Дефиницията на ИИ в термините на мулти-агентните системи

Dimiter Dobrev
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
„Acad. G. Bonchev“ Str., Bl. 8,
1113 Sofia, Bulgaria
dobrev@2-box.com

Резюме:

Въпросите, които ще разгледаме тук, са „Какво е ИИ“ и „Как да направим ИИ“. Тук ще ви представим дефиницията на ИИ в термините на мулти-агентните системи. Тоест, тук няма да намерите нов отговор на въпроса „Какво е ИИ“, но ще намерите един известен отговор, представен по нов начин.

Това ново представяне на дефиницията на ИИ е интересно за теорията на мулти-агентните системи, защото ни дава едно по-добро разбиране на тази теория. По-важното е, че тази работа ще ни помогне да отговорим на втория въпрос. Ние искаме да направим програма, която е в състояние да построи модел на заобикалящия я свят. Всеки мулти-агентен модел е еквивалентен на едно-агентен модел, но мулти-агентните модели са по-естествени и затова те са по-лесно откриваеми.

Ключови думи: Artificial Intelligence, Multi Agent Systems, AI Definition.

Въведение

В предишни статии на същия автор [1-13] се разглежда въпросът за дефинирането на ИИ. Там постановката на задачата е същата, като в статиите, посветени на мулти-агентните системи [14-16]. Може да се каже, че формализацията, използвана в [1-13], е частен случай на формализацията, използвана в [14-16]. Частният случай се получава, като се ограничим само до един агент.

Наистина формализацията в [1-13] се получава като частен случай от формализацията на [14-16], но въпреки това статиите [1-13] продължават да бъдат интересни, защото там въпросът, който се разглежда, е друг. Тоест, ако това са две задачи от типа „Дадено е ..., търси се ...“, то даденото в двете задачи е едно и също, но се търсят различни неща. Тоест, това са две различни задачи, като всяка от тях е интересна за себе си. Все пак съществува съществена връзка между тези задачи и тази връзка може да се използва и в двете посоки. Тоест, може от втората задача да се заимстват идеи и да се обогати първата, а може и обратното.

Постановка на задачата при мулти-агентните системи

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

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

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

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

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

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

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

Постановка на задачата при дефиницията на ИИ

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

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

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

В първата задача се пита дали съществува съответната стратегия, а във втората се търси една определена стратегия. По-точно търси се ИИ, който е една изчислима стратегия (в [1] казахме, че ИИ е програма, тоест той не може да бъде неизчислима стратегия). В [6, 7] се доказва, че тази стратегия (т.е. ИИ) съществува, дори се дава

алгоритъм, който я изчислява. За съжаление този алгоритъм е напълно безполезен, тъй като не работи за разумно време. Макар да има алгоритъм, завършващ работа за краен брой стъпки, този алгоритъм е безполезен, когато този краен брой е практически безкраен.

Как да се обогати задачата на мулти-агентните системи

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

Как да се обогати задачата на ИИ

Решението на задачата на ИИ минава през разбирането на света. Тоест, ИИ трябва да намери модел на непознатия свят, в който е поставен. (Алгоритъмът от [6, 7] не работи по този начин, но както вече казахме, този алгоритъм не върши работа.) Щом ще търсим модел на един непознат свят, трябва първо да изберем едно множество от формални модели, сред които да търсим модела, който ще пасне най-добре. Досега се ограничавахме в множеството на едно-агентните светове, но е по-разумно да се търси формален модел в по-голямото множество на мулти-агентните светове. За всеки мулти-агентен свят съществува един едно-агентен, който му е еквивалентен. Тоест, променяйки множеството на формалните модели, ние не променяме дефиницията на ИИ, която е дадена в [1]. Въпреки това, мулти-агентните светове са по-естествени и в тяхното множество ще ни е по-лесно да намерим модел на съответния свят. Както казахме, за всеки мулти-агентен свят съществува едно-агентен, който му е еквивалентен, но обикновено този едно-агентен свят е доста по-сложен и по-трудно разбираем от съответния си мулти-агентен.

Защо мулти-агентните светове са по-прости? Защото при тях можем да отделим част от сложността на света в отделен обект, който ще наречем агент. По подобен начин разсъждават и хората. Когато ние изучаваме света около нас, ние се сблъскваме с други хора, които бихме могли да приемем за част от пейзажа. Тоест, бихме могли да приемем, че живеем в едно-агентен свят и че сме еднички във вселената. Вместо това ние приемаме, че освен нас има и други хора, което прави модела на заобикалящия ни свят по-прост и по-разбираем. Често ние приемаме съществуването и на по-абстрактни обекти като Господ, съдба, късмет. Ако заключите един човек в една кула и го поставите по този начин в един едно-агентен свят, то той ще започне да си въвежда допълнителни агенти, от които по

принцип не се нуждае, за да обясни света около себе си. Например такъв човек може да приеме вятъра за мислещо същество, което иска нещо да му каже. Може да реши, че някоя от вещите му го гледа лошо и да я счупи. Тоест, човекът е създаден с нагласата да обяснява света около себе си като мулти-агентна система и дори е склонен на моменти да преиграва в тази насока.

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

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

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

Забележка 1

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

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

Това доказателство поставя три въпроса.

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

Вторият въпрос е дали функциите, описващи света, и стратегиите на агентите са детерминирани или недетерминирани. В [2] разглеждахме този въпрос и видяхме, че той не е от съществено значение. Нека предполагаме, че е на лице по-общият случай и че тези функции не са детерминирани.

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

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

Един конкретен пример

В [4, 5, 11, 12] се обсъжда един конкретен пример. Става дума за един изкуствен свят (изкуствен в смисъл направен от човешка ръка). В този свят ИИ трябва да се състезава с един изкуствен противник и да играе с него играта Морски шах. В тези статии се прави опит да се намери едно-агентен модел на този свят, което се оказва доста трудна задача, защото противникът е част от света, което прави модела доста сложен. Разбира се, погрижили сме се изкуственият противник да е добре определен, за да бъде и изкуственият свят добре определен. Ако имаше неопределеност в противника, то тя би се пренесла и на света.

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

противник е залегнала в основата на алгоритъма Min-Max (това е алгоритъмът, с който „мислят“ шахматните програми).

Забележка 2

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

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

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

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

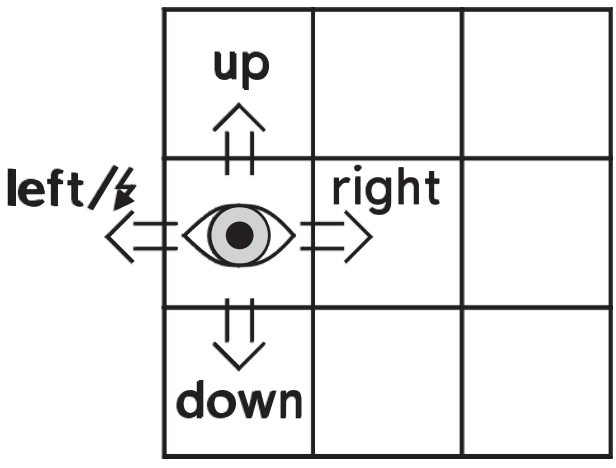
Да отговорим на въпроса съществува ли Бога на слънчевия загар. Отговорът е, че няма значение. Това, което е важно за нас, е дали ако дадем дарове на този бог, ще си осигурим повече слънчеви дни. Тоест, нас не ни интересува дали един модел е верен, а дали върши работа. Въобще дали може да говорим за верни и неверни модели. Ако разделим моделите на класове спрямо релацията еквивалентност, то тогава всеки модел има безбройно много, които са му еквивалентни. Разбира се, ние може да считаме, че еквивалентните модели са неразличими и да търсим кой

да е представител на този клас на еквивалентност. Това също не решава проблема, защото ние търсим модела на базата на „видимата история“, а има безброй много модели, които имат същата видима история, но не са еквивалентни помежду си. Тоест, различават се в бъдещето или другаде (в алтернативните варианти на миналото).

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

Формализация на света от примера

В [4] се разглежда една формализация на света, в който ИИ играе Морски шах. Там имаме едно табло (три на три) и ИИ разполага с едно око, което може да движи по таблото. Характерно за тази формализация, че ИИ вижда само едно от квадратчетата. Останалата част на таблото остава скрита и ИИ трябва да съумее да си я представи. Тоест, този свят е от интересните, защото в него ИИ не вижда всичко (т.е. функцията View не е инекция). ИИ има шест възможни хода: „надолу“, „нагоре“, „наляво“, „надясно“, „постави кръстче в текущото квадратче“ и „нова игра“. ИИ играе нов ход, като постави кръстче. Когато играта е свършила, ИИ трябва да изчисти таблото, за да започне новата игра. За това е нужна командата „нова игра“.



Фигура 1 дава идея за възможните ходове на ИИ.

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

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

Мулти-агентна формализация на горния свят

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

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

Защо предпочитаме симетричните модели?

Една от сериозните трудности при изкуствения интелект е това, че трябва да му осигурим обучение (разбира се, това е така и при естествения интелект). Имаме вариант да осигурим човек учител (треньор, спаринг партньор). Това би било скъпо и времеемко и затова предпочитаме за партньор да използваме друга

програма. За да избегнем необходимостта да пишем такава програма отделно, бихме предпочели да пуснем ИИ да играе сам срещу себе си и по този начин да се самообучава.

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

Ако имаме два агента в един симетричен свят, то спокойно може да предполагаме, че тях ги управлява един ИИ. Този общ интелект ще открива законите на света едновременно за двата си агента, но за всеки агент ще поддържа отделно текущо състояние. Например в света на Морския шах има един много важен краен автомат, който дава информация за това в коя от трите колони се намира окото на ИИ (виж в [4]). За да бъде открит този автомат, е нужно доста процесорно време, защото трябва да го търсим в множеството на всички крайни автомати. Затова е разумно този автомат да бъде открит само веднъж и да се използва и при двата агента. Разбира се, очите на единия и на другия агент ще бъдат в различни колони, т.е. този краен автомат ще е в различни състояния при различните агенти.

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

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

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

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

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

Как ще изглежда конкретният пример

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

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

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

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

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

Ако погледнем на този дву-агентен свят от гледната точка на първия агент, то ще получим един едно-агентен свят, който е подобен на света, описан в [4, 5, 11, 12]. Къде е разликата? Там, когато ИИ играеше кръстче, на таблото веднага се появяваше кръгче, докато сега това става след известен брой стъпки. По-рано вграденият противник виждаше всичко и затова можеше да играе веднага. Сега другият агент вижда същото, като ИИ, и има нужда от време (т.е. стъпки), за да може да обиколи таблото и да го разгледа.

Как да стигнем до процедурата Min-Max

В [11] ние успяхме да построим модел на този свят (там светът беше приблизително същият). Там намерихме краен автомат, който описва текущите координати на окото на ИИ. Освен това намерихме и предикат, който описва позицията на таблото (предикатът беше триместен, като аргументи му бяха x и y координатите на окото и времето, т.е. номерът на текущата стъпка). На базата на този предикат и формули от първи ред ние описахме условията за постигане на победа. Тоест, в [11] ние намерихме модел на света, който ни описва коректно правилата на играта. Въпреки това, ние не успяхме да довършим нещата и на базата на този модел да направим програма, която играе играта Морски шах.

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

Това, което остана недовършено в [11], е да стигнем до процедурата Min-Max или до друга подобна като Max-Sum (виж [6]). Именно идеята на мулти-агентния модел ще ни помогне да приложим Min-Max процедурата.

Когато искаме да накараме ИИ да планира бъдещите си ходове, е естествено да приложим Min-Max на принципа ИИ срещу света. Тоест, смятаме максимум по всички възможни стъпки, които може да направи ИИ и след това минимум по всички възможни реакции на света. В [9] обсъждахме подобна възможност и установихме, че тя директно води до комбинаторна експлозия. Обърнете внимание, че на всеки ход отговарят десетина стъпки (защото ИИ и неговият противник първо трябва да се придвижат до празното квадратче и чак тогава могат да играят). Това означава, че дървото на Min-Max процедурата става десетина пъти по-високо. Като се има предвид, че сложността на Min-Max е в експоненциална зависимост от височината на това дърво, то веднага разбираме, че по този начин директно стигаме до комбинаторна експлозия.

За да стане Min-Max процедурата работеща, тя трябва да бъде на принципа ИИ срещу другия агент. За целта ИИ трябва да осъзнае, че най-важното в света на Морския шах е позицията на таблото. Разбира се, тази позиция не е директно видима, тоест, става дума за една абстракция. Както видяхме в [11], ИИ може да открие предиката, описващ положението на таблото, тоест ИИ може да разбере тази абстракция.

Следващото нещо, което ИИ трябва да осъзнае, е, че той може да променя позицията на таблото. Не е трудно да се досети, че ако сложи кръстче на празно квадратче, то ще промени позицията (т.е. ще промени стойността на предиката за текущите x и y и следващото t). Не е трудно да осъзнае, че може да се движи по

таблото и да достигне до произволно празно квадратче. Тоест, можем да смятаме, че ИИ ще стигне до идеята, че може да постави кръстче на произволно празно квадратче. Това означава, че ИИ може да разбере кои са възможните ходове, с които разполага.

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

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

Варианти на света от примера

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

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

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

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

При всички варианти, които разгледахме досега, агентите играят един след друг (тоест, изчакват се). Нека разгледаме вариант на примера, в който агентите играят едновременно. Нека в началото на партията първият агент да има две кръстчета и да може да ги сложи на таблото, когато си поиска. Нека на всеки 10 стъпки той да получава ново кръстче. Аналогично и за втория агент. Тук няма да има ред (т.е. единият да е на ход, а другият да чака). Няма да има и цайтнот. Ако двамата

едновременно се опитат да играят в едно и също квадратче, нека приоритет да има първият. Ако и двамата едновременно направят линия, то нека играта да е реми. В светове като този също може да се използва процедура, подобна на Min-Max, макар дървото на възможните развития да е по-сложно, отколкото е в случая, когато агентите се редуват.

Заключение

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

References

[1] Dobrev D. D. AI - What is this. In: PC Magazine - Bulgaria, November'2000, pp.12-13 (in Bulgarian, also in [12] in English).

[2] Dobrev D. D. AI - How does it cope in an arbitrary world. In: PC Magazine - Bulgaria, February'2001, pp.12-13 (in Bulgarian, also in [12] in English).

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

[4] Dobrev D. D. Testing AI in One Artificial World. In: Proceedings of XI-th International Conference KDS 2005, June, 2005 Varna, Bulgaria, pp.461-464.

[5] Dobrev D. D. AI in Arbitrary World. In: Proceedings of 5th Panhellenic Logic Symposium, July 2005, University of Athens, Athens, Greece, pp. 62-67.

[6] Dobrev D. D. Formal Definition of Artificial Intelligence. In: International Journal "Information Theories & Applications", vol.12, Number 3, 2005, pp.277-285.

[7] Dobrev D. D. Formal Definition of AI and an Algorithm which Satisfies this Definition. In: Proceedings of XII-th International Conference KDS 2006, June, 2006 Varna, Bulgaria, pp.230-237.

[8] Dobrev D. D. Parallel between definition of chess playing program and definition of AI. International Journal "Information Theories & Applications", vol.1, Number 2, 2007, pp.196-199.

[9] Dobrev D. D. Two fundamental problems connected with AI, XII International Conference "Knowledge-Dialogue-Solution", June 2007, Varna, Bulgaria.

[10] Dobrev D. D. The “sunshine” Method for Finding Finite Automata, Trends in Mathematics and Informatics, July 2007, Sofia, Bulgaria.

[11] Dobrev D. D. Second Attempt to Build a Model of the Tic-Tac-Toe Game, International Book Series "Information Science and Computing", Book 2 - Advanced Research in Artificial Intelligence, June'2008, pp.146-152.

[12] Dobrev D. D. Generator of simple implications, http://www.dobrev.com/AI/app4.html

[13] Dobrev D. D. AI Project, http://www.dobrev.com/AI/

[14] Goranko V. Доклад изнесен във ФМИ, СУ, София, лятото на 2009 г.

[15] Goranko V. Coalition Games and Alternating Temporal Logics, Proc. of the 8th Conference on Theoretical Aspects of Rationality and Knowledge (TARK VIII, Siena, Italy, 8-10 July, 2001), J. van Benthem (ed.), Morgan Kaufmann, 2001, pp. 259-272.

[16] Goranko V.