472 Recitation

Week 8

Propositional Logic

- Syntax and Semantics
- Reasoning
  - Resolution
  - Forward/backward chaining
  - DPLL and local search

Resolution

Problem: $KB \models \alpha$?

Idea: proof by contradiction , $KB \models \alpha$ switch to show $KB \wedge \neg \alpha$ unsatisfiable

$Literal \rightarrow Symbol \mid \neg Symbol$

Clause: a disjunction of literals.

Conjunctive normal form (CNF): a conjunction of clauses

$Clause \rightarrow Literal_1 \vee \cdots \vee Literal_m$

$CNFSentence \rightarrow Clause_1 \wedge \cdots \wedge Clause_n$

- Convert $KB \wedge \neg \alpha$ to CNF
- Apply resolution rules
- Empty clauses or no new clause

$$\frac{l_1 \vee \cdots \vee l_i \vee \cdots \vee l_k, \qquad m}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k}$$

(l_i and m are complementary literals, i.e.,$l_i = \neg m$ or $m = \neg l_i$.)

Resolution

```
function PL-RESOLUTION(KB, α) returns true or false
  inputs: KB, the knowledge base, a sentence in propositional logic
          α, the query, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
  new ← { }
  while true do
      for each pair of clauses C_i, C_j in clauses do
          resolvents ← PL-RESOLVE(C_i, C_j)
          if resolvents contains the empty clause then return true
          new ← new ∪ resolvents
      if new ⊆ clauses then return false   // no new clauses can be added.
      clauses ← clauses ∪ new
```

Forward Chaining

Question $KB \models q?$

single proposition symbol

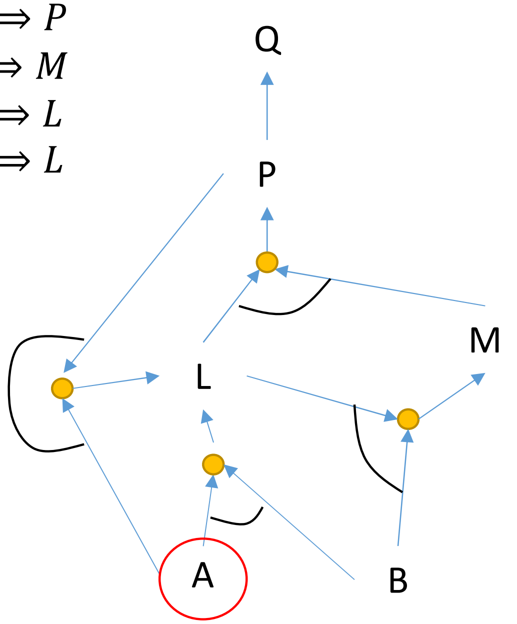
- Begins from positive literals (facts).
- If all the premises of an implications are known, then add its conclusion to KB (as a new fact).
- Continues until q is added or no further inferences can be made.

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional definite clauses
          q, the query, a proposition symbol
  count ← a table, where count[c] is initially the number of symbols in clause c's premise
  inferred ← a table, where inferred[s] is initially false for all symbols
  queue ← a queue of symbols, initially symbols known to be true in KB

  while queue is not empty do
     p ← POP(queue)
     if p = q then return true
     if inferred[p] = false then
        inferred[p] ← true
        for each clause c in KB where p is in c.PREMISE do
           decrement count[c]
           if count[c] = 0 then add c.CONCLUSION to queue
  return false
```

Forward Chaining

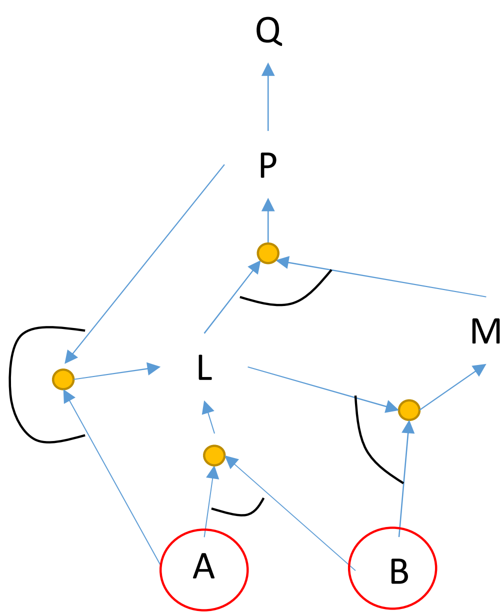
$P \Rightarrow Q$
$L \wedge M \Longrightarrow P$
$B \wedge L \Longrightarrow M$
$A \wedge P \Longrightarrow L$
$A \wedge B \Longrightarrow L$
A
B



Queue	Inferred	Count
A	A false	$P \Rightarrow Q$ 1
B	B false	$L \wedge M \Longrightarrow P$ 2
	L false	$B \wedge L \Longrightarrow M$ 2
	M false	$A \wedge P \Longrightarrow L$ 2
	P false	$A \wedge B \Longrightarrow L$ 2
	Q false	

Queue	Inferred	Count
B	A true	$P \Rightarrow Q$ 1
	B false	$L \wedge M \Longrightarrow P$ 2
	L false	$B \wedge L \Longrightarrow M$ 2
	M false	$A \wedge P \Longrightarrow L$ 1
	P false	$A \wedge B \Longrightarrow L$ 1
	Q false	

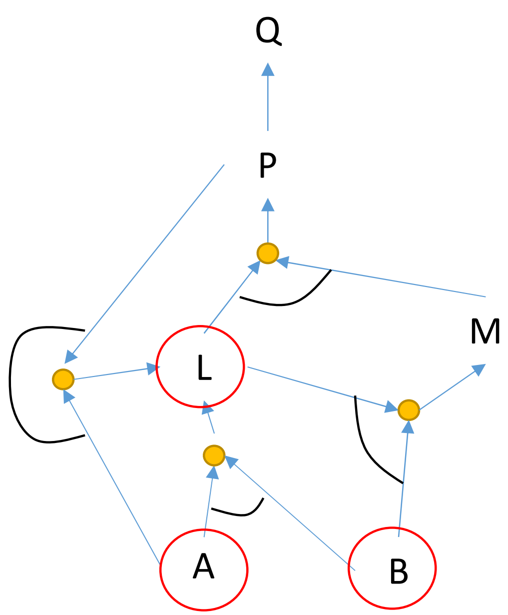
Forward Chaining



Queue	Inferred	Count	
B	A true	$P \Rightarrow Q$	1
	B false	$L \wedge M \Longrightarrow P$	2
	L false	$B \wedge L \Longrightarrow M$	2
	M false	$A \wedge P \Longrightarrow L$	1
	P false	$A \wedge B \Longrightarrow L$	1
	Q false		

Queue	Inferred	Count	
L	A true	$P \Rightarrow Q$	1
	B true	$L \wedge M \Longrightarrow P$	2
	L false	$B \wedge L \Longrightarrow M$	1
	M false	$A \wedge P \Longrightarrow L$	1
	P false	$A \wedge B \Longrightarrow L$	0
	Q false		

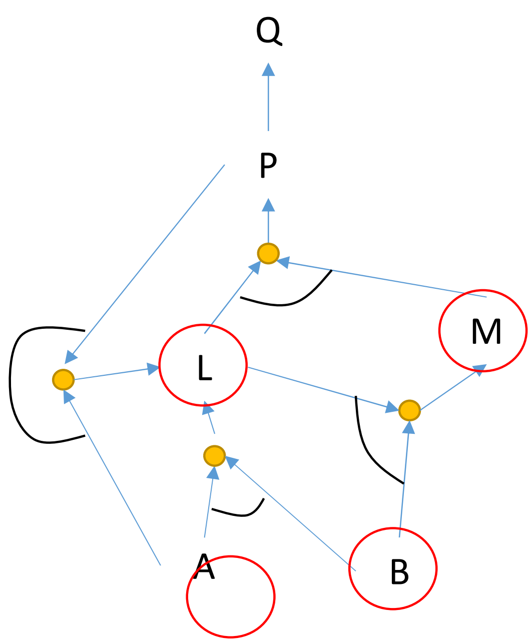
Forward Chaining



Queue	Inferred	Count
L	A true B true L false M false P false Q false	$P \Rightarrow Q$ 1 $L \wedge M \Longrightarrow P$ 2 $B \wedge L \Longrightarrow M$ 1 $A \wedge P \Longrightarrow L$ 1 $A \wedge B \Longrightarrow L$ 0

Queue	Inferred	Count
M	A true B true L true M false P false Q false	$P \Rightarrow Q$ 1 $L \wedge M \Longrightarrow P$ 1 $B \wedge L \Longrightarrow M$ 0 $A \wedge P \Longrightarrow L$ 1 $A \wedge B \Longrightarrow L$ 0

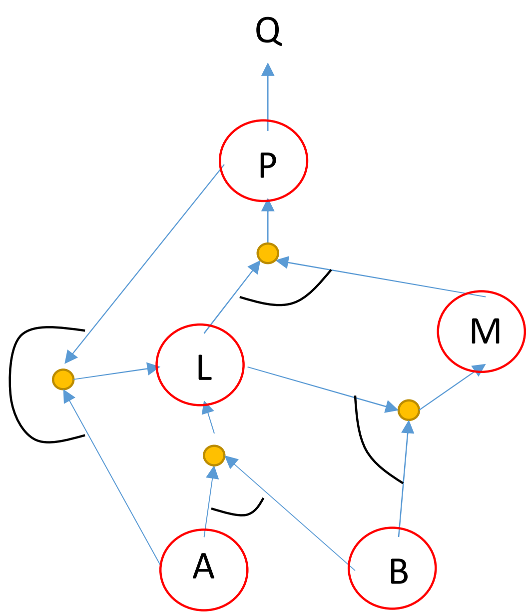
Forward Chaining



Queue	Inferred	Count
M	A true B true L true M false P false Q false	$P \Rightarrow Q$ 1 $L \wedge M \Rightarrow P$ 1 $B \wedge L \Rightarrow M$ 0 $A \wedge P \Rightarrow L$ 1 $A \wedge B \Rightarrow L$ 0

Queue	Inferred	Count
P	A true B true L true M true P false Q false	$P \Rightarrow Q$ 1 $L \wedge M \Rightarrow P$ 0 $B \wedge L \Rightarrow M$ 0 $A \wedge P \Rightarrow L$ 1 $A \wedge B \Rightarrow L$ 0

Forward Chaining



Queue	Inferred	Count
P	A true B true L true M true P false Q false	$P \Rightarrow Q$ 1 $L \wedge M \implies P$ 0 $B \wedge L \implies M$ 0 $A \wedge P \implies L$ 1 $A \wedge B \implies L$ 0

Queue	Inferred	Count
Q	A true B true L true M true P true Q false	$P \Rightarrow Q$ 0 $L \wedge M \implies P$ 0 $B \wedge L \implies M$ 0 $A \wedge P \implies L$ 1 $A \wedge B \implies L$ 0

Backward Chaining

- If q is true, no work is needed.
- Otherwise, finds implications in the KB whose conclusion is q.
- If all the premises of one of these implications can be proved true (recursively by backward chaining), then q is true.

- FC is data-driven, BC is goal-driven
- BC is more time efficient than FC

DPLL
- Based on BC
- Terminates the BC early under some cases
- Simplifies the BC process by making use of some features of the clauses

First order logic

- Propositional logic lacks the expressive power to describe an environment with many objects.
- Propositional logic assumes the world contains facts only.

FOL is Much more flexible than propositional logic

- Constants: ISU,..
- Variables: x, y...
- Functions: sum, times...
- Predicates: Brother, teacher...
- parenthesis: teach(x, y)…
- Connectives: $\land$, $\lor$, $\Rightarrow$, $\Leftrightarrow$, $\neg$
- Equality : =
- Quantifiers: $\forall$, $\exists$
  - $\forall$ with $\Rightarrow$
  - $\exists$ with $\land$

$$\neg\forall x\, P(x) \quad \equiv \quad \exists x\, \neg P(x)$$

$$\neg\exists x\, P(x) \quad \equiv \quad \forall x\, \neg P(x)$$

First Order Logic

Politicians can fool some of the people all the time, and they can fool all the people some of the time, but they can't fool all the people all the time.

$$\forall x\ Politician(x) \Rightarrow (\exists y\ Person(y) \wedge \forall t\ Fool(x, y, t))$$
$$\wedge ((\forall y\ Person(y) \Rightarrow \exists t\ Fool(x, y, t))$$
$$\wedge \neg(\forall y\ Person(y) \Rightarrow \forall t\ Fool(x, y, t))$$

- Politician(x): x is a politician
- Fool (x, y, t): x fools y in time t