Logic Intro

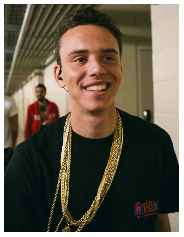
CompSci 370

Duke University

Ron Parr

Historical Perspective I

- Logic was one of the classical foundations of AI
- Dream: A Knowledge-Based agent
  - Tell the agent facts
  - Agent uses rules of inference to deduce consequences
  - Example: prolog
- Distinction between data and program
- Embodied in field of "Expert Systems"


Source: Wikipedia

Example: Minesweeper

- How do you play minesweeper?
- How would you program a machine to do it?
  - Hacking
  - Search/CSPs
  - Logic
- Logic approach
  - Tell the system of rules of minesweeper
  - System uses logic to make the best moves

What is logic, really?

- Syntax: Rules for constructing valid sentences
- Semantics: Relate syntax to the real world

Entailment

- Aim: Rule for generating (or testing) new sentences that are *necessarily* true
- The truth of sentence may depend upon the *interpretation* of the sentence

Interpretations

- An interpretation is a way of matching up objects in the universe with symbols in a sentence (or database).
- A sentence may be true in one interpretation, but false in another
- A *necessarily true* sentence is true in all interpretations = entailed by premises in our KB

Examples

- Premises (facts in our database or KB):
  - (X or Y)
  - Not X
  - Conclude: Y
- Premises
  - If P then Q
  - Q
  - Conclude: Nothing

Soundness & Completeness

- A (set of) rule(s) of inference is sound if it generates only sentences that are entailed by the knowledge base, i.e., only necessary truths
- A (set of) rule(s) of inference is complete if it can generate all necessary truths
- Can we have one w/o the other?

Historical Perspective II

- Things that are not true necessarily but still true are sometimes said to be “contingent,” “accidental,” or “synthetic,” truths.
- A deep understanding of this distinction evolved through thousands of years of philosophy and mathematics
- Arguably one of the most important intellectual accomplishments of mankind
  - Basis of mathematic proofs
  - Provides a rigorous procedure for verifying statements
  - **Foundation of rigorous thinking** – allows us to distinguish sound from unsound reasoning

Fun Exercise

- See dictionary of fallacies or one of many similar websites
  - http://www.ozarkia.net/bill/fallacies/index.html
  - https://en.wikipedia.org/wiki/List_of_fallacies
- Count how often you see these from politicians, e.g.,
  - No true Scottsman
  - Tu Quoque

- Make it a drinking game? (Don’t blame me if you get blitzed!)

Propositional Logic

- Propositional logic is the simplest logic
- All sentences are composed of
  - Atoms
  - Negation
  - Disjunction, conjunction (or, and)
  - Conditional, biconditionals
- Atoms can map to any *proposition* about the universe (depending upon the interpretation)

Checking Validity

- Classic method for checking validity: *truth table*
- Enumerate all possible values (t/f) of atomic elements of a sentence

$$\begin{array}{c} (P \vee H) \\ \neg H \\ \hline P \end{array}$$

Horizontal line separates premises from conclusion

- Enumerate all 4 (or more) combinations

Inference Rules

- Inference rules are (typically) sound methods of generating new sentences given a set of previous sentences
- Inference rules save us the trouble of generating truth tables all the time

Inference Rules I

- Modus Ponens

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

- And-Elimination

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

Inference Rules II

- And-Introduction

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

- Or-Introduction

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

Inference Rules III

- Double Negation Elimination

$$\frac{\neg\neg\alpha}{\alpha}$$

- Unit Resolution

$$\frac{\alpha \vee \beta, \neg\beta}{\alpha}$$

Resolution

$$\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma}$$

Resolution is perhaps the most important inference rule!

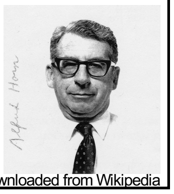
Why? Because it is sound and complete.

Complexity of Inference

- What is the complexity of exhaustively verifying the validity of a sentence with n literals (variables)?

$$2^n$$

- Special Case: Horn Logic
  - Horn clauses are disjunctions with at most one positive literal
  - Equivalent to $P_1 \wedge P_2 \wedge \ldots \wedge P_n \Rightarrow Q$



State dept. photo downloaded from Wikipedia

Remember De Morgan's Law?

- not(P and Q) = (not P) or (not Q)
- not(P or Q) = (not P) and (not Q)

Implications and Horn Clauses

- If P then Q
  - Same as: (not (P and (not Q))
  - Same as: (not P) or Q
  - …and this is horn!
- If (P1 and P2 and … Pn) then Q
  - Same as: (not ((P1 and P2 and … Pn) and (not Q))
  - Same as: not (P1 and P2 and … Pn) or Q
  - Same as: ((not P1) or (not P2) or … (not Pn) or Q)
  - …and this is horn!

Horn Clause Inference

- Horn clause inference is polynomial – Why?
  - Every sentence establishes exactly one new fact
  - Can add every possible new fact implied by our KB in n passes over our database
- What types of things are easy to represent with horn clauses?
  - Diagnostic rules
  - “Expert Systems”

Shortcomings of Horn Clauses

- Suppose you want to say, “If you have a runny nose and fever, then you have a cold *or* the flu.”
- If (runny_nose and fever) then (cold or flu)
- But this isn’t a horn clause:
  (not runny_nose) or (not fever) or (cold) or (flu)
- Does adding two separate horn clauses work?
  - (not runny_nose) or (not fever) or (cold)
  - (not runny_nose) or (not fever) or (flu)

Propositional Logic Conclusion

- Logic gives formal rules for reasoning
- Necessarily true = true in all interpretations
- Contrast with CSPs: Satisfiable = true in some, but not necessarily all interpretations
- Sound inference rules generate only necessary truths
- Resolution is a sound and complete inference rule
- Inference with a horn KB is poly time

Limitations of Propositional Logic

- Suppose you want to say: All humans are mortal
- For ~7B people, you would need ~7B propositions
- Suppose you want to stay that (at least) one person has perfect pitch
- You would need a disjunction of ~7B propositions

- There has to be a better way…

First Order Logic

- Propositional logic is very restrictive
  - Can't make global statements
  - Workarounds tends to have very large KBs
- First order logic is more expressive
  - Relations, quantification, functions
  - but… inference is trickier

Relations

- Assert relationships between objects
- Examples
  - Siblings(Luke, Leia)
  - Between(Canada, US, Mexico)
- Semantics
  - Object and predicate names are mnemonic only
  - Interpretation is imposed from outside
  - Often we imply the "expected" interpretation of predicates and objects with suggestive names

Functions

- Functions are special cases of relations
- Suppose $R(x_1,x_2,\ldots,x_n,y)$ is such that for every value of $x_1,x_2,\ldots,x_n$ there is a unique y
- Then $R(x_1,x_2,\ldots,x_n)$ can be used as a shorthand for y
  - Crossed(Right_leg_of(Ron), Left_leg_of(Ron))
- Remember that the object identified by a function depends upon the interpretation

Quantification

- For all objects in the world…

$$\forall x \text{happy}(x)$$

- For at least one object in the world…

$$\exists x \text{happy}(x)$$

Examples

- Everybody loves somebody

$$\forall X \exists Y: Loves(X, Y)$$

- Everybody loves everybody

$$\forall X, Y: Loves(X, Y)$$

- Everybody loves Raymond

$$\forall X: Loves(X, Raymond)$$

- Raymond loves everybody

$$\forall X: Loves(Raymond, X)$$

Equality

- Equality states that two objects are the same
  – Son_of(Barbara) = Ron
- Equality is a special relation that holds whenever two objects are the same
- We can imagine that every interpretation comes with its own identity relation
  – Identical(object27, object58)

Inference

- All rules of inference for propositional logic apply to first order logic
- We need extra rules to handle substitution for quantified variables

$\text{SUBST}(\{x/Harry, y/Sally\}, Loves(x,y)) = Loves(Harry, Sally)$

Inference Rules

- Universal Elimination

$$\frac{\forall v : \alpha(v)}{SUBST(\{v/g\}, \alpha(v))}$$

- How to read this:
  - We have a universally quantified variable v in α
  - Can substitute any constant g for v in α

Inference Rules

- Existential Elimination

$$\frac{\exists v : \alpha(v)}{\mathrm{SUBST}(\{v/k\}, \alpha(v))}$$

- How to read this:
  - We have a universally quantified variable v in α
  - Can substitute any new k* for v and α will still be true
  - *IMPORTANT: k must be a ***previously unused*** constant (*skolem* constant). Why is this OK?

Skolemization within Quantifiers

- Skolemizing w/in universal quantifier is tricky
- Everybody loves somebody

$$\forall x \exists y : loves(x, y)$$

- With Skolem constants, becomes:

$$\forall x : loves(x, object34752)$$

- Why is this wrong?
- Need to use skolem functions:

$$\forall x : loves(x, personlovedby(x))$$

Inference Rules

- Existential Introduction

$$\frac{\alpha(g)}{\text{SUBST}(\{v/g\}, \exists v : \alpha(v))}$$

- How to read this:
  - We know that the sentence α is true
  - Can substitute variable v for any constant g in α and (w/existential quantifier) and α will still be true
  - Why is this OK?

Generalized Modus Ponens Example

- If has_US_birth_certificate(X) then natural_US_citizen(X)
- has_US_birth_certificate(Obama)
- Conclude SUBST({Obama/X},natural_US_citizen(X))
- i.e., natural_US_citizen(Obama)

Generalized Modus Ponens

$$\text{SUBST}(\theta, p_i') = \text{SUBST}(\theta, p_i) \forall i$$

$$\frac{p_1', p_2', \ldots p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\text{SUBST}(\theta, q)}$$

- How to read this:
  - We have an implication which implies q
  - Any consistent substitution of variables on the LHS must yield a valid conclusion on the RHS

Unification

- Substitution is a non-trivial matter
- We need an algorithm called unify:

$$\text{Unify}(p, q) = \theta : \text{Subst}(\theta, p) = \text{Subst}(\theta, q)$$

- Important: Unification replaces variables:

$$\exists x \text{Loves}(John, x)$$

$$\exists x \text{Hates}(John, x)$$

- Are these the same x?

Unification Example

$\forall x Knows(John,x) \Rightarrow Loves(John,x)$

$Knows(John,Jane)$

$\forall y Knows(y,Leonid)$

$\forall y Knows(y,Mother(y))$

$\forall x Knows(x,Elizabeth)$

Note: All unquantified variables are assumed universal from here on.

Unify($Knows(John,x)$,$Knows(John,Jane)$) =	{X/Jane}
Unify($Knows(John,x)$,$Knows(y,Leonid)$) =	{Y/John, X/Leonid}
Unify($Knows(John,x)$,$Knows(y,Mother(y))$) =	{Y/John, X/Mother(John)}
Unify($Knows(John,x)$,$Knows(x,Elizabeth)$) =	{X1/John, X2/Elizabeth}

Most General Unifier

- Unify(Knows(John,x),Knows(y,z))
  - {y/John,x/z}
  - {y/John,x/z,w/Freda}
  - {y/John,x/John,z/John)
- When in doubt, we should always return the most general unifier (MGU)
  - MGU makes least commitment about binding variables to constants

Proof Procedures

- Suppose we have a knowledge base: KB
- We want to prove q
- Forward Chaining
  - Like search: Keep proving new things and adding them to the KB until we are able to prove q
- Backward Chaining
  - Like backward search in planning
  - Find $p_1 \ldots p_n$ s.t. knowing $p_1 \ldots p_n$ would prove q
  - Recursively try to prove $p_1 \ldots p_n$

Forward Chaining Example

$\forall x Knows(John,x) \Rightarrow Loves(John,x)$

$Knows(John,Jane)$

$\forall y Knows(y,Leonid)$

$\forall y Knows(y,Mother(y))$

$\forall x Knows(x,Elizabeth)$

Forward Chaining Example

$\forall x Knows(John,x) \Rightarrow Loves(John,x)$

$Knows(John,Jane)$

$\forall y Knows(y,Leonid)$

$\forall y Knows(y,Mother(y))$

$\forall x Knows(x,Elizabeth)$

Loves(John,Jane)
Knows(John, Leonid)
Loves(John,Jane)
Knows(John,Mother(John))
Loves(John,Mother(John))
Knows(John, Elizabeth)
Loves(John, Elizabeth)

A Note About Forward Chaining

- As presented, forward chaining seems undirected
- Can view forward chaining as a search problem
- Can apply heuristics to guide this search
- If you're trying to prove that Barack Obama is a natural born citizen, should you should start by proving that square127 is also a rectangle???
- Interesting AI history: AM/Eurisko controversy
  - Doug Lenat introduced what was essentially a forward chaining system for coming up with interesting math concepts
  - Claimed to (re)discover interesting concepts using only simple heuristics
  - Methodology sharply criticized due to opacity
    (see Ritchie and Hanna 1984 and response from Lenat and Brown 1984)

Backward Chaining Example

$\forall x Knows(John,x) \Rightarrow Loves(John,x)$
$Knows(John,Jane)$
$\forall y Knows(y,Leonid)$
$\forall y Knows(y,Mother(y))$
$\forall x Knows(x,Elizabeth)$

- Goal: Loves(John, Jane)?
- Subgoal: Knows(John, Jane)

Completeness

$\forall X : P(X) \Rightarrow Q(X)$
$\forall X : \neg P(X) \Rightarrow R(X)$
$\forall X : Q(X) \Rightarrow S(X)$
$\forall X : R(X) \Rightarrow S(X)$
$S(a)$???

- Problem: Generalized Modus Ponens not complete
- Forward/Backward chaining rely upon generalized MP
- Goal: Sound **and** complete procedure for first order logic

Generalized Resolution

$$\theta = \text{Unify}(p_j, \neg q_k)$$

$$\frac{(p_1 \vee \ldots p_j \ldots \vee p_m), (q_1 \vee \ldots q_k \ldots \vee q_n)}{\text{SUBST}(\theta, (p_1 \vee \ldots p_{j-1} \vee p_{j+1} \ldots \vee p_m \vee q_1 \vee \ldots q_{k-1} \vee q_{k+1} \ldots \vee q_n))}$$

- If the same term appears in both positive and negative form in two disjunctions, they cancel out when disjunctions are combined

Generalized Resolution Example

- Given:
  - (¬P(X) ∨ Q(X))
  - (P(X) ∨ R(X))
  - (¬ Q(X) ∨ S(X))
  - (¬ R(X) ∨ S(X))
- Can we conclude: S(a)?

Resolution Tree

$$\begin{tikzcd}
(\neg P(X) \vee Q(X)) \arrow[dr] & & (P(X) \vee R(X)) \arrow[dl] & & \\
& (R(X) \vee Q(X)) \arrow[dr] & & (\neg R(X) \vee S(X)) \arrow[dl] & \\
& & (Q(X) \vee S(X)) \arrow[dr] & & (\neg Q(X) \vee S(X)) \arrow[dl, "\{X/a\}"'] \\
& & & S(a) &
\end{tikzcd}$$

Deductions in Green
Substitutions in Blue

Resolution Properties

- Proof by refutation (asserting negation and resolving to nil) is sound and complete (NB: We did not do this in the first example)
- Resolution is not complete in a generative sense, only in a testing sense
- This is only part of the job
- To use resolution, we must convert everything to a canonical form, i.e., all sentences must be disjunctions with only implicit universal quantification and existential quantification replaced with skolemization

Same Example Using Refutation

(Not Required in This Case)

$$\begin{tikzcd}
(\neg P(X) \vee Q(X)) \arrow[dr] & & (P(X) \vee R(X)) \arrow[dl] & & & \\
& (R(X) \vee Q(X)) \arrow[dr] & & (\neg R(X) \vee S(X)) \arrow[dl] & & \\
& & (Q(X) \vee S(X)) \arrow[dr] & & (\neg Q(X) \vee S(X)) \arrow[dl] & \\
& & & S(X) \arrow[dr] & \{X/a\} & \neg S(a) \arrow[dl] \\
& & & & \{\} &
\end{tikzcd}$$

Deductions in Green
Substitutions in Blue
Asserted for contradiction in Red

A Note About Contradictions

- Proof by contradiction is very powerful
- If assuming P produces a contradiction, then
  P must be false
- Leaving contradictions in KB is disastrous
- It’s true in sci fi: https://youtu.be/Mw3zzMWOIvk

Contradictions Allow Us To Prove Anything

- Given:
  - happy(x)
  - ¬happy(x)
- Prove: flies(Ron)
- Two paths
  - Or introduction + resolution
  - Resolution with refutation

Or Introduction

- happy(X) ⊢ (happy(X) ∨ flies(Ron))

↑

Or Introduction

- (happy(X) ∨ flies(Ron)), ¬happy(X) ⊢ flies(Ron))

↑

Resolution

Resolution With Refutation

$$\begin{tikzcd}
\neg \text{flies(Ron)} & \text{happy(X)} \arrow[dr] & & \neg \text{happy(X)} \arrow[dl] \\
& & \{\} &
\end{tikzcd}$$

Observe that what we're trying to prove isn't even used

Canonical Form

- Eliminate Implications
- Move negation inwards (using e.g. DeMorgan's law)
- Standardize (apart) variables
- Move quantifiers Left
- Skolemize
- Drop universal quantifiers, e.g., (A ^ B) v (C ^ D)
- Distribute AND over OR
- Flatten nested conjunctions and disjunctions

Computational Properties

- We can enumerate the set of all proofs
- We can check if a proof is valid
- First order logic is complete (Gödel’s **completeness** result)

- What if no valid proof exists?
- Inference in first order logic is *semi-decidable*
- Compare with halting problem (halting problem is semi-decidable)

- As with propositional logic, horn clauses are a useful special case, though not as big of a win computationally - more about this when we discuss prolog

Gödel’s **Incompleteness** Result

- Gödel’s **incompleteness** result is, perhaps, better known
- Incompleteness applies to logical/mathematical systems rich enough to contain numbers and math
  - Need a way of enumerating all valid proofs within the system
  - Need a way of referring to proofs by number
- Construct a Gödel sentence:
  - S: For all i, i is not the number of a proof of the sentence j
  - (Equivalent to saying, there does not exist a proof of sentence j)
  - Suppose sentence S is sentence j
    - If S is false, then we have a contradiction
    - If S is true, then we can’t have a proof of it

Diagonalization

- Incompleteness can be seen as an instance of diagonalization:
  - Define a set
    (Rationals, TMs that halt, theorems that are provable)
  - Use rules of the system to create an impossible object

- Example: Proof that reals are not enumerable (i.e., not countable and therefore larger than the rationals)

Countability of Rationals

$$X = \frac{n_0 \times 2^0 + n_1 \times 2^1 + n_2 \times 2^2 \ldots}{d_0 \times 2^0 + d_1 \times 2^1 + d_2 \times 2^2 \ldots}$$

Label	n_0	d_0	n_1	d_1	…
0	0	0	0	0	…
1	1	0	0	0	…
2	0	1	0	0	…
3	1	1	0	0	…
…	…	…	…	…	…

Uncountability of Reals

- Given:

Label	n_0	d_0	n_1	d_1	…
0	0	0	0	0	…
1	1	0	0	0	…
2	0	1	0	0	…
3	1	1	0	0	…
…	…	…	…	…	…

- Construct by inverting along the diagonal:

Label	n_0	d_0	n_1	d_1	…
0	1	0	0	0	…
1	1	1	0	0	…
2	0	1	1	0	…
3	1	1	0	1	…
…	…	…	…	…	…

Implications of all this

- Sophomoric interpretation: AI is impossible/implausible because there will always be true things that cannot be discovered by logic

- A bit of reality:
  - Incompleteness talks about a system's ability to prove things about itself
  - For any given system, it may be possible to prove things by talking about the system in a more expressive language
  - Relationship of the unprovable to intelligence is murky at best: Are the things you can't justify the things that make you intelligent?
  - Not clear that anything interesting is unprovable in a practical sense (though plenty of interesting things remain unproven)

First Order Logic Conclusions

- First order logic adds relations and quantification to predicate logic
- Inference in first order logic is, essentially, a generalization of inference in propositional logic
  - Resolution is sound and complete
  - Use of resolution requires:
    - Conversion to canonical form
    - Proof by refutation
- In general, inference is first order logic is semi-decidable
- FOL + basic math is no longer complete

Logic in Practice

- Resolution in practice:
  - Convert to canonical form
  - Assert negation of the proof target
  - Resolve until “nil” is obtained
- Problem: In general, we can’t bound the number of steps of resolution needed.
  (In some cases, we can make assumptions about a restricted number of objects in the universe, but then we go from semi-decidable to exponential - still unpleasant.)

Speeding Up Resolution

- There are many heuristics for speeding up resolution – we can view it as a special kind of search
- Can also consider special cases
- AI has a colorful history of special case logics and special case reasoning engines for handling these logics

Prolog

- Prolog is a grand effort to make logic a practical programming method
- Japanese 5th generation computer project, essentially a massive prolog machine research project, burned $400M in the 1980s (target was $1.9B)
- Prolog is a *declarative* language
  - State the things that are true
  - Ask the system to prove things
  - All computations are essentially proofs
- Prolog makes many restrictions on KB
- My bias: Prolog is a fascinating way to think about logic and programming, but is of *waning* importance in AI

Prolog Properties I

- KB is sequences of sentences
  (all implicitly conjoined)
- All sentences must be horn
- Can use constants, variables, or functions
- Queries can include conjunctions or disjunctions
- Cannot assert negations
  - Closed world assumption
  - Everything not implied by the KB is assumed false

Prolog Syntax

- Variables are upper case
- Constants are lower case
- Implication :-
- Universal quantification is implicit
- Sentences are terminated with a .
- Specify RHS first: Mortal(X):-Man(X)
- Conjunction with ,: Mortal(X):-Man(X),Living(X).

Prolog UI

- Load a database using consult
- Consult(user) loads database from the command line ctrl-d to terminal
- Consult(file) loads database from a file.
- Some prologs use [file].

Prolog Bindings

- Use = to check if two bindings are same
- Use \== to check if they are different
- Hit enter at the end of query to stop search
- Use ; to get multiple answers

Some Prolog Data Types

- Lists [Head|Tail]
  - Head is bound to first element of list
  - Tail is bound to remainder of list
- Numbers
  - Numbers are assigned with "is"
  - Checked with =, =<, =>

Prolog Implementation

- Inferences are done with backward chaining
- Conjuncts are tried in left to right order
  (as entered in the KB)
- Tries implications in order they are entered
- Occupies a weird space between declarative and procedural programming

Weird/Interesting Stuff About Prolog

- Purely declarative (or purely functional) framework for programming leaves little room for “side effects” such as graphics, file output, etc.
- but… Prolog has lots of back doors that let you step outside of the purely declarative framework
- Prolog is Turing Complete
- Prolog programmers must be continually aware of the operation of the theorem proving engine
- You can easily write prolog programs that go into infinite loops, and it will not be obvious why this is happening until you have fully internalized the way the theorem prover works

Prolog Redux

- Despite its coolness and potential power, prolog is not widely used today
  - Horn restrictions are awkward in practice
  - Knowledge representation is hard in general
  - Alien paradigm to many, and awkward for things other than logic queries
- Prolog concepts live on in a restricted form in the database query language datalog:
  - Subset of prolog
  - Efficient implementations exist
- Datalog is used primarily for database research, but datalog concepts have influenced mainstream database implementations