Resolution Derivation

- A clause \( C \) is derivable from a set of clauses \( \Delta \) if there is a sequence of clauses \( \Psi_1, \ldots, \Psi_k \) terminating in \( C \) such that:
  1. \( \Psi_i \in \Delta \), or
  2. \( \Psi_i \) is resolvent of some \( \Psi_j \) and \( \Psi_k \) such that \( j < i \land k < i \)

- Example: Consider clauses
  \[ \Delta = \{ \text{happy}(x), \text{sad}(x) \}, \{ \neg \text{sad}(y) \} \]

- Here, \( \{ \text{happy}(x) \} \) is derivable from \( \Delta \)

- If a clause \( C \) is derivable from \( \Delta \), we write \( \Delta \vdash C \)

Resolution Refutation

- The derivation of the empty clause from a set of clauses \( \Delta \) is called resolution refutation of \( \Delta \)

- Consider set of clauses \( \Delta \):
  \[ \{ \text{happy}(x), \text{sad}(x) \}, \{ \neg \text{sad}(y) \}, \{ \neg \text{happy}(	ext{mother}(joe)) \} \]

- Resolution refutation of \( \Delta \):
  \[
  \begin{align*}
  \{ \text{happy}(x), \text{sad}(x) \} & \quad \{ \neg \text{sad}(y) \} & \quad \{ \neg \text{happy}(	ext{mother}(joe)) \} \\
  \{ \text{happy}(x) \} & \quad \{ \neg \text{happy}(\text{mother}(joe)) \} & \quad \{ \} \\
  \end{align*}
  \]

Refutational Soundness and Completeness

- Theorem: Resolution is sound, i.e., if \( \Delta \vdash C \), then \( \Delta \models C \)

- Corollary: If there is a resolution refutation of \( \Delta \), \( \Delta \) is indeed unsatisfiable

- In other words, we cannot conclude a satisfiable formula is unsatisfiable using resolution

- Resolution with implicit factorization is also complete, i.e., if \( \Delta \models C \), then \( \Delta \vdash C \)

- Corollary: If \( F \) is unsatisfiable, then there exists a resolution refutation of \( F \) using only resolution with factorization.

- This is called the refutational completeness of resolution.

Validity Proofs using Resolution

- How to prove validity FOL formula using resolution?

- Use duality of validity and unsatisfiability:
  \( F \) is valid iff \( \neg F \) is unsatisfiable

- We will use resolution to show \( \neg F \) is unsatisfiable.

- First, convert \( \neg F \) to clausal form \( C \).

- If there is a resolution refutation of \( C \), then, by soundness, \( F \) is valid.
Example

- Everybody loves somebody. Everybody loves a lover. Prove that everybody loves everybody.
- First sentence in FOL:
- Second sentence in FOL:
- Goal in FOL:
- Thus, want to prove validity of:

\[(\forall x.\exists y.\text{loves}(x,y) \land \forall u.\forall w.((\exists v.\text{loves}(u,v)) \rightarrow \text{loves}(w,u))) \land \exists z.\exists t.\neg\text{loves}(z,t)\]

▶ Now, move quantifiers into scope:

\[\forall x.\exists y.\exists z.\exists t.\neg\text{loves}(x,y) \land \forall u.\forall w.((\exists v.\text{loves}(u,v)) \rightarrow \text{loves}(w,u)) \land \neg\text{loves}(z,t)\]

Example, cont.

\[\neg(\forall x.\exists y.\text{loves}(x,y) \land \forall u.\forall w.((\exists v.\text{loves}(u,v)) \lor \text{loves}(w,u))) \lor \forall z.\forall t.\text{loves}(z,t)\]

▶ Push innermost negation in:

\[\neg(\forall x.\exists y.\text{loves}(x,y) \land \forall u.\forall w.\neg(\text{loves}(u,v) \lor \text{loves}(w,u))) \lor \forall z.\forall t.\text{loves}(z,t)\]

▶ Push outermost negation in:

\[\neg(\neg(\forall x.\exists y.\text{loves}(x,y) \land \forall u.\forall w.\neg(\text{loves}(u,v) \lor \text{loves}(w,u))) \lor \forall z.\forall t.\text{loves}(z,t))\]

Example, cont.

\[\forall u.\forall w.\forall z.\text{loves}(x,\text{lover}(z)) \land (\neg\text{loves}(u,v) \lor \text{loves}(w,u)) \land \neg\text{loves}(z,t)\]

▶ Now, move quantifiers to front. Restriction:

\[\exists z.\exists t.\forall x.\exists y.\forall u.\forall w.\text{loves}(x,y) \land (\neg\text{loves}(u,v) \lor \text{loves}(w,u)) \land \neg\text{loves}(z,t)\]

▶ Next, skolemize existentially quantified variables:

\[\forall u.\forall w.\forall z.\text{loves}(x,\text{lover}(z)) \land (\neg\text{loves}(u,v) \lor \text{loves}(w,u)) \land \neg\text{loves}(\text{joe},\text{jane})\]

Example, cont.

- Want to prove negation unsatisfiable:

\[\neg(\forall x.\exists y.\text{loves}(x,y) \land \forall u.\forall w.((\exists v.\text{loves}(u,v)) \rightarrow \text{loves}(w,u))) \lor \forall z.\forall t.\text{loves}(z,t)\]

▶ Convert to PNF: in NNF, quantifiers in front

\[\neg((\forall x.\exists y.\text{loves}(x,y) \land \forall u.\forall w.((\exists v.\text{loves}(u,v)) \lor \text{loves}(w,u))) \lor \forall z.\forall t.\text{loves}(z,t))\]

▶ Remove inner implication:

\[\neg((\forall x.\exists y.\text{loves}(x,y) \land \forall u.\forall w.((\exists v.\text{loves}(u,v)) \lor \text{loves}(w,u))) \lor \forall z.\forall t.\text{loves}(z,t))\]

▶ Remove outer implication:

\[\neg((\forall x.\exists y.\text{loves}(x,y) \land \forall u.\forall w.((\exists v.\text{loves}(u,v)) \lor \text{loves}(w,u))) \lor \forall z.\forall t.\text{loves}(z,t))\]
Example, cont.

- Finally, we can do resolution:
  \[ \{\text{loves}(x, \text{lover}(z))\} \]
  \[ \{\neg\text{loves}(u, v), \text{loves}(w, u)\} \]
  \[ \{\neg\text{loves}(\text{joe, jane})\} \]
  - Resolve first and second clauses. MGU:
  - Resolvent:
  - Resolve new clause with third clause.
  - Mgu:
  - Resolvent: {}  
  - Thus, we have proven the formula valid.

Example II, cont

\[ \exists y. \forall z. (\neg \text{p}(z, y) \lor \exists x. (\neg \text{p}(z, x) \lor \neg \text{p}(x, z)) \land \text{p}(z, y) \lor \exists w. (\text{p}(z, w) \land \text{p}(w, z))) \]

- Push negations in:
  \[ \exists y. \forall z. (\neg \text{p}(z, y) \lor \forall x. (\neg \text{p}(z, x) \land \neg \text{p}(x, z)) \land \text{p}(z, y) \lor \exists w. (\text{p}(z, w) \land \text{p}(w, z))) \]

- Rename quantified variables:
  \[ \exists y. \forall z. (\neg \text{p}(z, y) \lor \forall x. (\neg \text{p}(z, x) \land \neg \text{p}(x, z)) \land \text{p}(z, y) \lor \exists w. (\text{p}(z, w) \land \text{p}(w, z))) \]

Example II, cont.

\[ \forall z. \forall x. (\neg \text{p}(z, a) \land \neg \text{p}(z, x) \land \neg \text{p}(x, z)) \land (\text{p}(z, a) \land \text{p}(z, f(z)) \land \text{p}(f(z), z)) \]

- Drop quantifiers and convert to CNF:
  \[ (\neg \text{p}(z, a) \lor \neg \text{p}(z, x) \lor \neg \text{p}(x, z)) \land (\text{p}(z, a) \land \text{p}(z, f(z)) \land \text{p}(f(z), z)) \]

- In clausal form (with renamed variables):
  \[ C1: \{\neg \text{p}(z, a), \neg \text{p}(z, x), \neg \text{p}(x, z)\} \]
  \[ C2: \{\text{p}(y, a), \text{p}(y, f(y))\} \]
  \[ C3: \{\text{p}(w, a), \text{p}(f(w), w)\} \]

- Resolve \( C1 \) and \( C2 \) using factoring.
  - What is the MGU for \( p(z, a), p(z, x), p(x, z), p(y, a) \)?

- Resolvent:
Example II, cont.

\[
\begin{align*}
C_1 &: \{\neg p(z,a), \neg p(z,x), \neg p(x,z)\} \\
C_2 &: \{p(y,a), p(y,f(y))\} \\
C_3 &: \{p(w,a), p(f(w),w)\} \\
C_4 &: \{p(a,f(a))\} \\
\end{align*}
\]

- Now, resolve \(C_1\) and \(C_3\) (using factoring).
- What is the MGU for \(p(z,a), p(z,x), p(x,z), p(w,a)\)?
- Resolvent:

Example II, cont.

\[
\begin{align*}
C_1 &: \{\neg p(z,a), \neg p(z,x), \neg p(x,z)\} \\
C_2 &: \{p(y,a), p(y,f(y))\} \\
C_3 &: \{p(w,a), p(f(w),w)\} \\
C_4 &: \{p(a,f(a))\} \\
C_5 &: \{p(f(a),a)\} \\
\end{align*}
\]

- Resolve \(C_1\) and \(C_5\) (using factoring).
- What is the MGU of \(p(z,a), p(z,x)\) and \(p(f(a),a)\)?
- Resolvent:

Resolution and First-Order Theorem Provers

- Resolution (with factorization) forms the basis of most automated first-order theorem provers.
- However, to make relational refutation more efficient, there are typically three main improvements:
  - Ordered resolution
  - Removal of useless clauses (tautology elimination, identical clause elimination etc.)
  - Built-in reasoning about equality
- Won’t talk about these in this course...

Motivation for First-Order Theories

- First-order logic is very powerful and very general.
- But in many settings, we have a particular application in mind and do not need the full power of first order logic.
- For instance, instead of general predicates/functions, we might only need an equality predicate or arithmetic operations.
- Also, might want to disallow some interpretations that are allowed in first-order logic.

First-Order Theories

- First-order theories: Useful for formalizing and reasoning about particular application domains
  - e.g., involving integers, real numbers, lists, arrays, ...
- Advantage: By focusing on particular application domains, can give much more efficient, specialized decision procedures
### Signature and Axioms of First-Order Theory

- A first-order theory $T$ consists of:
  1. **Signature** $\Sigma_T$: set of constant, function, and predicate symbols
  2. **Axioms** $A_T$: A set of FOL sentences over $\Sigma_T$

- **$\Sigma_T$ formula**: Formula constructed from symbols of $\Sigma_T$ and variables, logical connectives, and quantifiers.

- **Example**: We could have a theory of heights $T_H$ with signature $\Sigma_H = \{taller\}$ and axiom:
  \[ \forall x, y. taller(x, y) \rightarrow \neg taller(y, x) \]
  - Is $\exists x, \forall z. taller(x, z) \wedge taller(y, w)$ legal $\Sigma_H$ formula? Yes
  - What about $\exists x, \forall z. taller(x, z) \wedge taller(joe, tom)$? No

### Models of $T$

- A structure $M = (U, I)$ is a model of theory $T$, or $T$-model, if $M \models A$ for every $A \in A_T$.

- **Example**: Consider structure consisting of universe $U = \{ A, B \}$ and interpretation $I(taller) : \{ (A, B), (B, A) \}$
  - Is this a model of $T$? No

- Now, consider same $U$ and interpretation $\{ A, B \}$. Is this a model? Yes

- Suppose our theory had another axiom:
  \[ \forall x, y, z. (taller(x, y) \land taller(y, z) \rightarrow taller(x, z)) \]
  - Consider $I(taller) : \{ (A, B), (B, C) \}$. Is $(U, I)$ a model? No

### Questions

Consider some first order theory $T$:

- If a formula is valid in FOL, is it also valid modulo $T$? Yes
- If a formula is valid modulo $T$, is it also valid in FOL? No
- **Counterexample**: This formula is valid in “theory of heights”:
  \[ \neg taller(x, x) \]

### Equivalence Modulo $T$

- Two formulas $F_1$ and $F_2$ are equivalent modulo theory $T$ if for every $T$-model $M$ and for every variable assignment $\sigma$:
  \[ M, \sigma \models F_1 \iff M, \sigma \models F_2 \]

- Another way of stating equivalence of $F_1$ and $F_2$ modulo $T$:
  \[ T \models F_1 \leftrightarrow F_2 \]

- **Example**: Consider a theory $T_\rho$ with predicate symbol $=\!=$ and suppose $A_T$ gives the intended meaning of equality to $=\!=$.
  - Are $x = y$ and $y = x$ equivalent modulo $T_\rho$? Yes
  - Are they equivalent according to FOL semantics? No
  - Falsifying interpretation: $U = \{ \Box, \Diamond \}, I(=\!=) : \{ (\Diamond, \Box) \}$

### Satisfiability and Validity Modulo $T$

- Formula $F$ is satisfiable modulo $T$ if there exists a $T$-model $M$ and variable assignment $\sigma$ such that $M, \sigma \models F$

- Formula $F$ is valid modulo $T$ if for all $T$-models $M$ and variable assignments $\sigma$, $M, \sigma \models F$

- **Question**: How is validity modulo $T$ different from FOL-validity?

- **Answer**: Disregards all structures that do not satisfy theory axioms.

- If a formula $F$ is valid modulo theory $T$, we will write $T \models F$.

- Theory $T$ consists of all sentences that are valid in $T$. 

### Axioms of First-Order Theory

- The axioms $A_T$ provide the meaning of symbols in $\Sigma_T$.

- **Example**: In our theory of heights, axioms define meaning of predicate $taller$
  - Specifically, axioms ensure that some interpretations legal in standard FOL are not legal in $T$

- **Example**: Consider relation constant $taller$, and $U = \{ A, B, C \}$
  - In FOL, possible interpretation: $I(taller) : \{ (A, B), (B, A) \}$

- In our theory of heights, this interpretation is not legal b/c does not satisfy axioms
A theory $T$ is **complete** if for every sentence $F$, if $T$ entails $F$ or its negation:

$$T \models F \text{ or } T \models \neg F$$

**Question**: In first-order logic, for every closed formula $F$, is either $F$ or $\neg F$ valid?

**Answer**: No! Consider $p(a)$: Neither $p(a)$ nor $\neg p(a)$ is valid.