Изкуствен интелект - летен семестър, 2023/2024 учебна година

Тема 7: Представяне и използване на знания чрез системи от продукционни правила

Обща характеристика на системите от продукционни правила като формализъм за ПИЗ

Декларативен в основата си формализъм с елементи на процедурност на по-ниско ниво. Негова основна характеристика е декомпозирането на знанията на малки части (правила) от типа условие – следствие (ситуация – действие).

Архитектура на системите, основани на правила

- работна памет (контекст)
- база от правила
- интерпретатор на правилата

Работна памет

Съдържа данните за конкретно решаваната задача, които са установени (известни) към текущия момент. Произход на данните в работната памет:

- от потребителя (зададени по условие при дефинирането на задачата или получени като отговор на уточняващ въпрос, зададен от интерпретатора на правилата в процеса на обратен извод);
- от интерпретатора на правилата (получени в процеса на логическия извод).

Работната памет е списък от т. нар. *елементи на работната памет* (ЕРП). Структурата на ЕРП е близка до (дори обикновено съвпада със) структурата на елементите на левите страни на правилата (т. нар. *елементи на условието* в левите страни на правилата). Най-често ЕРП описват отделни обекти от предметната област чрез принадлежността им към съответни типове и множеството от стойности на определени атрибути (свойства/характеристики). В такъв случай всеки ЕРП е вектор от вида

(*type* $attribute_1$:$value_1$ . . . $attribute_n$:$value_n$),

където *type* и всички $attribute_i$ и $value_i$ са атоми.

База от правила

Съхранява знанията за предметната област, представени под формата на импликации от вида

(<име_на_правило> <лява_страна> <дясна_страна>)

Лявата страна на правилото се нарича негово *условие* (*предпоставка*), а дясната страна на правилото – негово *следствие* (*действие*).

Най-често лявата страна на правилото представлява конюнкция от т. нар. елементи на условието (ЕУ). ЕУ могат да бъдат положителни или отрицателни (ще смятаме, че отрицателните ЕУ имат вида ~cond, където cond е положителен ЕУ).

Положителните ЕУ обикновено са вектори от вида

(*type* *attribute*$_1$:*specification*$_1$ . . . *attribute*$_n$:*specification*$_n$),

където всеки спецификатор *specification*$_i$ може да бъде:

- атом;
- променлива;
- израз, който може да бъде оценен (такива изрази ще заграждаме в квадратни скоби „[ ]“);
- тест (сравнение, заградено в „[ ]“);
- конюнкция (∧) на спецификатори, дизюнкция (∨) на спецификатори или логическо отрицание (¬) на спецификатор.

Дясната страна на едно продукционно правило обикновено описва поредица от действия и има процедурна интерпретация. Списъкът на допустимите действия в десните страни на правилата обикновено включва поне следните видове действия:

- ADD *pattern*: означава добавяне към работната памет на нов ЕРП, специфициран от образеца *pattern*;
- REMOVE *i*: означава изключване от работната памет на ЕРП, който е съпоставен успешно с *i*-тия ЕУ в лявата страна на правилото;
- MODIFY *i* (*attribute specification*): означава модифициране на ЕРП, който е съпоставен успешно с *i*-тия ЕУ в лявата страна на правилото, като съществуващата стойност на атрибута *attribute* в този ЕРП се заменя със *specification*.

Интерпретатор на правилата

Програмна система, чието основно предназначение е да приложи описаните чрез правилата знания върху данните за конкретната задача.

Процесът на работа на интерпретатора на правилата е цикличен, като всяка стъпка от работния цикъл на интерпретатора се разделя на две фази: *избор на правило* и *изпълнение на избраното правило*.

При избора на правило, което да се изпълни на текущата стъпка, се извършват множество операции на *съпоставяне по образец*.

Стратегии на работа на интерпретатора на правилата:

- *прав извод* (forward chaining) или *извод, управляван от данните* (data-driven inference). Пример: OPS5;
- *обратен извод* (backward chaining) или *извод, управляван от целите* (goal-driven inference). Пример: Пролог.

Прав извод: съпоставят се елементите на условията в левите страни на правилата и елементите на работната памет. Генерират се верните следствия от съдържанието на работната памет и базата от правила.

Обратен извод: съпоставят се елементите на десните страни на правилата и предварително зададена цел. Проверява се дали целта е следствие от съдържанието на работната памет и базата от правила.

Конфликтно множество: множеството от правила, чиито леви страни се удовлетворяват от съдържанието на работната памет на текущата стъпка от работния цикъл на интерпретатора.

Стратегии за разрешаване на конфликтите: стратегии за избор на правило от конфликтното множество (избор на първото възможно правило, което не предизвиква зацикляне; избор на най-актуалното правило; избор на правилото с най-сложното/най-простото условие и др.).

Пример за прав извод върху система, основана на продукционни правила, при която не е необходимо определяне и използване на стратегия за разрешаване на конфликтите

Имаме три тухли с различни размери, поставени на купчина една върху друга. Определени са три различни позиции, в които искаме да поставим тухлите с помощта на ръката на интелигентен робот. Означаваме тези позиции като 1, 2 и 3. Нашата цел е да поставим тухлите в тези позиции по реда на техния размер, като най-голямата е в позиция 1, а най-малката е в позиция 3.

Приемаме, че при стартиране на интерпретатора на правилата работната памет съдържа следните елементи:

```
(counter value:1)
(brick name:A size:10 position:heap)
(brick name:B size:30 position:heap)
(brick name:C size:20 position:heap)
```

В този случай желаният резултат е: тухла B се намира на позиция 1, тухла C – на позиция 2 и тухла A – на позиция 3. Целта може да бъде постигната с използване на две подходящи продукционни правила.

Първото правило ще се грижи да постави най-голямата текущо налична тухла в ръката на робота, а второто ще постави тухлата, която в момента се намира в ръката на робота, на първата (по реда на номерата) свободна позиция:

```
1. IF (brick position:heap name:n size:s)
     ~(brick position:heap size:{> s})
     ~(brick position:hand)
   THEN MODIFY 1 (position hand)
2. IF (brick position:hand)
      (counter value:i)
   THEN MODIFY 1 (position i)
        MODIFY 2 (value [i+1])
```

В този пример на всяка стъпка от работния цикъл на интерпретатора е изпълнимо не повече от едно от двете правила – първото правило изисква ръката на робота да е празна, а второто изисква в нея да има тухла.

Проследяване на работата на интерпретатора на правилата при извод, управляван от данните:

1. Правило 2 не е изпълнимо, тъй като ръката на робота не държи тухла (няма тухла с позиция hand). Правило 1 е изпълнимо само за тухла B, тъй като тя е единствената, за която не съществува по-голяма тухла в купчината. При съпоставяне на Правило 1 с работната памет променливата n се свързва с B и s се свързва с 30. ЕРП, описващ тухлата B, се модифицира и получава следния вид:

```
(brick name:B size:30 position:hand)
```

2. Сега вече Правило 1 не е изпълнимо, но е изпълнимо Правило 2. В резултат на неговото изпълнение настъпват промени в ЕРП, описващи тухлата B и брояча (номера на първата свободна позиция) – те стават съответно

(brick name:B size:30 position:1) и

(counter value:2)

3. Тухла B вече не е в купчината, а заема позиция 1, затова сега Правило 1 е изпълнимо за тухла C, чиято позиция се променя:

(brick name:C size:20 position:hand)

4. Тази стъпка е подобна на стъпка 2. Правило 2 сега променя позицията на тухла C на 2 и увеличава брояча до 3:

(brick name:C size:20 position:2) и

(counter value:3)

5. Сега в купчината е останала само тухла A, така че Правило 1 е изпълнимо за нея и в резултат тя се премества в ръката на робота:

(brick name:A size:10 position:hand)

6. Правило 2 се изпълнява отново и този път тухла A се поставя на позиция 3:

(brick name:A size:10 position:3) и

(counter value:4)

7. Сега, тъй като няма тухли в купчината и в ръката на робота, не е изпълнимо никое от двете правила. Конфликтното множество е празно и работата на интерпретатора на правилата приключва. Работната памет има следното съдържание:

```
(counter value:4)
(brick name:A size:10 position:3)
(brick name:B size:30 position:1)
(brick name:C size:20 position:2)
```

Обща оценка на правилата като формализъм за ПИЗ

Специфични преимущества:

- естественост на представянето на експертни знания;
- модулност на базата от знания;
- възможност за обяснение на резултатите от извода.

Недостатъци и проблеми:

- проблеми при генериране на съдържателни обяснения на взетите решения (необходимост от използване на добре структурирана база от знания, която освен знанията за предметната област, представени чрез правила, включва и различни типове метазнания, подпомагащи генерирането на обяснения);
- недостатъчна изразителна сила;
- потенциална неефективност на работата на интерпретатора на правилата (решения: алгоритъм RETE, архитектури от тип „черна дъска“).

Експертни системи

Основни характеристики на експертните системи

Експертната система (ЕС) е компютърна програма, която съдържа знания и извършва логически извод относно специализирана предметна област с цел решаване на определени задачи или даване на подходящи съвети. По обхвата, обема, съдържанието, структурата и организацията си знанията на една ЕС са сравними с тези на хората – експерти в съответната предметна област.

Обикновено ЕС се използват за решаване на следните типове задачи:

- Интерпретация на данни (например звукови сигнали);
- Диагностика на повреди или заболявания;
- Структурен анализ на сложни обекти (например химични съединения);
- Конфигурация на сложни обекти (например компютърни системи);
- Планиране на последователности от действия.

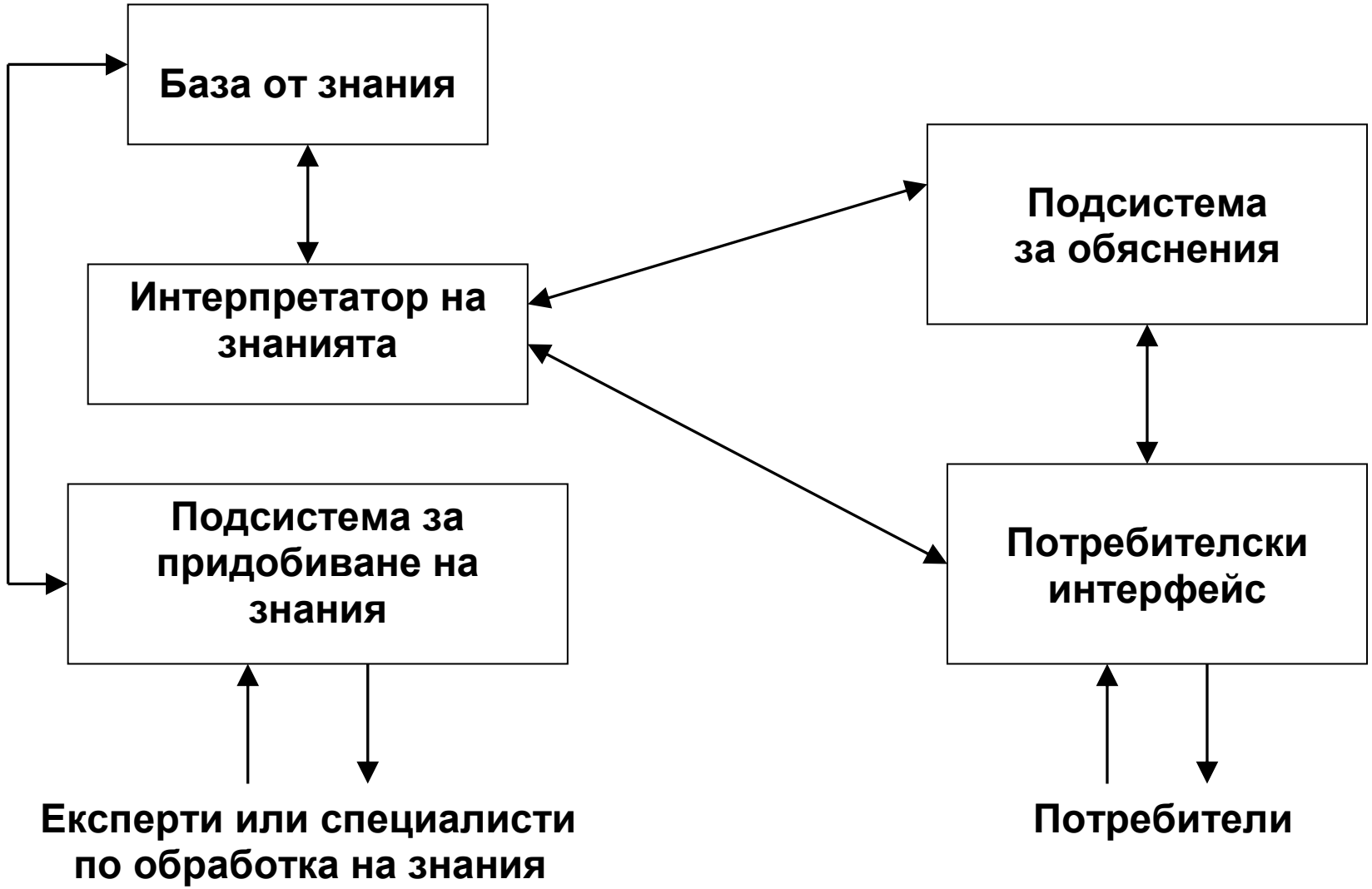
ЕС имат някои *особености*, които ги отличават от традиционните програмни системи:

- ЕС моделират начина, по който хората правят изводи в определена предметна област, а не самата област;
- ЕС извършват извод на базата на някакво представяне на човешки знания;
- ЕС обикновено решават задачи с помощта на евристични или приблизителни методи.

Евристиките са правила, извлечени от опита, които кодират определени знания за начина за решаване на даден проблем от определена област. *Евристичните методи* са приблизителни в смисъл, че не изискват точни данни и решенията могат да бъдат изведени от системата с определена степен на сигурност.

Архитектура на ЕС

ЕС съдържат традиционните за всяка СОЗ компоненти, към които са добавени още т. нар. *подсистема за обяснения* и *подсистема за придобиване на знания*.


База от знания
Интерпретатор на знанията
Подсистема за придобиване на знания
Подсистема за обяснения
Потребителски интерфейс
Експерти или специалисти по обработка на знания
Потребители

Инструментални средства за създаване на ЕС:

- редактори на знания и среди за придобиване на знания;
- празни ЕС (ядра на ЕС, ES shells) и среди за представяне на знания.