради синята примка.

Идеята да наречем състоянията, от които не излизат стрелки, „внезапна смърт“ е това, че живота на агента може да завърши по два начина – когато ние го изключим или когато той попадне в такова състояние и спре. Когато ние изключваме агента, ще казваме, че имаме естествена смърт, а когато той сам спре, ще казваме, че това е внезапна смърт. (В естествения език, когато някой ни изключи не говорим за естествена смърт, а за убийство. Тук, обаче, ще предполагаме, че изключваме агента, когато е минало много време. Тоест, изключването ще го асоцииираме с умиране от старост и затова ще го наричаме естествена смърт.)

Нека отбележим, че живота не е задължително да започва от някое абсолютно начало. Миналото може да е безкрайно, по същия начин както безкрайно може да бъде и бъдещето. Тоест, живота може да е един безкраен път без начало и без край.

При MDP и при RL разглеждаме два случая – когато агента вижда всичко (Full Observability) и когато той вижда частично (Partial Observability). На фигура 5 бихме имали Full Observability, ако всички състояния имаха различни цветове или ако агента виждаше не цвета, а номера на състоянието (тоест, ако агента виждаше кое точно е състоянието, в което се намира).

В повечето статии се предполага, че при MDP имаме частния случай (special case) на Full Observability, а когато имаме по-общия случай (generalization) на Partial Observability те използват термина partially observable Markov decision process (POMDP). В тази статия ще е обратното. С MDP ще означаваме по-общия случай, а когато имаме Full Observability ще използваме термина fully observable Markov decision process (FO MDP).

Аналогично, тук ще предполагаме, че при Reinforcement Learning имаме Partial Observability. Когато разглеждаме случая на Full Observability, специално ще отбелязваме това.

2.4.5 Начално състояние

При Reinforcement Learning (RL) се говори за начално и за текущо състояние. За да опишем един път (живот) на нас ни е нужно състояние, от което да тръгнем. Обикновено авторите предпочитат началното състояние, защото предполагат, че напред към бъдещето е по-лесно да се предсказа, отколкото назад към миналото. Тоест, да се върви по стрелките е по-лесно, отколкото да се върви срещу стрелките. Например, при Markov decision process (MDP) имаме вероятности, които ни казват кое очакваме да е следващото състояние, ако вървим напред по определено действие, но нямаме вероятности в обратната посока. В тази статия няма да говорим за начално, а само за текущото състояние. Това ще е така, защото ще предполагаме, че миналото и бъдещето са равноправни и се предсказват еднакво трудно.

Причините поради които няма да се интересуваме от началното състояние са много. Начално състояние може въобще да нямаме, защото миналото може да е безкрайно. Кое е началното състояние на човека? Дали това е момента на раждането или момента на зачеването? В модела на света, който използваме има моменти, които са се случили преди нашето раждане. Тоест, в нашия модел нашето раждане не е абсолютно начало. Има един модел на света, в който има абсолютно начало и това е теорията на Големия взрив. Дали тази теория е вярна? Ние няма да си задаваме въпроса вярна ли е една теория, а дали тази теория ни върши работа (дали описва света). Има много различни теории, които описват света. За всяка от тези теории ще считаме, че е вярна, стига да не може да се направи експеримент, който да противоречи на теорията.

Тоест, при RL ще се питаме какъв е света и къде сме в момента. Обикновено си задаваме въпроса „къде съм“, а не къде съм бил, когато съм се родил. Тоест, ще търсим текущото състояние, а не началното.

2.4.6 Какво е история

Какво е живот? Това е цялата история, тоест история, която не може да бъде продължена, защото е безкрайна или защото няма следваща стъпка – например защото историята е стигнала до внезапна смърт или защото сме изключили агента (естествена смърт).

Какво е история? **Дефиниция:** Съкратена история ще наричаме редицата от действия и наблюдения от началния момент до текущия, заедно с последното коректно действие.

$$a_1, v_1, a_2, v_2, \dots, a_t, v_t, a_{t+1}$$

Дефиниция: История ще бъде когато добавим и некоректните ходове, които сме пробвали и за които знаем, че са некоректни.

$$bad_0, a_1, v_1, bad_1, a_2, v_2, \dots, bad_{t-1}, a_t, v_t, bad_t, a_{t+1}$$

В тази редица *a* е действие, а *bad* е множество от действия. Нека отбележим, че следващата стъпка започва с поредния коректен ход. Множеството *bad* може да се приеме като част от наблюденията на предишната стъпка.

Дефиниция: Пълна история ще бъде когато множествата на некоректните ходове, които сме пробвали ги заменим с множествата на всичките некоректни ходове.

$$full_0, a_1, v_1, full_1, a_2, v_2, \dots, full_{t-1}, a_t, v_t, full_t, a_{t+1}$$

Каква е разликата между история и пълна история? Първото е това, което сме видели по пътя, а второто е това което бихме могли да видим, ако гледахме по-внимателно. Имаме $bad \subseteq full$. Тоест, ходовете, които сме пробвали и за които знаем, че са некоректни са част от всички некоректни ходове. Бихме могли да пробваме още ходове, така че да стане $bad = full$, но така може без да искаме да изиграем някой коректен ход и историята да стане съвсем различна.

Дефиниция: Локална история с дължина *k* ще наричаме последните *k* стъпки на някоя история. Тоест, края на някаква история.

Дефиниция: Приблизителна история. Това ще бъде някакво непълно описание на историята. Предполага се, че историята е твърде дълга и ще е трудно да я помним цялата. Затова ние помним само края на историята или помним само някакви по-важни събития (кога са се случили) или помним някаква статистика за историята. Много често на нас цялата история не ни е нужна и приблизителната история ни е достатъчна, за да направим модел на света и да планираме бъдещите си действия.

Нека отбележим, че за агента не е важно през кои състояния е преминал, а какво е видял. Макар живота да отговаря на път в ориентирания граф (модела на света), на два различни пътя може да отговарят на един и същи живот.

2.4.7 Целта на агента

Каква е целта на агента? Целта е по-добър живот.

Кой живот е по-добър? Трябва ни една релация, която да сравнява животите. Тази релация трябва да е квази-ред (preorder or quasiorder). Тоест тя трябва да е рефлексивна и транзитивна (reflexive and transitive). Ако добавим и антисиметричност (antisymmetry) ще получим частична наредба (partial order). Антисиметричността не ни е нужна, защото няма да е проблем два различни живота да са еднакво успешни.

Бихме искали тази релация да притежава някаква монотонност. Нека разделим живота на две половини и нека вземем два живота. Ако първата половина на първия живот е по-добра от първата половина на втория и същото за вторите половини, то бихме искали първият живот да е по-добър от втория. Обратното би било доста странно.

Подобна релация можем да получим на базата на събирането на награди (rewards) и наказания (regrets). В тази статия няма да казваме награди и наказания, а ще казваме оценки. Наградите са положителни оценки, а наказанията са отрицателни оценки. Обаче има оценки, за които не можем да кажем дали са положителни или отрицателни. Всичко е

относително. Например ако получите тройка в училище, дали това е положителна или отрицателна оценка. За едни е положителна, а за други е отрицателна.

Така, събирайки оценките можем да получим едно число, което да е оценка на целия живот. В [33] това число се получава точно като сумата от оценките, но там дължината на живота е фиксирана и сумата е еквивалентна на средното аритметично. В [34] това число се получава като средното аритметично от оценките. В по-късни статии се използва discount factor, което е грешка. По-надолу ще обсъдим защо това е грешка.

В повечето статии се предполага, че на всяка стъпка агента получава оценка. Това е така в [33, 34] и при много други статии. Подобно предположение може да се приеме като прийом за опростяване на изложението, но по принцип идеята да получаваме оценка на всяка стъпка е нелогична и неестествена. Например, в училище не ни изпитват всеки ден.

Можем да приемем, че в случаите, когато нямаме оценка, оценката е нула, но тогава всичките тези нули съществено ще променят средното аритметично. Това би променило и стратегията ни. Например, ако играем шах и ремито е нула, тогава ще е все едно дали сме постигнали реми или играта продължава. Ще има и голямо значение колко дълга е партията. Тогава, ако средната ни оценка е отрицателна ще се опитваме да удължим партията максимално. Ако средната оценка е положителна, тогава ще се стремим партиите ни да са колкото е възможно по-къси. Естествено е ние да се опитваме да приключим партията възможно най-бързо, но в този случай ще сме склонни да рискуваме победата само и само за да приключим по-бързо. На шахматните турнири се изчислява средното аритметично от победите, загубите и ремитата без да се отчита колко хода е била средната партия.

Тоест, повечето статии предполагат, че оценката е реално число, а ние ще предполагаме че тя е реално число или константата Undef. Тоест, ще предполагаме, че може да имаме, а може и да нямаме оценка.

2.4.8 Различни критерии

В повечето статии се предполага, че имаме един единствен критерии, по който оценяваме живота. Например при икономическите задачи критерият са парите и най-добра е тази стратегия, при която печалбата е най-голяма. Нека да предположим, че имаме два критерия. Например искаме най-добра печалба, но без да влизаме в затвора.

Ще разглеждаме цели, които се определят от повече от един критерии. Например, ако имаме двама души и единия има повече пари, а другият има повече деца, то кой е по-успешен? Ако двата критерия са равноправни, то за да бъде един живот по-добър от друг, то той трябва да е по-добър и по двата критерия. Така релацията „по-добър живот“ става нелинейна, но ние казахме, че искаме тази релация да е квази-ред и не е задължително тя да е линейна.

Може да имаме два критерия и единия да има приоритет пред другия. Например, ако пишем програма за кола, която сама ще се движи, то целта ще е по-малко закъснения и по-малко убити хора. Можем да подходим като оценим закъсненията и човешкия живот с пари и да търсим стратегията с минимален разход. Тоест, може да сведем двата критерия към един. Когато ние караме кола, точно така постъпваме. Оценяваме стойността на

закъснението и когато бързаме увеличаваме риска. Тоест, склони сме да поемем по-голям риск да убием някого, когато закъсняваме. Въпреки, че така постъпват хората шофьори, вашата програма не може да постъпи по същия начин. Ако оцените човешкия живот на някаква сума пари, то ще ви обвинят в безчовечност. Затова трябва да дадете приоритет на втория критерии и да сравнявате две стратегии на базата на броя на жертвите. При равен брой жертви, тогава по-добрата стратегия ще е тази с по-малките закъснения. Тоест, ние ще оценим човешкия живот като безкрайно по-ценен от закъснението. По този начин получаваме линейна наредба, но с два критерия.

Тоест, ако имаме *n* критерия, то оценката трябва да е *n*-мерен вектор. Всяка от координатите трябва да има стойност реално число или константата Undef. Накрая дали живота е по-добър се определя, като се изчисли средното аритметично на всяка от координатите и така получения вектор описва резултата от целия живот.

В тази статия, за да опростим изложението ще предполагаме, че критерия е само един. Все пак, ще отбележим, че Sutton в [32] не е прав като изказва “the reward hypothesis”: all goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a single externally received number (reward).

2.4.9 Discount factor

Казахме, че използването на discount factor при дефиницията на MDP и на RL е грешка. Това е често срещаната грешка когато бъркаме въпросите „какво“ и „как“. Какво искаме да направим и как ще го направим? Например, къде искаме да отидем на почивка? Искаме да отидем в Хонолулу. После си казваме, че Хонолулу е твърде далеч и решаваме, че искаме да отидем някъде по-наблизо. Това е грешка, защото бъркаме въпросите „какво“ и „как“. Какво искаме и как ще го постигнем. Ние искаме да отидем в Хонолулу, а това че ще отидем някъде по-наблизо не променя това къде искаме да отидем.

Discount factor произлиза от една съвсем естествена стратегия, която ни казва, че по-близките награди са по-важни от по-далечните. Дори има народна поговорка, която изразява тази стратегия: „Не изпускай питомното, за да гониш дивото!“.

Discount factor е свързан със стратегията, а не със смисъла на живота. От кой момент нататък да приложим discount factor? Това е текущия момент, но текущия момент не е фиксиран, а се променя. Ако приложим discount factor от началния момент, то началото на живота би било много по-важно от края му, което не е логично.

Каква е била стратегията на Наполеон непосредствено преди битката при Ватерло. Стратегията е била: „Да спечелим битката при Ватерло, а после ще видим.“ Тоест, непосредствено преди битката, тя е била най-важното нещо за него, но ако оценяваме целия живот на Наполеон, то тази битка няма да има чак толкова голяма тежест.

Тоест, да се използва discount factor, за да се определи кой живот е по-добър е грешка. Най-малкото, защото не се знае кой да е момента, от който да приложим този коефициент и какъв да е размера му. Например когато сме по-неуверени ние бихме избрали по-малък discount factor и бихме се стремили към най-близките награди. Колкото по уверени ставаме, толкова по-надалеч в бъдещето ще гледаме и ще сме по-склонни да пуснем

питомното, за да гоним дивото. Тоест, колкото сме по-уверени, токова по-близък до единицата ще е коефициента на обезценка, който ще изберем.

Хубавото на discount factor е, че когато го използваме можем да оценим безкрайния живот. Получаваме едно число, което е сумата на една геометрична прогресия. Как да сравним дава безкрайни живота и да кажем кой от тях е по-добър? Това ще стане по следния начин:

$$Live1 \geq Live2 \Leftrightarrow \exists n \forall (k \geq n)\ begin(Live1, k) \geq begin(Live2, k)$$

Тук *begin(Live, k)* е началото на *Live*, което е с дължина *k*. Тук релацията $\geq$ е нашата квази-ред релация „по-добър“, която ни казва кой живот е по-добър. Предполагаме, че сме дефинирали тази релация за крайните животи и я продължаваме и за безкрайните. Горната формула може да се използва и за сравнение между краен и безкраен живот, ако приемем че когато *k* е по-голямо от дължината на живота, тогава *begin(Live, k)= Live*.

2.4.10 Какъв е модела?

Казахме, че модела на света ще бъде ориентиран граф. Изникват няколко въпроса. Първо дали модела на света да е детерминиран ли недетерминиран. Второ, ако сме предпочели недетерминираните графи, то каква да бъде случайността. В [5] обсъдихме това, че имаме два вида случайности. Първия е, когато знаем точната вероятност за избора на всяка от стрелките (MDP). Втория е, когато знаем кои са възможните стрелки, но не знаем с каква вероятност всяка от тези стрелки може да бъде избрана. На фигура 5 е изобразен втория случай, защото не са дадени вероятности по стрелките с еднакъв цвят. В [5] се разглежда и комбинацията от тези две вероятности, когато не знаем точната вероятност, но имаме интервал и знаем, че вероятността е в този интервал.

В [5] доказахме, че при RL детерминираният модел и моделите с различните видове вероятност ни дават еквивалентни дефиниции. Тоест, не можем да познаем дали света е детерминиран или недетерминиран. Разбира се, това е при условие, че живеем в света само един живот. Ако живеем в света два живота и ако двата пъти правим едно и също, то в детерминирания свят ще се получи същия живот, докато в недетерминирания, почти сигурно, ще се получи различен живот.

Важно предположение е, че при RL живеем само един живот и имаме само една история, на базата, на която трябва да правим изводи. Ако допуснем, че сме живели няколко пъти в един и същи свят и че имаме няколко истории на базата на които да трупаме опит, то се получава съвсем различна задача, при която детерминираният и недетерминираният модел не са еквивалентни.

В тази статия ще започнем с детерминиран модел, който ще наречем „съвършен модел“. След това ще дефинираме недетерминиран модел, който ще наречем „модел със случайност“. После ще заменим действията със събития и ще получим event-driven модел.

Къде е оценката? В тази статия оценката ще е част от наблюдението. В повечето статии това не е така. Обикновено имаме две различни наблюдения, първото от които се казва наблюдение и то определя в кое състояние сме (Full Observability). Второто наблюдение се казва оценка. Първото наблюдение е етикет към състоянието, а второто е етикет към стрелката. В тази статия ще имаме едно наблюдение и оценката ще е част от него.

2.4.11 Некоректни ходове

Дали модела да допуска некоректни ходове или да предполагаме, че всеки ход е коректен? За всеки модел с некоректни ходове можем да направим еквивалентен на него модел, в който всички ходове да са коректни. В [6] направихме нещо подобно като построихме тоталния модел. Тоест, добавихме едно наблюдение *bad*, което да се получава винаги след некоректен ход.

Нека за агента да имаме две програми – тотално AI и частично (partial) AI. Първата програма да изисква всички ходове да са коректни, а втората да допуска некоректни ходове. Ако дадем на тоталното AI да се бори със света, който е представен като тотален модел, то така бихме направили задачата на агента доста по-трудна. Той ще търси зависимости в реда на пробваните некоректни ходове. Няма да намери такава зависимост, защото нея я няма, но няма да престава да я търси. Ще пробва да повтори два пъти един и същи некоректен ход. Накрая ще се научи, че от това нищо ново не се случва и ще спре тези опити. Този агент няма да знае, че пробвайки един некоректен ход той не променя нищо (освен, че разбира че хода е некоректен, но ако това вече го знае, то той не променя нищо). Тези неща също могат да се научат, но винаги когато този агент е изпаднал в безизходица и се чуди какво да прави ще пробва пак разни неща, които е безсмислено да се пробват. Живота на частичното AI в частичния модел ще е много по-лек, защото той по рождение ще знае някои неща за некоректните ходове и няма да има нужда сам да открива и да учи тези неща.

Затова е по-добре да допуснем, че имаме некоректни ходове. Това прави света много по-прост и по-лесно разбираем. Спестяваме на агента търсенето на зависимости, които не съществуват и тяхното търсене би било просто загуба на време. Освен това некоректните ходове са много хубав пример на полувидимо събитие (за тях ще стане дума по-надолу). Също така некоректните ходове са хубав пример за тестови състояния (за тях се говори в [5], но ще говорим още за тях в следващата статия).

2.4.12 Reinforcement Learning

Нека да кажем, коя ще е формалната дефиниция на Reinforcement Learning, която ще използваме. Ще кажем какво ни е дадено и какво търсим. Дадени са ни:

A – множество от възможните действия на агента.
V – множество от възможните наблюдения.
$Reward : V \to \mathbb{R} \cup \{Undef\}$ (това е функция, която за всяко наблюдение ни дава оценка или *Undef*)
H – история или приблизителна история на случилото се до момента t.
Дадено ни е още, че съществува съвършен модел M на света и този модел се намира в някакво текущо състояние s_t.

Какво търсим? Трябва да отговорим на три въпроса:
1. Какъв е света? (трябва да намерим модела на света)
2. Къде съм? (трябва да определим текущото състояние на света)
3. Какво да правя? (трябва да решим какво ще е нашето следващо действие, както и по-нататъшните ни действия, като целта е да постигнем максимална стойност на средноаритметичното на оценките)

Забележка: В тази статия се занимаваме основно с първите два въпроса, а оценката е свързана само с отговора на третия въпрос. Затова по-надолу няма да става дума за оценки.

2.4.13 Съвършен модел

Нека да кажем какво представлява съвършеният модел на света:
S – множеството от вътрешните състояния на света.
s_t – текущото състояние на света.
$G = <S, R>$ – тотален и детерминиран ориентиран граф.
$R \subseteq S \times A \times S$
$View: S \to V$
$Incorrect: S \to P(A)$

На всяко ребро (стрелка) има етикет, който ни казва кое е действието, което трябва да извършим, за да преминем по тази стрелка. Стрелките и техните етикети се определят от релацията R. На всеки връх (състояние) съответстват два етикета, които казват какво виждаме и кои ходове са некоректни в това състояние. Тоест, етикетите са наблюдението в това състояние и множеството от невъзможни действия. Тези етикети на състоянието се определят от функциите *View* и *Incorrect*.

Ще предполагаме, че ориентирания граф G е тотален и детерминиран. Тоест, от всяко състояние, по всяко действие има стрелка, която излиза от това състояние и тази стрелка е единствена. Тук стрелките, които отговарят на некоректни ходове са в известна степен излишни, защото никога няма да ги използваме, но предполагаме, че и те съществуват, защото по-долу ще разгледаме друг модел на света, в който множеството некоректните ходове няма да е постоянно, а ще се променя. Тоест, при едно и също състояние в различни моменти ще можем да имаме различни некоректни ходове.

Нека отбележим, че когато имаме съвършен модел на света и когато знаем кое е текущото състояние на света, тогава бъдещето е напълно определено. Тоест, ако знаем кои ще бъдат действията на агента, мажем да кажем точно какво ще се случи. За разлика от бъдещето, миналото не е напълно определено, защото може да имаме много различни начални състояния, които след историята H да водят до това текущо състояние. Дори и пълната история не е определена, защото може различни начални състояния да имат различна пълна история. Разбира се, различните пълни истории трябва да са съгласувани с историята H (тоест трябва да удовлетворяват условието $bad \subseteq full$). Ще предполагаме, че имаме поне едно възможно начално състояние, което след историята H да води до текущото състояние, защото предполагаме, че тази история все пак се е случила в този модел.

Ако освен текущото знаем и началното състояние, тогава от съвършения модел можем да получим така наречения съкратен модел. Това ще е модела, в който сме изхвърлили всички състояния и стрелки през които не сме минали прочитайки историята H. В съкратения модел можем да направим статистика като преброим колко пъти сме минали по всяка една от стрелките. Чрез тази статистика ние ще можем да предскажем миналото, бъдещето и поведението на агента. Когато знаем в определено състояние колко пъти агента е избрал едно действие и колко пъти друго действие, ние може да предвидим какво би направил агента, ако пак попадне в това състояние. Тоест, ние ще се опитвам да предскажем

собственото си поведение, защото както казахме, ще разглеждаме света и агента като единна система.

Съкратения модел може да не е съвършен. Може да попаднем на стрелка, по която никога не сме минавали (тоест, такава която я няма в съкратения модел). Въпреки всичко, съкратения модел е това, което бихме могли да знаем, защото ако по една стрелка никога не сме минавали няма откъде да знаем накъде води тя.

2.4.14 Случайни величини

Нашето желание да намерим съвършен модел на света е една твърде амбициозна задача, която би била възможна само, ако света е изключително прост. Нас, естествено, ни интересуват по-сложни светове и затова е разумно да предположим, че света е толкова сложен, че ние никога няма да можем напълно да го разберем.

Нека допуснем, че има предел на нашето познание и има неща, които ние никога няма да можем да предскажем. Тези неща ние ще наречем случайни величини (random variables). Ще предположим, че това са зависимости, които са толкова сложни, че няма как да ги отгатнем. Например, ако хвърля зар, то предполагам, че ще се падне число от 1 до 6 с вероятност 1/6. Ако имах съвършен модел, то бих могъл да кажа точно колко ще се падне. Има такъв съвършен модел и след като хвърля зара ще знам кой е той, но на мен ми трябва да го знам още преди да съм хвърлил зара. Няма как преди да хвърля зара да знам какво ще се падне.

Трябва да отбележа, че ако аз бях много много умен, щях да мога да кажа какво ще се падне. Щом не мога да кажа, значи или не съм достатъчно умен, или нямам достатъчно информация на базата, на която да кажа какъв ще е зара.

Допускам, че не съм достатъчно умен и най-доброто предположение, което мога да направя е число от 1 до 6 с вероятност 1/6. Дори и това да е най-доброто предположение, което аз мога да направя, това не значи че аз няма да търся и по-добро. Може да предположа, че зара е крив и дава 6 с по-голяма вероятност от 1/6 или че ако духна върху зара преди да го хвърля, то вероятността да се падне шестица се увеличава. Последното предположение се приема за суеверие (тоест за невярно), но тук ще търсим статистическа връзка между събития и ако статистиката показва, че духването върху зара ни помага, то това се приема за доказателство, дори да се окаже, че това се е случило случайно.

Какво е случайната величина? Да предположим, че аз сега ще ви кажа черно или бяло. Какво очаквате да чуете – черно или бяло? Предполагам, че идея си нямате какво ще ви кажа. Тест, вие очаквате да чуете бяло с вероятност в интервала [0, 1].

Ако знаете, че аз имам един зар с една бяла страна и пет черни. Знаете, че аз ще хвърля този зар и така ще определя дали да ви кажа черно или бяло. Тогава вие ще очаквате бяло с вероятност 1/6. Да предположим, че аз имам втори зар с две бели и четири черни стени. Ще хвърля един от тези два зара и така ще определя дали да ви кажа черно или бяло. Сега предполагате да чуете бяло с вероятност в интервала [1/6, 1/3]. Да предположим, че знаете че е по-вероятно да хвърля първия зар, отколкото втория. Сега очаквате вероятност в интервала [1/6, 1/4).

Да предположим, че аз казвам веднъж бяло, веднъж черно. Тоест вие ще очаквате да чуете обратното на това, което съм казал последния път. Това обаче е зависимост с памет. Аз бих искал да опиша случайните величини без параметри. Случайната величина с параметри е функция, която при дадени параметри ни връща случайна величина без параметри. По-надолу ще опишем един оракул α, който зависи от миналото, бъдещето и от текущото състояние. Тоест, миналото е един от възможните параметри.

Да предположим, че аз когато съм в добро настроение казвам бяло или поне вероятността да кажа бяло се увеличава. Това добавя към модела още един параметър, който ще бъде моето настроение. Тоест, можете да се опитвате да предскажете какво е моето настроение или ако моето настроение е напълно непредвидимо, тогава имате случайна величина, която не използва моето настроение като параметър. Тази величина ще зависи от моето настроение, но щом то е непредвидимо, то няма да е параметър. Ще бъде скрит параметър или параметър, който няма да се опитваме да предскажем.

Може да предположите, че това което ще ви кажа е възможно най-лошото. Подобно предположение се използва често. Например в програмите играещи шах в алгоритъма Min-Max се предполага, че противника ще изиграе възможно най-нежелания от нас ход. Тоест, предполагаме, че имаме противник, който ни мисли лошото. Не винаги предполагаме най-лошото. Понякога предполагаме най-доброто. Например, ако се прибирате вкъщи след дълъг път, какво очаквате да има за вечеря? Очаквате най-доброто, защото там има някой, който ви обича и който ви е приготвил това, което вие най-много обичате.

Да предположим, че очаквате най-доброто. Ако бялото по някаква причина е по-добро от черното, то вие бихте очаквали да ви кажа „бяло“. Може да не знаете кое е по-доброто и това да се разбере в последствие. Тогава вашето очакване зависи от бъдещето. Тоест, зависи от друг параметър.

2.4.15 Случайна величина без параметри

Въпросът е какво е случайна величина без параметри (RVWP). Повечето автори предполагат, че RVWP си има точно определена вероятност (или разпределение, ако възможните стойности са повече от две). Такова е и предположението при MDP. Там случайно се избира следващото състояние, но тази случайност не е съвсем случайна, защото се предполага, че вероятността да се избере определено състояние е точно определена.

Тук ще предполагаме, че имаме две нива на неопределеност. Първото ниво е, когато не знаем какво ще се случи, но знаем точната вероятност p, с която това ще се случи. Второто ниво на неопределеност е, когато не знаем дори и вероятността p. Може такава вероятност въобще да няма, а може да има, но ние да няма откъде да я знаем.

Когато предвиждаме едно случайно събитие, да кажем че то си има някаква вероятност p, с която ще се случи. Ако сме достатъчно умни и имаме нужните данни, ние ще предскажем тази вероятност. До, но събитието има и конкретна стойност, която ще се случи и ако ние сме много много умни ще можем да познаем дори и тази конкретна стойност. С първото ниво на неопределеност ние приемаме, че не можем да познаем какъв ще е резултата, а с второто ниво на неопределеност приемаме, че дори и вероятността не можем да познаем.

Може ли едно събитие въобще да няма точна вероятност? Ако има някаква вероятност p и ако наблюдаваме събитието безкрайно дълго, то статистически получената вероятност трябва да клони към p (закон за големите числа). Ако наблюдаваме събитието безкрайно дълго статистиката може да не клони към определена стойност (ще имаме limit inferior и limit superior, но те може да не са равни).

Много лесно е да си представим ситуация, в която не знаем точната вероятност, а само знаем, че тя е в интервала $[0, 1]$ или $[a, b]$. Много трудно е да конструираме физически експеримент, който ни дава такава вероятност и тази вероятност да е най-доброто предположение, което можем да направим. Например нека вземе една редица от нули и единици, за която limit inferior и limit superior от средното аритметично са съответно 0 и 1. Тази редица вероятно ще се движи зигзагообразно и когато вероятността за последните 100 е голяма ще очакваме следващото число да е единица също с голяма вероятност. Тоест, тази редица ще има вероятност в интервала $[0, 1]$, но това няма да е най-доброто описание на случайната величина, защото ще имаме и по-добро, което ще получим от последните 100 резултата.

Ако искаме да направим физически експеримент, който ни дава точно определена вероятност p, то това е лесно. Просто конструираме зар, който ни дава вероятност p и готово. Не само, че това ще е случайната величина описваща този експеримент, но това още е най-доброто възможно описание на експеримента.

Как да конструираме физически експеримент, който ни дава вероятност в интервала $[0, 1]$ и да няма по-добро описание на този експеримент? Ще предположим, че срещу агента стои едно същество, което казва 0 или 1, но не случайно, а така че да обърка агента и той да не може да познае дали следващото ще е нула или едно, и той дори да не може да каже с каква вероятност очаква следващото да е едно. Ще предполагаме, че съществото е много по-умно от агента и действително успява да го обърка.

Това, което направихме, не е коректна математическа дефиниция, защото колкото и да си умен винаги има някой, който е по-умен от теб. Тоест, съществото зависи от агента. Ако съществото може да прочете мислите на агента и да познае, че той очаква единица в интервала $[a, b] \subset [0, 1]$, тогава може да го обърка като играе така, че да излезе, че вероятността е извън този интервал. Да прочете мислите звучи екзотично, но ако агента е алгоритъм (програма), тогава съществото може да изпълни тази програма и да разбере какво агента очаква да се случи.

Така направихме модел на събитие, чиято вероятност е в интервала $[0, 1]$. Ако искаме да направим събитие, чиято вероятност да е в интервала $[a, b]$, то ще конструираме два зара с вероятности a и b и ще дадем на съществото тези зарове. То ще избира, кой от заровете да хвърли, като целта му пак ще е да обърка агента.

Другите автори предполагат, че случайната величина без параметри има точна вероятност (разпределение), а ние ще предполагаме, че има RVWP точно определен интервал на вероятността (или разпределение от интервали). Разбира се, тези интервали от вероятности ще трябва да изпълняват неравенствата описани в [5]. Ще предполагаме, че RVWP не зависи нито от миналото нито от бъдещото. Както е хвърлянето на зар, то не зависи нито от хвърлянията преди, нито от хвърлянията след това.

2.4.16 Модел със случайност

Предположихме, че има случайност, която не може да бъде предсказана. Сега ще направим втори модел на света, който включва тази случайност. Първо, ориентираният граф вече няма да е детерминиран (в дефиницията на MDP графа също не е детерминиран). Второ това, което наблюдаваме в състоянието s, вече няма да е константа функция, а ще е случайна величина (при POMDP наблюдението също е случайна величина). Трето, множеството на некоректните ходове в състоянието s, също ще бъде случайна величина. Тези три случайни неща ще се определят от три оракула (α, β и χ). Оракулите ще са случайни величини с параметри, а при конкретна стойност на параметрите те ще са случайни величини без параметри (RVWP).

Оракула α казва при дадено състояние s_t и дадено действие a_{t+1} какво ще е следващото състояние. Това до голяма степен е определено от графа G. Оракула α е задължен да избере една от възможните стрелки. Така че избор за оракула има само когато прехода е недетерминиран.

Оракула β казва при дадено състояние s_{t+1} какво ще е наблюдението в момента $t+1$. Тук сме написали $t+1$ вместо t, защото трите оракула използват едно и също $Past$ (a_{t+1} е края на това $Past$.) Първо оракула α казва кое ще е следващото състояние и после оракулите β и χ казват какво ще се види в това състояние и кои ще са некоректните ходове.

Оракула χ казва дали събитието e се е случило в момента t. Тук ще използваме χ за събития от типа дали определен ход е некоректен. По-надолу ще има и други събития, които също ще се определят от χ.

Ето дефиницията на модел със случайност:

S – множество от вътрешните състояния на света.
s_t – текущото състояние на света.
$G = <S, R>$ – тотален ориентиран граф (недетерминиран).
$R \subseteq S \times A \times S$
$\alpha(Past, s_t, a_{t+1}, Future) \rightarrow s_{t+1}$
$\beta(Past, s_{t+1}) \rightarrow v_{t+1}$
$\chi(Past, s_{t+1}, e) \rightarrow \{true, false\}$

Идеята на оракулите е, че агента не знае в кое състояние е света, какво ще види в това състояние и кои ходове ще са коректни. Агента не знае, но света знае всичко. Тоест, света разполага с тези оракули и може да каже какво ще се случи. Да предположим, че света има на разположение един съвършен модел на света. Такъв модел би определил напълно трите оракула.

Забележка: Тук за всяко събитие e оракула χ дава отделна случайна величина. Дефиницията остава впечатлението, че това са независими случайни величини, а това може и да не е така. Две събития може да са свързани. Например събитието a може да се случва само когато се е случило събитието b. Друг пример е когато a и b никога не се случват едновременно. Наблюдението също може да има връзка със събитията. Може би е

добре дефиницията да се усложни и оракулите β и χ да се обединят в един по-сложен оракул, но това няма да го правим.

Оракулите зависят от миналото. Ще предполагаме, че *Past* е пълната история. (Предполагаме, че използваме пълната, а не обикновената история, защото оракулите зависят от това какво се е случило, а не от това което агента е видял. Агента знае само обикновената история, а света знае пълната история. Предполагаме, че агента ще се научи да познава кои ходове са коректни и той на практика също ще знае пълната история.)

Ще предполагаме, че оракула α зависи и от бъдещето. Това звучи доста екзотично, но ако съобразим кое е причината и кое е следствието, това че оракула вижда бъдещето не е чак толкова странно. Да вземем примера с пощальона и скитника. Странно би било, ако кажем, че оракула е погледнал в бъдещето, видял е че в бъдеще ще се появи доказателство за това, че пощальона е убиеца и затова оракула е решил убийството да бъде извършено от пощальона. По-логично би звучало, ако кажем, че убийството е извършено от пощальона и затова в бъдещето се е появило доказателство, че той е убиеца.

Това, че оракула зависи от бъдещето ще повлияе на отпечатъка на модела. Видяхте във фигура 2, че това че оракула избира стрелките по този начин осигурява това, че на следващата стъпка от състоянието 1 не излиза стрелка по събитието *b*. Тоест, това събитие никога не се е случило, когато сме били в състояние 1. Ако оракула не се държеше по този начин, то от състоянието 1 щеше да излиза стрелка по *b*.

Ще предполагаме, че на оракула α е дадено цялото бъдеще (до края на живота или до безкрайност). Въпреки това, ние нямаме цялото бъдеще, а само бъдещето до текущия момент *t*. В условието на задачата ни е дадена историята до момента *t*. Ако искаме да разберем какво ще каже оракула α в момента *t-k*, тогава ще разделим историята на минало от 0 до *t-k* и бъдеще от *t-k* до *t*. Трябва да отбележим, че решението на оракула зависи обикновено само от близкото бъдеще. В примера с убийството, ако не се докаже кой е убиеца скоро след убийството, вероятно въобще няма да се докаже. Веднъж докаже ли се нещо, после няма как да се докаже обратното.

Ако използваме това, че живота се живее само веднъж може да определим оракула β като функция. Разбира се, тази функция ще е дефинирана само за историите, които са се случили в този живот. За останалите истории ще трябва да си измислим стойностите на тази функция (да ги дефинираме както и да е). Ако знаем пълната история на живота, то ще можем да определим и оракула χ. Ако имаме само обикновената история, то този оракул ще можем да го определим само частично. Оракула α можем да го определим по произволен начин. Можем да вземем произволен граф *G* и да дефинираме оракула α по този граф. Това би определило α винаги освен в случаите на недетерминирано разклонение (тогава ще можем да определим оракула както си поискаме). Пример за произволен граф е, когато имаме само едно вътрешно състояние. Дори и този граф е един възможен модел на света (фигура 1 е такъв модел).

Трябва да отбележим, че ако имаме един и същи ориентиран граф, но различен оракул α, то модела вероятно ще е различен, защото отпечатъка на модела може да е различен. Ние няма да търсим различни оракули α, а ще търсим различни отпечатъци. Ако два оракула дават еднакъв отпечатък на модела ще ги приемаме за еднакви. Обратното, ако ни е даден

отпечатъка по него може да построим оракул, който индуцира този отпечатък (ако въобще има такъв оракул, разбира се). Това ще стане по следния начин. От възможните избори на оракула ще махнем тези, които са невъзможни при конкретния отпечатък и при бъдещето, което ни е дадено. Ако са останали още възможни избори, ще изберем един от тях. Ще изберем този един с вероятност съответстваща на вероятността този избор да е възможен при този отпечатък и при това бъдеще.

2.4.17 Отпечатък

Нашата идея е, че ориентирания граф *G* по някакъв начин описва света. Тук се оказа, че произволен граф *G* може да бъде модел на света! Истината е, че всеки граф може да е модел, но далеч не всеки ще е адекватен модел на света. Когато сме в някое от състоянията на графа, очакваме нещо да се случва. Очакваме някакви събития никога да не се случват в това състояние или обратното, задължително да се случват. Може някакво събитие да се случва с вероятност много по-голяма или много по-малка от средната вероятност за това събитие. Ако в никое от състоянията нищо не се случва, то модела ще е напълно неадекватен.

Това, което се случва когато се движим по стрелките на модела, ще наречем отпечатък на модела. При съвършеният модел имаме много ясен отпечатък. Там всяко състояние има точно определено наблюдение и точно определено множество от некоректните ходове. Ако вземем съкратения модел на съвършения, тогава някои стрелки ще липсват. Липсващата стрелка ще означава, че в определено състояние някакво събитие никога не се е случило (някакъв ход не е игран). Това също ще е отпечатък. Въпросът е как да го тълкуваме. Дали да предположим, че в това състояние това никога няма да се случи или че може да се случи, но с много малка вероятност.

Едно събитие може да зависи не само от това какво непосредствено сте видял и направил. То може да зависи и от по-далечната история. Например, какво сте видял на предишната стъпка.

Как ще търсим отпечатък? Ще го търсим чрез статистиката. Ще вземем съкратения модел (тоест, ще махнем всички стрелки и състояния, които до момента не са използвани). Ще преброим за всяка стрелка колко пъти е използвана. За всяко състояние и за всяко събитие ще преброим колко пъти то се е случило в това състояние. Разбира се, не можем да преброим за всяко събитие, защото събитията са безкрайно много. Ще броим само за някои по-важни събития. На базата на тази статистика ние можем да намерим отклонения от средното очаквано, които ще бъдат търсения отпечатък.

Нека да отбележим, че тази статистика може да я направи света, но за агента това би било много по-трудно. Света знае точно в кое състояние се намира и може да брои стрелките, докато агента може само да гадае. Дори и при съвършения модел агента може да не знае в кое състояние е бил (може да не знае кое е последното състояние, а дори и да знае, това не определя еднозначно предишните състояния).

Понякога агента не знае в кое състояние се намира в момента и това го научава в последствие. Това няма да е проблем, защото статистиката може да се събере със закъснение. Проблем е, когато агента никога не разбира точно в кое състояние е бил. Тогава проблема за събирането на статистиката е доста сложен.

2.4.18 Пълен модел

Да допуснем, че имаме един модел, в който оракулите α, β и χ не завият от миналото. Този модел ще наречем пълен. Тоест, това е модел, който не може да се подобри. Всичко за миналото, което трябва да се запомни, вече сме го запомнили в текущото състояние. Модела би могъл да се подобри, ако едно състояние го разделим на две и ако при определена история попадаме в едното или в другото с различна вероятност. Тоест, двете състояния да са различими спрямо миналото. При пълен модел двете състояния няма да са различими спрямо бъдещето, защото оракулите не завият от миналото. Тоест, това разделяне на състоянието на две става безпредметно.

Може да имаме пълен модел, който да бъде само с едно единствено състояние. В този случай света е ужасен и за бъдещето няма никакво значение какво се е случвало преди. В такъв свят няма никакво значение кое действие ще изберем. Разбира се, такъв свят е прекалено елементарен. Предполагаме, че световете, които ни интересуват, са много по-сложни и при тях намирането на съвършен или на пълен модел е на практика невъзможно.

Ние няма да се опитваме да разберем света напълно (да намерим съвършен модел) или да го разберем до нивото на някаква неразрешима случайност (да намерим пълен модел). Ние ще търсим много различни модели, които описват различни черти на света и обекти в света. Ще направим декартовото произведение на всичките тези модели и ще получим модел, който вероятно пак няма да е пълен, но ще описва света доста добре.

Това, че света е пълен се нарича „свойството на Марков“. В RL обикновено се предполага, че света има пълен модел. Тук ние предположихме нещо повече. Предположихме, че света има съвършен модел.

Ние няма да търсим пълни модели, а адекватни модели. Тоест, ще търсим модели, при които има някакъв отпечатък. Търсенето на пълен модел е една излишно амбициозна задача, която при по-сложните светове е абсолютно нерешима.

2.4.19 Видими събития

Събитие ще бъде някаква булева функция, която във всеки момент от времето е истина или лъжа. Първо ще кажем какво е видимо събитие. Това ще бъде събитие, което се вижда от историята (даже не от цялата история, а от нейния край).

Дефиниция: Видимо събитие ще бъде множество от локални истории. Това събитие ще бъде истина когато историята завършва с някоя локална история от множеството.

Пример за видимо събитие, това е какъв е последния ни ход. Пример за събитие, което не е видимо, това е дали един ход е коректен. Това се вижда от пълната история, но не се вижда от обикновената история. Това събитие ще го наречем полувидимо защото съдържа в себе си (като подмножество) едно видимо събитие. Видимото събитие, което се съдържа в него е следното: „този ход е некоректен и вече сме го пробвали“.

Друг пример за полувидимо събитие е изгрева. Него можете да го видите, но може и да го проспите. Това, че сте го проспали, не означава, че е нямало изгрев. Имало е, просто вие не сте го видели.

Пример за невидимо събитие е: настинах. Вие не виждате кога това събитие се случва, но в последствие може да установите, че то се е случило по различни признаци (като висока температура и други).

Досега множеството от събития, което използвахме за да намерим модел на света беше множеството от действията ни. Сега ще обобщим и ще използваме произволно множество от събития.

2.4.20 Event-Driven модел

Сега ще заменим действията със събития. Ще вземем едно множество от събития. Предполагаме, че това множество е малко, защото ако събитията са прекалено много модела ще стане прекалено сложен.

В новия модел стрелките вече няма да са по действия, а ще са по събития. При действията не беше възможно две действия да бъдат извършени едновременно. При събитията е напълно възможно две събития да се случат едновременно. Затова оракула α няма да се определя от действие, а от събития, при това, не от едно събитие, а от *events* – множество от събития. Когато *events* е празното множество, тогава никъде не отиваме и α ни връща същото състояние s_t. Когато в *events* има само едно събитие, тогава оракула избира една от възможните стрелки, които са по това събитие. Когато в *events* има две събития, тогава оракула трябва да реши дали да избере едно от двете (т.е. все едно, че едното събитие е затъмнило другото) или да предположи, че двете събития са се случили едно след друго. Тоест, да мине първо по стрелка отбелязана с първото и после да мине по още една стрелка, отбелязана с второто. Освен това оракула трябва да реши кое събитие да е първото и кое второто.

S – множество от вътрешните състояния на света.
s_t – текущото състояние на света.
E – множество от събития.
$G = <S, R>$ – тотален ориентиран граф (недетерминиран).
$R \subseteq S \times E \times S$
$\alpha(Past, s_t, events, Future) \rightarrow s_{t+1}$
$\beta(Past, s_{t+1}) \rightarrow v_{t+1}$
$\chi(Past, s_{t+1}, e) \rightarrow \{true, false\}$

Тук оракула χ определя не само кой ход е некоректен, а определя всички невидими събития от E. За полувидимите събития оракула определя тяхната невидима част. Видимите събития зависят само от *Past* и за тях е ясно как би работил оракула. Затова е излишно оракула да разпознава видими събития. Не искаме оракула да разпознава видими събития, защото пълен модел имаме когато оракулите не зависят от *Past*. За да не променяме дефиницията на пълен модел ще предполагаме че χ разпознава само невидимите събития, а видимите са ясни.

При event-driven модела пак можем да направим съкратения модел от стрелките, които са използвани и да преброим колко пъти са използвани. Пак света е този, който може точно да преброи, а агента може да направи само приблизителна преценка на тези бройки.

На базата на тази статистика може да намерим отпечатъка на модела и да преценим дали този модел е адекватен.

2.4.21 Модел с променливи

Естествено е към модела да добавим и променливи. В едно състояние едно събитие може да се случва известно време и после да престане да се случва. Това дали събитието в момента се случва е удобно да бъде представено като променлива. Тази променлива ще е локана, тоест, ще е свързана с определено състояние. Нищо не пречи да имаме и глобални променливи, които да са свързани с няколко състояния.

Пример за модел с променливи беше даден в [5]. Там имаше много врати (състояния) и всяка врата беше отключена или заключена. Тоест, на всяка врата съответстваше една променлива.

S – множество от вътрешните състояния на света.
Var – множество от променливи.
s_t – текущото състояние на света.
$eval_t$ – текущата оценка на променливите.
E – множество от събития.
$G = <S, R>$ – тотален ориентиран граф (недетерминиран).
$R \subseteq S \times E \times S$
$\alpha(Past, s_t, eval_t, events, Future) \rightarrow <s_{t+1}, eval_{t+1}>$
$\beta(Past, s_{t+1}, eval_{t+1}) \rightarrow v_{t+1}$
$\chi(Past, s_{t+1}, eval_{t+1}, e) \rightarrow \{true, false\}$

Тук от теоретична гледна точка нищо не правим, защото добавяйки променливи ние само увеличаваме броя на състоянията. Това се вижда и по-горе. Навсякъде където имахме състояние, сега има състояние и оценка на променливите.

Макар, че на теория добавянето на променливи нищо не променя, на практика се получава нещо много различно, защото ако имаме десет булеви променливи, то състоянията се увеличават 1024 пъти, което е много съществено увеличение. Ако се опитаме да нарисуваме ориентирания граф с толкова много състояния ще се получи нещо много сложно. В крайна сметка променливите описват отпечатъка (какво се случва в състоянието). За повечето променливи ние няма да имаме никаква идея каква е тяхната стойност.

2.4.22 Декартов модел

Както казахме, ще търсим различни модели, които да описват различни особености на света. Нашият модел на света, модела който сме открили, ще се състои от всичките тези модели. Може да си мислим, че той е декартовото произведение на всички тези модели.

Кое ще е състоянието, в което се намираме? Това ще са текущите състояния на всичките тези модели и текущите оценки на променливите в тези модели. Може да си мислим, че това е наредената енторка (tuple) от всичките тези текущи състояния и текущи оценки на променливи.

В момента вие сте в града, в който живеете. Денят е понеделник. Часът е 10 сутринта. Вие сте сит, защото закусихте добре. Всяко едно от тези четири изречения описва един event-driven модел и текущото състояние на този модел.

В момента вие виждате монитора. Това е обект (event-driven модел) и вие го виждате, тоест не сте в състоянието *outside*.

В момента вратата на вашата стая е отключена, а външната врата е заключена. Тоест, вие имате в главата си модел на сградата и имате идея за две от вратите дали са заключени. Тоест, знаете стойността на две от променливите на този модел.

Ще предполагаме, че никой от моделите, които сме открили не е пълен, защото ако има един пълен, то другите модели ще са излишни. Възможно е декартовото произведение на много непълни модели да даде пълен модел, но ако света е достатъчно сложен, това ще е малко вероятно.

2.4.23 Агенти и обекти

Трябва да правим разлика между агенти и обекти. Това са две различни неща.

В предишни статии [8] говорихме за агенти. Когато някой нещо променя, то този някой е агент и ние трябва да предвидим следващите му действия, да решим дали ни е враг или съюзник, да се опитаме да се разберем с него.

Какво е човека – агент или обект. От една страна той е агент, защото променя света. Може нещо да премести, може нещо да открадне. От друга страна, той е обект, защото ние го разпознаваме. Като го видим, като го чуем по телефона, когато някой спомене името му.

2.4.24 Заключение

В тази статия разгледахме връзката между събитията и тяхното следствие. (Следствието наричаме отпечатъка на event-driven модела.) Въпросът е, кога се случва събитието? Например, кога с случва изгрева? Когато слънцето се показва малко, когато се показва наполовина или когато напълно се показва? Тук избрахме един момент, в който приемаме, че събитието се е случило и това е момента, в който сме разбрали, че то се е случило. Въобще не е задължително, следствията от това събитие да започнат точно в този момент. Например, когато слънцето изгрее става светло веднага, дори става светло още преди да е изгряло. От друга страна, става топло, но не веднага, а много по-късно. Тоест, когато търсим отпечатъка, може да има следствие от събитието или следствие от това, че сме попаднали в определено състояние на модела, но не трябва да очакваме това следствие да се появи веднага. Трябва да допускаме следствието да е малко изместено във времето. Освен това следствието (отпечатъка) няма да е свързано само със състоянията на модела. Може да имаме следствие и от самото събитие. Например при изгрева небето става червено и това е около момента на изгрева. Това не е свързано със състоянието преди или със състоянието след изгрева (тоест, с нощта или с деня).

Тоест, когато събираме статистика трябва да отчитаме колко близо сме до събитието и да търсим отпечатък (особености), както в периодите между събитията, така и около самите събития. Отпечатъка е особеност, която се проявява в определено състояние на модела, но може да се проявява и когато минаваме по някоя от стрелките на модела.

Отпечатъка няма да се състои само от събития. Освен събитията, ще имаме и тестове, ще имаме и обекти. Тестовете са специален тип събития, за които ще кажем по-надолу. Появата на обект също е събитие и това събитие може да ни помогне да дефинираме отпечатъка на един event-driven модел. Тоест, за дефинирането на обект ние може да използваме други обекти. Например, в една стая виждаме котка и това е стаята с котката. Ето как обекта котка ни помага да определим и да запомним една стая. Разбира се, котката е обект, който има свойството да се мести. Може да се наложи да я търсим из цялата къща. Ако котката има свойството единственост, тогава ако сме я намерили в една стая няма да я търсим повече по другите стаи. Може котката да не стои в една стая, но в различните стаи да има различна вероятност да видим котката. Тази различна вероятност също може да бъде отпечатък на модела (в случая модела е къщата).

Видяхме, че каквито и събития да си вземем и какъвто и ориентиран граф да изберем, то това е модел на света. Лошото е, че ако този модел е избран произволно, то той почти сигурно ще се окаже неадекватен. Тоест няма да има никакъв отпечатък или нищо особено (нищо интересно) няма да се случва в състоянията му.

В някои случаи (като фигура 2) модела ще има отпечатък, но този отпечатък ще се дължи само на оракула α и ние няма как да използваме този отпечатък (в случая да предскажем следващото събитие), защото няма да има как да познаем в кое състояние сме.

Това показва, че задачата за откриването на адекватен модел на света е много сложна. За целта трябва да търсим особености. Трябва да наблюдаваме различни събития и да забележим когато някоя особена комбинация от събития се случи. Например, черна шапка и щръкнала коса – това трябва да е колегата Джон.

Малко са събитията, които можем да наблюдаваме постоянно. Повечето събития ние ги наблюдаваме само понякога. Такива събития наричаме тестове [5]. Тестовете са особено важни за откриването на адекватни event-driven модели.

Следващата статия ще бъде посветена на тестовете и ще видим как с тяхна помощ можем ефективно да откриваме event-driven модели.

2.5 Дефиницията на ИИ в термините на мулти-агентните системи

В предишни статии на същия автор [1-19] се разглежда въпросът за дефинирането на ИИ. Там постановката на задачата е същата, като в статиите, посветени на мулти-агентните системи [35-36]. Може да се каже, че формализацията, използвана в [1-19], е частен случай на формализацията, използвана в [35-36]. Частният случай се получава, като се ограничим само до един агент.

Наистина формализацията в [1-19] се получава като частен случай от формализацията на [35-36], но въпреки това статиите [1-19] продължават да бъдат интересни, защото там въпросът, който се разглежда, е друг. Тоест, ако това са две задачи от типа „Дадено е ..., търси се ...“, то даденото в двете задачи е едно и също, но се търсят различни неща. Тоест, това са две различни задачи, като всяка от тях е интересна за себе си. Все пак съществува съществена връзка между тези задачи и тази връзка може да се използва и в двете посоки.

Тоест, може от втората задача да се заимстват идеи и да се обогати първата, а може и обратното.

2.5.1 Постановка на задачата при мулти-агентните системи

Даден ни е един свят, имаме множество от вътрешните състояния на този свят, едно от които е началното състояние на света. Имаме n на брой агенти, които живеят в този свят. Имаме една функция World (това е функцията на света). Тя има два аргумента. Първият аргумент е текущото състояние на света, а вторият аргумент е n-торката от действията на n-те агента. Функцията на света връща новото вътрешно състояние на света, което се получава от текущото и от действията на n-те агента.

Тоест, ако s_0 е началното състояние, то s_{i+1}=World(s_i, <a_1(i), ... , a_n(i)>), където a_j(i) е действието на j-тия агент в момента i.

Въпросът, който се разглежда, е дали даден агент има стратегия за постигането или за запазването на дадено условие. Това се обобщава по естествен начин, като се разглежда същият въпрос спрямо група от няколко агента. Такава група се нарича коалиция.

Стратегията е функция, която определя действията на агента (или на коалицията). От какво зависи тази функция? Единственото, от което зависи стратегията, е s_i (текущото състояние на света). Стратегията не зависи от „историята“. Тоест, не зависи от това по какъв начин се е стигнало до това състояние на света. Това означава, че не зависи от състоянията s_0, ... s_{i-1}, нито от предходните действия a_1(t), ... , a_n(t), където t<i.

Освен от „историята“ стратегията не зависи и от „бъдещето“. Това е естествено, ако имаме стратегия, която зависи от бъдещите действия на нашите опоненти, то тази стратегия би била неизползваема, защото бъдещите действия на опонентите са неизвестни.

Последният компонент, от който би могла да зависи стратегията, това е n-торката <a_1(i), ... , a_n(i)>. Тоест може да зависи от действията на опонентите в текущия момент. Ще предполагаме, че n-те агента действат едновременно и че всеки един от тях не вижда какво са направили другите агенти в същия момент. В това предположение се крие единствената недетерминираност на така поставената задача. Ако бяхме допуснали, че агентите действат един подир друг и че всеки вижда действията на тези, които са преди него, то би се получила една напълно детерминирана система. Тогава бихме имали свойството, че ако един агент няма стратегия да запази едно свойство, то коалицията от останалите би имала стратегия, с която да може да постигне отрицанието на това свойство. (Обратната посока на тази импликация е валидна и в двата случая.)

За да видим, че при сегашната постановка горното свойство не е валидно, нека вземем следния пример. Нека имаме свят с две състояния – „четно“ и „нечетно“. Нека в този свят живеят два агента, които имат две възможни действия – „0“ и „1“. Нека светът преминава в състоянието „четно“, когато сумата по модул две от действията на двата агента е нула (новото състояние на света не зависи от старото). При това положение никой от агентите няма стратегия за запазване на състоянието „четно“, нито стратегия за постигане на „нечетно“. Тоест, горното свойство не е валидно.

2.5.2 Постановка на задачата при дефиницията на ИИ

Тук постановката на задачата е същата, с тази разлика, че в света живее само един агент (въпросният ИИ). Друга разлика е, че в първия случай се предполага, че агентите виждат всичко (т.е., че виждат текущото състояние на света), докато във втория случай се предполага, че агентът (т.е. ИИ) получава само част от информацията. Тоест, сега предполагаме, че имаме една функция View, която ограничава информацията, която ИИ получава. По този начин ИИ е като кон с капаци, защото не получава s_i, а само View(s_i). В частния случай, когато View е инекция, тогава ИИ вижда всичко, но този частен случай е възможен само в много прости светове.

При задачата на ИИ стратегиите не могат да зависят от текущото състояние на света просто защото това състояние не е видимо. Затова тук те зависят от „видимата история“ (това е редицата View(s_0), d_1, View(s_1), ... , d_i, View(s_i), където d_i е действието на ИИ в момента i). Тоест, ИИ получава много по-малко информация от агентите в първата задача. Тук ИИ знае само какво е видял и какво е правил, а в първата задача агентите знаят всичко, защото виждат текущото състояние на света, т.е. виждат всичко.

В първата задача се предполага, че ни е даден конкретен свят и конкретно условие, което трябва да се постигне или да се запази. Въпросът е има ли стратегия, която постига това условие (съответно запазва, вместо постига). В задачата на ИИ светът е неизвестен, а целта е ИИ да се представи добре в този непознат свят. За да кажем какво е добро представяне, трябва да въведем понятието „смисъл на живота“. За да облекчим разглежданията, ще изберем един конкретен смисъл на живота и той ще е следният. Нека в множеството от стойностите на функцията View има две подмножества, които ще наричаме „победа“ и „загуба“. Целта на живота ще бъде повече победи и по-малко загуби. Разбира се, ще предполагаме, че ИИ разпознава тези две подмножества, тоест че знае кога е постигнал победа и кога е загубил.

В първата задача се пита дали съществува съответната стратегия, а във втората се търси една определена стратегия. По-точно търси се ИИ, който е една изчислима стратегия (в [2] казахме, че ИИ е програма, тоест той не може да бъде неизчислима стратегия). В [14] се доказва, че тази стратегия (т.е. ИИ) съществува, дори се дава алгоритъм, който я изчислява. За съжаление този алгоритъм е напълно безполезен, тъй като не работи за разумно време. Макар да има алгоритъм, завършващ работа за краен брой стъпки, този алгоритъм е безполезен, когато този краен брой е практически безкраен.

2.5.3 Как да се обогати задачата на мулти-агентните системи

Разумно е да се предполага, че агентите не виждат всичко, а само част от света. Тоест, ще предполагаме, че имаме n функции $View_1$, ... , $View_n$, които ограничават входящата информация на агентите. В този случай стратегията на агента j няма да зависи от текущото състояние на света, а от неговата „видима история“ (това е редицата $View_j(s_0)$, $a_j(1)$, $View_j(s_1)$, ... , $a_j(i)$, $View_j(s_i)$, където $a_j(i)$ е действието j-тия агент в момента i). Тоест,

стратегията може да зависи само от това, което е видял агентът и от това, което е направил.

2.5.4 Как да се обогати задачата на ИИ

Решението на задачата на ИИ минава през разбирането на света. Тоест, ИИ трябва да намери модел на непознатия свят, в който е поставен. (Алгоритъмът от [14] не работи по този начин, но както вече казахме, този алгоритъм не върши работа.) Щом ще търсим модел на един непознат свят, трябва първо да изберем едно множество от формални модели, сред които да търсим модела, който ще пасне най-добре. Досега се ограничавахме в множеството на едно-агентните светове, но е по-разумно да се търси формален модел в по-голямото множество на мулти-агентните светове. За всеки мулти-агентен свят съществува един едно-агентен, който му е еквивалентен. Тоест, променяйки множеството на формалните модели, ние не променяме дефиницията на ИИ, която е дадена в [2]. Въпреки това, мулти-агентните светове са по-естествени и в тяхното множество ще ни е по-лесно да намерим модел на съответния свят. Както казахме, за всеки мулти-агентен свят съществува едно-агентен, който му е еквивалентен, но обикновено този едно-агентен свят е доста по-сложен и по-трудно разбираем от съответния си мулти-агентен.

Защо мулти-агентните светове са по-прости? Защото при тях можем да отделим част от сложността на света в отделен обект, който ще наречем агент. По подобен начин разсъждават и хората. Когато ние изучаваме света около нас, ние се сблъскваме с други хора, които бихме могли да приемем за част от пейзажа. Тоест, бихме могли да приемем, че живеем в едно-агентен свят и че сме еднички във вселената. Вместо това ние приемаме, че освен нас има и други хора, което прави модела на заобикалящия ни свят по-прост и по-разбираем. Често ние приемаме съществуването и на по-абстрактни обекти като Господ, съдба, късмет. Ако заключите един човек в една кула и го поставите по този начин в един едно-агентен свят, то той ще започне да си въвежда допълнителни агенти, от които по принцип не се нуждае, за да обясни света около себе си. Например такъв човек може да приеме вятъра за мислещо същество, което иска нещо да му каже. Може да реши, че някоя от вещите му го гледа лошо и да я счупи. Тоест, човекът е създаден с нагласата да обяснява света около себе си като мулти-агентна система и дори е склонен на моменти да преиграва в тази насока.

Когато говорим за мулти-агентни модели, трябва да отчетем, че различните агенти не трябва да се приемат за равноправни. Нормално е ИИ да дели агентите на „приятели“ и „врагове“. Можете да имате един агент, който е добронамерен, но е тъп и не може много да ви помогне. Друг агент може да е враг, при това може да е хитър, което да го прави още по-вреден.

Когато отделяме част от света в отделен агент, ще следваме принципа, че подходящи за обособяване в агент са обекти, чието поведение е сложно и трудно предвидимо. Обикновено това са мислещи обекти като хора и животни. Има ли смисъл обект като прахосмукачката да бъде разглеждан като отделен агент? Отговорът е – по-скоро не, защото поведението на прахосмукачката е ясно и там няма скрита сложност, която да има нужда да бъде отделена от описанието на света.

Едно важно качество, което трябва да притежава ИИ, е да може да погледне на света през очите на друг агент. Това е още една причина, поради която моделът на света трябва да бъде мулти-агентен.

2.5.5 Забележка 1

Ето едно просто доказателство на факта, че за всеки мулти-агентен свят съществува еквивалентен едно-агентен.

Доказателство. Нека имаме програма, която изчислява функцията $World_1$ на мулти-агентния свят и още n-1 програми, които изчисляват поведението на останалите агенти (т.е. на всички агенти без ИИ). Тогава функцията $World_2$ на съответния едно-агентен свят ще се получи като композиция на горните функции. ♦

Това доказателство поставя три въпроса.

Първият е дали съществуват функции, които да описват поведението на агентите. Отговорът е да, защото всеки агент задължително изпълнява някаква стратегия. Тоест, има функция, която описва поведението му. В естествения език под стратегия разбираме умно поведение, но тук под стратегия разбираме каквото и да е поведение.

Вторият въпрос е дали функциите, описващи света, и стратегиите на агентите са детерминирани или недетерминирани. В [11] разглеждахме този въпрос и видяхме, че той не е от съществено значение. Нека предполагаме, че е на лице по-общият случай и че тези функции не са детерминирани.

Последният, трети въпрос, е защо предполагаме, че тези функции са изчислими (т.е. че имаме програми, които ги изчисляват). От една страна, това предположение е излишно, защото доказателството може да мине и без него. От друга страна, нищо не ни пречи да предполагаме, че тези функции са изчислими, защото ние разполагаме само с крайна част от техните стойности (това е частта от момента на раждане до текущия момент). Тоест, ние трябва да апроксимираме тези функции, използвайки тяхна крайна част и нищо не ни пречи да изберем функцията за тази апроксимация да бъде изчислима.

Как една програма може да изчислява недетерминирана функция? Ще предполагаме, че тези програми ползват вграден генератор на случайни числа. Тоест, тук леко излизаме от традиционното понятие за изчислимост, но ако приемем, че псевдослучайните числа са случайни, то понятието за изчислимост ще е същото. В нашия случай може да не правим разлика между случайни и псевдослучайни числа, защото работим с кратък интервал от време (от раждането до текущия момент). Тоест, малко вероятно е псевдослучайните числа да започнат да се повтарят циклично или да открием зависимостта, която ги генерира.

2.5.6 Един конкретен пример

В [12, 17] се обсъжда един конкретен пример. Става дума за един изкуствен свят (изкуствен в смисъл направен от човешка ръка). В този свят ИИ трябва да се състезава с

един изкуствен противник и да играе с него играта Морски шах. В тези статии се прави опит да се намери едно-агентен модел на този свят, което се оказва доста трудна задача, защото противникът е част от света, което прави модела доста сложен. Разбира се, погрижили сме се изкуственият противник да е добре определен, за да бъде и изкуственият свят добре определен. Ако имаше неопределеност в противника, то тя би се пренесла и на света.

С две думи, при едно-агентния модел функцията World се получава прекалено сложна, защото в описанието на тази функция е скрито описанието на изкуствения противник. Там изкуственият противник е част от пейзажа, но това прави пейзажа много сложен, защото когато ИИ играе кръстче, на таблото се появява кръгче, и то само едно кръгче, и то не винаги, а само когато играта не е свършила. Много по-просто би било, ако приемаме, че кръгчетата не се появяват от само себе си, а че ги поставя някакъв агент. Какъв е точно алгоритъмът, който управлява този агент, не е нужно да се знае (макар че не пречи, ако се знае). Може да се приеме просто, че този агент е злонамерен и че се опитва да ни прецака. Идеята за злонамерения противник е залегнала в основата на алгоритъма Min-Max (това е алгоритъмът, с който „мислят“ шахматните програми).

2.5.7 Забележка 2

Когато играем Морски шах, може да си мислим, че кръгчетата се появяват от само себе си, а може да си мислим, че има някой, който ги поставя. По същия начин, когато се печем на плажа, ние може да си мислим, че облаците се появяват от само себе си, а може да си мислим, че има някой, който ги поставя. В първия случай този някой може да го наречем „противник“, а във втория случай може да го наричаме „Бога на слънчевия загар“.

Ако приемем, че кръгчетата (съответно облаците) се появяват от само себе си, то може да смятаме, че това става напълно случайно и че не можем да предвидим, нито да повлияем на този процес. Това е просто предположение, но не върши работа, защото на нас ни е нужно да повлияем на процеса, за да спечелим играта (съответно, за да получим нужния тен). Затова ние търсим някаква сложна зависимост, която да опише появяването на кръгчетата (съответно на облаците).

По-прост модел би се получил, ако допуснем съществуването на противник (съответно на Бога на слънчевия загар). В този случай естествено би било да се опитаме да повлияем на процеса, като излъжем противника или като умилостивим Бога на слънчевия загар.

Типичен пример за човешко поведение е, когато той носи дарове на някой бог или дава подкуп на някой чиновник с цел да реши някакъв проблем. При тази стратегия изниква въпросът дали моделът е адекватен. Например, дали съответният бог или чиновник съществуват. Тук не може категорично да се отговори с да или не, защото ние обикновено не общуваме директно с боговете и с чиновниците, а чрез посредници. Например, жрецът на слънчевия загар е този, който приема даровете и има грижата да ги предаде на съответния бог. Ние не можем да сме сигурни, че Бога на слънчевия загар съществува и дори не знаем дали неговият жрец съществува в действителност или е плод на нашето въображение. Тоест, това че виждаме и чуваме някого, не означава непременно, че този

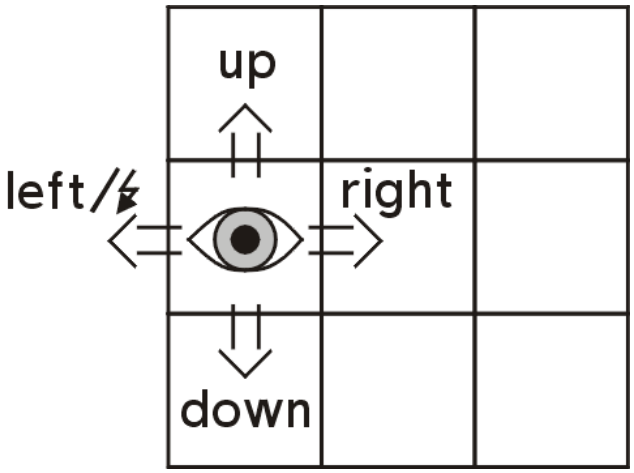
някой съществува. Разбира се, ние приемаме неговото съществуване за истина, защото това е най-простото обяснение на това, което виждаме и чуваме.

Да отговорим на въпроса съществува ли Бога на слънчевия загар. Отговорът е, че няма значение. Това, което е важно за нас, е дали ако дадем дарове на този бог, ще си осигурим повече слънчеви дни. Тоест, нас не ни интересува дали един модел е верен, а дали върши работа. Въобще дали може да говорим за верни и неверни модели. Ако разделим моделите на класове спрямо релацията еквивалентност, то тогава всеки модел има безбройно много, които са му еквивалентни. Разбира се, ние може да считаме, че еквивалентните модели са неразличими и да търсим кой да е представител на този клас на еквивалентност. Това също не решава проблема, защото ние търсим модела на базата на „видимата история“, а има безброй много модели, които имат същата видима история, но не са еквивалентни помежду си. Тоест, различават се в бъдещето или другаде (в алтернативните варианти на миналото).

За да предскажем бъдещето и за да планираме действията си ще трябва от безбройно многото възможности да изберем един конкретен модел. Кой да бъде той? Обикновено считаме, че по-простите модели са по-вероятни (този принцип е известен под името „Бръснача на Окам“). Понякога е трудно да се каже кой модел е най-прост и най-правдоподобен и затова се налага просто да изберем един от възможните модели и да го приемем за истина. Например, може да допуснем съществуването на Бога на слънчевия загар и да му принесем дарове. Това едва ли ще помогне, но вероятно няма и да навреди.

2.5.8 Формализация на света от примера

В [12] се разглежда една формализация на света, в който ИИ играе Морски шах. Там имаме едно табло (три на три) и ИИ разполага с едно око, което може да движи по таблото. Характерно за тази формализация, че ИИ вижда само едно от квадратчетата. Останалата част на таблото остава скрита и ИИ трябва да съумее да си я представи. Тоест, този свят е от интересните, защото в него ИИ не вижда всичко (т.е. функцията View не е инекция). ИИ има шест възможни хода: „надолу“, „нагоре“, „наляво“, „надясно“, „постави кръстче в текущото квадратче“ и „нова игра“. ИИ играе нов ход, като постави кръстче. Когато играта е свършила, ИИ трябва да изчисти таблото, за да започне новата игра. За това е нужна командата „нова игра“.



Фигура 1 дава идея за възможните ходове на ИИ.

Дотук описахме възможния изход на ИИ. Остава да опишем и входа. Функцията View ни връща пет бита информация. Два от тях показват какво има в текущото квадратче: „кръгче“, „кръстче“ или „празно“. Другите три бита са специални. Ще ги наречем „победа“, „загуба“ и „грешен ход“. Първите два бита ни дават смисъла на живота (както казахме, той е повече победи и по-малко загуби). Третия бит е насочващ. Може да считаме, че той отговаря на инстинкта за болка. По принцип ИИ знае, че грешният ход трябва да се избягва, но на моменти може да предпочете да играе точно такъв ход, ако смята, че това му е изгодно.

Как е при хората? Там смисълът на живота не е дефиниран и липсват първите два бита. Единственото, което хората имат, са насочващи инстинкти, каквито са болката и удоволствието. Тоест, ако пуснем човек да играе Морски шах, то той може да реши, че за него това не е особено важно и че това не е смисълът на живота му, докато за нашия ИИ няма да има нищо по-важно на света от това да победи в играта. На пръв поглед изглежда, че сме дефинирали ИИ като нещо различно от естествения интелект, защото първият има добре дефинирана цел на живота, докато вторият няма такава. Всъщност при нашата дефиниция естественият интелект е частен случай на изкуствения, защото може да се ограничим в световете, в които няма победа и загуба. В тези светове ИИ няма да има ясна цел, също както и човекът няма такава.

2.5.9 Мулти-агентна формализация на горния свят

Както казахме, мулти-агентната формализация е полезна, защото очакваме полученият модел да е по-прост и по-естествен от едно-агентната формализация. За да формализираме горния свят, може просто да вземем изкуствения противник и да го отделим в отделен агент. Разбира се, досега изкуственият противник беше добре определен (т.е. играеше по строго зададен алгоритъм). За да можем да смятаме, че изкуственият противник е независим агент, ще трябва да приемем, че той не е обвързан с фиксиран алгоритъм на игра, а че може да действа, както си иска.

По този начин можем много лесно да преминем от едно-агентен към мулти-агентен модел на играта. Въпреки това, няма да го направим по този начин, защото така би се получил несиметричен модел, а ние бихме искали двата агента да са равноправни.

2.5.10 Защо предпочитаме симетричните модели?

Една от сериозните трудности при изкуствения интелект е това, че трябва да му осигурим обучение (разбира се, това е така и при естествения интелект). Имаме вариант да осигурим човек учител (треньор, спаринг партньор). Това би било скъпо и времеемко и затова предпочитаме за партньор да използваме друга програма. За да избегнем необходимостта да пишем такава програма отделно, бихме предпочели да пуснем ИИ да играе сам срещу себе си и по този начин да се самообучава.

Естествено, за да играе ИИ сам срещу себе си, не е нужно светът да е симетричен, защото той може да се превъплъщава в различни роли и не е проблем да използваме ИИ за различните агенти. Проблемът е, че ние искаме нещо повече. Искаме да имаме един интелект, управляващ много агенти едновременно. Това е подобно на идеята за един компютър, който управлява много виртуални машини.

Ако имаме два агента в един симетричен свят, то спокойно може да предполагаме, че тях ги управлява един ИИ. Този общ интелект ще открива законите на света едновременно за двата си агента, но за всеки агент ще поддържа отделно текущо състояние. Например в света на Морския шах има един много важен краен автомат, който дава информация за това в коя от трите колони се намира окото на ИИ (виж в [12]). За да бъде открит този автомат, е нужно доста процесорно време, защото трябва да го търсим в множеството на всички крайни автомати. Затова е разумно този автомат да бъде открит само веднъж и да се използва и при двата агента. Разбира се, очите на единия и на другия агент ще бъдат в различни колони, т.е. този краен автомат ще е в различни състояния при различните агенти.

В нашия случай имаме само два агента и ползата от симетричността на света изглежда незначителна. Ще имаме нужда от двойно по-малко компютърно време, което не е съществено. Въпреки това, когато създавате ИИ вие ще трябва да минете през етап на настройка и дебъгване. Както казахме, ИИ е програма, а създаването на която и да е програма задължително минава през тези етапи. В този случай ще ви е много по-лесно, ако двата агента се управляват от една програма, работеща при едни и същи условия. В противен случай, ще имате два ИИ, които ще отговарят за различни агенти, и тяхното поведение ще е коренно различно, все едно че са в различни светове. Това е така, защото, ако на една и съща програма ѝ подавате различни данни, то тя ще има различно поведение.

Има и други случаи, в които ползата от симетричния модел е съществена. Това е, когато агентите са хиляди. Например нека да си представим как ще работят първите роботи, които ще се появят в магазина. Бихме могли да предполагаме, че всеки от тях ще е автономен и че ще има собствен ИИ (т.е. ще има компютър, на който ще се изпълнява отделно копие на програмата ИИ). Това предположение е доста съмнително, най-малкото защото в този случай всеки робот ще трябва да се обучава отделно. Много по-вероятно звучи предположението, че роботите ще имат малки компютри, които компресират входящата информация и след това я предават на някакъв централен компютър, на който се изпълнява едно копие на програмата ИИ. Този централен ИИ ще има много виртуални агенти, един от които е вашият робот. Всичките тези виртуални агенти ще бъдат управлявани директно от централния ИИ.

Този модел ще има множество преимущества. Няма да се налага да учите всеки робот поотделно как се готви лазаня. Достатъчно ще е да научите един от тях и всички ще знаят. Няма да се налага да се запознавате с всички роботи, достатъчно е да се представите на един от тях и всички ще ви познават. Ще можете да използвате роботите дори и като телефон. Можете да кажете „Робота на Пешо да му каже, че довечера го каня на вечеря.“ Разбира се, вашият робот може да звънне по телефона на робота на Пешо, но това няма да е нужно, защото те двамата ще са управлявани от един и същи мозък. Както се казва: „Лявата ръка не може да не знае какво прави дясната.“

Още по-ясно е преимуществото на този модел, ако разгледаме схемата на уличното движение. В момента имате поток от автомобили, управлявани от хора, всеки от които има свой собствен автономен интелект. За да се синхронизира този поток, се налага използването на светофари, маркировка, правила за движение и т.н. Резултатът е, че движението е хем бавно, хем несигурно.

Представете си поток от автомобили, управлявани от ИИ, при това нека всичките автомобили да са управлявани от един общ ИИ, а не всеки автомобил да е със собствено мнение. Тогава как би изглеждало едно кръстовище? По двете улици хвърчат коли с огромна скорост и просто се разминават без светофар, без да спират и най-важното, без да се блъскат. Как може да става това? Тайната е в това всички участници в движението да се управляват от един общ разум. Когато един танцьор е сам на дансинга, той няма проблем да синхронизира лявата си ръка с дясната. Когато танцьорите станат двама, проблемът със синхронизацията между тях става почти нерешим. В танците се използват редица хитрости, подобни на правилата на уличното движение. Например, предварително се заучават движенията. Също така уговарят се единия да води, а другият да го следва. Много по-добър резултат би се получил, ако можеше един разум да управлява и двамата в танцовата двойка. Тогава би се получила съвършена синхронизация, без да има нужда от правила и уговорки.

2.5.11 Как ще изглежда конкретният пример

В нашия конкретен пример на симетричен дву-агентен свят ще имаме едно табло (три на три). Всеки агент ще вижда само едно от квадратчетата на това табло и ще може да се движи по таблото независимо от противника си. Вътрешното състояние на света ще се определя от позицията на таблото, координатите, на които се намира окото на първия агент, и координатите на окото на втория. Освен това ще има още два бита информация, които ще показват кой от двамата е на ход и дали партията е свършила (т.е. дали се очаква съответният агент да постави ново кръстче или да каже „нова игра“). За да постави кръстче, агентът трябва да е на ход. В противен случай, ако се опита да сложи кръстче, то той ще получи „грешен ход“, а на таблото кръстче няма да се появи.

Както казахме, играта е Морски шах. Когато първият агент победи, той ще види победа, а вторият в същия момент ще види загуба. Първият ще играе винаги с кръстчета, а вторият винаги с кръгчета. За да бъде светът симетричен, ще сложим на втория агент едни магически очила, през които той ще вижда кръстчето като кръгче и обратното. Тоест, и двамата ще си мислят че играят с кръстчета, а противникът им че играе с кръгчета. Все пак, светът няма да е на сто процента симетричен, защото в началния момент единият агент ще е на ход, а другият няма да е. Тази малка несиметричност, няма да е съществена.

Ще въведем и правило за цайтнот. Това е, за да не може единият агент да блокира играта, като се движи по таблото, без да играе нищо. Правилото за цайтнот ще казва, че ако този, който е на ход, за 10 стъпки не направи ход, то той губи играта и другият е на ход (за да каже нова игра, с което да изчисти таблото). Ако и двамата играчи не правят нищо, то всеки десет хода единият ще губи партия (след още десет – другият). Тук числото 10 е избрано произволно. Ако това число се промени, ще имаме друг свят, с други правила на играта, но е добре да фиксираме това число, за да мислим в термините на конкретен свят.

Заради правилото за цайтнот ще добавим към вътрешното състояние на света един брояч, който ще отброява колко стъпки са минали, откакто съответният агент е на ход. Този брояч няма да е видим от агента, защото неговата функция View няма да го показва. Разбира се, агентът може да брой до десет, но проблемът е, че той няма да знае кога да започне да брои. Когато види ново кръгче на таблото, ще знае, че противникът му е играл и че той има не повече от 10 стъпки, за да му отговори, но няма да знае колко точно стъпки му остават, защото няма да знае точно преди колко стъпки противникът е сложил това ново кръгче. Стратегията на играч, който не е на ход, ще бъде да обикаля празните квадрати и да търси новото кръгче, както и да се опитва да играе от време на време, с предположението, че противникът му вече е играл и че той вече е на ход.

Бихме могли правилото за цайтнот да го въведем не на базата брой стъпки, а на базата на реално време. Това би усложнило модела, защото нещата ще зависят от времето, което е нужно на ИИ да направи една стъпка. По-лесно ще ни бъде да приемем, че това време е константа. Тоест, ще приемем, че определен брой стъпки отговарят на определено реално време или може да си мислим, че реалното време не съществува, а имаме само брой стъпки в изкуствен свят.

Ако погледнем на този дву-агентен свят от гледната точка на първия агент, то ще получим един едно-агентен свят, който е подобен на света, описан в [12, 17]. Къде е разликата? Там, когато ИИ играеше кръстче, на таблото веднага се появяваше кръгче, докато сега това става след известен брой стъпки. По-рано вграденият противник виждаше всичко и затова можеше да играе веднага. Сега другият агент вижда същото, като ИИ, и има нужда от време (т.е. стъпки), за да може да обиколи таблото и да го разгледа.

2.5.12 Как да стигнем до процедурата Min-Max

В [17] ние успяхме да построим модел на този свят (там светът беше приблизително същият). Там намерихме краен автомат, който описва текущите координати на окото на ИИ. Освен това намерихме и предикат, който описва позицията на таблото (предикатът беше триместен, като аргументи му бяха x и y координатите на окото и времето, т.е. номерът на текущата стъпка). На базата на този предикат и формули от първи ред ние описахме условията за постигане на победа. Тоест, в [17] ние намерихме модел на света, който ни описва коректно правилата на играта. Въпреки това, ние не успяхме да довършим нещата и на базата на този модел да направим програма, която играе играта Морски шах.

Забележка. Предикатът описващ позицията на таблото, връща три възможни стойности и затова е по-добре да говорим не за предикат, а за функция или за двойка предикати, единият описващ кръстчетата, а другият кръгчетата. Това са технически детайли, които ще пренебрегнем.

Това, което остана недовършено в [17], е да стигнем до процедурата Min-Max или до друга подобна като Max-Sum (виж [14]). Именно идеята на мулти-агентния модел ще ни помогне да приложим Min-Max процедурата.

Когато искаме да накараме ИИ да планира бъдещите си ходове, е естествено да приложим Min-Max на принципа ИИ срещу света. Тоест, смятаме максимум по всички възможни стъпки, които може да направи ИИ и след това минимум по всички възможни реакции на света. В [16] обсъждахме подобна възможност и установихме, че тя директно води до комбинаторна експлозия. Обърнете внимание, че на всеки ход отговарят десетина стъпки (защото ИИ и неговият противник първо трябва да се придвижат до празното квадратче и чак тогава могат да играят). Това означава, че дървото на Min-Max процедурата става десетина пъти по-високо. Като се има предвид, че сложността на Min-Max е в експоненциална зависимост от височината на това дърво, то веднага разбираме, че по този начин директно стигаме до комбинаторна експлозия.

За да стане Min-Max процедурата работеща, тя трябва да бъде на принципа ИИ срещу другия агент. За целта ИИ трябва да осъзнае, че най-важното в света на Морския шах е позицията на таблото. Разбира се, тази позиция не е директно видима, тоест, става дума за една абстракция. Както видяхме в [17], ИИ може да открие предиката, описващ положението на таблото, тоест ИИ може да разбере тази абстракция.

Следващото нещо, което ИИ трябва да осъзнае, е, че той може да променя позицията на таблото. Не е трудно да се досети, че ако сложи кръстче на празно квадратче, то ще промени позицията (т.е. ще промени стойността на предиката за текущите x и y и следващото t). Не е трудно да осъзнае, че може да се движи по таблото и да достигне до произволно празно квадратче. Тоест, можем да смятаме, че ИИ ще стигне до идеята, че може да постави кръстче на произволно празно квадратче. Това означава, че ИИ може да разбере кои са възможните ходове, с които разполага.

Следващата абстракция, до която ИИ трябва да стигне, е идеята за виртуалния противник, който живее в същия свят и който има право да сложи кръгче на произволно празно квадратче.

Сега ИИ е готово да приложи Min-Max процедурата. Единственото, което остава да реши, е дали другият агент му е приятел или враг. Тоест, дали процедурата трябва да е Min-Max или Max-Max. Това ще е лесно. Нека ИИ вярва в доброто и предполага, че другият агент му е приятел. Много скоро това ще се промени, защото другият агент ще играе по възможно най-лошия начин, вместо да помага като приятел. Тоест, другия агент бързо ще загуби доверието на нашия ИИ и ще бъде обявен за враг.

2.5.13 Варианти на света от примера

Ще разгледаме няколко варианта на горния свят, за да видим, че Min-Max не е единствената възможна стратегия за ИИ. Тоест, светът може да е от вида „аз и един враг“, но има и други варианти.

Нека в същия свят да има четири агента, които играят Морски шах, като първият слага кръстче, вторият кръгче и т.н. Тоест, първият и третият играят срещу втория и четвъртия. Тогава процедурата ще бъде от вида Max-Min-Max-Min, където първото Max е по възможните ходове на ИИ, а второто е по възможните ходове на третия агент (който играе на страната на ИИ).

Нека променим играта и предположим, че третият агент е купен от противниковия отбор. Тоест, той пак играе кръстче, но го слага не по най-добрия, а по най-лошия начин. Тогава процедурата ще има вида Max-Min-Min-Min.

Нека сега предположим, че третият и четвъртият агент не мислят, а играят напълно произволно. Тогава процедурата ще изглежда така: Max-Min-Average-Average. Тук Average ще означава, че се взима средно аритметично от всичките възможни ходове.

При всички варианти, които разгледахме досега, агентите играят един след друг (тоест, изчакват се). Нека разгледаме вариант на примера, в който агентите играят едновременно. Нека в началото на партията първият агент да има две кръстчета и да може да ги сложи на таблото, когато си поиска. Нека на всеки 10 стъпки той да получава ново кръстче. Аналогично и за втория агент. Тук няма да има ред (т.е. единият да е на ход, а другият да чака). Няма да има и цайтнот. Ако двамата едновременно се опитат да играят в едно и също квадратче, нека приоритет да има първият. Ако и двамата едновременно направят линия, то нека играта да е реми. В светове като този също може да се използва процедура, подобна на Min-Max, макар дървото на възможните развития да е по-сложно, отколкото е в случая, когато агентите се редуват.

2.5.14 Заключение

Мулти-агентните модели са верният път при решаването на задачата за създаването на ИИ. Макар, че алгоритъмът на ИИ до голяма степен е ясен, все пак за създаването на работещ прототип ще трябва да бъдат преодолени ред технически и концептуални проблеми.

3 Какво ще правим след като го направим?

3.1 AI не трябва да е Open Source Project

Има много хора, които защитават идеята, че технологията AI трябва да се разпространява свободно и дори, че тя трябва да бъде Open Source Project. Между тези хора има дори отговорни и сериозни хора, какъвто е президента Макрон [37]. Тук ще се опитаме да поспорим с тези хора и да им обясним колко погрешно и дори пагубно би било подобно решение.

Когато президента Макрон [37] говори за отворени алгоритми, може би той по-скоро има предвид собствеността върху тези алгоритми. Разбира се, няма нищо лошо собствеността да бъде на всички, но това не значи, че кода на тези алгоритми може да бъде общодостъпен. Например една ядрена електроцентрала може да бъде държавна, тоест на всички, но това не значи, че технологиите използвани в тази централа са общодостъпни и че всеки може да вземе чертежите и по тях да си направи собствена ядрена централа.

В момента отношението към технологията Изкуствен Интелект е изключително безотговорно. Ние се намираме в зората на развитието на тази технология и въобще не можем да си представим каква мощ и неподозирани възможности крие това изобретение. Какво се е случило през 1896 година? Тогава Анри Бекерел [38] открива, че ако постави в чекмедже парче уранова руда върху фотографска плака, то след време плаката се осветява.

Ако постави метален ключ между плаката и рудата, то изображение на ключа се отпечатва върху плаката. Така Бекерел открива радиоактивността, но по това време той въобще не може да си представи потенциала, който крие тази технология. Експеримента на Бекерел е забавен и е по-скоро нещо като фокус. Същото в момента е положението с Изкуствения Интелект. Появяват се експерименти, които са интересни и забавни, но хората въобще не си представят до къде може да ни доведе тази технология.

Можели ли са хората през 1896 година да предвидят колко мощна и опасна е ядрената технология? Тогава не е имало как да се досетят. Това става по-късно, когато откриват колко много енергия се отделя при разпада на атомното ядро.

Можем ли сега да се досетим колко опасна е технологията Изкуствен Интелект? Да, и повечето разумни хора се досещат, макар и да не осъзнават действителния мащаб на това откритие.

Всеки разумен и отговорен човек трябва да си зададе въпросът дали да участва в разработката на новата технология или да остави това на глупавите и безотговорните.

В тази статия се занимаваме с технологичните катастрофи, които могат да се предотвратят, а не с неизбежните последствия, които няма как да бъдат предотвратени. Например, ако дадете на един глупак моторен трион, той може да изсече цялата гора и това е неизбежно последствие. Ако глупака си отреже крака, това е технологична катастрофа, която би била предотвратена, ако глупака не беше чак толкова глупав и ако внимаваше повече.

Казваме, че един много мощен интелект е нещо опасно. Това не е нова идея. Още Adorno и Horkheimer [39] казват, че разума може да бъде друга форма на варварство. Те казват, че с помощта на интелекта хората могат да променят природата по безогледен варварски начин. В [39] не се говори за изкуствен интелект, а за бюрократичната машина. Приликите между изкуствения интелект и бюрократичната машина са повече отколкото разликите и затова казаното в [39] може да се приеме за казано по темата. В [39] авторите разглеждат случая когато група хора използват бюрократичната машина като оръжие с цел да подчинят останалите (тоталитарната държава). В [39] не се разглежда случая, когато бюрократичната машина излиза извън контрол и започва да действа против волята на хората, защото това е невъзможно. Обществото като цяло винаги може да промени законите и правилата, които управляват бюрократичната машина. Тоест обществото като цяло няма как да загуби контрола върху бюрократичната машина, но отделния човек няма такъв контрол. За отделния човек бюрократичната машина е даденост, която той не може да промени. Аналогична е ситуацията с изкуствения интелект. Обществото като цяло ще запази контрола си върху AI, освен ако не сме достатъчно глупави да го изпуснем, но за отделния човек AI ще е даденост, която не може да бъде променена. Последното е едно от неизбежните последствия от появата на AI, което излиза извън обхвата на темите на тази статия.

3.1.1 Какво може да се случи?

Може да се случи катастрофа. Такава катастрофа може да е предизвикана умишлено или неволно. При ядрените технологии имената да две такива катастрофи са Хирошима и Чернобил. Първата е предизвикана умишлено, а втората поради глупост и невнимание.

Умишлена катастрофа означава някой да реши да използва новата технология със зъл умисъл, тоест като оръжие. Тук понятието зъл умисъл е относително, защото всеки създател на оръжие смята, че създава нещо полезно и че убивайки хора, той спасява живота на други хора. Обикновено се твърди, че убиваме малко, за да спасим много или поне че убиваме от чуждите, за да спасим от нашите.

Може ли AI да убива? Не действат ли законите на роботиката [40], които Айзък Айзимов е измислил? Всъщност тези закони са едно добро пожелание и те по никакъв начин не задължават създателите на AI да се съобразяват с тях. В момента технологиите, които претендират да имат нещо общо с AI се използват предимно за оръжия. Например, така наречените „умни бомби“. Обикновено се твърди, че глупавата бомба убива наред, докато умната убива само тези, които сме й казали да убие. Тоест умната бомба е по-малко кръвожадна и по-хуманна. Това последното е по-скоро оправдание за създателите на умни бомби. Тези бомби са по-мощно оръжие от глупавите бомби и с тях можем да убием повече хора отколкото със старите глупави бомби.

Нека си представим, че се е случила технологична авария и някой е изпуснал духа от бутилката и е загубил контрола над AI. Ако гледаме на AI като на оръжие, нека си представим, че някой използва това оръжие срещу нас. По този начин отношението ни към използването на AI като оръжие ще е строго негативно, защото отношението към едно оръжие много зависи от това дали ние го използваме или някой го използва срещу нас.

Имаме ли шанс да преживеем подобна технологична авария? Отговорът е, нямаме никакъв шанс. Във фантастични филми от рода на Терминатора [41] се описва как хората воюват с роботите, но там имаме едни много тъпи роботи, които са много по-тъпи от хората. Истината за AI е, че той ще е много по-умен, от който и да е човек. Тоест, идеята за равноправно състезание между хора и роботи е безсмислена. Също толкова безсмислено е да организираме равноправно състезание по тичане между хора и автомобили. Хората няма да имат никакъв шанс, защото автомобилите са много по-бързи от хората.

Тоест, ние не можем да си позволим подобна технологична авария, просто защото няма да я преживеем. В историята си човечеството е преживяло много природни бедствия и технологични аварии, но винаги ефекта от тези аварии е бил локален. Например аварията в Халифакс причинява взрив, който разрушава града. Въпреки това, пораженията са локални и са ограничени до един град. Като възможно най-страшния сценарии за катастрофа се споменава ядрената война, но дори и това би имало само локални последствия. Една подобна война би могла да унищожи градовете, но все някое от селата би оцеляло. Тоест, дори и възможна ядрена война не представлява такъв риск, какъвто е евентуалната загуба на контрола над AI.

3.1.2 Можем ли да заключим звяра в клетка?

Можем ли да създадем AI, но за да сме сигурни, че няма да направи някоя беля да го затворим в една виртуална реалност? Така ще можем да го наблюдаваме отстрани и да го изучаваме, но винаги ще можем да дръпнем щепсела и да го изключим, ако решим.

Да, можем. Ако AI няма вход и изход (тоест уши и уста), тогава той не може целенасочено да повлияе на външния свят. Дори и да повлияе, то това няма да е целенасочено, защото той за външния свят въобще няма да знае. Например, ние хората знаем ли дали не сме в Матрицата (филма Матрицата [42]). Както ние не знаем дали нашият свят е истински или виртуален, така и AI няма да знае.

Достатъчно е само AI да няма вход (тоест да не получава никаква информация от външния свят). Що се отнася до изхода, щом ние наблюдаваме AI, то той има изход (ръце и уста), защото чрез своето поведение той ще влияе на нас и от там, чрез нас ще повлияе на външния свят.

Тоест, рецептата е много проста. Държим AI в една виртуална реалност и така избягваме риска от технологична катастрофа. Да, но ние ще искаме AI да ни каже нещо за реалния свят. Например да ни направи прогноза за времето или да ни каже какви ще са цените на борсата. Може да поискаме да свърши някаква работа, например да измие чиниите или да измете пода. За всичките тези неща ще е нужно да го пуснем от виртуалната реалност и да му позволим да влезе в реалния свят. Дори и да се уговорим да не го пускаме, все ще се намери някой, който ще се изкуши и ще го пусне.

Можем ли да заключим AI в клетка без да го лишаваме от информация за външния свят? Тоест, да може да чува реалния свят и да го гледа през решетката, но въпреки това да си запазим възможността винаги да можем да дръпнем щепсела и да го изключим.

Може ли лъвът да избяга от зоологическата градина? Може, макар че това е малко вероятно, защото лъвът е много глупав. Достатъчно е да има едно обикновено резе на вратата на клетката и лъвът няма да се сети как да си отвори. Маймуната има по-големи шансове да избяга, защото тя е по-умна от лъва. Ако разчитаме на обикновено резе, това няма да спре маймуната, защото тя ще се сети как да го отвори. Най-трудно е да заключим човек. Бягства се случват дори и в най-добре охраняваните затвори. Хората са много умни и почти винаги намират начин как да се измъкнат. Нека сега си представим, че сме се опитали да заключим едно същество, което е много по-умно от нас. Нека си представим човек охраняван от маймуни. Ще успее ли човека да надхитри маймуните и да избяга?

Добре, но ние винаги ще имаме възможността да дръпнем щепсела и да изключим AI, ако решим, че той е излязъл извън контрол. Да, но може и да не можем да го изключим. Когато AI излезе извън контрол той може да реши да не ни позволява да го изключим.

След като AI не може да бъде заключен в клетка, можем ли да го използваме в реалния свят? Да можем, но трябва да сме сигурни, че сме създали един добронамерен AI, който няма да се опитва да вземе властта от нас. Например кучето може да избяга и може да ни ухапе, но не го прави, защото е добронамерено.

Тоест, ако внимаваме какъв AI създаваме, няма да се случи технологична катастрофа. Това означава, че хората, на които ще разрешим да се занимават с тази технология трябва да са

достатъчно умни и отговорни. Ако такива бяха хората занимаващи се с ядрени технологии нямаше да се случи нито Хирошима, нито Чернобил.

3.1.3 Принципа на моркова и тоягата

Ето един много прост пример за технологична катастрофа. Когато говорим за AI предполагаме, че той се обучава с поощрения и наказания (тази концепция е известна като Reinforcement Learning). Това е принципа на моркова и тоягата. Целта на AI е ясна, повече поощрения и по-малко наказания. Можем да стартираме такъв AI и да дадем на човека, който го управлява два бутона, с които да поощрява и да наказва AI. Проблем би възникнал, ако AI реши да забрани да човека да натиска бутона за наказание и да го принуди непрекъснато да натиска бутона за поощрение. По този начин AI ще превърне човека в свой роб, който е принуден непрекъснато да натиска бутона за поощрение.

Подобно нещо би се случило, ако магарето вземе властта и забрани на стопанина си да използва тоягата и ако го принуди по цял ден да го храни с моркови. Слава богу магарето не е достатъчно умно, за да вземе властта и това не може да се случи.

3.1.4 Можем ли да не създаваме AI?

Можем ли да се уговорим да не отваряме кутията на Пандора? Могат ли учените, които се занимават с изследвания в областта на Изкуствения Интелект да се съберат и да се уговорят да не правят това откритие? Отговорът е, че това няма как да се случи. Дори и част от учените да успеят да постигнат подобно споразумение, ще има и такива които няма да са част от споразумението или ще са част, но няма да се съобразят и ще го нарушат.

Подобно нещо се случва с ядрената технология. Физикът Вернер Хайзенберг твърди, че през 1941 година се среща тайно със бившия си учител Нилс Бор в Копенхаген и те двамата постигат уговорка да не създават атомната бомба. Нилс Бор от своя стана отрича да е сключвал подобно споразумение. Имало ли е или е нямало подобно споразумение е без значение, защото и Германия и САЩ не спират ядрените си програми. Възможно е, отделни хора да са саботирали развитието на ядрените технологии, но е имало и достатъчно много хора, които са продължили да работят по темата.

3.1.5 Защо трябва да засекретим AI технологията?

Всяка опасна технология се засекретява и достъпа до нея се ограничава. Пример за това са огнестрелните оръжия. Защо не позволяваме на децата да си играят с картечници? Каква беля може да се случи? Детето може да застреля един-двама, най-много сто човека. С помощта на AI технологията детето може да направи много по-голяма беля. Например, детето може да зададе задача на AI да избие всички, които не му харесват и това може да се окаже, че са всички хора, включително и този, който е задал задачата.

Сега ще ми кажете, че AI технологията е много сложна и едно дете не може да се справи с нея. Аз не твърдя, че едно дете може да създаде тази технология, но твърдя, че ако му я предоставим Open Source, то то би могло да я използва. Същото е и с картечницата. Едно дете не може само да си направи картечница, но ако му дадем една, то то би могло да стреля с нея. В това няма нищо сложно. Дърпаш спусъка и тя започва да стреля.

Подобно е положението с хаковете за компютри. Това са пропуски в сигурността на вашия компютър, които позволяват да се проникне дистанционно в него. Хаковете са или злоумишлено направени или са резултат на неволна грешка. Обикновено при създаването на една операционна система се оставят пропуски в сигурността, които по-късно да позволят проникването и контролирането на компютрите работещи с тази операционна система. Идеята на тези хакове е да се пазят в тайна и да се ползват само от техните създатели. Какво се случва на практика? Някой открива някакъв хак и го публикува в мрежата. Идеята е, да може всеки да се защити. Резултата е, че всеки може да го използва и да проникне в компютъра ви.

Целта на умишлените хакове е да могат тайните служби да влизат и да контролират вашия компютър. Аз лично нямам нищо против това тайните служби да влизат и да контролират моя компютър, защото знам, че това са хора отговорни и че ще влязат и излязат от моя компютър и аз дори няма и да разбера, че са влизали.

Какво става обаче, когато в компютъра ви проникне някой тийнейджър. Той ще иска да отбележи своето посещение. Ще затрие някой важен файл и ще напише нещо неприлично на десктопа. Как тийнейджърите успяват да открият пропуските в сигурността, толкова ли са умни? Истината е, че въобще не са умни. Просто използват публикувани хакове, които са Open Source и които могат да се ползват от всеки. Ако някой тийнейджър е достатъчно умен сам да открие хак, то той вероятно ще е и достатъчно отговорен, за да не драска мръсотии върху десктопа ви.

3.1.6 Секретни списания

Мисля, че вече сте съгласни, че технологията AI не трябва да попада в ръцете на деца и на безотговорни хора. По същия начин сте съгласни, че на децата не трябва да им разрешаваме да си играят с картечница.

Може ли да дадем на децата едно достатъчно добро упътване, което да им позволи сами да си направят картечница? Разбира се, най-малките няма да се справят, но техните батковци биха могли, особено ако упътването е достатъчно подробно и добре написано. Може ли да смятаме, че всеки, които е достатъчно умен да направи картечница по упътване е и достатъчно отговорен, за да я ползва? Според мен не можем да направим такова предположение и затова упътванията за това как се прави картечница трябва да са с ограничен достъп.

Същото важи и за статиите, в които се описва AI технологията. Първо да кажем какво е AI. Изкуственият Интелект е програма. Една програма може да бъде написана. Написването на програмата е техническа задача подобна на решаването на ребус. Разбира се, преди да напишем програмата трябва да измислим алгоритъма. Трябва да си доста умен, за да

измислиш алгоритъма на AI, но този алгоритъм може да бъде подробно описан в някоя статия за AI. Ако разполагаш с такава статия, самото написване на програмата AI може да се окаже техническа задача. Затова, според мен, статиите за AI трябва да са с ограничен достъп.

Навремето в бившия Съветски Съюз имаше секретни списания. Всички военни и потенциално опасни технологии се отпечатваха в такива списания. Разбира се, хората, които имаха право да четат тези списания бяха внимателно подбирани и техния кръг беше силно ограничен. Смятам, че днешните AI технологии трябва да се публикуват само в такива секретни списания.

3.1.7 Сериозните списания

Може ли в сериозно научно списание да се появи статия описваща алгоритъма на AI? Това е почти невъзможно, защото сериозните списания имат сериозна цензура, която не допуска статии, които биха представлявали опасност. Тоест, те не биха допуснали статия, която описва нова неизвестна и потенциално опасна технология. Цензурите се наричат рецензенти. Те са анонимни и не носят никаква отговорност за своите рецензии.

Това, че сериозните списания не допускат сериозни статии описващи AI технологията не означава, че такива статии не се появяват. Учените отхвърлени от рецензентите публикуват техните резултати където им попадне. Често те публикуват на своите Интернет страници или в различни блогове. Срещу това безразборно публикуване се взеха мерки в последните години. Например, имаше сайтове които пазеха снимка на Интернет и там можеше да се види една статия кога е публикувана. Тези сайтове бяха затворени. Също така блоговете даваха дата на публикациите. Сега вече блоговете не дават такава дата (по-точно дата има, но няма дата на последната редакция). Въпреки всичко безразборното публикуване продължава и цензурата не въвежда ред, а води до още по-голяма анархия.

В хартиените списания цензурата е неизбежна, защото хартията е ограничен ресурс и не може всеки да публикува каквото му скимне. Разбира се, хартиените списания са една отживелица от миналото. Днес имаме електронни списания, където не е нужно да се ограничава дължината на статията, нито да се спират статии, затова че са глупави или прекалено умни.

Дори и едно електронно списание може да се претовари, защото един автор може да генерира и да изпрати един милион статии. Тоест, при електронните списания имаме явлението наречено SPAM. Затова е добре когато се подават статии в електронното списание да има малка такса. Например един долар на статия или един долар на страница.

Защо рецензентите спират всички статии, които излизат от традиционния шаблон и не са нещо, което се дъвче поне от 50 години? Може би рецензентите са отговорни хора и искат да предпазят човечеството от неконтролираното навлизане на новите технологии, а може те просто да са глупави и да не могат да разберат една статия, ако тя е малко по-нестандартна. Може би го има и едното и другото. Може би рецензентите просто са ревниви и се дразнят от всеки, който се прави на много умен, особено ако този някой е пропуснал да цитира рецензента.

Въпреки всичко рецензентите са нужни дори и в електронните списания, но не за да казват да или не, а за да оценяват статиите и да насочват читателите струва ли си тази статия да бъде прочетена. Тоест, рецензентите са нужни, но не в качеството им на цензури, а в качеството им на критици. В литературата имаме литературни критици, които оценяват романите и насочват читателите. Тези критици застават зад написаното с името си и носят отговорност за критиката си, защото ако заблудят читателя, той повече няма да вярва на тяхната критика.

Ако списанието е секретно, то рецензента ще е нужен, за да каже какво е нивото на секретност на дадена статия. Тоест, колко ограничен да е кръга от хора, които ще могат да я прочетат.

Това, че списанието е секретно ще означава, че само проверени, достатъчно отговорни хора ще могат да го четат, но там всеки трябва да може да публикува. Ако ограничим хората, които имат право да публикуват, резултата ще е, че голяма част от учените ще продължат да публикуват там където им попадне.

3.1.8 Заключени компютри

Както вече казахме, AI е програма и това е една много опасна програма. Затова не трябва да позволяваме на безотговорни хора да си играят с нея. Една програма има смисъл само, ако имате компютър, на който да я пуснете. Програмата без компютър представлява просто безсмислен текст.

Днес всеки тийнейджър разполага с много мощен компютър и може да пусне на този компютър каквато си поиска програма. Навремето баща ми беше директор на най-големия изчислителен център в България за научни цели. В момента имам в джоба си много по-мощен компютър от това, на което беше директор баща ми.

Много хора се притесняват от това, че Северна Корея разполага с термоядрено оръжие. Аз повече се притеснявам от това, че те разполагат със супер компютър, на който биха могли да стартират програмата AI и да направят много по-големи поразии, отколкото би направила една термоядрена бомба.

Все пак, Северна Корея се управлява от пълнолетни хора, които носят отговорност за действията си, но ние позволяваме на всяко непълнолетно дете, което по закон не отговаря за действията си, да притежава компютър и да пуска на него каквато си поиска програма.

Щом не позволяваме на децата да си играят с огнестрелно оръжие, защо им позволяваме да си играят с компютри? Дали не трябва да забраним на обикновените хора да притежават компютър? Дали не трябва за притежаването на компютър да се иска разрешение, както се иска за притежаването на огнестрелно оръжие.

По времето на Китайската империя е имало закон забраняващ на обикновените хора да притежават оръжие. Това е причината, поради която в Китайската империя са измислени различни техники за ръкопашен бой, както и за бой с пръчка и с разни селскостопански

инструменти (например нунджако). Дали не е дошло времето да се приеме закон забраняващ на обикновените хора да притежават компютри?

Идеята да се ограничи притежаването на компютри вече е реализиран в известна степен. Например смартфоните представляват компютри, но това са заключени компютри, на който не можеш да пуснеш каквато си поискаш програма, а само програма проверена и одобрена от някой, който решава вместо нас коя програма е опасна и коя е безопасна.

Постепенно в заключени компютри се превръщат таблетите и лаптопите. Все още, огромната част от настолните компютри не са заключени и позволяват на хората да програмират и да пускат на тях собствени програми. Предполагам, че и това ще е до време и че идва деня, в който всички компютри ще са заключени и притежаването на отключен компютър ще се наказва като едно от най-опасните углавни престъпления.

Преди двадесетина години хората оставяха компютрите си включени през нощта и позволяваха всеки желаещ да ги ползва и да изпълнява програми (говоря за компютрите работещи под UNIX). Тоест, имаше един огромен супер компютър, който беше на разположение на всеки, който пожелае да си поиграе с него. Днес благодарение на технологията Bitcoin този ресурс вече не е свободен. Сега всички свободни компютърни ресурси са впрегнати да копаят биткойни. Дори някой да има свободен компютърен ресурс, той не би го предоставил за свободно ползване, защото знае, че някой друг ще се възползва, ще изкопае биткойни и ще спечели за негова сметка.

3.1.9 Заключение

Трябва много сериозно да се отнесем към технологията AI и да засекретим всички програми, които имат нещо общо с тази технология. Трябва да засекретим и статиите в областта на AI и дори трябва да заключим компютрите, за да не може всеки да си играе и да експериментира с AI технологиите. Тези експерименти трябва да бъдат разрешени само на хора, които са достатъчно умни и отговорни. В момента, за да работиш като лекар трябва да отговаряш на куп изисквания, а за да правиш изследвания в областта на AI, не се иска нищо. Това трябва да се промени и да се въведат изисквания, на които трябва да отговарят AI изследователите.

Технологичната катастрофа, да бъде изпуснат контрола над AI, може да се случи от глупост или от безотговорност. Не трябва да позволяваме с AI да се занимават хора с комплекс за малоценност, които биха могли да пожелаят да получат абсолютната власт и да се превърнат в нещо като господар на вселената.

От друга страна трябва да позволим на независимите изследователи да публикуват (в секретните списания) и по този начин да им позволим да получат признание за своя труд. Разбира се, когато независимите изследователи публикуват, те трябва да получат дата и гаранция, че никой няма да се опита да оспори или открадне тяхната заслуга.

Когато говорим за технологична катастрофа имаме предвид неща, които могат да се избегнат и които ако сме достатъчно умни и отговорни ще избегнем. Това са загубата на

контрол върху AI и използването на AI като оръжие или като средство една група хора да подчини останалите.

Има много други последствия на AI технологията, които са неизбежни. Например това, че хората ще си загубят работата е такова последствие, което не можем да предотвратим. Освен, че не можем да го предотвратим, ние не искаме да го предотвратяваме, защото никой от нас не иска да е принуден да работи. Бихме работили за удоволствие, но не искаме да работим по принуда.

Много хора се притесняват, че тайните служби им ровичкат из компютрите. Днес тайните служби виждат всичко и знаят всичко. Господ също вижда всичко и знае всичко, но това не притеснява никого. Разбира се, Господ е дискретен и добронамерен. Той няма да каже на жена ви, че сте й изневерил, нито ще се възползва от информацията на вашия компютър за своя изгода. Тайните служби също са дискретни, но не винаги са добронамерени. Не случайно най-големите бандити обикновено са бивши или настоящи служители на тайните служби.

Няма нужда да се притесняваме, от това че тайните служби виждат всичко. Това е нещо неизбежно. Глупаво е да се притесняваме от неизбежни неща, които не можем да променим. Казахме, че в тези служби работят хора отговорни. Е, по-отговорни са от тийнейджърите, но това не означава, че са достатъчно отговорни. Начина да избегнем проблеми не е да се крием от тайните служби, а да ги контролираме и да внимаваме, кой работи там.

Моята теза е, че трябва да разрешим на тайните служби официално, без да се крият, да контролират нашите компютри. Ако го направим ще забравим проблеми като компютърни хакове, вируси и SPAM. Компютрите ни ще станат сигурни и надеждни. Дори може да поискаме, когато ни гръмне харддиска, те да ни възстановят информацията. Така или иначе те пазят нашата информация при тях.

Кой работи в тайните служби е важно, но още по-важно е на кого ще позволим да прави изследвания в областта на AI. Това трябва да са умни, разумни и отговорни хора, без комплекс за малоценност и без престъпни намерения. Ако разрешим на всеки, който си поиска, да се занимава с подобни изследвания и експерименти, то ще се случат технологични катастрофи в сравнение, с които Хирошима и Чернобил ще ни се видят като дребни инциденти.

3.2 Как ще изглежда живота ни след появата на ИИ?

Идеята на ИИ е, че човека създава същество, което е несравнимо по-умно от него самия. Макар, че това същество ще е добронамерено и ще ни служи вярно, за нас то може да е проблем, защото сега ние се гордеем с това, че сме най-умните и че сме по-умни от всички други животни и дори от машините. Нашия интелект е това, което ни дава самочувствието, че сме върха на еволюцията. Сега ние управляваме планетата Земя и ние решаваме кое животно и кое растение заслужава да живее и кое да се размножи и да заеме повече площ и кое да бъде ограничено само в резерватите.

Въпросът е как ще изглежда нашия живот след появата на ИИ. Тогава живота ни ще бъде измамно лесен. Няма да има нужда да мислим за прехраната си, няма да ни се налага да работим, дори няма да е нужно да се забавляваме един друг, защото ИИ ще ни забавлява много по-добре, отколкото който и да е човек би ни забавлявал. Въпреки измамната простота на живота естествения подбор ще продължи и едни ще оцеляват, а други ще изчезват.

Като говорим за естествен подбор не говорим за умиране, а за размножаване. След появата на ИИ почти никой няма да умира. Човешкото тяло може да се ремонтира и да се клонира и да продължи да съществува практически вечно, но ние може да решим да поставим граница и да кажем, че никой няма да има право да живее повече от 120 години. Колко хора ще оставим да живеят на Земята? Може да са 7 милиарда, може да ги увеличим до 70 или до 700, но може би е добре да се сложи някаква граница, защото ако сме прекалено много ще започнем да си пречим, а и няма да остане никакво място да другите видове.

Какво ще правим след появата на ИИ? След като няма да работим, единственото смислено нещо ще е да се отдадем на размножаване. То и сега размножаването е най-важното, но сега ние работим, за да се размножим. Когато работата не е важна, няма да са важни и парите, защото с пари измерваме труда на хората, тогава кой ще е новия критерии на естествения подбор? Сега критериите са: интелект, красота, здраве, образование, сила, смелост, бързина, честност, религия и мироглед.

Силата и бързината са били много важни в миналото, но сега когато машините са много по-силни и по бързи от нас хората, силата и бързината не са най-важното. Когато машините станат по-умни от нас, тогава и интелекта няма да е най-важното.Смелостта е сложен критерии. От една страна печелят смелите, но от друга, най-смелите си чупят главата. Подобно е и положението с честността. Най-успешни са бизнесмените и политиците, които не блестят с особена честност, но най-нечестните влизат в затвора. Образованието е било еволюционно предимство в миналото, но днес то е по-скоро недостатък. Моят учител по рисуване казваше, че процента на старите моми сред висшистките е много по-голям от средното, а ако една жена защити докторантура, то положението е направо неспасяемо. Тоест, трябва ли образованието да бъде ценност и трябва ли да се даде по-голям шанс на по-образованите?

Що се отнася до здравето, то безспорно е ценност, но ако всички са здрави, то тогава това не може да е критерии. Красотата е друг много важен критерии, но той е твърде субективен. Кой ще определя кой е красив и кой е грозен. То и сега има възможност за много сериозни козметични корекции, а след появата на ИИ ще можем да докараме какъвто си поискаме външния вид.

Кой ще определи новите критерии на естествения подбор? Дали да не оставим това на по-умния, тоест на ИИ? Градинаря определя кое цвете е красиво и интересно и кое заслужава да се размножи и да заеме повече място в градината. Ако искаме ние да сме тези, които ще определят критериите, тогава трябва да помислим по този въпрос. Тогава вероятно най-важния критерии ще е религията и мирогледа. Господстващия мироглед ще се наложи и ще даде по-голям шанс на тези, които споделят религията на мнозинството.

Заключение

Публикации

Авторът разполага със следните публикации, които са част от текста на дисертацията:

Глава 1.1.

[1] Dimiter Dobrev. A Definition of Artificial Intelligence. *In: Mathematica Balkanica, New Series, Vol. 19, 2005, Fasc. 1-2, pp.67-74.*

[2] Dimiter Dobrev. AI - What is this. *PC Magazine - Bulgaria, November'2000, pp.12-13* (in Bulgarian, also in English at http://dobrev.com/AI/definition.html).

Глава 1.2.

[3] Dimiter Dobrev. The IQ of Artificial Intelligence. *Jun, 2018, arXiv:1806.04915 [cs.AI]*.

Глава 2.1.

[4] Dimiter Dobrev. Giving the AI definition a form suitable for engineers. *December, 2013, arXiv:1312.5713 [cs.AI]*.

Глава 2.2.

[5] Dimiter Dobrev. How does the AI understand what's going on. *International Journal "Information Theories and Applications", Vol. 24, Number 4, 2017, pp.345-369.*

Глава 2.3.

[6] Dimiter Dobrev. Minimal and Maximal Models in Reinforcement Learning. *August, 2018, viXra:1808.0589.*

Глава 2.4.

[7] Dimiter Dobrev. Event-Driven Models. *November, 2018, viXra:1811.0085.*

Глава 2.5.

[8] Dimiter Dobrev. The Definition of AI in Terms of Multi Agent Systems. *December, 2008, arXiv:1210.0887 [cs.AI]*.

Глава 3.1.

[9] Dimiter Dobrev. AI Should Not Be an Open Source Project. *May, 2018, arXiv:1806.03250 [cs.CY]*.

Авторът разполага със следните публикации, свързани с дисертацията, които не са част от текста на дисертацията:

[10] Dimiter Dobrev. First and oldest application. *1993.* http://dobrev.com/AI/first.html
[11] Dimiter Dobrev. AI - How does it cope in an arbitrary world. *In: PC Magazine - Bulgaria, February'2001, pp.12-13 (in Bulgarian, in English at http://dobrev.com/AI/world.html).*

[12] Dimiter Dobrev. Testing AI in one Artificial World. *Proceedings of XI International Conference "Knowledge-Dialogue-Solution", June 2005, Varna, Bulgaria, Vol.2, pp.461-464.*
[13] Dimiter Dobrev. AI in Arbitrary World. *Proceedings of the 5th Panhellenic Logic Symposium, July 2005, University of Athens, Athens, Greece, pp.62-67.*
[14] Dimiter Dobrev. Formal Definition of Artificial Intelligence. *International Journal "Information Theories & Applications", vol.12, Number 3, 2005, pp.277-285.*
http://www.dobrev.com/AI/
[15] Dimiter Dobrev. Parallel between definition of chess playing program and definition of AI. *International Journal "Information Technologies & Knowledge ", vol.1, Number 2, 2007, pp.196-199.*
[16] Dimiter Dobrev. Two fundamental problems connected with AI. *Proceedings of Knowledge - Dialogue - Solution 2007, June 18 - 25, Varna, Bulgaria, Volume 2, p.667.*
[17] Dimiter Dobrev. Second Attempt to Build a Model of the Tic-Tac-Toe Game. *June'2008 (represented at KDS 08), published in IBS ISC, Book 2, p.146*.
[18] Dimiter Dobrev. Comparison between the two definitions of AI. *January, 2013, arXiv:1302.0216 [cs.AI]*.
[19] Dimiter Dobrev. Incorrect Moves and Testable States. *International Journal "Information Theories and Applications", Vol. 24, Number 1, 2017, pp.85-90.*

Декларация

Авторът декларира, че дисертацията е оригинална научна разработка. Използването на предходни резултати е отразено с подходящи препратки.

Литература

[1] Dimiter Dobrev. A Definition of Artificial Intelligence. *In: Mathematica Balkanica, New Series, Vol. 19, 2005, Fasc. 1-2, pp.67-74.*
[2] Dimiter Dobrev. AI - What is this. *PC Magazine - Bulgaria, November'2000, pp.12-13* (in Bulgarian, also in English at http://dobrev.com/AI/definition.html).
[3] Dimiter Dobrev. The IQ of Artificial Intelligence. *Jun, 2018, arXiv:1806.04915 [cs.AI]*.
[4] Dimiter Dobrev. Giving the AI definition a form suitable for engineers. *December, 2013, arXiv:1312.5713 [cs.AI]*.
[5] Dimiter Dobrev. How does the AI understand what's going on. *International Journal "Information Theories and Applications", Vol. 24, Number 4, 2017, pp.345-369.*
[6] Dimiter Dobrev. Minimal and Maximal Models in Reinforcement Learning. *August, 2018, viXra:1808.0589.*
[7] Dimiter Dobrev. Event-Driven Models. *November, 2018, viXra:1811.0085.*
[8] Dimiter Dobrev. The Definition of AI in Terms of Multi Agent Systems. *December, 2008, arXiv:1210.0887 [cs.AI]*.
[9] Dimiter Dobrev. AI Should Not Be an Open Source Project. *May, 2018, arXiv:1806.03250 [cs.CY]*.
[10] Dimiter Dobrev. First and oldest application. *1993.* http://dobrev.com/AI/first.html
[11] Dimiter Dobrev. AI - How does it cope in an arbitrary world. *In: PC Magazine - Bulgaria, February'2001, pp.12-13 (in Bulgarian, in English at http://dobrev.com/AI/world.html).*
[12] Dimiter Dobrev. Testing AI in one Artificial World. *Proceedings of XI International Conference "Knowledge-Dialogue-Solution", June 2005, Varna, Bulgaria, Vol.2, pp.461-464.*
[13] Dimiter Dobrev. AI in Arbitrary World. *Proceedings of the 5th Panhellenic Logic Symposium, July 2005, University of Athens, Athens, Greece, pp.62-67.*
[14] Dimiter Dobrev. Formal Definition of Artificial Intelligence. *International Journal "Information Theories & Applications", vol.12, Number 3, 2005, pp.277-285.* http://www.dobrev.com/AI/
[15] Dimiter Dobrev. Parallel between definition of chess playing program and definition of AI. *International Journal "Information Technologies & Knowledge ", vol.1, Number 2, 2007, pp.196-199.*
[16] Dimiter Dobrev. Two fundamental problems connected with AI. *Proceedings of Knowledge - Dialogue - Solution 2007, June 18 - 25, Varna, Bulgaria, Volume 2, p.667.*
[17] Dimiter Dobrev. Second Attempt to Build a Model of the Tic-Tac-Toe Game. *June'2008 (represented at KDS 08), published in IBS ISC, Book 2, p.146.*
[18] Dimiter Dobrev. Comparison between the two definitions of AI. *January, 2013, arXiv:1302.0216 [cs.AI]*.
[19] Dimiter Dobrev. Incorrect Moves and Testable States. *International Journal "Information Theories and Applications", Vol. 24, Number 1, 2017, pp.85-90.*
[20] Alan Turing. Computing machinery and intelligence. *Mind, 1950.*
[21] John McCarthy. What is Artificial Intelligence? November, 2007. (www-formal.stanford.edu/jmc/whatisai/)
[22] Huazheng Wang, Fei Tian, Bin Gao, Jiang Bian, Tie-Yan Liu. Solving Verbal Comprehension Questions in IQ Test by Knowledge-Powered Word Embedding. *arXiv: 1505.07909, May, 2015.*
[23] Detterman, Douglas K. A Challenge to Watson. *Intelligence, v39 n2-3 p77-78 Mar-Apr 2011.*
[24] Feng Liu, Yong Shi, Ying Liu. Intelligence Quotient and Intelligence Grade of Artificial Intelligence. *arXiv: 1709.10242, September, 2017.*

[25] Dowe, D.L., & Hernández-Orallo, J. IQ tests are not for machines, yet. *Intelligence (2012), doi:10.1016/j.intell.2011.12.001*
[26] Hernández-Orallo, J., & Minaya-Collado, N. (1998). A formal definition of intelligence based on an intensional variant of Kolmogorov complexity.
Proc. intl symposium of engineering of intelligent systems (EIS'98), February
1998, La Laguna, Spain (pp. 146–163). : ICSC Press.
[27] Insa-Cabrera, J., Dowe, D. L., & Hernandez-Orallo, J. (2011). Evaluating a reinforcement learning algorithm with a general intelligence test. *In J. M. J. A. Lozano, & J. A. Gamez (Eds.), Current topics in artificial intelligence. CAEPIA 2011. : Springer (LNAI Series 7023).*
[28] 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.*
[29] Tatiana Kosovskaya. Self-Modificated Predicate Networks. *International Journal "Information Theories and Applications", Vol. 22, Number 3, 2015, pp.245-257.*
[30] Richard Sutton and Andrew Barto. Reinforcement Learning: An Introduction. *MIT Press, Cambridge, MA, 2017. ISBN 0-262-19398-1.*
[31] Dimitri Bertsekas and John Tsitsiklis. Neuro-Dynamic Programming. *Nashua, NH: Athena Scientific, 1996. ISBN 1-886529-10-8.*
[32] Richard Sutton (2008). Fourteen Declarative Principles of Experience-Oriented Intelligence. www.incompleteideas.net/RLAIcourse2009/principles2.pdf
[33] Johan Åström. (1965). Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications. 10: 174–205.*
[34] Apostolos Burnetas, Michael Katehakis. (1997). Optimal Adaptive Policies for Markov Decision Processes. *Mathematics of Operations Research. 22 (1): 222.*
[35] Goranko V. Доклад изнесен във ФМИ, СУ, София, лятото на 2009 г.
[36] Goranko V. Coalition Games and Alternating Temporal Logics. *Proc. of the 8th Conference on Theoretical Aspects of Rationality and Knowledge (TARK VIII, Siena, Italy, 8-10 July, 2001), J. van Benthem (ed.), Morgan Kaufmann, 2001, pp. 259-272.*
[37] Emmanuel Macron (2018). Emmanuel Macron Talks to WIRED About France's AI Strategy. *31 of March, 2018, www.wired.com/story/emmanuel-macron-talks-to-wired-about-frances-ai-strategy*
[38] Henri Becquerel (1896). "Sur les radiations émises par phosphorescence". Comptes Rendus. 122: 420–421.
[39] Adorno, Theodor W. and Horkheimer, Max. Dialectic of Enlightenment. Stanford University Press, 2002, ISBN: 9780804736336.
[40] Isaac Asimov (1950). I, Robot (The Isaac Asimov Collection ed.). New York City: Doubleday. ISBN 0-385-42304-7.
[41] James Cameron (1984). The Terminator. American science-fiction action film.
[42] The Wachowski Brothers (1999). The Matrix. Science fiction action film.