Satisfiability
\( X \models \alpha \) (Model Checking)

Provability
\( X \vdash \alpha \) (Theorem Proving)

What's the relationship between them?

**Soundness**
\( X \vdash \alpha \) \( \implies \) \( X \models \alpha \).
(Proof by induction on formula and rule.)

**Completeness**
\( X \not\vdash \alpha \) \( \implies \) \( X \not\models \alpha \).

First, strengthen the statement
\( X \not\vdash \bot \land X \models \alpha \) \( \implies \) \( X \not\models \alpha \).
\( x \) consistent.

(A)

\[
\frac{x \vdash \bot}{x \vdash \beta \land \\neg \beta} \frac{x \vdash \beta}{x \vdash \alpha} \frac{x \vdash \neg \beta}{x \vdash \alpha} (\forall \alpha)
\]

(B)

\[
\frac{x, \alpha \vdash \bot}{x, \alpha \vdash \alpha} \frac{\alpha \vdash \alpha}{x, \alpha \vdash \alpha} \frac{x \vdash \alpha}{x \vdash \alpha} (\forall \alpha)
\]

\( x \not\vdash \alpha \), \( x \not\vdash \bot \) \( \implies \) \( x \not\vdash \bot, x, \neg \alpha \not\vdash \bot \)
\( \implies \) \( x \cup \neg \alpha \models \) consistent
implies \( X \not\models \alpha \).
\[ Y \subseteq X \cup \neg \alpha \]

Maximally consistent superset of \( X \) containing \( \neg \alpha \)

\[ Y \text{ satisfiable } \Rightarrow X \text{ satisfiable.} \]

But also \( X \cup \neg \alpha \) is satisfiable.

**Completeness Proof**

\[ X \vdash \alpha \quad (\wedge \text{ } X \text{ consistent}) \]

\[ \Rightarrow X \cup \neg \alpha \text{ consistent} \]

\[ \Rightarrow \exists Y \quad Y \subseteq X \cup \neg \alpha \text{ & } Y \text{ Maximaly consistent} \]

\[ \text{Lindenbaum's Lemma} \]

\[ \vdash Y \text{ satisfiable} \]

\[ \Rightarrow X \cup \neg \alpha \text{ satisfiable} \]

\[ \Rightarrow X \vdash \alpha \quad \Box \]

---

**Defn.** (a) \( \alpha \in F \) is called **inconsistent**

\[ \text{if } X \vdash \alpha \quad \forall \alpha \in F; \text{ otherwise, consistent.} \]

(b) \( Y \in F \) is called **maximally consistent**

\[ \text{if } Y \text{ consistent but each } \neg \exists Y \text{ is inconsistent.} \]

**Lindenbaum's Lemma:**

Every consistent set \( X \in F \) can be extended to a maximally consistent set \( X \cup \neg \alpha \).

**Satisfiability Lemma:**

Every maximally consistent set \( Y \) is satisfiable.
Lindenbaum's Lemma:

Proof: Let \( H \) be the set of all consistent \( Y \supseteq X \), partially ordered with respect to \( \subseteq \) relation.

\[ H = \{ Y | Y \supseteq X \land Y \neq \emptyset \} \]

(a) \( H \neq \emptyset \) \( \iff \) \( x \in H \)

(b) \( \exists k \in H \) \( k \) is a chain. \( \iff \forall Y, Z \in k \) \( Y \subseteq X \lor Z \subseteq Y \).

\( U \) is \( \supseteq U_k \), upper bound for the chain \( k \).

- \( U \neq \emptyset \) (\( U \) is consistent)

  Suppose not: \( U \neq \emptyset \)

  \[ \Rightarrow u_0 \neq \emptyset \land u_0 \text{ is finite} \iff u_0 = \{ a_1, \ldots, a_n \} \]

  \[ \therefore a_1, \ldots, a_n \neq \emptyset \]

  \[ a_i \in Y, i \in K \]

  Let \( Y \) be the biggest among \( Y_1, \ldots, Y_n \).

  \[ \Rightarrow a_1, \ldots, a_n \subseteq Y \]

  \[ \frac{a_1, \ldots, a_n \neq \emptyset}{Y \neq \emptyset} \quad (\text{by NR}) \]

  \[ \Rightarrow Y \notin H \Rightarrow \# \]

(c) By Zorn's Lemma: \( H \) has a maximal element, \( x' \).

\( x' \supseteq x \) and \( x' = \text{maximally consistent} \).

\( \square \)
Satisfiability Lemma:

\[ \forall \alpha \ V \vdash \alpha \iff \omega \models \alpha. \]

\( \iff \) (i.e. \( \omega \) is model for \( \forall \))

\( \forall \beta \ V \vdash \alpha, \beta \) (\( \vdash \forall \beta \ \& \ \forall \alpha \))

\( \iff \omega \models \alpha \) and \( \omega \models \beta \)

\( \iff \omega \models \alpha \land \beta. \)

\[ \vdash \alpha \] (\( \vdash \) maximal of \( \forall \))

\[ \iff \omega \not\models \alpha \]

\[ \iff \omega \models \alpha \]

\( \vdash \alpha \land \beta \) \( \vdash \alpha \)

\( \vdash \alpha \lor \beta \)

is a consistent extension of \( \forall \)

\( \Rightarrow \) \( \forall \alpha \in \forall \)

\( \Rightarrow \forall \vdash \alpha. \)
Satisfiability: Hornsat.

Horn Clause:
A Horn clause is a disjunction of literals in which all or nearly all of the literals are complemented.
(At most one of its literals is pure.)

Example:
\[ x \equiv T \equiv x \quad (w: x \mapsto 1) \]
\[ \overline{x} \lor \overline{y} \equiv x \land y \equiv 1 \quad (w: x \mapsto 0, y \mapsto 0) \]
\[ x \lor \overline{y} \lor z \equiv y \lor z \equiv x \quad (w: x \mapsto 1, y \mapsto 0, \\
or z \mapsto 0) \]

Hornsat Algorithm (Greedy)

1. **Initialize**
   Assign all variables false.
   (Thus initially all clauses of the following form will be satisfied.
   \[ x \land y \equiv 1 \]
   \[ \overline{w} \lor x \land y \equiv z \].
   But not \[ T \equiv x \].

2. **Update**
   For a clause whose r.h.s. are not satisfied
   choose one & FLIP the truth assignment to true.

3. **Reevaluate** all clauses and repeat (2), until no more variable can be FLIPPED (unsatisfiable)
   or a satisfiable assignment has been found.

# Clauses = m , # variables = n.
Complexity = \( O(mn) \).
**PROLOG PROGRAM**

**Defn:**
1. A ** Horn clause ** is a clause that contains at most one positive literal.
2. A ** program clause ** is one that contains exactly one positive literal.
   
   \[ \text{A} : = \ B_1, B_2, \ldots, B_n. \]

3. If a ** program clause ** contains some negative literals it is called a rule \((n > 0)\)

4. A ** unit clause ** (fact) is one that consists of exactly one positive literal.
   
   \[ \text{A.} \text{ or } \text{A} : = \]

5. A ** goal clause ** is one that contains no positive literals.

\[ ? - \]

6. A ** PROLOG program ** is a set of clauses containing only ** program clauses **.

\[ \{ \text{Rules and Facts} \} \]

**Lemma:** If a set of Horn clauses \( S \) is unsatisfiable then \( S \) must contain at least one fact and one goal clause.

**Proof:** If \( S \) contains no fact, then assign every prime variable \( \text{false} \). \( S \Rightarrow \text{Satisfiable} \).

If \( S \) contains no goal clause then assign every prime variable \( \text{true} \). \( S \Rightarrow \text{Satisfiable} \).
General view of a Prolog Program:

Given: A collection of facts and rules \( \equiv \text{Program } P \).

Deduce: If a conjunction of some facts \( \exists q_1, q_2, \ldots, q_n \)

\[ \text{?- } q_1, q_2, \ldots, q_n \]

is a consequence of \( P \).

\[ G = \{ q_1, q_2, \ldots, q_n \} \]

**Lemma:** \( q_i \)'s are consequences of \( P \) iff \( P \cup G \) is unsatisfiable.

**Proof:** H.W.

---

More Complex Prolog Program (Need use 1st order logic).

parent \((x, y)\) :- mother \((x, y)\).
parent \((x, y)\) :- father \((x, y)\).
daughter \((x, y)\) :- mother \((y, x)\), female \((x)\).
son \((x, y)\) :- mother \((y, x)\), male \((x)\).
child \((x, y)\) :- son \((x, y)\).
child \((x, y)\) :- daughter \((x, y)\).
daughter \((x, y)\) :- father \((y, x)\), female \((x)\).
son \((x, y)\) :- father \((y, x)\), & male \((x)\).

male \((\text{sam})\).
male \((\text{tom})\).
female \((\text{kim})\).
father \((\text{bud, kim})\).
female \((\text{jane})\).
mother \((\text{jane, sam})\).

\[ \text{?- parent } (\text{bud, sam}) \]