L08 – Propositional Logic

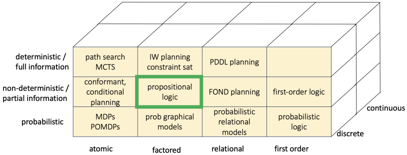
AIMA4e: Required: 7.3, 7.4, 7.6.1;
For 16.420 MiniProj : 7.7
Suggested: 7.1, 7.2, 7.6.2, 7.6.3

What you should know after this lecture

- Definition of logic: syntax and semantics
- What is logic good for?
- Propositional logic
- Logical inference
  - Model-checking: enumerative and efficient
  - Theorem proving (will do in detail next time)

Reasoning about partially-specified world states

Factored states
Boolean-valued factors



What is propositional logic and what is it good for?

- Assume a very large (for now, finite) set of possible states
- Representation is factored into of a set of Boolean state variables, called propositions
- Language for specifying huge sets of states with short descriptions (which ones depends on how we do the formalization)

It is raining $\wedge$ Nick is at the beach

- Inference procedures for determining the truth of some statement given the truth of others: semantics-preserving syntactic manipulation. Domain independent!

Logic, in general

- possible worlds: set of all possible ways the world could be (states of the environment)
- syntax: set of sentences that you can write down on paper; compositionally defined
- semantics: relationship between syntactic sentences and sets of possible worlds; also compositionally defined
- inference: ways of generating new syntactic expressions from given ones, which
  - preserve semantics,
  - no matter what the semantics are!

Propositional logic syntax

propositional symbols: uppercase letters, **True**, **False**

- propositional symbols are sentences // Called "atoms"
- if α is a sentence, then $\neg\alpha$ is a sentence // negation
- if α and β are sentences, then
  - $\alpha \vee \beta$ is a sentence // or
  - $\alpha \wedge \beta$ is a sentence // and
  - $\alpha \Rightarrow \beta$ is a sentence // implies
  - $\alpha \Leftrightarrow \beta$ is a sentence // iff

literal: an atomic sentence or a negated atomic sentence

Propositional logic models

Can think of this in two steps:

1. Imagine a domain (set of possible worlds (environment states)) you'd want to describe (e.g. classrooms of students, or hiking trips, or cars)
2. Assign a meaning of each propositional symbol to a subset of that domain that is interesting or important to your problem: e.g.,
   - P: there were more than 10 students
   - Q: there were fewer than 20 students
   - R: the lecturer was witty

For any given possible world and interpretation of the symbols, we end up with

$\boxed{\underline{\text{model}}\text{: propositional symbols} \rightarrow \text{truth value in } \{\mathit{true}, \mathit{false}\}}$

Propositional logic semantics

Model m <u>satisfies</u> sentence α if and only if one of the following holds:

- α is **True**
- α is a propositional symbol: $m(\alpha) = \textit{true}$
- $\alpha = \neg\beta$: m **does not** satisfy β
- $\alpha = (\beta \vee \gamma)$: m satisfies β **or** m satisfies γ
- $\alpha = (\beta \wedge \gamma)$: m satisfies β **and** m satisfies γ
- $\alpha = (\beta \Rightarrow \gamma)$: m satisfies $\neg\beta$ **or** m satisfies γ
- $\alpha = (\beta \Leftrightarrow \gamma)$: m satisfies $\beta \Rightarrow \gamma$ **and** m satisfies $\gamma \Rightarrow \beta$

Logical terminology

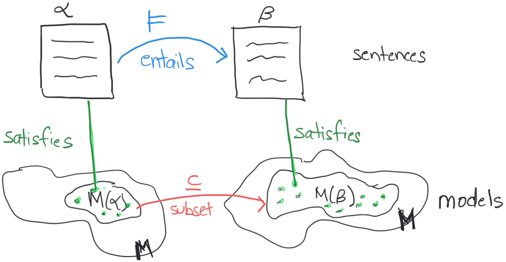
- model: a mapping between objects in the syntax and objects in the semantics; also called an interpretation
- satisfies: a model m satisfies a sentence α if α is true in m
  - Sometimes (but not in our book) written $m \models \alpha$
  - Sometimes we say m is a model of α
  - Sometimes we say α holds in m
  - $M(\alpha)$: set of all models of α
- entails: a sentence α entails sentence β, $\alpha \models \beta$, if and only if $M(\alpha) \subseteq M(\beta)$
- valid: a sentence is valid if it is satisfied in all models
- unsatisfiable: a sentence is unsatisfiable if it not satisfied in any model
- satisfiable: a sentence is satisfiable if there is at least one model in which it is satisfied

Entailment

A sentence α entails sentence β, $\alpha \models \beta$, if and only if

$$M(\alpha) \subseteq M(\beta)$$

That is, no matter whether you're thinking about hiking trips or classrooms or llamas, and what you think your symbols stand for, any model that satisfies α will also satisfy β.



Formalization practice

- W: lecturer is witty
- T: more than 10 students in class
- Z: students are asleep
- R: it's raining

Statements:

1. If the lecturer is witty, there will be more than 10 students in class.
2. Unless the lecturer is witty, the students will be asleep.
3. More than 10 students will come to class only if it's not raining.

More formalization practice

AA: Alice admits; BA: Barbara admits; AP: Alice prison; BP Barbara prison

1. If both Alice and Barbara admit to having hacked into government computers, then neither of them will receive a prison sentence.
2. But if either of them admits to having hacked into a computer while the other doesn't, she will be sentenced to imprisonment while the other won't.
3. So unless both don't admit the deed, it cannot happen that both receive a prison sentence.

Inference

- Given some information (observations) (α) what can I conclude <u>must</u> be true about the world (β)?
- Does α entail β??

Note that we can always take several observed sentences $\alpha_1, ..., \alpha_k$ and make them into a single sentence

$$\alpha_1 \wedge \ldots \wedge \alpha_n$$

Proof

Generally, a <u>proof procedure</u> takes two sentences, α and β, and tells you whether it can prove β from α:

$$\alpha \vdash \beta$$

Proof procedure is

- <u>sound</u> iff for all α, β, if $\alpha \vdash \beta$ then $\alpha \models \beta$
- <u>complete</u> iff for all α, β, if $\alpha \models \beta$ then $\alpha \vdash \beta$

Completely in syntax-land!

Stupidest possible propositional inference procedure

Recall that a model is an assignment of truth values to propositional symbols; we know the set of symbols for any given domain.

STUPID-ENTAILMENT(α, β)

> **for** each possible model m:
>> **if** SATISFIES(m, α) and not SATISFIES(m, β):
>>> **return False**
>
> **return True**

How many possible models are there?
When would this be especially painful?

Reduction of proof to satisfiability testing

Recall that:

- A sentence is unsatisfiable if it is not true in any model
- If $\alpha \wedge \neg\beta$ is unsatisfiable then $\alpha \models \beta$.

Why??

Sometimes it's easier to think up algorithms for testing satisfiability (SAT). Two strategies:

- Backtracking (DPLL)
- Local search (e.g. simulated annealing, WalkSat, etc.)

Clausal form (conjunctive normal form (CNF))

Many provers first convert all of their input to clausal form, which makes subsequent operations easier.

1. Turn all instances of $\alpha \Leftrightarrow \beta$ into $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
2. Turn all instances of $\alpha \Rightarrow \beta$ into $(\neg\alpha \vee \beta)$
3. Push negations all the way "in" using deMorgan's laws: $\neg(\alpha \wedge \beta) = (\neg\alpha \vee \neg\beta)$ and $\neg(\alpha \vee \beta) = (\neg\alpha \wedge \neg\beta)$
4. Distribute $\vee$ over $\wedge$: convert $\alpha \vee (\beta \wedge \gamma)$ to $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

You end up with a formula of the form

$$(\alpha \vee \beta \vee \ldots) \wedge (\gamma \vee \delta \vee \ldots) \wedge \ldots \wedge (\epsilon \vee \zeta \vee \ldots)$$

where all the components are literals (negated or non-negated atoms). Elements of the form $(\alpha \vee \beta \vee \ldots)$ are called clauses.

DPLL: SAT via smart backtracking

- Called "model checking" because it is operating at the level of models (assignments of values to variables).
- Assume procedure $\textsc{holds}(C, m)$ which takes a clause C and partial model m and returns one of $\{\textbf{True}, \textbf{False}, \textbf{None}\}$. Return **None** when truth value of the clause can't be determined given bindings in m.
- Initial call DPLL$(C, S, \{\,\})$ where C and S are the clauses and propositional symbols in our formula.
- A symbol p is pure in a sentence if it only appears as p or only appears as $\neg p$.
- A clause c is a unit clause in a sentence if c contains a single literal, p.

DPLL: SAT via smart backtracking

DPLL(C, S, m)

> **if** $\forall c \in C.\ \textsc{holds}(c, m) =$ **True return True**
> **if** $\exists c \in C.\ \textsc{holds}(c, m) =$ **False return False**
> $p, v := \textsc{find-pure-symbol}(S, C, m)$
> **if** p **return** DPLL$(C, S/\{p\}, m \cup \{p = v\})$
> $p, v := \textsc{find-unit-clause}(C, m)$
> **if** p **return** DPLL$(C, S/\{p\}, m \cup \{p = v\})$
> **return** DPLL$(C, S[1:], m \cup \{S[0] = \textbf{True}\})$ or
> DPLL$(C, S[1:], m \cup \{S[0] = \textbf{False}\})$

Theorem
DPLL *is sound and complete.*

So, DPLL$(\textsc{clause-form}(\alpha \wedge \neg\beta)) =$ **False** iff $\alpha \models \beta$.

SAT solving in practice

Applications of SAT solvers:

- automated testing of circuits
- product configuration
- package management
- computational biology
- cryptanalysis
- particle physics
- solving many graph problems . . .

`https://www.cs.utexas.edu/~isil/cs389L/lecture4-6up.pdf`

Next time

- Propositional theorem proving