Lemma: Learning

\[ A \lor \neg A \]

\[ C \lor C \]

\[ D \lor D \]

Closed by \( C \lor \neg A \)

Closed by \( D \lor \neg A \)

Resolution ("Cut")

\[ D \lor \neg A \]

Allows to derive its equivalent tree earlier:
Translation of DLs into FO logic

T-Box:

\[ \forall x \ (\text{Professor}(x) \land \exists y \ (\text{supervises}(x, y) \land \text{Student}(y))) \rightarrow \text{Busy Person}(x) \]

"general"

Now use any First-order logic Theorem Prover.

Problems:

- Efficiency
- It might not terminate at all

Research challenge:

- Find refinements that avoid problems
Conclusions

"Parameter Space":

- Logic: Propositional - Descriptive logic - First-Order

- Theories: *all* models -

  specific models (integers...)

- Services: Theorem Proving (validity)
  - Disproving (unsatisfiability)
  - Abduction

  \[
  \frac{A \rightarrow B}{B} \quad \frac{B}{A}
  \]

Challenges: Build a theorem prover that matches your application...
\[ (\exists x \ P(x)) \land \neg P(x) \]
\[ \Rightarrow \]
\[ \exists x \ (P(x) \land \neg P(x)) \]

But

\[ (\exists x \ P(x)) \land \neg P(x) \]
\[ \Rightarrow \]
\[ \exists y \ (P(y) \land \neg P(x)) \]
Skolemization notes

(1) Not an equivalence transformation
Take \( N \cup \{ "f(x) = x" \} \models \forall x \exists y \, x \neq y \)
but \( N \cup \{ "f(x) = x" \} \not\models \forall x \exists y \, f(x) \)
(The converse does hold)

(2) "Mini-scooping" in provemnt
\[
\forall x \forall y \, \exists z \in \mathbb{R} \, P(x, y) \land Q(x, z) \\
\overset{\text{"official"}}{=}
\forall x \forall y \left( \forall z \, P(x, y) \land Q(x, z) \right) \land \exists z \, Q(x, z)
\overset{\text{\( \exists x \in \mathbb{R} \) version}}{=}
\forall x \exists z \, \forall y \, P(x, y) \land Q(x, z)
\overset{\text{\( \forall x \) version}}{=}
\forall y \exists z \, \forall x \, P(x, y) \land Q(x, z)
\overset{\text{Better?}}{=}
\forall x \forall z \, P(x, y) \land Q(x, f(x))
\]
Duplication of subformulas

\[ \varphi' : (A_1 \land \ldots \land A_m) \lor (B_1 \land \ldots \land B_m) \]

\[ \land \]
\[ m \times n \text{ clauses} \]

In general exponential blowup!

Solution: Linear transformation by "flattening" subformulas:

\[ A \rightarrow (A) \]

Transform \( \varphi \) into:

\[ A \leftrightarrow (A_1 \land \ldots \land A_m) \]
\[ \lor \]
\[ B \leftrightarrow (B_1 \land \ldots \land B_m) \]
\[ A \lor B \]

\[ \rightarrow m+1 \text{ clauses disj} \]
\[ \rightarrow m+1 \text{ clause disj} \]
\[ \rightarrow m+1 \text{ clause disj} \]
\[ \sum = m+n+3 \text{ clauses disj} \]

( a little more complicated for FOCL, but idea remains the same)
Constructing a Herbrand Interpretation 

from an Interpretation (by example)

Recall \( U_N = \{ 0, 1, 2, \ldots \} \)
\( \leq_N = \{ (0,0), (0,1), \ldots, (1,1), (1,2), \ldots \} \)
\( S_N = \lambda x \cdot x + 1 \)

Herbrand Interpretation for \( \Sigma \) pre

Take any ground \( \Sigma \) pre \( \Delta \) atom

e.g. \( \Delta = ( s(c) \leq s(c) + s(c) ) \)

\( \leq_{\Delta, \Delta} \)

\( (1, 1, 2) \in \leq_N \Rightarrow \Delta \in \Sigma \)

Therefore set

\( \Delta = ( s(c) = 0 ) \)

\( \in \)

\( (1, 0) \in \leq_N \Rightarrow \Delta \not\in \Sigma \)

In Construction \( J \) "says the same"

as \( U_N \) wrt. predicates
Indirect: \( \forall p \) is unsatisfiable

\[ \neg \exists p \wedge A \neg p \]

Completeness, Soundness of \( \mathcal{M} \)

Proof is a finite object -

Use witnesses of terms for \( \exists \)-variables to determine the ground instances of \( p \)

Direct (clause logic)

To show: Every finite set of ground instances of \( \mathcal{M} \) is satisfiable

Proof idea: enumerate interpretations,

\[ P(a, q), \neg P(a, a) \]

\[ P(a, f(q)), \neg P(a, f(q)) \]

\[ P(a, f(q)), \neg P(a, f(q)) \]

\[ P(a, f(q)), \neg P(a, f(q)) \]

At any stage

At least one non-closed branch must exist (otherwise some finite set of ground instances is unsatisfiable)

Which defines a model for \( \mathcal{M} \)
Soundness and (Refutation) Completeness

Given: N set of clauses

Soundness of Resolution

To show:

For any clause set N:
Prop Resolution derives the empty clause I from N then N is unsatisfiable

Equivalently:

If N is satisfiable then
Prop Resolution does not derive I

It suffices to show that
Derived clauses are consequences of parent clauses

Proof Idea:

Assume \( J = N \)

Show \( J = N \cup \exists C \) for any derived clause C
(\( J \) follows \( C \neq I \))

Induction on length of derivation
To show: $N$ is unsatisfiable $\implies$ Resolution derives $\bot$

Proof by "Semantic tree method".

Example:

$N = \{ \neg A \lor B \lor D, \neg D, \neg A \lor B, A \}$

\begin{align*}
(1) & \\
(2) & \\
(3) & \\
(4) & \\
(5) & \text{new clause}
\end{align*}

Fact: $N$ is unsatisfiable $\implies T$ is closed.

Read off Resolution inferences:

Repeat this until root-only tree derived (gives $\bot$)
Resolution Strategies
(for better efficiency)

- Ordered Resolution
  - Ordering: \( A > B > D \)
  - Resolution and Factoring on maximal literals only

\[
\begin{align*}
\neg A \lor B & \quad \lor \quad \neg B \lor C \\
\phantom{\neg A \lor B} \quad \lor \quad \neg B \lor D \\
\neg A \lor D
\end{align*}
\]

\( \neg A \lor B \quad \lor \quad \neg B \lor C \quad \mathrm{ok} \)

\( \neg A \lor D \quad \neg B \lor C \quad \mathrm{not \ ok} \)  
\( \neg B \lor \neg A \lor D \)  
(\( B \) not maximal)

- Completeness: "easy" - use semantic tree
  - according to ordering (smaller literals towards root)

Help/Preresolution
  - only positive literals \( \lor \)
  - all negative resolved

\[
\begin{align*}
A \lor B & \quad \lor \quad C \lor D \quad \neg A \lor \neg C \lor E \\
\phantom{A \lor B} \quad \lor \quad \neg D \lor \neg E \\
\neg B \lor D \lor E
\end{align*}
\]

very effective for Horn clauses (Clines)
Linear Resolution (for Horn Clauses)

a la 'Prolog'

?- A, B
A <= C
C <= B

\[ \text{Query: } \quad \text{Prove } A \land B \]

\[ \text{Program: } \quad A \lor C \]

\[ \text{C \lor B} \]

\[ \text{B} \]

\[ \downarrow \text{Backward chaining} \]

?- C, B
?- B, B
?- B

\[ \text{?-} \]
Unifiers

\[ P(f(x), y) \quad P(z, z) \]

\[ \sigma \]

Unifier but not most general one

\[ P(f(a), f(a)) \] Common instance

\[ f = \sigma \circ \sigma \]
Lifting Completeness of FO-Resolution

\[ N \text{ unsatisfiable} \rightarrow N + 1 \quad (13) \]

\[ \text{Herbrand} \]
\[ \text{Theorem} \]

\[ N' \text{ unsatisfiable complete set of ground instances} \]

\[ \text{Prop. Resolution} \]

---

\[ \frac{\text{Lifting}}{CN} \]

"Liftd" \[ C_1, \ldots, C_k \]

\[ \rightarrow \]

\[ 0 \]

\[ \vdash \]

\[ \downarrow \]

\[ \text{Given:} \quad C_1', \ldots, C_k' \]

\[ \rightarrow \]

\[ = N' \]

\[ \frac{\text{By FO-Resolution}}{3} \]

\[ C_{k+1}, C_{k+2}, \ldots, C_n (=1) \]

\[ \bigtriangleup \]

\[ \downarrow \]

\[ \text{To show: Prop. Resolution steps (2)} \]

\[ \text{Simulated by} \quad \text{instances of} \]

\[ \text{FO-Resolution steps (3)} \]

\[ \text{with more general clauses} \]
Equality (1)

"=" - Predicate with special meaning.
E.g. \( P(\bar{a}) \land a=b \land \neg P(\bar{b}) \land P(\bar{c}) \)

is \( \neg \) unsatisfiable.

How to prove that?

1. Add "Equality axioms" (with each quantifier minimal):

\[
\begin{align*}
ex & = x \quad \text{(Ref)} \\
ex = y & \rightarrow y = x \quad \text{(Sym)} \\
ex = y \land y = z & \rightarrow x = z \quad \text{(Trans)}
\end{align*}
\]

Commutativity

\( x = y \rightarrow f(\ldots, x, \ldots) = f(\ldots, y, \ldots) \) for each \( f \) function.
E.g., \( x = y \rightarrow f(\bar{z}, x) = f(\bar{z}, y) \) symbol.

Substitution

\( x = y \land P(\ldots x \ldots) \rightarrow P(\ldots y \ldots) \) for each predicate symbol \( P \).

Problem: Search space explodes.
Equality (2)

2. Para-modal calculus specific inference rules

- Para-modal calculus (needs both)

\[ S = \vdash v C \quad L[S'] \vdash D \]

\[ \exists \{ \neg \neg [v] \vdash C \otimes v D \} \]

Where \( S = \text{msg}(s, s') \)

Example:

\[ f(x, a) = x \lor P(x) \quad Q(f, y) \lor R(x) \]

\[ S = \{ x \rightarrow b \quad y \rightarrow a \} \]

\[ Q(b) \lor R(a) \]

Important improvements

- \( S' \) is not a variable
- \( s \cup \Delta \vdash \) (for some \( \Delta \) well-founded term ordering)

- add clause \( x = x \) (handless)
Most important: “subsumption”

Def: $C$ is a clause $C_0$ \textit{subsumes} $D$ if $C_0 \subseteq D$

E.g. $P \lor Q \not\subseteq \text{subsume}$

\[
P_a \lor Q_b \lor R_c
\]

Subsumed clauses can (usually) be deleted
First-Gen Inference Rule

on slide: \[
\begin{align*}
-\neg P(a) & \quad P(x) \lor Q(x) \quad \text{mgu}(x \rightarrow a) \\
-\neg P(a) & \quad P(a) \lor Q(a)
\end{align*}
\]

general \[
\begin{align*}
(C \lor A) & \quad D \lor \neg B \\
(C \lor A) \lor (D \lor \neg B) \\
\end{align*}
\]
\[\sigma = \text{mgu}(A, B)\]

Compare with Resolution!

Differences?
Nelson-Oppen congruence closure

To decide \( \Sigma \)-satisfiability of a set of ground unit (dis)equations

Example

1. \( f(a, b) = a \)
2. \( f(f(a, b), b) = c \)
3. \( g(a) \neq g(c) \)

No avoids building the full (infinite!) congruence closure:

1. Build term graph of input terms
2. Add congruences for input equations
   1. \( f(a, b) = a \)
   2. \( f(f(a, b), b) = c \)
3. Close under congruence:
   Add:
   Between terms with equal congruent subterms
   3. \( f(a, b) = a \)
   4. \( f(f(a, b), b) = c \)
4. Check if \( s \) and \( t \) are congruent for some \( s \neq t \) (yes)
model $f, g, h$