Мультиагентно-орієнтоване програмування

3-й рівень навчання, осінь 2021

- Доц. Баклан І.В.
- Email: iaa@ukr.net
- Web: baklaniv.at.ua

Лекція 3

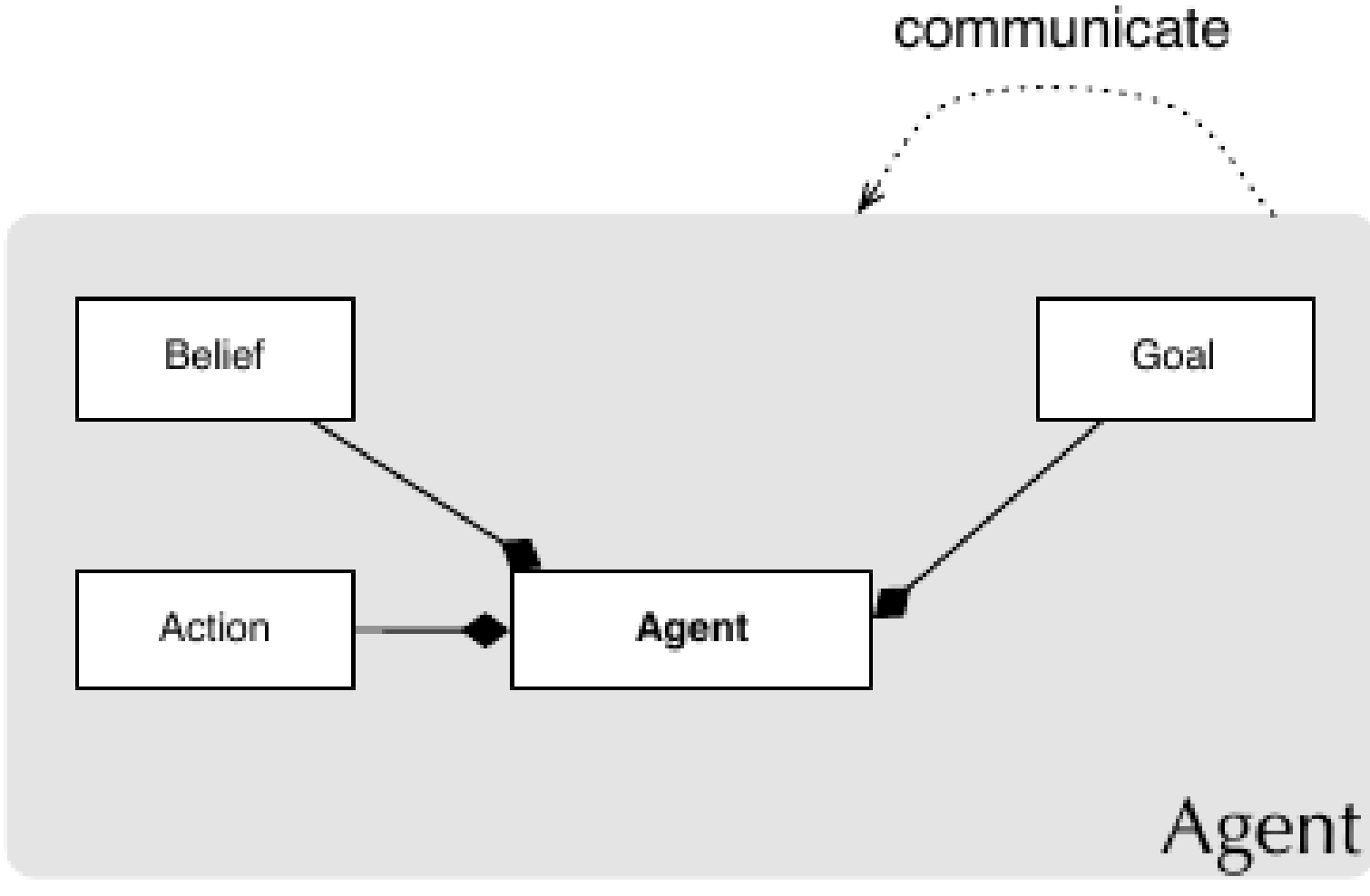
Вимір агентів в JaCaMo

У цій лекції ми заглиблюємось у деталі першого виміру МАОР, який ми розглядаємо в цій дисципліні, - виміру агента. Ми почнемо, згадуючи загальну картину цього виміру, наведену в попередньому розділі. Ми також прагнемо донести важливість цього виміру в контексті багатоагентного орієнтованого програмування. Потім ми розглянемо кожну окрему концепцію програмування та абстракцію, пов'язану з цим виміром у структурі JaCaMo, зазначивши, що більшість із цих концепцій використовується загальніше у МАОР. Після цього ми обговорюємо модель виконання агента, пояснюючи подальші концепції, пов'язані з агентом. Ми закінчуємо лекцію додатковими нотатками, включаючи деякі історичні замітки для студента, зацікавленого у відстеженні подій, що ведуть до сучасного стану техніки.

Огляд

Щоб допомогти нам згадати загальну картину про агентів та основні поняття в цьому вимірі, ми беремо з рисунка 1.3 лише вимір агента і показуємо його на малюнку 3.1. Підхід, який ми використовуємо до програмування агентів, базується на архітектурі BDI (див. четверту частину лекції для отримання довідки про цю архітектуру агента).

По суті, агенти мають переконання щодо поточного стану навколишнього середовища, а також переконання щодо інших агентів та стану організації, і мають цілі, які представляють майбутні стани середовища, бажані для агента (і, можливо, для його дизайнера) . На основі цих даних (як інформації про поточний стан, так і представництва держав, до яких він би хотів отримати), агент міркує, щоб прийняти рішення про найкращі дії, які слід вжити для досягнення бажаних ситуацій. Дія зазвичай змінює стан середовища, в якому знаходяться агенти, певним чином визначеним чином.

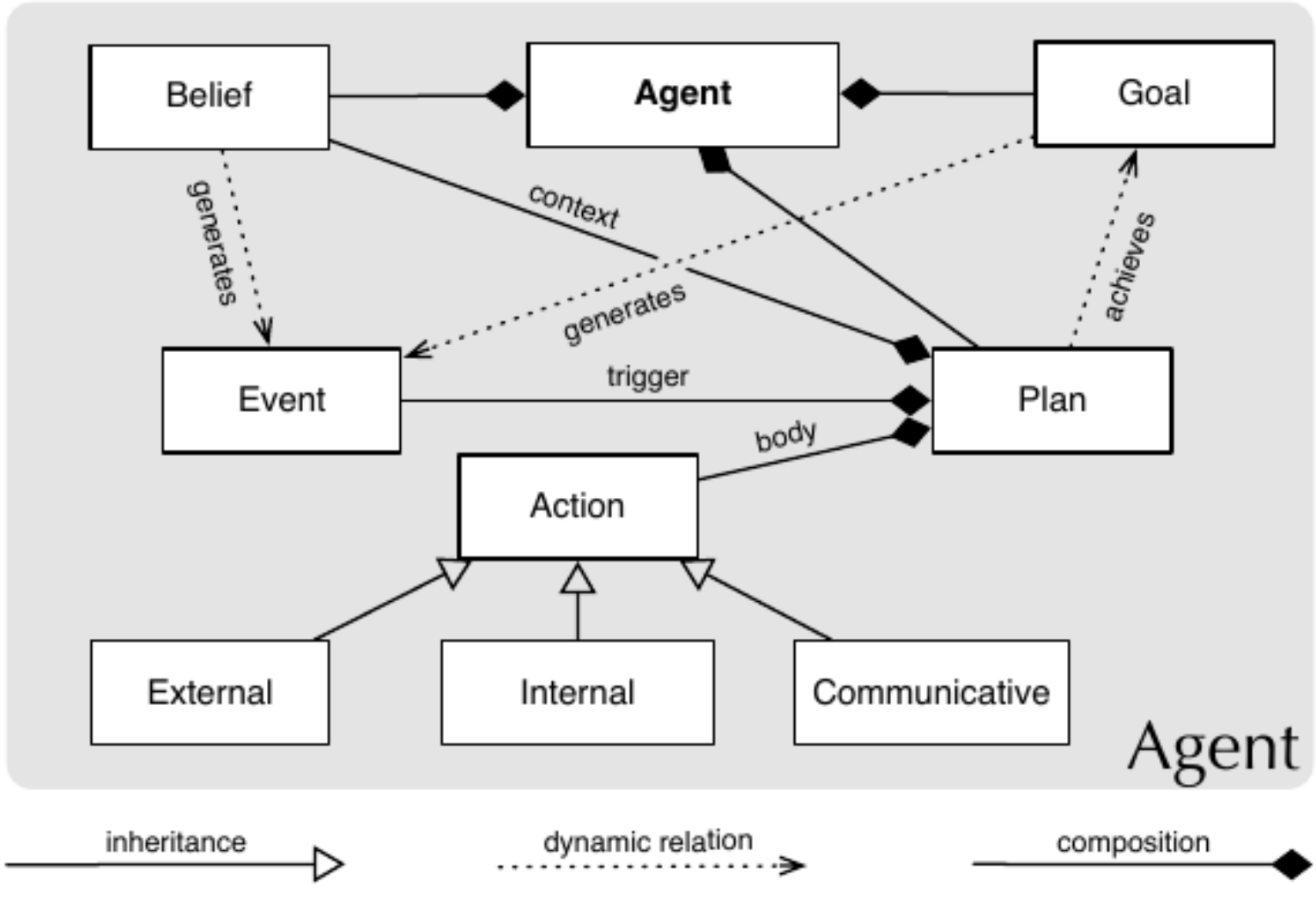


Мал.3.1 Основні поняття в вимірі агента.

Один особливий тип дій, також показаний на малюнку 3.1, який можуть здійснювати агенти, - це комунікативна дія, тобто дія, яка дозволяє агенту безпосередньо спілкуватися з одним або кількома, можливо, усіма іншими агентами в багатоагентній системі. Спілкування в МАОР ґрунтується на теорії мовленнєвих актів (ми згадуємо це в заключній частині лекції). На практиці це означає, що коли агент надсилає повідомлення іншому, окрім фактичного вмісту, що представляє певні знання, уподобання чи ноу-хау, існує явне представлення цільової мети повідомлення для агента, який надсилає повідомлення , що, у свою чергу, дозволяє агенту-одержувачу (агентам) знати, що робити із вмістом цього повідомлення. Призначена мета повідомлення виражається за допомогою перформативного дієслова, тобто такого слова, як сказати, досягти або запитати, що вплине на те, що агент-одержувач робить із змістом повідомлення.

Наприклад, якщо агент отримує повідомлення зі своїм вмістом, що виражає властивість, яка була завершена щодо торта в пекарні, наслідки для агента-одержувача будуть відрізнятися залежно від продуктивності, пов'язаної з повідомленням. Якщо перформатив "скажіть", агент-одержувач знатиме, що агент-відправник хотів, щоб він повірив, що торт закінчений. Якщо натомість перформатив має значення "запитати", агент-одержувач знатиме, що агент-відправник хоче, щоб він відповів, чи торт був готовий. Якщо перформатив був "досягнутим", агент-одержувач знатиме, що агент-відправник хотів, щоб він вжив заходів для того, щоб завершити торт.

Звичайно, в системі з декількома автономними агентами потрібна набагато подальша підтримка взаємодії агентів. Деякий з них охоплює організаційний вимір, і на лекції 8 ми бачимо на практиці, як це можна запрограмувати. Ми також обговорюємо в останній лекції такі теми, як аргументація.



3.2 Поняття в вимірі агента.

Агентські абстракції

Тепер ми розгортаємо рисунок 3.1, щоб показати всі поняття цього виміру на малюнку 3.2.

У наступному розділі ми розглянемо концепції, пов'язані з виконанням програми-агента, тобто структури, специфічні для середовища виконання, а не абстракції, що використовуються для програмування автономних агентів у підході, використаному в наших лекціях.

Інформація, якою наразі володіє агент, про його оточення, включаючи інших агентів у ньому та агентські організації, представлена як сукупність переконань. У переконаннях немає нічого особливого в порівнянні з тим, як інформація представлена в інших мовах програмування. Термін «переконання» корисний, щоб нагадати нам, що агенти зазвичай мають інформацію, яка може бути неправильною стосовно фактичного стану навколишнього середовища; наприклад, у разі, коли сфери застосування включають середовища, які постійно змінюються, часто непередбачуваними способами, або у випадку, коли отримання такої інформації (наприклад, за допомогою датчиків) може бути несправним або неточним. Крім того, агенти зазвичай не мають доступу до всієї наявної на даний момент інформації про спільне середовище.

Таким чином, термін віра нагадує нам, що агентам доводиться мати справу з недостовірною та неповною інформацією.

У стилі програмування, що використовується для вимірювання агента в JaCaMo, ми пишемо переконання за допомогою літералів, як це звичайно в логічному програмуванні, за винятком того, що у нас також є анотації, укладені в квадратні дужки. Вони можуть бути використані для зберігання метеоінформації про віру, включаючи спеціальну анотацію з назвою джерело, яку JaCaMo використовує для відстеження походження цієї інформації (наприклад, ім'я агента, сприйняття чи самості у випадку психічних записок створений самим агентом). Наприклад, якщо агент має у своїх переконаннях таку віру:

```
finished(cake)[source(percept)]
```

це означає, що агент наразі вважає, що пиріг був готовий, тому що він зазначив це, спостерігаючи за навколишнім середовищем. Зауважте, що замість терміну кекс для позначення певного торта можна було б використати змінну. Змінні позначаються ідентифікаторами, що починаються з великої літери (наприклад, `SomeCake`).

ідентифікатори, що починаються з великої літери (наприклад, якщо переконання дещо говорить про поточний стан речей, цілі позначають властивості спільного середовища, які агент хотів би перетворити в істину, і тому наразі не вважає, що це правда. Явні уявлення про цілі мають принципове значення в проактивній поведінці: вони ведуть агента до дії, включаючи спілкування з іншими агентами, для досягнення іншого стану речей. Для цього агенту потрібен план дій (згодом концепція плану обговорюється). Іншими словами, план містить рецепт дій, які можуть привести агента до досягнення однієї із своїх цілей. Ми також можемо використати сценарій пекарні для прикладу поставленої мети. Припустимо, ми не віримо, що торт був готовий. У цьому випадку ми можемо побажати, щоб він був завершений.

У нашій мові цілі виражаються подібно до переконань, за винятком того, що перед ними стоїть знак оклику!, а джерело мети говорить нам, агент, який делегував цю мету. Тому, якщо в нашому прикладі ми хочемо закінчити торт, оскільки кухар на ім’я Джон попросив нас закінчити його, це можна представити як

```
!finished(cake)[source(john)]
```

Окрім проактивної поведінки, ми також хочемо, щоб наш агент демонстрував реактивну поведінку. Якщо агент переслідує якусь мету від нашого імені, нам потрібно буде бути уважним до того, що відбувається в навколишньому середовищі, оскільки дії інших агентів у навколишньому середовищі можуть перешкодити агенту досягти його цілей або навіть допомогти йому. Обхід проблем та використання нових можливостей має першорядне значення для того, щоб агент проявляв розумну поведінку. Подія представляє один з двох різних видів речей: зміни цілей агента або зміни його переконань. Перший асоціюється з проактивною поведінкою, тоді як зміни у переконаннях важливі для реактивної поведінки. Зауважте також, що ми обговорюємо зміни, так що подія відображає, наприклад, що агент має нову мету для досягнення або що він більше не дотримується певних переконань.

Саме такі зміни (доповнення або видалення) насправді призводять агента до виконання плану. Наприклад, подія

```
-finished(cake)
```

означає, що агент більше не вірить, що торт закінчений (наприклад, тому, що агент зрозумів, що якийсь предмет прикраси впав з торта, і, можливо, захоче діяти відповідно до цієї події), тоді як подія

```
+!finished(cake)
```

означає, що агент щойно прийняв нову мету, щоб досягти стану, в якому агент вважатиме, що торт був готовий належним чином.

Яке досягнення цілей використовувати – декларативне чи процедурне?

Зазвичай цілі досягнення повинні бути запрограмовані декларативно; тобто мета - це той факт, який на даний момент агент вважає неправдою, і якщо мета досягнута, агент вважатиме, що відповідна пропозиція стала істинною. Наприклад, ціль `!finished(cake)` була використана декларативно. Агент має на меті закінчити торт, тому що він не вірить, що торт ще готовий. Після виконання всієї роботи, сподіваюся, агент подивиться на торт і побачить, що він зараз повністю готовий, і тому він повірить, що готовий (торт). Однак наш підхід MAOP не вимагає, щоб цілі були програмовані таким чином.

Ми також можемо використовувати їх процедурним способом як просто ім’я для послідовності дій, які ми б хотіли виконати агентом. Наприклад, припустимо, що у нас є процедура збивання вершків, а шеф-кухар просто хоче, щоб агент виконував цю процедуру; якщо вершки не правильно збити в кінці, це виправить це іншим способом. Якщо агент має план! `Whip_cream`, шеф-кухар може просто делегувати цю ціль досягнення допоміжному агенту. Коли всі дії будуть виконані, запитуване завдання буде завершено, незалежно від досягнутих результатів.

План — це рецепт дій, зокрема рецепт, який дозволяє агенту досягти однієї з можливих цілей, які він міг би мати, або який дозволяє йому реагувати на можливості чи потенційні проблеми, які сприймаються зі стану спільного середовища. Тобто план стосується конкретного типу події, яка має відношення до агента. Набір планів, які агент має в конкретний момент, становить його ноу-хау. Синтаксично план складається з трьох частин.

1. Тригер Тригер або ініційна подія плану чітко вказує, про яку мету чи реакцію йдеться у плані. Архітектура програмного забезпечення, що лежить в основі автономного агента, відстежує події, тобто нові цілі, які він має досягти (оскільки, наприклад, інший план вимагає цього або інший агент цього вимагав), і нові переконання, які вимагають від агента реакції. При вирішенні видатних подій, записаних у відповідній структурі архітектури агента, відповідними планами, які слід розглянути, є плани, які мають відповідний тригер. Наприклад, агент Джон має план впоратися з тим фактом, що у нього є новий екземпляр мети — закінчити торт, а помічники пекарні мають плани реагувати, коли помітять, що Джон намагається закінчити певний торт. Тригерна частина цих планів буде записана так:

```
+!finish(cake) : ...

+finishing(john, cake) : …
```

2. Контекст Може бути багато різних планів щодо обробки однієї події. Наприклад, щоб закінчити торт, можна підготувати глазур і прикраси, а потім закінчити торт, але, можливо, більш здоровим варіантом було б просто змастити зверху невеликою кількістю збитих вершків, щоб натомість дотримуватися простого стилю торта. За різних обставин застосовуються різні плани для однієї і тієї ж мети. Ось що говорить нам контекст плану. Зазвичай він складається з поєднань переконань, які агент повинен мати під час вибору плану дій для події:

```
+!finished(cake)
:   order(cake)[source(Client)]   &   lifestyle(Client,
healthy)
<- ...
+!finished(cake)
: is_for(cake, wedding)
<- …
```

3. Тіло Тіло плану - це рецепт дій. Як правило, дія - це те, що агент може виконати, щоб змінити стан середовища (згодом ми розповімо більше про дії). Орган плану визначає дії, які очікується здійснити агентом, щоб обробляти подію, яка викликала план (наприклад, новий екземпляр цілі або нещодавно набуту віру). Крім того, складні плани можуть вимагати від агента досягнення (інших) конкретних цілей серед дій, які необхідно виконати. Перш ніж ми розглянемо різні типи дій, які можуть з’явитися у плані, ми підсумовуємо приклади, розпочаті вище. Плани щодо Джона наступні.

```
+!finished(cake)
:order(cake)[source(Client)]& lifestyle(Client,healthy)
<- whip(cream);
spread(cream,top);
!have(decoration);
decorate.
+!finished(cake)
: is_for(cake, wedding)
<- !have(marzipan);
cover(cake,marzipan);
!piping_decorated(cake).
```

Перший план передбачає, що агент виконує дві дії одну за одною (збивання та намазування), щоб покласти трохи збитих вершків лише поверх торта. Тоді агенту знадобиться інший план для досягнення мети отримання необхідної прикраси, що в свою чергу може включати, наприклад, подрібнення горіхів. Нарешті, шеф -кухар виконує декоративну дію (припускаючи це одну конкретну дію, яку агент може безпосередньо виконати). Другий план використовується для красивішого вигляду торта, відповідного для більш офіційних заходів. Він передбачає покриття всього торта марципаном перед виконанням типово складного плану з використанням техніки трубопроводів, щоб прикрасити торт.

У свою чергу, помічники мають наступний план (який має реагувати на спостережувану подію і завжди застосовний) для досягнення мети допомогти кондитерові в приготуванні торта:

```
+finishing(john, cake) <- !assist_finishing(cake).
```

Досягнення цієї мети може спонукати цих агентів діяти, наприклад, таким чином, щоб приготувати або принести марципан, глазур, горіхи тощо до столу, де працює шеф-кухар. Звичайно, різні завдання потрібно скоординувати, оскільки, наприклад, деякі форми глазурі не можна готувати задовго до фактичного декорування, оскільки вона висохне. У наступній лекції ми розглянемо, як найкраще досягти такої координації.

Як згадувалося раніше, дія - це те, що може виконати агент; дія має один з наступних трьох типів:

Зовнішній Це те, що ми зазвичай вважаємо дією агента. *Зовнішня дія* виконується *ефекторами* або *актуаторами агента*, тобто специфічними засобами, які мають агенти для зміни середовища, в якому він знаходиться. Наприклад, якщо агент керує роботом за допомогою руки, переміщення цієї руки, як це реалізується базовим програмним забезпеченням керування, є зовнішньою дією. Термін зовнішній підкреслює, що цей тип дії реалізується поза межами міркування агента; міркування агента саме про те, які такі дії дозволять агенту досягти своїх цілей.

Внутрішня Внутрішня дія, навпаки, реалізується в архітектурі агента, і вона дозволяє агенту (атомарно) запускати деякий доступний фрагмент коду як частину свого циклу міркувань. Наприклад, якщо в якийсь момент агенту потрібно запустити деякий застарілий код або запустити фрагмент коду, який краще написаний традиційними мовами програмування (наприклад, деякий код обробки зображень), тоді потрібна внутрішня дія. Однак, оскільки ці фрагменти коду виконуються атомарно, слід подбати про те, щоб такі дії не блокували цикл міркувань (описаний у наступному розділі); інакше здатність агента реагувати на зміни та одночасно виконувати різні інші плани дій порушується.

Хоча внутрішні дії відносяться до будь-якого коду, що виконується в межах циклу міркувань агента, існує дуже важливий тип внутрішньої дії, який використовується для зміни його внутрішнього стану, точніше, для зміни психічних установок, які визначають психічний стан автономного агента. Наприклад, існують внутрішні дії, які можна використовувати, щоб змусити агента відмовитися від певної мети, яку він намагається досягти, або щоб перевірити, які поточні цілі переслідує агент. Оскільки вони використовуються для розширеного MAOP, ми не наводимо їх у цій лекції.

Комунікативні Ми також можемо окремо описати комунікативні дії, які дозволяють агентам безпосередньо спілкуватися один з одним. Як обговорювалося в попередньому розділі, ці дії використовуються для надсилання повідомлень іншим агентам. Натхнені теорією мовленнєвого акту, такі повідомлення явно представляють три різні речі: (1) агента або агентів, які повинні отримати повідомлення; (2) ілокутивна сила мовленнєвого акту, явно позначена перформативним дієсловом, таким як «розповісти» або «досягнути»; і (3) фактичний зміст повідомлення, наприклад, деякі знання чи ноу-хау.

На даному етапі варто згадати ще кілька конструкцій програмування. Крім цілей досягнення, мова програмування також має цілі тестування, які позначаються префіксом `?` а не `!`. Ця конструкція використовується для отримання інформації з бази переконань у тілі плану, щоб отримати найновішу інформацію. Наприклад, припустимо, що нам потрібно отримати те, що, на нашу думку, є поточним номером телефону клієнта, чий торт ми щойно закінчили; тоді ми могли б мати такий план:

```
+!finished(cake)
      : order(cake)[source(Client)]
     <- ... ;
          ?phone(Client, Number);
... .
```

Рядок коду, що починається з `?` перевіряє поточний стан бази переконань агента, щоб спробувати знайти номер для конкретного клієнта, екземпляр якого є змінною `Client`. Якщо ця інформація недоступна в базі переконань, може бути план, який розповідає агенту, як отримати таку інформацію. Якщо ми хочемо передати ці знання нашому агенту, ми можемо написати план, починаючи з цього:

```
+?phone(Client, Number) : ... <- …      .
```

Ще одна важлива особливість — вміти міркувати про переконання. З цією метою також можуть бути правила висновку, подібні до правил Прологу для міркування про віру в базу переконань, а також прості факти, які вважаються істинними агентом. Наприклад, припустимо, що якщо ми знаємо, що торт не містить продуктів тваринного походження або продуктів від компаній, які завдають шкоди тваринам, то можна зробити висновок, що це веганський торт. Ми могли б додати до переконань агента таке правило:

```
vegan(Cake) :- has_no_animal_products(Cake) &
      not (uses(Cake,Product)&produced_by(Product,Co) &
      animal_harming(Co)).
```

Зауважимо, що `has_no_animal_products(Cake)` сам по собі може бути висновком з іншого правила, яке перевіряє весь список інгредієнтів, наприклад.

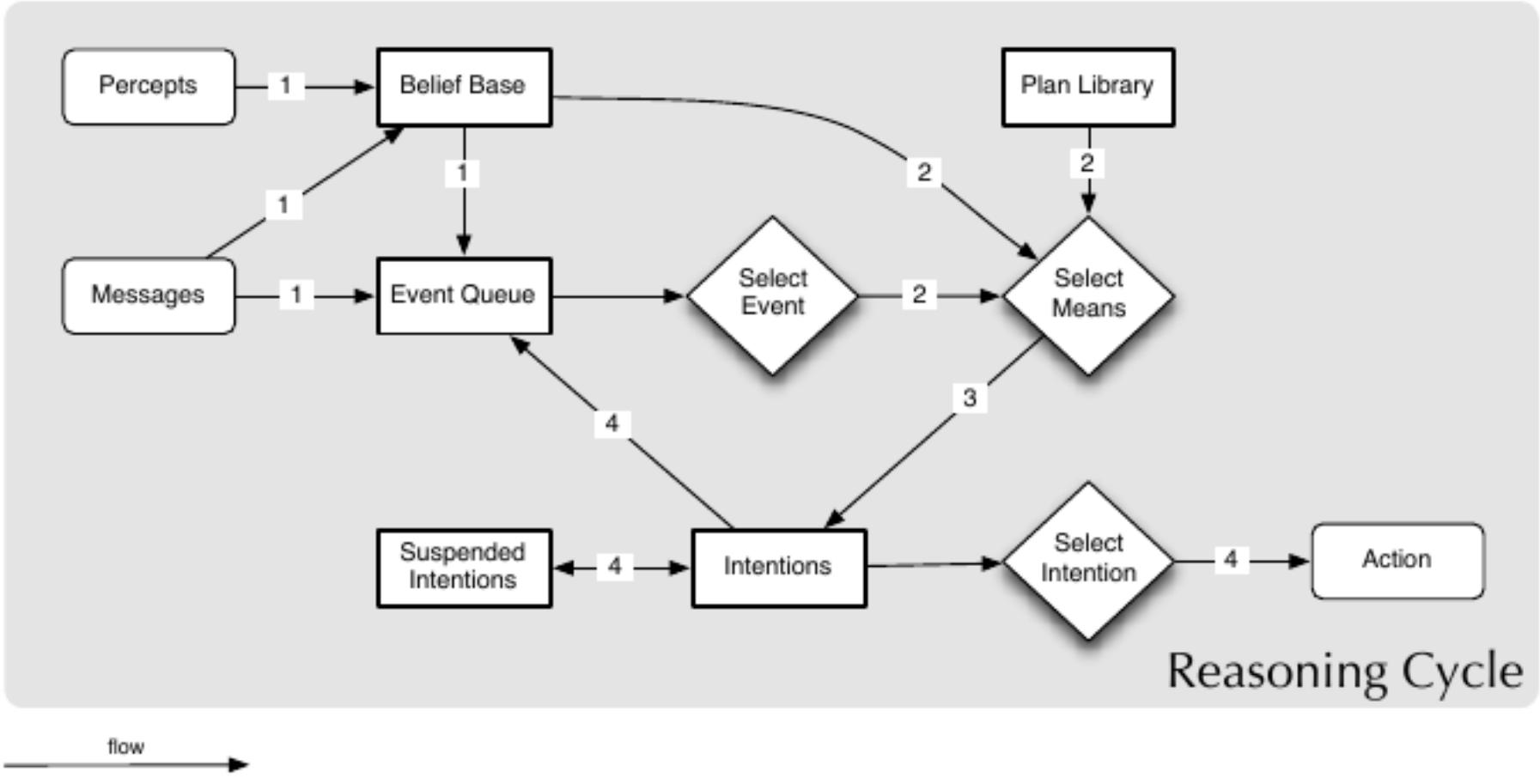


Рис. 3.3 Цикл міркування — концепції часу виконання у вимірі агента.

Виконання агента

Тепер перейдемо до способу визначення поведінки агента під час виконання. Агенти багаторазово проходять цикли міркувань, які починаються зі сприйняття стану спільного середовища і закінчуються вибором певної дії, яку потрібно виконати, можливо, зміною стану середовища. На малюнку 3.3 показані основні кроки циклу міркування агента.

Перш ніж описувати чотири основні кроки такого циклу міркувань, ми коротко пройдемося через визначення основних понять, показаних на цьому малюнку, та кількох інших пов’язаних понять.

Сприйняття — це символічно представлена частина інформації про стан спільного середовища. У реальних умовах сприйняття зазвичай отримують з датчиків або камер. Вони безпосередньо впливають на переконання агента, як це видно з наших подальше обговорення циклу міркувань.

Повідомлення - це частина зв'язку, отримана (асинхронно) від інших агентів. Як обговорювалося раніше, повідомлення можна використовувати, щоб повідомити агента про щось, попросити агента щось зробити або надати агенту план, який дозволить йому щось виконати. Як і багато інших аспектів міркування агента циклу, існує функція, яку можна визначити для кожного агента, щоб він вибирав лише відповідні повідомлення (наприклад, повідомлення з джерел, яким агент може довіряти, або повідомлення з джерел, які мають відповідні повноваження). Подальші деталі наведені Bordini et al. (2007, розділ 7).

База переконань — це структура, яка відстежує всю інформацію, доступну на даний момент агенту. Це може включати інформацію про стан середовища (як сприйнятий самим агентом, так і інформацію про інших агентів), інформацію про інших агентів і речі, які агент хотів запам'ятати про себе в попередніх внутрішніх станах. Усі переконання в базі переконань анотовані міткою, яка чітко вказує джерело цього переконання. Джерелом може бути сприйняття у випадку переконання, що походить від інформації, зібраної агентом від його власних датчиків. Уявлення, створені в результаті спілкування з іншими агентами, анотуються іменем агента, який надіслав повідомлення. Нарешті, джерелом може бути сам агент (щось на кшталт заміток, згаданих раніше). Анотації до переконань також можна використовувати для різних цілей, як показано в більш складних прикладах у наших лекціях.

Агент програма зазвичай має принаймні кілька переконань і кілька планів. Останнє дає агенту початкове ноу-хау, тоді як переконання в коді формують початковий стан бази переконань агента (тобто переконання, які буде мати агент, коли він вперше починає працювати).

Черга подій відстежує всі події, які фактично відбулися. Нагадаємо, що подія означає деяку зміну психічного стану агента. Наприклад, агент має нову мету, яку потрібно досягти, або переконання, що виникло внаслідок сприйняття, яке було істинним у попередньому циклі міркувань, більше не діє, тому переконання було автоматично видалено (це лише приклади можливих подій, які можуть бути в черга подій).

Бібліотека планів зберігає всі ноу-хау агента. Це сукупність планів — часто різних планів для однієї інцидентної події. Як пояснювалося вище, для досягнення однієї і тієї ж мети, наприклад, можна мати різні плани, які використовуються в різних контекстах. Нагадаємо, що програма-агент забезпечує не лише початковий стан бази переконань, а й бібліотеки планів. Однак зверніть увагу, що агенти можуть змінювати свої ноу-хау під час їх виконання. У той час як переконання, як правило, змінюються постійно, плани, як правило, змінюються не так сильно; однак змінити плани в бібліотеці планів агента не тільки можливо, але й досить легко зробити. Агенти можуть обмінюватися планами через комунікацію, або методи планування AI можуть використовуватися для нових планів, які будуть створені агентом (див. посилання в цій лекції).

Враховуючи конкретну подію, план вважається релевантним, якщо його ініційна подія відповідає цій конкретній події. План називається застосовним, якщо його контекст відповідає дійсності, враховуючи поточні переконання агента. Часто агент може мати різні застосовні плани для певної події. Раніше ми бачили приклад, у якому було два різних плани закінчити торт. Якщо ви знову подумаєте про ці плани, зверніть увагу на те, як обидва можуть бути застосовні, якщо клієнт, який веде здоровий спосіб життя, одружується. Зауважте також, що обидва плани досягають однієї і тієї ж мети — завершення торта — але різними способами. Зрозуміло, що для конкретного торта потрібно вибрати лише один із цих планів для виконання. Як і переконання, плани також можуть мати анотації, і одне з них — допомогти вибрати план, коли різні плани застосовні до поточного контексту.

Коли агент вибирає один план для обробки певної події, копія цього плану з бібліотеки планів стає призначеним засобом. Тобто, якщо подія має на меті досягти нової мети, агент зобов'язується це зробити, дотримуючись того рецепту дій, який виражається в цьому передбачуваному засобі.

Усі передбачувані засоби, обрані агентом, входять у набір намірів. Намір — це набір передбачуваних засобів, тому що план може вимагати досягнення (під)цілі, перш ніж виконуватимуться подальші дії, і ця залежність між деякими передбачуваними засобами також повинна зберігатися. Кожен окремий намір у наборі намірів представляє різний фокус уваги для агента. Наприклад, агент Джон може розгортати марципан, коли спрацьовує таймер, щоб сказати, що тісто може бути готове незабаром, тому він буде стежити за духовкою, розгортаючи марципан.

Як обговорювалося, дія — це те, що агент здатний зробити, щоб змінити стан спільного середовища. Часто дії пов’язані з операціями, доступними в артефактах в моделі середовища або реалізованими ефекторами (наприклад, за допомогою механічних засобів, доступних у роботі). З точки зору аргументації агента, дії, які необхідно виконати, визначаються безпосередньо з поточного набору намірів агента.

Архітектура агента використовує три визначені користувачем функції вибору. Насправді для них доступні реалізації за замовчуванням, але можливо, що для певних додатків потрібні функції, специфічні для домену. Три функції вибору такі:

Подія Функція вибору події вибирає одну конкретну подію з черги подій для обробки в кожному циклі міркувань. Зазвичай функції вибору реалізують політику FIFO, звідси й назва черги подій. Однак певним агентам може знадобитися, наприклад, визначити пріоритети певних конкретних подій, і в цьому випадку потрібна функція вибору, визначена користувачем.

Засоби Функція вибору засобів (часто її називають функцією вибору опцій) вибирає один конкретний план, який буде виконано, якщо в даний момент агенту доступні кілька застосовних планів. Зазвичай використовується перший застосовний план у порядку їх появи в бібліотеці планів; однак, певний агент може, наприклад, захотіти міркувати про властивості, анотовані в планах, щоб вибрати план, який має більші шанси на успіх або який зазвичай має вищі виплати.

Намір Функція вибору намірів вибирає одну конкретну інтенцію серед наявних у наборі намірів (нагадаємо, що кожен окремий намір відноситься до різного фокусу уваги агента щодо його середовища, включаючи інших агентів). Це, у свою чергу, визначає, які дії будуть вжиті в цьому конкретному випадку цикл міркувань. Зазвичай ця функція керує цикловою політикою, тобто дає справедливу можливість для всіх намірів виконати дії. Знову ж таки, окремим агентам можуть знадобитися, наприклад, складні форми міркування про взаємозв'язки між намірами, щоб гарантувати, що вони плануються ефективно.

Наміри можуть бути призупинені з різних причин. Наприклад, дії виконуються ефекторами агента в частині архітектури агента, відмінній від частини аргументу агента. Отже, після того, як намір запитує виконання дії, і до того, як запит на виконання дії буде підтверджено, намір призупиняється в тому сенсі, що подальші дії цього наміру тимчасово не можуть бути обрані для виконання. Крім того, поки намір очікує вибору плану для досягнення підцілі, він не повинен бути обраний для виконання, і тому він тимчасово призупинений.

Тепер ми перейдемо до високорівневого опису циклу міркування агента, як показано на малюнку 3.3. Наступний опис процесу циклу міркування посилається на цей малюнок. Кожен цикл міркувань починається з отримання всієї доступної інформації, що надходить із середовища, і закінчується вибором однієї дії, яка має бути виконана. Структури, які взаємодіють між резонатором агента та іншими компонентами архітектури агента (наприклад, датчиками та виконавчими механізмами), показані у вигляді округлених прямокутників. Інші основні структури даних показані у вигляді квадратів із товщею рамкою. Функції вибору відображаються як ромби. Групуючи пов’язані частини процесу міркування, ми можемо розділити цикл на чотири основні кроки таким чином:

1. Кожен цикл міркувань починається з вилучення з частини архітектури агента, відповідальної за взаємодію із спільним середовищем, поточного набору доступних сприйняття, а також повідомлень, що надходять від інших агентів. Сприйняття безпосередньо впливають на переконання агента, тоді як повідомлення можуть впливати як на набір переконань, так і на цілі та плани агента. Нагадаємо, що всі зміни в переконаннях або цілях представлені у вигляді подій і поміщені в чергу подій. У базі переконань агента переконання, що виникли від сприйняття в попередньому циклі міркувань, явно анотуються як такі. Це дозволяє архітектурі агента визначати зміни в перцептивній інформації.

2. У цій частині міркування функція вибору події використовується для вибору однієї події, для якої всі плани з відповідними ініційними подіями витягуються з бібліотеки планів. З цим набором відповідних планів нам потрібно звірити контекст кожного плану з поточним станом бази переконань, щоб визначити набір застосовних планів.

3. На цьому етапі викликається функція вибору засобів з набором застосовних планів. Його результатом є конкретні передбачувані засоби, обрані агентом для обробки події, вибраної на кроці 2. Це має перейти до набору намірів або як новий намір (якщо цей план починає новий фокус уваги для агента, для наприклад, тому, що він реагує на якусь нещодавно сприйняту зміну в навколишньому середовищі або тому, що приймає мету, запитану іншим агентом) або як частина наявного наміру у випадку, якщо обраний план має досягти мети, яку необхідно досягти як частина іншого плану.

4. Нарешті, функція вибору наміру вибирає один конкретний намір, і наступна необхідна дія цього наміру вибирається для виконання в цьому циклі міркувань. Отже, наміри безпосередньо визначають дії, які необхідно виконати. Нагадаємо, що в плані можуть з’явитися три різні типи дій, і насправді частина плану, обрана для виконання, може бути не саме дією, а, наприклад, новою метою, яку агент повинен прийняти, або доповненням. переконання (або розумову нотатку) про те, що агенту пізніше потрібно буде запам’ятати, і тому базу переконань і чергу подій, можливо, потрібно буде оновити як частину цього кроку.

Крім того, намір може бути призупинено (як пояснювалося раніше), і в усіх випадках набір намірів необхідно оновити: намір може бути переміщено до призупиненого набору або черги подій, щоб дочекатися плану для мети, і якщо він зберігається в наборі намірів, принаймні той факт, що була виконана ще одна дія активного плану, необхідно оновити в цьому намірі.

Джерела до лекції 3

Boissier, Olivier, Rafael Bordini, Jomi Fred Hübner, Alessandro Ricci, and Andrea Santi. 2013. Multi-agent oriented programming with JaCaMo. Science of Computer Programming 78 (6): 747–761. https://doi.org/10.1016/j.scico.2011.10.004 .

Boissier, Olivier, Rafael H. Bordini, Jomi F. Hübner, and Alessandro Ricci. 2019. Dimensions in programming multi-agent systems. The Knowledge Engineering Review 34. https://doi.org/10.1017/S026988891800005X .

Bordini, Rafael H., Jomi Fred Hübner, and Michael Wooldridge. 2007. Programming multi-agent systems in AgentSpeak using Jason. John Wiley & Sons.

Ricci, Alessandro, Michele Piunti, Mirko Viroli, and Andrea Omicini. 2009. Multi-agent programming: Languages, tools and applications, eds. Amal El Fallah Seghrouchni, Jürgen Dix, Mehdi Dastani, and Rafael Bordini, 259–288. Springer. https://doi.org/10.1007/978-0-387-89299-3_8 .

Hübner, Jomi Fred, Jaime Simão Sichman, and Olivier Boissier. 2007. Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. International Journal of Agent-Oriented Software Engineering 1 (3-4): 370–395.

Наступна лекція буде присвячена детальному програмуванню виміру середовища на платформі JaCaMo.