3. Resolution

Goal: Check $\Phi \models \Psi$ automatically.

The definition of entailment is not suitable for automation, because one would have to check all (infinitely many) interpretations $I$ and find out whether $I \models \Phi$ implies $I \models \Psi$.

Instead: develop a calculus that allows to prove $\Phi \models \Psi$ in a syntactical (automatable) way.

Calculus is sound if: if the calculus can deduce $\Psi$ from $\Phi$, then $\Phi \models \Psi$ really holds.

Calculus is complete if: whenever $\Phi \models \Psi$, then the calculus can deduce $\Psi$ from $\Phi$.

Logic programming uses the resolution calculus, which is indeed sound and complete.

Idea: instead of entailment ($\Phi \models \Psi$),
we examine an unsatisfiability problem

**Lemma 3.0.1 (From Entailment to Unsatisfiability)**

Let \( \psi_1, \ldots, \psi_k, \psi \in \mathcal{F}(\Sigma, \Delta, \mathcal{U}) \).

Then we have \( \{ \psi_1, \ldots, \psi_k \} \models \psi \) iff

the formula \( \psi_1 \wedge \ldots \wedge \psi_k \wedge \neg \psi \) is unsatisfiable.

**Proof:**

\( \{ \psi_1, \ldots, \psi_k \} \models \psi \)

iff for all interpretations \( I \) with \( I \models \{ \psi_1, \ldots, \psi_k \} \),

we have \( I \models \psi \)

iff there is no interpretation \( I \) with \( I \models \{ \psi_1, \ldots, \psi_k \} \) and \( I \not\models \neg \psi \)

iff \( \psi_1 \wedge \ldots \wedge \psi_k \wedge \neg \psi \) is unsatisfiable. \( \blacksquare \)

**Ex. 3.0.2**

To show that in the logic prog. with the fact

motherOf (ren, sus)

the query \( \neg \text{motherOf}(X, \text{sus}) \) holds, one has to show:

\( \{ \text{motherOf}(\text{ren}, \text{sus}) \} \models \exists X \text{ motherOf}(X, \text{sus}) \)

Instead, one can show unsatisfiability of
\[ \{ \text{motherOf}(\text{ren}, \text{sus}), \exists X \text{ motherOf}(X, \text{sus}) \} \]

In general: Unsatisfiability of logic formulas is undecidable. (This: Entailment is also undecidable.)

This means: There is no terminating algorithm that can determine whether \( \overline{\Phi} \models \Psi \) holds or not.

But: Unsatisfiability (and entailment) is semi-decidable.

This means: There is an algorithm which terminates whenever \( \overline{\Phi} \models \Psi \) holds (and which determines that \( \overline{\Phi} \models \Psi \) holds). But if \( \overline{\Phi} \not\models \Psi \), then the algorithm might not terminate.

The resolution calculus is such a semi-decision algorithm for unsatisfiability.

Ex: empty prog.

query: \( \exists X \) motherOf \( (X, sus) \)

To check: \( \models \exists X \) motherOf \( (X, sus) \)

Equivalently: \( \{ \exists X \) motherOf \( (X, sus) \} \) unsatisfiably?
Goal: Introduce technique to check unsatisfiability of formulas automatically.

3.1. Skolem Normal Form

Aim: Simplify any formula to the following form:

\[ \forall X_1, \ldots, X_n \eta \]

where \( \eta \) is quantifier-free and \( \text{Var}(\eta) \subseteq \{X_1, \ldots, X_n\} \)

First step: Transform any formula to prenex normal form.

**Def 3.1.1 (Prenex Normal Form)**

A formula \( \eta \) is in prenex normal form iff it has the form

\[ Q_1 X_1 Q_2 X_2 \ldots Q_n X_n \eta \]

with \( Q_n, \ldots, Q_1 \in \{\forall, \exists\} \) and \( \eta \) is quantifier-free.

**Thm 3.1.2 (Transformation into prenex normal form)**

For every formula \( \eta \) one can automatically construct an equivalent formula \( \eta' \) in prenex normal form.

Proof:

- Replace all sub-formulas \( \eta_1 \rightarrow \eta_2 \) by
\( (\phi_1 \rightarrow \phi_2) \land (\phi_2 \rightarrow \phi_1) \)

* Replace all sub-formulas \( \phi_1 \rightarrow \phi_2 \) by \( \neg \neg \phi_1 \lor \phi_2 \).

Then we apply the following algorithm \textsc{Prenex}:

* If \( \phi \) is quantifier-free, then return \( \phi \).

* If \( \phi = \neg \phi_1 \), then compute \( \textsc{Prenex}(\neg \phi_1) = \neg \neg \phi_1 \land \exists x_1 \ldots \exists x_n \phi \).

Then return \( \neg \neg \phi_1 \land \exists x_1 \ldots \exists x_n \phi \),

where \( \neg = \exists \), \( \exists = \forall \).

* If \( \phi = \phi_1 \cdot \phi_2 \) where \( \cdot \in \{ \land, \lor \} \), then compute

\begin{align*}
\textsc{Prenex}(\phi_1) & = \neg \neg \phi_1 \land \exists x_1 \ldots \exists x_n \phi_1 \\
\textsc{Prenex}(\phi_2) & = \forall y_1 \ldots \forall y_m \phi_2
\end{align*}

By renaming bound variables, ensure that \( x_1, \ldots, x_n \) do not occur in \( \forall y_1 \ldots \forall y_m \phi_2 \)
and that \( y_1, \ldots, y_m \) do not occur in \( \neg \neg \phi_1 \land \exists x_1 \ldots \exists x_n \phi_1 \).

Then return:

\begin{align*}
\neg \neg \phi_1 \land \exists x_1 \ldots \exists x_n \phi_1 \land \forall y_1 \ldots \forall y_m \phi_2
\end{align*}

* If \( \phi = QX \phi_1 \) with \( Q \in \{ \forall, \exists \} \).

Then compute \( \textsc{Prenex}(\phi_1) = \forall x_1 \ldots \forall x_n \phi_1 \).

By renaming bound variables, ensure that
Ex. 3.1.3 Transform the following formula into prenex normal form:

\[
\neg \exists X (\text{married}(X, Y) \lor \neg \exists Y \text{mother of}(X, Y))
\]

\[
\forall Y \neg \text{mother of}(X, Y)
\]

\[
\forall Z \neg \text{mother of}(X, Z)
\]

\[
\neg \exists X \forall Z (\text{married}(X, Y) \lor \neg \text{mother of}(X, Z))
\]

\[
\forall X \exists Z \neg (\text{married}(X, Y) \lor \neg \text{mother of}(X, Z))
\]

Ex 3.14

Log. Req. with the fact \( \neg \text{mother of}(\text{ren, sus}) \).

Query \( \neg \text{mother of}(X, \text{sus}) \).

We have to show unsatisfiability of \( \neg \text{mother of}(\text{ren, sus}) \lor \exists X \text{mother of}(X, \text{sus}) \).

This can be transformed to prenex normal form:

\[
\neg \text{mother of}(\text{ren, sus}) \lor \forall X \neg \text{mother of}(X, \text{sus})
\]

\[
\forall X (\neg \text{mother of}(\text{ren, sus}) \lor \neg \text{mother of}(X, \text{sus}))
\]
Def 3.4.5 (Skolem Normal Form)
A formula \( \phi \) is in Skolem normal form iff it is closed and it has the form \( \forall X_1, \ldots, X_n \phi \), where \( \phi \) is quantifier-free.

For every formula, there is an equivalent formula in prenex normal form.
This is not true for Skolem normal forms.
For example, \( \exists X \text{ female}(X) \) or \( \exists X \text{ female}(X) \)
have no equivalent formula in Skolem normal form.
We only need that the original formula is satisfiable iff the corresponding formula in Skolem normal form is satisfiable.

In the example:
\[
\exists X \text{ female}(X) \quad \text{and} \quad \text{female}(a)
\]
are satisfiability-equivalent.

\[
\forall Y \exists X \text{ married}(X,Y) \quad \text{and} \quad \forall Y \text{ married}(f(Y),Y)
\]
are fresh function symbols of arity 0.
Thm 3.1.6 (Transformation into Skolem Normal Form)

For every formula $\varphi$, one can automatically construct a formula $\varphi'$ in Skolem normal form such that $\varphi$ is satisfiable iff $\varphi'$ is satisfiable.

**Proof:** First, $\varphi$ is transformed to prenex normal form as in Thm 3.1.2. This results in a formula $\varphi_1$.

Let $X_1, \ldots, X_n$ be the free variables of $\varphi_1$.

Then transform $\varphi_1$ to $\varphi_2 = \exists X_1, \ldots, X_n \varphi_1$. Clearly $\varphi_2$ and $\varphi_1$ are satisfiability-equivalent:

1. $I \models \varphi_1$ for $I = (A, \alpha, \beta)$
2. $I \models \exists X_1/\beta(X_1), \ldots, X_n/\beta(X_n) \varphi_1$ \models \varphi_1
3. $I \models \exists X_1, \ldots, X_n \varphi_1$.

$I \models \varphi_2$ with $I = (A, \alpha, \beta)$

4. $I \models \exists X_1, \ldots, X_n \varphi_1$

5. There exist $a_1, \ldots, a_n \in A$ with $I \models X_1/a_1, \ldots, X_n/a_n \varphi_1 = \varphi_1$
6. $\varphi_1$ is satisfiable.

Now $\varphi_2$ is a closed formula in prenex normal form.
We eliminate the existential quantifiers from the outside to the inside.

If \( \phi_2 \) is \( \forall X_n, \ldots, X_1 \exists Y \psi \),

then replace it by \( \forall X_1, \ldots, X_n \forall Y \exists Y / f(X_1, \ldots, X_n) \)

where \( f \) is a fresh function symbol of arity \( n \).

One can prove that this does not change satisfiability of the formula (by substitution lemma 2.2.3).

\[
\text{Ex} \begin{array}{ll}
3.17 & \rightarrow \exists X (\text{married}(X,Y) \lor \exists Y \text{mother}_f(X,Y)) \\
\downarrow \text{prefix normal form (Ex. 3.1.3)}
\end{array}
\]

\[
\forall X \exists T \rightarrow (\text{married}(X,Y) \lor \neg \text{mother}_f(X,Y))
\]

\[
\downarrow \text{get rid of free var. } Y
\]

\[
\exists Y \forall X \exists T \rightarrow (\text{married}(X,Y) \lor \neg \text{mother}_f(X,Y))
\]

\[
\downarrow \text{replace } Y \text{ by fresh } a \in \Sigma.
\]

\[
\forall X \exists T \rightarrow (\text{married}(X,a) \lor \neg \text{mother}_f(X,a))
\]
\[ \forall X \neg (\text{married}(X, a) \lor \neg \text{mother of}(X, f(X))) \]