Logic Programming

Basic concepts

Joey W. Coleman, Stefan Hallerstede

AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

8 April 2014

Facts

- A fact has the following shape: `p(t1,...,tn).`
  Example:

  ```
  parent(tom,peter).
  ```

  This fact states that
  - `tom` is the parent of `peter`
  - the relation `parent` holds between the individuals `tom` and `peter`
- From `p(t1,...,tn).` we can deduce `p(t1,...,tn)`
- Another name for relation is ***predicate***
- Names of individuals are known as ***atoms***
- Atoms and numbers are called ***constants***
- The number of arguments of a predicate is called its ***arity***
  Example: the arity of predicate `parent` is `2`
- We refer to a predicate p with arity n by $\mathrm{p/n}$
  Example: `parent/2`

Functors

- A ***functor*** has one or more arguments: `f(t1,...,tn)`.
  Example:

  `s(0)`

- The name `f` of a functor is an atom
- The number of arguments of a functor is called its ***arity***
- We can use functors in arguments of predicates
  Example:

  `parent(s(0),s(s(0))).`

Operators

- Any atom may be designated as an ***operator***
  Example:

  `3+4`

  where + is declared as an infix operator
- An operator can be written in functor notation
  Example: `3+4` is the same as `+(3,4)`
- Some common operators:

Operator	Class	Priority	Used for
`:-`	xfx	1200	Separating head and body of a clause
`,`	xfy	1000	Separating goals in a clause
`is`	xfx	700	Arithmetic evaluation

- Lower priorities bind stronger
- The class is used to encode position and associativity
  - The “f” represents the operator “y” and “x” the subterms
  - “y” is used to indicate associativity:
    Example: , associates to the right: `p,q,r` equals `p,(q,r)`

Queries

- A query `?- p(t1,...,tn).` asks whether a relations holds between objects
  Example:

  ```
  ?- parent(tom,mary).
  ```

  Given the fact `parent(tom,peter).` the answer is ***no***
- We call the predicate `p(t1,...,tn)` of a query a ***goal***

- A query starts a computation
- At the moment we can only do very primitive computations
- In the next sections we introduce the concepts that are missing in order to arrive at ***programs*** that can perform more complicated computations

Facts and queries

Logical variables

More on facts and queries

Rules

Recursion

Logical variables

- A ***logical variable*** stands for an unspecified individual
- Variables are valuable in queries:

  To find out who is child of `tom` we could ask a series of queries

  ```
  ?- parent(tom,mary).
  ?- parent(tom,john).
  ?- parent(tom,tim). ...
  ```

  A better way is to ask

  ```
  ?- parent(tom,X).
  ```

  to which the answer is `X=peter`
- Used in this way variables are a means to summarise many queries
- Convention:
  Variables begin with an upper-case letter or an underscore "_"

Terms

- A term is the only data structure in logic programs
- We define terms inductively:
    - Constants and variables are terms
    - Structures are terms.
      A structure comprises
        a functor and
        a sequence of one or more arguments, which are terms
- Structures are also called compound terms
- Example of a structure:

```
tree(tree(nil,3,nil),5,R).
```

Substitution

- A ***substitution*** is a (possibly empty) finite set of pairs of the form `Xi=ti`, where
  - `Xi` is a variable and `ti` is a term with `Xi`≠`ti`, and
  - `Xi`≠`Xj` for every `i`≠`j`.
  - (It is called ***solved*** if `Xi` does not occur in `tj` for any `i` and `j`.)
- $A\theta$ denotes the result of applying substitution θ to term A
- $A\theta$ is obtained by
  replacing every occurrence of `X` by `t` in A
  for every pair `X=t` in θ
- Substitutions are applied to predicates by applying them to the contained terms: `p(t1,...,tn)`θ is `p(t1`θ`,...,tn`θ`)`
- Example:
  Applying `{X=peter}` to the predicate `parent(tom,X)` yields the predicate `parent(tom,peter)`
- A is an ***instance*** of B if there is a substitution θ such that $A=B\theta$
  Example: `parent(tom,peter)` is an instance of `parent(tom,X)`

Universal facts

- Variables are also useful in facts:

  Instead of stating that `tom` likes each individual
  `likes(tom,mary).`
  `likes(tom,john).`
  `likes(tom,tim). ...`

  we can state the fact
  `likes(tom,X).`
  saying that `tom` likes everyone
- Variables are means of summarising many facts
- A fact `p(t1,...,tn).` reads that for all `X1,...,Xk`, where the `Xi` are the variables occurring free in the fact, `p(t1,...,tn)` is true
- From a universal fact one can deduce any instance of it
  Example:
  from `likes(tom,X).` we can deduce `likes(tom,mary).`

Existential queries

- Variables in queries are existentially quantified
- A query `?- p(t1,...,tn).` reads that are there `X1,...,Xk`, where the `Xi` are the variables occurring free in the query, such that `p(t1,...,tn)` is true
- Example:
  `?- parent(tom,X).` reads:
  Does there exist an `X` such that `tom` is the parent of `X` ?
- From an instance `p(t1,...,tn)`θ we can deduce the existential query `?- p(t1,...,tn).`

Repeated variables

- Variables can occur in several places in the same fact or query
- Because they can only be instantiated once
  this means that the terms in this locations must be the same
- Example:
  The fact `equals(X,X)` states that everything equals itself
- Example:
  The query `?- add(X,X,4)` asks for a number `X` that added to itself yields `4`

Operational interpretation

- C is a ***common instance*** of A and B if it is an instance of A and an instance of B
- Formally: C is a ***common instance*** of A and B if there are substitutions σ and θ such that $C{=}A\sigma$ and $C{=}B\theta$
- Operation interpretation of query:
  - To answer a query using a fact, ***search*** for a common instance of the query and the fact
  - The answer is ***yes*** and the substitution of the query if there is a common instance
  - Otherwise the answer is ***no***
- Remark: answering an existential query with a universal fact using a common instance requires two deductions

Conjunctive queries and shared variables

- A ***conjunctive query*** has the form `?- Q1,...,Qn` where the `Qi` are the goals of the query
- Example:
  `?- parent(tom,X),parent(X,michael)` asks whether `michael` is a grandson of `tom`
- The "," is logical conjunction
- The scope of a variables in a query is the entire query; they are called ***shared variables***
- A query `?- p(X),q(X)` reads
  Is there an `X` such that both `p(X)` and `q(X)` are true?
- Example:
  The query `?- parent(tom,X),parent(X,Y)` has two effects:
  - It restricts the children of `tom` to those who are themselves parent
  - It restricts the children `Y` to those whose parents are children of `tom`
- To solve a conjunctive query `?- Q1,...,Qn` find a substitution θ such that the goals $Qi\theta$ are common instances with facts `Pi`

Rules

- The query `?- parent(tom,X),parent(X,Y)` asks for the grandchildren of `tom`
- We can define this new relationship by means of a rule:
  `grandchild_of_tom(X) :- parent(tom,X),parent(X,Y)`
- In general, ***rules*** have the shape *A* `:-` *B1*`,...,`*Bn*`.`
- *A* is called the ***head*** of the rule
- *B1*`,...,`*Bn* is called the ***body*** of the rule
- The *Bi* are called ***goals***
- Rules, facts and queries are also called ***Horn clauses***, or ***clauses*** for short
- A fact is just a special case of a rule with n=0
- A ***logic program*** is a finite set of rules

Stock keeping

Now we have facts, queries and rules:

Facts:	`A.`
Queries:	`?- B1,...,Bn.`
Rules:	`A :- B1,...,Bn.`

- Facts are rules with an empty body
- Queries are rules without a head
- Rules encapsulate queries (similar to procedures)

Rule deduction

- From the rule
  A :- $B1, \ldots, Bn$.
  and the facts
  $D1$.
  ...
  Dn.
  the fact C can be deduced if
  C :- $D1, \ldots, Dn$
  is an instance of A :- $B1, \ldots, Bn$

Previous forms of deduction:

- From a fact A. we can deduce A
- From a fact A. we can deduce any instance $A\theta$. of it
- From an instance of a goal $A\theta$ we can deduce the goal A

Logical consequence

A goal `G` is a ***logical consequence*** of a program *P* if

- there is a clause in *P* with an instance `A :- B1,...,Bn.` such that
  - `B1,...,Bn` are logical consequences of *P* and
  - `A` is an instance of `G`.

This is a first approximation of what a logic program computes.
In practice, it does it quite differently, however!

Facts and queries

Logical variables

More on facts and queries

Rules

Recursion

Recursion

- We can define a predicate `grand_parent` by

```
grand_parent(X,Z) :- parent(X,Y), parent(Y,Z).
```

- The more general notion of `ancestor` requires recursion:

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Z) :- parent(X,Y), ancestor(Y,Z).
```

or

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Z) :- ancestor(X,Y), parent(Y,Z).
```

or

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Z) :- ancestor(X,Y), ancestor(Y,Z).
```

- The first version is special: it is ***tail-recursive***