Въведение в езика Пролог

Обща характеристика на езика Пролог

Пролог е най-широко разпространеният език за логическо програмиране. Той е език за програмиране, основан на предикатното смятане от първи ред, т.е. при него описанието на предметната област става с помощта на изрази, които са еквивалентни на ППФ от предикатното смятане от първи ред, и алгоритъмът за логически извод е вариант на метода на резолюцията, който е метод за доказателство при предикатното смятане от първи ред.

Програмата на Пролог се състои от факти и правила. Фактите задават твърдения за съответната предметна област, които се смятат за безусловно верни, а правилата задават твърдения, които се смятат за верни, ако са налице определени условия.

Програмирането на Пролог се състои от следните етапи:

- задаване на ***факти*** за обектите и отношенията между тях в съответната предметна област;
- задаване на ***правила*** за обектите и отношенията между тях;
- формулиране на ***въпроси***, свързани с обектите и отношенията между тях.

Интерпретаторът на Пролог разглежда въпросите като ***цели***, които се опитва да удовлетвори (за които се опитва да докаже, че са логическо следствие от въведените факти и правила). Като метод за доказателство се използва (т.е. е вграден в интерпретатора) ***методът на резолюцията.***

По същество това е метод за ***доказателство чрез опровергаване на противното*** - към известните факти и правила се добавя отрицанието на целта и се прави опит за достигане до противоречие.

Забележки:

- фактите, правилата и въпросите в Пролог се наричат ***клаузи***, т.е. програмата на Пролог е поредица от клаузи;
- фактите и правилата образуват ***базата от данни*** (БД) на Пролог.

Следователно, в Пролог терминът "база от данни" се използва в специфичен смисъл, различен от общоприетия в информатиката.

История

Пролог е създаден около 1970 г. от А. Колмерое (A. Colmerauer) в Университета в Марсилия във връзка с решаването на задачи, свързани с анализ на текстове на естествен език. Той е създаден като език за програмиране, основан на предикатното смятане от първи ред. Като стандарт на Пролог се утвърждава т. нар. Единбургски Пролог (Единбургски синтаксис на Пролог). Първата среда за програмиране на Пролог е създадена в Университета в Единбург.

- Среда за програмиране на Пролог, която ще бъде използвана в курса:

  SWI-Prolog

  Свободно разпространявана среда за програмиране на Пролог, разработена в Университета на Амстердам. Съответства в максимална степен на стандарта на езика и поддържа формализмите DCG и CHR.

Факти, въпроси, променливи, съставни цели

1. Факти

Задават (описват) отношения между (твърдения за) обектите и явленията от предметната област, които се смятат от интерпретатора на Пролог за безусловно верни.

Запис на фактите в Пролог:

- имената на отношенията се записват чрез идентификатори, които започват с малка буква (в повечето достъпни реализации имената на отношенията и обектите не могат да съдържат букви от кирилицата);
- най-напред се записва името на отношението и след него в кръгли скоби се изброяват в точно определен ред, разделени със запетая, имената на съответните обекти;
- всеки факт завършва с точка.

Примери:

```
likes(john,ann).
owns(john,money).
valuable(gold).
play(john,jane,badminton).
cold.
```

Терминология:

- името на отношението се нарича ***предикат***;
- имената на обектите, участващи в съответното отношение (ако има такива), се наричат ***аргументи*** на отношението.

Следователно, предикатите започват с малка буква. Досега бяха разгледани примери за аргументи, които също започват с малка буква. В тях бяха посочени конкретни обекти - ***константи***, ***атоми***. Ако името на някой аргумент започва с главна буква, то се смята, че е зададен обобщен обект - ***променлива***.

Дефиниции:

- съвкупността от всички факти (и правила), които са зададени в определен момент, се нарича ***база от данни*** (БД) на Пролог;
- фактите, правилата и въпросите в Пролог се наричат общо ***клаузи*** (но в някои учебници клаузи се наричат само фактите и правилата).

2. Въпроси

След като в БД са въведени необходимите факти (и правила), потребителят може да задава въпроси за съответните отношения.

В най-простия случай въпросите се записват почти по същия начин, както и фактите, с тази разлика, че пред тях се поставя специалният символ "?-".

Нека е дадена следната БД:

```
likes(ann,flowers).
likes(ann,wine).
likes(ann,books).
likes(john,wine).
likes(john,ann).
likes(john,books).
```

Примерни въпроси и съответни отговори на интерпретатора на Пролог (в съответствие с Единбургския стандарт):

```
?- likes(john,ann).
yes
?- likes(john,money).
no
?- likes(ann,books).
yes
?- likes(ann,john).
no
?- king(john,france).
no
```

В средата на SWI-Prolog:

```
1 ?- likes(john,ann).
true ;
false.

2 ?- likes(john,money).
false.

3 ?- likes(ann,books).
true.

4 ?- likes(ann,john).
false.

5 ?- king(john,france).
ERROR: toplevel: Undefined procedure: king/2
```

Действие на интерпретатора на Пролог при опит за отговор на въпрос (опит за удовлетворяване на цел):

След задаване на въпрос интерпретаторът на Пролог инициира процедура на търсене в БД, като търси факти, съпоставими със зададения въпрос. Както видяхме, въпросите (в разгледания им досега вид) имат формата на факти, така че по същество е необходимо да се определи кога два факта са съпоставими.

Два факта са съпоставими, ако техните предикати са еднакви и съответните им аргументи са съпоставими (засега - съвпадат). Ако интерпретаторът намери в БД факт, който е съпоставим с въпроса, той отговаря с "да" (yes/true); ако в БД не съществува такъв факт, той отговаря с "не" (no/false).

Забележка. Отговорът "no" (“false”) не е еквивалентен на евентуалния отговор "лъжа е". Той означава, че няма елемент на БД (клауза от БД), който да е съпоставим със зададения въпрос. Следователно, отговор "no“ (“false”) означава, че ***целта е недоказуема от БД*** (не може да се докаже, че целта е логическо следствие от клаузите в БД).

В SWI-Prolog реакцията на интерпретатора при отговор на въпрос, в който участва недефиниран предикат, е различна от стандартната.

3. Променливи

Обикновено въпроси от типа на предходните са неинтересни и безполезни. Значително по-полезни са по-обобщените въпроси (въпросите от по-обобщен вид), при които целта е по-сложна от елементарното отговаряне на въпроса, дали даден конкретен факт се съдържа в БД. За задаване на такива обобщени въпроси (т.е. за задаване на по-общи цели) се използват ***променливи***.

Пример. Нека е дадена примерната БД от т. 2. Тогава можем да зададем различни въпроси на интерпретатора на Пролог, в това число и такива, еквивалентни на следните въпроси, формулирани на български език: "Какво харесва Джон?" или "Кои са тези неща, които Джон харесва?".

Такива въпроси се задават, като се използват променливи за означаване на някои от аргументите на предикатите. Променливите означават обобщените, в този случай неизвестните обекти, например:

```
?- likes(john,X).
```

Тук X е ***променлива, която е използвана за означаване на кой да е обект***.

Променливите се означават (записват) чрез ***идентификатори, в които първият знак е главна буква***.

Стойности на променливите в Пролог

Променливите в Пролог могат да имат или да нямат конкретни стойности. В първия случай се казва, че променливата е свързана (конкретизирана) - тогава е ясно кой точно обект означава тя. Във втория случай се казва, че променливата е свободна (несвързана, неконкретизирана) - тогава тя не означава конкретен обект (не е ясно или известно какво точно означава тя).

При търсенето на отговор на зададен въпрос, който съдържа променлива, интерпретаторът на Пролог претърсва фактите (клаузите) в БД с цел да намери обект, който тази променлива би могла да означава.

Примерен въпрос:

```
?- likes(john,X).
```

Интерпретаторът на Пролог ще претърси фактите от БД, за да намери такъв факт, който има предикат likes и първи аргумент john.

Ако намери такъв факт, интерпретаторът смята, че е намерил решение (отговор) на въпроса и конкретизира (свързва) променливата X с втория аргумент на намерения факт. След това интерпретаторът чака следващи указания от потребителя.

Работа на интерпретатора на Пролог при опит за отговор на въпрос, който съдържа променливи

Нека разгледаме въведената в т. 2 БД на Пролог:

```
likes(ann,flowers).
likes(ann,wine).
likes(ann,books).
likes(john,wine).
likes(john,ann).
likes(john,books).
```

и да проследим процеса на удовлетворяване на следната цел:

?- likes(john,X).

Отговорът на интерпретатора ще бъде: X=wine, след което интерпретаторът ще спре в очакване на следващи указания.

Обобщение. При постъпване на такъв въпрос към интерпретатора на Пролог променливата, участваща във въпроса, първоначално е свободна (неконкретизирана). Интерпретаторът започва търсене в БД, като целта му е да намери първия факт в БД, който е съпоставим със зададения въпрос. Ако в ролята на аргумент във въпроса участва свободна променлива, интерпретаторът смята, че такава променлива е съпоставима с всеки аргумент, който се намира на съответната позиция във факта.

В конкретния пример се търси първият факт в БД, който има предикат likes и първи аргумент john (по-точно, първи аргумент, който е съпоставим с константата john). Вторият аргумент в този случай може да бъде произволен, тъй като във въпроса вторият аргумент е свободна променлива. При намиране на такъв факт променливата X се свързва с втория аргумент на факта (тя вече означава обекта, посочен като втори аргумент във факта).

Интерпретаторът разглежда фактите в БД в реда, в който те са въведени. Следователно, най-напред ще бъде намерен фактът likes(john,wine). От този момент X ще се свърже с атома wine. С помощта на специален маркер интерпретаторът отбелязва мястото в БД, на което е извършено съпоставянето.

Роля на маркера:

След като намери факт, съпоставим с въпроса, интерпретаторът извежда стойностите (т.е. имената на обектите), с които се свързват в резултат на съпоставянето използваните променливи. В нашия пример има само една променлива X, която се свързва с wine.

След това интерпретаторът спира работа, очаквайки по-нататъшни указания.

Ако в тази ситуация се отговори чрез натискане на клавиша <Enter>, това означава, че потребителят е удовлетворен от получения отговор и не е необходимо интерпретаторът да продължава търсенето в БД. Ако потребителят отговори с ";", то това означава, че той желае да получи и по-нататъшни отговори на поставения въпрос. ***В този случай интерпретаторът продължава търсенето в БД, като започва от мястото, отбелязано с маркера (а не от началото на БД)***.

В такива ситуации се говори за ***преудовлетворяване*** (повторно удовлетворяване, намиране на алтернативно решение) на поставената цел (зададения въпрос).

При опит за преудовлетворяване на дадена цел търсенето в БД започва от мястото след последния поставен маркер при предходно удовлетворяване на същата цел (след маркера, поставен при последното предходно удовлетворяване на същата цел).

Ако при поредния опит за преудовлетворяване на дадена цел съпоставянето завърши с неуспех, то по стандарт интерпретаторът отговаря с "no" и очаква въвеждане на следваща цел, като изтрива всички маркери, поставени в процеса на удовлетворяване и преудовлетворяване на последната цел.

Примери върху посочената БД:

```
?- likes(john,X).
X = wine;
X = ann;
X = books;
no
?- likes(X,wine).
X = ann;
X = john;
no
```

```
1 ?- likes(john,X).
X = wine ;
X = ann ;
X = books.

2 ?-
```

4. Съставни цели (конюнкции от цели)

Нека отново е дадена примерната БД от т. 2:

```
likes(ann,flowers).
likes(ann,wine).
likes(ann,books).
likes(john,wine).
likes(john,ann).
likes(john,books).
```

Освен въпроси от разглеждания досега тип в Пролог могат да се задават и по-сложни въпроси, които съдържат конюнкции от "елементарни“ въпроси. С други думи, могат да се поставят съставни цели, които са конюнкции от "прости" цели. Конюнкциите от въпроси (цели) се задават чрез изреждане, разделени със запетаи, на отделните "елементарни“ въпроси, от които те се състоят. Една съставна цел се смята за удовлетворена, когато се удовлетворят (последователно, от ляво на дясно) всички цели, от които е съставена тя.

Примери:

```
?- likes(john,ann), likes(john,wine).
yes
?- likes(john,ann), likes(ann,john).
no
```

```
3 ?- likes(john,ann), likes(john,wine).
true ;
false.

4 ?- likes(john,ann), likes(ann,john).
false.

5 ?-
```

При комбиниране на възможностите, които дават конюнкциите и променливите, могат да се формулират доста сложни въпроси.

Пример:

```
?- likes(ann,X), likes(john,X).
```

Този въпрос може да бъде разтълкуван така:

- съществува ли нещо, което се харесва едновременно от Ан и Джон?;
- кои са тези неща, които се харесват едновременно от Ан и Джон?

По същество този въпрос се състои от две цели:

- съществува ли обект X, който се харесва от Ан?;
- харесва ли Джон намерената стойност на X?

Работа на интерпретатора на Пролог при удовлетворяване на съставни цели, които съдържат променливи

Интерпретаторът на Пролог започва да отговаря на сложния (съставния) въпрос, като се опитва да удовлетвори (да намери съпоставяне за) първата, т.е. най-лявата цел. Ако в БД бъде намерен факт, съпоставим с тази цел, то интерпретаторът маркира намереното място и се опитва да удовлетвори втората цел.

Ако и тя бъде удовлетворена, то интерпретаторът отново маркира в БД съответното място и в крайна сметка се оказва, че е намерено решение, удовлетворяващо и двете цели.

Важно е да се помни, че всяка цел има свой собствен маркер за означаване на мястото в БД, на което е станало удовлетворяването на тази цел.

Ако се случи така, че след удовлетворяването на първата цел не може да бъде намерено решение (съпоставяне) за втората цел, то интерпретаторът се опитва да намери ново решение за първата цел (да преудовлетвори първата или, по-общо, предходната цел). В такъв случай търсенето на ново решение на предходната цел започва не от началото на БД, а от съответния на тази цел маркер в БД.

Проследяване на работата на интерпретатора на Пролог върху пример от разглеждания тип

Въпрос:

```
?- likes(ann,X), likes(john,X).
```

Работа на интерпретатора:

1) Започва претърсване на БД (от нейното начало) с цел намиране на факт, съпоставим с първата цел. Първият такъв факт е likes(ann,flowers). Променливата X ***се свързва*** с flowers и от този момент всяко срещане на тази променлива в съставната цел е свързано с посочената стойност.

Маркира се мястото в БД, където е бил намерен съпоставимият факт, за да може при необходимост търсенето на алтернативно решение за първата цел да започне от този маркер. Освен това, интерпретаторът "запомня", че променливата X е била конкретизирана (свързана) на това място и в случай на необходимост от преудовлетворяване на първата цел той ще може да "забрави" стойността на X (т.е. ще може да направи X отново свободна променлива или, с други думи, ще може да ***освободи*** променливата X).

2) Започва търсене в БД (от нейното начало) за факт, съпоставим с втората цел - likes(john,flowers), конкретизирана след извършеното в т. 1 свързване на променливата X. В БД няма такъв факт и, следователно, е необходимо да се намери ново решение за предходната цел. Затова интерпретаторът се опитва да преудовлетвори целта likes(ann,X). За целта започва търсене в БД от маркера на първата цел, като преди това променливата X се освобождава.

3) Търсенето в БД този път започва от факта, непосредствено следващ likes(ann,flowers). Следващият факт, съпоставим с първата цел, е likes(ann,wine). Сега променливата X се свързва с wine и интерпретаторът отново маркира достигнатото в БД място.

4) Интерпретаторът отново се опитва да удовлетвори втората цел, която този път е конкретизирана като likes(john,wine). По същество това е нова за интерпретатора цел и той започва да търси от началото на БД факт, съпоставим с тази цел. Такъв факт в случая съществува, следователно цялата съставна цел може да се смята за удовлетворена.

Интерпретаторът маркира мястото, на което е намерено решение за втората цел (за да може при евентуален опит за преудовлетворяване на тази цел да започне търсенето в БД от мястото, отбелязано със съответния маркер), извежда съобщение за намереното решение на съставната цел и чака следващи указания от потребителя.

5) Сега и двете цели (a, следователно, и съставната цел) са удовлетворени. Променливата X е свързана с атома wine. Маркерът на първата цел отбелязва в БД факта likes(ann,wine), а маркерът на втората цел - факта likes(john,wine). Ако потребителят сега въведе ";", т.е. поиска намиране на алтернативно решение на съставната цел, започва търсене на нови решения, което започва съответно от поставените маркери на целите в БД.

Обобщение

При обработка на конюнкция от цели интерпретаторът на Пролог се опитва да съпостави с клаузи от БД всички цели от конюнкцията, като преглежда конюнкцията от ляво на дясно. Ако се намери факт, съпоставим с текущо разглежданата цел, то интерпретаторът поставя на това място в БД маркер, свързан с разглежданата цел. Възможно е при това някои свободни до момента променливи да се свържат с конкретни стойности.

Ако дадена променлива се свърже (конкретизира), то се конкретизират всички появявания на тази променлива в съставната цел. След това интерпретаторът на Пролог се опитва да удовлетвори десния съсед на разглежданата досега цел, като започва да търси от началото на БД факт, съпоставим с новата цел. Ако тази цел бъде удовлетворена, извършват се същите действия, както при удовлетворяването на първата цел, и се преминава към следващата цел.

Всеки път, когато текущата цел пропадне (т.е. не се намери съпоставим с нея факт от БД), интерпретаторът се връща назад и се опитва да преудовлетвори левия съсед на тази цел, като започва търсене в БД от мястото, отбелязано от съответния маркер. При това се освобождават всички променливи, които са били свързани при удовлетворяването на тази цел (левия съсед на пропадналата цел).

Ако не се намери ново решение за левия съсед, се преминава още една стъпка на ляво и т.н. Ако не се намери ново решение и на най-лявата цел от конюнкцията, това означава, че цялата съставна цел е неудовлетворима (съставната цел пропада).

Ако на дадена стъпка се намери алтернативно решение на текущо достигнатата (като ляв съсед) цел, то започва обратно (на дясно) удовлетворяване на следващите цели. В такъв случай търсенето в БД за тези цели отново започва от началото на БД.

Забележки

- Полезно в този контекст е понятието ***област на действие на дадена променлива***. В Пролог областта на действие на една променлива съвпада със съответната (съставна) цел, в която е включена тази променлива, т.е. с въпроса, в който е използвана тя, започвайки от "?-" и завършвайки с ".“

- Този механизъм, при който при невъзможност за удовлетворяване на някаква цел се осъществява връщане назад и търсене на алтернативни решения на вече удовлетворени цели, за да може после отново да се тръгне напред, се нарича ***backtracking*** (търсене с възврат; механизъм на възврат; ***възврат***; връщане назад; обратно проследяване и др.)

```
5 ?- likes(ann,X), likes(john,X).
X = wine ;
X = books.

6 ?-
```

5. Правила. Съпоставяне при използване на правила в Пролог

Обща идея: правилата се използват за означаване на твърдения, които са верни при определени условия (когато са верни други твърдения).

Примери

1) Джон обича всеки, който обича вино.

Запис като ППФ от предикатното смятане от първи ред:

(∀X) (likes(X,wine) -> likes(john,X))

Запис на Пролог:

```
likes(john,X):- likes(X,wine).
```

2) Джон обича всички хора.

Запис като ППФ от предикатното смятане от първи ред:

(∀X) (human(X) –> likes(john,X))

Запис на Пролог:

```
likes(john,X):- human(X).
```

Синтаксис (запис) на правилата в Пролог

Всяко правило се състои от ***глава*** и ***тяло***. Главата и тялото се съединяват (свързват, обединяват) чрез символа ":–" (neck), който обикновено се чете "ако“ (“когато”). Главата на правилото е факт. Тялото на правилото в общия случай е конюнкция (или дизюнкция) от факти. Всяко правило завършва с точка.

Обикновено в правилата на Пролог се използват (често много на брой) променливи. И тук, както и при съставните цели, трябва да се има предвид, че ако в даден момент някоя променлива, участваща в правилото, се конкретизира (свърже), то се свързват всички включвания на тази променлива в правилото. С други думи, ***областта на действие на една променлива, участваща в дадено правило, е цялото правило***, което започва с главата и завършва с точката.

Действие на интерпретатора на Пролог при използване на правила

Нека като пример разгледаме БД, която се основава на твърдението:

"Даден човек X може да открадне определен предмет Y, ако човекът X е крадец и харесва предмета Y.".

Примерна БД:

```
/*1*/ thief(john).
/*2*/ likes(ann,food).
/*3*/ likes(ann,wine).
/*4*/ likes(john,X):- likes(X,wine).
/*5*/ may_steal(X,Y):- thief(X),
                       likes(X,Y).
```

Примерен въпрос:

```
?- may_steal(john,X).
```

Действие на интерпретатора:

1) Интерпретаторът търси в БД клауза, която описва предиката may_steal. Такава е клаузата с пореден номер 5, която е правило. Интерпретаторът се опитва да съпостави намереното правило с въпроса. Ако това съпоставяне завърши успешно, то целта е удовлетворена. Тогава интерпретаторът маркира използваното правило в БД, извежда съответно съобщение и чака следващи указания.

Кога една цел е съпоставима с дадено правило? За целта е необходимо:

- главата на правилото да е съпоставима с целта (въпроса);
- тялото на правилото да е съпоставимо с БД (със съдържанието на БД).

Следователно, при опит за съпоставяне на дадена цел с дадено правило се пораждат ***нови цели*** (***подцели***), които трябва да бъдат удовлетворени, за да се смята за удовлетворена основната цел. В конкретния случай главата на правилото е съпоставима с целта и затова се преминава към опит за съпоставяне на тялото на правилото с БД. При това, при съпоставянето на главата на правилото с целта променливата X от правилото се свързва с john, а променливата Y от правилото се ***съвместява*** с променливата X от целта, като и двете в момента са свободни.

Забележки:

- две променливи са съвместени, когато те или и двете са свободни, или са свързани с един и същ обект. Винаги, когато едната от тях се свързва с някаква константа, другата автоматично се свързва със същата константа;

- в действителност интерпретаторът на Пролог не работи с имената на променливите във вида, в който те са зададени от потребителя. За удобство при опит за удовлетворяване на дадена цел той дава нови, системни имена на всички използвани променливи. На съвместените променливи той дава еднакви системни имена.

В конкретно решаваната задача при опита за съпоставяне на тялото на правилото с БД най-напред се прави опит за удовлетворяване на най-лявата цел - thief(john).

2) Тази цел е удовлетворима, тъй като тя съвпада с факта – клауза с номер 1 от БД. Интерпретаторът маркира това място в БД. Ново свързване на променливи не се извършва. След това се преминава към удовлетворяване на целта likes(john,Y).

3) Целта likes(john,Y) се съпоставя с главата на правилото - клауза с номер 4 от БД. Следователно, прави се опит за удовлетворяване (съпоставяне с БД) на тялото на това правило - likes(X,wine). При това променливата Y от целта се съвместява с променливата X от тази подцел (в резултат на съпоставянето на целта likes(john,Y) с главата на правилото). Следователно, текущата цел в момента е likes(X,wine).

4) Тази цел се удовлетворява, тъй като тя е съпоставима с факта - клауза с номер 3 от БД. При това променливата X се свързва с ann.

5) Тъй като подцелта, породена от правилото - клауза с номер 4, е удовлетворена, то цялото правило е съпоставимо с целта likes(john,Y). При това Y се свързва с ann, тъй като Y е съвместена с X, която в т. 4 се свързва с ann. Следователно, целта likes(john,Y) се удовлетворява и Y се свързва с ann.

6) При това положение основната цел е съпоставена с правилото - клауза с номер 5. При това променливата X от тази цел, която беше съвместена с Y, също като Y се свързва с ann. Следователно, основната цел успява и отговорът е: X = ann, след което интерпретаторът спира и очаква следващи указания.

```
1 ?- may_steal(john,X).
X = ann ;
false.

2 ?-
```