Automated Theorem Proving

Scott Sanner, Guest Lecture
Topics in Automated Reasoning
Thursday, Jan. 19, 2006
Introduction

- **Def. Automated Theorem Proving:**
  
  *Proof of mathematical theorems by a computer program.*

- **Depending on underlying logic, task varies from trivial to impossible:**
  - Simple description logic: Poly-time
  - Propositional logic: NP-Complete (3-SAT)
  - First-order logic w/ arithmetic: Impossible
Applications

• Proofs of Mathematical Conjectures
  – Graph theory: Four color theorem
  – Boolean algebra: Robbins conjecture

• Hardware and Software Verification
  – Verification: Arithmetic circuits
  – Program correctness: Invariants, safety

• Query Answering
  – Build domain-specific knowledge bases, use theorem proving to answer queries
Basic Task Structure

• **Given:**
  - Set of axioms (KB encoded as axioms)
  - Conjecture (assumptions + consequence)

• **Inference:**
  - Search through space of valid inferences

• **Output:**
  - Proof (if found, a sequence of steps deriving conjecture consequence from axioms and assumptions)
Many Logics / Many Theorem Proving Techniques

Focus on theorem proving for logics with a model-theoretic semantics (TBD)

• **Logics:**
  - Propositional, and first-order logic
  - Modal, temporal, and description logic

• **Theorem Proving Techniques:**
  - Resolution, tableaux, sequent, inverse
  - Best technique depends on logic and app.
Example of Propositional Logic Sequent Proof

- **Given:**
  - Axioms: None
  - Conjecture: \( A \lor \neg A \) ?

- **Inference:**
  - Gentzen Sequent Calculus

- **Direct Proof:**

  1. (I) \( A \vdash A \)
  2. (\neg R) \( \vdash \neg A, A \)
  3. (\lor R2) \( \vdash A \lor \neg A, A \)
  4. (PR) \( \vdash A, A \lor \neg A \)
  5. (\lor R1) \( \vdash A \lor \neg A, A \lor \neg A \)
  6. (CR) \( \vdash A \lor \neg A \)
Example of First-order Logic Resolution Proof

- **Given:**
  - **Axioms:**
    \[ \forall x \; \text{Man}(x) \Rightarrow \text{Mortal}(x) \]
    \[ \text{Man}(\text{Socrates}) \]
  - **Conjecture:**
    \[ \exists y \; \text{Mortal}(y) ? \]

- **Inference:**
  - **Refutation Resolution**

- **CNF:**
  \[ \neg \text{Man}(x) \lor \text{Mortal}(x) \]
  \[ \text{Man}(\text{Socrates}) \]
  \[ \neg \text{Mortal}(y) \quad \text{[Neg. conj.]} \]

- **Proof:**
  1. \[ \neg \text{Mortal}(y) \quad \text{[Neg. conj.]} \]
  2. \[ \neg \text{Man}(x) \lor \text{Mortal}(x) \quad \text{[Given]} \]
  3. \[ \text{Man}(\text{Socrates}) \quad \text{[Given]} \]
  4. \[ \text{Mortal}(\text{Socrates}) \quad \text{[Res. 2,3]} \]
  5. \[ \bot \quad \text{[Res. 1,4]} \]
  Contradiction \( \Rightarrow \) Conj. is true
Example of Description Logic

Tableaux Proof

- **Given:**
  - Axioms: None
  - Conjecture: ¬∃ Child. ¬Male ⇒ ∀ Child. Male ?

- **Inference:**
  - Tableaux

- **Proof:**
  
  Check unsatisfiability of
  ∃ Child. ¬Male ⊢ ∀ Child. Male

  x: ∃ Child. ¬Male ⊢ ∀ Child. Male
  x: ∀ Child. Male [ ⊢ -rule ]
  x: ∃ Child. ¬Male [ ⊢ -rule ]
  x: Child y [ ∃-rule ]
  y: ¬ Male [ ∃-rule ]
  y: Male [ ∀-rule ]
  <CLASH>

  Contradiction ⇒ Conj. is true
Lecture Outline

- **Common Definitions**
  - Soundness, completeness, decidability

- **Propositional and first-order logic**
  - Syntax and semantics
  - Tableaux theorem proving
  - Resolution theorem proving
    - Strategies, orderings, redundancy, saturation optimizations, & extensions

- **Modal, temporal, & description logics**
  - Quick overview of logics / TP techniques
Entailment vs. Truth

• For each logic and theorem proving approach, we’ll specify:
  – Syntax and semantics
  – Foundational axioms (if any)
  – Rules of inference

• Entailment vs. Truth
  – Let $KB$ be the conjunction of axioms
  – Let $F$ be a formula (possibly a conjecture)
  – We say $KB \vdash F$ (read: $KB$ entails $F$) if $F$ can be derived from $KB$ through rules of inference
  – We say $KB \models F$ (read: $KB$ models $F$) if semantics hold that $F$ is true whenever $KB$ is true
Model-theoretic semantics

- **Model-theoretic semantics for logics**
  - An interpretation is a truth assignment to atomic elements of a KB: \( I^{\langle C,D \rangle} = \{ \langle F,F \rangle, \langle F,T \rangle, \langle T,F \rangle, \langle T,T \rangle \} \)
  - A model of a formula is an interpretation where it is true: \( I^{\langle C,D \rangle} = \langle F,T \rangle \) models \( C \lor D \), \( C \Rightarrow D \), but not \( C \land D \)
  - Two properties of a formula \( F \) w.r.t. axioms of KB:
    - Validity: \( F \) is true in all models of KB
    - Satisfiability: \( F \) is true in \( \geq 1 \) model of KB

- **Think of truth in a set-theoretic manner**

\( KB \models C \)

Models of KB \( \subseteq \) Models of C
Soundness, Completeness, and Decidability

- **Two properties of ATP inference systems:**
  - **Soundness:** If $KB \vdash C$ then $KB \models C$
  - **Completeness:** If $KB \models C$ then $KB \vdash C$

- **For a given logic, an ATP decision procedure returns** true or false for $KB \vdash C$

- **For a logic, a sound and complete decision procedure** has one of following properties:
  - **Decidable:** Decision procedure guaranteed to terminate in finite time
  - **Semidecidable:** Decision procedure guaranteed to terminate for either true or false, but not both
  - **Undecidable:** No termination guarantee
Prop. Logic Syntax

• Propositional variables: \( p, \text{rain}, \text{sunny} \)
• Connectives: \( \Rightarrow \leftrightarrow \neg \land \lor \)
• Inductive definition of well-formed formula (wff):
  - Base: All propositional vars are wffs
  - Inductive 1: If \( A \) is a wff then \( \neg A \) is a wff
  - Inductive 2: If \( A \) and \( B \) are wffs then \( A \land B, A \lor B, A \Rightarrow B, A \Leftrightarrow B \) are wffs
• Examples:
  - \( \text{rain}, \text{rain} \Rightarrow \neg \text{sunny} \)
  - \( (\text{rain} \Rightarrow \neg \text{sunny}) \Leftrightarrow (\text{sunny} \Rightarrow \neg \text{rain}) \)
Prop. Logic Semantics

For a formula $F$, the truth $I(F)$ under interpretation $I$ is recursively defined:

- **Base:**
  - $F$ is prop var $A$ then $I(F) = \text{true}$ iff $I(A) = \text{true}$

- **Recursive:**
  - $F$ is $\neg C$ then $I(F) = \text{true}$ iff $I(C) = \text{false}$
  - $F$ is $C \land D$ then $I(F) = \text{true}$ iff $I(C) = \text{true}$ & $I(D) = \text{true}$
  - $F$ is $C \lor D$ then $I(F) = \text{true}$ iff $I(C) = \text{true}$ or $I(D) = \text{true}$
  - $F$ is $C \Rightarrow D$ then $I(F) = \text{true}$ iff $I(\neg C \lor D) = \text{true}$
  - $F$ is $C \Leftrightarrow D$ then $I(F) = \text{true}$ iff $I(C \Rightarrow D) = \text{true}$ & $I(D \Rightarrow C) = \text{true}$

- Truth defined recursively from ground up!
**CNF Normalization**

- Many prop. theorem proving techniques req. KB to be in clausal normal form (CNF):
  - Rewrite all $C \leftrightarrow D$ as $C \Rightarrow D \land D \Rightarrow C$
  - Rewrite all $C \Rightarrow D$ as $\neg C \lor D$
  - Push negation through connectives:
    - Rewrite $\neg(C \land D)$ as $\neg C \lor \neg D$
    - Rewrite $\neg(C \lor D)$ as $\neg C \land \neg D$
  - Rewrite double negation $\neg \neg C$ as $C$
  - Now NNF, to get CNF, distribute $\lor$ over $\land$:
    - Rewrite $(C \land D) \lor E$ as $(C \lor E) \land (D \lor E)$
- A clause is a disj. of literals (pos/neg vars)
- Can express KB as conj. of a set of clauses
Given KB with single formula:
\[ \neg (\neg \neg (\neg \neg \neg \neg \neg \text{rain} \lor \neg \neg \neg \neg \neg \text{wet}) \lor (\neg \neg \neg \neg \neg \text{inside} \land \neg \neg \neg \neg \neg \text{warm}) \]

Rewrite all \( C \Rightarrow D \) as \( \neg C \lor D \)
\[ \neg \neg \neg \neg \neg \neg \neg \text{rain} \lor \neg \neg \neg \neg \neg \text{wet} \lor (\neg \neg \neg \neg \neg \text{inside} \land \neg \neg \neg \neg \neg \text{warm}) \]

Push negation through connectives:
\[ (\neg \neg \neg \neg \neg \neg \neg \text{rain} \lor \neg \neg \neg \neg \neg \text{wet}) \lor (\neg \neg \neg \neg \neg \text{inside} \land \neg \neg \neg \neg \neg \text{warm}) \]

Rewrite double negation \( \neg \neg C \) as \( C \)
\[ (\neg \text{rain} \lor \text{wet}) \lor (\text{inside} \land \text{warm}) \]

Distribute \( \lor \) over \( \land \):
\[ (\neg \text{rain} \lor \text{wet} \lor \text{inside}) \land (\neg \text{rain} \lor \text{wet} \lor \text{warm}) \]

CNF KB:
\[ \{\neg \text{rain} \lor \text{wet} \lor \text{inside}, \neg \text{rain} \lor \text{wet} \lor \text{warm}\} \]
• **A ⇒ B** iff **A ∧ ¬B** is unsatisfiable

• Decision procedure for propositional logic is decidable, but **NP-complete** (reduction to 3-SAT)

• **State-of-the-art** prop. unsatisfiability methods are **DPLL-based**

• Many optimizations, more next week
Prop. Tableaux Methods

Given negated query $F$ (in NNF), use rules to recursively break down:

- $\alpha$-Rule: Given $A \land B$ add $A$ and $B$
- $\beta$-Rule: Given $A \lor B$ branch on $A$ and $B$
- $\langle$Clash$\rangle$: If $A$ and $\neg A$ occur on same branch
- Clash on all branches indicates unsat!

Note: Inverse method is inverse of tableaux - bottom up
Propositional Resolution

• **One rule:**

<table>
<thead>
<tr>
<th>Rule:</th>
<th>Example application:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \lor B \quad \neg B \lor C$</td>
<td>$\neg \text{precip} \lor \neg \text{freezing} \lor \text{snow} \quad \neg \text{snow} \lor \text{slippery}$</td>
</tr>
<tr>
<td>$A \lor C$</td>
<td>$\neg \text{precip} \lor \neg \text{freezing} \lor \text{slippery}$</td>
</tr>
</tbody>
</table>

• **Simple strategy is to make all possible resolution inferences**

• **Refutation resolution is sound and complete**
Resolution Strategies

Need strategies to restrict search:

- **Unit resolution:**
  - Only resolve with unit clauses
  - Complete for Horn KB
  - Intuition: Decrease clause size

- **Set of support:**
  - SOS starts with query clauses
  - Only resolve SOS clauses with non-SOS clauses and put resolvents in SOS
  - Intuition: KB should be satisfiable so refutation should derive from query

- **Input resolution:**
  - At each step resolve only with input (KB or query)
  - I.e., don’t resolve non-input clauses
  - Linear input: also allow ancestor $\Rightarrow$ complete
Ordering Strategies

• **Refutation of a clause** requires refutation of all literals

• **Enforce an ordering** on proposition elimination to restrict search
  - Example order: $p$ then $r$ then $q$
  - General idea behind Davis-Putnam (DP) & directional resolution (Dechter & Rish)

• **Effective, but does not work with all resolution strategies**, e.g. SOS + ordered resolution is incomplete
Prop. Inference Software

- Mainly DPLL SAT algorithms
  - *zChaff* – highly optimized & documented DPLL solver, source available
  - *siege* – best performing DPLL solver, source not available
  - *2clseq* – DPLL solver with constraint propagation (balance search / reasoning)

- For some applications: BDDs
  - BDDs maintain all possible models in a canonical data structure
  - CUDD ADD/BDD Package – very efficient
First-order logic

- Refer to objects and relations b/w them
- Propositional logic requires all relations to be propositionalized
  - Scott-at-home, Scott-at-work, Jim-at-subway, etc...
- Really want a compact relational form:
  - at(Scott, home), at(Scott, work), at(Jim, subway), etc...
- Then can use variables and quantify over all objects:
  - $\forall x \text{ person}(x) \Rightarrow \exists y \text{ at}(x,y) \land \text{ place}(y)$
First-order Logic Syntax

- **Terms** (technical definition is inductive b/c of fns)
  - Variables: \( w, x, y, z \)
  - Constants: \( a, b, c, d \)
  - Functions over terms: \( f(a), f(x, y), f(x, c, f(f(z))) \)

- **Predicates:** \( P(x), Q(f(x, y)), R(x, f(x, f(c, z), c)) \)

- **Connectives:** \( \Rightarrow \equiv \neg \land \lor \)

- **Quantifiers:** \( \forall \exists \)

- **Inductive wff definition:**
  - Same as prop. but with following modifications...
  - Base: All predicates over terms are wffs
  - Inductive: If A is a wff and x is a variable term then \( \forall x A \) & \( \exists x A \) are wffs
First-order Logic Semantics

• **Interpretation** $I = (\Delta^I, \cdot^I)$
  - $\Delta^I$ is a non-empty domain
  - $\cdot^I$ maps from predicate symbols $P$ of arity $n$ into a subset of $\times_{1 \ldots n} \Delta^I$ (where $P$ is true)

• **Example**
  - $\Delta^I$ is $\{\text{Scott, Jim}\}$
  - $\cdot^I$ maps $\text{at}(\cdot, \cdot)$ into $\{ \langle \text{Scott, loc(Scott)} \rangle, \langle \text{Jim, loc(Jim)} \rangle \}$
  - All other ground predicates are false in $I$, e.g. $\text{at(Scott, loc(Jim))}, \text{at(Scott, Scott)}$

• **NB:** FOL has $\infty$ interpretations/models!
Substitution and Unification

• **Substitution**
  - A substitution list $\theta$ is a list of variable-term pairs
    - e.g., $\theta = \{x/3, y/f(z)\}$
  - When $\theta$ is applied to an FOL formula, every free occurrence of a variable in the list is replaced with the given term
    - e.g. $(P(x,y) \land \exists x P(x,y))\theta = P(3,f(z)) \land \exists x P(x,f(z))$

• **Unification / Most General Unifier**
  - The unifier $UNIF(x,y)$ of two predicates/terms is a substitution that makes both arguments identical
    - e.g. $Unif( P(x,f(x)), P(y, f(f(z))) ) = \{x/f(1), y/f(1), z/1\}$
  - The most general unifier $MGU(x,y)$ is just that... all other unifiers can be obtained from the MGU by additional subst. (MGU exists for unifiable args)
    - e.g. $MGU( P(x,f(x)), P(y, f(f(z))) ) = \{x/f(z), y/f(z)\}$
Skolemization

- Skolemization is the process of getting rid of all $\exists$ quantifiers from a formula while preserving (un)satisfiability:
  - If $\exists x$ quantifier is the outermost quantifier, remove the $\exists$ quantifier and substitute a new constant for $x$
  - If $\exists x$ quantifier occurs inside of $\forall$ quantifiers, remove the $\exists$ quantifier and substitute a new function of all $\forall$ quantified variables for $x$

- Examples:
  - $\text{Skolemize}(\exists w \exists x \forall y \forall z P(w,x,y,z)) = \forall y \forall z P(c,d,y,z)$
  - $\text{Skolemize}(\forall w \exists x \forall y \exists z P(w,x,y,z)) = \forall w \forall y P(w,f(w),y,f(x,y))$
CNF Conversion

CNF conversion is the same as the propositional case up to NNF, then do:

- **Standardize apart variables** (all quantified variables should have different names)
  - e.g. $\forall x \ A(x) \land \exists x \ \neg A(x)$ becomes $\forall x \ A(x) \land \exists y \ \neg A(y)$
- **Skolemize formula**
  - e.g. $\forall x \ A(x) \land \exists y \ \neg A(y)$ becomes $\forall x \ A(x) \land \neg A(c)$
- **Drop universals**
  - e.g. $\forall x \ A(x) \land \neg A(c)$ becomes $A(x) \land \neg A(c)$
- **Distribute $\lor$ over $\land$**
First-order Theorem Proving

- **Tableaux methods**
  - Preferred for some types of reasoning and for subsets of FOL (guarded fragment, set theory)
  - Highly successful for description and modal logics which conform to guarded fragment of FOL

- **Resolution Methods**
  - Most successful technique for a variety of KBs
  - But... search space grows very quickly
  - Need a variety of optimizations in practice
    - strategies, ordering, redundancy elimination

- **FOL TP complete 😊, but semidecidable 😞**
  - Will return in finite time if formula entailed
  - May run forever if not entailed
First-order Tableaux

Given negated query $F$ (in NNF), use rules to recursively break down:

- $\alpha$-Rule, $\beta$-Rule: Same as for prop tableaux
- $\gamma$-Rule: Given $\forall x A(x)$ add $A(?v)$ for variable $?v$
- $\delta$-Rule: Given $\exists x A(x)$ add $A(f)$ for Skolem function $f$
- $\langle$Clash$\rangle$: If unifiable $A$ and $\neg A$ occur on same branch

$$\forall x A(x) \land \exists x \neg A(x) \lor \exists x,y \neg B(x,y) \land \forall x,y B(x,y)$$

$$\forall x A(x) \land \exists x \neg A(x) \quad \beta\text{-Rule}$$

$A(?y)$ $\alpha/\gamma$ -Rule
$\neg A(c)$ $\alpha/\delta$ -Rule
$\langle$Clash$\rangle$

$$\exists x,y \neg B(x,y) \land \forall x,y B(x,y) \quad \beta\text{-Rule}$$

$\neg B(c,d)$ $\alpha/\delta/\delta$ -Rule
$B(?y,?z)$ $\alpha/\gamma/\gamma$ -Rule
$\langle$Clash$\rangle$
First-order Resolution

**Binary Resolution Rule**

Rule:

\[
C \lor D \quad \neg E \lor F \\
\therefore (C \lor F) \theta = \text{MGU}(D, E)
\]

Example application:

\[
P(z) \lor Q(3) \lor R(y) \quad \neg Q(y) \\
P(3) \lor R(f(x))
\]

**Factoring Rule**

Rule:

\[
C \lor D \lor E \\
\therefore C \theta \lor E = \text{MGU}(C, D)
\]

Example application:

\[
P(z) \lor Q(3) \lor Q(z) \\
P(3) \lor Q(3)
\]
Example of First-order Logic Resolution Proof

- **Given:**
  - **Axioms:**
    \[ \forall x \text{ Man}(x) \Rightarrow \text{Mortal}(x) \]
    \[ \text{Man}(\text{Socrates}) \]
  - **Conjecture:**
    \[ \exists y \text{ Mortal}(y) \]

- **Inference:**
  - **Refutation Resolution**

- **CNF:**
  \[ \neg \text{Man}(x) \lor \text{Mortal}(x) \]
  \[ \text{Man}(\text{Socrates}) \]
  \[ \neg \text{Mortal}(y) \quad \text{[Neg. conj.]} \]

- **Proof:**
  1. \[ \neg \text{Mortal}(y) \quad \text{[Neg. conj.]} \]
  2. \[ \neg \text{Man}(x) \lor \text{Mortal}(x) \quad \text{[Given]} \]
  3. \[ \text{Man}(\text{Socrates}) \quad \text{[Given]} \]
  4. \[ \text{Mortal}(\text{Socrates}) \quad \text{[Res. 2,3]} \]
  5. \[ \bot \quad \text{[Res. 1,4]} \]

  **Contradiction \Rightarrow Conj. is true**
Importance of Factoring

• Without the factoring rule, binary resolution is incomplete

• For example, take the following refutable clause set:
  \[ \{ A(w) \lor A(z), \neg A(y) \lor \neg A(z) \} \]

• All binary resolutions yield clauses of the same form

• Clause set is only refutable if one of the clauses is first factored
Additional refinements of prop strategies yield goal-directed / bottom-up search:

- **SLD Resolution**
  - KB of definite clauses (i.e. Horn rules), e.g. Uncle(?x,?y) := Father(?x,?z) ∧ Brother(?x,?y)
  - Resolution backward chains from goal of rules
  - With negation-as-failure semantics, SLD-resolution is logic programming, i.e. Prolog

- **Negative and Positive Hyperresolution**
  - All negative (positive) literals in nucleus clause are *simultaneously* resolved with completely positive (negative) satellite clauses
  - Positive hyperres yields backward chaining
  - Negative hyperres yields forward chaining
Database-style Inference

- Naïve approaches to resolution perform one inference per step
- For SLD or neg. hyperres and KBs w/ large numbers of constants / functions, can store clause terms and perform DB-like res, e.g.
  - CNF KB = \{ R(a,b), R(b,a), R(b,c), R(c,b),
    ¬R(x,y) ∨ ¬R(y,z) ∨ R(x,z) \}
  - Use DB join/project during SLD or neg. hyperres:
    \[
    \begin{align*}
    R(x,y) & \quad \{ \langle a,b \rangle, \langle b,a \rangle, \langle b,c \rangle, \langle c,b \rangle \} \\
    \times & \\
    R(y,z) & \quad \{ \langle a,b \rangle, \langle b,a \rangle, \langle b,c \rangle, \langle c,b \rangle \} \\
    \Rightarrow & \\
    R(x,z) & \quad \{ \langle a,a \rangle, \langle a,c \rangle, \langle b,b \rangle, \langle c,c \rangle \}
    \end{align*}
    \]
- Can cache inferences for reuse (tabling)
- Huge improvement for instance-heavy KBs
Term Indexing

- **Term indexing** is another general technique for fast retrieval of sets of terms / clauses matching criteria

- **Common uses in modern theorem provers:**
  - Term \( q \) is unifiable with term \( t \), i.e., \( \exists \theta \text{ s.t. } q\theta = t\theta \)
  - Term \( t \) is an instance of \( q \), i.e., \( \exists \theta \text{ s.t. } q\theta = t \)
  - Term \( t \) is a generalization of \( q \), i.e., \( \exists \theta \text{ s.t. } q = t\theta \)
  - Clause \( q \) subsumes clause \( t \), i.e., \( \exists \theta \text{ s.t. } q\theta \subseteq t \)
  - Clause \( q \) is subsumed by clause \( t \), i.e., \( \exists \theta \text{ s.t. } t\theta \subseteq q \)

- **Techniques:** (Google for “term indexing”)
  - Path indexing
  - Code, context, & discrimination trees
Age-weight Ratio

- **During a resolution strategy, have two sets:**
  - **Active:** Set of active clauses for resolving with
  - **Frontier:** Candidate clauses to resolve with Active

- **Idea:** Store the frontier in two queues
  - **Age queue:** Standard FIFO queue
  - **Weight queue:** Priority queue where clause priority determined by heuristic measure:
    - Number of literals, number of terms, etc...

- **A:W ratio:** Choose A clauses from age queue for every W chosen from weight queue
  - Retains completeness of strategy if A is non-zero
    - I.e., fair b/c all clauses eventually selected
  - Can speed up inference by orders of magnitude!
Redundancy Control

- Redundancy of clauses is a huge problem in FOL resolution
  - For clauses $C$ & $D$, $C$ is redundant if $\exists \theta$ s.t. $C\theta \subseteq D$ as a multiset, a.k.a. $\theta$-subsumption
  - If true, $D$ is redundant and can be removed
    • Intuition: If $D$ used in a refutation, $C\theta$ could be substituted leading to even shorter refutation

- Two types of subsumption where $N$ is a new resolvent and $A \in$ Active:
  - Forward subsumption: $A \theta$-subsumes $N$, delete $N$
  - Backward subsumption: $N \theta$-subsumes $A$, delete $A$

- Forward/backward subsumption expensive but saves many redundant inferences
Saturation Theorem Proving

- **Given a set of clauses** $S$:
  - $S$ is saturated if all possible inferences from clauses in $S$ generate forward subsumed clauses
  - Thus, all new inferences can be deleted without sacrificing completeness
  - If $S$ does not contain the empty clause then $S$ is satisfiable

- Saturation implies no proof possible!
- Usually need ordering restrictions to reach saturation (if possible)
Simplification Orderings

For complete ordered resolution in FOL, must use term simplification orderings:

- **Well-founded (Noetherian):** If there is no infinitely decreasing chain of terms s.t. 
  \[ t_0 \succ t_1 \succ t_2 \succ \ldots \succ t_\infty \]
- **Monotonic:** If \( s \succ t \) then \( f[s] \succ f[t] \) (\( f[s] \) and \( f[t] \) are identical except for [term])
- **Stable under Subst.:** If \( s \succ t \) then \( s\theta \succ t\theta \)

**Examples:** (Google for following keywords)

- Knuth-Bendix ordering
- Lexicographic path ordering
Literal Ordering & Selection

• Can extend term ordering to literals $\succsim_{\text{lit}}$:
  – If literals equal but opposite sign, then negative literal $\succsim_{\text{lit}}$ positive literal
  – Otherwise, treat literals as terms (modulo sign) and literal ordering $\succsim_{\text{lit}}$ is just term ordering $\succsim$

• A selection function selects literals, and must adhere to following rules:
  – At least one literal must be selected
  – Either a negative literal is among the selection, or all maximal positive literals w.r.t. $\succsim_{\text{lit}}$ are selected

• Show selected literals by underscore
  – e.g., \{ $A \lor \neg B \lor \neg C$, $D \lor E \lor \neg F$, $\neg G \lor H \lor I$ \}
**Ordered Resolution w/ Selection**

- **Binary Ordered Res w/ Selection**
  
  **Rule:**
  
  \[
  \frac{C \lor D \quad \neg E \lor F}{(C \lor F)\theta} \quad \theta = \text{MGU}(D, E)
  \]

  **Example application:**
  
  \[
  \frac{P(3) \lor Q(f(x)) \lor R(y) \quad \neg Q(y)}{P(3) \lor R(f(x))}
  \]

- **Ordered Factoring w/ Selection**

  **Rule:**
  
  \[
  \frac{C \lor D \lor E}{C\theta \lor E} \quad \theta = \text{MGU}(C, D)
  \]

  **Example application:**
  
  \[
  \frac{P(z) \lor Q(3) \lor Q(z)}{P(3) \lor Q(3)}
  \]
Clause Orderings & Redundancy

• Must define **specialized redundancy criterion** for forward and backward subsumption / deletion when using **ordered resolution**:
  - Define bag (clause) extension of literal ordering:
    - \( \{x, y_1, \ldots, y_m\} \succ^\text{bag} \{x_1, \ldots, x_n, y_1, \ldots, y_m\} \) if \( \forall i \ x \succ^\text{lit} x_i \)
  - Can define redundancy w.r.t. \( \succ^\text{bag} \) ordering:
    - Clause \( C \) is redundant w.r.t. set of clauses \( S \), if \( \exists C_1, \ldots, C_n \in S, n \geq 0, \ s.t. \ \forall i \ C_i \succ^\text{bag} C \) and \( C_1, \ldots, C_n \vdash C \)
    - Under ordered res, even if \( C \theta\)-subsumes \( D \), \( D \) is not redundant (and can’t be deleted) unless \( C \prec^\text{bag} D \)

• **NB**: Search restrictions of ordered res far outweigh weakened notion of redundancy

• **Ordered res is effective saturation strategy!**
Equality

- A predicate w/ special interpretation
- Could axiomatize:
  - \( x=x \) (reflexive)
  - \( x=y \Rightarrow y=x \) (symmetric)
  - \( x=y \land y=z \Rightarrow x=z \) (transitive)
  - For each function \( f \):
    - \( x_1=y_1 \land \cdots \land x_n=y_n \Rightarrow f(x_1,\ldots,x_n)=f(y_1,\ldots,y_n) \)
  - For each predicate \( P \):
    - \( x_1=y_1 \land \cdots \land x_n=y_n \land P(x_1,\ldots,x_n) \Rightarrow P(y_1,\ldots,y_n) \)
- Too many axioms... better to reason about equality in inference rules
Inference
Rules for Equality

• **Demodulation (incomplete)**

Rule:

\[
\begin{align*}
  x = y & \quad L[z] \lor D \\
  L[y\theta] & \lor D
\end{align*}
\]

\[\theta = \text{MGU}(x,z)\]

Example application:

\[
\begin{align*}
  x = f(x) & \quad P(3) \lor Q \\
  P(f(3)) & \lor Q
\end{align*}
\]

\[\theta = \{x/3\}\]

• **Paramodulation (complete)**

Rule:

\[
\begin{align*}
  x = y \lor C & \quad L[z] \lor D \\
  (L[y] \lor C \lor D)\theta
\end{align*}
\]

\[\theta = \text{MGU}(x,z)\]

Example application:

\[
\begin{align*}
  x = f(x) \lor C & \quad P(3) \lor Q \\
  P(f(3)) \lor C \lor Q
\end{align*}
\]

\[\theta = \{x/3\}\]
Equational Programming

- Used extensively for algebraic group theory proofs
- All axioms and conjectures are unit equality predicates with arithmetic functions on the LHS and RHS, e.g.
  - $a*(x+y) = a*x+a*y$?
- In this case, associative-commutative (AC) unification (Stickel) important for efficiency, e.g.
  - $\text{MGU}(x+3*y*y, z*3*z+1) = \{x/1, y/z\}$
First-order theorem proving software

Many highly optimized first-order theorem proving implementations:

- **Vampire** (1\textsuperscript{st} place for many years in CADE TP competition)
- **Otter** (Foundation for modern TP, still very good, usually 2\textsuperscript{nd} place in CADE)
- **SPASS** (Specialized for sort reasoning)
- **SETHEO** (Connection tableaux calculus)
- **EQP** (Equational theorem proving system, proved Robbins conjecture)
First-order TP Progress

- Ever since the **1970s** I at various times investigated using automated theorem-proving systems. But it always seemed that *extensive human input*—typically from the creators of the system—was needed to make such systems actually find non-trivial proofs.

- In the late **1990s**, however, I decided to try the latest systems and was surprised to find that some of them could *routinely produce proofs hundreds of steps long* with little or no guidance. … the overall ability to do proofs—*at least in pure operator systems*—seemed vastly to exceed that of any human.

---Steven Wolfram, “A New Kind of Science”
On the other hand...

- **Success** of modern theorem provers relies largely on **heuristic tuning**
- **Input KBs** are analyzed for properties which **determine strategies** and various **parameters** of inference
- Still an **art as much as a science**, much room for more **principled tuning** of parameters, e.g.
  - Automatic partitioning of KBs to induce good literal orderings (McIlraith and Amir)
Gödel’s Incompleteness Theorem

• FOL inference is complete (Gödel)

• So what is Gödel’s incompleteness theorem (GIT) about?

• GIT: Inference in FOL with arithmetic (+, *, exp) is incomplete b/c set of axioms for arithmetic is not recursively enumerable.

• Read: Inference rules are sound and complete, but no way to generate all axioms required for arithmetic!
Modal Logic

• **Logic of knowledge and/or belief, e.g.**
  - English: Scott knows that you know that Scott knows this lecture is boring
  - Modal Logic $K_n$ (n agents): $K_{Scott} K_{you} K_{Scott}$ LIB

• **Possible worlds (Kripke) semantics**
  - Each modal operator $K_i$ corresponds to a set of possible interpretations (i.e., possible worlds)
  - Different axioms ($T,D,4,5,...$) correspond to relations b/w worlds, Axiom 4: $K_i \phi \Rightarrow K_i K_i \phi$
  - Semantics: $K_i \phi$ iff $\phi$ is true in all worlds agent $i$ considers possible according to axioms & KB

• **Postpone reasoning until DL...**
Temporal Logic

• A modal logic where the possible worlds are linked by time:
  – **LTL: Linear temporal logic**
    • World states evolve deterministically
    • State can involve action
  – **CTL: Computation tree logic**
    • World states can evolve non-deterministically

• Temporal operators specify conditions on world evolution
• Used for verification, safety checks
LTL Temporal Operators

- **G f: always f**
  - $f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f}$

- **F f: eventually f**
  - $f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f}$

- **X f: next state**
  - $f \xrightarrow{f} Xf \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f}$

- **f U r: until**
  - $f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f} f \xrightarrow{f}$

- **f R r: releases**
  - $r \xrightarrow{r} r \xrightarrow{r} r \xrightarrow{r} r_{r,f} \xrightarrow{r_{r,f}} r_{r,f} \xrightarrow{r_{r,f}} r_{r,f} \xrightarrow{r_{r,f}} r_{r,f}$
Temporal Logic Inference

- Because time evolves infinitely, propositional SAT methods won’t work for LTL/CTL verification (will branch infinitely)
- However, LTL/CTL inference is monotonic!
  - To check condition, start with set of all worlds
  - Evolve world one step, remove states not satisfying condition
  - Continue evolution until set does not change... this is set of all states for which condition holds
- For propositional temporal logic, number of worlds is finite $\Rightarrow$ termination $\Rightarrow$ decidable!
- BDD data structure used to compactly encode sets of worlds and evolve worlds.
**Description Logic**

- **A concept oriented logic:**

<table>
<thead>
<tr>
<th>English</th>
<th>FOL</th>
<th>DL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog with a Spot (DWS)</td>
<td>$\text{DWS}(x) \iff \text{Dog}(x) \land (\exists y \cdot \text{has}(x,y) \land \text{Spot}(y))$</td>
<td>$\text{DWS} \iff \text{Dog} \sqsubseteq \exists \text{has}.\text{Spot}$</td>
</tr>
<tr>
<td>Large Dog with a Dark Spot (LDWDS)</td>
<td>$\text{LDWDS}(x) \iff (\text{Dog}(x) \land \text{Large}(x)) \land (\exists y \cdot \text{has}(x,y) \land (\text{Spot}(y) \land \text{Dark}(y)))$</td>
<td>$\text{LDWDS} \iff \text{Dog} \sqsubseteq \text{Large} \sqsubseteq \exists \text{has}.(\text{Spot} \sqsubseteq \text{Dark})$</td>
</tr>
</tbody>
</table>

- **Guarded fragment subset of FOL**
Description Logic (DL)

Inference

• Natural correspondence between ALC DL and modal logic (Schild):
  – Modal propositions are concepts that hold in possible worlds $w$, e.g. lecture is boring: $\text{LIB}(w)$
  – Modal operators $K_i$ are DL roles that link possible worlds: $K_{\text{scott}}(w_1, w_2)$
  – If Scott knows that the lecture-is-boring then $\forall w_2 K_{\text{scott}}(w_1, w_2) \Rightarrow \text{LIB}(w_2)$ (w$_1$ is a free variable)
  – Or in DL notation $\forall K_{\text{scott}}\text{LIB}$

• Since decidable tableaux methods known for modal logics, these were imported into DL and later extended to expressive DLs

• Benefit of DL: Decidable subset of FOL that is ideal for conceptual ontology reasoning!
Example of Description Logic

Tableaux Proof

- **Given:**
  - **Axioms:** None
  - **Conjecture:** ¬∃ Child.¬Male ⇒ ∀ Child.Male?

- **Inference:**
  - **Tableaux**

- **Proof:**
  
  Check unsatisfiability of
  
  ∃Child.¬Male ∩ ∀ Child.Male

  \[ x: \exists \text{Child}.\neg \text{Male} \] ∩ ∀ \text{Child.Male} 

  \[ x: \forall \text{Child.Male} \] [ ∩ -rule ]

  \[ x: \exists \text{Child}.\neg \text{Male} \] [ ∩ -rule ]

  \[ x: \text{Child} \ y \] [ ∃-rule ]

  \[ y: \neg \text{Male} \] [ ∃-rule ]

  \[ y: \text{Male} \] [ ∀-rule ]

  \langle CLASH \rangle

  Contradiction ⇒ Conj. is true
DL Reasoner
Output (FaCT++)

Taxonomy encodes all ⇒ relations
Modal, Verification, and DL Inference Software

- **Modal logic**
  - MSPASS (converts modal formula to FOL)
  - By correspondence, also DL reasoners

- **Verification** (temporal and non-temporal)
  - PVS (interactive TP for HW/SW verification)
  - ALLOY (first-order HW/SW model checker)
  - NuSMV (BDD-based LTL/CTL HW/SW verif.)

- **DL Reasoning**
  - Classic (limited DL, poly-time inference)
  - Racer (expressive DL, highly optimized)
  - FaCT++ (very expr. DL, highly optimized)
Repositories of TP Problems

Many repositories of theorem proving knowledge bases:

- **TPTP**: Thousands of Problems for TPs
  - Algebraic group theory, geometry, set theory, topology, software verification, NLP KBs

- **SATLIB**: Library of Prop. SAT problems
  - Hardware verification, industrial planning problems, hard randomized problems

- **Open/ResearchCyc**: Public version of Cyc
  - Large common-sense repository expressed in higher-order logic

- **Semantic Web**: DL ontologies in OWL
  - The web is the limit!
Concluding Thoughts

- Many logics, inference techniques, and computational guarantees.
- Have to balance expressivity and computational tradeoffs with task-specific needs (Brachman & Levesque, 1985).
- Woods (1987): Don’t blame the tool!
  - A poor craftsman blames the tool when their efforts fail.
  - An experienced craftsman uses the right tool for the job.