Modeling Using Propositional Logic

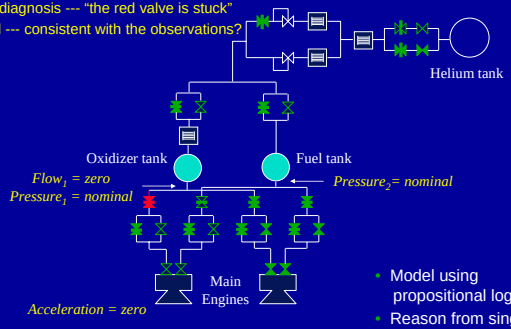
Brian C. Williams
16.410-13
Session 15

Reading Assignment: Propositional Logic & Satisfiability

- AIMA Ch. 6 – Propositional Logic

How Do We Reason About Complex Systems at a Commonsense Level?

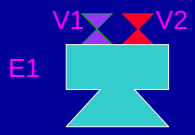
Is the diagnosis --- "the red valve is stuck" closed --- consistent with the observations?



- Model using propositional logic.
- Reason from single model to operate, diagnose and repair.

Propositional Sentences: Engine Example

An Engine E1 can either be okay or broken, in some unknown way. When E1 is okay, it will thrust when there is a flow through V1 and v2.



(mode(E1) = ok or mode(E1) = unknown) and
not (mode(E1) = ok and mode(E1) = unknown) and
(mode(E1) = ok implies
(thrust(E1) = on if and only if flow(V1) = on and flow(V2) = on))

Propositional Satisfiability

Find a truth assignment that satisfies logical sentence T:

- Reduce sentence T to clausal form.
- Perform search similar to MAC = (BT+CP) [Davis, Logmann & Loveland, 1962]

Propositional satisfiability testing:
1990: 100 variables / 200 clauses (constraints)
1998: 10,000 - 100,000 vars / 10^6 clauses

Novel applications:
e.g. diagnosis, planning, software / circuit testing, machine learning, and protein folding

Outline

- Propositional Logic
  - Syntax
  - Semantics
  - Clausal Reduction
- Appendices

What formal languages exist for describing constraints?

Logic:

- Propositional logic — truth of facts
- First order logic — facts,objects,relations
- Temporal logic — time, ….
- Modal logics — knowledge, belief …
- Probability — degree of belief
- Algebra — values of variables

Logic in General

- Logics
  - formal languages for representing information such that conclusions can be drawn.
- Syntax
  - defines the sentences in the language.
- Semantics
  - defines the "meaning" of sentences;
  - ⇨ truth of a sentence in a world.

Logic Example: Arithmetic

- Syntax – legal sentences
  - "X + 2 > Y" is a legal sentence.
  - "X 2 + Y >" is not a legal sentence.
- Semantics - truth in world
  - "X + 2 > Y" is true iff the number x + 2 is not less than or equal to the number y
  - "X + 2 > Y" is true in a world where X = 7, Y = 1
  - "X + 2 > Y" is false in a world where X = 0, Y = 6

Propositional Logic: Syntax

Propositions

- A statement that is true or false
  - (valve v1)
  - (= voltage high)

Propositional Sentences (S)

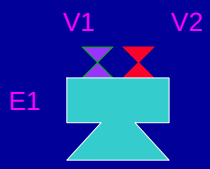
- S ::= proposition |
- (NOT S) |
- (OR S1 ... Sn) |
- (AND S1 ... Sn)

Some Defined Constructs

- (implies S1 S2) => ((not S1) OR S2)
- (IFF S1 S2) => (AND (IMPLIES S1 S2)(IMPLIES S2 S1))

Propositional Sentences: Engine Example

(mode(E1) = ok or mode(E1) = unknown) and
not (mode(E1) = ok and mode(E1) = unknown) and
(mode(E1) = ok implies
(thrust(E1) = on if and only if flow(V1) = on and flow(V2) = on))



Outline

- Propositional Logic
  - Syntax
  - Semantics
  - Clausal Reduction
- Appendices

Propositional Logic: Semantics

Interpretation I of sentence S assigns true or false to every proposition P of S

- S = (A or B) and C
- I = {A=True, B=False, C=True}
- I = {A=False, B=True, C=False}

All Interpretations →

A	B	C
True	True	True
True	True	False
True	False	True
True	False	False
False	True	True
False	True	False
False	False	True
False	False	False

Propositional Logic: Semantics

The truth of sentence S wrt interpretation I is defined by a composition of boolean operators applied to I:

- "Not S" is True iff "S" is False

Not S	S
False	True
True	False

Propositional Logic: Semantics

The truth of sentence S_i wrt Interpretation I:

- "Not S" is True iff "S" is False
- "S_1 and S_2" is True iff "S_1" is True **and** "S_2" is True
- "S_1 or S_2" is True iff "S_1" is True **or** "S_2" is True

S1 and S2	S1	S2
True	True	True
False	True	False
False	False	True
False	False	False

S1 or S2	S1	S2
True	True	True
True	True	False
True	False	True
False	False	False

Propositional Logic: Semantics

The truth of sentence S_i wrt Interpretation I:

- "Not S" is True iff "S" is False
- "S_1 and S_2" is True iff "S_1" is True **and** "S_2" is True
- "S_1 or S_2" is True iff "S_1" is True **or** "S_2" is True
- "S_1" implies "S_2" is True iff "S_1" is False or "S_2" is True
- "S_1" iff S_2 is True iff "S_1implies S_2" is True and "S_2 implies S_1" is True

Example: Determining the truth of a sentence

(mode(E1) = ok implies
[(thrust(E1) = on if and only if (flow(V1) = on and flow(V2) = on)) and
(mode(E1) = ok or mode(E1) = unknown) and
not (mode(E1) = ok and mode(E1) = unknown)])

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(True implies
[(False if and only if (True and False)) and
(True or False) and
not (True and False)])

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(True implies
[(False if and only if (True and False)) and
(True or False) and
not (True and False)])

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(True implies
[(False if and only if (True and False)) and
(True or False) and
not False])

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(True implies
[(False if and only if (True and False)) and
(True or False) and
True])

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(True implies
[(False if and only if False) and
True and
True])

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(True implies
[(False if and only if False) and
True and
True])

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(True implies
[(False implies False) and (False implies False)) and
True and
True])

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(True implies
[(not False or False) and (not False or False)) and
True and
True])

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(True implies
[(True or False) and (True or False)) and
True and
True])

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(True implies
[(True and True) and
True and
True])

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(True implies
[True and
True and
True])

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(True implies
True)

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(not True or
True)

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

(False or True)

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

Example: Determining the truth of a sentence

True!

Interpretation:

mode(E1) = ok	is True
thrust(E1) = on	is False
flow(V1) = on	is True
flow(V2) = on	is False
mode(E1) = unknown	is False

If a sentence S evaluates to True in interpretation I, then:

- I satisfies S
- I is a *Model* of S

Outline

- Propositional Logic
  - Syntax
  - Semantics
  - Clausal Reduction
- Appendices

Propositional Clauses: A Simpler Form

- Literal: A proposition or its negation
  - B, Not A
- Clause: A disjunction of literals
  - (not A or B or E)
- Conjunctive Normal Form: A conjunction of clauses
  - Phi = (A or B or C) and
    (not A or B or E) and
    (not B or C or D)
  - Viewed as a set of clauses

Reduction to Clausal Form: Engine Example

(mode(E1) = ok implies
(thrust(E1) = on iff (flow(V1) = on and flow(V2) = on))) and
(mode(E1) = ok or mode(E1) = unknown) and
not (mode(E1) = ok and mode(E1) = unknown)

not (mode(E1) = ok) or not (thrust(E1) = on) or flow(V1) = on;
not (mode(E1) = ok) or not (thrust(E1) = on) or flow(V2) = on;
not (mode(E1) = ok) or not (flow(V1) = on) or not (flow(V2) = on)
or thrust(E1) = on;
mode(E1) = ok or mode(E1) = unknown;
not (mode(E1) = ok) or not (mode(E1) = unknown);

Reducing Propositional Formula to Clauses (CNF)

See Appendix for Detailed Example:

1) Eliminate IFF and Implies
   - E1 iff E2 => (E1 implies E2) and (E2 implies E1)
   - E1 implies E2 => not E1 or E2
2) Move negations in towards propositions using De Morgan's Theorem:
   - Not (E1 and E2) => (not E1) or (not E2)
   - Not (E1 or E2) => (not E1) and (not E2)
   - Not (not E1) => E1
3) Move conjunctions out using Distributivity
   - E1 or (E2 and E3) =>(E1 or E2) and (E1 or E3)

Outline

- Propositional Logic
  - Syntax
  - Semantics
  - Clausal Reduction
- Appendices

Required Appendices

You are responsible for reading and knowing this material:

1. Reduction to Clausal Form

Reduction to Clausal Form: Engine Example

(mode(E1) = ok implies
  (thrust(E1) = on iff flow(V1) = on and flow(V2) = on)) and
(mode(E1) = ok or mode(E1) = unknown) and
not (mode(E1) = ok and mode(E1) = unknown)

not (mode(E1) = ok) or not (thrust(E1) = on) or flow(V1) = on;
not (mode(E1) = ok) or not (thrust(E1) = on) or flow(V2) = on;
not (mode(E1) = ok) or not (flow(V1) = on) or not (flow(V2) = on) or
  thrust(E1) = on;
mode(E1) = ok or mode(E1) = unknown;
not (mode(E1) = ok) or not (mode(E1) = unknown);

Reducing Propositional Formula to Clauses (CNF)

1) Eliminate IFF and Implies
- E1 iff E2 => (E1 implies E2) and (E2 implies E1)
- E1 implies E2 => not E1 or E2

Eliminate IFF: Engine Example

(mode(E1) = ok implies
(thrust(E1) = on iff (flow(V1) = on and flow(V2) = on))) and
(mode(E1) = ok or mode(E1) = unknown) and
not (mode(E1) = ok and mode(E1) = unknown)

(mode(E1) = ok implies
((thrust(E1) = on implies (flow(V1) = on and flow(V2) = on)) and
((flow(V1) = on and flow(V2) = on) implies thrust(E1) = on))) and
(mode(E1) = ok or mode(E1) = unknown) and
not (mode(E1) = ok and mode(E1) = unknown)

Eliminate Implies: Engine Example

(mode(E1) = ok implies
((thrust(E1) = on implies (flow(V1) = on and flow(V2) = on)) and
((flow(V1) = on and flow(V2) = on) implies thrust(E1) = on))) and
(mode(E1) = ok or mode(E1) = unknown) and
not (mode(E1) = ok and mode(E1) = unknown)

(not (mode(E1) = ok) or
((not (thrust(E1) = on) or (flow(V1) = on and flow(V2) = on)) and
(not (flow(V1) = on and flow(V2) = on)) or thrust(E1) = on))) and
(mode(E1) = ok or mode(E1) = unknown) and
not (mode(E1) = ok and mode(E1) = unknown)

Reducing Propositional Formula to Clauses (CNF)

2) Move negations in towards propositions using De Morgan's Theorem:

- Not (E1 and E2) => (not E1) or (not E2)
- Not (E1 or E2) => (not E1) and (not E2)
- Not (not E1) => E1

Move Negations In: Engine Example

(not (mode(E1) = ok) or
((not (thrust(E1) = on) or (flow(V1) = on and flow(V2) = on)) and
(**not (flow(V1) = on and flow(V2) = on)**) or thrust(E1) = on))) and
(mode(E1) = ok or mode(E1) = unknown) and
not (mode(E1) = ok and mode(E1) = unknown)

(not (mode(E1) = ok) or
((not (thrust(E1) = on) or (flow(V1) = on and flow(V2) = on)) and
(**not (flow(V1) = on) or not (flow(V2) = on)**) or thrust(E1) = on)) and
(mode(E1) = ok or mode(E1) = unknown) and
(**not (mode(E1) = ok) or not (mode(E1) = unknown)**))

Reducing Propositional Formula to Clauses (CNF)

3) Move conjunctions out using distributivity

- E1 or (E2 and E3) =>(E1 or E2) and (E1 or E3)

Move Conjunctions Out: Engine Example

(not (mode(E1) = ok) or
  ((not (thrust(E1) = on) or (flow(V1) = on and flow(V2) = on)) and
   (not (flow(V1) = on) or not (flow(V2) = on) or thrust(E1) = on))) and
(mode(E1) = ok or mode(E1) = unknown) and
(not (mode(E1) = ok) or not (mode(E1) = unknown))

(not (mode(E1) = ok) or
  (((not (thrust(E1) = on) or flow(V1) = on) and
    (not (thrust(E1) = on) or flow(V2) = on)) and
   (not (flow(V1) = on) or not (flow(V2) = on) or thrust(E1) = on))) and
(mode(E1) = ok or mode(E1) = unknown) and
(not (mode(E1) = ok) or not (mode(E1) = unknown))

Move Conjunctions Out: Engine Example

(not (mode(E1) = ok) or
  (((not (thrust(E1) = on) or flow(V1) = on) and
    (not (thrust(E1) = on) or flow(V2) = on)) and
   (not (flow(V1) = on) or not (flow(V2) = on) or thrust(E1) = on))) and
(mode(E1) = ok or mode(E1) = unknown) and
(not (mode(E1) = ok) or not (mode(E1) = unknown))

(not (mode(E1) = ok) or not (thrust(E1) = on) or flow(V1) = on) and
(not (mode(E1) = ok) or not (thrust(E1) = on) or flow(V2) = on)) and
(not (mode(E1) = ok) or not (flow(V1) = on) or not (flow(V2) = on)
  or thrust(E1) = on) and
(mode(E1) = ok or mode(E1) = unknown) and
(not (mode(E1) = ok) or not (mode(E1) = unknown))

Reducing Propositional Formula to Clauses (CNF)

1) Eliminate IFF and Implies
   - E1 iff E2 => (E1 implies E2) and (E2 implies E1)
   - E1 implies E2 => not E1 or E2

2) Move negations in towards propositions using De Morgan's Theorem:
   - Not (E1 and E2) => (not E1) or (not E2)
   - Not (E1 or E2) => (not E1) and (not E2)
   - Not (not E1) => E1

3) Move conjunctions out using Distributivity
   - E1 or (E2 and E3) =>(E1 or E2) and (E1 or E3)