Logic Programming Systems

CPS 270

Ron Parr

Automating Reasoning

- Want a sound and complete procedure
- Need to represent information in our database in a canonical form
- Need to understand the factors that influence the efficiency of our reasoning system

Proofs by Resolution

- Convert to canonical form
- Assert negation of the proof target
- Resolve until “nil” is obtained
- Why can’t we bound the number of resolution steps we need to take?
- Example on board…

Speeding Up Resolution

- There are many heuristics for speeding up resolution – we can view it as a special kind of search
- As with propositional logic, we can also consider special cases
- AI has a colorful history of special case logics and special case reasoning engines for handling these logics

Implementation Issues

- Any reasoning system must be able to identify relevant sentences in its KB rapidly
- Maintain multiple indices:
  - A list of positive literals for each predicate symbol
  - A list of negative literals for each predicate symbol
  - A list of sentences with this predicate as conclusion
  - A list of sentences with this predicate as premise
- More sophisticated, tree-based indexing schemes are possible

Unification

- We need to avoid circular unifications
- Consider Unify(P(x,f(x)),P(y,y))=???
- What happens:
  - Bind x to y
  - Bind f(x) to y
  - This implies x is bound f(x)
  - This is circular
- Checking called an “occurs check”
- $O(n^2)$ to check this (many systems don’t)

Prolog

- Prolog is a grand effort to make logic a practical programming method
- Prolog is a *declarative* language
  - State the things that are true
  - Ask the system to prove things
  - All computations are essentially proofs
- Prolog makes many restrictions on KB
- My bias: Prolog is a fascinating way to think about logic and programming, but is of *waning* importance in AI

Prolog Properties

- KB is sequences of sentences (all implicitly conjoined)
- All sentences must be horn
- Can use constants, variables, or functions
- Queries can include conjunctions or disjunctions
- Cannot assert negations
  - Closed world assumption
  - Everything not implied by the KB is assumed false

Prolog Properties

- All syntactically distinct terms refer to distinct objects
  - Two variables can be =
  - Two objects cannot be =
- Built in predicates for arithmetic
- Build in list handling as part of the unification process

Prolog Implementation

- Inferences are done with backward chaining
- Is this complete?
- What is the computational complexity?
- Conjuncts are tried in left to right order (as entered in the KB)
- Tries implications in order they are entered
- No occurs check (in most Prologs)

Prolog UI

- Load a database using consult
- Consult(user) loads database from the command line ctrl-d to terminal
- Consult(file) loads database from a file.
- Some prologs use [file].

Prolog Syntax

- Variables are upper case
- Constants are lower case
- Implication :-
- Universal quantification is implicit
- Sentences are terminated with a .
- Specify RHS first: Mortal(X):-Man(X)
- Conjunction with ,: Mortal(X):-Man(X),Living(X).

Prolog Syntax

- Lists [Head|Tail]
  - Head is bound to first element of list
  - Tail is bound to remainder of list
  - Append
- Numbers
  - Numbers are assigned with “is”
  - Checked with =, =<, =>

Prolog Bindings

- Use = to check if two bindings are same
- Use \== to check if they are different
- Hit enter at the end of query to stop search
- Use ; to get multiple answers