The Foundations: Logic and Proofs

1.2 Applications of Propositional Logic

With Question/Answer Animations

Chapter Summary

- Propositional Logic
  - The Language of Propositions
  - Applications
  - Logical Equivalences
- Predicate Logic
  - The Language of Quantifiers
  - Logical Equivalences
  - Nested Quantifiers
- Proofs
  - Rules of Inference
  - Proof Methods
  - Proof Strategy

Propositional Logic Summary

- The Language of Propositions
  - Connectives
  - Truth Values
  - Truth Tables
- Applications
  - Translating English Sentences
  - System Specifications
  - Logic Puzzles
  - Logic Circuits
- Logical Equivalences
  - Important Equivalences
  - Showing Equivalence
  - Satisfiability

Applications of Propositional Logic: Summary

- Translating English to Propositional Logic
- System Specifications
- Boolean Searching
- Logic Puzzles
- Logic Circuits
- AI Diagnosis Method (Optional)

Translating English Sentences

- Steps to convert an English sentence to a statement in propositional logic
  - Identify atomic propositions and represent using propositional variables.
  - Determine appropriate logical connectives
- "If I go to Harry's or to the country, I will not go shopping."
  - p: I go to Harry's
  - q: I go to the country.
  - r: I will go shopping.

If p or q then not r.

$$(p \vee q) \rightarrow \neg r$$

Example

Problem: Translate the following sentence into propositional logic:

“You can access the Internet from campus only if you are a computer science major or you are not a 1st-yr student.”

One Solution: Let

a = “You can access the internet from campus”

c = “You are a computer science major”

f = “You are a 1st-yr student”

$$a \rightarrow (c \vee \neg f)$$

System Specifications

- Engineers often take requirements in English and express them in logic.

Example: Express in propositional logic:

“The automated reply cannot be sent when the file system is full”

Solution: Let

p = “The automated reply can be sent”

q = “The file system is full.”

$$q \rightarrow \neg p$$

Consistent System Specifications

Definition: A list of propositions is *consistent* if it is possible to assign truth values to the proposition variables so that each proposition is true.

Exercise: a)Are these specifications consistent?

- "The diagnostic message is stored in the buffer or it is retransmitted."
- "The diagnostic message is not stored in the buffer."
- "If the diagnostic message is stored in the buffer, then it is retransmitted."

Solution: Let p = "The diagnostic message is stored in the buffer"

q = "The diagnostic message is retransmitted"

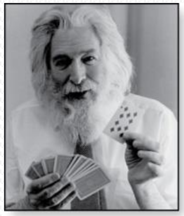
The specifications can be written as: $p \vee q$, $\neg p$, $p \rightarrow q$.

If p = F and q = T all three statements are true. So consistent.

b) What if "The diagnostic message is not retransmitted" is added?

Solution: Now we are adding $\neg q$ and there is no satisfying assignment. So the specification is not consistent.

Logic Puzzles



Raymond
Smullyan
(Born 1919)

- An island has two kinds of inhabitants, *knights*, who always tell the truth, and *knaves*, who always lie.
- You go to the island and meet A and B.
  - A says "B is a knight."
  - B says "The two of us are of opposite types."

Example: What are the types of A and B?

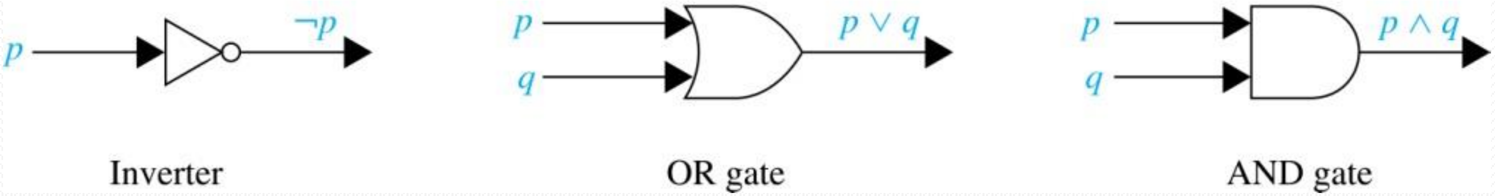
Solution: Let p = A is a knight -> $\neg p$ = A is a knave

q = B is a knight -> $\neg q$ = B is a knave

- If p is true -> q is true (since knights always tell the truth.
- -> $(p \wedge \neg q) \vee (\neg p \wedge q)$ is true, but it is not. So p is false so A is a knave.
- If A is a knave, -> B is knave as well, since knaves always lie.
- *B's* statement is now a lie but this is consistent with identifying B as a knave.

Logic Circuits (See Chapter 12 for more)

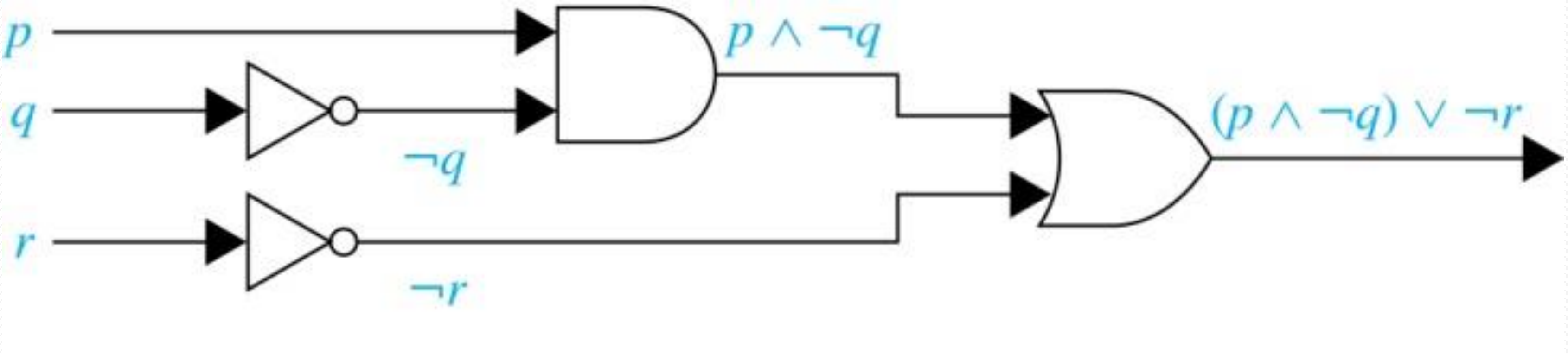
- Electronic circuits; each input/output signal can be viewed as a 0 or 1.
  - 0 represents **False**
  - 1 represents **True**
- Complicated circuits are constructed from three basic circuits called gates.



  - The inverter (**NOT gate**)takes an input bit and produces the negation of that bit.
  - The **OR gate** takes two input bits and produces the value equivalent to the disjunction of the two bits.
  - The **AND gate** takes two input bits and produces the value equivalent to the conjunction of the two bits.

Logic Circuits (continued)

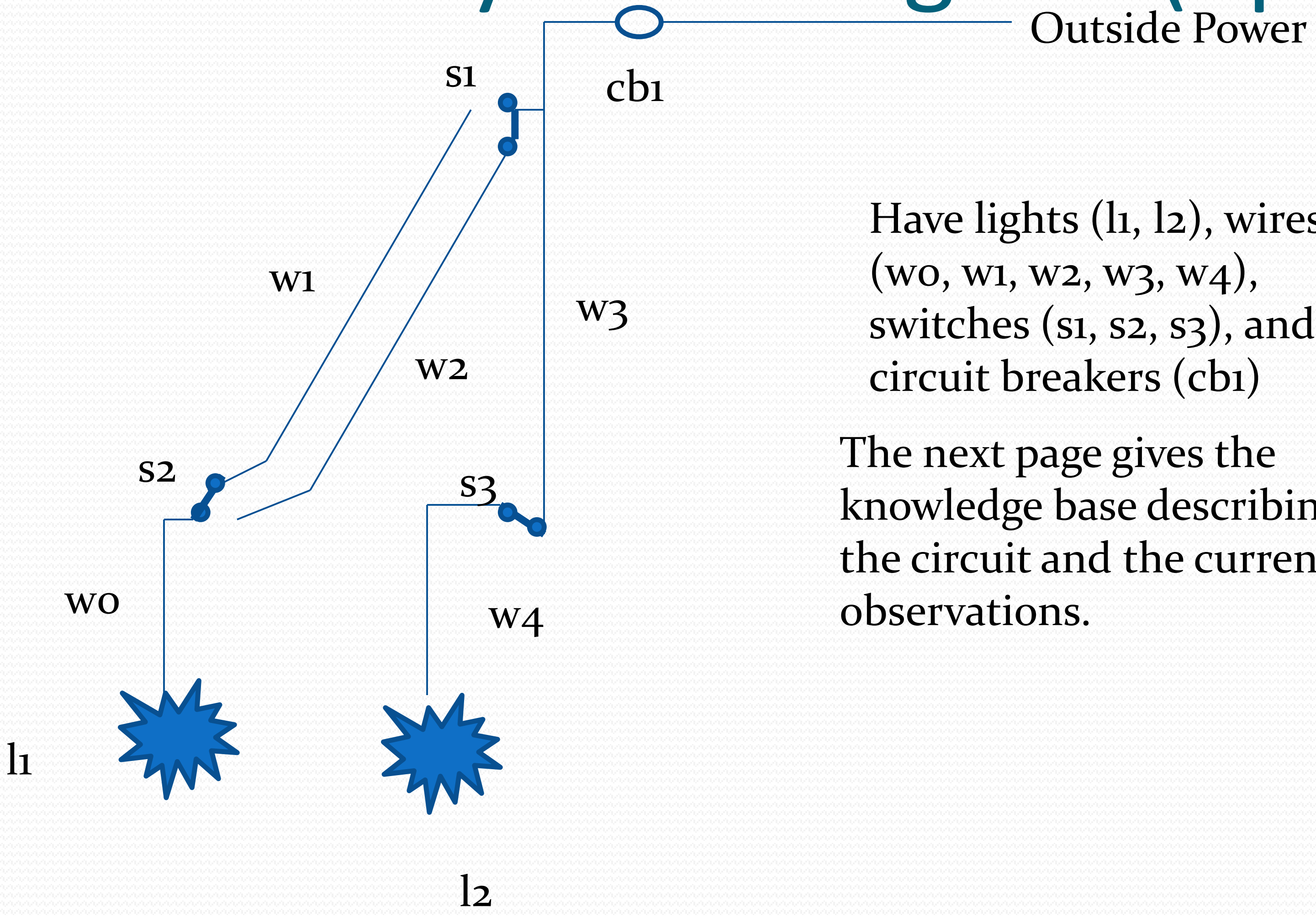
- More complicated digital circuits can be constructed by combining the basic circuits to produce the desired output given the input signals by building a circuit for each piece of the output expression and then combining them, e.g.,



Diagnosis of Faults in an Electrical System (*Optional*)

- AI Example (from *Artificial Intelligence: Foundations of Computational Agents* by David Poole and Alan Mackworth, 2010)
- Need to represent in propositional logic the features of a piece of machinery or circuitry that are required for the operation to produce observable features. This is called the **Knowledge Base (KB)**.
- We also have observations representing the features that the system is exhibiting now.

Electrical System Diagram (optional)



Have lights (l1, l2), wires (w0, w1, w2, w3, w4), switches (s1, s2, s3), and circuit breakers (cb1)

The next page gives the knowledge base describing the circuit and the current observations.

Representing the Electrical System in Propositional Logic

- We need to represent our common-sense understanding of how the electrical system works in propositional logic.
- For example: “If l1 is a light and if l1 is receiving current, then l1 is lit.
  - light_l1 ∧ live_l1 ∧ ok_l1 → lit_l1
- Also: “If w1 has current, and switch s2 is in the up position, and s2 is not broken, then w0 has current.”
  - live_w1 ∧ up_s2 ∧ ok_s2 → live_w0
- This task of representing a piece of our common-sense world in logic is a common one in logic-based AI.

Knowledge Base (*opt*)

- live_outside
- light_l1
- light_l2
- live_w0 → live_l1
- live_w1 ∧ up_s2 ∧ ok_s2 → live_w0
- live_w2 ∧ down_s2 ∧ ok_s2 → live_w0
- live_w3 ∧ up_s1 ∧ ok_s1 → live_w1
- live_w3 ∧ down_s1 ∧ ok_s1 → live_w2
- live_w4 → live_l2
- live_w3 ∧ up_s3 ∧ ok_s3 → live_w4
- live_outside ∧ ok_cb1 → live_w3
- light_l1 ∧ live_l1 ∧ ok_l1 → lit_l1
- light_l2 ∧ live_l2 ∧ ok_l2 → lit_l2

We have outside power.

Both l1 and l2 are lights.

If s2 is ok and s2 is in a down position and w2 has current, then w0 has current.

Observations (*opt*)

- Observations need to be added to the KB
  - Both Switches up
    - up_s1
    - up_s2
  - Both lights are dark
    - ¬lit_l1
    - ¬ lit_l2

Diagnosis (*opt*)

- We assume that the components are working ok, unless we are forced to assume otherwise. These atoms are called *assumables*.
- The assumables (ok_cb1, ok_s1, ok_s2, ok_s3, ok_l1, ok_l2) represent the assumption that we assume that the switches, lights, and circuit breakers are ok.
- If the system is working correctly (all assumables are true), the observations and the knowledge base are consistent (i.e., satisfiable).
- The augmented knowledge base is clearly not consistent if the assumables are all true. The switches are both up, but the lights are not lit. Some of the assumables must then be false. This is the basis for the method to diagnose possible faults in the system.
- A diagnosis is a minimal set of assumables which must be false to explain the observations of the system.

Diagnostic Results (*opt*)

- See *Artificial Intelligence: Foundations of Computational Agents* (by David Poole and Alan Mackworth, 2010) for details on this problem and how the method of consistency based diagnosis can determine possible diagnoses for the electrical system.
- The approach yields 7 possible faults in the system. At least one of these must hold:
  - Circuit Breaker 1 is not ok.
  - Both Switch 1 and Switch 2 are not ok.
  - Both Switch 1 and Light 2 are not ok.
  - Both Switch 2 and Switch 3 are not ok.
  - Both Switch 2 and Light 2 are not ok.
  - Both Light 1 and Switch 3 are not ok.
  - Both Light 1 and Light 2 are not ok.