15-780: Graduate AI

Lecture 3. FOL proofs

Geoff Gordon (this lecture)
Tuomas Sandholm
TAs Erik Zawadzki, Abe Othman
Admin
HW1

- Out today
- Due Tue, Feb. 1 (two weeks)
  - hand in hardcopy at beginning of class
- Covers propositional and FOL
- Don’t leave it to the last minute!
Collaboration policy

- **OK to discuss general strategies**
- **What you hand in must be your own work**
  - written with no access to notes from joint meetings, websites, etc.
- **You must acknowledge all significant discussions, relevant websites, etc., on your HW**
Late policy

- 5 late days to split across all HWs
  - these account for conference travel, holidays, illness, or any other reasons
- After late days, out of 70th %ile for next 24 hrs, 40th %ile for next 24, no credit thereafter (but still must turn in)
- Day = 24 hrs or part thereof, HWs due at 10:30AM
Office hours

- My office hours this week (usually 12–1 Thu) are canceled

- Email if you need to discuss something with me
Review
NP

- Decision problems
  - Reductions: A reduces to B means B at least as hard as A
    - Ex: k-coloring to SAT, SAT to CNF-SAT
    - Sometimes a practical tool
  - NP = reduces to SAT
  - NP-complete = both directions to SAT

- P = NP
Propositional logic

- Proof trees, proof by contradiction
- Inference rules (e.g., resolution)
- Soundness, completeness
- First nontrivial SAT algorithm
- Horn clauses, MAXSAT, nonmonotonic logic
FOL

- **Models**
  - objects, function tables, predicate tables

- **Compositional semantics**
  - object constants, functions, predicates
  - terms, atoms, literals, sentences
  - quantifiers, variables, free/bound, variable assignments
Proofs in FOL

- Skolemization, CNF
- Universal instantiation
- Substitution lists, unification
- MGU (unique up to renaming, exist efficient algorithms to find it)
Proofs in FOL
Quiz

- Can we unify
  
  \( \textit{knows}(\text{John}, x) \quad \textit{knows}(x, \text{Mary}) \)

- What about
  
  \( \textit{knows}(\text{John}, x) \quad \textit{knows}(y, \text{Mary}) \)
Quiz

- Can we unify
  \[ \text{knows}(John, x) \quad \text{knows}(x, Mary) \]
  \[ \text{No!} \]

- What about
  \[ \text{knows}(John, x) \quad \text{knows}(y, Mary) \]
  \[ x \rightarrow \text{Mary}, \ y \rightarrow \text{John} \]
Standardize apart

- But $\text{knows}(x, \text{Mary})$ is logically equivalent to $\text{knows}(y, \text{Mary})$!

- Moral: standardize apart before unifying
First-order resolution

- Given clauses \((α ∨ c), (¬d ∨ β)\), and a substitution list \(L\) unifying \(c\) and \(d\)
- Conclude \((α ∨ β) : L\)
- In fact, only ever need \(L\) to be MGU of \(c, d\)
Example

\[ \text{rains \( n \) outside}(x) \Rightarrow \text{wet}(x) \]

\[ \text{wet}(x) \Rightarrow \text{rusty}(x) \lor \text{rustproof}(x) \]

\[ \text{robot}(x) \equiv \neg \text{rustproof}(x) \]

\[ \text{rains} \]

\[ \text{guidebot}(\text{Rotby}) \]

\[ \text{guidebot}(x) \equiv \text{robot}(x) \lor \text{outside}(x) \]
\[
\text{rains n outside}(x) \Rightarrow \text{wet}(x)
\]
\[
\text{wet}(x) \Rightarrow \text{rusty}(x) \lor \text{rustproof}(x)
\]
\[
\text{robot}(x) \Rightarrow \neg \text{rustproof}(x)
\]

rains

\text{guidebot}(\text{Robby})

\text{guidebot}(x) \Rightarrow \text{robot}(x) \lor \text{outside}(x)
First-order factoring

- When removing redundant literals, we have the option of unifying them first
- Given clause \((a \lor b \lor \theta)\), substitution \(L\)
- If \(a : L\) and \(b : L\) are syntactically identical
- Then we can conclude \((a \lor \theta) : L\)
- Again \(L = \text{MGU}\) is enough
Completeness

- First-order resolution (w/ FO factoring) is sound and complete for FOL w/o equality (famous theorem due to Herbrand and Robinson)

- Unlike propositional case, may be infinitely many possible conclusions

- So, FO entailment is semidecidable (entailed statements are recursively enumerable)
Algorithm for FOL

- Put $KB \land \neg S$ in CNF
- Pick an application of resolution or factoring (using MGU) by some fair rule
  - standardize apart premises
- Add consequence to KB
- Repeat
Variations
Equality

- *Paramodulation is sound and complete for FOL+equality* (see RN)
- *Or, resolution + factoring + axiom schema*
Restricted semantics

- Only check one finite, propositional KB
  - NP-complete much better than RE
- Unique names: objects with different names are different (John $\neq$ Mary)
- Domain closure: objects without names given in KB don’t exist
- Known functions: only have to infer predicates
Uncertainty

- Same trick as before: many independent random choices by Nature, logical rules for their consequences
- Two new difficulties
  - ensuring satisfiability (not new, harder)
  - describing set of random choices
Markov logic

- Assume unique names, domain closure, known fns: only have to infer propositions
- Each FO statement now has a known set of ground instances
  - e.g., loves(x,y) ⇒ happy(x) has $n^2$ instances if there are $n$ people
- One random choice per rule instance: enforce w/p $p$ (KBs that violate the rule are $(1-p)$ times less likely)
Independent Choice Logic

- Generalizes Bayes nets, Markov logic, Prolog programs—incomparable to FOL
- Use only acyclic KBs (always feasible), minimal model (cf. nonmonotonicity)
- Assume all syntactically distinct terms are distinct (so we know what objects are in our model—perhaps infinitely many)
- Label some predicates as choices: values selected independently for each grounding
Inference under uncertainty

- Wide open topic: lots of recent work!
- We’ll cover only the special case of propositional inference under uncertainty
- The extension to FO is left as an exercise for the listener
Second order logic

- **SOL adds quantification over predicates**
- **E.g., principle of mathematical induction:**
  - $\forall P. P(0) \land (\forall x. P(x) \Rightarrow P(S(x)))$
  - $\Rightarrow \forall x. P(x)$
- **There is no sound and complete inference procedure for SOL** (Gödel’s famous incompleteness theorem)
Others

- Temporal and modal logics ("P(x) will be true at some time in the future," “John believes P(x)")
- Nonmonotonic FOL
- First-class functions (lambda operator, application)
- ...
  

Who? What? Where?
Wh-questions

- We’ve shown how to answer a question like “is Socrates mortal?”

- What if we have a question whose answer is not just yes/no, like “who killed JR?” or “where is my robot?”

- Simplest approach: prove $\exists x. \text{killed}(x, \text{JR})$, hope the proof is constructive
  - may not work even if constr. proof exists
Answer literals

- Instead of $\neg P(x)$, add $(\neg P(x) \lor \text{answer}(x))$
  - 	extit{answer} is a new predicate

- If there’s a proof of $P(\text{foo})$, can eliminate
  $\neg P(x)$ by resolution and unification, leaving $\text{answer}(x)$ with $x$ bound to $\text{foo}$
Example

\[ \text{kills (Jack, Cat)} \lor \text{kills (Curiosity, Cat)} \]
\[ \neg \text{kills (Jack, Cat)} \]
\[ \neg \text{kills (x, Cat)} \]
Example

\[\text{kills (Jack, Cat)} \lor \text{kills (Curiosity, Cat)}\]

\[\neg \text{kills (Jack, Cat)}\]

\[\neg \text{kills (x, Cat)}\]
Example

\text{kills (Jack, Cat) } \lor \text{ kills (Curiosity, Cat)}

\neg \text{kills (Jack, Cat)}

\neg \text{kills (x, Cat)} \lor \text{Answer (x)}
Instance Generation
Bounds on KB value

- If we find a model $M$ of $KB$, then $KB$ is satisfiable

- If $L$ is a substitution list, and if $(KB: L)$ is unsatisfiable, then $KB$ is unsatisfiable
  - e.g., $\text{mortal}(x) \rightarrow \text{mortal}(\text{uncle}(x))$
Bounds on KB value

- $KB_0 = KB$ w/ each syntactically distinct atom replaced by a different 0-arg proposition
  - $\text{likes}(x, \text{kittens}) \lor \neg\text{likes}(y, x) \rightarrow A \lor \neg B$
- $KB$ ground and $KB_0$ unsatisfiable $\Rightarrow$ $KB$ unsatisfiable
Propositionalizing

- Let $L$ be a ground substitution list
- Consider $KB' = (KB: L)_0$
  - $KB'$ unsatisfiable $\Rightarrow$ $KB$ unsatisfiable
  - $KB'$ is propositional
- Try to show contradiction by handing $KB'$ to a SAT solver: if $KB'$ unsatisfiable, done
- Which $L$?
Example

\[ \text{kills (Jack, Cat)} \lor \text{kills (Curiosity, Cat)} \]

\[ \neg \text{kills (Jack, Cat)} \]

\[ \neg \text{kills (x, Cat)} \]
Lifting

- Suppose \( KB' \) satisfiable by model \( M' \)
- *Try to lift* \( M' \) *to a model* \( M \) *of* \( KB \)
  - assign each atom in \( M \) the value of its corresponding proposition in \( M' \)
  - *break ties by* **specificity** *where possible*
  - *break any further ties arbitrarily*
Example

\[ \neg \text{kills}(\text{Jack, Cat}) \vee \text{kills}(\text{Curiosity, Cat}) \]

\[ \neg \text{kills}(\text{Jack, Cat}) \]

\[ \neg \text{kills}(x, \text{Cat}) \]

\[ \neg \text{kills}(\text{Curiosity, Cat}) \]

\[ \neg \text{kills}(\text{Foo, Cat}) \]

\[ M' \]
Discordant pairs

- Atoms $\text{kills}(x, \text{Cat}), \text{kills}((\text{Curiosity, Cat})$
  - each **tight** for its clause in $M'$
  - assigned opposite values in $M'$
  - unify: MGU is $x \rightarrow \text{Curiosity}$
- Such pairs of atoms are **discordant**
- They suggest useful ways to instantiate
Example

\[ \text{Kills (Jack, Cat)} \lor \text{Kills (Curiosity, Cat)} \]

\[ \neg \text{Kills (Jack, Cat)} \]

\[ \neg \text{Kills (x, Cat)} \]

\[ \neg \text{Kills (Curiosity, Cat)} \]
InstGen

- Propositionalize \( KB \rightarrow KB' \), run SAT solver
- If \( KB' \) unsatisfiable, done
- Else, get model \( M' \), lift to \( M \)
- If \( M \) satisfies \( KB \), done
- Else, pick a discordant pair according to a fair rule; use to instantiate clauses of \( KB \)
- Repeat
Soundness and completeness

- We’ve already argued soundness

- Completeness theorem: if $KB$ is unsatisfiable but $KB'$ is satisfiable, must exist a discordant pair wrt $M'$ which generates a new instantiation of a clause from $KB$—and, a finite sequence of such instantiations will find an unsatisfiable propositional formula
Agent Architectures
Situated agent

Agent → Perception ← Environment

Agent → Action → Environment
Inside the agent

perception → WM → reasoning → action

LTM
Inside the agent

- Raw signal
  - Signal to symbol
    - High level perception
      - WM
        - High level action
          - Executive
            - Low level action

- Reasoning
  - LTM
Knowledge Representation
Knowledge Representation

- *is the process of*
  - Identifying relevant objects, functions, and predicates
  - Encoding general background knowledge about domain (reusable)
  - Encoding specific problem instance
- Sometimes called knowledge engineering
Common themes

- RN identifies many common idioms and problems for knowledge representation
- Hierarchies, fluents, knowledge, belief, ...
- We’ll look at a couple
Taxonomies

- $isa(Mammal, Animal)$
- $disjoint(Animal, Vegetable)$
- $partition\{Animal, Vegetable, Mineral, Intangible\}, Everything$
Inheritance

- Transitive: \( \text{isa}(x, y) \land \text{isa}(y, z) \Rightarrow \text{isa}(x, z) \)
- Attach properties anywhere in hierarchy
  - \( \text{isa}(\text{Pigeon}, \text{Bird}) \)
  - \( \text{isa}(x, \text{Bird}) \Rightarrow \text{flies}(x) \)
  - \( \text{isa}(x, \text{Pigeon}) \Rightarrow \text{gray}(x) \)
- So, \( \text{isa}(\text{Tweety}, \text{Pigeon}) \) tells us Tweety is gray and flies
Physical composition

- `partOf(Wean4625, WeanHall)`
- `partOf(water37, water3)`

Note distinction between **mass** and **count** nouns: any `partOf` of a mass noun also `isa` that mass noun
Fluents

- Fluent = property that changes over time
  - at(Robot, Wean4623, 11AM)
- Actions change fluents
- Fluents chain together to form possible worlds
  - at(x, p, t) ∧ adj(p, q) ⇒ poss(go(x, p, q), t) ∧ at(x, q, result(go(x, p, q), t))
Frame problem

- Suppose we execute an unrelated action (e.g., `talk(Professor, FOL)`)
- Robot shouldn’t move:
  - if `at(Robot, Wean4623, t)`, want
    `at(Robot, Wean4623, result(talk(Professor, FOL)))`
- But we can’t prove it without adding appropriate rules to KB!
Frame problem

- The **frame problem** is that it’s a pain to list all of the things that don’t change when we execute an action

- Naive solution: **frame axioms**
  - for each fluent, list actions that can’t change fluent
  - **KB size**: $O(AF)$ for $A$ actions, $F$ fluents
Frame problem

- Better solution: *successor-state* axioms

- For each fluent, list actions that can change it (typically fewer): if go(x, p, q) is possible,
  \[
  \text{at}(x, q, \text{result}(a, t)) \iff \ a = \text{go}(x, p, q) \lor (\text{at}(x, q, t) \land a \neq \text{go}(x, q, z))
  \]

- Size $O(AE+F)$ if each action has $E$ effects
Debugging KB

- Sadly always necessary…
  - Severe bug: logical contradictions
  - Less severe: undesired conclusions
  - Least severe: missing conclusions

- First 2: trace back chain of reasoning until reason for failure is revealed
- Last: trace desired proof, find what’s missing
Examples
A simple data structure

- \((ABB) \equiv \text{cons}(A, \text{cons}(B, \text{cons}(B, \text{nil})))\)
- input\((x) \iff r(x, \text{nil})\)
- \(r(\text{cons}(x, y), z) \iff r(y, \text{cons}(x, z))\)
- \(r(\text{nil}, x) \iff \text{output}(x)\)
Caveat

- \( \text{input}(x) \iff r(x, \text{nil}) \)
- \( r(\text{cons}(x, y), z) \iff r(y, \text{cons}(x, z)) \)
- \( r(\text{nil}, x) \iff \text{output}(x) \)
A context-free grammar

- $S := NP \ VP$
- $NP := D \ Adjs \ N$
- $VP := Advs \ V \ PPs \ | \ Advs \ V \ DO \ PPs \ | \ Advs \ V \ IO \ DO \ PPs$
- $PP := Prep \ NP$
- $DO := NP$
- $IO := NP$
- $Adjs := Adj \ Adjs \ | \ {}$
- $Advs := Adv \ Advs \ | \ {}$
- $PPs := PP \ PPs \ | \ {}$
- $D := a \ | \ an \ | \ the \ | \ {}$
- $Adj := errant \ | \ atonal \ | \ squishy \ | \ piquant \ | \ desultory$
- $Adv := quickly \ | \ excruciatingly$
- $V := throws \ | \ explains \ | \ slithers$
- $Prep := to \ | \ with \ | \ underneath$
- $N := aardvark \ | \ avocado \ | \ accordion \ | \ professor \ | \ pandemonium$
A context-free grammar

- **S**: NP VP
- **NP**: D Adjs N
- **VP**: Advs V PPs | Advs V DO PPs | Advs V IO DO PPs
- **PP**: Prep NP
- **DO**: NP
- **IO**: NP
- **Adjs**: Adj Adjs | {}
- **Advs**: Adv Advs | {}
- **PPs**: PP PPs | {}
- **D**: a | an | the | {}
- **Adj**: errant | atonal | squishy | piquant | desultory
- **Adv**: quickly | excruciatingly
- **V**: throws | explains | slithers
- **Prep**: to | with | underneath
- **N**: aardvark | avocado | accordion | professor | pandemonium

- the errant professor explains the desultory avocado to the squishy aardvark
- a piquant accordion quickly excruciatingly slithers underneath the atonal pandemonium
Shift-reduce parser

input(x) ⇒ parse(x, nil)

parse(cons(x, y), z) ⇒ parse(y, cons(x, z))

parse(x, (VP NP . y)) ⇒ parse(x, (S . y))

parse(x, (N Adjs D . y)) ⇒ parse(x, (NP . y))

parse(x, y) ⇒ parse(x, (Adjs . y))

parse(x, (aardvark . y)) ⇒ parse(x, (N . y))

... parse(nil, (S)) ⇒ parsed
An example parse

- input((the professor slithers))
input(x) \land input(y) \Rightarrow (x = y)

NP \neq VP \land NP \neq S \land NP \neq \text{the} \land \text{avocado} \neq \text{aardvark} \land \text{avocado} \neq \text{the} \land \ldots

terminal(x) \iff x = \text{avocado} \lor x = \text{the} \lor \ldots

input(x) \iff \text{parse}(x, \text{nil})

parse(nil, (S)) \iff \text{parsed}
More careful (cont’d)

terminal(x) \Rightarrow

\[\text{parse(cons(x, y), z) } \Leftrightarrow \text{ parse(y, cons(x, z))}]\]

\[\text{parse(x, (aardvark . y)) } \lor \text{ parse(x, (avocado . y))} \lor \ldots] \Leftrightarrow \text{ parse(x, (N . y))}\]

\[\text{parse(x, y) } \lor \text{ parse(x, (Adjs Adj . y))} \Leftrightarrow \text{ parse(x, (Adjs . y))}\]

\ldots
Extensions

- Probabilistic CFG
- Context-sensitive features (e.g., coreference: John and Mary like to sail. His yacht is red, and hers is blue.)