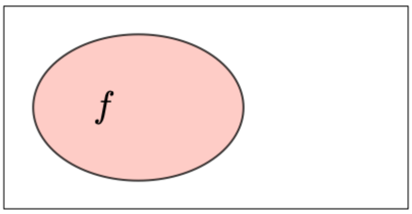
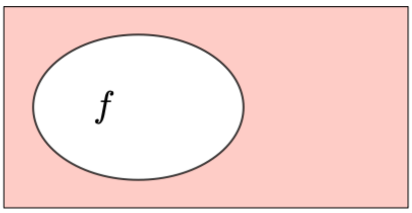
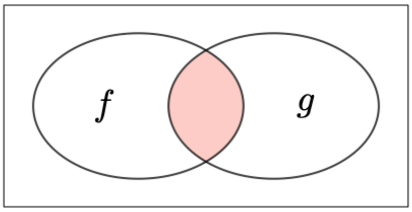
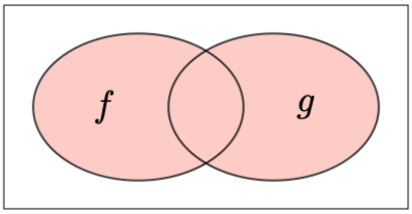
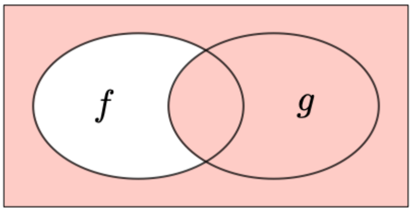
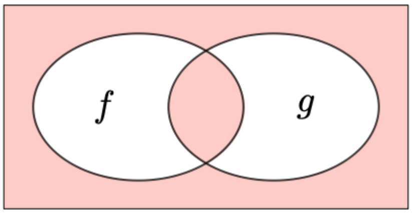
(https://stanford.edu/~shervine/teaching/cs-221/cheatsheet-logic-models#cheatsheet)Logic-based models with propositional and first-order logic

By Afshine Amidi (https://twitter.com/afshinea) and Shervine Amidi (https://twitter.com/shervinea)

(https://stanford.edu/~shervine/teaching/cs-221/cheatsheet-logic-models#basics) Basics

❐ **Syntax of propositional logic** — By noting f, g formulas, and $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ connectives, we can write the following logical expressions:

Name	Symbol	Meaning	Illustration
Affirmation	f	f	
Negation	$\neg f$	not f	

Conjunction	$f \wedge g$	f and g	
Disjunction	$f \vee g$	f or g	
Implication	$f \rightarrow g$	if f then g	
Biconditional	$f \leftrightarrow g$	f, that is to say g	

Remark: formulas can be built up recursively out of these connectives.

❒ **Model** — A model w denotes an assignment of binary weights to propositional symbols.

Example: the set of truth values $w = \{A : 0, B : 1, C : 0\}$ *is one possible model to the propositional symbols* A, B *and* C.

❒ **Interpretation function** — The interpretation function $\mathcal{I}(f, w)$ outputs whether model w satisfies formula f:

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

❒ **Set of models** — $\mathcal{M}(f)$ denotes the set of models w that satisfy formula f. Mathematically speaking, we define it as follows:

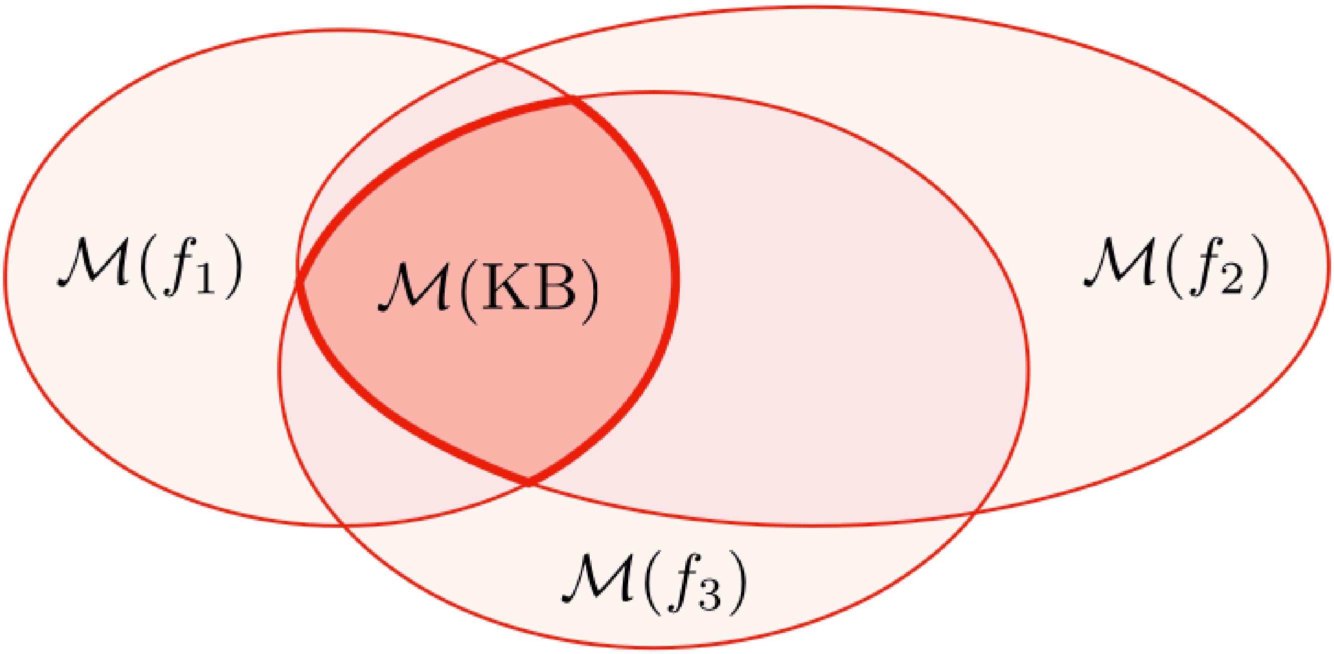
$$\boxed{\mathcal{M}(f) = \{w \,|\, \mathcal{I}(f,w) = 1\}}$$

(https://stanford.edu/~shervine/teaching/cs-221/cheatsheet-ogic-models#knowledge-base)

Knowledge base

❒ **Definition** — The knowledge base $\textrm{KB}$ is the conjunction of all formulas that have been considered so far. The set of models of the knowledge base is the intersection of the set of models that satisfy each formula. In other words:

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



❒ **Probabilistic interpretation** — The probability that query f is evaluated to 1 can be seen as the proportion of models w of the knowledge base $\textrm{KB}$ that satisfy f, *i.e.*:

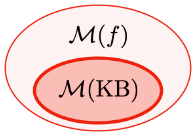
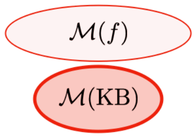
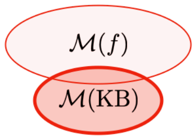
$$\boxed{P(f\,|\,\textrm{KB})=\frac{\displaystyle\sum_{w\in\mathcal{M}(\textrm{KB})\cap\mathcal{M}(f)}P(W=w)}{\displaystyle\sum_{w\in\mathcal{M}(\textrm{KB})}P(W=w)}}$$

❒ **Satisfiability** — The knowledge base $\textrm{KB}$ is said to be satisfiable if at least one model w satisfies all its constraints. In other words:

$$\boxed{\textrm{KB satisfiable}\Longleftrightarrow\mathcal{M}(\textrm{KB})\neq\varnothing}$$

Remark: $\mathcal{M}(\textrm{KB})$ denotes the set of models compatible with all the constraints of the knowledge base.

❒ **Relation between formulas and knowledge base** — We define the following properties between the knowledge base KB and a new formula f:

Name	Mathematical formulation	Illustration	Notes
KB entails f	$\mathcal{M}(\mathrm{KB}) \cap \mathcal{M}(f) = \mathcal{M}(\mathrm{KB})$		• f does not bring any new information • Also written $\mathrm{KB} \models f$
KB contradicts f	$\mathcal{M}(\mathrm{KB}) \cap \mathcal{M}(f) = \varnothing$		• No model satisfies the constraints after adding f • Equivalent to $\mathrm{KB} \models \neg f$
f contingent to KB	$\mathcal{M}(\mathrm{KB}) \cap \mathcal{M}(f) \neq \varnothing$ and $\mathcal{M}(\mathrm{KB}) \cap \mathcal{M}(f) \neq \mathcal{M}(\mathrm{KB})$		• f does not contradict KB • f adds a non-trivial amount of information to KB

❒ **Model checking** — A model checking algorithm takes as input a knowledge base KB and outputs whether it is satisfiable or not.

Remark: popular model checking algorithms include DPLL and WalkSat.

❒ **Inference rule** — An inference rule of premises $f_1, ..., f_k$ and conclusion g is written:

$$\boxed{\frac{f_1, ..., f_k}{g}}$$

❒ **Forward inference algorithm** — From a set of inference rules Rules, this algorithm goes through all possible $f_1, ..., f_k$ and adds g to the knowledge base KB if a matching rule exists. This process is repeated until no more additions can be made to KB.

❒ **Derivation** — We say that KB derives f (written $\mathrm{KB} \vdash f$) with rules Rules if f already is in KB or gets added during the forward inference algorithm using the set of rules Rules.

❐ **Properties of inference rules** — A set of inference rules Rules can have the following properties:

Name	Mathematical formulation	Notes
Soundness	$\{f \mid \mathrm{KB} \vdash f\} \subseteq \{f \mid \mathrm{KB} \models f\}$	• Inferred formulas are entailed by KB • Can be checked one rule at a time • *"Nothing but the truth"*
Completeness	$\{f \mid \mathrm{KB} \vdash f\} \supseteq \{f \mid \mathrm{KB} \models f\}$	• Formulas entailing KB are either already in the knowledge base or inferred from it • *"The whole truth"*

(https://stanford.edu/~shervine/teaching/cs-221/cheatsheet-logic-models#propositional-logic) Propositional logic

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.

❐ **Horn clause** — By noting $p_1, ..., p_k$ and q propositional symbols, a Horn clause has the form:

$$\boxed{(p_1 \wedge ... \wedge p_k) \longrightarrow q}$$

Remark: when $q = \mathrm{false}$*, it is called a "goal clause", otherwise we denote it as a "definite clause".*

❐ **Modus ponens** — For propositional symbols $f_1, ..., f_k$ and p, the modus ponens rule is written:

$$\boxed{\frac{f_1, ..., f_k, \quad (f_1 \wedge ... \wedge f_k) \longrightarrow p}{p}}$$

Remark: it takes linear time to apply this rule, as each application generate a clause that contains a single propositional symbol.

❒ **Completeness** — 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.

❒ **Conjunctive normal form** — A conjunctive normal form (CNF) formula is a conjunction of clauses, where each clause is a disjunction of atomic formulas.

Remark: in other words, CNFs are $\wedge$ of $\vee$.

❒ **Equivalent representation** — Every formula in propositional logic can be written into an equivalent CNF formula. The table below presents general conversion properties:

Rule name		**Initial**	**Converted**
Eliminate	$\leftrightarrow$	$f \leftrightarrow g$	$(f \rightarrow g) \wedge (g \rightarrow f)$
	$\rightarrow$	$f \rightarrow g$	$\neg f \vee g$
	$\neg\neg$	$\neg\neg f$	f
Distribute	$\neg$ over $\wedge$	$\neg(f \wedge g)$	$\neg f \vee \neg g$
	$\neg$ over $\vee$	$\neg(f \vee g)$	$\neg f \wedge \neg g$
	$\vee$ over $\wedge$	$f \vee (g \wedge h)$	$(f \vee g) \wedge (f \vee h)$

❒ **Resolution rule** — For propositional symbols $f_1, ..., f_n$, and $g_1, ..., g_m$ as well as p, the resolution rule is written:

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

Remark: it can take exponential time to apply this rule, as each application generates a clause that has a subset of the propositional symbols.

❒ **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 False is derived

(https://stanford.edu/~shervine/teaching/cs-221/cheatsheet-logic-models#first-order-logic) First-order logic

The idea here is to use variables to yield more compact knowledge representations.

❐ **Model** — A model w in first-order logic maps:

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

❐ **Horn clause** — By noting $x_1, ..., x_n$ variables and $a_1, ..., a_k, b$ atomic formulas, the first-order logic version of a horn clause has the form:

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

❐ **Substitution** — A substitution θ maps variables to terms and $\textrm{Subst}[\theta, f]$ denotes the result of substitution θ on f.

❐ **Unification** — Unification takes two formulas f and g and returns the most general substitution θ that makes them equal:

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

Note: $\textrm{Unify}[f, g]$ *returns* $\textrm{Fail}$ *if no such* θ *exists.*

❐ **Modus ponens** — By noting $x_1, ..., x_n$ variables, $a_1, ..., a_k$ and $a'_1, ..., a'_k$ atomic formulas and by calling $\theta = \textrm{Unify}(a'_1 \wedge ... \wedge a'_k, a_1 \wedge ... \wedge a_k)$, the first-order logic version of modus ponens can be written:

$$\boxed{\frac{a'_1, ..., a'_k \quad \forall x_1, ..., \forall x_n (a_1 \wedge ... \wedge a_k) \rightarrow b}{\textrm{Subst}[\theta, b]}}$$

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

❒ **Resolution rule** — By noting $f_1, ..., f_n$, $g_1, ..., g_m$, p, q formulas and by calling $\theta = \textrm{Unify}(p, q)$, the first-order logic version of the resolution rule can be written:

$$\boxed{\frac{f_1 \vee ... \vee f_n \vee p, \quad \neg q \vee g_1 \vee ... \vee g_m}{\textrm{Subst}[\theta, f_1 \vee ... \vee f_n \vee g_1 \vee ... \vee g_m]}}$$

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

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

(https://twitter.com/shervinea) (https://linkedin.com/in/shervineamidi) (https://github.com/shervinea) (https://scholar.google.com/citations?user=nMnMTm8AAAAJ) (https://www.amazon.com/stores/author/B0B37XBSJL)