

VIP Cheatsheet: Logic-based models

Afshine AMIDI and Shervine AMIDI

May 26, 2019

Concepts

In this section, we will go through logic-based models that use logical formulas and inference rules. The idea here is to balance expressivity and computational efficiency.

□ **Syntax of propositional logic** – By noting f, g formulas, and $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ connectives, here are the following possible logical expressions that we can write:

Remark: formulas can be built up recursively.

□ **Model** – A model w is an assignment of truth values to propositional symbols.

□ **Interpretation function** – Let f be a formula, w be a model, then the interpretation function $\mathcal{I}(f, w)$ is such that:

$$\mathcal{I}(f, w) \in \{0, 1\}$$

□ **Set of models** – We note $\mathcal{M}(f)$ the set of models w for which we have:

$$\forall w \in \mathcal{M}(f), \quad \mathcal{I}(f, w) = 1$$

□ **Knowledge base** – A knowledge base KB is defined to be a set of formulas representing their conjunction, as follows:

$$\mathcal{M}(\text{KB}) = \bigcap_{f \in \text{KB}} \mathcal{M}(f)$$

□ **Entailment** – A knowledge base KB that is said to entail f is noted $\text{KB} \models f$. We have:

$$\text{KB} \models f \iff \mathcal{M}(\text{KB}) \subseteq \mathcal{M}(f)$$

□ **Contradiction** – A knowledge base KB contradicts f if and only if we have the following:

$$\mathcal{M}(\text{KB}) \cap \mathcal{M}(f) = \emptyset$$

Remark: KB contradicts f if and only if KB entails $\neg f$.

□ **Contingency** – f is said to be contingent when there is a non-trivial overlap between the models of KB and f , i.e. when we have:

$$\emptyset \subsetneq \mathcal{M}(\text{KB}) \cap \mathcal{M}(f) \subsetneq \mathcal{M}(\text{KB})$$

Remark: we can quantify the uncertainty of the overlap of the two by computing the following quantity:

$$P(f|\text{KB}) = \frac{\sum_{w \in \mathcal{M}(\text{KB} \cup \{f\})} P(W = w)}{\sum_{w \in \mathcal{M}(\text{KB})} P(W = w)}$$

□ **Satisfiability** – A knowledge base KB is said to be satisfiable if we have:

$$\mathcal{M}(\text{KB}) \neq \emptyset$$

□ **Model checking** – Model checking is an algorithm that takes as input a knowledge base KB and checks whether we have $\mathcal{M}(\text{KB}) \neq \emptyset$.

Remark: popular model checking algorithms include DPLL and WalkSat.

□ **Modus ponens inference rule** – For any propositional symbols p_1, \dots, p_k, q , we have:

$$\frac{p_1, \dots, p_k, \quad (p_1 \wedge \dots \wedge p_k) \longrightarrow q}{q}$$

Remark: this can take linear time.

□ **Inference rule** – If f_1, \dots, f_k, g are formulas, then the following is an inference rule:

$$\frac{f_1, \dots, f_k}{g}$$

□ **Derivation** – We say that KB derives f , and we note $\text{KB} \vdash f$, if and only if f eventually gets added to KB.

□ **Soundness/completeness** – A set of inference rules can have the following properties:

Propositional logic

□ **Definite clause** – By noting p_1, \dots, p_k, q propositional symbols, we define a definite clause as having the following form:

$$(p_1 \wedge \dots \wedge p_k) \longrightarrow q$$

Remark: the case when $q = \text{false}$ is called a goal clause.

□ **Horn clause** – A Horn clause is defined to be either a definite clause or a goal clause.

□ **Modus ponens on Horn clauses** – Modus ponens is complete with respect to Horn clauses if we suppose that KB contains only Horn clauses and p is an entailed propositional symbol. Applying modus ponens will then derive p .

□ **Resolution inference rule** – The resolution inference rule is a generalized inference rule and is written as follows:

$$\frac{f_1 \vee \dots \vee f_n \vee p, \quad \neg p \vee g_1 \vee \dots \vee g_m}{f_1 \vee \dots \vee f_n \vee g_1 \vee \dots \vee g_m}$$

Remark: this can take exponential time.

□ **Conjunctive normal form** – A conjunctive normal form (CNF) formula is a conjunction of clauses.

□ **Conversion to CNF** – Every formula f in propositional logic can be converted into an equivalent CNF formula f' :

$$\mathcal{M}(f) = \mathcal{M}(f')$$

□ **Resolution rule** – By noting $\theta = \text{Unify}(p, q)$, we have:

$$\frac{f_1 \vee \dots \vee f_n \vee p, \quad \neg q \vee g_1 \vee \dots \vee g_m}{\text{Subst}[\theta, f_1 \vee \dots \vee f_n \vee g_1 \vee \dots \vee g_m]}$$

□ **Resolution-based inference** – The resolution-based inference algorithm follows the following steps:

- Step 1: Convert all formulas into CNF
- Step 2: Repeatedly apply resolution rule
- Step 3: Return unsatisfiable if and only if derive false

First-order logic

The idea here is that variables yield compact knowledge representations.

□ **Model** – A model w in first-order logic maps:

- constant symbols to objects
- predicate symbols to tuple of objects

□ **Definite clause** – By noting x_1, \dots, x_n variables and a_1, \dots, a_k, b atomic formulas, a definite clause has the following form:

$$\forall x_1, \dots, \forall x_n, \quad (a_1 \wedge \dots \wedge a_k) \rightarrow b$$

□ **Modus ponens** – By noting x_1, \dots, x_n variables and a_1, \dots, a_k, b atomic formulas, a modus ponens has the following form:

$$\frac{a_1, \dots, a_k \quad \forall x_1, \dots, \forall x_n (a_1 \wedge \dots \wedge a_k) \rightarrow b}{b}$$

□ **Substitution** – A substitution θ is a mapping from variables to terms. For instance, $\text{Subst}(\theta, f)$ returns the result of performing substitution θ on f .

□ **Unification** – Unification takes two formulas f and g and returns a substitution θ which is the most general unifier:

$$\boxed{\text{Unify}[f, g] = \theta} \quad \text{s.t.} \quad \boxed{\text{Subst}[\theta, f] = \text{Subst}[\theta, g]}$$

or fail if no such θ exists.

□ **Completeness** – Modus ponens is complete for first-order logic with only Horn clauses.

□ **Semi-decidability** – First-order logic, even restricted to only Horn clauses, is semi-decidable.

- if $\text{KB} \models f$, forward inference on complete inference rules will prove f in finite time
- if $\text{KB} \not\models f$, no algorithm can show this in finite time