Абстрактная архитектура интеллектуального агента и методы ее реализации

Д. Ю. Бугайченко
arhangel@tepkom.ru

И. П. Соловьев
solo@is1483.spb.edu

В работе рассматриваются современные подходы к определению концепции агента, области применения мультиагентных систем, математическая модель агента, а также современные методы проектирования и реализации интеллектуальных агентов. Описываемые в работе понятия и подходы иллюстрируются на примере задачи управления автономным исследовательским зондом.

Введение

Одной из основных особенностей большинства современных информационных систем является то, что они не предназначены для самостоятельного принятия решения в тех или иных ситуациях. Предположительно все возможные варианты поведения таких систем должны быть спроектированы человеком и заложены в них на этапе разработки. Попадание подобной системы в условия, не учтенные ее разработчиками, может приводить к аварийному завершению или более тяжелым последствиям, вплоть до повреждения дорогостоящего оборудования и даже гибели людей. В то же время экспоненциальный рост вычислительных возможностей современных процессоров ведет к непрерывному росту спектра задач, поддающихся полной или частичной автоматизации, и увеличению их сложности. В таких условиях становится все сложнее учитывать

все особенности поведения систем на этапе их разработки, а также увеличивается вероятность ошибок в системах, в том числе и критических.

Частично проблему корректности помогает решить тщательное и обширное тестирование, но оно содержит тот же недостаток — поведение системы обычно проверяется только в тех условиях, которые запланированны человеком и, как правило, соответствуют именно тем условиям, которые учитывались уже на этапе разработки. В случае действительно сложных систем полное тестирование, проверяющее все возможные комбинации условий, невозможно за разумное время.

Одним из подходов, направленных на решение этой проблемы, является агентно-ориентированное программирование и применение мультиагентных систем. Важным отличительным свойством концепции *агента* является наличие внешней среды, с которой агент способен взаимодействовать, но не обладает возможностью ее контролировать, поэтому агент всегда должен быть готов к тому, что предпринятые им действия не приведут к желаемым результатам. Таким образом, агент является системой, способной адекватно реагировать на изменения внешней среды, не предусмотренные явно его поведенческими механизмами. Именно это свойство и делает концепцию агента привлекательным инструментом для решения многих задач, возникающих сегодня в области информационных технологий, таких, как, например, распределенное управление и искусственный интеллект.

В работе рассматриваются современные подходы к определению концепции агента, области применения мультиагентных систем, математическая модель агента, а также современные методы проектирования и реализации интеллектуальных агентов. Описываемые в работе понятия и подходы иллюстрируются на примере задачи управления автономным исследовательским зондом.

1. Постановка задачи

В данной работе мы рассмотрим современные подходы к определению концепции интеллектуального агента (раздел 2), область применения интеллектуальных агентов и мультиагентных систем (раздел 3), а также методы абстрактного описания агента (раздел 4) и его реализации (раздел 5). Для иллюстрации представленных подходов в разделе 6 мы приведем примеры их применения

для решения задачи управления автономным исследовательским зондом.

В основе приведенных методов абстрактного описания интеллектуального агента лежат работы М. Вулдриджа [1, 2] и Н. Дженингса [3]. В данной статье мы предложим несколько расширений классических методов, позволяющих описывать такие особенности поведения интеллектуального агента, как самообучаемость (раздел 4.4), целеполагание и прогнозирование (раздел 4.5), а также планирование (раздел 4.6). В разделе 4.7 мы рассмотрим концепцию «намерений» и широко распространенную ментальную (BDI) архитектуру интеллектуального агента, а также предложим методы их описания в рамках разработанного формализма.

2. Понятие агента

Можно сформулировать следующее определение агента, адаптированное многими современными исследователями [3]:

Определение 1. Агент — вычислительная система, помещенная во внешнюю среду, способная взаимодействовать с ней, совершая автономные рациональные действия для достижения определенных целей.

Однако приведенное выше определение не выделяет явно свойства *интеллектуального* агента, предполагающие гибкость его поведения. Обычно считается, что интеллектуальный агент должен обладать следующими свойствами [2]:

- *Реактивностью* — способностью ощущать внешнюю среду и реагировать на изменения в ней, совершая действия, направленные на достижение целей.
- *Проактивностью* — способностью показывать управляемое целями поведение, проявляя инициативу, совершая действия, направленные на достижение целей.
- *Социальностью* — способностью взаимодействовать с другими сущностями внешней среды (другими агентами, людьми и т. д.) для достижения целей.

Любого из первых двух свойств достичь достаточно легко. Например, свойством проактивности, в широком смысле, обладает любой компилятор, основная «цель» которого — сформировать низкоуровневый объектный код на основе программного кода на языке

программирования высокого уровня. По сути любая программная функция может быть рассмотрена как система, целью которой является преобразование некоторых входных данных в выходные. Однако очевидно, что изменение во входных данных функции *во время* ее работы практически наверняка приведет к краху или, как минимум, несоответствию выходных данных входным. Таким образом, произвольная программная система, вообще говоря, не обладает свойством реактивности.

С другой стороны, только реактивные системы (только реагирующие на изменения во внешней среде) тоже достаточно просты — даже самая примитивная продукционная система демонстрирует реактивность. Совмещение же в системе обоих свойств в нужных пропорциях является непростой задачей. Если агент жестко следует сценарию достижения цели, не реагируя на изменения во внешней среде и не обладая способностью замечать необходимость корректировки плана, вряд ли он сможет достичь поставленной цели. С другой стороны, поведение, ограниченное лишь реакцией на поступающие извне стимулы, без какого-либо планирования, не позволит агенту достичь целей, требующих последовательных и взаимосвязанных действий.

Что же касается третьего свойства, *социальности*, то оно тоже не так просто, как может показаться на первый взгляд. С одной стороны, каждый день миллионы компьютеров взаимодействуют между собой, обмениваясь пакетами бинарных данных. Но такое поведение еще нельзя считать социальным. Помимо коммуникации социальное поведение подразумевает кооперацию, заключающуюся в разделении целей между отдельными сущностями, совместное планирование и координацию действий, направленных на достижение общих целей. Социальное поведение, как минимум, предполагает наличие у агента представлений о целях других сущностей и том, как они планируют этих целей достичь.

3. Область применения

Подробно область применения агентов и мультиагентных систем описана в [5] и [7]. Можно выделить три основных класса систем, при реализации которых удобно использовать агентно-ориентированный подход:

- *Открытые системы* — системы, структура которых может

изменяться в процессе их функционирования. Самая большая и самая открытая на сегодняшний день система — Интернет. Социальность и автономность агента позволяют эффективно применять его в качестве элемента открытой системы.

- *Сложные системы* — наилучшими современными методами борьбы со сложностью являются модульность и абстракция. Благодаря автономности агент представляет собой пример абстрактного модуля.

- *Интерактивные системы* — большинство существующих современных систем, несмотря на графические интерфейс и мощную систему справки, требуют серьезных усилий со стороны потенциального пользователя для их освоения. С помощью агентов можно построить интерактивную систему, которая будет не просто принимать и выполнять команды пользователя, а активно и интеллектуально взаимодействовать с ним, стремясь к достижению общих целей.

В промышленности мультиагентные системы наиболее распространены в следующих областях:

- *Автоматизация управления* сложными системами — область, в которой давно и эффективно применяются интеллектуальные агенты. В качестве примеров можно привести платформу ARCHON [8], систему управления производством YAMS [9] и систему управления воздушным движением OASIS [10].

- *Сбор и обработка информации* — агенты часто используются для реализации систем, собирающих и обрабатывающих информацию из всемирной сети Интернет. Большинство современных поисковых машин реализовано с использованием агентов.

- *Игры* — сегодня в компьютерных играх противниками игрока-человека часто становятся игроки, реализованные как интеллектуальные агенты.

4. Абстрактная архитектура

Абстрактная архитектура агента — это удобный инструмент, позволяющий проектировать поведение агентов с использованием

четких формальных методов, а затем проверять корректность полученных систем с использованием методов автоматической проверки корректности. В основу приведенной архитектуры положена модель, описанная в [2].

Для начала предположим, что внешняя среда агента может быть описана с помощью множества S *состояний среды* (*environment states*). Возможные действия агента описываются с помощью множества A *действий* (*actions*). Абстрактно агент может представляться как функция:

$$action : S^* \to A \tag{1}$$

Таким образом, выбор конкретного действия из множества возможных агент осуществляет, основываясь на текущем состоянии внешней среды, а также истории, описывающей все предыдущие состоянии. При этом, как уже упоминалось выше, действия агента могут влиять на окружающую среду, но не контролировать ее полностью. Недетерминированное поведение внешней среды в этом случае можно описать следующей функцией:

$$env : S \times A \to 2^S \tag{2}$$

В этом случае недетерминизм внешней среды выражается в том, что в зависимости от своего текущего состояния и выбранного агентом действия среда может перейти в одно состояние из определенного множества. Если для любой пары состояние-действие множество возможных состояний среды состоит из одного элемента, то такую среду можно считать детерминированной.

Взаимодействие агента и внешней среды можно представлять с помощью *истории* (*history*), которая является упорядоченной последовательностью пар состояние-действие:

$$h : s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \dots \tag{3}$$

где s_0 представляет начальное состояние внешней среды, a_i представляет i-е действие, выбранное агентом, а s_i — состояние внешней среды после осуществления агентом $(i-1)$-ого действия. В этом случае во внешней среде env для агента $action$ история h будет *допустимой* (*possible history*), если выполнены следующие два условия:

- $\forall n \in \mathbb{N}, a_n = action((s_0, s_1, s_2, ..., s_n))$ — т.е. поведение агента определяется моделирующей его функцией;

- $\forall n \in \mathbb{N}, s_{n+1} \in env(s_n, a_n)$ — т.е. поведение среды определяется моделирующей функцией и поведением агента.

Характеризующим поведением (*characteristic behavior*) агента будем называть множество всех допустимых историй агента во внешней среде. При этом если некое свойство ϕ сохраняется для всех историй агента, то его назовем *инвариантным* (*invariant property*) свойством агента. Такие свойства играют особую роль в проектировании агентов, так как отражают изначально заложенные в агента цели, т. е. именно те цели, для достижения которых и проектируется агент. Так, для агента, управляющего атомным реактором, примером подобного инварианта будет свойство реактора не взрываться.

Множество всех допустимых историй агента *agent* во внешней среде *environment* мы будем обозначать как *hist* (*agent, environment*). Два агента ag_1 и ag_2 будут *поведенчески эквиваленты* (*behaviorally equivalent*), если *hist* (ag_1, *environment*) = *hist* (ag_2, *environment*). В общем случае взаимодействие агента с внешней средой никогда не заканчивается, т. е. последовательность истории является бесконечной.

4.1. Только реагирующий агент

Можно выделить определенный класс агентов, конкретное действие которых определяется не всей историей изменений внешней среды, а только ее текущим состоянием. В этом случае агент будет представляться функцией $action : S \to A$, т. е. для каждого состояния среды у агента есть четко определенное действие. Этот класс агентов называется *только реагирующими*. Очевидно, что только реагирующий агент является частным случаем агента. Только реагирующие агенты обычно реализуются с помощью продукционных систем, их поведение легко предсказуемо, но, как правило, не слишком интеллектуально.

4.2. Восприятие

Часто для описания агента удобно использовать модель *восприятия* окружающей среды. Для этого необходимо ввести множество

P возможных восприятий и функцию $see : S \rightarrow P$, описывающую, каким образом определенные состояния среды воспринимаются агентом. В этом случае агент может быть описан с помощью функции

$$action : P^* \rightarrow A, \tag{4}$$

т. е. действие такого агента определяются в общем случае текущим восприятием состояния внешней среды, а также множеством предыдущих восприятий.

Очевидно, что модель агента с восприятием эквивалентна базовой, и на первый взгляд может показаться, что подобная модификация не дает никаких преимуществ и только усложняет представление. Но на самом деле модель восприятия обладает следующим важным свойством:

$$\neg(see(s_1) = see(s_2) \Rightarrow s_1 = s_2),$$

т. е. разные состояния среды могут одинаково восприниматься агентом. Таким образом, модель восприятия позволяет явно отразить следующие два аспекта деятельности агента:

- *Неполнота информации* — только в редких случаях агент может обладать полной информацией о внешней среде, достаточной для того, чтобы однозначно идентифицировать состояние. Чаще всего агенту доступна лишь частичная информация, позволяющая предположить, что внешняя среда находится в одном из состояний класса с некоторым набором инвариантных свойств. Модель восприятия позволяет явно описать эти классы.

- *Избыточность информации* — с другой стороны, чаще всего для принятия решения полная информация о состоянии внешней среды и не требуется, а интересна лишь информация о тех частях системы, которые могут повлиять на выполнение действия и его последствия. В этом случае мы опять приходим к классам состояний, обладающих инвариантными свойствами.

Несложно заметить, что функция восприятия задает отношение эквивалентности «$\approx$» на множестве S такое, что

$$s \approx s' \Leftrightarrow see(s) = see(s').$$

Это отношение разбивает множество S на классы эквивалентности. Мощность соответствующего множества классов эквивалентности (обозначим ее $|\approx|$) позволяет судить о сенсорных возможностях агента — чем больше эта мощность, тем четче агент способен воспринимать внешнюю среду. При $|\approx| = |S|$ агент обладает совершенными сенсорными способностями и может отличить любые два различных состояния внешней среды. В другом предельном случае, когда $|\approx| = 1$, сенсорные способности у агента отсутствуют — он не способен отличить ни одно состояние окружающей среды от другого.

Поскольку выразительная мощность модели агента с восприятием эквивалентна базовой модели агента, но обладает рядом полезных свойств, далее в рассуждениях мы будем развивать именно ее. Однако, при желании, адаптировать дальнейшие рассуждения к базовой архитектуре агента не составит труда.

4.3. Агент с состоянием

Еще одной интересной модификацией абстрактной архитектуры является *агент с состоянием*. Считается, что такой агент содержит некоторые внутренние структуры данных (что вполне естественно для любой информационной системы), которые он модифицирует в зависимости от восприятия текущего состояния внешней среды, и на основе полученных результатов выбирает действие. Для формализации этого процесса введем множество I внутренних состояний агента и функцию

$$refine : I \times P \to I, \tag{5}$$

отвечающую за обновление внутреннего состояния в соответствии с текущим восприятием среды. Агент же в этом случае будет описываться с помощью функции

$$action : I \to A, \tag{6}$$

т. е. действие будет выбираться на основе текущего внутреннего состояния. При этом выбор действия на самом деле осуществляется с помощью суперпозиции функций $action(refine(i, see(s)))$. Для корректного описания поведения агента с состоянием необходимо определить начальное состояние i_0. Тогда изменение внутреннего

состояния агента и выбор действия будет происходить следующим образом:

$$\forall n \in \mathbb{N}, i_{n+1} = refine(i_n, see(s_n)) \wedge a_n = action(refine(i_n, see(s_n))).$$

Несложно доказать, что модель агента с состоянием эквивалентна базовой модели. С одной стороны, состояние агента определяется исключительно его историей, т. е. базовая модель по выразительной мощности не меньше модели агента с состоянием. С другой стороны, в состоянии агента может храниться вся его история, т. е. модель с состоянием по выразительной мощности не меньше базовой модели.

Ценное преимущество модели агента с состоянием заключается в удобстве ее практической реализации. Состояние позволяет основывать выбор действия не на последовательности входных данных переменной (и потенциально бесконечной) длины, а на одном конкретном элементе состояния. Оказывается, что чаще всего вся история агента и не нужна для принятия конкретного решения, а достаточно лишь какой-то полученной из нее конкретной информации.

4.4. Самообучающийся агент

Описанная выше абстрактная архитектура обладает одним существенным недостатком — определенный таким образом агент не получает информации о совершенных им самим действиях, что резко ограничивает его возможности в накоплении опыта и анализе потенциальных последствий действий.

Можно считать, что информация о действиях агента является частью информации об окружающей среде, но такой подход не является наглядным и интуитивно понятным. Более правильным решением возникшей проблемы представляется включение информации о совершаемых действиях явно во входные данные функции выбора действия:

$$action : (P \times A)^* \to A \tag{7}$$

В таком виде агент явно получает информацию о всей истории взаимодействия с окружающей средой, а не только о последовательности состояний, в которые окружающая среда переходила.

Для агента с состоянием анализ и накопление опыта осуществляются функцией обновления состояния (5), следовательно, информацию о действиях агента логичнее обрабатывать именно с ее помощью:

$$refine : I \times P \times A \to I \tag{8}$$

Заметим, что параметром функции преобразования состояния передается не последовательность всех действий агента, а только информация о последнем предпринятом им действии. При этом предполагается, что информацию о предыдущих действиях агент уже проанализировал и отразил результаты анализа в своем внутреннем состоянии, сделав их доступными для всех последующих итераций.

4.5. Управляемый целями агент

В этом разделе мы рассмотрим такое важное отличительное свойство интеллектуального агента, как управляемое целями поведение. Одним из возможных способов описания цели агента является определение оценочной функции:

$$goal : P \to [0, 1] \tag{9}$$

Эта функция позволяет агенту для каждого восприятия состояния среды определить, насколько оно соответствует поставленной перед ним цели. Часто можно встретить ситуацию, когда цель агента является «неделимой», т. е. агент либо достигает цели полностью, либо не достигает ее вообще. В этом случае значения оценочной функции логичнее рассматривать не на отрезке $[0, 1]$, а на множестве из двух элементов $\{0, 1\}$, т. е. $goal : P \to \{0, 1\}$. Таким образом, цель агента считается достигнутой только в том случае, если внешняя среда перешла в одно из состояний s, таких, что $see(s) \in goal^{-1}(1)$, и не достигнутой иначе. Существует другая распространенная ситуация, когда агенту следует не стремиться к определенным состояниям среды, а, наоборот, избегать их. Такое поведение можно смоделировать с помощью симметричной функции $1 - goal$.

Обычно агент преследует не одну конкретную цель, а некоторый их набор, при этом цели агента являются частью его внутреннего состояния: $I = I' \times 2^{Goals}$, где $Goals = \{g \mid g : P \to [0, 1]\}$ есть

конечное множество всех оценочных функций, а I' есть остальная часть состояния агента, не относящаяся к целеполаганию. Тогда общая оценочная функция агента для состояния (i, G) может быть определена, например, так:

$$goal(s) = \frac{1}{|G|} \sum_{g \in G} g(p), \tag{10}$$

где p есть оцениваемое восприятие состояния среды. В случае, когда агент не имеет ни одной цели ($G = \emptyset$), значение оценочной функции определяется равным нулю ($goal \doteq 0$).

На практике чаще всего цели агента имеют различный приоритет. В этом случае структура множества целей усложняется:

$$Goals = \{(g, w) \mid g : P \to [0, 1], w \in [0, +\infty)\},$$

где w есть неотрицательное число, определяющее приоритет конкретной цели для агента. Общая оценочная функция агента для состояния (i, G) тогда может иметь вид:

$$goal(s) = \frac{1}{\sum\limits_{(g,w) \in G} w} \sum_{(g,w) \in G} (w \cdot g(p)) \tag{11}$$

В случае, когда агент не имеет ни одной цели с ненулевым приоритетом ($G = \emptyset$ или $\sum\limits_{(g,w) \in G} w = 0$), значение оценочной функции определяется равным нулю ($goal \doteq 0$).

Однако для эффективного управляемого целями поведения агенту недостаточно просто определить свои цели, в большинстве случаев необходимым условием для эффективности является способность *предсказывать* последствия своих действий для внешней среды. Как часть своего состояния агент должен включать *прогнозирующую функцию*

$$prog : P \times A \to 2^P \tag{12}$$

принимающую в качестве параметров текущее восприятие состояния внешней среды и действие агента, а возвращающая множество возможных восприятий состояний среды, в одно из которых она перейдет после выполнения действия. Более гибкий вариант прогнозирующей функции включает вероятность перехода внешней среды в то или иное состояние:

$$prog : P \times A \to 2^{P \times [0,1]}, \tag{13}$$

т. е. в паре $(p_2, \theta) \in prog(p_1, a)$ число θ есть вероятность того, что при выполнении агентом действия a внешняя среда из состояния, воспринимаемого как p_1, перейдет в состояние, воспринимаемое как p_2. Очевидно, что простую прогнозирующую функцию можно выразить с помощью вероятностной, используя предельные значения вероятности 0 и 1.

Прогнозирующая функция является важной составляющей внутреннего состояния агента и строится на основе его аккумулированного опыта. Обозначим множество всех прогнозирующих функций через

$$Progs = \{prog \mid prog : P \times A \to 2^{P \times [0,1]}\}$$

В этом случае структура внутреннего состояния агента примет вид $I = I' \times 2^{Goals} \times Progs$.

Использование прогнозирующих функций позволяет представить задачу принятия решения агентом как задачу нахождения точки максимального значения функции *потенциального эффекта действия* $v : A \to [0, +\infty)$, выражаемой через функции $goal$ и $prog$. Для простой прогнозирующей функции это $v(a) = \sum\limits_{p \in prog(p_{cur}, a)} goal(p)$, где p_{cur} есть восприятие текущего состояния внешней среды. Для вероятностной прогнозирующей функции это выражение сложнее:

$$v(a) = \sum_{(p,\theta) \in prog(p_{cur}, a)} (\theta \cdot goal(s))$$

Если же обратиться к вычислению функции $goal$ (см. уравнение 11), то можно получить следующее итоговое выражение функции v для агента с состоянием $(i, G, prog)$ и внешней среды в состоянии, воспринимаемом как p_{cur}:

$$v(a) = \sum_{(p,\theta) \in prog(p_{cur}, a)} \left(\frac{\theta}{\sum\limits_{(g,w) \in G} w} \cdot \sum_{(g,w) \in G} (w \cdot g\,(p)) \right) \tag{14}$$

Заметим, что для повышения эффективности вычислений нормировочный коэффициент $\left(\sum\limits_{(g,w)\in G} w\right)^{-1}$ можно вынести за знак суммы или вообще отбросить, так как он не влияет на расположение точки максимума функции. В этом случае функция v примет вид:

$$v(a) = \sum_{(p,\theta)\in prog(p_{cur},a)} \left(\theta \cdot \sum_{(g,w)\in G} (w \cdot g\,(s))\right) \tag{15}$$

Следует заметить, что в случае ограниченного ресурсами агента точно решить задачу поиска точки максимума функции для достаточно сложных проблем может оказаться невозможным за разумное время. В таких случаях агент может использовать приближенные методы решения, а функция v позволит оценить эффективность поведения агента.

4.6. Планирование

Как указывалось выше, интеллектуальный агент предположительно должен демонстрировать проактивное поведение, что фактически означает строить некоторые планы и рассчитывать свои действия на несколько ходов вперед. Таким образом, помимо *оперативных целей*, достигаемых на текущем действии (именно такие цели рассматривались в разделе 4.5), у агента появляются *перспективные цели*, для достижения которых ему потребуется выполнить последовательность из нескольких действий. На самом деле именно перспективные цели являются определяющими, и на их основе агент формирует цели оперативные с помощью *планирования*, использующего имеющийся у него опыт.

В большинстве случаев процесс планирования, помимо формирования оперативных целей, включает в себя поддержание вспомогательной структуры данных, являющейся частью общего состояния агента. Обозначим множество всех таких структур как $Plans$, тогда состояние агента будет включать множество перспективных целей, прогнозирующую функцию, информацию о плане и остальную вспомогательную информацию: $I = I' \times 2^{Goals} \times Progs \times Plans$. В итоге процесс планирования можно смоделировать с помощью *планирующей функции*

$$plan : 2^{Goals} \times Progs \to Plans,$$

которая на основе данных перспективных целей формирует структуру с описанием плана, используя для этого прогнозирующую функцию, и функцию *формирования оперативных целей*

$$oper : Plans \times P \to 2^{Goals},$$

которая на основе плана и восприятия текущего состояния внешней среды осуществляет формирование множества оперативных целей.

Можно отметить, что в большинстве случаев процесс планирования имеет бóльшую вычислительную сложность, чем процесс принятия решения о конкретном оперативном действии. Для оптимизации этого процесса можно запоминать основу однажды составленного плана и адаптировать ее к возникающим задачам, что может оказаться значительно эффективнее создания нового плана «с нуля».

4.7. Намерения и ментальная архитектура агента

Корни ментальной (*beliefs-desires-intentions*) архитектуры интеллектуальных агентов лежат в философских подходах к анализу мыслительной деятельности человека, того, как люди на практике принимают решения о том, что им следует делать. Можно выделить следующие отдельные этапы в принятии решения *ментальным* агентом: сначала агент должен понять, *чего* он хочет, затем определить, *какие* цели из желаемых он будет пытаться реализовать, а затем понять, *как* он будет реализовывать выбранные цели. При этом в состоянии агента четко разграничиваются следующие компоненты:

- *Представления (beliefs)* — некоторая информация о закономерностях и текущем состоянии внешней среды, которой располагает агент. При этом предполагается, что эта информация может быть ошибочной и неполной, поэтому ее можно рассматривать только как представления, но не как достоверные знания. Заметим, что прогнозирующая функция (см. определение 13 в разделе 4.5) является частью представлений агента. Таким образом, множество представлений имеет структуру $Beliefs = I' \times Progs$.

- *Желания (desires)* — множество всех целей, которых агент хотел бы добиться. При этом множество может быть большим и противоречивым. Маловероятно, что агент, ограниченный ресурсами, сможет реализовать все свои желания. По своей структуре множество желаний совпадает с множеством целей, описанном в разделе 4.5, взвешенному или нет ($Desires = 2^{Goals}$).

- *Намерения (intentions)* — множество тех целей, которых агент решил добиться. Сформированное множество целей должно быть *выполнимо* по представлениям агента, т. е. все намерения агента должны быть достижимы в совокупности. Для представления намерений также можно использовать взвешенное множество целей, при этом веса намерений будут зависеть от весов соответствующих желаний, но, возможно, не совпадать с ними.

Под выполнимостью намерений в случае планирующего агента (см. раздел 4.5) можно понимать наличие у агента плана, ведущего к осуществлению всех намерений. Более того, этот план становится частью структуры намерений агента $Intentions = 2^{Goals} \times Plans$, а также влияет на присваиваемые намерениям веса (например, намерение, на реализацию которого было потрачено много усилий и которое близко к завершению, будет иметь больший приоритет). В случае, если для представления желаний агент пользуется взвешенными целями, можно предложить механизм оценки сформированного множества намерений, основанный на критерии *общей полезности*

$$utility(G) = \sum_{(g,w) \in Des \wedge g \in G} w,$$

где Des есть множество желаний, а G есть оцениваемое множество намерений. Таким образом, при формировании множества намерений агент решает задачу оптимального выбора, пытаясь максимизировать общую полезность, сохранив выполнимость множества намерений.

Новым ключевым понятием процесса принятия решения в ментальной архитектуре является именно *намерение*. В целом можно выделить следующие свойства намерений:

- Намерения задают направление деятельности — агент пытается найти действия, способные осуществить намерения и выполнить их.
- Намерения ограничивают будущий выбор — агент не может формировать новые намерения, несовместимые с уже принятыми, т. е. ведущие к *невыполнимости* множества намерений.
- Намерения имеют долгое время жизни — если агент сформировал план реализации намерения, но он провалился, то агент будет формировать новые планы и пытаться реализовать намерение другим способом. Намерение может быть отброшено только при осуществлении определенного ментального усилия в случаях, если агент пришел к выводу, что реализовать намерение невозможно (не удается сформировать план, ведущий к достижению намерения) или оно уже не актуально для агента.
- Намерения влияют на рассуждения о будущем и, соответственно, планы — если агент выработал намерение, то он может строить планы на будущее с предположением, что это намерение реализовано.

Сам же процесс принятия решений ментального агента состоит из последовательных этапов, на каждом из которых используются результаты предыдущих. Поэтапное преобразование выполняется при помощи следующих функций:

1. *Актуализация представлений* ($brf : Beliefs \times P \times A \rightarrow Beliefs$) — на этом этапе модифицируется представления агента о текущем состоянии среды на основе восприятия, а также производится анализ и обобщение полученного опыта.
2. *Формирование желаний* ($option : Beliefs \times Desires \rightarrow Desires$) — на этом этапе агент формирует множество всех своих желаний. При этом используется множество уже существующих желаний и актуализированные представления агента.
3. *Фильтрация желаний* ($filter : Beliefs \times Desires \times Intentions \rightarrow Intentions$) — на этом этапе определяется, какие именно цели из множества желаемых агент будет пытаться реализовать. Осуществляется выбор из множества всех

желаний, с учетом представлений и уже выбранных намерений, создается план реализации намерений и формируются веса намерений с учетом плана. Важным условием работы фильтра является то, что сформированное множество намерений может содержать только те элементы, которые изначально присутствовали в одном из входных множеств: $filter(Bel, Des, Int) \subseteq Des \cup Int$.

4. *Формирование оперативных целей* ($oper : Beliefs \times Intentions \to 2^{Goals}$) — на этом этапе агент, используя множество намерений в качестве перспективных целей, формирует множество оперативных целей и веса для них. В этом процессе агент также использует представления и сформированный для реализации намерений план.

5. *Выбор действия* ($select : Beliefs \times 2^{Goals} \to A$) — на этом этапе агент выбирает оптимальное действие, на основе своих представлений, а также оперативных целей и их весов.

5. Методы проектирования и реализации

В предыдущем разделе мы выявили три основных аспекта деятельности агента: формирование множества целей (иногда многоступенчатое), прогнозирование поведения окружающей среды и планирование действий. В этом разделе мы рассмотрим известные подходы, позволяющие реализовать эти аспекты деятельности агента на практике.

5.1. Цели агента

Можно предложить следующие подходы к представлению цели агента и формированию множества целей:

- *Логический* — в этом случае цель агента описывается некоторым выражением формальной логики, интерпретация которого зависит от состояния внешней среды. При этом класс состояний среды, в которых цель считается достигнутой, составляют те состояния, на которых это выражение интерпретируется как истинное. В случае использования нечеткой логики можно получить цели, выполнимые частично. Формальная логика

является мощным и привлекательным инструментом с богатой теоретической базой, но обладает и рядом недостатков, связанных в основном с вычислительной сложностью и возможной неполнотой формального аппарата.

- *Перечисление* — в этом случае агент явным образом перечисляет состояния внешней среды и степень их соответствия данной цели. Очевидным плюсом данного метода является простота и низкая вычислительная сложность для небольших задач. Однако в случае сложных задач явное перечисление может потребовать слишком много памяти и приведет к снижению эффективности.

- *И-ИЛИ граф* — мощный инструмент декомпозиции задач на подзадачи. Идея метода заключается в формировании ориентированного графа, на одной "стороне" которого находятся высокоуровневые, сложные цели агента, а на другой — элементарные цели, представленные подходящим образом. Внутреннюю часть графа составляют вершины двух типов: "И" и "ИЛИ". В случае вершины "И" для выполнения соответствующего вершине условия требуется выполнение условий для всех вершин, соединенных с рассматриваемой исходящими ребрами. Для вершин "ИЛИ" достаточно выполнения условия хотя бы в одной из вершин, соединенных исходящей дугой. Можно заметить, что И-ИЛИ граф достаточно легко моделируется средствами других методов с помощью соответствующих логических или теоретико-множественных операций, однако явное задание графа дает целый ряд весьма существенных для последующего планирования преимуществ. Например, граф позволяет явно идентифицировать наиболее критичные участки, задействованные сразу в нескольких целях.

5.2. Прогнозирование

Можно предложить следующие подходы к формированию прогнозирующей функции агента:

- *Логический* — в этом случае агент поддерживает базу знаний, содержащую множество утверждений формальной логики, описывающих причинно-следственные связи внешней среды. Для определения возможных переходов внешней среды

при совершении агентом определенного действия рассматривается множество логических следствий из множества утверждений базы знаний, интерпретируемых на текущем состоянии внешней среды как истинные, в предположении, что агент выполнил действие. Множество состояний среды, на которых эти логические следствия интерпретируются как истинные, и будет определять возможные переходы среды. Внедрение нечеткой логики позволит получить вероятностную прогнозирующую функцию. Плюсом логического подхода является высокая точность и формальность, к минусам же можно отнести высокую вычислительную сложность.

- *Фактографический* — агент поддерживает базу знаний, содержащую информацию обо всех имевших место переходах внешней среды. При этом прогнозируемые переходы определяются на основе такой базы знаний очень просто — возможны лишь те переходы, которые имели место раньше, и вероятность перехода тем выше, чем чаще такой переход встречался ранее. К плюсам метода можно отнести простоту реализации и быстродействие, к минусам же — отсутствие обобщения полученного опыта.

- *Основанный на применении нейронных сетей* — широко распространенный в последнее время метод приближенного моделирования. В отличие от полной фактографии в данном подходе обнаруженный переход не запоминается явно, а используется как элемент обучающей последовательности для моделирующей нейронной сети. Плюсом метода является невысокая вычислительная сложность, не зависящая от объема полученного агентом опыта, к минусам же можно отнести полное отсутствие наглядности у полученной модели.

5.3. Планирование

Реализация эффективного планирующего механизма является, безусловно, наиболее сложной из задач агента. Основной принцип, лежащий в основе любого планирующего механизма, — это рекурсивное разбиение задачи на подзадачи, однако в каждом конкретном методе декомпозиция может отличатся по глубине проведенной

детализации и по ее ширине. Можно предложить следующие способы декомпозиции:

- *Последовательность задач* — в этом случае план является плоской цепью задач, которые следует выполнить одну за другой. При этом каждый элемент цепи является простой целью, достижимой с помощью единственного действия. Особое внимание здесь уделяется организации механизма контроля, позволяющего определить, завершился ли предыдущий шаг плана успешно или нет, поэтому для организации планирования удобнее использовать неделимые цели, позволяющие однозначно определить, достигнуты они или нет. Плюсом такой декомпозиции является относительная простота, минусом же — отсутствие средств обработки незапланированных ситуаций.

- *Дерево задач* — в отличие от простой последовательности в дереве учитываются различные результаты действий агента на каждом шаге, вследствие чего и возникает ветвление дерева. Такой план сложнее построить, однако при его реализации меньше вероятность того, что встретится незапланированная ситуация и придется строить новый план «с нуля».

- *Сеть задач* — в этом случае допускается последующее слияние разошедшихся ветвей плана с базовой линией. План в таком виде получает более экономичное представление и может быть быстрее построен, хотя метод его представления сложнее реализовать.

Методы выполнения декомпозиции следует основывать на прогнозирующей функции, определяя для каждого шага плана возможные его последствия. Можно выделить два принципиальных подхода к построению плана:

- прямой — в этом случае агент на основе текущих условий, используя прогнозирующую функцию, пытается определить действия, способные привести к достижению цели;

- обратный — агент на основе функции, обратной прогнозирующей, и предполагаемой цели пытается найти последовательность действий и переходов, реализующую путь из текущего узла к данной цели.

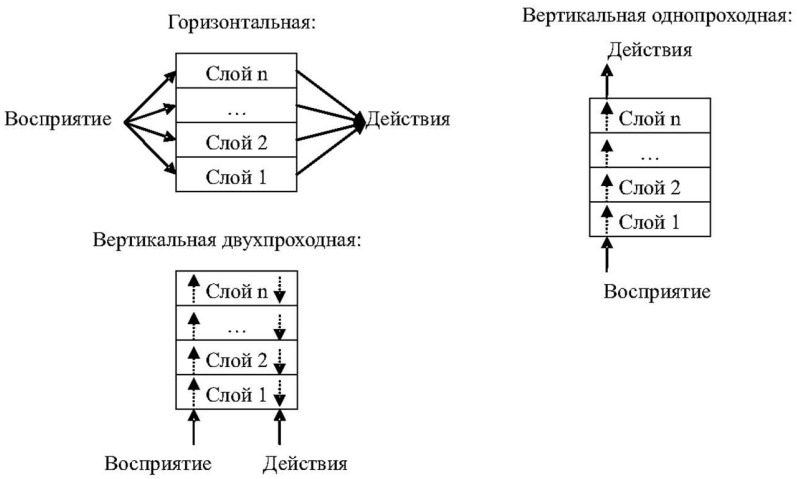

Рис. 1. Варианты многослойных реализаций агента

5.4. Многослойный агент

Как было показано выше, существует много подходов к реализации интеллектуальных агентов, каждый из которых обладает своими плюсами и минусами. Основная идея многослойной архитектуры заключается в том, что различные подходы можно сочетать в единой системе благодаря введению *архитектурных слоев* (*layers*), каждый из которых решает свой спектр задач наиболее эффективным для него способом.

Само определение интеллектуального агента уже подсказывает применение трех архитектурных слоев для реализации трех основных свойств (реактивности, проактивности и социальности). Однако ничто не мешает ввести гораздо большее число слоев. В целом многослойные архитектуры можно разделить на два больших класса:

- *Горизонтальные* — когда каждый из слоев напрямую подключен к входной информации о восприятии окружающей среды и к выходу для принятых решений.
- *Вертикальные* — когда напрямую и к входным, и к выходным данным непосредственно подключаются не более одного архи-

тектурного слоя. Вертикальные архитектуры, в свою очередь, можно поделить на *однопроходные* (когда к входу подключен самый нижний слой, а к выходу самый верхний) и *двухпроходные* (когда и к входу, и к выходу подключен нижний архитектурный слой).

Различные типы архитектур схематично отображены на рис. 1.

Большим преимуществом многослойных архитектур является модульность, а также возможность сочетания различных способов реализации. Если агент должен демонстрировать n различных типов поведения, то мы можем ввести n архитектурных слоев и реализовать каждый тип поведения наиболее удобным для этого способом.

Многослойная архитектура порождает проблему согласования решений, принятых на разных архитектурных слоях. Ведь то, что каждый из слоев в отдельности выполняет свою задачу корректно, не гарантирует того, что объединение соответствующих решений также будет корректным.

В горизонтальных архитектурах для устранения проблемы согласования можно, в частности, использовать единый модуль–диспетчер, принимающий решение о том, какой именно из архитектурных слоев должен управлять агентом в данный момент. Но такой диспетчер может значительно ограничить эффективность системы по двум причинам. Во-первых, весьма непросто такой механизм реализовать. Например, в архитектуре предусмотрено n слоев, каждый из которых способен предложить m возможных действий, следовательно, разработчику контролирующего механизма придется учесть m^n возможных сценариев взаимодействия слоев, что может иметь довольно высокую вычислительную сложность. Во-вторых, такой централизованный механизм является узким местом системы, что часто оказывается неэффективным и ненадежным.

Указанная проблема частично устраняется в вертикальных архитектурах. Рассмотрим, например, двухпроходную вертикальную архитектуру, в которой входные данные сначала поднимаются с нижнего слоя до верхнего, а затем принятое решение спускается с верхнего слоя до нижнего. Можно провести интересную параллель двухпроходной архитектуры с деятельностью организаций, когда информация поднимается от рядовых сотрудников к руководству, а затем от руководства спускаются приказы, которые рядовые со-

трудники выполняют. В случае n архитектурных слоев, каждый из которых может предложить m действий, мы имеем $2 * m * (n - 1)$ возможных сценариев взаимодействия, причем их реализация не сосредоточена в одном месте, а распределена по слоям. Однако вертикальные архитектуры имеют свои серьезные недостатки. Одним из них является то, что управление должно пройти через *каждый* архитектурный слой, прежде чем решение будет принято. Это, во-первых, не слишком эффективно, а во-вторых, опасно — сбой в любом из слоев приводит к полному краху системы.

6. Примеры

Проиллюстрируем рассмотренные выше подходы к определению архитектуры интеллектуальных агентов на примере модельной задачи проектирования автономного исследовательского зонда. Неформальная спецификация агента, реализующего соответствующую архитектуру, предполагает, что

- агент управляет устройством, перемещающимся в плоском, поделенном на клетки мире, в каждой клетке которого находится один из трех типов объектов — препятствие, полезные ископаемые или пустое пространство;
- агент способен воспринимать только содержимое клетки непосредственно перед ним и выполнить одно из трех действий — подобрать полезное ископаемое, продвинуться вперед или развернуться на 90 градусов по часовой стрелке: $A = \{pickup, move, turn\}$;
- задача агента заключается в сборе образцов полезных ископаемых и избегании столкновений с препятствиями.

Для оценки эффективности деятельности агента мы будем использовать функцию ϕ, значение которой не зависит от внутреннего состояния агента и динамика изменения этого значения определяется лишь внешними относительно агента закономерностями. При этом изменение функции ϕ является частью восприятия состояния внешней среды, поэтому в случае описанного выше агента-«собирателя» для моделирования оценочной функции можно ввести три параметра устройства, которые будут частью внешней среды для агента.

Итак, для моделирования оценочной функции описанного нами агента-собирателя можно ввести три параметра устройства, которые будут частью внешней среды агента:

- Количество собранных объектов $amount\_of\_samples$. Параметр увеличивается при каждом выполнении агентом действия *pickup*.
- Количество собранных полезных ископаемых $amount\_of\_gold$. Параметр увеличивается, если агент выполняет действие *pickup*, при условии, что перед ним находится образец полезных ископаемых.
- Состояние корпуса устройства $hull\_condition$. Данный параметр уменьшается, если агент при попытке осуществить действие *walk*, когда перед ним находится препятствие.

В итоге, функцию ϕ определим следующим образом:

$$\phi = amount\_of\_gold * 2 + hull\_condition - amount\_of\_samples.$$

Таким образом, значение оценочной функции увеличивается, если агент подобрал полезное ископаемое, и уменьшается, если агент столкнулся с препятствием или подобрал что-то, не являющееся полезным ископаемым.

6.1. Логический агент

При использовании подходов, основанных на формальной логике, восприятие агента описывается как набор утверждений этой логики. Для решения описанной выше задачи мы будем использовать формальную логику предикатов первого порядка, включающую следующие объекты:

- множества предметной области и предметные константы:
  - $O = \{wall, gold, empty\}$ — множество возможных объектов внешней среды;
  - $\Phi = \{-1, 0, 1\}$ — множество возможных изменений оценочной функции;

– $A = \{pickup, move, turn\}$ — множество возможных действий агента.

- предметные переменные:
  – $o \in O$ — объект, находящийся непосредственно перед устройством;
  – $\Delta\phi$ — изменение оценочной функции по сравнению с предыдущим состоянием.

- предикатные символы:
  – «=» — двуместный инфиксный предикат равенства, который может применяться к паре значений из любого множества, входящего в предметную область;
  – «<» и «>» — двуместные инфиксные предикаты числового сравнения, которые могут быть применены к паре значений из множества Ф;
  – $Done$ — одноместный предикат, применимый к значениям из множества A и используемый для обозначения того, что на предыдущем шаге агентом было выполнено действие $a \in A$.

Восприятие агента является выражением вида

$$o = obj \wedge \Delta\phi \bowtie 0,$$

где obj есть предметная константа из множества O, а $\bowtie$ — любой из предикатов численного сравнения или равенства. Таким образом, восприятие содержит в явном виде два факта: какой объект внешней среды находится непосредственно перед устройством и как изменилось значение оценочной функции по сравнению с предыдущим шагом.

Цели агента тоже можно описать в виде утверждений формальной логики следующим образом:

- Основной целью агента является увеличение значения функции ϕ, что можно записать с помощью выражения $\Delta\phi > 0$, т. е. целью для агента являются те состояния внешней среды, для которых значение оценочной функции будет большим, чем для текущего. Данной цели присвоим вес $w = 2$.

- Дополнительной целью для агента является недопущение снижения значения функции ϕ, что можно записать с помощью выражения $\neg(\Delta\phi < 0)$, т. е. целью агента является избежание состояний внешней среды, для которых значение оценочной функции будет меньше, чем для текущего. Данной цели присвоим вес $w = 1$.

Теперь опишем базу знаний Ω (здесь и далее имеется в виду лишь множество фактов базы знаний агента), используемую агентом для реализации прогнозирующей функции. В базе знаний агент хранит выражения следующего вида

$$(o = obj \wedge Done(act)) \Rightarrow \Delta\phi \bowtie 0,$$

где obj есть предметная константа из множества O, $\bowtie$ — любой из предикатов численного сравнения или равенства, а act — предметная константа из множества A. Таким образом, агент запоминает, что определенное действие, выполненное в определенных условиях, ведет к определенному изменению оценочной функции.

В общем случае, при использовании формальной логики прогнозирующая функция вычисляет набор возможных логических следствий конъюнкции всех выражений базы знаний, текущего восприятия и предположения, что агент выполнил определенное действие. Простая прогнозирующая функция (см. определение 12) логического агента может быть описана следующим образом:

$$prog(a) = \left\{ \varphi \mid p_{cur} \wedge Done\,(a) \wedge \bigwedge_{\xi \in \Omega} \xi \rightarrow \varphi \right\},$$

где p_{cur} есть выражение, описывающее восприятие текущего состояния внешней среды агентом.

Для определения вероятностной прогнозирующей функции можно использовать следующий принцип:

- если утверждение φ является логическим следствием конъюнкта, то вероятность его выполнения равна 1;
- если отрицание утверждения φ НЕ является логическим следствием конъюнкта, то вероятность его выполнения равна 1/2.

В этом случае вероятностная прогнозирующая функция (см. определение 12) может иметь вид

$$prog(a) \quad = \quad \left\{ (\varphi, 1) \mid p_{cur} \wedge Done\,(a) \wedge \bigwedge_{\xi \in \Omega} \xi \to \varphi \right\}$$

$$\bigcup \left\{ (\varphi, 1/2) \mid p_{cur} \wedge Done\,(a) \wedge \bigwedge_{\xi \in \Omega} \xi \nrightarrow \neg\varphi \right\},$$

где p_{cur} есть выражение, описывающее восприятие текущего состояния внешней среды агентом.

Если и для представления прогнозирующей функции, и для представления целей используется одна и та же логическая система, то нет необходимости просчитывать все возможные логические следствия, а достаточно лишь проверить, являются ли следствиями целевые утверждения или их отрицания. В случае описываемого агента для принятия решения достаточно рассмотреть выводимость всего лишь четырех утверждений: $\Delta\phi > 0$, $\neg\,(\Delta\phi > 0)$, $\neg\,(\Delta\phi < 0)$, $\neg\neg\,(\Delta\phi < 0) = \Delta\phi < 0$.

На рис. 2 отображен пример процесса взаимодействия такого агента с внешней средой (черные клетки означают стены, серые – образцы полезных ископаемых, а белые — пустое пространство).

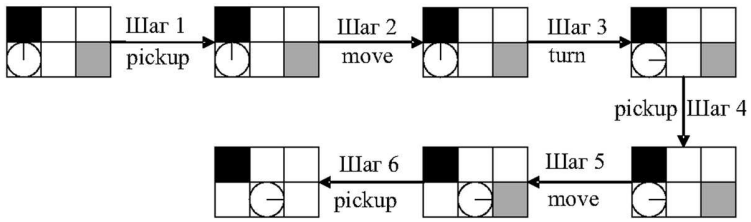

Рис. 2. Перемещение агента

На *первом шаге* агента база знаний не содержит накопленного опыта, а только информацию о текущем состоянии среды $p = wall$. Следовательно, оценка прогнозирующей функции равна $1/2$ во всех случаях, и агент выберет первое в списке действие *pickup*. В этом случае значение оценочной функции уменьшится, и в базе знаний агента появится первый факт:

$$(p = wall \wedge Done(pickup)) \Rightarrow \Delta\phi < 0$$

На *втором шаге* агент, руководствуясь фактом 1, заключает, что выполнение действия *pickup* не целесообразно, так как в этом случае вероятность достижения второй цели равна 0: $p = wall \wedge Done(pickup) \wedge ((p = wall \wedge Done(pickup)) \Rightarrow \Delta\phi < 0) \longrightarrow \Delta\phi < 0$, поэтому агент выберет второе действие из списка — *move*. Значение оценочной функции при этом, очевидно, уменьшится, и в базе знаний агента появится следующий факт:

$$(p = wall \wedge Done(move)) \Rightarrow \Delta\phi < 0$$

На *третьем шаге* агент, руководствуясь фактами 1 и 2, заключает, что выполнение действий *pickup* и *move* не целесообразно, так как в этом случае вероятность достижения второй цели равна 0. Следовательно, агент выбирает действие *turn*. При этом значение оценочной функции не изменится, но изменится ориентация агента. В базе знаний появится следующий факт:

$$(p = wall \wedge Done(turn)) \Rightarrow \Delta\phi = 0$$

На *четвертом шаге* агент оказывается в ситуации, когда ни один из ранее полученных фактов не может помочь выбрать действие, так как изменилось состояние окружающей среды: $p = empty$. Следовательно, агент выберет первое в списке действие *pickup*, что приведет к уменьшению значения оценочной функции и появлению в базе знаний факта:

$$(p = empty \wedge Done(pickup)) \Rightarrow \Delta\phi < 0$$

На *пятом шаге* агент, руководствуясь фактом 4, заключает, что выполнение действия *pickup* не целесообразно, следовательно, будет выбрано второе по списку действие *move*, что не приведет к изменению оценочной функции, но приведет к изменению позиции агента и появлению в базе знаний факта

$$(p = empty \wedge Done(move)) \Rightarrow \Delta\phi = 0$$

На *шестом шаге* агент опять оказывается в новой для него ситуации: $p = gold$, что приведет к выбору действия *pickup*. При этом впервые агент получит позитивный опыт — значение оценочной функции увеличится, а в базе знаний появится факт:

$$(p = gold \wedge Done(pickup)) \Rightarrow \Delta\phi > 0$$

В итоге — после шести итераций принятия решения база знаний агента включит в себя следующие значимые для принятия решения факты:

- $(p = gold \wedge Done(pickup)) \Rightarrow \Delta\phi > 0$ — благодаря этому факту агент будет подбирать образцы полезных ископаемых;
- $(p = wall \wedge Done(pickup)) \Rightarrow \Delta\phi < 0$ и $(p = empty \wedge Done(pickup)) \Rightarrow \Delta\phi < 0$ — благодаря этим фактам агент не будет собирать что-либо, помимо полезных ископаемых;
- $(p = wall \wedge Done(move)) \Rightarrow \Delta\phi < 0$ — благодаря этому правилу агент будет избегать столкновений с препятствиями.

6.2. Полное перечисление

Реализация агента с помощью метода полного перечисления и полной фактографии может оказаться значительно проще и эффективнее для не слишком объемных задач. Рассмотрим применение этих методов для реализации контроллера устройства-собирателя.

Множество возможных восприятий агентом состояния внешней среды можно определить следующим образом:

$$P = \{wall, gold, empty\} \times \{-1, 0, 1\}$$

Целями агента являются состояния внешней среды, воспринимаемые как

- $\{(x, 1) \mid x \in \{wall, gold, empty\}\}$ — такие состояния внешней среды отличает то, что значение оценочной функции увеличилось по сравнению с предыдущим шагом. Данной цели присвоим вес $w = 2$;
- $\{(x, y) \mid x \in \{wall, gold, empty\}, y \in \{0, 1\}$ — такие состояния внешней среды отличает то, что значение оценочной функции не уменьшилось по сравнению с предыдущим шагом. Данной цели присвоим вес $w = 1$.

База знаний Ω агента будет состоять из троек

$$(o, a, \Delta\phi) \in \{wall, gold, empty\} \times A \times \{-1, 0, 1\}$$

где o — объект внешней среды, находившийся перед устройством на предыдущем шаге, a — предпринятое агентом действие, а $\Delta\phi$ — изменение оценочной функции по сравнению с предыдущим шагом.

Прогнозирующую фунцкцию можно определить с помощью следующего алгоритма.

- Если тройка $(o, a, \Delta\phi)$, где o есть находящийся непосредственно перед устройством объект внешней среды, входит в базу знаний ($(o, a, \Delta\phi) \in \Omega$), то вероятность перехода внешней среды в состояние, воспринимаемое как $(x, \Delta\phi)$ (где $x \in \{wall, gold, empty\}$), при выполнении агентом действия a равна 1.
- Если ни одна тройка вида (o, a, z), где o есть находящийся непосредственно перед устройством объект внешней среды, а $z \in \{-1, 0, 1\}$, НЕ входит в базу знаний ($(o, a, z) \notin \Omega$), то вероятность перехода внешней среды в состояние, воспринимаемое как (x, z) (где $x \in \{wall, gold, empty\}$), при выполнении агентом действия a равна $1/2$.

Нетрудно заметить, что поведение такого агента будет эквивалентно поведению агента, реализованого с использованием формальной логики, как было описанно в разделе 6.1. При этом полностью заполненая база знаний такого агента будет содержать следующие значимые факты:

- $(gold, pickup, 1)$ — благодаря этому факту агент будет подбирать образцы полезных ископаемых;
- $(wall, pickup, -1)$ и $(empty, pickup, -1)$ — благодаря этим фактам агент не будет собирать что-либо, помимо полезных ископаемых;
- $(wall, move, -1)$ — благодаря этому правилу агент будет избегать столкновений с препятствиями.

Заключение

Интеллектуальные агенты и мультиагентные системы на данный момент являются одной из наиболее привлекательных областей как теоретической, так и практической информатики. В данной работе рассмотрены некоторые методы построения и применения отдельного интеллектуального агента, тогда как связанные с

интеграцией нескольких агентов в единую систему задачи остались за рамками и являются предметом дальнейшего исследования. К таким задачам можно отнести, например, организацию коммуникации, кооперации, совместного планирования, разделения задач и результатов, совместного обучения, проблему безопасности и т. д. Данным задачам уделяется немало внимания в зарубежной научной литературе — коммуникация и кооперация подробно рассмотрены в [4], совместное планирование в [11], а совместное обучение в [12]. В отечественной же литературе эта тематика представлена слабо, и этот пробел мы планируем со временем частично заполнить.

Список литературы

[1] *Wooldridge M. J.* The Logical Modeling of Computational Multi-Agent Systems. PhD thesis. — Manchester. — 1992.– 153 p.

[2] *Wooldridge M. J.* Intelligent Agents // Multiagent Systems. — 2001. — P. 27–79.

[3] *Wooldridge M. J., Jennings N. R.* Intelligent Agents: Theory and Practice // The Knowledge Engineering Review. — 1995.

[4] *Huhns M. N., Stephens L. M.* Multiagent Systems and Societies of Agents // Multiagent Systems. — 2001. — P. 79–121.

[5] *Jennings N. R., Wooldridge M. J.* Applications of Intelligent Agents. — London: Queen Mary & Westfield College, University of London. — 2000. — 27 p.

[6] *Miraftabi R.* Agents on the Loose: An Overview of Agent Technologies. — Joensuu: Department of Computer Sciense, University of Joensuu, — 2000. — 17 p.

[7] *Parunak H.* Van Dyke. Industrial and Practical Application of DAI // Multiagent Systems. — 2001, — P. 27–79.

[8] *Jennings N. R., Corera J., Laresgoiti I.* e. a. Using ARCHON to Develop Real-World DAI applications for electricity transportation management and Particle Acceleration Control // IEEE Expert Special Issue on Real World Applications of DAI systems. — 1996.

[9] *Parunak H.* Van Dyke. Applications of Distributed Artificial Intelligence in Industry // Foundations of Distributed Artificial Intelligence. — 1994.

[10] *Ljunberg M., Lucas A.* The OASIS Air Traffic Management System // Proceedings of the Second Pacific Rim International Conference on AI (PRICAI-92). — 1992.

[11] *Durfee E. H.* Distributed Problem Solving and Planning // Multiagent Systems. — 2001. — P. 121–165.

[12] *Sen S., Weiss G.* Learning in Multagent Systems // Multiagent Systems. — 2001. — P. 259–299.