

Введение в ИИ на примере языка Prolog

Операции с внутренней базой данных и агрегирование решений

<https://github.com/Inscriptor/IntroductionToAI/tree/master/pdf>

Федеральное государственное автономное образовательное учреждение высшего образования
«Новосибирский национальный исследовательский государственный университет»

18 ноября 2019 г.

Операции с внутренней базой данных

Операции с внутренней базой данных

Стандарт языка Prolog предлагает операции для управления динамическими базами знаний, которые могут изменяться в процессе исполнения запросов.

Основные операции реализуют следующие предикаты:

- ▶ `assert/1`
- ▶ `asserta/1`
- ▶ `assertz/1`
- ▶ `retract/1`
- ▶ `retractall/1`
- ▶ `abolish/1`

Операции с внутренней базой данных

Assert

- ▶ Допустим, что мы имеем пустую базу знаний, не содержащую ни фактов, ни правил.
- ▶ В этом случае запрос `listing.` вернет ответ `true` и больше ничего.
- ▶ Выполнив запрос вида `assert(f(x,y)).` получим ответ `true`. **Предикат `assert` всегда завершается успешно.**
- ▶ Теперь запрос `listing(f/2).` вернет примерно следующее:

```
:- dynamic f/2.  
f(x, y).
```

Операции с внутренней базой данных

Assert

- ▶ Во-первых, предикат $f/2$ был объявлен динамическим.
- ▶ Во-вторых, теперь база знаний содержит факт $f(x,y)$., соответственно запрос $f(x,y)$. вернет `true`.
- ▶ Аналогично можно поступать и с правилами: `assert(f(X,y) :- (X > 5))` .
Здесь мы добавили правило, что $f(X,y)$ выполняется для любых $X > 5$.
Теперь запрос $f(10,y)$. вернет `true`, а запрос $f(1,y)$ — `false`.
- ▶ Факт или правило будут добавлены в базу знаний столько раз, сколько раз для них будет вызвана операция вставки.

Операции с внутренней базой данных

Assert

- ▶ Для более гибкого управления динамической базой знаний используются предикаты `asserta/1` и `assertz/1`.
- ▶ `asserta/1` добавит выражение в начало динамической базы.
- ▶ `assertz/1` добавит выражение в конец.
- ▶ Предикат `assert/1` на настоящий момент объявлен устаревшим (deprecated), и его использование не рекомендуется. В SWI Prolog предикат `assert` работает аналогично `assertz`.

Операции с внутренней базой данных

Retract

- ▶ Для удаления динамических фактов и правил используются предикаты `retract/1` и `retractall/1`.
- ▶ Предикат `retract/1` удалит первое правило или факт, с которыми сможет унифицировать переданный ему терм.
- ▶ В случае, если терм нельзя унифицировать с чем-либо в базе знаний, он вернет `false`.
- ▶ Предикат `retractall/1` удалит **все** вхождения правил или фактов, с которыми сможет унифицировать свой аргумент.
- ▶ Аргументами предикатов `retract` и `retractall` могут выступать как термы, так и правила.
- ▶ В случае, когда в качестве аргумента передан терм T будут удалены как факты вида $F.$, так и правила вида $F :- \dots$ такие, что $T = F$.

Операции с внутренней базой данных

Примеры

Добавим с динамическую базу несколько фактов и одно правило.

```
assertz(f(x,y)).
```

```
assertz(f(x,y)).
```

```
assertz(f(a,b)).
```

```
assertz((f(X,y) :- g(X,a,b)).
```


Операции с внутренней базой данных

Примеры

База будет выглядеть примерно так, если задать вопрос `listing(f/2)`.

```
:- dynamic f/2.
```

```
f(x,y).
```

```
f(x,y).
```

```
f(a,b).
```

```
f(X,y) :- g(X,a,b).
```

Операции с внутренней базой данных

Примеры

Теперь попробуем удалить одно вхождение факта $f(x,y)$.

```
retract(f(x,y)).
```

```
f(x,y).
```

```
f(a,b).
```

```
f(X,y) :- g(X,a,b).
```

Операции с внутренней базой данных

Примеры

Из оставшегося удалим все факты и правила, такие, что их head унифицируется с термом $f(X,y)$.

$\text{retractall}(f(X,y)).$

$f(a,b).$

Операции с внутренней базой данных

Примеры

Рассмотрим еще один пример — вычисление и кеширование таблицы умножения.

```
multab(Scale) :- member(X,Scale),  
                  member(Y,Scale),  
                  Prod is X * Y,  
                  assertz(mult(X,Y,Prod)),  
                  fail.
```

Операции с внутренней базой данных

Abolish

- ▶ Предикаты `assert(a/z)`, `retract` и `retractall` работают только с **динамическими фактами и правилами**, т.е. такими, у которых `head` объявлен динамическим.
- ▶ Если в базе нет упоминания о предикате `p`, то при добавлении утверждений с его участием через `assert` он будет объявлен динамическим.
- ▶ В противном случае предикат `p` должен быть объявлен динамическим вручную.
- ▶ Попытка работать со статическими предикатами при помощи данных операций приведет к ошибке.
- ▶ Однако, предикат `abolish/1` дает возможность удалять как статические, так и динамические выражения.
- ▶ В стандарте `abolish/1` описан, как предикат, обладающий таким же функционалом, как и `retractall/1`, однако в SWI Prolog его функционал был расширен, т.к. существование двух предикатов в одинаковом функционалом было признано нецелесообразным.

Агрегирование решений

Агрегирование решений

В случае, когда ответ на запрос предполагает несколько решений, мы можем получить их последовательно поиском с возвратом, либо сначала собрать все возможные решения и получить их в виде списка. Следующие три предиката реализуют операции агрегирования решений.

- ▶ `findall/3`
- ▶ `bagof/3`
- ▶ `setof/3`

Агрегирование решений

Findall

Запрос вида

```
findall(Buff, Goal, Answers).
```

возвращает список всех возможных значений переменной `Buff`, удовлетворяющих цели `Goal`. Чаще всего `Buff` — это просто переменная. В случае, когда не удалось найти требуемых значений для `Buff`, `findall` возвратит **пустой список**.

Агрегирование решений

Findall

Например, мы хотим получить все результаты умножения девятки из таблицы умножения.

```
findall([X,Y],mult(9,X,Y),Res).
```

Res =

```
[[1,9],[2,18],[3,27],[4,36],[5,45],[6,54],[7,63],[8,72],[9,81]].
```

Агрегирование решений

Findall

Параллельно, мы можем задать требуемую структуру элементам результирующего списка.

```
findall(nineProds(X,Y),mult(9,X,Y),Res).
```

```
Res = [nineProds(1,9), nineProds(2,18), nineProds(3,27),  
nineProds(4,36), nineProds(5,45), nineProds(6,54),  
nineProds(7,63), nineProds(8,72), nineProds(9,81)].
```

Агрегирование решений

Bagof

Предикат `bagof/3` позволяет получить результаты в более упорядоченном виде. Предикат `findall` возвращает простой мешок решений, в котором собрано все ответы, удовлетворяющие заданной цели. При этом, последовательность, в которой расположены эти ответы, зависит только от обхода дерева поиска решений.

Запрос вида

```
findall(X,mult(Z,X,_),Res).
```

возвратит что-то похожее на [1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Возможно, это именно то, что нужно, но, очевидно, не всегда такое нас устроит.

Агрегирование решений

Bagof

Рассмотрим запрос:

```
bagof(X,D^mult(Z,X,D),Res).
```

bagof выдаст решения, сгруппированные по Z.

```
Z = 1,
```

```
Res = [1, 2, 3, 4, 5, 6, 7, 8, 9] ;
```

```
Z = 2,
```

```
Res = [1, 2, 3, 4, 5, 6, 7, 8, 9] ;
```

```
Z = 3,
```

```
Res = [1, 2, 3, 4, 5, 6, 7, 8, 9] ;
```

```
...
```

Агрегирование решений

Setof

Еще один предикат для агрегирования решений — это `setof/3`. В целом, он работает почти также, как и `bagof`, за исключением того, что `setof` сортирует агрегированные списки (если может) и удаляет повторяющиеся элементы, т.е. возвращает *упорядоченные множества решений*.

Агрегирование решений

Setof

Например, запрос

```
setof(X,D^Z^mult(Z,X,D),Res).
```

возвратит ответ `Res = [1, 2, 3, 4, 5, 6, 7, 8, 9]`, в то время, как `bagof` вернул бы то же самое, что и `findall` — неупорядоченный список всех найденных ответов, включая повторы.

Выводы

- ▶ Динамическая база знаний позволяет кешировать найденные решения для повторного использования и управлять фактами и правилами в процессе выполнения запросов.
- ▶ При этом, не рекомендуется перегружать программу большим количеством динамических операций, т.к. это сильно затрудняет отладку и читаемость кода.
- ▶ Операции агрегации решений `findall`, `bagof` и `setof` в совокупности предлагают достаточно возможностей для управления возвращаемыми результатами.
- ▶ В большинстве случаев будет достаточно `findall`.