Logical Agents

Chapter 7 continued
COSC 370
Spring 2013
Alan C. Jamieson

Some slide content from Russell & Norvig provided slides

- More Inference
- Equivalence, Validity, Satisfiability
- Forward and Backward Chaining
- Resolution

Recall: Wumpus World Sentences

- Let $P_{i,j}$ mean that there is a pit at square i,j
- Let $B_{i,j}$ mean that there is a breeze at square i,j
- Our KB:
  R_1: $\neg P_{1,1}$
  R_4: $\neg B_{1,1}$
  R_5: $B_{2,1}$
- "Pits cause breezes in adjacent squares"
  R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
  R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$
- "A square is breezy iff there is an adjacent pit"

How to leverage?

- Enumeration!

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Algorithm

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

   symbols ← a list of the proposition symbols in KB and α
   return TT-CHECK-ALL(KB, α, symbols, [ ])

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
   if EMPTY?(symbols) then
      if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
      else return true
   else do
      P ← FIRST(symbols); rest ← REST(symbols)
      return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
                TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

- In general, depth first enumeration. Complete, but time-intensive – $O(2^n)$ for n symbols.

Logical Equivalence

- $\alpha \equiv \beta$ iff $\alpha \vDash \beta$ and $\beta \vDash \alpha$

$$(\alpha \land \beta) \equiv (\beta \land \alpha) \quad \text{commutativity of } \land$$
$$(\alpha \lor \beta) \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor$$
$$((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land$$
$$((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor$$
$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \lor \beta) \quad \text{implication elimination}$$
$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$
$$\neg(\alpha \land \beta) \equiv (\neg\alpha \lor \neg\beta) \quad \text{De Morgan}$$
$$\neg(\alpha \lor \beta) \equiv (\neg\alpha \land \neg\beta) \quad \text{De Morgan}$$
$$(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor$$
$$(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land$$

Validity, Satisfiability, and Proofs

- A sentence is valid if it is true in all models.
- A sentence is satisfiable if it is true in some models
- Proof methods:
  - Application of inference – legitimate generation of new sentences from old, proof via inference rule application, typically requires translation into a normal form.
  - Model checking – truth table enumeration, allows for improved backtracking and heuristic search

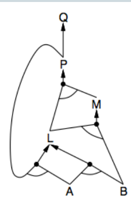
Forward and Backward Chaining

- First a normal form – Horn Form
  KB = conjunction of Horn clauses
  Horn clause – proposition symbol OR
  conjunction of symbols ⇒ symbol
- Example: KB = C ∧ (B ⇒ A) ∧ (C ∧ D ⇒ B)
- Can be used by forward and backward chaining in linear time.
- Chaining – way of reasoning while leveraging a KB. Utilizes modus ponens "P implies Q. P is true, thus Q is true".

Forward Chaining

- Idea – start from the premise, then add things to the KB as we infer from the Horn clauses.

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



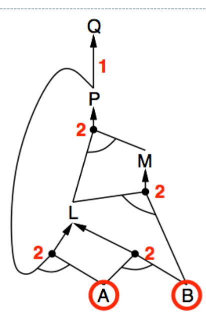
FC Algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
   inputs: KB, the knowledge base, a set of propositional Horn clauses
           q, the query, a proposition symbol
   local variables: count, a table, indexed by clause, initially the number of premises
                    inferred, a table, indexed by symbol, each entry initially false
                    agenda, a list of symbols, initially the symbols known in KB

   while agenda is not empty do
       p ← POP(agenda)
       unless inferred[p] do
           inferred[p] ← true
           for each Horn clause c in whose premise p appears do
               decrement count[c]
               if count[c] = 0 then do
                   if HEAD[c] = q then return true
                   PUSH(HEAD[c], agenda)
   return false
```

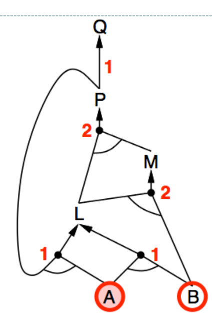
Run-Through

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



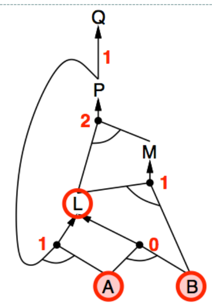
Run-Through

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



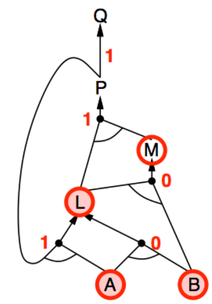
Run-Through

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



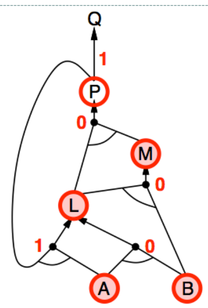
Run-Through

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



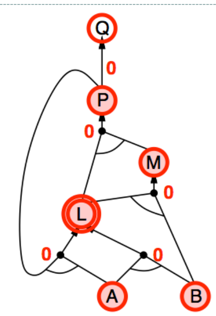
Run-Through

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



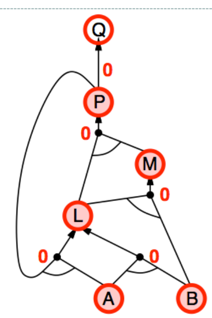
Run-Through

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



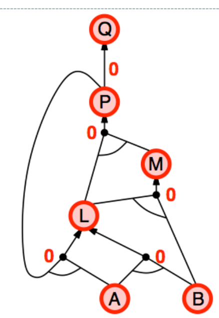
Run-Through

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



Run-Through

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

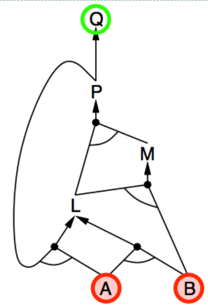


Backward Chaining

- Work back from the query q:
  - Check to see if q is already known
  - prove by BC all premises of rules that imply q.
- Avoidance of loops.
- Avoidance of repeated work.

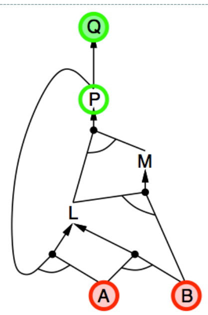
Run-Through

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



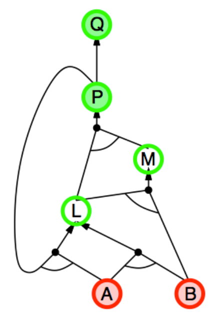
Run-Through

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



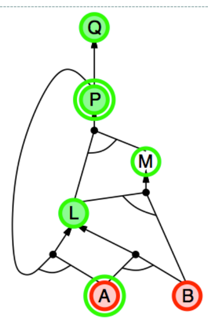
Run-Through

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



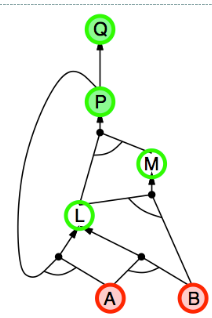
Run-Through

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



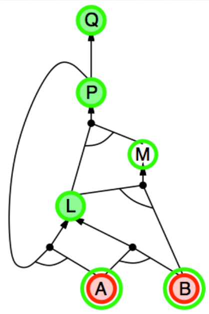
Run-Through

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



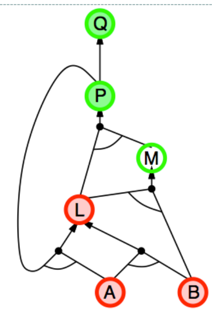
Run-Through

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



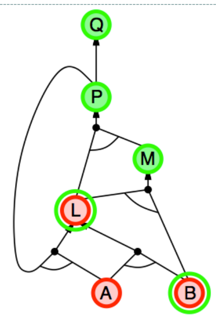
Run-Through

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



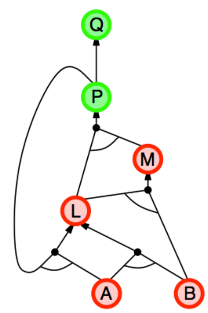
Run-Through

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



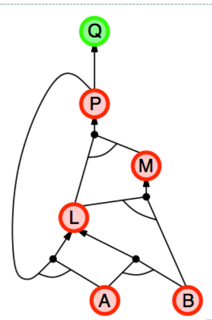
Run-Through

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



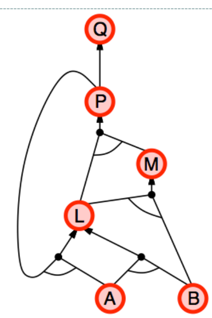
Run-Through

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



Run-Through

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



FC vs. BC

- FC is data-driven, automatic processing
  - May do lots of work that's irrelevant!
- BC is goal-driven, appropriate for problem-solving (a bit more complex)
  - However, complexity is still linear!

CNF and Resolution

- Conjunctive Normal Form (CNF) – conjunction of disjunction of literals (clauses)
- Example: (A ∨ ¬B) ∧ (B ∨ ¬C ∨ ¬D)
- Resolution – like FC and BC, a way to query a KB, figure out if a particular value can be inferred.

Conversion to CNF

1.) Eliminate ⇔, by replacing A⇔B with A⇒B ∧ B⇒A.

2.) Eliminate ⇒, by replacing A⇒B with ¬A ∨ B.

3.) We move our negations to be only attached to literals (not clauses):

¬(¬A) ≡ A (double-negation elimination)

¬(A ∧ B) ≡ ¬A ∨ ¬B (De Morgan's Law)

¬(A ∨ B) ≡ ¬A ∧ ¬B (De Morgan's Law)

4.) Apply distributivity law to distribute ∨ over ∧:

(A ∨ (B ∧ C)) ≡ (A ∨ B) ∧ (A ∨ C)

Resolution

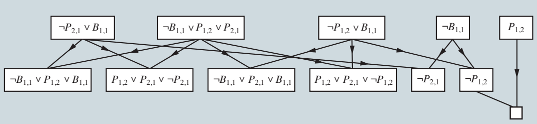
- Proof by contradiction!

```
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 ← { }
  loop do
     for each Ci, Cj in clauses do
        resolvents ← PL-RESOLVE(Ci, Cj)
        if resolvents contains the empty clause then return true
        new ← new ∪ resolvents
     if new ⊆ clauses then return false
     clauses ← clauses ∪ new
```

Resolution Example

- KB = $(B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$



Exercise

- CNF conversion practice + Resolution – 7.18 b&c