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 отчисляване с право на защита

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

Научен ръководител: доц. д-р Любомир Иванов

София 2022

Дисертацията съдържа 119 страници, от които 6 страници уводни раздели, 107 страници основен текст и 6 страници заключителен материал, включително библиография с 57 заглавия.

Съдържание

Увод

Тази дисертация е посветена на трите основни въпроса свързани с Изкуствения Интелект. Първият въпрос е „Какво представлява този обект?“ Следващият въпрос, който ще си зададем е „Как да създадем програмата ИИ?“. Последния и най-важен въпрос е за бъдещето, за последствията, за отговорността на учения и затова дали не е глупаво да създаваме машина, която ще е по-умна от нас самите.

Първият въпрос е свързан с дефиницията на ИИ. Този въпрос е важен от философска, от практическа, но и от математическа гледна точка. В математиката много често ни се случва да си дефинираме някакъв нов обект и да докажем, че този нов обект притежава най-различни интересни свойства. Проблем е, когато не можем да докажем съществуването на този обект и най-големият проблем се появява, когато докажем, че такъв обект не съществува. От последното тривиално следват всичките интересни и всичките безинтересни свойства на обекта. Както обичам да казвам на моите студенти: „За елементите на празното множество можем да кажем всичко, и всичко ще е вярно, защото никой не може да ни опровергае, като посочи елемент на празното множество, за който нещо да е лъжа.“

Дефиницията на ИИ, която се дава в тази дисертация гарантира, че такъв обект съществува. Ние ще дефинираме ИИ като програма, която е по-умна от човек. Първо ще кажем какво значи една програма да е по-умна от друга. Това ще го направим като въведем IQ (коефициент на интелигентност). Този коефициент ще бъде резултатът от един изпит (ще го наречем IQ test). Ще докажем, че има програма, която е най-умната и тя ще даде възможно най-добрия резултат на изпита. Тази програма гарантирано ще е по-умна от човек, защото самият човек може да бъде емулиран с компютърна програма. Това е според тезиса на Чърч, (Church, 1941). Тази, най-умната програма формално отговаря на дефиницията на ИИ, но не върши работа, защото е твърде тромава (нуждае се от безкрайно бърз компютър). Затова ще добавим изискването за ефективност и ще търсим програма, която е по-умна от човек, но е достатъчно ефективна, за да работи на реален компютър (такъв компютър какъвто можете да си купите).

Изискването за ефективност поставя под въпрос съществуването на ИИ. Има ли програма, която е по-умна от човек и е достатъчно ефективна, за да може да работи на вашия компютър? Отговорът на този въпрос зависи от това какъв компютър можете да си позволите и дали въобще има такъв компютър (дали такъв компютър въобще съществува). Дори и такъв компютър да няма в момента, предполагаме, че в бъдеще ще има, защото компютрите стават все по-бързи и все по-мощни.

Най-доброто доказателство за съществуването на един обект е неговото директно построяване. Тук стигаме до втория от основните въпроси свързани с ИИ. Това е въпросът: „Как да построим програма, която е по-умна от човек, тоест която в произволен свят се справя не по-зле от човек.“ За да се справи добре ИИ в един свят, той трябва да разбере света, да го разбере значи да го опише. За да опишете света вие се нуждаете от език за описание на светове. Това трябва да е такъв език, че описанията на световете, които са написани на този език, трябва да мога да се търсят автоматично. Точно с въпроса за създаването на този език се занимаваме в тази дисертация.

Създаването на език за описание на светове не е достатъчно за създаването на ИИ. Трябва още да се създаде програма, която може да намери описанието на света написано на този език и чрез това описание да може да планира поведението си така, че да оптимизира постигането на някакви цели. Разбира се, трябва да зададем и цел, защото за да бъде смислено едно поведение, то трябва да преследва някаква цел. В тази дисертация ние стигаме само до езика за описание на светове и не продължаваме нататък. Не търсим описания написани на този език и не планираме бъдещото поведение на устройството. Също така в тази дисертация ние не фиксираме конкретна цел, която трябва да бъде преследвана от ИИ. Нашата концепция е, че ако разберем света, ние можем да си поставяме и да постигаме различни цели. Тоест, въпросът за поставената цел е второстепенен и може да се остави за накрая, защото разбирането на света не зависи от целта, която ще си поставим.

Третият и най-важен въпрос е свързан с последствията от създаването на ИИ. Този въпрос много малко хора си го задават. Хората са като любопитни деца, които обичат да си играят с кибрит без да мислят за това, че може да предизвикат пожар. Макар и малко, все пак има научни статии, които обсъждат бъдещето, което ще дойде след създаването на ИИ. Пример за такава статия е Alfonseca et al. (2021).

Основният въпрос е: „Трябва ли да създаваме ИИ и можем ли да не го създаваме?“ Отговорът е, че не можем да не го създаваме, защото ако ние не го създадем, ще го създаде някой друг. По-добре е да помислим и да се подготвим за последствията и все пак да внимаваме да не изтървем духа от бутилката.

Отговорът на третия въпрос ще ни е нужен чак когато вече сме направили ИИ. 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).

Приноси на дисертацията:

1. Предлага се неформална дефиниция на ИИ и тази неформална дефиниция днес е приета от много хора (двете статии Dobrev (2000) и Dobrev (2005a), в които тези неформална дефиниция се представя имат общо 53 цитирания). Първата от тези две статии е включена като част от тази дисертация.
2. Дава се строга математическа дефиниция на ИИ. Статията Dobrev (2005b), в която тази строга дефиниция е дадена първоначално има 15 цитирания. Подобрен вариант на тази статия е Dobrev (2019d), която е включена като част от тази дисертация. По отношение на строгата дефиниция на ИИ не сме първи, защото в статията Hernández-Orallo et al. (1998) в голяма степен това вече е направено, но предложената от нас дефиниция е съществено подобрена и изчистена от грешки и проблеми и затова може да се приеме за самостоятелен принос.
3. Въвежда се език за описание на светове, при който описанието може да се търси автоматично без помощта на човек. Има и други езици за описание на светове, които са създадени преди нашия език, но предимство на предложения от нас език е, че при него описанието на света може да се търси автоматично, докато при другите подходи се предполага, че има човек, който е разбрал света и който ще го опише на съответния език.

Структура на дисертацията:
Тази дисертация се състои от три части, като всяка от тези части е посветена на един от трите основни въпроси свързани с ИИ.

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 Dobrev (2000) и Dobrev (2019d).

Втората част разглежда въпроса „Как да направим ИИ?“. Този част по същество се състои от статията Dobrev (2020c).

Третата част се състои от две глави. Първата е статията Dobrev (2019c), а втората глава предстои да бъде допълнена и издадена като отделна статия.

1 Какво е Изкуствен Интелект?

1.1 Неформална дефиниция

В тази статия ще си зададем въпросите: "Трябва ли ни да знаем какво е AI?" и "Какво е интелект?". След това ще дадем една дефиниция на Изкуствен Интелект. Накрая от тази дефиниция ще получим алгоритъм, който след краен брой стъпки ще открие AI.

1.1.1 AI - Какво е това?

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.

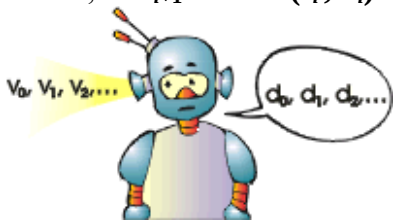
Тюринговия тест е доста прост. Поставяме нещо зад една завеса и то разговаря с нас. Ако не можем да го различим от човек, то това е AI. Тази дефиниция е по-стара от петдесет години и затова ще се опитаме да направим нова, по-съвременна.

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

1.2.1 IQ-то на Изкуствения Интелект

За да кажем кои програми са AI е достатъчно да проведем един изпит и да признаем за AI тези програми, които са издържали изпита. Оценката от изпита ще наречем IQ. Не можем да кажем, колко точно е минималното IQ, за да бъде една програма AI, но ще изберем една конкретна стойност. Така нашата дефиниция за AI ще бъде всяка програма, чието IQ е над определената стойност. Тази идея вече е реализирана в Dobrev (2005b), но тук ще повторим тази конструкция, като внесем някои подобрения.

1.2.2 Въведение

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 Dobrev (2005b) 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 Dobrev (2005b) се нуждае от подобрение, са следните:

1. В Dobrev (2005b) се разглеждат едновременно въпросите „Какво е AI?“ и „Как можем да създадем AI?“ Не е добра идея да се обърква 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 Dobrev (2005b) дефинираме AI като програма, а тук ще дефинираме AI като стратегия. В Dobrev (2005b) 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 Dobrev (2005b) 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 Dobrev (2005b) 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 Dobrev (2013). Света е достатъчно сложен и е трудно да го разберем. Ако добавим едно излишно кодиране, света ще стане още по-неразбираем.

5. В Dobrev (2005b) 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 Dobrev (2005b), 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 Dobrev (2005b) 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 Dobrev (2005b) 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).

1.2.3 Литературен обзор

Тюринг в Turing (1950) 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 Turing (1950) 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 McCarthy (2007) 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 Dobrev (2005b) 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 Wang et al., 2015) 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 Detterman (2017) 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 Dobrev (2005b) 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 Liu et al. (2017) 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 Liu et al. (2017) 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 Dowe and Hernández-Orallo (2012) се прави много сериозно обсъждане на въпроса за това как се мери IQ на AI. Също така, в Dowe et al. (2012) е направен много добър обзор на различните работи посветени на тази тема. Известен недостатък на Dowe et al. (2012) 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 (Hernández-Orallo et al. (1998) and Insa-Cabrera, Dowe, & Hernandez-Orallo (2011)) 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 Dowe et al. (2012) 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 Dowe et al. (2012), както и в Liu et al. (2017), се търси универсално IQ, което да е приложимо към всички програми и дори и към хората. Тук се разминаваме с авторите на Liu et al. (2017) и на Dowe et al. (2012). 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 Hernández-Orallo et al. (1998), 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 Hernández-Orallo et al. (1998) 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 Hernández-Orallo et al. (1998) в Dobrev (2005b). Това е един пропуск, който сега коригираме.

Въпреки сериозността и задълбочеността на Hernández-Orallo et al. (1998), там са допуснати някои неточности, които не можем да подминем. Това, че обръщаме внимание на някои пропуски и неточности на Hernández-Orallo et al. (1998) 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 Hernández-Orallo et al. (1998) 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 Hernández-Orallo et al. (1998) 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 Dobrev (2007b) 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 Hernández-Orallo et al. (1998) 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 Hernández-Orallo et al. (1998) е малко по-специален. Той или разбира света изцяло или въобще не го разбира. Тоест, наблюдател, който работи на принципа „всичко или нищо“.

В Hernández-Orallo et al. (1998) 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 Hernández-Orallo et al. (1998) полагат сериозни усилия да направят зависимостите exception-free. Много интересно е определението за exception-free зависимости, което се дава в Hernández-Orallo et al. (1998). 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 Hernández-Orallo et al. (1998) наблюдател залага на принципа „всичко или нищо“.

Имаме още няколко дребни препоръки към Hernández-Orallo et al. (1998).

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 Hernández-Orallo et al. (1998) 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 Hernández-Orallo et al. (1998), милион

пъти повече време. Нашата препоръка е зависимостите да се търсят след всяка стъпка, а не само след последната.

В Hernández-Orallo et al. (1998) не ни харесва още това, че програмата, която генерира теста не работи поради комбинаторна експлозия. Това е програмата наречена „The Generator“. Тоест, в Hernández-Orallo et al. (1998) 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 Hernández-Orallo et al. (1998) 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 Hernández-Orallo et al. (1998) е локалното IQ, а не глобалното. Все пак, да не забравяме, че локалното IQ клони към глобалното, но в Hernández-Orallo et al. (1998) нищо не се казва за програмата „зубрач“, която е основния проблем на локалното IQ.

Как да коригираме дефиницията от Hernández-Orallo et al. (1998), 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 Hernández-Orallo et al. (1998) е, че програмата, която дефинира е всъщност програмата Dobrev (1993). 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 Hernández-Orallo et al. (1998) не е по-добра от Dobrev (1993). 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 Hernández-Orallo et al. (1998) 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 Insa-Cabrera, Dowe, & Hernandez-Orallo (2011)) той явно е разбрал основния пропуск на Hernández-Orallo et al. (1998) 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 Insa-Cabrera, Dowe, & Hernandez-Orallo (2011) 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 Insa-Cabrera, Dowe, & Hernandez-Orallo (2011) 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 Dobrev (2000), 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 Dobrev (2005b), 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 Insa-Cabrera, Dowe, & Hernandez-Orallo (2011) 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 Insa-Cabrera, Dowe, & Hernandez-Orallo (2011), 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 Dobrev (2005b) и в тази статия е да няма „коефициент на обезценка“, но живота да е ограничен. Казано с други думи, подхода тук и в Dobrev (2005b) е коефициента на обезценка да е едно до един момент (края на живота) и да бъде нула от този момент нататък.

1.2.4 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 Dobrev (2000) и в Dobrev (2005b) 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.

1.2.5 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.

1.2.6 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.

1.2.7 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.

1.2.8 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.

1.2.9 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.

1.2.10 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.

1.2.11 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 Dobrev (2005b) 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.

1.2.12 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. Тава, 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.

1.2.13 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“.

1.2.14 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.

1.2.15 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 падне ези избираме с вероятност 1/2 числото 0, в противен случай отново хвърляме монета и ако се падне ези избираме с вероятност 1/2 числото 1 и т.н. Когато стигнем до k, ако не се е паднало ези 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.

1.2.16 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).

1.2.17 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 Dobrev (2005b). Тази стойност и тук и в Dobrev (2005b) 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.

1.2.18 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 Dobrev (2007a), но тъй като настоящата статия повтаря и подобрява конструкцията описана в Dobrev (2005b), тук ще повторим и конструкцията от Dobrev (2007a), 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.

1.2.19 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 Език за описание на светове

Ще сведем задачата за създаването на ИИ към задачата за намирането на подходящия език за описание на света. Този език няма да е език за програмиране, защото езиците за програмиране описват само изчислими функции, докато този език ще опише малко по-широк клас от функции. Друга особеност на този език ще е, че при него описанието ще може да бъде разбито на отделни модули. Това ще ни позволи да търсим описанието на света автоматично, като го откриваме модул по модул. Подходът ни за създаването на този нов език ще бъде да започнем от един конкретен свят и да напишем описанието на тази конкретен свят. Идеята ни е, че езикът, който може да опише този конкретен свят, ще е подходящ за описанието на произволен свят.

2.2 Въведение

Тази статия представя един нов подход в изследването на ИИ. Това е Event-Driven (ED) подходът. Идеята, която стои зад ED подхода е, че моделът не трябва да поема цялата входно-изходна информация, а трябва да отрази само важните събития.

Всяко действие е събитие. Всяко наблюдение също е събитие. Ако моделът отразява всички действия и всички наблюдения, то той ще се претовари с прекалено много информация. Ако моделът се ограничи само до няколко „важни“ събития, тогава това претоварване ще бъде избегнато. По този начин стигаме до идеята за Event-Driven моделите.

Недостатък (или може би предимство) на ED модела е, че той не описва света напълно, а го описва само частично. По-точно ED моделът описва клас от светове (това са световете изпълняващи определена зависимост).

С какво тази статия е различна. Обикновено, когато се разглежда мулти-агентна система се предполага, че светът е даден, а това което се търси е стратегия. Тоест, светът е част от известните неща в условието на задачата, а стратегията е неизвестното. В тази статия ще предполагаме, че светът не е даден, а той е това, което се търси.

Обикновено, когато се предполага, че светът е даден, тогава предполагаме, че ни е дадена една релация, която описва света напълно. Тук ще се опитваме да опишем света и ще го описваме частично чрез ED модели.

Структура на езика. Описанието на света няма да прилича на хомогенна система състояща се от един единствен пласт. По-скоро описанието на света ще има структура състояща се от много различни пластове. На фигура 1 представяме тази структура като пирамида, където първият пласт е основата на пирамидата.

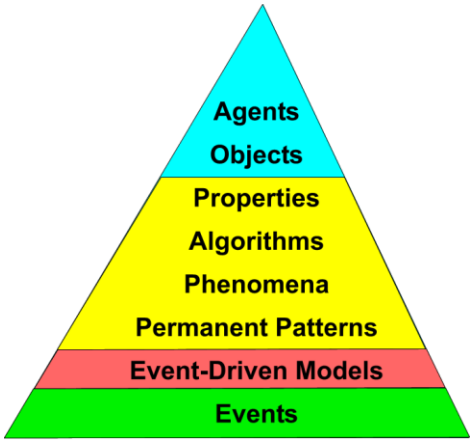


Figure 1

Първият пласт това ще са събитията. С тях ще опишем Event-Driven моделите. С тези модели ще описваме зависимости. Важно е състоянията на ED моделите да не са еднакви. За да се различават състоянията трябва в поне едно от тях нещо специално да се случва. Това специалното нещо ще го наречем особеност. Множеството от особеностите ще го наречем „следа“.

Първата ни идея за следа е това да е нещо постоянно. (Например, да имаме едно състояние, където винаги е студено.) Следващата ни идея е подвижната следа, тоест специалното поведение може да се появи и да изчезне или да се премести. (Например, студа да се премести в съседната стая.)

Следващите пластове в пирамидата това ще са различни видове зависимости (всички зависимости ще представяме с ED модели). Първо ще сложим постоянните зависимости. Това са зависимостите, които се наблюдават през цялото време. В повечето статии се предполага, че всички зависимости са постоянни, но тук ще предположим, че освен тях имаме зависимости, които се наблюдават от време на време. Непостоянните зависимости ще наречем „явления“. Тоест, както „следата“ може да е постоянна или подвижна, така и зависимостите могат да бъдат постоянни или „явления“.

Алгоритмите също са непостоянни зависимости (наблюдават се докато алгоритъма се изпълнява). Ще представим алгоритъма като последователност от събития. Обикновено алгоритмът се разглежда като последователност от действия, защото се предполага, че главният герой е този, който го изпълнява. В тази статия алгоритмите ще са някакви зависимости и ако някой от агентите действа така, че да запази зависимостта ще считаме, че агентът изпълнява алгоритъма.

Когато едно явление е свързано с наблюдението на обект, това явление ще го наречем „свойство“. По този начин стигаме до абстракция от по-високо ниво. Това е абстракцията за „обект“. Обектите не ги наблюдаваме директно, а ги засичаме благодарение на техните свойства.

Следващата абстракция, до която ще стигнем, това са агентите. Тях също не можем да ги наблюдаваме директно, но можем да ги засечем благодарение на техните действия. За да опишем света, трябва да опишем агентите, които живеят в него и да кажем какво знаем за

тях. Най-важното е дали са ни приятели и дали с действията си ще се опитват да ни помогнат или да ни попречат.

Дотук описваме изчислими светове (такива, които могат да се емулират с компютърна програма). Ако вътре в света има неизчислим агент, това ще направи света неизчислим, но има и друг начин да опишем неизчислим свят. Можем да добавим правило зависещо от това дали съществува някакъв алгоритъм (по-точно съществува ли изпълнение на този алгоритъм). Въпросът дали съществува изпълнение на алгоритъм е неизчислим (halting problem, Turing (1937)).

Приноси.

1. Event-Driven модел. (Това понятие вече е въведено в Dobrev (2018), но там не е дадена интерпретация на ED модела. Именно интерпретацията е това, което дава смисъл на модела и това което разграничава адекватните от неадекватните модели.)
2. Показваме, че Markov decision process (MDP) е частен случай на ED модел и че ED моделът е естественото обобщение на MDP.
3. Simple MDP. Опростяваме MDP и получаваме по-прост модел, който описва повече светове.
4. Разширен модел. Това е моделът, при който състоянието знае всичко. Чрез този модел ние ще въведем интерпретация на събитията и на Event-Driven моделите. (Разширеният модел е въведен в Dobrev (2019a), но там състоянието знае само какво се е случило и какво ще се случи, но не знае какво е възможно да се случи. Тоест, в Dobrev (2019a) състоянието на разширения модел не знае за пропуснатите възможности. В Dobrev (2019a), разширеният модел се нарича „максимален“.)
5. Дефиниция на понятието алгоритъм. Представяме алгоритъма като последователност от събития в произволен свят. Представяме машината на Тюринг като ED модел, който се намира в един специален свят, в който имаме безкрайна лента. По този начин показваме, че новата дефиниция обобщава понятието „Машина на Тюринг“ и разширява понятието алгоритъм.
6. Език за описание на светове, при който описанието може да се търси автоматично без помощта на човек.

Структура на статията. Първо (параграф 2) ще кажем кой е конкретният свят, който ще опишем. После (параграф 3) ще покажем, че досега известните инструменти за описание на светове не са подходящи за този свят.

В параграфи от 4 до 8 ще дадем теоретичната основа на статията. Ще дефинираме понятията жизнен опит, живот, Event-Driven модел, събитие и свят. Ще дефинираме интерпретация, която ще даде смисъл на тези понятия. Ще покажем, че MDP е частен случай на ED модел.

В параграфи от 9 до 13 ще направим конкретно описание на света, в който агентът играе играта шах:

В параграф 9 ще опишем няколко прости зависимости, които ще представим чрез ED модели (например зависимостите Horizontal и Vertical).

В параграф 10 ще кажем как се движат фигурите. За целта ще ни се наложи да разширим понятието алгоритъм. Алгоритмите за движение на фигурите също ще представим чрез ED модели.

В параграф 11 ще представим шахматните фигури като обекти. Обектите ще са абстракция от по-високо ниво. Свойствата ще са това, което е конкретно (което се дефинира чрез ED модели) и което ще определя обектите.

В параграф 12 ще добавим втори играч. За целта ще трябва да направим още една абстракция от по-високо ниво. Това ще е абстракцията „агент“. Агентите няма да ги наблюдаваме директно, а чрез техните действия. Агентите ще направят света неизчислим, но и без агенти светът може да стане неизчислим, ако добавим неизчислимо правило.

Накрая в параграф 13 ще разгледаме агентите и различни взаимодействия между тях.

2.3 Играта шах

Кой ще е конкретният свят, който ще използваме за да създадем новия език за описание на светове? Това ще е светът на играта шах.

Първо ще отбележим, че ще искаме светът да е partially observable, защото, ако агентът вижда всичко, светът не е интересен. Ако вижда всичко, на агента няма да му е нужна фантазия. Най-важното за агента е да си представи тази част от света, която той не вижда в текущия момент.

За да бъде светът partially observable, ще предположим, че агентът не вижда цялото табло, а само едно квадратче от таблото (фигура 2). Окото на агента ще се намира в квадратчето, което вижда в момента, но той ще може да мести окото си и по този начин ще може да огледа цялото табло. Формално погледнато няма разлика между това дали виждате цялото табло или виждате само едно квадратче, но можете да местите поглед и да огледате всичко. Няма разлика, ако знаете, че местейки поглед оглеждате цялото табло. На практика агентът не знае нищо и той ще трябва да си изгради представата за цялото табло, а това няма да е прост процес и ще изисква фантазия.

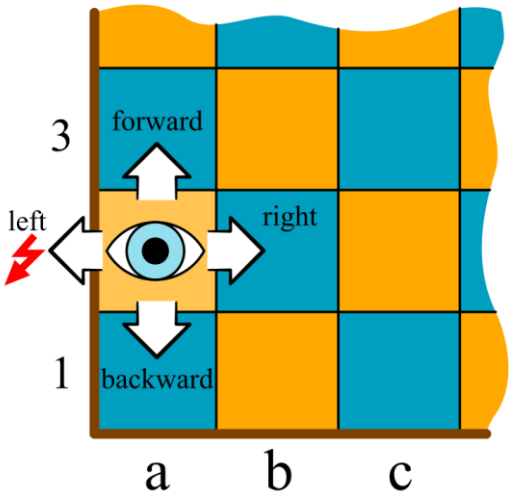


Figure 2

На фигура 2 окото на агента се намира в квадратчето **a2** и агентът може да го мести в четирите посоки. (В този момент не може да го премести наляво, защото е в края на таблото и ходът наляво е некоректен.) Освен местенето на окото в четирите посоки, агентът има още две действия и това са „вдигни фигурата, която виждаш“ и „спусни вече вдигнатата фигура в квадратчето, което виждаш“. Тези две действия ще означим с up и down. При помощта на тези шест действия агентът може да огледа таблото и да премести фигура, а това е всичко, което му е нужно, за да играе шах.

2.3.1 Шах с един играч

Ще разгледаме два варианта на играта шах. Ще разгледаме шах с един играч и шах с двама играчи.

Какво означава шах с един играч? Това означава, че играем с белите, обръщаме дъската и играем с черните и т.н.

Първо ще опишем по-простия вариант, когато агентът е сам и играе сам срещу себе си. Това е по-простият вариант, защото в този свят има само един агент и това е главният герой. По-долу ще разгледаме и по-сложния вариант, при който в света има и втори агент, който е противник на главния герой.

Въпросът е, каква ни е целта, когато играем сами срещу себе си?

2.3.2 Цел

В повечето статии за ИИ се избира цел, но в тази статия няма да фиксираме конкретна цел. Ние искаме само да опишем света, а когото сме разбрали света бихме могли да си поставим различни цели. Например при играта шах може целта ни е да спечелим или да загубим. Когато играем сами срещу себе си може целта ни да се променя. Може когато играем с белите целта ни да е „да спечелят белите“ и обратното.

Разбирането на света не е пряко свързано с поставянето на конкретна цел. Естественият интелект (човекът) обикновено няма ясно дефинирана цел, но това не му пречи да живее.

Има два въпроса: „Какво става?“ и „Какво да правя?“. Повечето статии за ИИ се опитват директно да отговарят на втория въпрос без да отговорят на първия. Тоест, обикновено директно се търси стратегия, а за да има стратегия трябва да има цел. В тази статия се търси отговор само на първия въпрос, а вторият въпрос въобще не се разглежда. Тоест, в тази статия не ни е нужна цел.

В повечето статии целта се дефинира чрез rewards (целта е да се съберат повече награди). Когато говорим за Markov decision process (MDP) ще предполагаме, че от дефиницията са махнати наградите, защото те са ни нужни само, ако ще търсим стратегия, а ние няма да го правим.

2.4 Related work

Може ли светът на играта шах да бъде описан при помощта на вече известните инструменти за описание на светове? Ще разгледаме известните до момента инструменти за описание на светове и ще покажем, че те не са подходящи.

2.4.1 Markov decision process

За описание на светове най-разпространения инструмент е Markov decision process (MDP). Може ли избрания от нас свят да бъде описан чрез MDP? Първо ще отбележим, че ще

трябва да използваме Partially observable MDP (POMDP), защото светът, който искаме да опишем е Partially observable.

Разбира се, този свят може да се представи като POMDP, но колко състояния ще са нужни? Ще ни трябват толкова състояния, колкото са позициите на шахматната дъска, а това е ужасно много (някъде около 10^{45} според Tromp (2021)). Ще ни трябват дори малко повече състояния, защото състоянието ще трябва да помни освен позицията на дъската и координатите на окото. Тук 64 пъти повече е малко повече, защото при толкова големи числа добавянето на още две нули към числото изглежда незначително увеличение. Тоест, числата 10^{45} и 10^{47} ни изглеждат близки.

Ако искаме да опишем подобен POMDP като таблица, тогава това описание ще е толкова огромно, че паметта на никой компютър няма да е в състояние да го събере. Разбира се, съхранението на описанието е най-малкият проблем. Много по-сериозен проблем е, че ние тази таблица би трябвало да я намерим и да я построим на базата на жизнения си опит, а за толкова голяма таблица ще ни е нужен на практика безкраен жизнен опит.

Тоест, идеята да търсим описание на този свят под формата на POMDP е обречена на неуспех.

2.4.2 Situation Calculus

Първото предложение за език за описание на светове е направено от Raymond Reiter и това е неговият Situation Calculus описан в Reiter (2001).

Светът на играта шах може да се представи чрез формализма предложен от Raymond Reiter, но има два проблема (малък проблем и голям проблем).

Първият (малкият) проблем е, че Reiter представя състоянието на света получено след действие чрез функционален символ. Тоест, той предполага, че следващото състояние е еднозначно определено. Това предположение не е проблем за детерминистични светове, какъвто е светът на играта шах, но би било проблем за игри със зарове. Разбира се, това е малък проблем, защото можем да предполагаме, че следващото състояние е определено, но ние не знаем точно кое е то. (Тоест, можем да предположим, че има съдба, която еднозначно определя бъдещето, макар че това не ни помага да предскажем какво ще се случи.)

Един опит за решаването на първия (малкия) проблем е направен в параграф 3.2. на Boutilier, Reiter and Price (2001). Там всяка стъпка е заменена с две стъпки (plies). Вместо една стъпка на агента имаме ply на агента и ply на nature. Идеята е, че стъпката на агента е недетерминирана, защото светът (природата) може да отговори по много различни начини, но, ако разделим действието на агента от отговора на природата, тогава резултата от всяка стъпка (ply) е детерминиран. Ние използваме по същество същата идея, когато създаваме Simple MDP (по-долу в статията).

Вторият (големият) проблем на Situation Calculus е, че Reiter негласно предполага, че има човек (програмист), който е разбрал устройството на света и който ще го опише чрез формули от първи ред. Всички ние искаме да стигнем до описание на света чрез формули от първи ред, но целта е това описание да може да бъде намерено автоматично без намесата на човек. Вярно е, че в настоящата статия се дава описание на играта шах, което е

направено от човек, но целта ни е това описание да се търси автоматично и даденото тук описание е такова, което би могло да се търси и намери автоматично.

2.5 Жизнен опит

Задачата е да опишем света на базата на жизнения си опит. Тоест, търсим модел обясняващ най-добре това, което се е случило до момента. Затова първият въпрос, който ще си зададем, е: „Какво е жизнен опит?“

Обикновено се предполага, че имаме само един живот и жизненият опит е това, което се е случило до момента в този живот. Тук ще предположим, че имаме повече от един живот и че жизненият опит е това, което се е случило в текущия живот и във всички досега изживяни животи. Дори ще предполагаме, че може да имаме повече от един текущ живот.

Защо правим това предположение? Ако мислим от гледната точка на индивида, той има само един живот, но ако погледнем нещата от гледната точка на популацията, то тогава животите са много. В нашия случай имаме изкуствен агент, който живее в изкуствен свят. Бихме могли да пуснем агента многократно да живее в света и да получим много животи, на базата, на които да събираме наблюдения и да търсим зависимости. Бихме могли дори да пуснем много агенти, които едновременно да живеят в света. Тогава ще имаме много текущи животи едновременно.

Дефиниция: Жизнен опит ще наричаме крайна последователност от животи.

Следващият въпрос е: „Какво е живот?“

2.5.1 Живот

Животът ще бъде крайна последователност от действия и наблюдения. Това е, което виждаме, но дали в живота има и неща, които не виждаме?

Нека си представим два живота. В първия ние минаваме покрай гърне с жълтици, но не го отваряме и продължаваме нататък. Във втория живот ние минаваме покрай празно гърне, което не отваряме и пак продължаваме нататък. Двата живота като действия и наблюдения са абсолютно идентични, но все пак тези два живота са различни, защото в първия ние сме имали шанса да намерим жълтиците, а във втория живот не сме имали този шанс.

Затова в описанието на живота ще сложим две неща. Първото е това, което сме видели (последователността от действия и наблюдения). Тази последователност ще наречем „следата на живота“.

Забележка: В тази статия говорим за „следата на живота“ и за „следата на ED модела“. Това са две различни понятия.

Второто, което ще определи живота е какво е било възможно да се случи (без значение дали действително се е случило). Възможното минало и възможното бъдеще се описват от състоянието на света. Важно е какво е било състоянието във всеки момент. Затова към описанието на живота ще добавим последователността от състояния, през които е преминал светът. Ще наречем тази последователност „гръбнака на живота“.

Следата и гръбнакът описват живота от гледна точка на света. Ако погледнем на света от гледната точка на агента, животът ще се опише чрез следата и предположенията, които

може да направи агентът във всеки един момент. Например, в един момент можем да предположим, че след две стъпки ще видим определено наблюдение. (Моментът на предположението е моментът, в който можем да го направим, а не моментът, за който се отнася.)

За всяко предположение са важни две неща. Първото са предпоставките, които трябва да са налице в момента, когато правим предположението. Второто е правилото (евристиката) на базата на която предположението ще бъде направено. Ще предполагаме, че евристиката е получена от целия ни жизнен опит (включително бъдещия жизнен опит). Тоест, когато жизненият опит се увеличава се променят евристиките и оттам се променят и предположенията (дори и вече направените предположения). Например, в един момент разбираме, че шумът от изстрел е свързан с опасност. Тогава преразглеждаме миналото и установяваме, че когато сме чували изстрел е било опасно.

2.5.2 Евристика

Казахме, че в живота има неща, които ние не виждаме, но които е важно да се опитаме да предскажем. Например: „Има ли жълтици в гърнето?“ Ще се опитаме да предскажем това чрез нашето шесто чувство или чрез някакви евристики, които сме намерили от нашия жизнен опит чрез събиране на статистика. Пример за подобна евристика е: „Ако гърнето изглежда старо, то то е пълно с жълтици.“ Разбира се, евристиката си има някакъв коефициент на достоверност. Нека този коефициент да е само 2%. Дори и при този нисък коефициент на достоверност си струва да отворим гърнето и да проверим за жълтици, когато то изглежда старо.

Евристиката ще се състои от предпоставка, заключение и коефициент на достоверност. Ще предполагаме, че заключението е атомарна формула, а предпоставката е конюнкция от атомарни формули. (По-долу ще кажем какво е атомарна формула.)

Дефиниция: Евристиката ще бъде импликация с коефициент на достоверност:

$$A_1 \,\&\, A_2 \,\&\, \ldots \,\&\, A_n \Rightarrow A_0 \ \textit{(процент)}$$

Ето един пример за евристика:

$$Ob(t-1)=o_1 \,\&\, Act(t)=a \Rightarrow Ob(t+1)=o_2 \ (\ 10\%\ )$$

Тук $Ob(t)=o$ означава, че наблюдението в момента t е o, а $Act(t)=a$ означава, че действието в момента t е a. (При записа използваме равенство, защото наблюдението и действието еднозначно зависят от t.)

Събитието $Ob(t)$ има смисъл само в четните моменти, а $Act(t)$ има смисъл само в нечетните моменти, защото редуваме действие-наблюдение. Евристиката е валидна за всяко t. Затова ще предполагаме, че предпоставката е известна в момента t и че това е първият момент, в който предпоставката става известна. (Това можем да го постигнем като заменим t с $t+i$ за някое i.)

Тук заключението е свързано с бъдещо видимо събитие (няма смисъл да предсказваме минало видимо събитие). Заключението би могло да бъде свързано и с невидимо събитие (минало или бъдеще). Например:

$$Ob(t)=o_1 \Rightarrow PossNext(o_2,\ t-2) \ \textit{(някакъв процент)}$$

Тук невидимото събитие *PossNext(o, t)* означава, че е възможно следващото наблюдение да бъде *o*, т.е. възможно е *Ob(t+2)=o*. (При записа не използваме равенство, защото може да имаме повече от едно възможно следващо наблюдение, а равенството предполага, че това наблюдение е единствено.)

При $o_1=o_2$ горната евристика ще е безсмислена, защото ще е тавтологично вярна, но при $o_1 \neq o_2$ ще има смисъл, за всеки минал и бъдещ момент.

На евристиките гледаме като на правила, които са валидни винаги, но бихме могли да предположим, че едно правило е валидно само понякога. Тогава това непостоянно правило ще дефинира едно невидимо събитие. Това събитие ще е истина когато правилото е валидно и лъжа, когато не е. Нека разгледаме правилото:

$$\varnothing \Rightarrow Ob(t+2) \neq o$$

Тук предпоставката е празното множество, тоест без предпоставка или винаги. Това правило ни казва, че следващото наблюдение не може да бъде *o*. Това понякога е вярно, а понякога не е, но ние не виждаме дали това е вярно (освен в случая когато следващото наблюдение е *o*, но в общия случай не виждаме). Тоест, това е едно невидимо събитие и неговото отрицание е точно *PossNext(o, t)*.

2.5.3 Атомарна формула

Агентът не може да наблюдава всички събития и ще трябва да се ограничи до краен брой. На всяко наблюдавано събитие агентът ще даде име и тези събития ще наречем атомарни формули.

Дефиниция: Атомарна формула е събитие, на което сме му дали име.

Например *Ob(t)=o* е атомарна формула. (По-точно това са много атомарни формули, по една за всяко *o*.)

Когато наблюдаваме едно събитие, практически ние наблюдаваме и неговото отрицание. Затова отрицанието на атомарна формула също е атомарна формула.

Конюнкция от атомарни формули също може да бъде атомарна формула, ако сме решили да я наблюдаваме и сме ѝ дали име. Разбира се, тези конюнкции са безбройно много и не можем да ги наблюдаваме всичките.

От събитието *A(t)* ние можем да направим безброй събития *A(t+i)*, които ще са същото събитие, но шифтвано във времето. Безсмислено е да наблюдаваме всичките тези събития, защото те са почти еднакви. Затова ще наблюдаваме само едно от тях. Например при конюнкция от атомарни формули ще считаме, че наблюдаваме това събитие в момента, който е максимумът от моментите на атомарните формули участващи в конюнкцията.

При повечето случай няма значение точно в кой момент се случва наблюдаваното събитие, но има и изключения. Например при ED моделите е важно кога точно се случва всяко от наблюдаваните събития (дали се случва няколко стъпки по-рано или няколко стъпки по-късно).

2.5.4 Моментно събитие

Дефиниция: Моментно събитие е такова, което зависи само от един момент.

Нека моментът е *t*. Тогава моментно видимо събитие е такова, което зависи само от едно наблюдение или само от едно действие (от *Ob(t)* или от *Act(t)*). Моментно невидимо събитие е такова, което зависи само от едно от състоянията на света (s_t).

Конюнкция от моментни събития може да е моментно събитие, ако всичките атоми на конюнкцията са моментни събития и се отнасят за един и същи момент.

Видимите моментни събития са крайно много, а невидимите са толкова колкото са подмножествата от състояния на света (крайно или континуум).

Атомарната формула може да е моментно събитие, а може и да не е.

2.5.5 Предположение

В конкретни моменти от времето агентът ще прилага евристиките, за да получи някакви предположения. Всяко предположение ще се състои от заключение и от коефициент на достоверност. Заключението ще бъде атомарна формула и тя ще се отнася към конкретен момент от времето *t* (този момент може да е в миналото или в бъдещето). Тоест, евристиката беше за всяко *t*, а предположението е за едно конкретно *t*.

Дефиниция: Предположението има вида:

A(t) (процент)

Множеството на всички предположения (приложения на евристики) ще го означим с *Guesses*.

Нека имаме още една евристика: „Ако гърнето е пълно с жълтици, то можем да си купим самолет.“ Нека коефициентът на достоверност на тази евристика е 50%. Тогава в момент, когато виждаме гърне, което изгледа старо, тогава прилагаме първата евристика и получаваме „гърнето е пълно с жълтици“ (с достоверност 2%). От там получаваме, че можем да си купим самолет (с достоверност 1%).

Важното, което се вижда от този пример е, че можем да прилагаме евристиките каскадно и че в предпоставките на евристиката може да имаме невидимо събитие. Каскадното прилагане на две евристики може да се разглежда като една нова евристика, но за директното намиране на тази нова евристика ще трябва твърде много жизнен опит. По-добре е да намираме по-прости евристики и да ги комбинираме като получаваме по-сложни. Можем да намерим много евристики за това, че в гърнето има жълтици и чрез каскадното прилагане да заключим, че във всичките тези случай можем да си купим самолет. Ако не използваме каскадното прилагане, тогава за всеки отделен случай ще трябва да съберем отделна статистика, която да докаже, че в този конкретен случай можем да си купим самолет.

2.5.6 Дефиниция на живот

Ще разгледаме живота като игра между двама играчи. Първият играч ще бъде агентът (главният герой), а вторият играч ще го наречем „природа“ и това ще бъде самият свят. Агентът „природа“ не прави каквото си иска. Той действа по някакви правила, макар че той може да има известна свободна воля. По-нататък ще се опитаме да си обясним света и

ще заменим природата с група агенти. Тази група ще се състои от нула, един, двама или повече агенти. Когато това да е празната група, ще казваме, че в света няма други агенти освен главния герой.

Животът ще се състои от две неща: следа и гръбнак.

Дефиниция: Живот:
$a_1, o_2, a_3, o_4, \ldots, a_{k-1}, o_k$, където $a_i \in \Sigma$, $o_i \in \Omega$.
$u_0, w_1, u_2, w_3, \ldots, w_{k-1}, u_k$, където $u_i \in U$, $w_i \in W$.

Тук Σ са възможните действия, Ω са възможните наблюдения на агента, U са състоянията на света, когато агента е на ход, а W са състоянията на света, когато природата е на ход. С k сме означили дължината на живота (ако животът е L, тогава дължината му е $|L|$).

Ако напишем следата и гръбнака на един ред това ще изглежда така:
$$u_0, a_1, w_1, o_2, u_2, a_3, w_3, \ldots, a_{k-1}, w_{k-1}, o_k, u_k$$

От гледна точна на света животът е следа и гръбнак, от гледна точка на агента животът е следа и предположения, от гледна точка на двамата животът е живот с предположения.

Дефиниция: Живот с предположения:
- следа,
- гръбнак,
- предположения на агента (приложения на евристики):
$$g_0, g_1, g_2, \ldots, g_k, \text{ където } g_i \subseteq Guesses.$$

Множеството g_i се състои от предположенията, които агентът може да направи в момента i (не по-рано, а точно в този момент). Всички предположения валидни в момента t са:
$$G_t = \bigcup_{i \leq t} g_i$$

Множеството G_t от предположенията може да е противоречиво. За някое j можем да предположим едновременно $A(j)$ и $\neg A(j)$ (евентуално с различна достоверност). Предположенията $A(j)$ и $\neg A(j)$ може да се появят в един момент (в едно g_i), а може да се появят в два различни момента.

Агентът мисли само в четните моменти, защото тогава той е на ход. Ако t е четен момент, тогава агентът не може да избере предположенията g_t, защото те се определят от миналото, но той може да избере предположенията g_{t+1}, защото те се определят от миналото и от неговото следващо действие. (Тоест, агентът ще пробяга възможните действия и ще избере това, което би му дало най-доброто g_{t+1}.)

2.6 Event-Driven модели

Преди още да сме казали какво е събитие, ще кажем какво е Event-Driven модел. Грубо казано ED моделът е един ориентиран граф, в който върховете отговарят на състояния на света, а стрелките са етикирани със събития. Предполагаме, че светът си седи в едно от състоянията докато не се случи някое от наблюдаваните събития. Тогава светът преминава

в друго състояние, като преходът става по стрелка етикирана със събитието, което се е случило.

Важно нещо за ED модела това са особеностите. Особеност ще наричаме, когато в едно състояние едно събитие се случва с вероятност различна от очакваната (от средната). Особеностите ще ни помагат от една страна да разберем в кое състояние на света сме, а от друга страна ще ни помагат да предсказваме какво ще се случи и какво се е случило. Множество от особености ще наричаме следа.

Стрелките в ED модела, както и особеностите ще си имат вероятност. Вместо точна вероятност ние ще използваме вероятностни интервали. Първо ще кажем защо предпочитаме да използваме вероятностни интервали. Второ ще кажем защо правим разлика между невъзможно и възможно, но невероятно. Трето, ще въведем статистиката, която ще даде смисъл на тези вероятности. Чак след това ще дадем дефиниция на ED моделите.

2.6.1 Вероятностен интервал

Когато играете баскетбол и се опитвате да вкарате кош, каква е вероятността да уцелите? Можем да направим сто опита и да изчислим тази вероятност чрез статистиката. Може да има и други фактори, които да влияят на тази вероятност. Например, какво е осветлението и дали сте уморен. Можем да направим по-сложен модел, който да включва и тези фактори, но въпреки това винаги ще останат фактори, които не сме успели да включим. Тези неизвестни фактори ще ни дадат някакво отклонение във вероятността и тя няма да бъде точна стойност, а ще бъде някакъв интервал.

Най-важният от тези допълнителни фактори е нашата свободна воля. Когато стреляме, много важно е дали искаме да вкараме или не искаме. Да допуснем, че когато не искаме, ние вкарваме по погрешка с вероятност a, а когато искаме, вкарваме с вероятност b (тогава вкарваме нарочно). Тоест, когато стреляме по коша ние ще вкараме с вероятност, която е в интервала $[a, b]$.

2.6.2 Възможно, но невероятно

Дали има разлика между невъзможно и невероятно? Ще правим разлика между липсваща стрелка и стрелка с вероятност нула (т.е. вероятностният интервал $[0, 0]$).

Вероятността на едно събитие може да клони към нула. Например, ако събитието се случва веднъж в първите десет, веднъж в следващите сто и т.н. Тогава събитието е възможно, но неговата вероятност е нула (или клони към нула).

Когато разглеждаме действията на света (наблюденията на агента), тогава разликата между невъзможно и невероятно е малка, но когато разглеждаме действията на агента, тогава разликата е много съществена, защото това е начина по който дефинираме понятието „некоректен ход“.

Когато стрелките са по действия и една стрелка липсва, ще предполагаме, че действието е некоректно (некоректен ход). Това ще е отделно събитие, което ще наречем „полувидимо“. Когато има стрелка с вероятност нула, това ще означава, че действието е коректно, но по някаква причина нашият агент това действие никога не го извършва (или почти никога).

2.6.3 Статистика

Основният инструмент, който агентът използва, за да разбере света, е статистиката. Тук ще разгледаме два вида статистика. Първата е статистиката на агента и тя е на базата само на действията и наблюденията. Това е реалната статистика, с която агентът разполага.

Ще разгледаме и втори вид статистика. Ще я наречен статистиката на света. Тази статистика освен действията и наблюденията ще отчита още и състоянията на света. Статистиката на света можем да си я мислим като статистика направена от някакъв супер агент, които е в състояние да види кое е състоянието на света. Разбира се, такъв супер агент не съществува. Предполагаме, че светът знае кое е състоянието, в което се намира. Затова светът би могъл да направи такава статистика, но той няма да я сподели с агента.

Тоест, статистиката на света е нещо абстрактно, което агентът не може да види, но ние ще използваме тази статистика, за да дефинираме и да дадем смисъл на някои от въведените понятия. Например, каква е вероятността по една стрелка да се премине от едно състояние в друго. Нека предположим, че вероятностите са фиксирани и ще определим тези фиксирани вероятности чрез статистиката на света. Ще направим N стъпки и за всяко състояние и за всяка стрелка ще преброим колко пъти сме минали през тях (колко пъти сме влезли в състоянието, без значение колко стъпки сме били в него). Ако през една стрелка сме минали k пъти, а през състоянието, от което тя излиза сме минали m пъти, тогава вероятността на стрелката ще бъде приблизително k/m. Разбира се, при малко N грешката ще е голяма, но при голямо N може да приемем, че вероятността е точно k/m. Тоест, вероятността ще е границата на k/m, когато N клони към безкрайност.

Така използвахме статистиката на света, за да дадем смисъл на вероятността на стрелките. Можем ли да я използваме, за да дефинираме вероятностните интервали? Това ще е малко по-трудно. Нека отново сме направили N стъпки. Нека n е най-голямото цяло число, за което $n^2 \leq N$. Ще разделим интервала N на n интервала, всеки от които с не по-малко от n стъпки. За всеки от тези n интервала ще направим статистика и ще получим някаква вероятност p_i. Вероятностният интервал на стрелката ще бъде приблизително $[min(p_i), max(p_i)]$. Когато N клони към безкрайност би трябвало да получим точната стойност, но все пак, за да бъде това вярно, ще трябва да направим едно предположение.

Ще предположим, че светът и агентът променят фиксираната си стратегия, но го правят рядко и оттам рядко се променя фиксираната стратегия на ED модела (по-долу казваме какво е фиксирана стратегия). Когато светът и агентът изпълняват фиксирана стратегия, тогава вероятността на всички стрелки в ED модела също ще е фиксирана. Ако светът и агентът сменят стратегията си прекалено често, тогава този метод за статистика няма да даде нужния резултат (т.е. ще даде по-тесни интервали от действителните).

2.6.4 Дефиниция на ED модел

Дефиниция: Event-Driven моделът ще се състои от три части:

Топология:

- S – множество от състоянията на модела (крайно или изброимо).
- E – множество от наблюдаваните събития (малък брой събития).
- $R \subseteq S \times E \times S$ – релация между състоянията етикетирана с наблюдаваните събития.
- Правило за разрешаване на колизиите.

Частична следа:

- $Trace \subseteq Distinctions$ – множество от особености.

Вероятности:

- Probability: $R \rightarrow [0, 1] \times [0, 1]$ – функция, която на всяка стрелка връща вероятностен интервал.
- Back Probability: Същото като Probability, но вероятностите са за миналото (когато вървим обратно на стрелките).

Дефиниция: Особеност ще бъде наредена тройка състояща се от състояние, събитие и вероятност. Вероятността ще бъде вероятностен интервал или константата *never*. Вероятността трябва да бъде различна от средната вероятност на събитието (очакваната вероятност). Множеството на всички такива наредени тройки ще бъде *Distinctions*.

Ще предполагаме, че една стрелка липсва, точно тогава когато в следата има особеност, че в това състояние това събитие не може да се случи. Тоест:

$$\forall s_1 \in S \; \forall e \in E \; (\neg \exists s_2 \in S \; \langle s_1, e, s_2 \rangle \in R \Leftrightarrow \langle s_1, e, never \rangle \in Trace)$$

Наблюдаваните събития са тези, които са върху стрелките на модела. Събитията, които участват в следата ще ги наречем допълнителни събития. Наблюдаваните и допълнителните събития може да имат сечение, а може и да нямат.

Наричаме *Trace* частична, защото предполагаме, че съдържа само част от особеностите на модела. Ако предположим, че всички особености на модела са вътре, тогава ще говорим за пълна следа. (Ако моделът има повече от една интерпретация, тогава трябва да говорим за следата на интерпретацията, вместо за следата на модела.)

Правилото за разрешаване на колизиите ни казва кое е следващото състояние на ED модела, ако събитията a и b от E се случат едновременно.

Правилото може да е детерминистично:

1. Събитието a има приоритет пред събитието b и в този случай се изпълнява a.
2. В този случай се изпълняват последователно и двете събития, като първо се изпълнява a, а после b.
3. Има отделна стрелка, която ни казва къде да отидем, когато се случи a & b.

Правилото може да бъде и недетерминистично:

4. Избираме събитието a с вероятност p (или с вероятност в интервала $[p_1, p_2]$).

2.6.5 Интерпретация

Кога ED моделът е модел на света? За да отговорим на този въпрос ще дефинираме интерпретация и характеристика. При тази дефиниция ще използваме статистиката на света. Тоест, ще използваме информацията за това кое е състоянието на света, а това е нещо, което агентът не знае.

За да бъде ED моделът модел на света, трябва да има връзка между живота и състоянието му. Тази връзка ще наричаме „интерпретация“ и тя ще ни даде за всеки момент от живота в кое състояние се намира ED моделът (ще ни го даде точно или приблизително). Тоест,

интерпретацията е функция, която за всеки момент от живота ни дава състояние на ED модела (или множество от състояния или *belief*).

Интерпретацията може да не е единствена, но ако разширим частичната следа тя ще стане единствена. (Можем да я разширим малко или да я разширим до пълната следа.)

Разбира се, интерпретацията не е произволна функция, а такава която е съгласувана с ED модела. Когато в няколко последователни момента никое от наблюдаваните събития не се случва, тогава интерпретацията трябва да връща едно и също състояние за тези моменти. Когато се случи някое от наблюдаваните събития, тогава в следващия момент интерпретацията трябва да върне ново състояние, но трябва в ED модела да има стрелка по това събитие от предишното състояние към новото.

Ще разгледаме няколко вида интерпретации:

1. Проста интерпретация. Това е функция, която за всяко състояние на света ни връща състояние на ED модела.
2. Еднозначна интерпретация. Това е функция, която за всеки момент от всеки безкраен живот ни връща състояние на ED модела. Всяка проста интерпретация е еднозначна, но не и обратното.
3. Интерпретация чрез множества. Същото като еднозначната интерпретация, но функцията връща множество от състояния на ED модела вместо едно състояние.
4. Интерпретация чрез *beliefs*. Същото като интерпретация чрез множества, но функцията връща *belief* от състояния на ED модела (тоест, множество с вероятности).

За да опростим изложението, няма да разглеждаме случаите 3 и 4. Тоест, когато говорим за интерпретация, ще имаме предвид еднозначна интерпретация.

Функциите над моментите в живота ще ги наричаме многозначни събития. Обикновеното събитие е двузначно, то или се случва или не се случва. Ако едно събитие е *n*-значно, то ще го представим като *n* двузначни събития, които не се пресичат (не могат да се случат едновременно). Тоест, интерпретацията е многозначно събитие. Ще предполагаме, че интерпретацията е дефинирана при всеки безкраен живот. Когато живота е краен може да имаме проблем с дефиницията на някое събитие. Ако не ни достига информация за миналото или за бъдещето на живота, стойността на събитието може да е „не знам“. Затова приемаме, че интерпретациите са дефинирани върху безкрайните животи.

Дефиниция: Безкраен живот ще се състои от безкраен гръбнак и от безкрайна следа (безкрайността е в двете посоки):

$$\ldots, o_{-2}, \quad a_{-1}, \quad o_0, \quad a_1, \quad o_2, \quad a_3, \quad \ldots$$
$$\ldots, u_{-2}, \quad w_{-1}, \quad u_0, \quad w_1, \quad u_2, \quad w_3, \ldots$$

На един ред ще изглежда така:

$$\ldots, o_{-2}, u_{-2}, a_{-1}, w_{-1}, o_0, u_0, a_1, w_1, o_2, u_2, a_3, w_3, \ldots$$

За интерпретацията ще кажем, че е видима, ако тя се описва от видими събития. Видимата интерпретация е единствена.

Приемаме, че ED моделът се описва от топологията и от частичната следа. По това описание търсим интерпретация (една от възможните). После от интерпретацията можем да получим пълната следа и вероятностите.

2.6.6 Характеристика

За да бъде ED моделът модел на света той трябва да има интерпретация, но това не е достатъчно. Тази интерпретация трябва да има смислена характеристика.

Всяко събитие има характеристика и това е размито множество от състоянията на света, в които събитието се случва.

Дефиниция: Характеристика на събитие е функция, която на всяко състояние на света съпоставя вероятността това събитие да се случи, когато света е в това състояние.

Разликата между *belief* и характеристика е, че *belief* описва едно състояние, докато характеристиката описва множество от състояния. Сумата от вероятностите на *belief* е единица, а при характеристиката тази сума е между нула и безкрайност.

Интерпретацията е многозначно събитие, тоест тя се състои от много събития, всяко от които си има характеристика. Характеристика на интерпретацията ще бъде множеството от всички тези характеристики. За да бъде характеристиката на интерпретацията смислена, трябва тези характеристики да не са еднакви.

Забележка: Ще предполагаме, че всичко което състоянията на света „помнят“ и „знаят“ е съществено. (По-долу ще кажем, че предполагаме, че моделът на света е минимален.) Тоест, ако характеристиките на две събития са различни, то има нещо съществено, което ги различава.

Забележка: Когато характеристиката на ED модела не е смислена, тогава пълната следа е празното множество. Тоест, нищо интересно не се случва.

Най-хубав е случаят когато характеристиките не са размити множества, а са обикновени множества. Тогава имаме проста интерпретация и смислена характеристика.

Важно е дали ED моделът „помни“ съществени неща. Ако помни само съществени неща, тогава имаме проста интерпретация. Ако моделът помни само несъществени неща, тогава нямаме интерпретация или имаме, но характеристиката ѝ не е смислена.

Да вземем като пример Fully observable MDP (FOMDP). Този ED модел „помни“ кое е последното наблюдение. Да допуснем, че в нашия свят това е нещо несъществено. Интерпретацията на FOMDP е видима, но характеристиката на тази единствена интерпретация ще е безсмислена, защото, каквото и да е последното наблюдение, очакваното състояние на света ще е едно и също (защото предположихме, че в нашия свят това е безсмислена информация). Ако в нашия свят последното наблюдение беше съществена информация, тогава характеристиката на интерпретацията на FOMDP щеше да е смислена.

Нека сега да предположим, че последното наблюдение се помни от състоянието на света (тоест, това е съществена информацията и всяка буква от тази информация е съществена).

Щом светът „помни“ кое е последното наблюдение можем да разделим състоянията по това наблюдение и това ще е релацията на еквивалентност. Тогава FOMDP има проста интерпретация и тя съвпада с видимата.

Забележка: Това, което описахме по-горе се отнася за постоянните зависимости с постоянна следа. Зависимостта може да е временна (явление) и тогава интерпретацията няма да е върху целия живот, а ще бъде върху части от живота. (Явленията не се случват постоянно, а от време на време). Подвижната следа също усложнява модела.

2.6.7 Примери

Нека вземем едно събитие *a* и ED моделът, който помни дали това събитие се е случило четен или нечетен брой пъти (фигура 3).

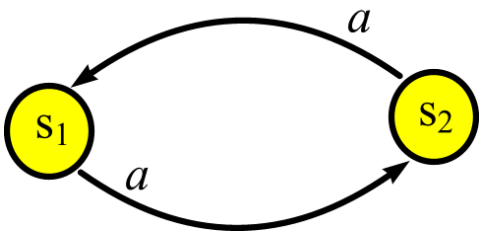


Figure 3

Нека предположим, че в нашия свят има значение дали събитието се е случило четен или нечетен брой пъти. Тогава всяко състояние на света „знае“ това и можем да разделим състоянията на две множества (такива, в които *a* се е случило четен брой пъти и останалите). Тогава имаме две прости интерпретации на ED модела. Можем да съпоставим на s_1 множеството от състояния, в които *a* се е случило четен брой пъти, а можем да го съпоставим на s_2. Всяка от тези две интерпретации ни дава следа. Можем да вземем пълната следа, но за да различим двете интерпретации е достатъчно да вземем произволна непразна част от пълната следа. Вероятностите по стрелките също се определят от интерпретацията, но в случая те са единица навсякъде.

Нека сега предположим, че в нашия свят няма никакво значение дали събитието *a* се е случило четен или нечетен брой пъти. При това предположение ED моделът от фигура 3 помни само несъществени неща. В този случай моделът няма да има интерпретация, защото тази интерпретация трябва за нещо да се хване. Състоянието на света не знае кое е състоянието на модела (s_1 или s_2), а миналото и бъдещето също не ни дават тази информация. Тоест, няма функция, която еднозначно да определя s_1 или s_2.

Ако се откажем от изискването моделът на света да помни само съществени неща, тогава бихме могли да получим нов модел, в който ще имаме интерпретация, но тази интерпретация нищо няма да ни даде (няма да ни даде следа). Нека да разширим модела на света като добавим тази несъществена информация (дали *a* се е случило четен брой пъти). Ще заменим всяко състояние *s* със състоянията <*s, even*> и <*s, odd*>. Ако състоянието *s* „помни“, че *a* се е случило четен брой пъти. Тогава състоянието <*s, even*> ще е достижимо, а състоянието <*s, odd*> ще е недостижимо (все едно, че го няма). Щом никое състояние не помни този факт, тогава всичките тези нови състояния ще са достижими. Тогава на s_1 ще съответстват всички състояния <*s, even*> или всички <*s, odd*>. И в двата случая няма да имаме никаква следа, защото вероятността за всяко събитие ще бъде средната вероятност.

Например в нашия свят имаме събитието „нов ден“. Това е важно събитие, но няма никакво значение дали денят е четен или нечетен. От друга страна е много важно кой е денят по модул седем, защото това са дните от седмицата. Състоянието на нашия свят „помни“ кой е денят по модул седем. Получаваме ED модел със седем състояния, който има следа. Следата е, че в неделя не се работи. Благодарение на тази следа ние може да синхронизираме всеки достатъчно дълъг живот и да кажем за всеки момент, кой ден от седмицата е (кое е състоянието в ED модела). Този ED модел има седем възможни интерпретации, но благодарение на частичната следа „в неделя не се работи“ ние фиксираме една от тези възможни интерпретации.

Друг пример, при който моделът има повече от една интерпретация. Да вземем ED модел, в който имаме недетерминиран преход към състоянията s_1 и s_2. Нека множеството от състоянията на света, които отговарят на s_1 и s_2 да го разделим по различни признаци на две. Може да го разделим на „сини“ и „зелени“ или на „големи“ и „малки“. На всяко едно такова разделяне ще отговаря проста интерпретация. Тези интерпретации може да са много (ако множеството е изброимо разделянията ще са континуум.) Разбира се, колкото повече са особеностите в частичната следа, толкова по-малко са възможните интерпретации, а при пълната следа ще имаме само една възможна интерпретация.

2.7 Събития

Ще разгледаме два вида събития: релевантни и нерелевантни. Релевантните събития ще ги разделим на три: видими, невидими и полувидими.

2.7.1 Видими събития

Това ще са събитията, които зависят само от следата на живота (действията и наблюденията). Зависят от това, което сме видели и това, което ще видим.

Забележка: Защо считаме, че събитието може да зависи от бъдещето? Защото бъдещето в един следващ момент ще стане минало и защото бъдещото може да се предскаже. Тоест, видимото събитие в по-късен момент ще се види, а чрез предположенията може да се предскаже с известна степен на достоверност още преди да сме го видели. Бъдещето е неизвестно от гледна точка на текущия момент, но спрямо друг момент то може вече да е известно. Например, спрямо края на живота, това бъдеще вече е част от миналото и е известно.

Нека имаме множеството *All* от всички безкрайни редици от действия и наблюдения:

$$\ldots, a_{t-3}, o_{t-2}, a_{t-1}, o_t, a_{t+1}, o_{t+2}, a_{t+3}, \ldots$$

Тези редици са безкрайни и в двете посоки (към миналото и към бъдещето). Тук е важно кой е текущият момент „нула“. Тоест, безкрайните редици можем да ги приемем за функции: $\mathbb{Z} \to \Sigma \cup \Omega$.

Дефиниция: Всяко подмножество на *All* ще го наричаме видимо събитие.

Тоест, за всяка безкрайна редица от действия и наблюдения и за всеки конкретен момент t ние можем да кажем дали събитието се случва или не се случва.

При тази дефиниция видимите събития станаха прекалено много. Елементите на *All* са континуум много, което значи, че видимите събития са две на степен континуум.

Ние не разполагаме с безкрайна редица от действия и наблюдения, а имаме само един краен откъс (имаме следата на един конкретен живот). Ще искаме видимите събития да се определят само от този краен откъс. Тогава видимото събитие ще има три възможни стойности. То ще може да е истина, да е лъжа или да е „не знам“.

Дефиниция: Видимото събитие върху откъс:
1. ще е истина, ако както и да продължим откъса до безкрайна редица се получава редица, за която събитието е истина.
2. ще е лъжа, ако както и да продължим откъса се получава редица, за която събитието е лъжа.
3. ще е „не знам“ в противен случай.

Ограничавайки видимите събития до откъси ние силно намалихме техния брой. От две на степен континуум, те станаха континуум много. Кой събития отпаднаха? Например: „Събитието *A* ще се случи безкраен брой пъти.“ Ако ограничим това до краен откъс, то винаги ще даде „не знам“, защото можем да продължим редицата и по двата начина. Разбира се, такива събития не са интересни, защото нас ни интересуват само събития, които можем да разберем в даден момент.

Няма да разглеждаме всички видими събития, а само изчислимите и то само малка част от изчислимите събития. Най-важните видими събития са атомарните: *Ob(t)* и *Act(t)*. Съставните събития направени от видими събития също ще са видими събития.

2.7.2 Съставни събития

От атомарните събития можем да правим съставни събития. Конюнкцията на атомарни събития е съставно събитие.

Ако искаме да дадем име на една конюнкция и да я превърнем в атомарна формула, ще въведем съкращаващ символ, който ще бъде името на конюнкцията. (Няма да наблюдаваме дизюнкция, но можем да наблюдаваме конюнкцията от отрицанията, а това е същото.) За всяко съставно събитие можем да въведем съкращаващ символ и да започнем да го наблюдаваме.

Съкращаващите символи за конюнкция ще ги въведем със специални евристики с коефициент на достоверност 100%. Евристиките ще имат вида:

$$A_1(t\text{-}i_1)\ \&\ A_2(t\text{-}i_2)\ \&\ \dots\ \&\ A_n(t) \Rightarrow Abbreviation(t)\ (100\%),\ i_j \geq 0,\ i_n = 0.$$

За отрицанията ще добавим още *n* евристики:

$$\neg A_j(t\text{-}i_j) \Rightarrow \neg Abbreviation(t)\ (100\%)$$

Тоест, конюнкцията на атомарни събития ще стане атомарно събитие, когато определим съкращаващ символ означаващ тази конюнкция.

Обикновено евристиките са получени чрез статистика и имат коефициент на достоверност по-малък от 100%. Затова ще приемем, че въведените тук евристики са добавени служебно.

Друго съставно събитие е „*A* ще се случи преди *B*“. Ще запишем това събитие така: *exist A after t before B in our case*. Това събитие е видимо, когато *A* и *B* са видими. Ако в един безкраен живот събитията *A* и *B* не се случват след *t*, тогава събитието ще е лъжа.

2.7.3 Невидими събития

Тук ще говорим за моментните невидими събития. Разбира се от тях можем да направим и други невидими събития (например чрез конюнкция).

Моментно невидимо събитие (TUE) ще бъде такова, което отразява възможното минало и възможното бъдеще в един конкретен момент *t*. То ще зависи само от едно състояние на света. Интерпретация на TUE ще бъде множество от състояния на света.

Забележка: Защо интерпретацията на TUE е множество от състояния на света, а не е множество от *beliefs*? Защото считаме, че моделът на света е фиксиран. За всяко състояние на света TUE е или истина или лъжа. Имаме *belief*, когато не знаем точно в кое състояние сме, но считаме, че светът „знае“ в кое състояние се намира. Ако сменим модела на света, тогава някое от състоянията на новия свят може да съответства на *belief* от състояния на стария. Сменяйки света ще сменим и интерпретацията.

Интерпретацията дава смисъла на TUE в конкретен свят, но ние ще опишем TUE в произволен свят. Тоест, ще погледнем нещата от гледната точка на агента, а не на света.

Възможното бъдеще може да се представи като дървото от възможните бъдещи развития. Ще приемем, че и възможното минало може да се представи като дърво. Тогава възможното минало и бъдеще могат да се представят като двойка дървета. Нека *All2* е множеството от всички двойки от такива дървета.

Дефиниция: Всяко подмножество на *All2* ще го наричаме TUE.

Според тази дефиниция TUEs са прекалено много. Ще опишем шест типа, които ще са важни за нас.

2.7.3.1 Винаги се случва

Пример за такова събитие е следното: „Събитието *A* винаги ще се случва в бъдеще.“ (Аналогично за миналото: „Събитието *A* винаги се е случвало.“) Бихме искали да сложим някаква граница. Затова ще обобщим това събитие до следното: „Събитието *A* винаги ще се случва в бъдеще до момента, в който се случи събитието *B*.“ Ще запишем това събитие така: *A from t until B*.

Нормално е да потърсим евристики, които да предскажат това TUE. За целта ще използваме статистика върху жизнения опит. Избираме една конюнкция, която е кандидат за предпоставка на евристиката. Нека *m* пъти тази конюнкция е била истина и нека за *k* от тези случай *A from t until B* да е било истина. Тогава добър кандидат за коефициент на достоверност е *k/m*. Този коефициент не отчита това, че при малки *m* достоверността е малка, затова е по-добре да вземем *(k-1)/m*. Тогава коефициента ще е по-малък от 100%, но ще клони към 100%, когато *k=m* и когато *m* клони към безкрайност.

Нека в конюнкцията, която сме избрали, има невидимо събитие. Нека според някакви евристики това невидимо събитие да е истина. Тази информация можем да я вземем от предположенията G_t. Ако има няколко предположения за невидимото събитие, тогава нека

R е най-големият възможен коефициент на достоверност. В този случай ще добавим към броячите *R* вместо единица (броячите където акумулираме *k* и *m*). Тоест, ще отчетем това, че не е сигурно, че предпоставката е истина.

2.7.3.2 Ще се случи

Това събитие е подобно на събитието „*A* винаги се случва“, но не може да бъде получено от него чрез две отрицания, защото така ще получим събитието: „*A* може да се случи“. Последното означава, че по някой от пътищата на дървото на възможното бъдеще ще се случи *A*, но ние искаме *A* да се случи по всеки път (във всеки възможен живот).

Затова въвеждаме ново атомарно невидимо събитие: „винаги *A* се случва преди *B*“. Ще запишем това събитие така: *exist A after t before B in all cases*.

Пак може чрез статистика да търсим евристики, които да ни предсказват това събитие. Само трябва да отбележим, че тук имаме и случая „не знам“. Този случай се получава, когато живота свършва преди да се е случило едно от събитията *A* или *B*. В този случай не знаем дали, ако живота беше продължил щеше първо да се случи *A* или първо *B*.

Има разлика между „*in our case*“ и „*in all cases*“. Първото означава, че това ще се случи в нашия живот, а второто означава, че това ще се случи във всяко възможно продължение на бъдещето. Събитието „*exist A after t before B in all cases*“ е невидимо, дори и когато *A* и *B* са видими.

2.7.3.3 Ще се случи с вероятност p

Събитието „*A* се случва между *t* и *B* с вероятност *p*“ е различно от горните две събития. Горните две събития имаха свойствата, че запазват нулата и единицата. Едно събитие да запазва нулата, означава че: когато то е лъжа в две състояния, то то е лъжа и във всеки *belief* направен от тези две състояния. Аналогично и за свойството „запазва единицата“.

Горното събитие може да е лъжа в две състояния, но да е истина в *belief* съставен от тях. Например, нека в двете състояния *A* се случва с вероятности *1/4* и *3/4*. Тогава от двете състояния може да направим *belief*, в който *A* се случва с вероятност *1/2*.

2.7.3.4 Достижимо събитие

Събитието „*A* е достижимо“ е между събитията „*A* може да се случи“ и „*A* ще се случи“.

„*A* е достижимо“ означава, че съществува алгоритъм *P*, такъв че, ако агентът изпълнява *P*, тогава *A* ще се случи. Пример за алгоритъм е *Act(t+1)=a*. Разбира се, алгоритмът може да бъде и по-сложен.

Как можем да намерим евристика за „*A* е достижимо“? Пак ще изберем една конюнкция, която е кандидат за предпоставка на евристиката. След това ще разгледаме различни случай. За всеки от случаите ще намерим алгоритъм *P* и евристика, която ни казва, че ако предпоставка е налице и ако изпълним алгоритъма *P*, тогава *A* ще се случи. Ако сме покрили всички случай, тогава можем да заключим, че при тази предпоставка *A* е достижимо.

2.7.3.5 Тестово събитие

В Dobrev (2017a) подробно са разгледани тестовите събития.

Пример за тестово събитие е: „Ако натиснем дръжката на вратата, то тя ще се отвори.“ Това тестово събитие може да го наречем „вратата не е заключена“. Стойността на тестовото събитие не зависи от това дали сме провели теста. Тоест, врата може да е заключена независимо от това дали сме го проверили или не сме.

2.7.4 Състояние на ED модел

Нека имаме един ED модел. Ще търсим интерпретация на този модел. За всяко състояние S_i на ED модела ще добавим свойството E_i, което ще е истина, когато ED моделът е в състоянието S_i. Ще търсим проста интерпретация, което значи, че предполагаме, че E_i е TUE.

За всяка стрелка на ED модела ще добавим по една евристика:

$$E_i(t) \,\&\, A(t) \Rightarrow E_j(t+1) \ (процент)$$

Тук A е наблюдавано събитие и стрелката е по това събитие от S_i към S_j.

Частичната следа също ще я опишем по подобен начин. За всяка особеност ще добавим евристиката:

$$E_i(t) \Rightarrow A(t) \ (процент)$$

Тук S_i е състоянието, A е събитието от особеността, а процентът отговаря на вероятността.

С помощта на тези евристики ще можем да предположим в кое състояние се намира ED моделът и с помощта на следата да предскажем допълнителните събития.

2.7.5 Полувидими събития

Заради некоректните ходове ще добавим полувидимите събития. За да разберем, че един ход е некоректен трябва да се опитаме да го играем. Затова тези събития ще са между видимите и невидимите. Това ще са събития, които ще видим, ако погледнем.

Ще въведем едно атомарно събитие (*Correct*) и две евристики, които ще дефинират това събитие.

$$Act(t)=a \Rightarrow Correct(a, t) \ (\ 100\%\ )$$

Тази евристика ни казва, че ако в момента t сме изпълнили действието a, това означава, че в този момент действието a е било коректно. Следващата евристика ни казва, че ако в момента t сме се опитали да изпълним действието a и не сме успели, тогава в t действието a е било некоректно.

$$UnsuccessfulTry(a, t) \Rightarrow \neg\, Correct(a, t) \ (\ 100\%\ )$$

Предположението $UnsuccessfulTry(a, t)$ ще бъде едно различно предположение. То няма да се получава чрез евристики, а служебно ще го добавим в g_t всеки път когато неуспешно сме се опитали да изпълним действието a в момента t. По този начин добавяме информацията за неуспешните опити.

Защо информацията за неуспешните опити я добавяме към предположенията? Предполагаме, че светът във всеки момент знае точно кой ход е коректен и кой е некоректен, но не знае кои от некоректните ходове агентът се е опитал да играе. При агента е обратното. Затова информацията за неуспешните опити ще е видима само за агента.

Двете служебни евристики, които добавихме са с коефициент на достоверност 100%, но освен това може да има и други евристики намерени чрез статистиката, които също да ни

казват дали ходът е некоректен. Нормално е да предположим, че когато агентът се обучи, той ще знае кой ход е некоректен и това ще го знае без да се е опита да го играе. Това ще стане благодарение на допълнителните евристики, който ще му даде статистиката.

2.7.6 Нерелевантни събития

Освен релевантните събития ще имаме и нерелевантни. Това са събития, които ние ще игнорираме и няма да се опитваме да разберем.

Ето три примера за такива събития:
1. „Кой живот живеем?“ При положение, че жизнения ни опит е последователност от много животи, можем да си зададем въпроса: Кой по ред е този живот?

Може да предположим, че светът във всеки живот се държи еднакво, но за агента е естествено да предполагаме, че той в първия живот прави грешки от неопитност и по-нататък тези грешки не ги допуска. Въпреки това, ще игнорираме този въпрос и ще считаме, че той е нерелевантен.

2. „Колко съм стар и колко ми остава?“ И този въпрос ще игнорираме. Тоест, ще игнорираме параметрите t и k-t. Причината да игнорираме този въпрос е: Считаме, че нашият агент е вечно млад и не старее с напредването на живота. Що се отнася до k-t, никой не знае колко му остава.

По-долу ще дефинираме разширения модел, в който t и k-t имат значение, но ние се интересуваме само от събития, които, ако са верни в един живот са верни и ако удължим живота (удължаваме напред и назад). Тоест, нас ни интересуват само събития, които не зависят от t и k-t.

3. Събитията, които носят несъществен информация също ще считаме за нерелевантни. В параграф 5.7. имаше такъв пример за това дали едно събитие се е случило четен или нечетен брой пъти.

2.7.7 Related work

Какво е събитие и кой въвежда понятието събитие в ИИ? Първият, който въвежда това понятие е Xi-Ren Cao. Той въвежда събитията в статията си Cao (2005).

По-късно, през 2008, Xi-Ren Cao заедно с Junyu Zhang в статията си Cao and Zhang (2008) дават дефиниция на понятието събитие. Тази дефиниция не е добра, защото зависи от модела и защото дефиницията предполага, че се помни последното състояние, в което е бил моделът, а това е нещо, което няма причина да бъде запомнено. Ето как изглежда тяхната дефиниция на събитие:

$$E \subseteq S \times S, \text{ при конкретен модел}$$
$$E = \{ \langle s_{i-1}, s_i \rangle \mid \text{ако } E \text{ се случва в момента } i \}$$

Защо тази дефиниция зависи от модела? Защото авторите на Cao and Zhang (2008) предполагат, че светът има само един единствен модел, а това не е така. В Dobrev (2019a) е показано, че светът има много модели. Дори в Dobrev (2019a) е показано, че светът има минимален и максимален модел. Тук минимален и максимален е по отношение на това какво знаят състоянията в модела за миналото и за бъдещето.

Каква трябва да бъде дефиницията?

$$E \subseteq S, \text{ при произволен модел}$$
$$E = \{s_i \mid \text{ако } E \text{ се случва в момента } i\}$$

При различни модели състоянието може да „помни“ повече или по-малко от миналото. При дефиницията на Xi-Ren Cao състоянието помни последното състояние, в което е бил моделът. Както казахме, това е нещо, което няма причина да бъде запомнено. Ако решим да помним нещо, по-добре е да запомним последното действие на агента. Така ще е очевидно, че действията на агента са събития.

Дефиницията в Cao and Zhang (2008) не е съвършена и може и трябва да се подобри, но това по никакъв начин не намалява заслугата на Xi-Ren Cao, защото той е първият, който забелязва, че не е достатъчно да наблюдаваме само действията и че трябва да обобщим до по-широк клас от събития.

Всъщност, действията ни казват всичко, но те ни дават прекалено много информация. Когато моделът следи всички действия, той е залят и претоварен с прекалено много информация. Обобщавайки действията до произволни събития ние ограничаваме входящата информация и можем да се ограничим до „важните“ неща.

Забележка: В MDP наблюденията не са сред проследените събития, но те се отчитат чрез следата. Следата уточнява кое е текущото състояние и по този начин се отчитат наблюденията.

Забележка: Терминът събитие се използва в много статии, но обикновено в друг смисъл. Например в Lamperti, Zanella and Zhao (2020) терминът събитие се използва в смисъл на наблюдение. Наблюдението е частен случай на събитие, но при нас събитието е по-общо понятие.

2.8 Свят

За да създадем език за описание на светове, първо трябва ясно да кажем какво е свят. Ще дефинираме света чрез неговите съвършени модели. Идеята е следната: Представете си, че имате картина и нейно съвършено копие, което е толкова добро, че не може да се различи копието от картината. В този случай може да приемем, че копието и картината са едно и също нещо.

Светът ще се състои от модел (който ще бъде Simple MDP) и от един начален *belief*, който ще ни покаже откъде се очаква да започне животът. Моделът ще искаме да е съвършен (да не може да се подобри) и да е минимален.

Минималният модел не е единствен, дори и съвършеният минимален модел не е единствен, но за всеки два съвършени минимални модели можем да кажем, че състоянията на първия могат да се изразят като *belief* от състоянията на втория.

2.8.1 Минимален модел

Да е минимален един модел означава състоянията му да не знаят нищо излишно. Тоест, да не помнят нищо излишно от миналото и да не знаят нищо излишно за бъдещето. Ненужен

(излишен) факт от миналото е такъв, който не влияе на бъдещето. Щом не влияе на бъдещото, тогава защо да го помним? Излишен факт за бъдещето е такъв, който не зависи от миналото (който не може да се предскаже при помощта на миналото). Има ли смисъл моделът да знае за бъдещето ненужни (излишни) неща? Например, получили сте писмо, но още не сте го отворили, знае ли светът какво пише в писмото. Ще допуснем, че миналото по никакъв начин не може да ни помогне да предскажем какво пише в писмото. В този случай, какво пише в писмото, е един факт излишен за света и той няма нужда да го знае, защото той може да реши това чак когато вие отворите писмото.

Разбира се, вие предполагате, че живеете в реален свят и че светът знае какво пише в писмото още преди да сте го отворили. Представете си, че вие живеете в компютърна програма (като филма „Матрицата“) и че светът решава какво да се случи в последния момент. Тоест, решава какво пише в писмото не когато то е написано, а чак когато вие го отваряте.

Какво пише в писмото е факт излишен за света, но не и за агента. Този факт е важен за агента, защото е свързан с неговото бъдеще. Този факт ще е непредсказуем за агента, защото не е свързан с миналото. Въпреки, че този факт е непредсказуем, агентът ще се опитва да го предскаже, защото той няма как да знае, че този факт не може да бъде предсказан и че опитите му ще бъдат напразни.

2.8.2 Съвършен модел

Казахме, че състоянията на минималния модел не знаят нищо излишно. За състоянията на съвършения модел ще кажем, че знаят всичко полезно. Тоест, ако един модел е съвършен и минимален, тогава неговите състояния знаят точно това, което трябва (всичко полезно и нищо излишно).

Това, че моделът е съвършен може да се каже по много начини:
1. Моделът притежава свойството на Марков.
2. Бъдещето зависи само от това от кое състояние тръгваме и не зависи от това как сме стигнали до това състояние.
3. Моделът не може да се подобри и да се направи друг модел, който на базата на миналото да дава по-добра прогноза за бъдещето.
4. Никое състояние не може да се раздели на две състояния, така че новите две състояния да имат различно минало и различно бъдеще. Да имат различно минало означава на базата на това, което се е случило, да можем да ги различим. Да ги различим означава да ги различим със сигурност или да кажем, че едното е по-вероятно от другото. Не е нужно всяко възможно минало да различава двете състояния. Достатъчно е да има едно възможно минало, което да ги различава. По аналогичен начин дефинираме какво означава две състояния да имат различно бъдеще.

Тези дефиниции на съвършен модел са еквивалентни. Състоянието на съвършения модел няма смисъл да се разделя на две, защото то знае всичко полезно. Ако на базата на миналото го разделим на две, то новите две състояния ще знаят още нещо за миналото, но това нещо няма да е нещо полезно и следователно двете състояния ще имат еднакво бъдеще.

Въпрос: Ако вземем произволен модел, дали той е съвършен модел на някой свят? Отговорът е да. Всеки модел е съвършен модел на някой свят. Всеки модел описва някакъв

свят и ако предположим, че това е възможно най-доброто описание и че описанието не може повече да се подобри, тогава моделът е съвършен. Разбира се, има безбройно много светове, за които този модел дава частично описание, което може да се подобри. Въпросът е дали сме намерили съвършения модел на света, когото се опитваме да опишем? Отговорът на последния въпрос е, че не знаем. Това, което знаем за света е нашия жизнен опит. Има безбройно много светове, в които този опит може да се случи. Разбира се, ние търсим най-простия модел отговарящ на нашия жизнен опит (този принцип е известен като бръснача на Окам).

Трябва да отбележим, че ние дори не сме сигурни, че моделът, който сме построили на базата на нашия жизнен опит, е коректен. В модела има някакви вероятности, които сме определили чрез статистика. Тези вероятности може да не са точни, поради малката статистика, но ако предположим, че статистиката е достатъчна и че вероятностите са точни, тогава остава само въпросът дали моделът е съвършен.

Ние няма да търсим съвършен модел на света. Ние ще търсим достатъчно добър модел, който да ни свърши работа. В Dobrev (2019b) показахме, че търсенето на съвършен модел е прекалено амбициозна цел. Въпреки това, ние ще предположим, че съвършеният модел на света съществува и че този модел е дефиницията на света.

Също така няма да се опитваме да намерим минимален модел на света, защото това означава да приемем, че определени събития са непредсказуеми. Обикновено ние се опитваме да предскажем всяко събитие, макар че за някои събития (като хвърлянето на зар) приемаме, че резултатът е непредсказуем.

2.8.3 Базов модел

Дефиниция: Базов модел на света ще бъде всеки съвършен и минимален модел на света.

Ще предположим, че всеки свят си има поне един базов модел и това ще е определението на този свят. Един модел може да е базов за един свят (да е съвършен), но за друг свят той може да е обикновен модел (да не е съвършен).

Състоянията на базовия модел знаят всичко за миналото (какво би могло да се е случило) и всичко за бъдещето (какво би могло да се случи).

Забележка: Някои автори използват различна терминология. Например в Schofield and Thielscher (2019) се използва термина „игра“ вместо „свят“. Също така в Schofield and Thielscher (2019) се използва термина “Game Description Language” вместо “Language for Description of Worlds” и “Imperfect Information”вместо “Partial Observability”. Можем да разглеждаме света като игра, както и играта като отделен самостоятелен свят. Както се пее в песен на група Mystery “The World is a Game”.

2.8.4 Simple MDP

Обикновено светът се описва чрез Markov decision process (MDP). Ние ще опростим този модел и ще създадем Simple MDP.

Защо е нужно MDP да се опрости? Защо в MDP има излишно усложняване? Причините за това са две.

Първата причина е, че MDP крие факта, че това е модел не само на света, но и на агента. Истината е, че светът и агентът са една единна система и не можем да опишем само едното без да описваме другото. Тоест, MDP описва не само поведението на света, но и поведението на агента. Ако се вгледате в MDP, ще видите, че MDP казва, че агентът прави каквото си иска. Да правиш каквото си искаш, това също е поведение, макар и това да е възможно най-свободното поведение. Каква е вероятността агентът да избере определено действие? Не знаем каква е тази вероятност, агентът прави каквото си иска, което означава, че вероятността е в интервала [0, 1].

Втората причина е, че MDP ограничава света и го принуждава да използва една точно определена фиксирана стратегия. Тоест, при MDP агентът прави каквото си иска, а светът е ограничен до фиксирана стратегия.

2.8.5 Фиксирана стратегия

Какво е фиксирана стратегия? Ако при определена ситуация завивате винаги наляво, това означава, че изпълнявате фиксирана стратегия. В този случай стратегията дори е екстремна, защото винаги наляво и винаги надясно са двете екстремни възможности. Ако хвърляте монета и когато се паднеези завивате наляво, тогава вие изпълнявате фиксирана стратегия с вероятност 1/2 (точно 1/2). Ако вероятността е друга, тогава и фиксираната стратегия, която изпълнявате е друга. Ако завивате наляво или надясно както си поискате, тогава вие не изпълнявате фиксирана стратегия, а завивате наляво с вероятност, която е в интервала [0, 1].

Имате ли свободна воля, когато изпълнявате фиксирана стратегия? Отговорът е не. Когото хвърляте монета, тогава не решавате вие а монетата. Тоест, при MDP агентът има напълно свободна воля и не е ограничен от нищо, а светът е напълно ограничен и е принуден да изпълнява фиксирана стратегия.

2.8.6 Екстремна стратегия

Екстремната стратегия е фиксирана стратегия, която е избрана така, че всяка вероятност е избрана на минимума или на максимума.

Как избираме екстремна стратегия? Ако вероятностите са определени с интервалите $[a_i, b_i]$, тогава избираме стойността на един от интервалите екстремно (т.е. избираме a или b). Ако сме избрали b, това може да стесни останалите интервали. След това от останалите (евентуално стеснени) интервали избираме един от тях и продължаваме нататък. По този начин избираме фиксирана стратегия, в която всяка вероятност е екстремна (тоест, не може да се увеличи или не може да се намали).

Когато говорим за стратегия в Simple MDP ще имаме предвид обща стратегия на агента и на света. Тоест, двамата са се наговорили и играят две стратегии, които взети заедно са общата стратегия на Simple MDP. Когато се говори за стратегия на MDP се има предвид стратегия само на агента, защото светът в MDP има фиксирана стратегия и няма накъде да мърда. Кого се говори за стратегия на агента в MDP се има предвид екстремна стратегия (всяка вероятност е или 0 или 1). Защо в MDP екстремните стратегии са достатъчни? Когато преследваме една цел и трябва да решим „наляво или надясно“, тогава обикновено има три възможности. За нашата цел по-добре е наляво, по-добре е надясно или все едно е дали наляво или надясно. Ако приемем, че в третия случай избираме наляво, тогава получаваме екстремна стратегия. Затова в MDP екстремните стратегии са достатъчни.

Разбира се, това не важи за всяка цел. Ако целта е разнообразието (т.е. да обиколим повече), тогава трябва да редуваме ляво и дясно.

Хубавото на екстремните стратегии е, че когато състоянията са крайно много, тогава и екстремните стратегии са крайно много, докато фиксираните стратегии обикновено са континуум.

2.8.7 От MDP към Simple MDP

С MDP можем да опишем само част от световете. Това са едни много специални светове, в които светът няма свободна воля и е принуден да изпълнява фиксирана стратегия. Ако вътре в света живее агент със свободна воля, този свят не може да се опише с MDP. Нека разгледаме света на играта шах, в който има втори агент, който играе срещу главния герой. Нека този втори агент не е детерминиран (не играе екстремна стратегия). Нека дори да не е длъжен да играе с фиксирана стратегия. Нека този агент да играе както си поиска. Тогава този свят не може да се представи с MDP, но ще можем да го представим чрез Simple MDP. Разликата между двата модела е, че вместо точни вероятности в Simple MDP имаме вероятностни интервали.

MDP е излишно усложнен, защото е скрито, че вероятността на действията на агента е интервалът [0, 1]. Върху стрелките с действия има вероятности, но тези вероятности не определят вероятността на действието, а индиректно определят вероятността на наблюдението. (Наблюдението влияе върху това кое ще е следващото състояние на MDP модела, но това влияние е много сложно и индиректно. Стрелките недетерминирано определят няколко възможни състояния и на база на наблюдението някои от тези състояния отпадат. При Simple MDP нещата са много по-прости, защото там има стрелки по наблюденията и тези стрелки директно ни показват как ще повлияе наблюдението на това кое е следващото състояние.)

В MDP състоянието има минало, настояще и бъдеще. Тоест, нещо се е случило преди състоянието, нещо се случва вътре в състоянието и нещо ще се случи след състоянието. В Simple MDP има само минало и бъдеще, защото нищо не се случва вътре в състоянието. В състоянието на MDP има някакво наблюдение. Има две еквивалентни дефиниции на MDP. Ще ги наречем едноцветна и многоцветна дефиниция. При едноцветната дефиниция има само едно възможно наблюдение в състоянието, а при многоцветната в състоянието има няколко възможни наблюдения, всяко от които с точно определена вероятност. Тоест, при едноцветната дефиниция моделът не е минимален, защото състоянието „знае“ кое точно ще е наблюдението, а това знание може да не следва от миналото. Може миналото да дава недетерминистичен преход към няколко едноцветни състояния. Минималност ще се получи, ако тези няколко едноцветни състояния бъдат заменени с едно многоцветно, в което всяко наблюдение да се вижда със съответната вероятност.

2.8.8 Дефиниция на Simple MDP

Ще представим Simple MDP като finite automaton (без изискването броят на състоянията да е краен). По-точно ще го представим като probabilistic automaton защото върху стрелките ще имаме вероятности. Още по-точно ще го представим като interval-valued probabilistic automaton, защото вместо вероятности върху стрелките ще има вероятностни интервали.

При MDP състоянията са само от един тип, защото там винаги агентът е на ход. При Simple MDP ще разгледаме света като игра между агента и света. Там състоянията на света ще са два типа – състояния, при които агентът е на ход и такива, при които светът е на ход.

При MDP стрелката отговаря на един ход. Това е действие на агента и реакция на света. (Реакцията на света е наблюдението, което вижда агентът.) При Simple MDP стрелката ще отговаря на ply (полу-ход). Това или е действието на агента или е реакцията на света.

Дефиниция: Simple MDP ще бъде граф $(U \cup W, A \cup O)$ с два вида върхове и два вида стрелки.

- U са състояния на света, когато агентът е на ход (ще извърши действие).
- W са състояния на света, когато светът е на ход (ще покаже наблюдение на агента).
- A са стрелки отговарящи на действия.
- O са стрелки отговарящи на наблюдения.
- Ако $u \in U$, тогава стрелките излизащи от u са от A и стрелките влизащи в u са от O.
- Ако $w \in W$, тогава стрелките излизащи от w са от O и стрелките влизащи в w са от A.
- $A \to \Sigma \times [0, 1] \times [0, 1]$, на всяка стрелка от A съответства действие и вероятностен интервал.
- $O \to \Omega \times [0, 1] \times [0, 1]$, на всяка стрелка от O съответства наблюдение и вероятностен интервал.

Трябва да кажем още нещо за вероятностните интервали. Когато вероятността е фиксирана (интервали с дължина нула), тогава сумата от вероятностите на стрелките излизащи от един връх трябва да е единица. Когато вероятността не е фиксирана, тогава вероятностните интервали (на стрелките излизащи от един връх) ги разглеждаме като описание на множество от вектори от фиксирани вероятности, всеки от които има сума равна на единица. Тоест, интервалите $[a_i, b_i]$ описват вектора p_i, където $a_i \leq p_i \leq b_i$ и $\Sigma(p_i) = 1$. Ще искаме това описание да описва поне един вероятностен вектор (т.е. множеството от описаните вектори да не е празното). Ще искаме още описанието да е оптимално (т.е. при всяко стесняване на интервалите да се губи по някой вектор от множеството). В Dobrev (2017b) са дадени няколко неравенства, които трябва да са изпълнени, за да бъде описанието непразно и оптимално.

2.8.9 MDP като Simple MDP

Как можем да представим MDP като Simple MDP? Всички стрелки в MDP са по действие, тоест са от тип A. Всяка стрелка си остава, само вероятността ѝ p се заменя с интервала $[0, p]$. Тоест, ако агентът избере това действие, тогава ще избере тази стрелка с вероятност p, а в противен случай ще я избере с вероятност 0. В случаите, когато действието е само едно (няма стрелки по други действия излизащи от същото състояние), тогава p се заменя с интервала $[p, p]$.

Как променяме състоянията? Всяко състояние s се заменя с две състояния w и u, които ще са от типове W и U съответно. Всички стрелки, които досега са влизали в s сега ще влизат в w, а тези които са излизали от s, сега ще излизат от u. Колкото са били възможните наблюдения ob в s, толкова допълнителни стрелки ще добавим от w към u. Всяка стрелка ще е от тип O, ще бъде по съответното наблюдение ob и ще ѝ съответства вероятностният интервал $[p, p]$, където p е вероятността на наблюдението ob. Какво правим в случая на едноцветна дефиниция (т.е. когато всяко състояние има само по едно наблюдение)? Тогава стрелката от w към u ще е само една и тя ще е с вероятностния интервал $[1, 1]$.

Така полученият Simple MDP ще описва същия свят като модела MDP. Ако моделът MDP е бил съвършен, тогава и полученият Simple MDP ще бъде съвършен. Същото важи и за минималността.

Показахме, че всички светове, които могат да се представят като MDP могат да се представят и като Simple MDP, но не и обратното. Тоест, Simple MDP разширява идеята ни за свят.

2.8.10 Начален *belief*

За да опишем света ще ни трябва да добавим още едно начално състояние. Ние ще предпочетем началното състояние да не е едно, а да бъде множество от възможни начални състояния. Всяко едно от тези възможни състояния ще има някаква вероятност. Ако тази вероятност е фиксирана, ще получим една структура, която ще наречем фиксиран *belief*.

Дефиниция: Фиксиран *belief*:

- $M \subseteq U \cup W$
- $M \to [0, 1]$

Тоест, имаме множество от състояния и на всяко състояние от множеството сме съпоставили фиксирана вероятност. Сумата от тези фиксирани вероятности трябва да е равна на единица. Тук няма да има съществена разлика между $s \notin M$ и вероятността на s да е нула. В повечето статии това, което ние наричаме фиксиран *belief* се нарича *belief*, но при нас *belief* ще е нещо по-сложно.

Дефиниция: Обобщен *belief* ще наричаме множество от фиксирани *belief*s.

За по-кратко ще наричаме обобщения *belief* просто *belief*.

Ще предполагаме, че светът се описва чрез един Simple MDP и един начален *belief*. Кое е началното състояние, от което очакваме да тръгнем? Първо избираме един фиксиран *belief* от тези, които са елементите на началния *belief*. Как го избираме? Избираме ме го както си искаме, тук имаме случайност с неизвестна вероятност. След това от избрания фиксиран *belief* избираме едно конкретно начално състояние с вероятността, която е дадена от този фиксиран *belief*.

Смисъл на понятието начален *belief* може да се даде от жизнения опит и статистиката на света. Това е очакването за това кое ще е началното състояние на света.

2.9 Интерпретация

Вече казахме какво е интерпретация на ED модел, но още не сме казали какво е интерпретация на събитие.

Ще въведем разширения модел. Този модел ще се получи от базовия като добавим миналото и бъдещето. Тоест, състоянията на базовия модел знаят какво може да се е случило и какво може да се случи, а състоянията на разширения модел освен това ще знаят още какво точно се е случило да момента и какво точно ще се случи от този момент нататък.

Интерпретация на събитие ще бъде множество от състояния на разширения модел. Всяко събитие ще си има вероятност, защото всяко състояние на разширения модел ще си има вероятност.

За да опростим изложението ще предположим, че светът е обозрим. Обозримостта се характеризира с две неща:
1. Всяко събитие, което се е случило, може пак да се случи.
2. Началният *belief* е неподвижният *belief*. Тоест, въпросът „В кое състояние очакваме да сме в първия момент?“ съвпада с въпроса „В кое състояние очакваме да сме в произволен момент?“

Предимство на обозримия свят е, че при него обратната вероятност не зависи от t. Също така за всяко състояние на базовият модел можем да дефиниране вероятност и тази вероятност също няма да зависи от t.

2.9.1 Възможно бъдеще

Възможното бъдеще е последователност от действия, наблюдения и множества от некоректни ходове. Това са крайни множества и затова възможното бъдеще е дума над азбука с $n+m+2^m$ букви.

За всяко състояние s ние можем да опишем възможното бъдеще като безкрайно дърво съдържащо всички думи, които могат да се получат като бъдеще, ако тръгнем от s. На всеки връх в това дърво ще съответства дума (възможно бъдеще) и вероятност (вероятностен интервал).

Вероятността е равна на произведението на вероятностите, които сме събрали в Simple MDP модела тръгвайки от s и прочитайки тази дума. (На стрелките съответстват действия и наблюдения, а на състоянията от U съответстват множества от некоректни ходове.) Ако тази дума може да се прочете по повече от един начин, тогава вероятността е равна на сумата от различните вероятности, които биха се получили по различните начини.

В случая, когато вероятността е интервал, трябва да кажем как се събират и умножават вероятностни интервали.

Умножение:
$[a_1, b_1] \,.\, [a_2, b_2] = [a_1.a_2, b_1.b_2]$

Събиране:
$[a_1, b_1] + [a_2, b_2] = [a_1+a_2, min(b_1+b_2, 1)]$

Стрелките по действия не могат да са повече от една (по едно действие, излизащи от един връх), но стрелките по наблюдения могат. Когато всички ходове са коректни, тогава дървото е просто и стрелките по наблюдения също не могат да се разклоняват. Когато има некоректни ходове дървото е по-сложно, защото от един връх може да излизат няколко стрелки по едно и също наблюдение, но те ще отиват във върхове, на които съответстват различни множества от възможни ходове.

Ще считаме, че на никоя от стрелките не отговаря вероятност нула. Ако има такава вероятност ще отстраним това поддърво. Ако позволим да има стрелки с вероятност нула, тогава дървото няма да е единствено.

2.9.2 Възможно минало

Искаме да дефинираме възможното минало по същия начин както дефинирахме възможното бъдеще. Тук обаче имаме проблем. За всяка стрелка ще ни трябва обратната вероятност. Това е вероятността да сме дошли от тази стрелка. (Ние вече имаме нормалната вероятност, но тя е нещо друго, тя е вероятността да сме тръгнали по стрелката.)

Можем да дефинираме обратната вероятност чрез статистиката на света. Тя ще бъде *k/m*, но тук *m* ще бъде колко пъти сме минали през главата на стрелката (а не през опашката ѝ). Проблемът е, че когато дефинирахме нормалната вероятност ние предположихме, че тя е постоянна и не зависи от *t* (т.е. не зависи от това на коя стъпка сме), а когато дефинираме обратната вероятност може да се окаже, че тя зависи от *t*.

За да сметнем обратната вероятност ще ни трябва нормалната вероятност и още нещо. Това още нещо е вероятността на всяко от състоянията. Тази вероятност в първия момент ние представяме чрез начален *belief*. За всеки следващ момент ние можем да изчислим следващия *belief* чрез предишния *belief* и нормалната вероятност.

Формулите за целта са следните:

$$m_i . p_i = m . q_i$$

$$q_i = \frac{m_i . p_i}{m}$$

$$m = \sum m_i . p_i$$

Тук q_i са обратните вероятности на някакво състояние *s*. Индексът *i* пробягва предишните състояния (тези от които излиза стрелка влизаща в *s*). Вероятностите p_i са нормалните вероятности на стрелките излизащи от предишните състояния и влизащи в състоянието *s*. Вероятностите m_i са получени от предишния *belief*. Това са вероятностите да сме в някое от предишните състояния в предишния момент, а *m* е вероятността да сме в състоянието *s* в следващия момент.

Първото равенство го получаваме от това, че вероятността да излезем от едно състояние по една стрелка е равна на вероятността да влезем в следващото състояние по същата стрелка. Горните формули определят еднозначно обратните вероятности q_i винаги освен в един случай. Това е случаят, когато *m=0*. В този случай можем да поставим на q_i каквито си искаме стойности (стига сумата на q_i да е равна на единица).

По този начин за всяка стъпка *t* получаваме *belief* и обратни вероятности, които зависят от стъпката *t*. Възможното бъдеще дефинирахме като едно безкрайно дърво, а възможното минало ще бъде много по-сложно. То ще бъде изброимо множество от дървета. За момента *0* няма да имаме минало (т.е. дърво с дълбочина *0*). За момента *t* възможното минало ще се представя чрез дърво с дълбочина *t*.

Бихме искали възможното минало също да е просто както възможното бъдеще. За целта ни трябва очакването за това в кое състояние сме да е постоянно и да не зависи от стъпката *t*.

2.9.3 Минало vs Бъдеще

В тази статия миналото и бъдещето са напълно симетрични. Идеята за тази симетрия идва от факта, че представяме живота като път в един граф. Няма разлика, дали ще вървим напред по посока на стрелките или ще вървим назад в обратна посока.

На пръв поглед изглежда, че има разлика между вероятностите напред и назад, но всъщност и там няма разлика. Вероятностите напред са постоянни, а тези назад зависят от *t*, но това е защото първо задаваме вероятностите напред и по тях определяме вероятностите назад. Ако бяхме подходили наобратно (ако бяхме задали първо вероятностите назад и по тях да определим вероятностите напред), тогава вероятностите назад щяха да са постоянни, а тези напред щяха да зависят от *k*-*t*. Разбира се, за да вървим наобратно би трябвало да зададем заключителен вместо начален *belief*. Тоест, ще трябва да тръгнем от някакъв заключителен *belief*.

По-долу ще въведем понятието „Обозрим свят“. Това ще е специален свят, при който вероятностите напред и назад не зависят от *t*. Тоест, ще покажем, че в интересния случай миналото и бъдещето са съвсем симетрични.

Има и други статии, където миналото и бъдещето са симетрични. Например в Guelev and Moszkowski (2021) е така, но там симетрията не е пълна, защото в Guelev et al. (2021) времето може да се разклонява напред, но не и назад. Тоест, възможното бъдеще е дърво, но възможното минало е без разклонения.

2.9.4 Свободна воля

Предположихме, че светът и агентът могат да имат свободна воля. Тоест, предполагаме че светът и агентът могат да правят каквото си искат (но в някакви граници). Описваме света чрез статистиката и затова играчите трябва да използват свободата си, защото ако те не я използват моделът няма да я отчете.

Ако светът и агентът не използват напълно свободата си, чрез статистиката ще получим стеснен модел на света, който ще отрази реалното поведение на играчите.

Например, при играта шах светът и агентът имат някакви коректни ходове и могат да играят всеки един от тях. В действителност, те не играят всеки коректен ход. Например, можем да предположим, че те не правят елементарни грешки. Ако чрез статистиката направим модел на играта шах, вероятно ще получим модел, в който вероятностните интервали са по-тесни. Тоест, ще получим модел, в който възможните ходове са по-малко. Това ще отрази реалното поведение на агента и на неговия противник (има възможни ходове, които те двамата никога не играят).

В играта шах има едно правило: „Не е позволен ход, ако след този ход, противникът ти може да ти вземе царя.“ Може да се каже, че такъв ход би бил елементарна грешка и че правилата на играта забраняват тази елементарна грешка. Тоест, това правило намалява броя на коректните ходове. Ако забраним и други елементарни грешки, ще поучим още по-стеснен модел, в който поведението на света и агента е още по-ограничено.

2.9.5 Неподвижен *belief*

Искаме началният *belief* да е такъв, че следващият *belief* да е същият и всеки следващ *belief* да е същият. Тоест, искаме началният *belief* да е неподвижен (неподвижна точка).

Дефиниция: Капан ще наричаме група състояния, в които може да се влезе, но връщане назад няма.

Ако в света няма капани, тогава всяка компонента на свързаност е силно свързан граф. Нека да предположим, че във всяка компонента на свързаност сме избрали една силно свързана компонента и нека тя да е главната (ще я наречем ядрото на компонентата на свързаност). В света няма капани точно тогава когато няма вливащи се и изтичащи пътища.

Дефиниция: Вливащ се път (influx) ще наричаме група състояния, от които може да се влезе в ядрото, но връщане назад няма.

Дефиниция: Изтичащ път (outflow) ще наричаме група състояния, в които може да се влезе от ядрото, но връщане назад няма.

За да има неподвижен *belief* трябва вероятностите върху вливащите се и изтичащите пътища да са нула. Освен това трябва вероятността да влезем в изтичащ път да е нула, защото в противен случай постоянната вероятност ще изтече в изтичащият път. Това е причината, поради която е по-добре да предполагаме, че тези пътища ги няма. Щом вероятността им е нула е все едно, че ги няма.

2.9.6 Обозрим свят

Първо ще кажем какво е компактен свят:
Дефиниция: Компактен свят ще бъде такъв, в който няма вливащи се и изтичащи пътища.

Дефиниция: Обозрим свят ще бъде такъв, който:
1. Между всеки две състояния има път (силно свързан граф).
2. Вероятностният интервал на всяка от стрелките е различен от нула (т.е. може да е *[0, p]*, но не може да е *[0, 0]*).
3. Началният *belief* на света е множеството от всички неподвижни фиксирани *beliefs*.

Първото условие означава, че обозримият свят е компактен и че е с една компонента на свързаност. Второто условие гарантира, че във всяко състояние можем да попаднем с някаква вероятност, при известно съдействие на света и на агента (чрез тяхната свободна воля). Третото условие означава, че тези два въпроса имат един и същи отговор:

1. От кое състояние очакваме да тръгнем?
2. Кое е състоянието, в което очакваме да се намираме, след много дълъг живот, без значение от кое състояние сме тръгнали?

Ще предполагаме, че светът е компактен и дори че е обозрим. Предимството на компактния свят е, че обратните вероятности са еднозначно определени и възможното минало е просто (то е едно дърво). Предимство на обозримия свят е, че можем еднозначно да определим неподвижния *belief* и по този начин по-простичко да дефинираме разширения модел. В противен случай очакването да се намираме в определено състояние ще зависи от *t* и разширеният модел ще е по-сложен.

В Dobrev (2000) предположихме, че търсим ИИ, който ще се справи добре в световете, в които няма фатални грешки. Предположението в Dobrev (2000) беше, че ако се справя добре в такива светове, ще се справи добре и в произволен свят. Същото предположение можем да направим и за обозримите светове.

Можем ли да кажем, че ако един свят е обозрим, то в него няма фатални грешки. По-долу ще видим, че ако света е обозрим и ако света не е враждебен, то тогава никоя грешка не е фатална. Какво значи света да е враждебен. Казахме, че света има известна свободна воля и може да избере своята фиксирана стратегия измежду определено множество от фиксирани стратегии. Ако света е враждебен той може целенасочено да избира своята стратегия срещу агента. Тогава агента може да попадне във вдлъбнатина и да не може да излезе от нея, защото светът не го пуска да излезе. Ако светът не е враждебен (т.е. ако е безпристрастен или е благосклонен и се опитва да помага) тогава агентът не може да попадне във вдлъбнатина, от която да не може да излезе и тогава няма да има фатални грешки.

2.9.7 Единствен *belief*

Кога неподвижният фиксиран *belief* е единствен? Ще разгледаме по-простия случай, когато имаме фиксирана стратегия (т.е. вероятностите p_i са фиксирани). Нека още вероятностите p_i да са различни от нула. Нека още светът да е обозрим.

В този случай имаме единствен неподвижен *belief* (той ще е фиксиран). Можем да го изчислим като решим система от уравнения. Друг начин за получаването на този фиксиран *belief* е чрез статистиката на света. Можем да вземем един много дълъг живот. Няма значение от кое състояние ще сме тръгнали. Тогава нека m е вероятността да сме в състоянието s. Тази вероятност ще клони към това, което ни дава този *belief*. Не става дума за вероятността в момента t, а за средната вероятност, защото вероятността може да се движи на вълни и вероятността в момента t може да не е сходяща, но средната вероятност е сходяща.

Тоест, получихме един неподвижен *belief*, който отговаря на естествения въпрос: В кое състояние очаквате да се намира света?

Нека света сега да не е обозрим, а да е компактен и да има два компонента на свързаност. Тогава неподвижният *belief* няма да е единствен защото може да разпределим вероятността между двата компонента на свързаност в произволно съотношение (например, *1:1* или *1:2*).

Нека сега света да е обозрим, но във фиксираната стратегия, която сме избрали да има вероятности равни на нула. Тогава тази стратегия разделя света на вдлъбнатини с плоско дъно и околности на вдлъбнатините. Попадайки на дъното на вдлъбнатина повече не можем да излезем. Ако сме тръгнали от околност на вдлъбнатина ще попаднем на дъното на тази или на някоя друга вдлъбнатина. В този случай неподвижният *belief* няма да е единствен, защото в околностите на вдлъбнатините вероятността ще е нула (тя ще е разпределена по дъната). Проблемът е, че вероятността може да се разпредели по различните дъна в произволно съотношение.

Как в този случай да направим един единствен *belief*? Ще разгледаме границата, когато вероятностите p_i са различни от нула, но клонят към тези, които имаме. Така ще намерим

естественото разпределение на вероятностите върху различните дъна. Така намираме единствен неподвижен *belief* при фиксирана стратегия.

Ако вземем множеството на всички фиксирани стратегии (които са в тези вероятностни интервали) ще получим множество от фиксирани *beliefs* и това ще е обобщения *belief,* който ще ни отговаря на въпроса „В кое състояние очаквам да съм?“.

Този обобщен *belief* ни дава за всяко състояние на базовия модел по един вероятностен интервал. Този интервал ще ни даде вероятността да се намираме в дадено състояние на базовия модел. Ще означим този интервал с функцията *expected(s)*.

2.9.8 Разширен модел

Казахме, че в базовия модел състоянията знаят всичко полезно и нищо излишно. В разширения модел ще добавим една излишна информация. Ще добавим целия живот (гръбнака и следата на живота). Тази информация е „излишна“, защото състоянието на базовия модел „знае“ какво е възможното минало и възможното бъдеще, но не знае какво точно е било миналото и какво точно ще бъде бъдещето.

Множеството от състоянията на разширения модел ще бъде S'.

$$S' = \{ < t, L> \mid 0 \leq t \leq \mathrm{k},\ L \text{ is possible life}\}$$

Тук $k=|L|$. Какво наричане „възможен живот“? Това е произволен краен път в графът, който описва базовия модел. (Всяко състояние може да е началното, защото предположихме, че светът е обозрим.)

Множеството S' ще го наречем множеството на моментите, защото всеки момент от всеки възможен живот е елемент на това множество.

Ако разгледаме разширения модел като граф, той ще има много проста структура състояща се от непресичащи се нишики. Всяка нишка ще отговаря на един възможен живот.

Състоянието на разширения модел ще знае точно какво се е случило и какво ще се случи, но това не означава, че ако знаем миналото и модела ще можем да кажем какво ще е бъдещето. Вярно е, че състоянието „знае“ всичко, но ние няма откъде да знаем точно в кое състояние сме.

Когато $t=0$, тогава началният *belief* ни определя в кое състояние на разширения модел сме. Новият начален *belief* ще го получим от множеството S_0' като на всяко състояние добавим вероятността *expected(<0, L>)* (тези дефиниции са дадени по-долу).

$$S_0' = \{ < 0, L > \mid L \text{ is possible life} \}$$

Тоест, новият начален *belief* ще съдържа началния момент на всеки възможен живот. Това са много елементи, защото дори и да фиксираме началното състояние u_0, то елементите в новия начален *belief* пак ще са изброимо много, защото възможните животи започващи със u_0 са изброимо много (заради различните дължини и заради разклоненията). Тоест,

състоянието $<0, L>$ „знае“ всичко за бъдещето, но ние няма да го знаем, защото няма откъде да знаем, че се намираме точно в състоянието $<0, L>$.

Функцията *expected*, която е дефинирана за състоянията на базовия модел, ще я продължим, като я дефинираме за моментите (състоянията на разширения модел). За целта ще вземем вероятността на живота L и ще я разделим на броя на моментите в L.

Искаме сумата от вероятностите на всички пътища да е единица и затова ще предположим, че вероятността пътят да е с дължина k е $(1-\lambda).\lambda^k$. (Тоест, разпределяме единицата върху всички възможни дължини на пътя.) Тук ще изберем един коефициент λ, но изборът на този коефициент не е съществен, защото предположихме, че събитията запазват верността си при разширяване на живота. (Формулата за *expected* ще зависи от L, но не и от t, защото вероятността е разпределена равномерно между моментите от живота.)

$$expected(< t, L >) = \frac{expected(s_0).p_1.p_2.\ldots.p_k}{k+1}.(1-\lambda).\lambda^k$$

Тук $k=|L|$, s_0 е първото състояние от живота L, p_i е вероятността на стрелката от s_{i-1} към s_i. Нека q_i е обратната вероятност на стрелката от s_{i-1} към s_i. Тогава вероятността на един живот с дължина k да бъде животът L можем да запишем по три начина:

$$expected(s_0).p_1.p_2.\ldots.p_k$$
$$q_1.q_2.\ldots.q_k.expected(s_k)$$
$$q_1.\ldots.q_t.expected(s_t).p_{t+1}.\ldots.p_k$$

Тоест, можем да започнем от вероятността на което и да е от състоянията от гръбнака на живота и да го умножаваме по вероятността да се премине към следващото (или към предишното) състояние.

Дефиниция: Интерпретация на събитието A ще наричаме две множества P и Q, където P са моментите когато A се случва, Q са моментите когато A не се случва, а в останалите моменти не знаем дали A се случва.

Дефиниция: Вероятността на събитието A ще бъде:

$$expected(A) = \frac{expected(P)}{expected(P) + expected(Q)}$$

Тоест, вероятността се определя от моментите, в които знаем каква е стойността на събитието.

Предположихме, че събитията запазват верността си при разширяване на живота, което означава, че предполагаме, че множествата P и Q са затворени спрямо операцията разширяване на живота.

Дадената от нас дефиниция на събитие зависи от избрания базов модел. При друг базов модел, други ще са множествата P и Q. Въпреки това събитието ще е същото, защото ще има същата вероятност и ще се държи по същия начин спрямо следата на живота.

2.9.9 MDP as ED модел

Като приложение на горното ще покажем, че MDP е частен случай на ED модел. В MDP моделите наблюдаваните събития са действията на агента:

$$act(t)=a_i,\ 1 \le i \le m$$

Тези събития не се пресичат и затова нямаме нужда от правила за разрешаване на колизиите. Допълнителните събития (тези, които участват в частичната следа) са текущите наблюдения:

$$ob(t)=o_i \vee ob(t+1)=o_i,\ 1 \le i \le n$$

Тук не е ясно дали t е четно или нечетно и дали текущото наблюдение току-що се е случило или ще се случи на следващата стъпка. Затова събитието се определя с дизюнкция.

2.9.9.1 Fully observable Markov decision process

Първо ще разгледаме Fully observable MDP (FOMDP). В този случай топологията на ED модела е проста. Той има n състояния (колкото са възможните наблюдения) и между всеки две състояния, по всяко действие, има стрелка. Следата на ED модела е ясна и хубава. Във всяко състояние има едно текущо наблюдение, което се случва с вероятност единица (всяко състояние се разпознава еднозначно по текущото наблюдение). Единственото, което трябва да добавим, за да определим модела FOMDP са вероятностите по стрелките:

$$p_{iaj} = expected(ob(t+2)=o_j \mid ob(t)=o_i \ \&\ act(t+1)=a\ ) \quad (1)$$

Тоест, следващото наблюдение е o_j, при условие че предишното наблюдение е било o_i и последното действие е било a. Трябва да кажем, какво е събитие при условие. Това събитие ще е истина или лъжа, когато условието е изпълнено и ще бъде „не знам“ когато условието не е изпълнено.

Функцията *expected* може да върне фиксирана вероятност или да върне вероятностен интервал. Когато в (1) *expected* връща фиксирани вероятности ще получим класическата дефиниция на FOMDP. Разбира се, нищо не ни пречи да обобщим дефиницията на FOMDP и да позволим да има преходи, които не са точна вероятност.

Забележка: FOMDP има една единствена еднозначна интерпретация и тя се определя от събитието „Кое е текущото наблюдение?“. За всеки момент от живота ние знаем кое е текущото наблюдение с изключение на първия момент (момента нула). Това, че не знаем кое е текущото наблюдение в първия момент, не е проблем, защото интерпретацията не е нужно да бъде дефинирана за краен живот. Тя трябва да е дефинирана върху всеки безкраен живот, а в безкрайния живот няма първи момент.

Така получихме модел с n състояния и матрица p_{iaj} на вероятностите на преходите. Този модел е известен в литературата като Fully observable MDP. Изниква въпросът дали този модел изпълнява свойството на Марков, тоест дали моделът е съвършен (най-добрият възможен модел, който да не може повече да се подобри).

От една страна е добре моделът да изпълнява свойството на Марков, защото това означава, че сме намерили най-доброто възможно решение. От друга страна това не е добре, защото означава, че сме достигнали една граница, която не можем да преминем и оттук нататък не можем повече да подобряваме модела.

За всеки модел ние ще предполагаме, че той може и да изпълнява свойството на Марков, но че е по-вероятно да не го изпълнява. Тоест, ще предполагаме, че за този свят това може и да е най-добрият възможен модел, но по-вероятно е да има и по-добри модели.

2.9.9.2 Partially observable Markov decision process

Това ще е следващият модел, който ще разгледаме. Топологията отново е проста. Имаме произволен брой състояния и стрелки по всяко действие между всеки две състояния. Имаме вероятности по стрелките, но имаме и вероятности в следата. За всяко състояние на ED модела имаме n особености. Във всяко състояние всяко наблюдение ще се наблюдава с някаква вероятност и тази вероятност в общия случай ще е различна от средната.

Въпросът е дали така описаният Partially observable MDP (POMDP) има интерпретация. Той може да има, а може и да няма. (Ако това е един неадекватен модел, тогава няма да има интерпретация.) Затова ще разгледаме множеството на възможните интерпретации и за всяка интерпретация ще определим вероятностите по стрелките и в следата.

Ще разгледаме еднозначните интерпретации. Тоест, ще разбием множеството S' на класове на еквивалентност (S' е множеството на моментите). Ще съпоставим на всяко състояние на ED модела по един клас на еквивалентност при това разбиване.

Забележка: Релацията на еквивалентност R не е съвсем произволна, защото R трябва да запазва прехода по наблюдение. Тоест, за всеки нечетен момент $<t, L>$ трябва той и следващият момент $<t+1, L>$ да бъдат в един и същи клас на еквивалентност. Достатъчно условие за запазването на прехода по наблюдение е да направим R на базата на някоя релация на еквивалентност върху множеството O (множеството на стрелките по наблюдение).

Нека събитията C_i са тези, които определят класовете на еквивалентност на R.

$$p_{iaj} = expected(C_j(t+2) \mid C_i(t) \,\&\, act(t+1)=a\,)$$

Тоест, следващото състояние е j, при условие че предишното състояние е било i и последното действие е било a.

Каква ще е следата на POMDP? За всяко състояние i имаме n възможни наблюдения и всяко от тези наблюдения се случва с някаква вероятност q_{ij}.

$$q_{ij} = expected(ob(t)=o_j \mid C_i(t)\,)$$

Както казахме, функцията *expected* може да върне фиксирана вероятност или да върне вероятностен интервал. Ако искаме да получим класическата дефиниция на POMDP, ще трябва да предположим, че p_{iaj} и q_{ij} са фиксирани вероятности.

Забележка: Ако се ограничим със стандартната дефиниция за Partially observable MDP, тогава единствената възможна следа на модела ще са вероятностите q_{ij}. Ние обаче ще предположим, че може да има и други следи, т.е. други особености на състоянието C_i. Например може в това състояние определено действие да не може да се случи. Друга възможност за следа е определена конюнкция от събития да е невъзможна, макар събитията от конюнкцията поотделно да са възможни.

Забележка: Изниква въпросът дали светът има съвършен Partially observable MDP (такъв, който изпълнява свойството на Марков). Имаме много различни POMDP (например, за всяка релация на еквивалентност над O имаме такъв модел). Въпросът е има ли измежду всичките тези модели един, който да е съвършен. Отговорът е: „Да има“. Този модел ще го получим, ако вземем всички стрелки от O и за всяка стрелка да направи събитието „минавам по тази стрелка“. Това събитие ще е истина във всеки нечетен момент, в който тръгваме по стрелката, както и във следващия момент. Така полученият POMDP ще е съвършен и това следва от съвършенството на базовия модел. Този модел няма да е минимален, но ако обединим стрелките, които са с едно и също начало и един и същи край, тогава ще получим POMDP, който ще е съвършен и минимален (това следва от съвършенството и минималността на базовия модел).

2.9.10 Related work

Идеята за създаването на ED моделите е, че състоянията на света са твърде много и е добре да ги намалим, за да направим модела по-разбираем. Същата идея е залегнала и в Wang, Joshi and Khardon (2008). Там се въвеждат Relational MDP, които по същество са частен случай на ED моделите.

Казахме, че всяка релация на еквивалентност (която разбива множеството на стрелките) може да послужи за създаването на POMDP. Идеята за Relational MDP е да се изберат някои събития. (Вместо за събития, те говорят за predicates, actionsins and rewards.) Чрез тези събития се дефинира релация на еквивалентност (две състояния са еквивалентни, ако в тях се случват едни и същи събития от избраните). Върху фактор множеството на тази релация се дефинира POMDP и това е търсения Relational MDP.

Забележка: При дефиницията в Wang et al. (2008) не се използва фактор множество, а се говори за семейство от MDPs. Ако всеки един от тези MDPs има само по едно състояние от всеки един от класовете на еквивалентност, тогава той ще има структурата на фактор множеството, но в общия случай това не е така. Затова по-добре е Relational MDP да се дефинират чрез фактор множеството.

Как изглежда Relational MDP? Ще разгледаме три случая:

1. Нека входа да се състои от 10 бита. Нека тези 10 бита да бъдат десетте събития, които са избрани и чрез тях да е направен Relational MDP. Тогава този Relational MDP съвпада с FOMDP.
2. Нека сега само 5 от тези 10 бита да са избрани. Тогава ще получим Relational MDP с по-малко състояния от FOMDP.
3. Нека сега са избрани десетте бита на входа и още пет събития, които са невидими (не следват от стойността на входа). Нека с тези 15 събития да направим Relational MDP. Сега полученият Relational MDP ще има повече състояния от FOMDP.

2.10 Описание на играта шах

2.10.1 Компютърна емулация

Светът на играта шах сме го емулирали с компютърната програма (Dobrev, 2020a), която е написана на езика (Dobrev, 2020b). Тази програма използва правилата на играта, които са представени като ED модели.

Когато стартирате програмата (Dobrev, 2020a) в долната част на екрана ще видите, това което е изобразено на фигура 4.

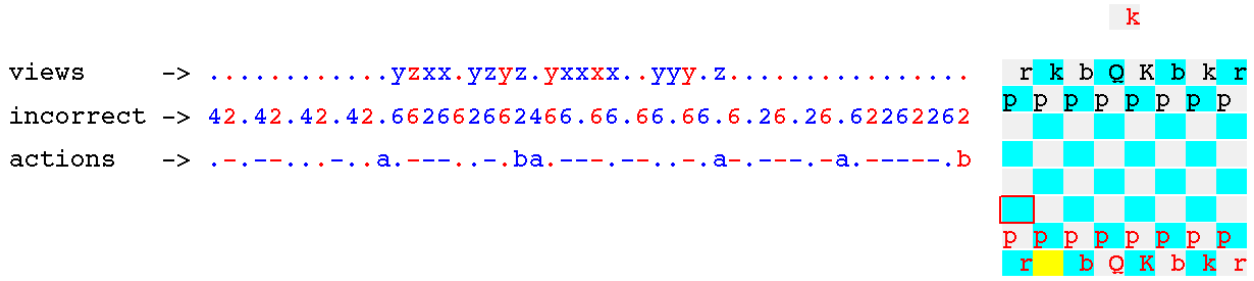


Figure 4

В лявата част се вижда потока входно-изходна информация. Това не е целият поток, а само последните 50 стъпки. На първия ред са наблюденията на агента, а на третия ред са действията му. Възможните наблюдения са четири {0, x, y, z}. Възможните действия също са четири {0, a, b, c}. За по-добра четливост нулата и символът „c“ са заменени с точка и с минус. На втория ред се вижда кои действия са позволени в конкретния момент (2 означава, че действието ***a*** е некоректно, 4 означава, че ***b*** е некоректно, 6 означава, че и ***a*** и ***b*** са некоректни).

Това, което е в лявата част на фигура 4, е това, което вижда агентът. Това, което е в дясната част на фигурата, агентът не го вижда, но трябва да си го представи, за да разбере света. В дясно се вижда позицията на таблото, вижда се коя фигура агентът е вдигнал (коня), вижда се и от къде я е вдигнал (жълтото квадратче), вижда се и кое е наблюдаваното в момента квадратче (ограденото с червена линия).

2.10.2 Използваме кодиране

Агентът ще може да извършва 8 действия. Той ще може да мести поглед (квадратчето, което наблюдава) в четирите посоки. Ще може да вдигне фигурата, която вижда и да пусне вдигната фигура в квадратчето, което вижда в момента. Седмото и осмото действие е да не прави нищо.

Ще ограничим действията на агента до четири букви {0, a, b, c}. Символите 0 и „c“ ще използваме за действията „не върша нищо“. Как с оставащите два символа ще опишем 6 действия? Това ще стане чрез кодиране. Ще разделим стъпките на три. На всяка първа стъпка ще казваме как движим квадратчето по хоризонтала (т.е. как движим прозореца ни на наблюдение). На всяка втора стъпка ще казваме как движим квадратчето по вертикала, а на всяка трета стъпка ще казваме дали вдигаме фигура или пускаме вдигнатата фигура.

В Dobrev (2013) споменахме, че трябва да избягваме излишното кодиране, защото светът е достатъчно сложен и няма нужда допълнително да го усложняваме. Тук обаче не става

дума за излишно кодиране, защото с това кодиране светът не се усложнява, а става по-прост, защото заменяме осем действия с четири.

2.10.3 Две празни действия

Защо въвеждаме две действия, които означават „не върша нищо“? Всъщност, когато агентът стои и не върши нищо, той наблюдава света. Въпросът е дали той ще е пасивен наблюдател или ще наблюдава активно?

Когато вие стоите и наблюдавате света, вие не сте пасивен наблюдател. Най-малкото е, че вие движите поглед.

Всички зависимости, които може да види пасивният наблюдател са периодични. В известен смисъл периодичните зависимости са малко и не са особено интересни. Много по-интересни са зависимостите, които може да види активния наблюдател.

Очакваме агентът да може да забележи определени зависимости (свойства). Например, вида и цвета на фигурите са такива свойства. Когато агентът седи в едно квадратче и не върши нищо, ще му е трудно да хване зависимостта (свойството), още повече, че може да му се наложи да хваща две или три зависимости едновременно. Ако агентът е активен и може да редува две действия, тогава зависимостите, които наблюдава ще са много по-ясни и по-бързо откриваеми.

За да различим двете действия „не върша нищо“, второто сме го нарекли „оглеждам“.

2.10.4 Едно, две, три

Първата зависимост, която ще съществува в този конкретен свят (света на играта шах) идва от това, че разделихме стъпките на три. Тази зависимост ще наречем „Едно, две три“. Моделът на тази зависимост е изобразен на фигура 5.

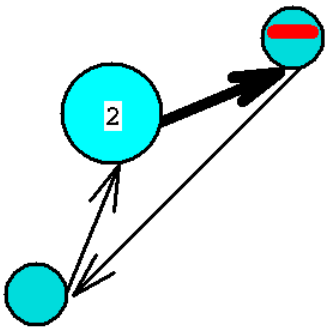


Figure 5

Какво представлява тази зависимост? Тя брои: едно, две, три.

Тази зависимост се представя с един Event-Driven модел. В конкретния случай това е модел с три състояния. В този модел имаме само едно събитие и това е събитието „винаги“ (тоест, „истина“ или „на всяка стъпка“).

2.10.5 Следа

Дали в състоянията на гореописания модел се случва нещо специално, което да можем да забележим и което да ни помогне да открием този модел? Тоест, има ли „следа“ (това е терминология въведена в Dobrev (2018)).

Да, в третото състояние задължително едно от действията ***a*** или ***b*** е некоректно (или и двете). Това е така, защото в третото състояние ние казваме дали вдигаме фигурата, която виждаме или пускаме вдигната вече фигура. Няма как тези две действия да са възможни едновременно.

Можем и без тази следа да опишем света, но без нея зависимостта „Едно, две три“ би била много по-трудно откриваема. Затова е добре, че имаме някаква следа в този модел.

Следата е това специалното, което дава смисъла на модела. Например, в хладилника има студена бира и това прави хладилника един по-специален шкаф. Ако във всички шкафове имаше студена бира, тогава хладилника нямаше да е по-специален и щеше да е все едно кой шкаф ще отворим.

Следата ни дава възможност да предскажем какво ще се случи. Ако отворим хладилника, очакваме вътре да има студена бира. Освен това, следата ни помага да познаем в кое състояние сме и да ограничим недетерминираността. Например, отваряме бял шкаф, но не знаем дали това е хладилника или друг бял шкаф. Ако вътре има студена бира, тогава ще разберем, че сме отворили хладилника и по този начин ще ограничим недетерминираността.

Ще разглеждаме два вида следа – постоянна и подвижна. Постоянна следа ще са специалните неща (явления), които всеки път се случват, а подвижна следа ще са нещата, които се случват временно.

Например, да си представим една къща като един Event-Driven модел. Състоянията на този модел ще са стаите. Нещо постоянно за стаите ще е броят на вратите. Временни явления, които се явяват и изчезват са „светла“ и „топла“. Тоест, постоянната следа може да ни каже коя стая е преходна, а подвижната следа ще ни каже, коя стая в момента е топла.

Стаите могат да са свързани с различни обекти. Тези обекти си имат свойства (явленията, които се наблюдават, когато наблюдаваме съответния обект). Обектите могат да са постоянни или подвижни и съответно техните свойства ще са относително постоянни или временни явления (относително постоянно е това явление, което в дадено състояние се явява винаги). Пример за постоянни обекти са мебелите (особено по-тежките). Пример за подвижни обекти са хората и животните. Тоест, постоянната следа ще описва това което е постоянно, а подвижната следа ще описва това, което е временно.

2.10.6 Horizontal и Vertical

Следващият Event-Driven модел, който ще ни е нужен за описанието на света е моделът „Horizontal“ (фигура 6).

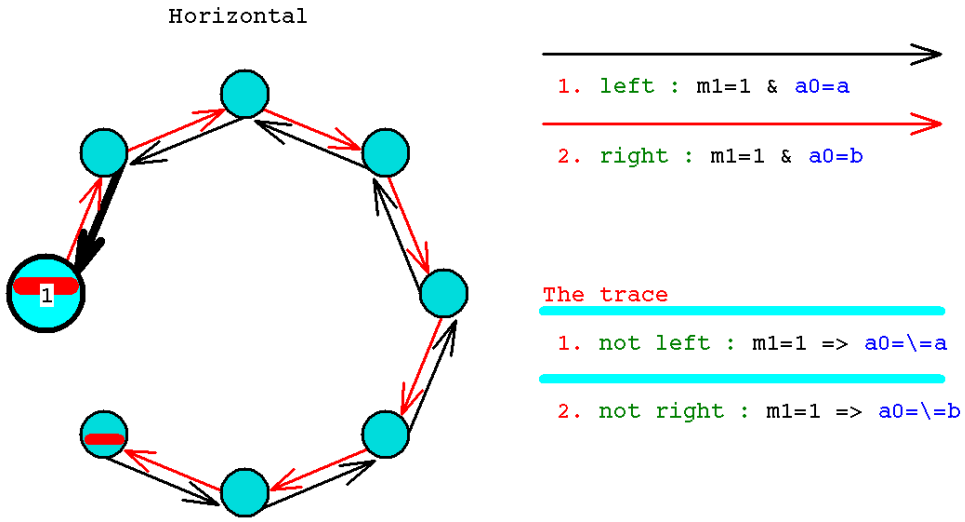


Figure 6

Този модел ще ни даде отговор на въпроса, в коя колона на шахматната дъска се намира квадратчето, което наблюдаваме.

Тук имаме две събития и това са събитията „наляво“ и „надясно“. Тоест, агентът мести поглед наляво или надясно. Тоест, той извършва действията ***a*** и ***b***, когато моделът 1 е в състоянието 1. Имаме и две следи. В състоянието 1 не може да се играе наляво. Тоест, когато сме в състояние 1 събитието „наляво“ не може да се случи. Аналогично, е със състоянието 8 и следата, че там не може да се играе надясно. Тези две следи ще направят моделът откриваем. Например вие, ако сте в тъмна стая широка 8 стъпки, ще установите, че след седем стъпки наляво по-наляво не може. Ще го установите, защото ще се блъснете в стената. Тоест, сблъсъка със стената е следата в случая. Такъв сблъсък ще има само на първата и на последната позиция.

Тази следа, освен че ще ни помогне да открием модела, тя ще е полезна още за да ни обясни света. Как иначе бихте си обяснили защо в най-лявата колона не можете да играете „наляво“?

Съвсем аналогичен на модела „Horizontal“ е моделът „Vertical“ (фигура 7).

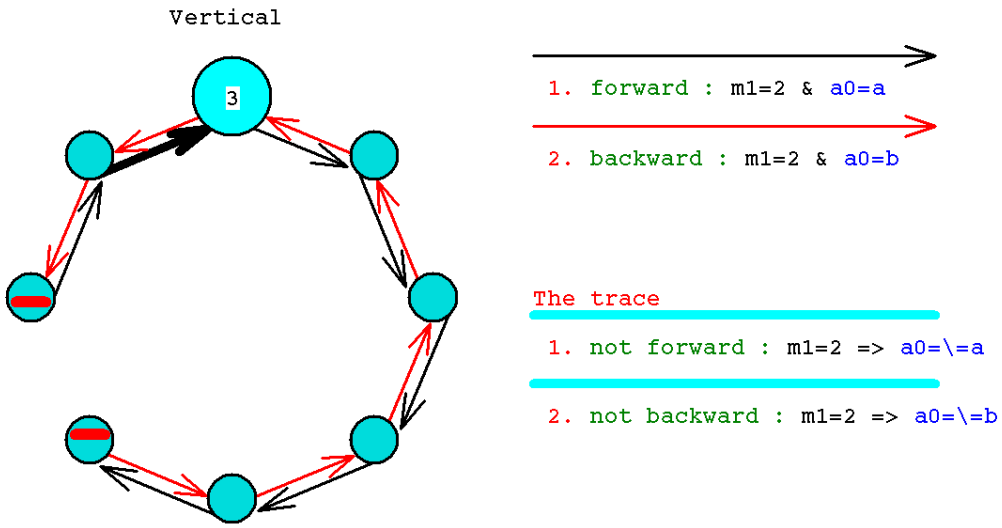


Figure 7

Този модел ще ни каже в кой ред се намира квадратчето, което наблюдаваме. Аналогично имаме две събития („напред“ и „назад“), както и две следи („не може напред“ и „не може назад“)

Логично е да направим декартовото произведение на горните два модела и да получим модел с 64 състояния, който ще отговаря на шахматното табло.

Лошото е, че в това декартово произведение няма постоянна следа. Тоест, нищо специално не се случва в някое от квадратчетата. Случват се разни работи, но те не са постоянни, а временни. Например, в едно квадратче може да виждаме бяла пешка и това да е сравнително постоянно, но не е напълно постоянно, защото пешката може да се премести.

Стигаме до извода, че следата може и да не е постоянна.

2.10.7 Подвижна следа

Както казахме, „подвижна следа“ ще са специалните неща, които се случват в едно състояние, но не се случват постоянно, а само временно.

Как да изобразим подвижната следа? Постоянната следа изобразявахме, като отбелязвахме върху състоянието дали някакво събитие винаги се случва в това състояние (винаги отбелязваме с червено, а със синьо отбелязваме, когато никога не се случва).

Подвижната следа ще изобразим като масив, който има толкова клетки, колкото състояния има съответния модел. Във всяка клетка ще запишем подвижните следи, които в момента са в съответното състояние. Тоест, масивът на подвижната следа ще мени стойностите си.

Ето как ще изглежда масива на подвижната следа на декартовото произведение на втория и третия модел:

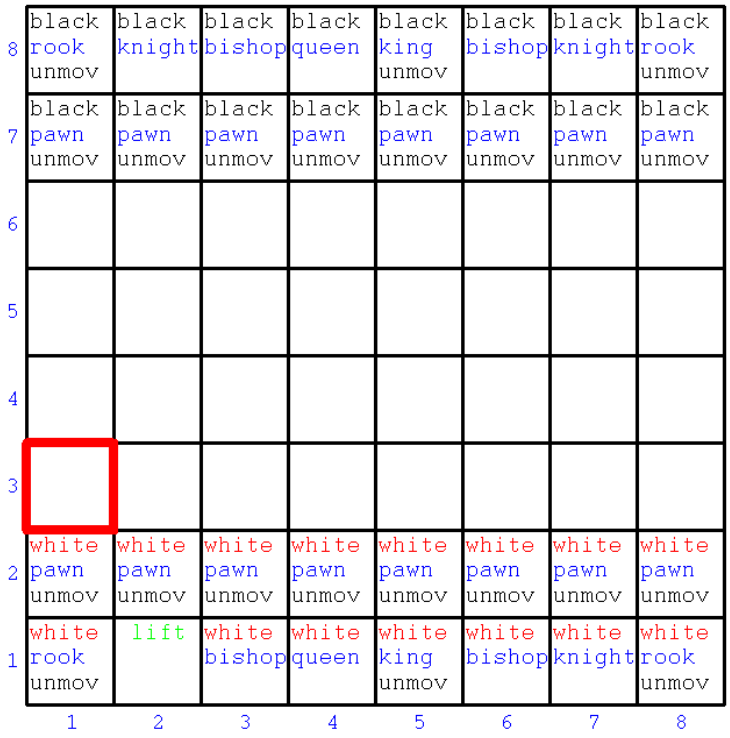


Figure 8

Тази подвижна следа е много сложна, защото това е подвижната следа на модел с 64 състояния. Нека да вземем подвижната следа на модел с две състояния (фигура 9). Това е моделът 4, който помни дали сме вдигнали фигура. Неговата подвижна следа ще помни коя е вдигнатата фигура. Разбира се, този модел освен подвижна следа си има и постоянна, която казва, че в състоянието 2 не може „нагоре“, докато в състоянието 1 не може „надолу“.

Подвижната следа на този модел представлява масив с две клетки, които съответстват на двете състояния на ED модела. Клетката, която съответства на текущото състояние е отбелязана, като е оградена с червена линия. Не е толкова важно какво има в клетката съответстваща на текущото състояние, а това което е в другите клетки, защото те ни казват какво ще се случи, когато някоя от другите клетки стане текуща. В случая, ако пуснем вдигната фигура ще отидем в състоянието 1 и там ще видим вдигната фигура. (Ще видим това, което сме пуснали. В случая ще видим „бял кон“.)

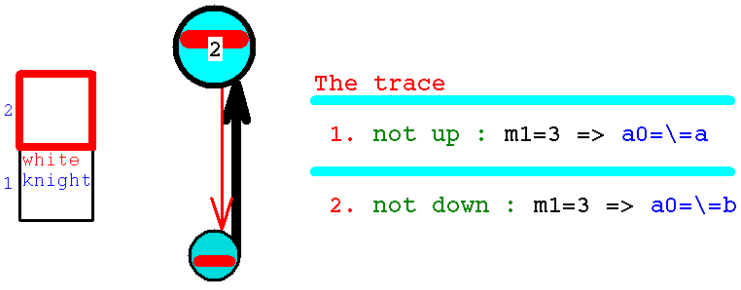


Figure 9

Казахме, че езикът за описание на светове ще ни каже кое е текущото състояние на света. Къде се записва това състояние? Записва се на две места. Първо, това е текущото състояние на всеки от ED моделите и второ, това е подвижната следа. Например на фигура 8 виждате как чрез подвижната следа се представя позицията на шахматната дъска.

Ако езика за описание на светове беше стандартен език за програмиране, неговата памет щеше да е стойността на променливите и на масивите. Тук можем да направим аналогията, че текущото състояние на един ED модел е стойността на една променлива, а стойността на една подвижна следа е стойността на един масив.

Стойността на текущото състояние на един ED модел обикновено е едно число, ако моделът е детерминиран, но може да е няколко числа, ако ED моделът има няколко текущи състояния (стойността може да е *belief* , ако различните състояния си имат различна вероятност). Стойността на всяка от клетките на подвижната следа ще се състои от няколко числа, защото в едно състояние може да има много подвижни следи. Разбира се, постоянните следи също може да са повече от една.

2.11 Алгоритми

След като описахме основните правила на играта шах и позицията на таблото, следващата стъпка е да кажем как се движат фигурите. За целта ни е нужно понятието алгоритъм.

Обикновено в литературата не се прави разлика между алгоритъм и изчислима функция. Това не е правилно, защото алгоритмът е действие, а изчислимата функция е резултата от това действие. Трябва да правим разлика между действие и резултат. Например приготвянето на палачинки е нещо различно от палачинките. Резултатът от алгоритъма зависи от това в кой свят го изпълняваме. Например, алгоритъма за приготвяне на палачинки в друг свят може да даде друг резултат. Този друг резултат може да бъде, например, изчислима функция или космическа ракета.

2.11.1 Какво е алгоритъм?

За повечето хора алгоритмът това е машина на Тюринг. Това е така, защото те разглеждат само функциите от $\mathbb{N}$ в $\mathbb{N}$ и за тях алгоритъм е нещо което изчислява такава функция. За нас алгоритмът ще описва последователност от действия в произволен свят. Например за нас алгоритми ще са готварските рецепти, танцовите стъпки, умението да се хване топка и т.н. Казахме последователност от действия. Нека се коригираме и да стане последователност от събития. Действието е събитие, но не всяко събитие е действие или поне не е наше действие, а може да е действие на някой друг агент. В описанието на алгоритъма освен наши действия ще има и други събития. Например, чакаме докато водата кипне. Кипването на водата е събитие, което не е наше действие.

При нашата дефиниция алгоритмът може да се осъществи въобще без нашето участие. Да вземем като пример Лунната соната. Това е алгоритъм, който ще изпълним, ако изсвирим Лунната соната, но ако я изсвири някой друг, тогава това пак ще е алгоритъм, но изпълнен от някой друг. Ако разпознаем Лунната соната, ние ще сме разпознали този алгоритъм, нищо че не го изпълняваме.

Няма да е много важно кой изпълнява алгоритъма. Нормално е един алгоритъм първо да ни го покаже някой друг, после да го изпълним и ние.

Ще разгледаме три варианта на алгоритъм:
1. Релсов път.
2. Планинска пътека.
3. Отивам си вкъщи.

При първия вариант ще предполагаме, че имаме ограничения, които не ни позволяват да се отклоним от изпълнението на алгоритъма. Например, когато се качим на рейса, ние пътуваме по маршрута и не можем да се отклоним, защото друг кара рейса. Когато слушаме Лунната соната, отново нищо не можем да променим, защото не свирим ние.

При втория вариант ние можем да се отклоним, но има последствия, ако се отклоним. Планинската пътека минава покрай пропаст. Ако се отклоним, ще паднем в пропастта. При третия вариант можем да се отклоним от пътя. След отклонението можем отново да се върнем в пътя, а може и да минем по друг път. Алгоритъма за прибиране у дома ни казва, че ако го изпълним, ще сме си вкъщи, но по никакъв начин не сме задължени да го изпълним или да го изпълним точно по този начин.

Обикновено, когато говорим за алгоритъм предполагаме детерминираност. Представяме си компютърна програма, при която за всеки следващ момент се знае точно кое ще е действието, което ще бъде извършено. Вече дори и компютърните програми не са еднонишкови. При многонишковите програми не е съвсем ясно кое ще е следващото действие, което ще бъде извършено. Още по-ясен е примерът с готварските рецепти. Когато правим палачинки, не е казано дали първо да сложим яйцата и после млякото или обратното. И в двата случая ще изпълним един и същ алгоритъм.

Представете си алгоритъма като движение в пещера. Можете да вървите напред, но можете да се върнете и назад. Галерията има разклонения и вие имате избор на къде да завиете. Само, ако излезете от пещерата, ще сте прекратил изпълнението на алгоритъма „движа се в пещерата“. Тоест, ще си представяме алгоритъма като ориентиран граф с много разклонения, а не като път без разклонения.

2.11.2 Алгоритъма на фигурите

С алгоритмите ще опишем движението на фигурите. Ние ще изберем варианта „релсов път“ (първият от разгледаните варианти). Тоест, когато вдигнете фигура ще се включва съответният алгоритъм, който няма да ви позволи да направите грешен ход.

Можеше да изберем и варианта „планинска пътека“. Тоест, да може да се отклоните от алгоритъма, но това да е с последствия. Например, вдигате фигурата и започвате да изпълнявате алгоритъма, но ако го нарушите, ще изпуснете вдигната фигура и тя ще се върне на мястото си.

Можеше да изберем и варианта „отивам си вкъщи“. При този вариант се движите както пожелаете, но можете да поставите фигурата само на тези места, където алгоритмът би могъл да я поставил, ако беше изпълнен. Тоест, имате пълна свобода на движението, а алгоритмът само ви дефинира кои ходове са коректните.

Ще изберем първият вариант главно защото сме пуснали агента да играе случайно и ако не го вкараме в релси, за него ще е много трудно да изиграе коректен ход. Освен това трябва да си помислим за това как агентът ще разбере света. Как ще ги открие тези алгоритми?

Ако го вкараме в релси, той ще научи алгоритъма по неволя, но ако го оставим свободно да се движи за него ще е много трудно да отгатне какви са тези правила на движение (какви са тези алгоритми). Например, ако покажете на един ученик алгоритъма за намиране на корен квадратен, то на него ще му е сравнително лесно да го научи. Много по-трудно би му било, ако му обясните какво е корен квадратен го оставите сам да намери алгоритъма за изчисляването му. Можете да покажете на ученика какво е корен квадратен с дефиниция или с примери, но по-лесно ще ви разбере, ако му покажете директно алгоритъма.

Какво ще представляват алгоритмите? Това ще са Event-Driven модели. Ще има някакво събитие, което ще е вход и което ще стартира алгоритъма и още някакво събитие, което ще е изход и след което алгоритъма ще престане да се изпълнява. По-нататък ще направим изходите да са два (успешен и неуспешен изход).

Всяка фигура ще си има алгоритъм:

2.11.3 Алгоритмите на царя и на коня

Най-простият алгоритъм ще бъде алгоритъма на царя (фигура 10). Входящото събитие ще бъде вдигам фигурата цар. Входящата точка ще бъде състоянието 1 (при всичките алгоритми това ще бъде входящата точка). Събитията ще са 4 (наляво, надясно, напред и назад).

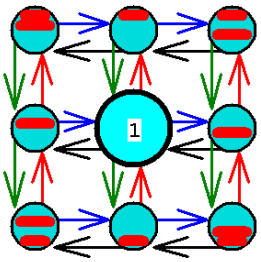


Figure 10

Следата ще се състои от четири събития (не може наляво, не може надясно и т.н.) Тези четири събития (следи) ще ограничат движението до девет квадратчета. Тези 4 събития (следи) ще са релсите, в които ще влезем и които няма да ни позволят да напуснем деветте квадратчета докато изпълняваме алгоритъма. На фигура 10 четирите следи са отбелязани с червени хоризонтални линии. Например, горните три състояния имат първата следа, което значи, че от тези три състояния не може напред.

Ще можем да пуснем вдигната фигура (царят) във всеки момент, когато пожелаем. Разбра се, може да има други правила или алгоритми, които да ни ограничават. Например, не можем да вземем собствена фигура, тоест има и други ограничения, но те не идват от този алгоритъм. Ако пуснем фигурата в състоянието 1, тогава ходът ни няма да е истински, а ще е фалшив. Ако пуснем фигурата в някое от другите състояния, тогава ще сме изиграли един истински ход.

Малко по-сложен е алгоритмът на коня (фигура 11). Основната разлика с алгоритъма на царя е, че тук има още една следа. Тази следа ни ограничава и в някои от състоянията няма да можем да спускаме вдигната фигура. (Тази следа е отбелязана на фигура 11, а другите 4 следи не са отбелязани.) Спазвайки този алгоритъм ние имаме само две възможности.

Първата е да изиграем коректен ход с коня, а втората е да изиграем фалшив ход като върнем коня там откъдето сме го взели.

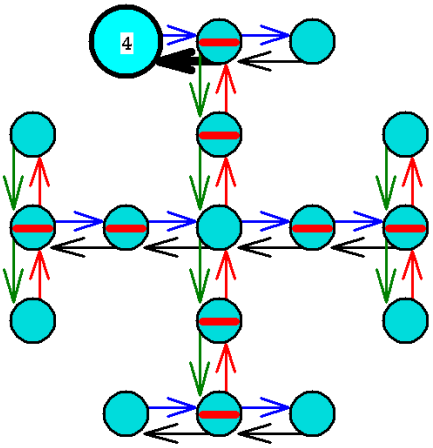


Figure 11

2.11.4 Алгоритмите на топа и на офицера

Макар че има малко състояния алгоритмът на топа е по-сложен (фигура 12). Причината за това е, че този алгоритъм е недетерминиран. Например в състоянието 3 когато играем „напред“, тогава има две стрелки които отговарят на това събитие. Съответно има две състояния, които могат да са следващите. Тази недетерминираност веднага се разрешава, защото в състоянието 1 задължително трябва да се вижда, че от това квадратче е вдигната фигура, докато в състоянието 3 задължително това не трябва да се вижда. Тоест, имаме следа която веднага разрешава тази недетерминираност.

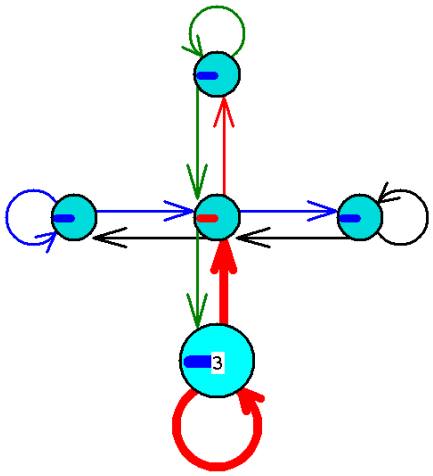


Figure 12

Алгоритмът на офицера е още по сложен (фигура 13). Основната причина за това е, че не можем да се придвижим директно по диагонала, а за целта трябва да направим две стъпки (първата по хоризонтала и втората по вертикала). Когато в състоянието 1 се случи събитието „наляво“, тогава ние не знаем дали сме тръгнали по диагонала „наляво и напред“ или по диагонала „наляво и назад“. Тогава се получава недетерминираност, която не може да бъде разрешена незабавно. Все пак, тази недетерминираност ще се разреши когато дойде едно от събитията „нанапред“ или „назад“. В двете възможни състояния имаме следи, които ни казват, че в състояние 8 не може „напред“, а в състояние 2 не може „назад“. Ако и в двете състояния не можеше „напред“, то тогава събитието „напред“ би

нарушило алгоритъма. В случая, в едното състояние може, а в другото не може. Тоест, събитието „напред“ е разрешено, но когато то се случи състоянието 8 ще престане да бъде активно и недетерминираността ще се разреши. (На фигура 13 сме отбелязали само следите „не може напред“ и „не може назад“.)

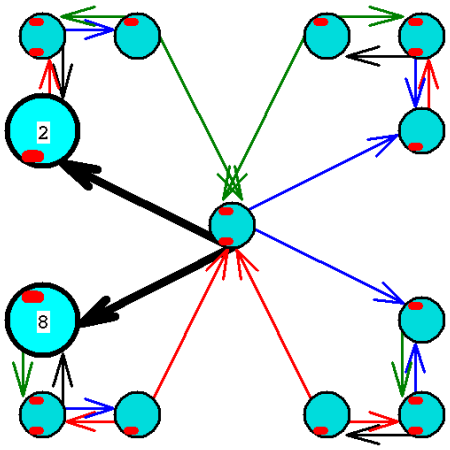


Figure 13

Най-сложен е алгоритмът на царицата, защото той е съчетание от алгоритмите на топа и на офицера. Алгоритмът на пешката не е сложен, но имаме четири такива алгоритъма, защото имаме различни алгоритми за бяла и за черна пешка, като и за преместена и за непреместена пешка.

2.11.5 Машината на Тюринг

Описахме алгоритмите на движение на шахматните фигури като Event-Driven модели. Можем ли да приемем, че всеки алгоритъм може да се представи като Event-Driven модел? Дали машината на Тюринг може да се представи по този начин?

Ще опишем един свят, който представлява машина на Тюринг. Първото нещо, което трябва да опишем в този свят е безкрайната лента. В играта шах описахме шахматното табло като подвижната следа на някакъв Event-Driven модел с 64 състояния. Тук отново ще използваме подвижната следа, но ще ни е нужен модел с изброимо много състояния. Да вземем модела от фигура 6. Това е модел на лента с осем клетки. Трябва ни същият модел, пак да има две събития (наляво и надясно), но да не е ограничен отляво и отдясно. Получава се Event-Driven модел с безкрайно много състояния. Досега използвахме ED модели само с крайно много състояния. Сега ще ни се наложи да добавим и някои безкрайни ED модели, но които имат проста структура като този. В случая моделът представлява просто един брояч, който помни едно цяло число (т.е. елемент на $\mathbb{Z}$). Броячът има две операции (минус едно и плюс едно) или (наляво и надясно). Добавянето на този безкраен брояч разширява езика, но както казахме ние ще разширяваме езика, за да покрием световете, които искаме да опишем.

Каква ще е паметта на този свят? Трябва да запомним стойността на брояча (коя клетка от лентата гледа главата на машината). Това е произволно цяло число. Освен това ще трябва да запомним и какво има записано върху лентата. За целта ще ни трябва безкрайна последователност от нули и единици, което е равномощно на континуум. Обикновено използваме Машините на Тюринг, за да изчисляваме функции от $\mathbb{N}$ в $\mathbb{N}$. В този случай бихме могли да се ограничим само с конфигурации, при които е използвана само крайна

част от лентата, тоест бихме могли да се ограничим само с изброимо много конфигурации, но всички възможни конфигурации на лентата са континуум много.

Забележка: Представата на агента за състоянието на света ще е изброима дори ако паметта на света е континуум. Казано по друг начин, агентът няма как да си представи всички възможни конфигурации върху лентата, а само изброима част от тези конфигурации. При горното разсъждение използваме това, че си мислим агента като абстрактна машина с безкрайна памет. Ако агента си го мислим като реален компютър с крайна памет, тогова в горното разсъждение трябва да заменим „изброима“ с „крайна“. Все пак, ако агентът е програма на реален компютър, то тази крайна памет е толкова голяма, че за по-просто ще си я мислим за изброима.

Описахме безкрайната лента на машината на Тюринг с един безкраен ED модел. За да опишем главата на машината (самия алгоритъм) ще ни трябва още един ED модел. Втория ED модел ще го построим използвайки машината на Тюринг.

Предположихме, че машината използва две букви {0, 1}. Ще направим Event-Driven модел с четири събития:

- *write(0),*
- *write(1),*
- *move left,*
- *move right.*

Тогава всяка от командите на машината ще изглежда така:

- if observe(0) then write Symbol_0, move Direction_0, goto Command_0
- if observe(1) then write Symbol_1, move Direction_1, goto Command_1

Тук Symbol_i, Direction_i и Command_i са заменени с конкретни стойности. Например:

- if observe(0) then write(1), move left, goto s_3
- if observe(1) then write(0), move right, goto s_7

Всяка команда ще заменим с четири състояния, които ще я опишат. Горната команда ще изглежда така:

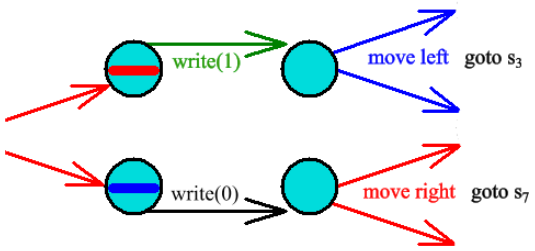


Figure 14

На фигура 14 входа е по събитието „move right“. Всъщност, ще се влиза от много места, понякога по събитието „move left“, а понякога по събитието „move right“. Важното е, че входът ще е недетерминиран, но веднага тази недетерминираност ще се разреши, защото

първите две състояния имат следа. В горното задължително трябва да се случи събитието „observe(0)“, а в долното задължително това събитие не трябва да се случва.

Тоест, всяко от състоянията на автомата се заменя с четири състояния, както е показано на фигура 14, след това отделните четворки се свързват помежду си. Например, четворката от фигура 14 се свързва с четворката съответстваща на s_3 чрез стрелки по събитието „move left“ и с четворката съответстваща на s_7 чрез стрелки по събитието „move right“.

Трябва да добавим още малко следа. За всяко едно от състоянията е възможно само едно от четирите събития. Трябва да добавим като следа, че другите три събития са невъзможни. Това е, ако искаме алгоритмът да е от тип „релсов път“. Ако предпочитаме да е от тип „планинска пътека“ трябва добавим следа, която да казва че ако се случи някое от другите три събития, ще настъпят съответните последствия. Ако искаме типът да е „отивам си вкъщи“, тогава другите три събития трябва да водят до прекратяване на алгоритъма.

По този начин представихме машината на Тюринг с Event-Driven модел. По-точно с два ED модела, първия с безкрайно много състояния и втория с краен брой (четири пъти повече от състоянията на машината).

Кой изпълнява алгоритъма на машината на Тюринг? Може да предположим, че четирите събития са действия на агента и че той е този, който изпълнява алгоритъма. Може да предположим, че тези събития ги изпълнява друг агент или че те просто се случват. Тогава агента не изпълнява алгоритъма, а е само наблюдател. В общият случай, една част от събитията на алгоритъма ще са действия на агента, а останалата част няма да са. Например, „сипвам вода“ е действие на агента, а „водата завира“ не е негово действие. Агентът може да влияе и на събитията, които не са негови действия. Това е описано в Dobrev (2019b). Спрямо тези събития той може да има някакво „предпочитание“ и чрез това „предпочитание“ той би могъл на влияе на това дали тези събития ще се случат.

2.11.6 Related work

Важно е, че в тази статия е дефинирано понятието алгоритъм. Много малко са хората, които въобще си задават въпроса какво е алгоритъм. Единствените опити за дефиниция на алгоритъм, които са ми известни са направени от Moschovakis (2001; 2018). В тези трудове Moschovakis казва, че повечето автори дефинират алгоритъма чрез някаква абстрактна машина и отъждествяват алгоритмите с програмите за тази абстрактна машина. Moschovakis формулира каква дефиниция на алгоритъм на нас ни е необходима. Той иска да създаде едно общо понятие, което да не зависи от конкретната абстрактна машина. Такова понятие е изчислимата функция, но това понятие е твърде общо за Moschovakis и той иска да направи по-специализирано понятие, което да отразява това, че една изчислима функция може да се изчисли от много принципно различни алгоритми. В Moschovakis (2001) не е постигната високата цел поставена от Moschovakis. Това, което той е направил може да се приеме за една нова абстрактна машина. Наистина тази машина е много интересна и е по-абстрактна от повечето известни машини, но отново имаме недостатъка, че програмата на машината може безсмислено да се усложни като се получи друга програма реализираща същия алгоритъм. Макар, че в Moschovakis (2001) не се постига целта да бъде създадена обща дефиниция на алгоритъм, самият Moschovakis казва, че за него по-важното е да постави въпроса, дори и да не успее да му отговори. Точните

думи на Moschovakis са: „my chief goal is to convince the reader that the problem of founding the theory of algorithms is important, and that it is ripe for solution.“

2.12 Обекти

2.12.1 Свойства

След понятието алгоритъм ще се опитаме да дефинираме още едно фундаментално понятие. Това ще е понятието свойство. За дефиницията на това понятие отново ще използваме Event-Driven модели. Свойствата са явленията, които се наблюдават когато се наблюдава обект със съответното свойство. Явленията са зависимости, които не се наблюдават постоянно, а само от време на време. Щом другите зависимости се представят с Event-Driven модели, естествено е и свойствата да се представят по същия начин.

Разликата между зависимост и свойство ще бъде, че зависимостта ще е активна постоянно (т.е. ще се наблюдава постоянно) докато свойството ще се наблюдава понякога (когато наблюдаваме съответния обект).

2.12.2 Какво е обект

Базовото понятие ще е свойство, а обектът ще е абстракция от по-висок ранг. Например, ако в света на играта шах се наблюдават свойствата „бял“ и „кон“, може да се направи извода, че има обект „бял кон“, който се наблюдава и който има тези две свойства. Може и да не стигаме до тази абстракция и да си мислим, че просто някакви свойства се местят. Тоест, че някакви явления се появяват и изчезват.

2.12.3 Второ кодиране

Изходът на агента се състои само от четири букви и затова използвахме кодиране за да представим осемте възможни действия на агента. Входът също е ограничен до четири букви. Вярно е, че входът трябва да ни даде информация само за едно от квадратчетата, а не за цялото табло. Въпреки това четири букви са твърде малко, защото в квадратчето може да има шест различни фигури с два различни цвята. Освен това, трябва ни допълнителна информация като това дали пешката е местена и дали от този квадрат не е вдигнатата фигура. Как да представим всичката тази информация с четири букви?

Тази информация не е задължително да идва до агента само за една стъпка. Той може да постои известно време върху квадратчето и да наблюдава входа. Той може да забележи различни зависимости докато наблюдава квадратчето. Наличието или отсъствието на всяка от тези зависимости ще е информацията, която ще получи агентът за квадратчето, което наблюдава. Макар буквите на входа да са само четири, зависимостите, които могат да се опишат с четири букви са безбройно много.

Тези зависимости ще наречем свойства и ще предполагаме, че агента може да разпознава (да хваща) тези зависимости. Ще предполагаме още, че той може да хване няколко зависимости, дори когато те са една върху друга. Например, агентът трябва да може да хване свойствата „бял“ и „кон“ дори когато те се проявяват едновременно.

Как изглеждат свойствата? Зависимостите и алгоритмите за движение на фигурите са написани от човек, който има идея какви са правилата на играта шах и как се движат фигурите. Свойствата не са написани от човек, а са генерирани автоматично. Например на фигура 15 е изобразено свойството „пешка“. Това свойство изглежда доста странно и нелогично. Това е така, защото, както казахме, то е генерирано автоматично по случаен начин. Това свойство не е написано от нас, защото ние не знаем как би изглеждала

пешката. Не е важно как изглежда тя. Важното е пешката да изглежда по някакъв начин и да може тя да бъде разпозната от агента. Тоест, пешката трябва да си има лице, но не е важно как ще изглежда нейното лице.

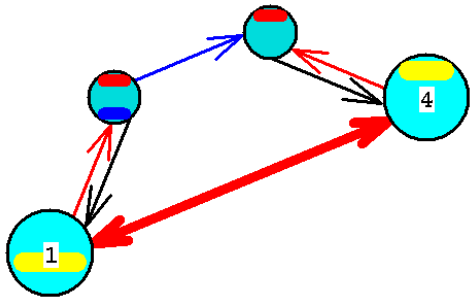


Figure 15

В нашата програма (Dobrev, 2020a) има 10 свойства и всяко едно от тях си има някаква следа. Когато няколко свойства са активни едновременно, тогава всяко едно от тях влияе (чрез следата си) на входа на агента. Понякога тези влияния могат да бъдат противоречиви. Например, едно свойство ни казва, че следващият вход трябва да е буквата ***x***, а друго свойство ни казва обратното (че не трябва да е ***x***). Тогава въпросът се решава с гласуване. Светът брои колко гласа има за всяко решение и избира решението, което е събрало най-много гласове. Що се отнася до противоречивите препоръки, те взаимно се обезсилват.

2.13 Шах с двама играчи

Ще усложним света на играта шах, като добавим още един агент. Това ще е противникът, който играе с черните фигури. Това ще доведе до недетерминираност, защото няма да можем да кажем точно как ще играе противникът. Дори противникът да е детерминиран, тази детерминираност може да е прекалено сложна и да не можем да я опишем.

2.13.1 Детерминиран свят

Описахме света на играта шах където агентът играе сам срещу себе си. Написали сме програмата (Dobrev, 2020a), която съдържа просто описание на този свят и чрез това описание го емулира. Можете да стартирате тази програма и да видите колко просто се е получило описанието на този свят (на играта шах). Описанието се състои от 24 модула, които представляват Event-Driven модели (това са ориентирани графи с по десетина състояния всеки). ED моделите са три вида (5 зависимости, 9 алгоритъма и 10 свойства, което прави общо 24). Освен ED моделите имаме още две подвижни следи (т.е. два масива). Освен 24-те модула и двата масива ни се е наложило да добавим още седем прости правила, които допълнително описват света. Тези правила ни дават допълнителна информация за това как се променя състоянието на света. Например, първото от тези правила ни казва, че ако вдигнем фигура, на нейното място ще се появи свойството „Lifted“. Правилото изглежда така:

$$up, here \Rightarrow copy(Lifted)$$

Ако вдигнем фигура и ако сме в квадратчето $<X, Y>$, тогава свойството „Lifted“ ще замести свойствата, които са в същото това квадратче в момента.

Тези правила ние можем да формулираме благодарение на това, че вече имаме контекста на шахматната дъска (подвижната следа от фигура 8). Ако не знаехме за съществуването на тази дъска, нямаше как да формулираме правила за поведението на фигурите върху

дъската. В демонстрационната програма (Dobrev, 2020a) агентът играе случайно (random). Разбира се, действията на агента не са интересни. Интересното е света и това, че ние сме го описали.

Описанието, което получихме, е детерминирано. Тоест, началното състояние е определено и всяко следващо състояние е определено. Детерминирано описанието означава, че в описания свят няма случайност. Трябва ли описанието на света да е детерминирано? Да се ограничим ли само с такива описания? Въобще не е сигурно, че светът е детерминиран, а дори и да е такъв, не е нужно да се ограничаваме само с детерминирани описания.

Ако опишем недетерминиран свят с детерминистично описание, то много скоро това описание ще покаже своето несъвършенство. Обратното, света може да е детерминиран, но тази детерминираност да е твърде сложна и да не можем да я разберем (да я опишем). Затова може вместо детерминистично описание на света да намерим едно недетерминистично, което да работи достатъчно добре.

Обикновено светът е недетерминиран. Когато стреляме по мишена може да не уцелим. Това означава, че не всяко наше действие води до резултат и че резултатите понякога могат да бъдат различни.

Ще допускаме, че моделът може да е недетерминиран. Повечето автори, когато говорят за недетерминираност, предполагат, че всяко възможно събитие има точно определена вероятност. В Dobrev (2018) и в Dobrev (2019b) показахме, че това последното е твърде детерминирано. Би било твърде силно изискването за всяко събитие да можем да кажем точната вероятност, с която то ще се случи. Затова ще предполагаме, че не знаем точната вероятност, а знаем само интервала *[a, b]*, в който е тази вероятност. Обикновено интервалът ще е *[0, 1]* и тогава няма да имаме никаква идея с каква вероятност ще се случи събитието.

2.13.2 Невъзможни събития

Казахме, че светът би бил по-интересен, ако не играем сами срещу себе си, а ако има още един агент, който да мести черните фигури.

За целта ще променим петия Event-Driven модел (този, който ни казва дали играем с белите или с черните фигури). Този модел има две състояния, които се превключват от събитието „change“. Това събитие беше дефинирано като събитието „real_move“ (това е когато играем реален ход, а „fake_move“ е когато само докосваме някоя фигура). Ще променим дефиницията на това събитие и ще го дефинираме като „never“ (това е обратното на „every time“). Чрез тази промяна получаваме свят, в който агентът не може да смени цвета си.

Има ли смисъл в модела да описваме събития, които няма как да се случат? Отговорът е, че има смисъл, защото тези събития може да се случват мислено. Тоест, тези събития са ни нужни за да разберем света, макар, че те не се случват. Например, ние не можем да летим и да си сменим пола, но мислено можем да го направим. Примерът не е много добър, защото ние вече можем да летим и да си сменим пола. Тоест, ние може да си мислим за невъзможни събития, освен това в един момент тези събития може от невъзможни да станат възможни.

Ще използваме невъзможното събитие „change“, за да добавим правилото, че нямаме право да играем ход, след който ще сме шах (след който могат да ни вземат царя). На фигура 16 е изобразен алгоритмът, който описва как сменяме (обръщаме дъската) и взимаме царя. Ако съществува изпълнение на този алгоритъм, тогава ходът не е коректен. (Дори да съществува изпълнение, алгоритъма не може да бъде изпълнен, защото съдържа невъзможно събитие.)

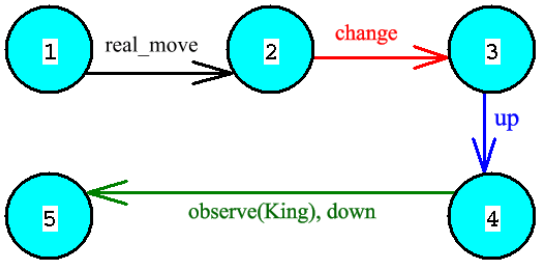


Figure 16

Този алгоритъм в по-голяма степен отговаря на представата ни за това как изглеждат алгоритмите. Докато алгоритмите на фигурите представляваха ориентирани графи с много разклонения, този алгоритъм представлява само един път без разклонения. Тоест, този алгоритъм е просто една последователност от действия без разклонения.

Този алгоритъм се нуждае още от някои ограничения (следи), които не сме отбелязали на фигура 16. Например, в състоянието 1 не можем да се движим в никоя от четирите посоки (иначе бихме могли да се преместим и да изиграем друг ход). Събитието „change“ не може да се случва в никое от състоянията освен състоянието 2. В състоянието 4 имаме ограничението „not observe(King) => not down“. Това последното означава, че единственият ход, който можем да направим, е да вземем цар.

В този алгоритъм участва невъзможното действие „change“. Както казахме, това действие е невъзможно, но можем да го извършим мислено. Това събитие може да участва в дефиницията на алгоритми, които няма да изпълняваме, а за които ще е важно само дали съществува изпълнение.

Забележка: В тази статия, когато казваме, че алгоритъм може да бъде изпълнен, имаме предвид, че той може да бъде изпълнен успешно. Това означава, че изпълнението може да завърши в крайно (приемащо) състояние или с изходящо събитие (с успешен изход).

2.13.3 Втори агент

Алгоритмът от фигура 16 би се опростил, ако допуснем съществуването на втори агент. Идеята е вместо да сменяме цвета на фигурите (да обръщаме дъската), да сменим агента с такъв, който винаги играе с черните фигури. Ще се получи алгоритъм изпълняван от повече от един агент, но такива алгоритми са естествени. Например: „Дадох пари на един човек и той купи нещо с тези пари“. Това е пример за алгоритъм изпълнен от двама агенти.

По важното е, че ще искаме, когато ние преместим бяла фигура, някой друг (друг агент) да премести черна фигура. В предишния случай си задавахме само въпроса „възможен ли е определен алгоритъм“, а тук ще искаме някакъв алгоритъм реално да бъде изпълнен. Не е все едно алгоритмът да е възможен и той реално да бъде изпълнен. Не е все едно „може ли някой да направи палачинки“ или „жена ви да ви направи палачинки реално“. В единия случай знаете нещо за света, а във втория случай реално ядете палачинки. Когато някой

агент реално изпълнява даден алгоритъм, не е все едно кой е агентът, който ще изпълни алгоритъма. Например, предполагаме, че жена ви ще направи палачинките по-добре отколкото вие бихте ги направили.

Ще предполагаме, че след всеки наш „real_move“ агентът, който играе с черните фигури, ще изпълни алгоритъма от фигура 17.

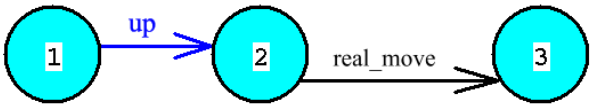


Figure 17

Един алгоритъм не се изпълнява за една стъпка, а за това са нужни много стъпки. Тук обаче ще предполагаме, че противника ще играе с черните фигури веднага (за една стъпка). Хората, когато си мислят, че някой ще направи нещо, обикновено си представят резултата, без да отчитат, че това се извършва в продължение на известно време. Например, когато си мислите: „Днес имам рожден ден и жена ми ще ми направи палачинки“. При това разсъждение вие приемате палачинките за направени без да отчитате, че това отнема време.

Както казахме, не е все едно кой е агентът, който играе с черните фигури. Много важно е дали ни е съюзник или противник (дали ще ни помага или ще ни пречи). Също така, важно е доколко е умен (защото той може да има някакви намерения, но до колко ще ги осъществи зависи от това доколко е умен). Важно е още какво знае и какво вижда агентът. При играта шах предполагаме, че агентът вижда всичко (цялото табло), но в други светове бихме могли да предположим, че агентът знае и вижда само част от информацията. Може да е важно и къде се намира агентът. Тук предполагаме, че това не е важно. Предполагаме, че където и да се намира агентът, той може да се придвижи до произволно квадратче и да вдигне фигурата, която е там. Бихме могли да предположим, че позицията на агента има значение и че за по-близките фигури е по-вероятно да бъдат преместени, отколкото по-далечните.

2.13.4 Собствено състояние

Тук предположихме, че вторият агент си има собствено състояние на света. Тоест, има си собствена позиция $<x, y>$ на таблото (квадратчето, което наблюдава). Също така предполагаме, че той играе с черните, за разлика от главния герой, който играе с белите.

Предполагаме, че двамата агенти променят света по един и същи модел, но паметта на модела (състоянието на света) е различна за двамата агенти. Бихме могли да предположим, че двете състояния на света нямат нищо общо, но тогава действията на втория агент по никакъв начин няма да влияят на света на главния герой. Затова ще предполагаме, че позицията на таблото е обща (т.е. обща е следата от фигура 8). Ще предполагаме, че всеки агент си има собствени координати и собствен цвят, с който играе (т.е. Event-Driven моделите 2, 3 и 5 имат различни активни състояния при двамата агенти). За останалите ED модели, както и за следата от фигура 11 също ще предполагаме, че те са отделни за отделните агенти, макар че нищо не пречи да предположим и обратното.

Ако предполагахме, че двамата агенти споделят едно и също състояние на света, тогава алгоритъма от фигура 17 щеше да е много по-сложен. Противника първо щеше да обърне дъската („change“), после щеше изиграе своя ход и пак да обърне дъската, за да остави

света на главния герой непроменен. Освен това противникът трябваше да се погрижи да се върне на същите координати $<x, y>$, от които е тръгнал (това са координатите на главния герой). Би било много неестествено различните агенти да са съвсем еднакви и да се намират на едно и също място. Много по-естествено е предположението, че агентите са различни и че имат различно състояние на света, но че част от състоянието е обща. Например, „В момента аз правя палачинки и жена ми прави палачинки.“ Може ние да правим едни и същи палачинки, а може моите палачинки да нямат нищо общо с нейните.

Забележка: Не е много точно да казваме, че света има две различни състояния за двата агента. Светът е един и неговото състояние е едно единствено. По-точно ще е да кажем, че сме променили света и вече имаме свят с по-сложно състояние. Нека новото множество от състояния да е S''. Можем да предполагаме, че $S'' = S \times S$. Въпросите, които са общи за двамата агенти са си останали непроменени, но другите въпроси са се раздвоили. Например въпросът „Къде съм?“ е заменен от въпросите „Къде е главният герой?“ и „Къде е противникът?“. От модела, при който състоянията са S, сме направили нов модел, при който състоянията са S''. Разликата между S и S'' е, че състоянията в S описват състоянието на един агент (без да се казва кой е той), докато състоянията в S'' описват състоянието на двата агента. (И в двата случая общото състояние на света също се описва.) Новият модел описва света чрез двата агента и това как те променят състоянието си по първия модел. Въпреки всичко, по-естествено е да си мислим, че светът има различни състояния за двата агента и че тези агенти променят състоянията си по първия модел, който работи само с въпросите, които са само за единия агент.

2.13.5 Неизчислимо правило

Описахме първия свят, в който агентът играеше сам срещу себе си и направихме програмата (Dobrev, 2020a), която емулира този свят. Програмата (Dobrev, 2020a) представлява модел, който е описание на първия свят. Описахме и втори свят, в който агентът играе срещу някакъв противник. Можем ли да направим емулираща програма и за втория свят?

Във втория свят добавихме твърдение от вида „този алгоритъм може да бъде изпълнен“. (Това твърдение трябваше да го добавим още в първия свят, защото и там не е позволено да се играе ход, ако след хода сме шах. За момента програмата (Dobrev, 2020a) позволява да играем такива ходове.) Във втория свят добавихме и операция от вида „противникът изпълнява алгоритъм“. Това твърдение и тази операция в общия случай са неразрешими (по-точно те са полуразрешими).

Да вземем например твърдението „този алгоритъм може да бъде изпълнен“. В конкретния случай става дума за това дали противника може да ни вземе царя и това е напълно разрешимо, защото шахматната дъска е крайна и има крайно много позиции и всички алгоритми работещи над шахматната дъска са разрешими. В общия случай алгоритмът може да бъде машина на Тюринг и тогава това твърдение е равносилно на стоп проблема (halting problem).

Същото може да се каже и за операцията „противникът изпълнява алгоритъм“. Алгоритмът може да се изпълни по много различни начини, но задачата да намерим поне един от тези начини е полуразрешима. В конкретния случай, когато имаме играта шах, лесно можем да намерим един от начините, по които се изпълнява алгоритмът. Тук дори

можем да намерим всички начини (това са всички възможни ходове), но в общия случай тази задача е полуразрешима.

Тоест, в конкретния случай ние можем да напишем програма, която емулира този втори свят. Само трябва да изберем поведението на противника, защото за това поведение има много възможности. С други думи казано, за да създадем програма, която да емулира света на играта шах, трябва вътре в нея да вградим програма емулираща шахматен играч.

В общия случай обаче ние няма да можем да напишем програма емулираща описаният от нас свят. Тоест, езика за описание на светове вече описва такива светове, които няма как да бъдат емулирани с компютърна програма. Още в началото казахме, че моделът може да се получи неизчислим. Няма как да напишем програма, която да изчислява неизчислим модел.

Това, че не можем да напишем програма емулираща описания от нас свят не е голям проблем, защото нашата цел не е да емулираме света, а да напишем програмата ИИ, която на базата на това, че е разбрала света (намерила е описанието му) ще планира успешно бъдещите си ходове. Разбира се, ИИ би могла да процедира като направи една емулация на света и да разиграе няколко от възможните бъдещи развития, като избере това, което е най-доброто. (По същество така работи алгоритмът Min-Max, с който шахматните програми играят.) Тоест, ако можем да направим емулация на света, няма да е лошо, макар и да не е задължително.

ИИ не само, че няма да може да направи пълна емулация на света (когато моделът е неизчислим), но дори ИИ може да не разбере кое точно е текущото състояние на света (когато възможните състояния са континуум много). Въпреки това, ИИ ще може да направи частична емулация и да разбере състоянието на света частично. Например, ако в света има безкрайна лента и върху нея има безкрайно много информация, тогава няма как ИИ да разбере текущото състояние на света, но може да опише някаква крайна част от лентата и информацията върху тази крайна част.

Дори и Min-Max алгоритмът не е пълна емулация, заради комбинаторната експлозия. Вместо това, Min-Max прави частична емулация като обхожда само първите няколко хода. Когато в описанието на света има полуразрешимо правило, тогава ИИ ще използва това правило само в едната посока. Например правилото „Ако съществува доказателство, тогава твърдението е вярно“. Хората използват това правило, когато има доказателство и когато те са го намерили. Когато няма доказателство, тогава това правило не се използва, защото няма как да разберем, че доказателство действително няма.

2.14 Агенти

Следващата абстракция, това е агентът. Също като обектите, агентите няма да можем да ги засечем директно. Тях ще ги наблюдаваме индиректно чрез техните действия.
Откриването на агенти е трудна задача. Хората успяват да открият агенти, но за целта те ги търсят навсякъде. Когато нещо се случи, хората веднага намират обяснение в някакъв агент, който го е извършил. Зад всяко събитие хората виждат като извършител или човек, или животно, или божество. Много рядко приемат, че това се е случило от само себе си.
ИИ трябва да подходи по същия начин като хората и да търси агентите навсякъде.

Когато ИИ намери агент трябва да започне да го изучава и да се опитва да се свърже с него. Да намери агент, значи да го измисли. Когато ИИ измисли съществуващ агент, тогава можем да кажем, че го е намерил. Когато си измисли несъществуващ агент, тогава е по-добре да кажем, че си е измислил нещо несъществуващо. Дали агентите са реални или измислени няма голямо значение. Важното е описанието на света получено чрез тези агенти да е адекватно и да дава добри резултати.

2.14.1 Взаимодействие между агенти

ИИ ще изучава агентите като ги класифицира като приятели и като врагове. Ще отбелязва дали са умни и дали са благодарни (съответно отмъстителни). ИИ ще се опитва да се свързва с агентите. За целта първо трябва да разбере към какво се стреми всеки от тях и да му предложи това, което иска агентът и в замяна да се опита да получи нещо полезно за себе си. Тази размяна на блага се нарича изпълнение на коалиционна стратегия. Обикновено се предполага, че агентите се срещат извън света и там се уговарят каква да бъде тяхната коалиционна стратегия. Тъй като няма как агентите да се срещнат извън света, ние ще предполагаме, че те общуват вътре в света. Принципът на общуването е: „Ще ти направя добро и очаквам да ми го върнеш.“ Другият принцип е: „Аз ще се държа предсказуемо и очаквам ти да разбереш какво е моето поведение и да започнеш да изпълняваш коалиционна стратегия (да се държиш така, че и за двама ни да има полза).“

По този начин ние общуваме с кучетата. Даваме им кокал и веднага се сприятеляваме. Какво получаваме в замяна? В замяна те не ни лаят и не ни хапят, а това никак не е малко. По-нататък може да се достигне до по-сложни комуникации. Може да покажем на агента алгоритъм и да искаме от него той да го изпълни. Така може да научим кучето да дава лапа. Още по-нататък може да се достигне до език като се асоциират обекти с явления. Например, произнесената дума е явление и ако това явление се асоциира с даден обект или алгоритъм, тогава агентът може като чуе думата да изпълни алгоритъма. Например, кучето, като си чуе името, може да дойде при вас или ако чуе „чехли“ може да ви донесе чехлите.

2.14.2 Сигнали между агенти

За да говорим за взаимодействие или за преговори, трябва да имаме някаква комуникация. Тук стигаме до въпроса за подаването на сигнали между агентите. Не става въпрос за предварително уговорени сигнали, а за такива, които един от агентите решава да подава, а другите успяват да отгатнат, на базата на наблюдението, което правят. Като пример ще дадем „Кучето на Павлов“ (Pavlov, 1902). Павлов е агентът, който решава да подава сигнал звънейки със звънче преди да нахрани кучето. Другият агент е кучето, което успява да разбере сигнала.

Когато един агент подава сигнал на друг, не е задължително вторият да разбира, че това е сигнал и че този сигнал е подаден от някой друг, който иска нещо да му каже. Например, кучето на Павлов въобще не разбира, че Павлов е този, който звъни със звънчето и че иска да му каже, че обядът е готов. Кучето просто свързва събитието звънене със събитието храна. Тоест, когато подаваме сигнал, можем да останем анонимни. Тоест, можем да повлияем на другия агент без той въобще да разбира, че някой му влияе.

Друг начин за подаване на сигнал е да покажем нещо (да дадем някаква информация). За да покажем нещо трябва да сме наясно кога и какво вижда другият агент. Например, когато кучето ни се озъби, то ни показва зъбите си. Ние виждаме, че кучето има зъби, а това е факт, който ние по принцип знаем, но виждаме, че кучето е решило да ни напомни

за този факт и разбираме посланието така: „Кучето ни предупреждава, че може да използва зъбите си срещу нас“.

Освен естествените (подразбиращите се) сигнали може да имаме и установени сигнали. Нека имаме група от агенти и между тях да има някакви вече установени сигнали. Когато се появява нов агент, той може да научи сигнала от един от агентите и после да го използва при комуникацията си с другите агенти. Такива сигнали са думите от естествения език. Научаваме думите от един агент (например от майка си) и след това използваме същите думи, за да комуникираме с другите агенти.

2.14.3 Обмен на информация

Когато агентите комуникират, те могат да обменят информация, да съгласуват действията си или да преговарят. Пример за обмен на информация е, когато един агент споделя някакъв алгоритъм с друг агент. Алгоритмът може да бъде описан на естествен език, тоест може да бъде представен като последователност от сигнали (думи), всеки от които се асоциира с обект, явление или алгоритъм. Например, ако искаме да кажем на някого как да стигне до магазина, ние описваме този алгоритъм с думи. Когато казваме „отвори вратата“ разчитаме, че другият агент асоциира думата „врата“ с обекта „врата“ и думата „отвори“ с алгоритъма „отвори“. Тоест, разчитаме, че другият агент знае думите и има представа за обектите асоциирани с тези думи.

Ако приемем, че агентът пази алгоритмите в паметта си под формата на Event-Driven модели, тогава той трябва да може от описанието на естествен език да постои ED модел (ако разбере смисъла), както и обратното, да може да направи описание на естествен език на някакъв ED модел (стига да разполага с нужните думи).

2.14.4 Комуникационен интерфейс

Когато създаваме света на ИИ, трябва да му осигурим комуникационен интерфейс, за да му позволим да общува с другите агенти.

Например, когато създаваме автономно движещ се автомобил, трябва да му осигурим лице, за да може да комуникира с пешеходците и с другите шофьори. Автомобила има клаксон и мигачи, но това не е достатъчно за пълноценна комуникация. Хубаво би било да добавим екран, който да изразява различни емоции. Усмивката и намигването ще са много полезни.

Ние обикновено търсим погледа на другия шофьор, защото за нас е много важно да знаем, че той ни е видял. Ако може това лице (този екран) да се обърне към нас, това би ни показало, че сме забелязани.

2.14.5 Related work

Има много статии, които се занимават с въпроса за взаимодействието между агентите. Тези статии не казват как ИИ ще открие агента, а приемат агента за вече открит и определят правила за разумно взаимодействие. Например в Goranko, Kuusisto and Rönnholm (2020) се разглежда случаят, когато всички агенти са приятели и всички са безкрайно умни. В Goranko et al. (2020) агентите комуникират на базата на това, че се досещат какво би направил другият (разчита се на това, че те са приятели и че са достатъчно умни, за да се сетят кое е от полза за всички). Най-интересното в Goranko et al. (2020) е, че там се поставят въпросите за йерархия между агентите (кой е по-важен) и за

това кой бърза повече (кой колко е търпелив). Това са принципи, които се използват от реалните хора в реалния свят и е логично ИИ също да ги използва.

Взаимодействието между агентите е толкова сложно, колкото взаимодействието между хората. Например в Mell, Lucas, Mozgai and Gratch (2020) агентите преговарят помежду си и дори мога да се лъжат един друг.

В Guelev (2020) се разглежда случая, когато агентите взаимодействат помежду си и образуват коалиции. Дори тези коалиции са временни и могат да се променят по време на играта. За съжаление в Guelev (2020) не се казва как агентите взаимодействат помежду си и как уговарят коалициите, а се предполага, че те се уговарят на някакъв език извън играта (извън света). Тоест, въпросът за уговарянето не е разгледан, а е прието че това се е случило по някакъв начин.

В статията Gurov, Goranko and Lundberg (2021) както и в настоящата статия се разглежда много-агентна система при която агентите не виждат всичко (Partial Observability). Основната разлика между Gurov et al. (2021) и настоящата статия е, че в Gurov et al. (2021) светът е даден (описан е чрез една релация) докато в настоящата статия светът не е даден и това е което се търси.

2.15 По-нататъшна работа

В тази статия ръчно описахме един свят (играта шах) и направихме компютърна програма (Dobrev, 2020a), която на базата на това описание емулира света. Следващата задача, която искаме да решим е обратната. Искаме да направим програма, която автоматично да намери същото това описание на света, което описахме ръчно. Програмата, която ще търси описанието ще използва емулацията на света (Dobrev, 2020a), благодарение на тази емулация програмата ще „живее“ вътре в света и ще трябва да го разбере (т.е. да го опише).

В този случай бихме могли да шмекеруваме, защото правим програма, която трябва да намери нещо, а ние предварително знаем какво е това нещо, което тя трябва да намери. Разбира се, не трябва да шмекеруваме, защото ако го направим ще получим програма, която би разбрала единствено и само този конкретен свят. Хубаво би било направената от нас програма да е в състояние да разбере (да опише) произволен свят. Последното изискване е твърде силно, защото това означава да постоим ИИ. Затова няма да искаме програмата да може да разбере произволен свят, но ще искаме да е в състояние да разбере дадения свят (Dobrev, 2020a) и световете, които са близки до него. Колкото по-голям клас от светове е в състояние да разбере направената от нас програма, толкова по-умна ще е тя.

2.16 Заключение

Задачата е да разберем света. За да го разберем, трябва да го опишем, а за да го опишем ни е нужен специален език за описание на светове.

Сведохме задачата за създаването на ИИ до една чисто логическа задача. От нас сега се иска да създадем език за описание на светове и този език ще е логически, защото на него ще могат да се опишат неизчислими функции. Ако езика описваше само изчислими функции, тогава това щеше да е език за програмиране, а не логически език.

Основните градивни елементи на нашия нов език това са Event-Driven моделите. Това са простите модули, които ще откриваме един по един. С тези модули ще представим зависимости, алгоритми и явления.

Направихме една абстракция като въведохме обектите. Обектите не могат да бъдат наблюдавани директно, а ги засичаме индиректно като наблюдаваме техни свойства. Свойството е специално явление, което се наблюдава когато наблюдаваме обект с това свойство. Тоест, свойството също се представя с ED модел.

Следващата абстракция, която въведохме са агентите и те също не могат да бъдат наблюдавани директно, а ги засичаме индиректно чрез техни действия.

Създадохме език за описание на светове. Това е не е напълно завършен език, а е само неговата първа версия, която се нуждае от допълнително развитие. Не дадохме формално описание на създадения от нас език, а го представихме с три примера. Тоест, вместо формално да описваме езика ние написахме описанията на три конкретни свята. Това са два варианта на играта шах (с един и с двама агенти) и свят, който представя работата на Машина на Тюринг.

Забележка: Не е голям проблем да се направи формално описание на език, което да покрива трите свята, които сме използвали като пример, но целта е друга. Целта е да се направи език, който може да опише произволен свят и този универсален език да се опише формално. Това е по-трудна задача, която не сме решили.

Показахме, че езикът за описание на светове може чрез простите модули, от които се състои, да описва доста сложни светове с много агенти и сложни взаимоотношения помежду им. Това, което надграждаме над простите модули не може да виси във въздуха и трябва да стъпи на някаква стабилна основа. Именно Event-Driven моделите са основата, която ще изгради езика за описание на светове и базата, на която ще изградим всички по-сложни абстракции.

3 Какво ще правим след като го направим?

3.1 AI не трябва да е Open Source Project

Кой трябва да притежава технологията Изкуствен Интелект? Тази технология трябва да е на всички, но не самата технология, а плодовете, които тя ще ни даде. Разбира се, не трябва да позволяваме AI да попада в ръцете на безотговорни хора. Аналогично, ядрените технологии трябва да носят полза на всички, но тези технологии трябва да се пазят в тайна и да не са общодостъпни за всеки.

3.1.1 Въведение

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 (Macron, 2018). 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 (Macron, 2018) 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 (Becquerel, 1896) 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 and Horkheimer (2002) 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 Adorno et al. (2002) 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 Adorno et al. (2002) може да се приеме за казано по темата. В Adorno et al. (2002) авторите разглеждат случая когато група хора използват бюрократичната машина като оръжие с цел да подчинят останалите (тоталитарната държава). В Adorno et al. (2002) 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.

3.1.2 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 Asimov (1950), 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 (Cameron, 1984) 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.

3.1.3 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 The Wachowski Brothers, 1999). 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.

3.1.4 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.

3.1.5 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.

3.1.6 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, които 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.

3.1.7 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.

3.1.8 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.

3.1.9 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.

3.1.10 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 откраднe 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

Тази дисертация води до някои изводи. Основният извод е, че Изкуственият Интелект е близо и че скоро той ще дойде и коренно ще промени живота ни.

Основната промяна ще е това, че трудът ще стане безплатен. Ние сме свикнали с това, че благодарение на техническия прогрес трудът непрекъснато поевтинява. Например, изкопаването на една дупка е било скъпо, когато хората са копали на ръка, но сега когато има багери, тази цена е паднала драстично. Ние сме свикнали с това, че стоките в магазина стават все по-евтини, защото компонентата „труд“, която е включена в тях, поевтинява непрекъснато. Става дума не за поевтиняване спрямо парите, а спрямо нещо твърдо, като златото например (макар че и златото поевтинява, поради това че машините копаят много повече злато, отколкото са можели да изкопаят хората на ръка). Ние сме свикнали с поевтиняването на труда, но не сме готови за момента, в който трудът ще стане безплатен.

Следствие на тази дисертация и на нейния основен извод е патентът Dobrev (2021). Този патент се отнася за схема да движение на автоматично метро (автоматично в смисъл, че влаковете се движат сами без машинисти). Автоматичното метро не е ИИ, както и автоматът за кафе не е. Разбира се, автоматичното метро е нещо по-сложно от автомата за кафе, но все пак не е ИИ. Ние вече знаем какво е ИИ, защото в тази дисертация има дефиниция на ИИ. Тоест, ние вече знаем или поне авторът си мисли, че знае какво е ИИ.

Патентът Dobrev (2021) не е пряко следствие от тази дисертация, но причината за създаването му е основният извод, който се прави тук. Щом трудът става безплатен, тогава е безсмислено да се инвестира в труд. Повечето изобретения целят да спестят труда на хората. Всички тези изобретения стават безсмислени. Патентът Dobrev (2021) цели спестяването на време и на енергия, а това са неща, които ще са ценни дори и след появата на ИИ. Разбира се, когато се появи термоядреният реактор, дори и енергията ще стане безплатна, но времето на хората ще си остане ценен ресурс и ние ще продължим опитите си да го пестим.

През 1988 на конференцията по логика посветена на Хейтинг във Варна присъстваше един от асистентите на Тюринг. Той се казваше Ганди и беше доайенът на конференцията. Ганди разправяше как преди войната Тюринг е решил да превърне всичките си спестявания в сребро и да ги зарови. По този начин той искал да избегне очакваното обезценяване на парите. Не е искал да инвестира и в сгради, защото не се знае кои ще са сградите, които ще оцелеят до края на войната.

Тоест, решението на Тюринг да купи сребро не е пряко следствие от войната, но е косвено последствие предизвикано от очакванията му свързани с войната. Аналогично, патентът Dobrev (2021) не е пряко следствие от тази дисертация, но неговото създаване и инвестицията в неговото патентоване е мотивирана от основния извод направен в тази дисертация.

По-нататък Ганди ни разказа как Тюринг забравил къде точно е закопал среброто. След войната той многократно се опитвал да го намери, но уви безуспешно. Дори и самият Ганди е участвал в такива експедиции, където заедно с Тюринг търсили сребро.

Изводът е, че ние може да се опитаме да се подготвим за предстоящи събития като войната и идването на ИИ, но едва ли ще успеем да го направим. Тези събития са твърде мащабни

и е трудно в такива случай човек да съобрази, коя ще е най-добрата стратегия, която да избере.

Публикации

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

Глава 1.

Dobrev, D. (2000). AI - What is this. *PC Magazine - Bulgaria, 11/2000, pp.12-13 (on https://dobrev.com/AI/definition.html in English).*

Dobrev, D. (2019d). The IQ of Artificial Intelligence. *Serdica Journal of Computing, Vol. 13, Number 1-2, 2019, pp.41-70.*

Глава 2.

Dobrev, D. (2020c). Language for Description of Worlds. *April, 2020, arXiv:2010.16243 [cs.AI].*

Глава 3.

Dobrev, D. (2019c). AI Should Not Be an Open Source Project. *International Journal "Information Content and Processing", Volume 6, Number 1, 2019, pp. 34-48.*

Авторът разполага още с 17 публикации и един патент, които са свързани с дисертацията, но не са част от текста на дисертацията:

Dobrev, D. (1993). First and oldest application. *1993. http://dobrev.com/AI/first.html.*

Dobrev, D. (2001). AI - How does it cope in an arbitrary world. *In: PC Magazine - Bulgaria, February'2001, pp.12-13 (on http://dobrev.com/AI/world.html in English).*

Dobrev, D. (2005a). A Definition of Artificial Intelligence. *In: Mathematica Balkanica, New Series, Vol. 19, 2005, Fasc. 1-2, pp.67-74.*

Dobrev, D. (2005b). Formal Definition of Artificial Intelligence. *International Journal "Information Theories & Applications", vol.12, Number 3, 2005, pp.277-285.*

Dobrev, D. (2005c). Testing AI in one Artificial World. *Proceedings of XI International Conference "Knowledge-Dialogue-Solution", June 2005, Varna, Bulgaria, Vol.2, pp.461-464.*

Dobrev, D. (2005d). AI in Arbitrary World. *Proceedings of the 5th Panhellenic Logic Symposium, July 2005, University of Athens, Athens, Greece, pp.62-67.*

Dobrev, D. (2007a). Parallel between definition of chess playing program and definition of AI. *International Journal "Information Technologies & Knowledge ", vol.1, Number 2, 2007, pp.196-199.*

Dobrev, D. (2007b). Two fundamental problems connected with AI. *Proceedings of Knowledge - Dialogue - Solution 2007, June 18 - 25, Varna, Bulgaria, Volume 2, p.667.*

Dobrev, D. (2008a). 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.*

Dobrev, D. (2008b). The Definition of AI in Terms of Multi Agent Systems. *December, 2008, arXiv:1210.0887 [cs.AI].*

Dobrev, D. (2013). Giving the AI definition a form suitable for the engineer. *arXiv:1312.5713 [cs.AI].*

Dobrev, D. (2014). Comparison between the two definitions of AI. *International Conference "Mathematics Days in Sofia", July 2014, Sofia, Bulgaria, pp. 28-29.*

Dobrev, D. (2017a). Incorrect Moves and Testable States. *International Journal "Information Theories and Applications", Vol. 24, Number 1, 2017, pp.85-90.*

Dobrev, D. (2017b). How does the AI understand what's going on. *International Journal*

"Information Theories and Applications", Vol. 24, Number 4, 2017, pp.345-369.

Dobrev, D. (2018). Event-Driven Models. *International Journal "Information Models and Analyses",* Volume 8, Number 1, 2019, pp. 23-58.

Dobrev, D. (2019a). Minimal and Maximal Models in Reinforcement Learning. *International Journal "Information Theories and Applications",* Vol. 26, Number 3, 2019, pp. 268-284.

Dobrev, D. (2019b). Before we can find a model, we must forget about perfection. *arXiv:1912.04964 [cs.AI]*.

Dobrev, D. (2021). Метод за управление на метрото, при който влаковете се движат без да спират на всички спирки. *BG патент № 67273 B1/ 15.03.2021 г., заявка № 112419 от 01.12.2016 (https://dobrev.com/patent.pdf).*

Декларация

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

Литература

Adorno, T. & Horkheimer, M. (2002) Dialectic of Enlightenment. Stanford University Press, 2002, ISBN: 9780804736336.

Alfonseca, M., Cebrian, M., Anta, A., Coviello, L., Abeliuk, A. & Rahwan I. (2021) Superintelligence Cannot be Contained: Lessons from Computability Theory. *Journal of Artificial Intelligence Research,* Vol. 70, 2021, pp. 65-76.

Asimov, I. (1950). I, Robot (The Isaac Asimov Collection ed.). New York City: Doubleday. ISBN 0-385-42304-7.

Cameron, J. (1984). The Terminator. *American science-fiction action film.*

Church, A. (1941) The Calculi of Lambda-Conversion. *Princeton: Princeton University Press.*

Becquerel, H. (1896). "Sur les radiations émises par phosphorescence". Comptes Rendus. 122: 420–421.

Boutilier, C., Reiter, R. & Price, B. (2001). Symbolic Dynamic Programming for First-Order MDPs. *In Proceedings of the International Joint Conference of Artificial Intelligence,* pp. 690-700.

Cao, X. (2005). Basic Ideas for Event-Based Optimization of Markov Systems. *Discrete Event Dynamic Systems: Theory and Applications,* 15, 169–197, 2005.

Cao, X. & Zhang, J. (2008). Event-Based Optimization of Markov Systems. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL,* VOL. 53, NO. 4, MAY 2008.

Detterman, D. (2017) A Challenge to Watson. *Intelligence, v39 n2-3 p77-78 Mar-Apr 2011.*

Dobrev, D. (1993). First and oldest application. *1993. http://dobrev.com/AI/first.html.*

Dobrev, D. (2000). AI - What is this. *PC Magazine - Bulgaria, 11/2000, pp.12-13 (on https://dobrev.com/AI/definition.html in English).*

Dobrev, D. (2001). AI - How does it cope in an arbitrary world. *In: PC Magazine - Bulgaria, February'2001, pp.12-13 (on http://dobrev.com/AI/world.html in English).*

Dobrev, D. (2005a). A Definition of Artificial Intelligence. *In: Mathematica Balkanica, New Series, Vol. 19, 2005, Fasc. 1-2, pp.67-74.*

Dobrev, D. (2005b). Formal Definition of Artificial Intelligence. *International Journal "Information Theories & Applications", vol.12, Number 3, 2005, pp.277-285.*

Dobrev, D. (2005c). Testing AI in one Artificial World. *Proceedings of XI International Conference "Knowledge-Dialogue-Solution", June 2005, Varna, Bulgaria, Vol.2, pp.461-464.*

Dobrev, D. (2005d). AI in Arbitrary World. *Proceedings of the 5th Panhellenic Logic Symposium, July 2005, University of Athens, Athens, Greece, pp.62-67.*

Dobrev, D. (2007a). Parallel between definition of chess playing program and definition of AI. *International Journal "Information Technologies & Knowledge ", vol.1, Number 2, 2007, pp.196-199.*

Dobrev, D. (2007b). Two fundamental problems connected with AI. *Proceedings of Knowledge - Dialogue - Solution 2007, June 18 - 25, Varna, Bulgaria, Volume 2, p.667.*

Dobrev, D. (2008a). 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.*

Dobrev, D. (2008b). The Definition of AI in Terms of Multi Agent Systems. *December, 2008, arXiv:1210.0887 [cs.AI].*

Dobrev, D. (2013). Giving the AI definition a form suitable for the engineer. *arXiv:1312.5713 [cs.AI].*

Dobrev, D. (2014). Comparison between the two definitions of AI. *International Conference "Mathematics Days in Sofia", July 2014, Sofia, Bulgaria, pp. 28-29.*

Dobrev, D. (2017a). Incorrect Moves and Testable States. *International Journal "Information Theories and Applications", Vol. 24, Number 1, 2017, pp.85-90.*

Dobrev, D. (2017b). How does the AI understand what's going on. *International Journal*

"Information Theories and Applications", Vol. 24, Number 4, 2017, pp.345-369.

Dobrev, D. (2018). Event-Driven Models. *International Journal "Information Models and Analyses",* Volume 8, Number 1, 2019, pp. 23-58.

Dobrev, D. (2019a). Minimal and Maximal Models in Reinforcement Learning. *International Journal "Information Theories and Applications",* Vol. 26, Number 3, 2019, pp. 268-284.

Dobrev, D. (2019b). Before we can find a model, we must forget about perfection. *arXiv:1912.04964 [cs.AI]*.

Dobrev, D. (2019c). AI Should Not Be an Open Source Project. *International Journal "Information Content and Processing", Volume 6, Number 1, 2019, pp. 34-48.*

Dobrev, D. (2019d). The IQ of Artificial Intelligence. *Serdica Journal of Computing, Vol. 13, Number 1-2, 2019, pp.41-70.*

Dobrev, D. (2020a). AI Unravels Chess. *http://dobrev.com/software/AI_unravels_chess.zip*

Dobrev, D. (2020b). Strawberry Prolog, version 5.1. *http://dobrev.com/*

Dobrev, D. (2020c). Language for Description of Worlds. *April, 2020, arXiv:2010.16243 [cs.AI]*.

Dobrev, D. (2021). Метод за управление на метрото, при който влаковете се движат без да спират на всички спирки. *BG патент № 67273 B1/ 15.03.2021 г., заявка № 112419 от 01.12.2016 (https://dobrev.com/patent.pdf).*

Dowe, D.L., & Hernández-Orallo, J. (2012) IQ tests are not for machines, yet. *Intelligence (2012), doi:10.1016/j.intell.2011.12.001*

Goranko, V., Kuusisto, A. & Rönnholm, R. (2020). Gradual guaranteed coordination in repeated win-lose coordination games. *24th European Conference on Artificial Intelligence - ECAI 2020, Santiago de Compostela, Spain, Frontiers in Artificial Intelligence and Applications,* Volume 325, pp. 115-122.

Guelev, D. (2020). Reasoning about Temporary Coalitions and LTL-definable Ordered Objectives in Infinite Concurrent Multiplayer Games. *Presented at the 8th International Workshop on Strategic Reasoning, arXiv:2011.03724 [cs.LO]*.

Guelev, D. & Moszkowski, B. (2021). A Separation Theorem for Discrete Time Interval Temporal Logic. *http://www.math.bas.bg/algebra/seminarAiL/2021/No33_2021_12_03_D_Guelev/SeparationITL-seminarSlides.pdf*.

Gurov, D., Goranko, V. & Lundberg E. (2021). Knowledge-Based Strategies for Multi-Agent Teams Playing Against Nature. *arXiv:2012.14851 [cs.MA]*.

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.

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).*

Lamperti, G., Zanella, M. & Zhao, X. (2020). Diagnosis of Deep Discrete-Event Systems. *Journal of Artificial Intelligence Research,* Vol. 69, 2020, pp. 1473-1532.

Liu, F., Shi, Y. & Liu. Y. (2017) Intelligence Quotient and Intelligence Grade of Artificial Intelligence. *arXiv: 1709.10242, September, 2017.*

Macron, E. (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*

McCarthy, J. (2007) What is Artificial Intelligence? November, 2007. (www-formal.stanford.edu/jmc/whatisai/)

Mell, J., Lucas, G., Mozgai, S. & Gratch J. (2020). The Effects of Experience on Deception in Human-Agent Negotiation. *Journal of Artificial Intelligence Research,* Vol. 68, 2020, pp. 633-660.

Moschovakis, Y. (2001). What is an algorithm? *Mathematics unlimited – 2001 and beyond, edited by B. Engquist and W. Schmid,* Springer, 2001, pp. 919-936.

Moschovakis, Y. (2018). Abstract recursion and intrinsic complexity. *Cambridge University Press, Lecture Notes in Logic,* Volume 48, ISBN: 9781108415583.

Pavlov, I. (1902). The work of the digestive glands. *London: Charles Griffin & Compa-ny, Limited.*

Schofield, M. & Thielscher, M. (2019). General Game Playing with Imperfect Information. *Journal of Artificial Intelligence Research,* Vol. 66, 2019, pp. 901-935.

Reiter, R. (2001). Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. *MIT Press, Cambridge, MA,* ISBN 0-262-18218-1.

The Wachowski Brothers (1999). The Matrix. Science fiction action film.

Tromp, J. (2021). Chess Position Ranking, https://tromp.github.io/

Turing, A. (1937). "On Computable Numbers, with an Application to the Entscheidungsproblem". *Proceedings of the London Mathematical Society*. Wiley. s2-42 (1): pp. 230–265.

Turing, A. (1950) Computing machinery and intelligence. *Mind, 1950.*

Wang, C., Joshi, S. & Khardon, R. (2008). First Order Decision Diagrams for Relational MDPs. *Journal of Artificial Intelligence Research,* Vol. 31, 2008, pp. 431-472.

Wang, H., Tian, F., Gao, B., Bian, J. & Liu, T. (2015) Solving Verbal Comprehension Questions in IQ Test by Knowledge-Powered Word Embedding. *arXiv: 1505.07909, May, 2015.*