Език за описание на светове

Dimiter Dobrev
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
d@dobrev.com

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.

Keywords: Artificial Intelligence, Language for description of worlds, Event-Driven Model, Definition of Property, Definition of Algorithm.

1. 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.

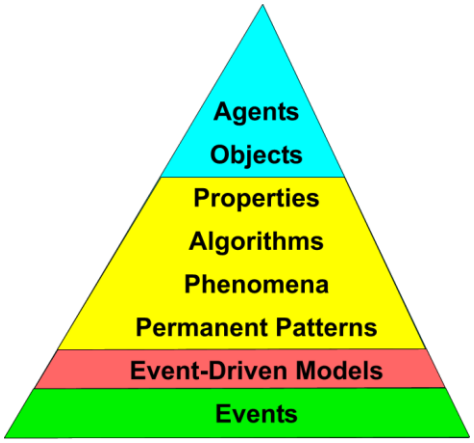

Figure 1

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 забелязаните особености 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. Максимален модел. Това е моделът, при който състоянието знае всичко. Чрез този модел ние ще въведем интерпретация на събитията и на Event-Driven моделите. (Максималният 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 максималния модел не знае за пропуснатите възможности.)
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 ще кажем кой е конкретният свят, който ще опишем. Ще покажем, че досега известните инструменти за описание на светове не са подходящи за този свят. 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.

Ще разширим понятието алгоритъм. Новото понятие за алгоритъм ще е важна част от езика за описание на светове. Накрая ще въведем две абстракции от по-високо ниво. Това са понятията обект и агент. Тези две абстракции също ще са част от езика за описание на светове.

Успоредно с въвеждането на нужните понятията ще опишем езика за описание на светове и ще дадем описание на конкретния свят.

2. 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.

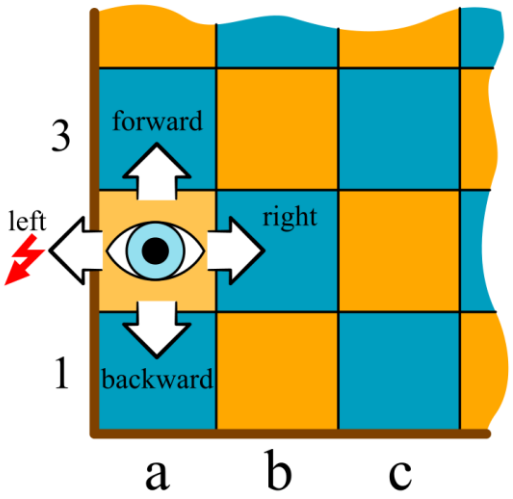

Figure 2

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. 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. 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.

3. 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.

3. 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.

3. 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.

4. 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?“

4. 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 три 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 състоянието“. 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“.

Третото, което ще опише живота са 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 направи, 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.

4. 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:

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

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)*.

4. 3. Атомарна формула

Агентът не може да наблюдава всички събития и ще трябва да се ограничи до краен брои. На всяко наблюдавано събития 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 едно от наблюдаваните събития (няколко стъпки по-рано или по-късно).

4. 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 се отнасят за един и същи момент. Тази конюнкция може и да не е моментно събитие.

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.

4. 5. Предположение

В конкретни моменти от времето ще прилагаме евристиките, за да получим 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.

4. 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 $a_i \in \Sigma$, $o_i \in \Omega$.

$s_0, w_1, s_2, w_3, \ldots w_{k-1}, s_k$, където $s_i \in S$, $w_i \in W$.

Тук Σ са възможните действия, Ω са възможните наблюдения на агента, S 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 живота е L, тогава $k = |L|$).

Животът с предположения ще се състои от три неща: следа, гръбнак и предположения на агента (приложения на евристики).

Дефиниция: Живот с предположения:
следа, гръбнак и предположения:
$g_0, g_1, g_2, \ldots, g_k$, където $g_i \subseteq Guesses$.

Множеството g_i се състои от предположенията, които можем да направим 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}.)

Към предположенията ще добавим и информация за опитите да играем некоректен ход. Тази информация е част от това което агента „вижда“ и тя също трябва да бъде част от живота.

5. 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.

5. 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 $[a, b]$.

5. 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).

5. 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).

5. 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 и b.

Правилото може да бъде и недетерминистично:
4. Избираме събитието a с вероятност p (или с вероятност в интервала $[p_1, p_2]$).

5. 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 съгласуван с ED модела. Когато никое от наблюдаваните събития не се случва, тогава интерпретацията трябва да връща едно и също състояние. Когато се случи някое от наблюдаваните събития, тогава интерпретацията трябва да започне да връща друго състояние, но трябва в ED модела да има стрелка по това събитие от първото състояние към второто.

Ще разгледаме няколко вида интерпретации:

1. Проста интерпретация. Това е релация на еквивалентност, която разбива множеството на състоянията на света на класове на еквивалентност, така че фактор множеството е изоморфно с ED модела. За всеки момент t от живота имаме състояние s_t на света и интерпретацията ще ни даде състоянието S_i на ED модела, което съответства на класа на еквивалентност на s_t.

2. Еднозначна интерпретация. Това е функция, която за всеки момент от всеки 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).

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):

$$\ldots, w_{-3}, s_{-2}, w_{-1}, s_0, w_1, s_2, w_3, s_4, \ldots$$
$$\ldots, a_{-3}, o_{-2}, a_{-1}, o_0, a_1, o_2, a_3, o_4, \ldots$$

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.

5. 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.

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 й 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.

5. 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).

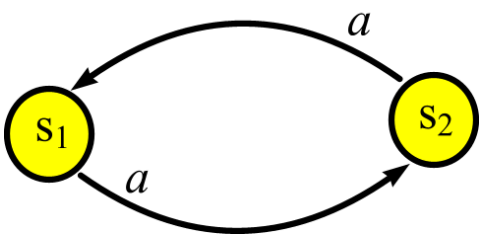

Figure 3

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.

5. 8. Related work

5. 8. 1. Relational MDP

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.

5. 8. 2. Понятието събитие

В основата на нашия език за описание на светове ще стоят Event-Driven (ED) моделите. Тези модели приличат на крайни автомати, но в ED моделите стрелките отговарят на събития. Markov decision process е частен случай на ED модел, но при него събитията са действията на агента. ED моделите са подробно описани в Dobrev (2018).

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:

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

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.

6. Събития

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

6. 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.

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.

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 текущия 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.

6. 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.

6. 3. Невидими събития

Тук ще говорим за моментните невидими събития (timepoint unvisible events). 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 *belief*s? 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.

6. 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.

6. 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.

6. 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*.

6. 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.

6. 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.

6. 3. 6. 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:

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

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:

$$E_i(t) \Rightarrow A(t) \ \textit{(процент)}$$

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.

6. 4. 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 това действие 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 този момент това действие 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 сме опитали неуспешно да изпълним действието a в момента t. По този начин добавяме информацията за некоректните ходове. Тази информация е част от това, което агентът вижда и бихме могли да я добавим към следата на живота, но за да не усложняваме следата ще я добавим към предположенията.

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 опитвал да го играе. Това ще стане благодарение на допълнителните евристики, който ще му даде статистиката.

6. 5. 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.

7. 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.

7. 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.

7. 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.

7. 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).

7. 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.

7. 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 падне ези 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.

7. 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 стратегия на агента и стратегия на света. Тоест, двамата са се наговорили и играят две стратегии, които взети заедно са стратегията 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.

7. 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 е интервалът [0, 1]. Върху стрелките с действия има вероятности, но тези вероятности не определят вероятността на действието, а индиректно определят вероятността на наблюдението. В Simple MDP има стрелки по действията. Вероятностите върху тези стрелки се отнасят за действията. Има и стрелки по наблюденията и вероятностите върху тези стрелки се отнасят за наблюденията.

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.

7. 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. 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 $(S \cup W, A \cup O)$ с два вида върхове и два вида стрелки.

- S 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 $s \in S$, тогава стрелките излизащи от s са от A и стрелките влизащи в s 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.

7. 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 d се заменя с две състояния w и s, които ще са от типове W и S съответно. Всички стрелки, които досега са влизали в d сега ще влизат в w, а тези които са излизали от d, сега ще излизат от s. Колкото са били възможните наблюдения ob в d, толкова допълнителни стрелки ще добавим от w към s. 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 s 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.

7. 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*:

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

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.

8. 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.

8. 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 S 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.

8. 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.

8. 3. 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.

8. 4. 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 *belief*s.

Тоест, обозримият свят е компактен свят и е с една компонента на свързаност.

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.

8. 5. 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 вероятността *m* да сме в състоянието *s* 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)*.

8. 6. 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 ще бъде „излишна“, защото началният *belief* е така дефиниран, че никога няма да можем да разберем точно в кое състояние сме. Тоест, състоянието ни казва точно какво се е случило и какво ще се случи, но няма как да разберем кое е началното състояние. Тоест, състоянието „знае“ нещо, което не следва от миналото.

Множеството на 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 графа, 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.

Новият начален *belief* ще го получим от множеството S_0' като на всяко състояние добавим вероятността *expected*(s_0) (s_0 е първото състояние от живота L).

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

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.

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 на път с дължина k е $(1-\lambda).\lambda^k$. Тук трябва да изберем един коефициент λ, но изборът на този коефициент не е съществен, защото предположихме, че събитията запазват верността си при разширяване на 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, 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.

9. Описание на играта шах

9. 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.

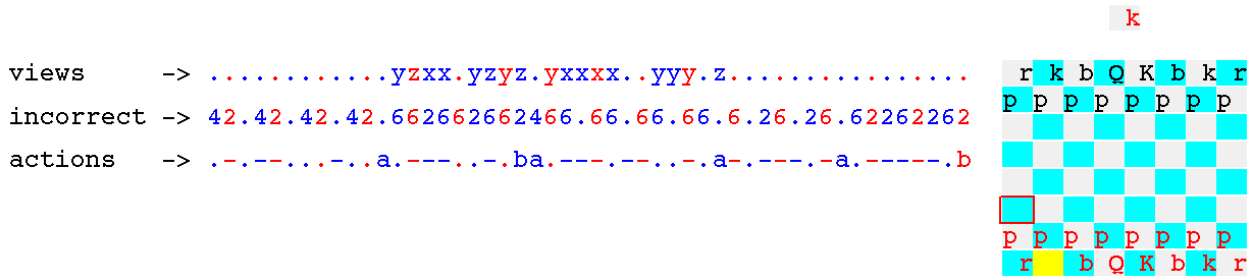

Figure 4

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 (оградено с червена линия).

9. 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.

9. 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“.

9. 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.

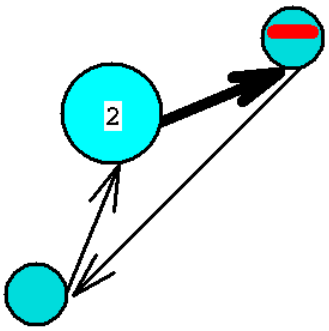

Figure 5

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“).

9. 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 $\boldsymbol{a}$ или $\boldsymbol{b}$ 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.

Постоянната следа ще предполагаме, че е вградени в дефиницията на функцията f'', а подвижната следа ще предполагаме, че се описва от състоянието на света. Все пак, когато подобряваме функцията f'' и променяме представата си за света, ще предполагаме, че част от постоянната следа може да стане подвижна. Например, „Колко врати има стаята?“ е нещо постоянно, докато някой ентусиаст не пробие още една врата.

9. 6. Horizontal и Vertical

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

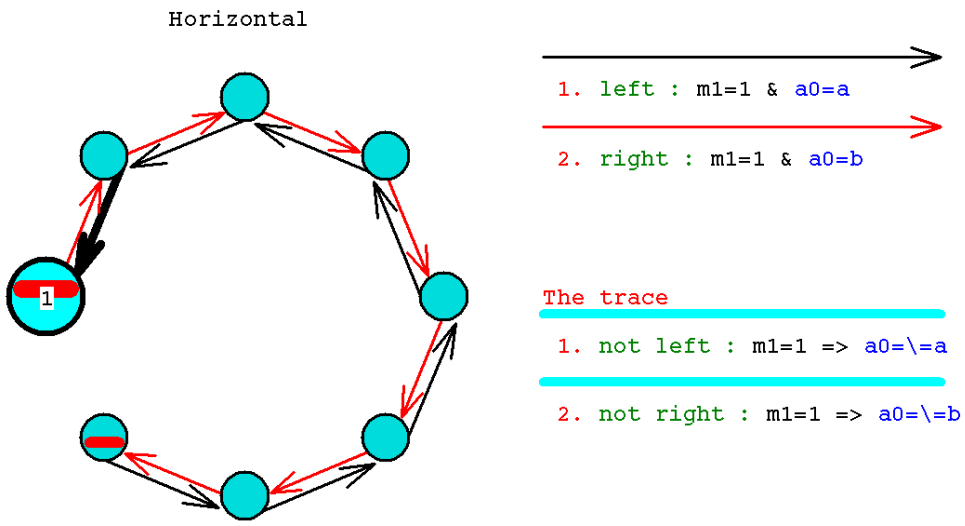

Figure 6

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).

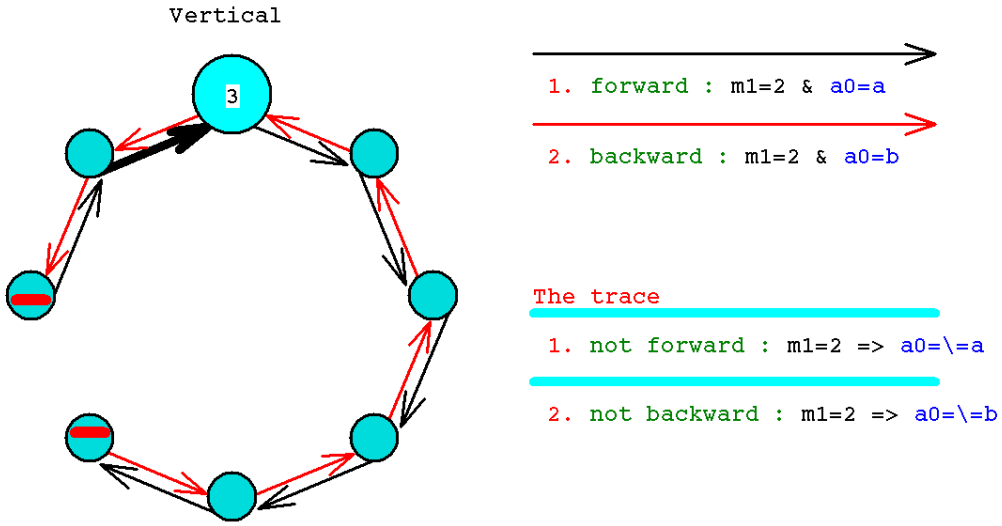

Figure 7

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.

9. 9. 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:

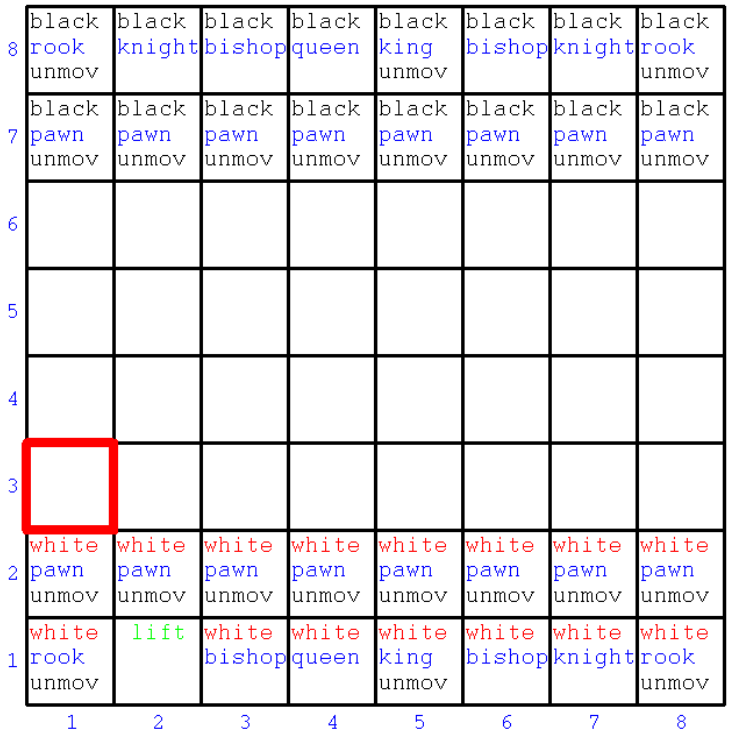

Figure 8

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“.)

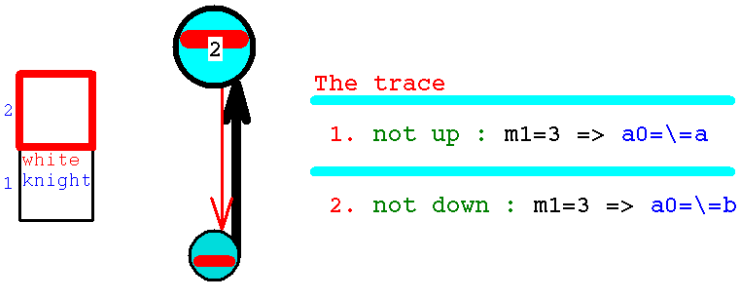

Figure 9

Казахме, че езикът за описание на светове ще ни каже как изглежда паметта на функцията f''. Къде се записва вътрешното състояние на света? Записва се на две места. Първо, това е текущото състояние на всеки от 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.

10. 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.

10. 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.

10. 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:

10. 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).

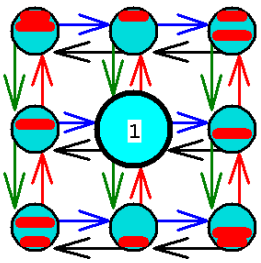

Figure 10

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.

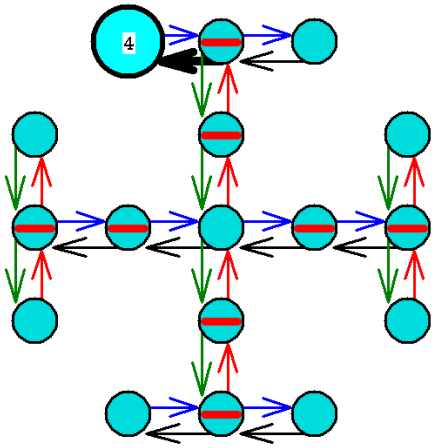

Figure 11

10. 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.

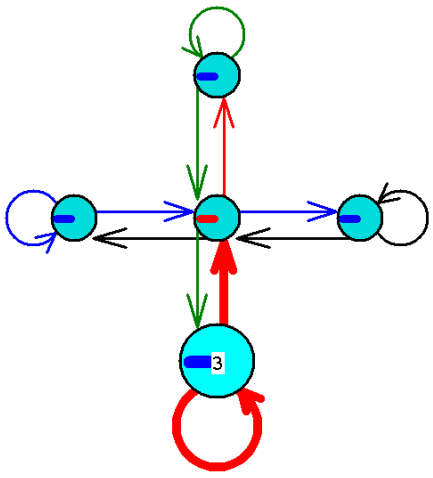

Figure 12

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“.)

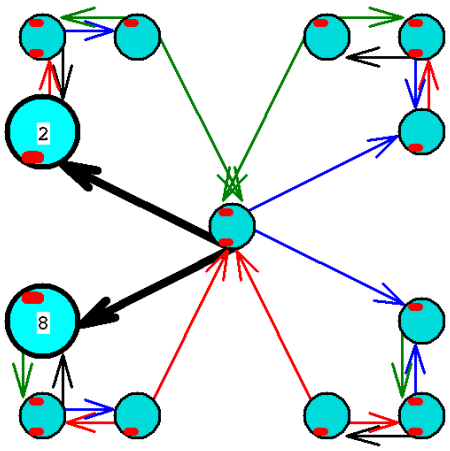

Figure 13

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.

10. 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:

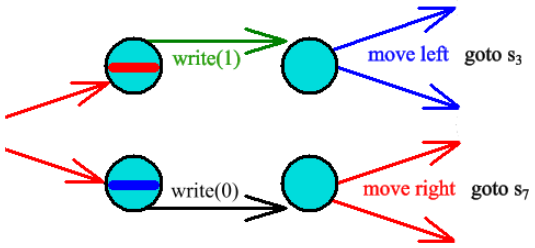

Figure 14

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.

10. 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.“

11. Обекти

11. 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).

11. 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.

11. 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.

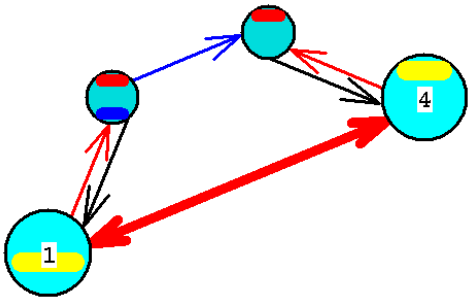

Figure 15

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.

12. 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. Затова ще опишем света с една недетерминистична функция f''.

12. 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, че ние сме описали функцията f''. Тоест, описали сме света.

Описанието, което получихме, е детерминирано. Тоест, началното състояние е определено и функцията f'' е детерминирана. 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 функцията f може да е недетерминирана. Повечето автори, когато говорят за недетерминираност, предполагат, че всяка възможна стойност на функцията 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.

12. 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 неможе да бъде изпълнен, защото съдържа невъзможно събитие.)

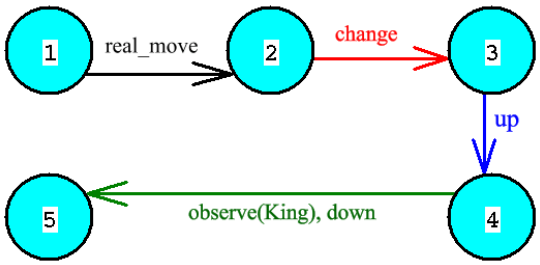

Figure 16

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).

12. 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.

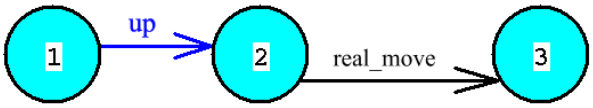

Figure 17

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.

12. 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, чрез една и съща функция f'', но паметта на функцията (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 функцията f'', която е дефинирана в S'', сме направили новата функция f''', която е дефинирана в 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. Общото състояние на света също се описва. Функцията f''' описва света чрез двата агента и това как те променят състоянието си чрез функцията f''. Въпреки всичко, по-лесно е да си мислим, че светът има различни състояния за двата агента и че тези агенти променят състоянията си чрез функцията f'', която работи само с въпросите, които са само за единия агент.

12. 5. Неизчислим свят

Описахме първия свят, в който агентът играеше сам срещу себе си и направихме програмата (Dobrev, 2020a), която емулира този свят. Програмата (Dobrev, 2020a) представлява функцията f'', която 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 функцията f'' може да се получи неизчислима. Няма как да напишем програма, която да изчислява неизчислима функция.

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 функцията f'' е неизчислима), 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.

13. 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.

13. 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.

В тази статия ние стигнахме до понятието агент и следващата стъпка трябва да бъде да изучим взаимодействието между агентите. Основните въпроси при взаимодействието са „Кои са ни приятелите?“ и „Кой колко е умен, силен, умел, какво вижда и колко знае?“.

13. 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.

13. 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).

13. 4. 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.

14. 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.

15. 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.

References

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.

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.

Craig Boutilier, Reiter, R., Bob Price (2001).

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

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

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

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]*.

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

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

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.

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

John Tromp

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.

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.

Chenggang Wang, Saket Joshi, Roni Khardon