A Prolog Primer (TDT4136)

Lars Bungum (Ph.D. stip.)
Department of Computer & Information Science
Outline
Programming in Logic

— Declarative
Programming in Logic

— Declarative
— Backtracking
Programming in Logic

— Declarative
— Backtracking
— Unification
Programming in Logic

— Declarative
— Backtracking
— Unification
— Good for logic and reasoning
Declarative

— Specify *what*, not *how* (procedures)
Declarative

— Specify *what*, not *how* (procedures)
— Programs consist of rules and facts
Declarative

— Specify *what*, not *how* (procedures)
— Programs consist of rules and facts
— Prolog interpreter will find goal (from query)
Declarative

— Specify *what*, not *how* (procedures)
— Programs consist of rules and facts
— Prolog interpreter will find goal (from query)
— Prolog also has procedural interpretations
Backtracking

— If one rule cannot find the goal, Prolog tries again
Backtracking

— If one rule can not find the goal, Prolog tries again
— Can find more goals that satisfy conditions
Unification

— Way of finding ways to satisfy the goal
Unification

— Way of finding ways to satisfy the goal
— Making two terms identical
Unification

— Way of finding ways to satisfy the goal
— Making two terms identical
— Unification is a general concept also used elsewhere
Good for logic and artificial intelligence

— Natural Language Processing (NLP)
Good for logic and artificial intelligence

— Natural Language Processing (NLP)
— Expert systems (decision making)
Good for logic and artificial intelligence

— Natural Language Processing (NLP)
— Expert systems (decision making)
— Watson (!)
A prolog program

— Data types
— Clauses (facts and rules)
— Database (storage)
Data objects (terms)

- Numbers

- Atoms (beginning with a lowercase letter a–z; a–zA–Z)

- Variables (stand for a term, not bound)

- Compound terms (record structure, functors with arities)

- Lists [1,2,3,4] First—Rest=[1,2,3,4] First=1 Rest=[2,3,4]
Data objects (terms)

- Numbers
- Atoms (beginning with a lowercase letter \([a–z][a–zA–Z]+\))
Data objects (terms)

- Numbers
- Atoms (beginning with a lowercase letter $[a-z][a-zA-Z]^{+}$)
- Variables (stand for a term, NB are not bound)
Data objects (terms)

— Numbers
— Atoms (beginning with a lowercase letter \([a–z][a–zA–Z]+\)\)
— Variables (stand for a term, NB are not \textit{bound})
— Compound terms (record structure, functors with arities)
Data objects (terms)

- Numbers
- Atoms (beginning with a lowercase letter \([a-z][a-zA-Z.]+\) )
- Variables (stand for a term, NB are not *bound*)
- Compound terms (record structure, functors with arities)
- Lists \([1,2,3,4]\) \([\text{First—Rest}]=[1,2,3,4]\) \text{First}=1 \text{Rest}=[2,3,4]
Clauses

— Facts and rules
Clauses

— Facts and rules
— Fact: dog(fido).
Clauses

— Facts and rules
— Fact: dog(fido).
— Rule: animal(X):- dog(X).
Clauses

— Facts and rules
— Fact: dog(fido).
— Rule: animal(X):- dog(X).
— Predicates (res: True/False) vs. functions (res: anything)
Clauses

— Facts and rules
— Fact: dog(fido).
— Rule: animal(X):- dog(X).
— Predicates (res: True/False) vs. functions (res: anything)
Prolog flavors

— SICStus (on IDI machines)
Prolog flavors

- SICStus (on IDI machines)
- SWI-Prolog (easily downloadable)
Production systems

— Rule-based forward chaining systems
Production systems

— Rule-based forward chaining systems
— Extended to programming languages
Production systems

— Rule-based forward chaining systems
— Extended to programming languages
— Deals with rules and facts
Proxy

— PROlog implementation of produXion system
Proxy

— **PROlog** implementation of produXion sYstem
Bibliography