CS 161 Intro. To Artificial Intelligence

Week 6, Discussion 1C

Li Cheng Lan

Today's Topics

- **Local Search for Propositional Inference**
  - Hill Climbing
  - Simulated Annealing
- First Order Logic (FOL)
  - Syntax
  - Semantics

Propositional Inference

What's **inference**?

- Given a KB Δ and β, we want to derive β from Δ with algorithm i
- Denote as: $KB \vdash_i \alpha$

Inference methods:

- Proof by enumeration – Model Checking
    - List all the models where Δ is True, check whether β is also True
    - E.g. Use truth table
- Proof by refutation (resolution)
    - Use resolution rule
    - Often use proof by contradiction
- **Search - SAT solver** → state-of-art
- Converting sentences to tractable forms (NNF circuits) → state-of-art DNNF, d-DNNF

SAT Solver for Propositional Inference

Key idea: reducing inference queries to SAT (a special case of CSP)

- Goal of satisfiable (SAT) problem: determine satisfiability (e.g. for propositional sentences)
  - α is **satisfiable** if $M(\alpha) \neq \emptyset$. In other words, there is some assignment (model) that makes α true

CNF: $\Delta = \underbrace{(A \lor B \lor \neg C)}_{\text{Clause 1}} \land \underbrace{(\neg A \lor C)}_{\text{Clause 2}} \land \underbrace{(A \lor C \lor \neg D)}_{\text{Clause 3}}$

Consider each clause in a CNF as constraint

Say if we assign: A ← F, B ← T, C← F, D ← F ➔ this is a world w

Clause 1: F ∨ T ∨ T = T; Clause 2: T ∨ F = T; Clause 3: F ∨ F ∨ T = T

Thus the world w ⊨ Δ, it is proven that Δ is satisfiable.

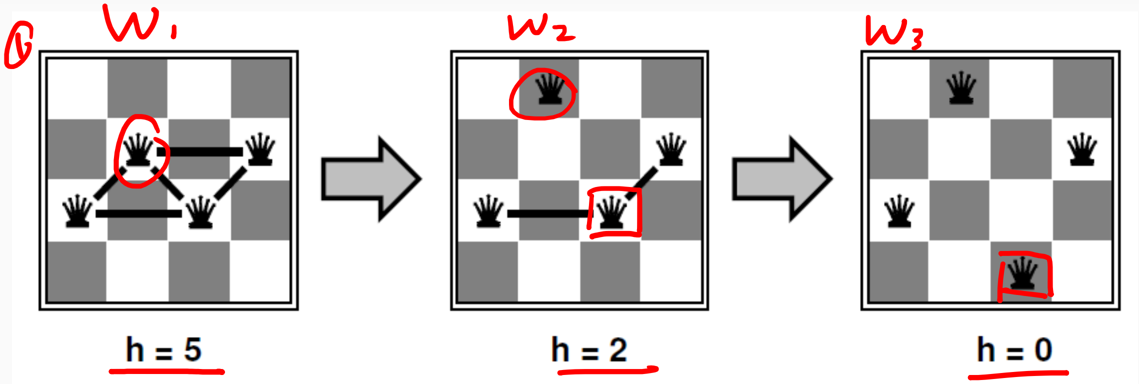
Local Search as SAT Solver

How do we solve SAT?

- Backtrack search: DFS + detecting failures early (arc consistency or forward checking)
  - Called **DPLL** (initials of four authors) in the context of SAT
- **Local search**:
  - Can be very fast, but NOT complete
  - Steps:
    - Guess a truth assignment (world w)
    - Check whether $w \models \Delta$: If yes → done. If no → find another w to try
  - Method: Pick a clause that is violated (unsatisfied), then flip a variable that either maximize # of satisfied clauses or minimize # of unsatisfied clauses → can go back and forth, thus not complete

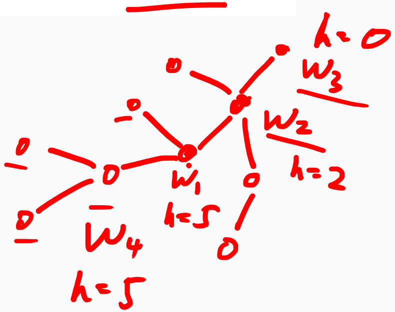
Local Search – Hill Climbing

Example: 4-queens puzzle



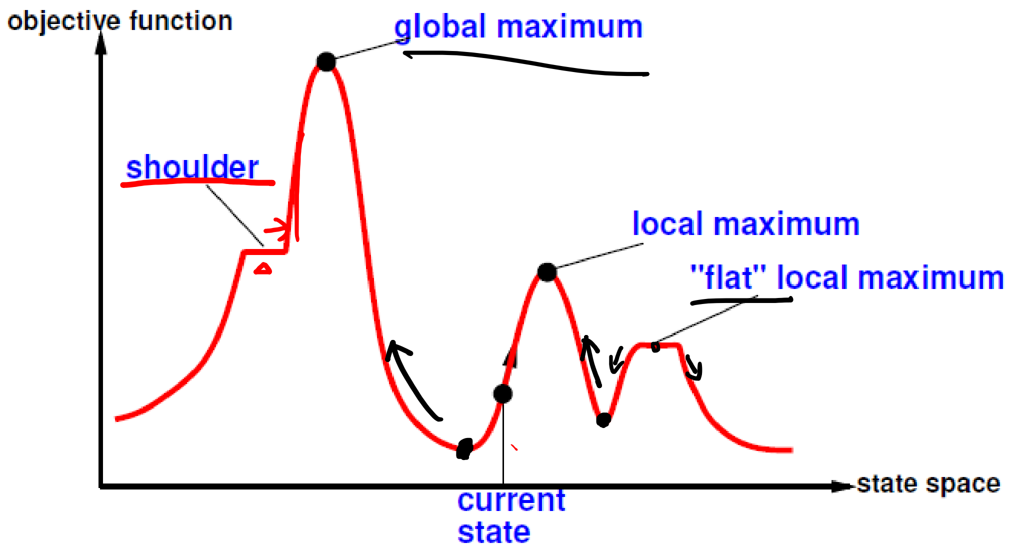
Method: **Hill-climbing** with min-conflict scoring function (h: # of conflicts)

- States with only one queen moved are **neighbors** of a full variable assignments
- Choose a neighbor that minimize # of conflict (h) to go to
- May fall in local minimum / maximum
  - Can use re-start strategy: randomly pickup another random assignment
  - Can allow side moves: move to a neighbor with similar h



Visualize Hill Climbing

Useful to consider **state space landscape**



- Escape from shoulders: **Random sideways moves**, maybe loop on flat maxima
- Escape from local maxima: **Random-restart hill climbing**, trivially complete

Local Search - Simulated Annealing

Idea: randomly pick a neighbor to move, and allow bad moves with a decreasing probability

- If the move improves the situation, it's always accepted. Otherwise, the algorithm accepts the move with a probability less than 1
- The probability decreases as exponentially with the "badness" of the move
- The probability also decreases as the search time increase

Claims:

- Simulated annealing allows us to find a global optimum if the probability decreases slowly enough
- Adding the randomness make local search methods complete as far as finding a solution if one exists, but still can't prove that a solution doesn't exit

Today’s Topics

- Local Search for Propositional Inference
  - Hill Climbing
  - Simulated Annealing
- **First Order Logic (FOL)**
  - Syntax
  - Semantics

Limitation of Propositional Logic

- Propositional logic is declarative: pieces of syntax correspond to facts
- Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases)
- Propositional logic is compositional: meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- Meaning in propositional logic is context-independent (unlike natural language, where meaning depends on context)

- Propositional logic has very limited expressive power
- **First-order logic**: similar to natural language → much more expressive power

First Order Logic (FOL) - Syntax

Syntax – how to write sentences:

- The same as in Propositional Logic, we have **sentences** in FOL.
- A sentence is evaluated as **True/False** with respect to a **model**.

objects

$Color(\cdot)$

functions

Red.

$Adjacent(\cdot, \cdot)$

$IsApple(\cdot)$

relations

T/F

Assume a world contains:

- **Objects**: A "thing in the world"
- **Relations**: Also called predicates, like a function, but returns True or False.
- **Functions**: Operator that maps object(s) to single object。

Function: object(s) $\longrightarrow$ object.

FOL - Basic Elements

Objects (terms):

- Constant: Apple, Mary, Unicorn, etc.
- Variable: x, y, z, etc. (by convention, variables are represented by lowercase letters)
- Complex terms (having functions): Mother(Mary), Color(Apple), Size(Unicorn), etc.
- Ground term: a term without variables
    - Apple, Color(Apple), etc.

      Red

object ?

Functions (Return another constant)

Predicates (evaluated as **True/False**):

- Properties (unary): UCLAStudent(Mary)
- Relations (n-ary): the set of tuples of objects that are related
    - Love(Richard, Mary), Brother(Richard, James), etc.

FOL - Basic Elements

Other elements:

an object

- Functions (**returns objects**): Sqrt, LeftLegOf, …
- Connectives: $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$
- Equality: $=$, $\neq$
- Quantifiers: $\forall$, $\exists$

FOL – Sentence Types

Atomic Sentences: one predicate and objects (terms)

- In the form of Predicate(term1, …, termN):
  - UCLAStudent(Mary) (predicate and constant)
  - UCLAStudent(x) (predicate and variable)
  - Married(Mother(Mary), Father(Mary)) (predicate, constant, function)

Complex Sentences:

- Made from atomic sentences using connectives:
  - Under20(Mary) ∧ UCLAStudent(Mary)
  - Color(Apple)=Red
  - Sold(John, Car1, Tom) ⟹ ¬ Owns(John, Car1)
  - ∀x UCLAStudent(x)⇒Person(x)
  - ∃x UCLAStudent(x)∧Under20(x)

A sentence is **evaluated as True/False** with respect to a model.

FOL - Quantifiers

Quantifiers: express properties of entire collections of objects, instead of enumerating objects by name.

- Universal quantification $\forall$ (**For all**) / For any
  - $\forall x\ King(x) \Rightarrow Person(x)$
  - Naturally uses $\Rightarrow$
- Existential quantification $\exists$ (**There exists**)
  - $\exists x\ King(x)\ \wedge OlderThan30(x)$
  - Naturally uses $\wedge$
- Uniqueness Quantifier $\exists!$
  - E.g. $\exists! x\ King(x)$ -- There is exactly one king
    - Equivalent to $\exists x\ king(x) \wedge [\forall y\ king(y) \Rightarrow (x = y)]$

uniqueness

FOL - Quantifiers

Nesting quantifiers:

- Same type quantifiers: order doesn't matter
  - ∀x ∀y (Parent(x,y) ∧ Male(y) ⇒ Son(y,x)) ∀x, y . . .
  - ∃x ∃y (Loves(x,y) ∧ Loves(y,x))
    - ∃x, y (Loves(x,y) ∧ Loves(y,x))

- Mixed quantifiers: order does matter
  - ∀x ∃y (Loves(x,y))
    - Everybody has someone they love.
  - ∃x ∀y (Loves(x,y))
    - There is someone who loves everyone.
  - ∀y ∃x (Loves(x,y))
    - Everybody has someone who loves them.
  - ∃y ∀x (Loves(x,y))
    - There is someone who is loved by everyone.

Quantifiers - Example

What do the following sentences mean?

Example for ∀:

- $\forall x$, IsBook(x) ⇒ HasAuthor(x)   Every book has author
- $\forall x$, IsBird(x) ⇒ HasFeather(x)   Every bird has feather

Example for ∃:

- $\exists x$, Person(x) ∧ Name(x, George)   Exists a x whose name is George.
- $\exists x$, Course(x) ∧ PreRequisite(x, CS161)

Quantifiers - Example

What do the following sentences mean?

Example for ∀:

- $\forall x, IsBook(x) \Rightarrow HasAuthor(x)$ Every book has an author
- $\forall x, IsBird(x) \Rightarrow HasFeather(x)$ Every bird has feather

Example for ∃:

- $\exists x, Person(x) \land Name(x, George)$ Exists some person whose name is George
- $\exists x, Course(x) \land PreRequisite(x, CS161)$ Exists some course whose pre-requisite is CS161

Quantifiers - Example

Are they equivalent? What do they mean?

- ① $\forall x\ King(x) \Rightarrow Person(x)$ ✓
- ② $\forall x\ King(x) \wedge Person(x)$ For every x, x is a king and x is a person. → too strong

- ③ $\exists x\ King(x) \wedge OlderThan30(x)$ ✓
- ④ $\exists x\ King(x) \Rightarrow OlderThan30(x)$ Exists some x, s.t. x is either not a king or x is older than 30. → too weak

$x \Rightarrow y$

$\neg x \vee y$

Quantifiers - Example

Are they equivalent? What do they mean?

They are NOT equivalent!

- $\forall x\ King(x) \Rightarrow Person(x)$ For any x, if x is king, then x is person
- $\forall x\ King(x) \land Person(x)$ For any x, x is a king and a person
  - Everything in the domain is both a king and a person (too strong)

- $\exists x\ King(x) \land OlderThan30(x)$ Exists some x, x is king and is older than 30
- $\exists x\ King(x) \Rightarrow OlderThan30(x)$ Exists some x, s.t. King(x) is false or OlderThan30(x) is true.
  - There is one person that's not a king, this is true. If there is anything that's older than 30, this is true (too weak)

Quantifiers – Logical Equivalence

- ∀ and ∃:

$$
\begin{aligned}
\forall x \;\; \neg P &\equiv \neg\exists x \;\; P & \neg(P \vee Q) &\equiv \neg P \wedge \neg Q \\
\neg\forall x \;\; P &\equiv \exists x \;\; \neg P & \neg(P \wedge Q) &\equiv \neg P \vee \neg Q \\
\forall x \;\; P &\equiv \neg\exists x \;\; \neg P & P \wedge Q &\equiv \neg(\neg P \vee \neg Q) \\
\exists x \;\; P &\equiv \neg\forall x \;\; \neg P & P \vee Q &\equiv \neg(\neg P \wedge \neg Q)\,.
\end{aligned}
$$

E.g.

∀x King(x) ⇒ Person(x) is equivalent to: ¬∃x ¬(King(x) ⇒ Person(x))

∃x Likes(x, Broccoli) is equivalent to: ¬∀x ¬Likes(x, Broccoli)

Quantifiers – Variable Scope

Variable scope:

- The **scope** of a variable is the sentence to which the quantifier syntactically applies.
  - $\forall \mathbf{x}$ King(x) ⇒ Person(x), this quantifier applies for all x
  - $\forall \mathbf{x}$ King(x) ∨ ($\exists$ $\mathbf{x}$ Brother(x, Richard))
    - This sentence is allowed. The variable belongs to the innermost quantifier that mentions it, so it will not be subject to any other quantification
    - Equivalent sentence: $\forall \mathbf{x}$ King(x) ∨ ($\exists \mathbf{z}$ Brother($\mathbf{z}$, Richard))
    - Cause confusion. Not recommended. **Use different variable names**!

for multiple quantifiers

- Not well-formed
  - $\exists$x P(y) ✗
    - All variables should be properly introduced!
- Ground expression
  - No variable → Don't need quantifier
  - E.g. King(Richard) ⇒ Person(Richard)

Translating English into FOL

How to write the following English sentences by FOL?

Richard has (at least) two brothers

$\exists x, y \; Brother(x, Richard) \wedge Brother(y, Richard) \wedge (x \neq y)$

- ~~∃ x, y Brother(x, Richard) ∧ Bother(y, Richard)~~
- ∃ x, y Brother(x, Richard) ∧ Bother(y, Richard) ∧ $(x\neq y)$

Translating English into FOL

How to write the following English sentences by FOL? What if we can only use ∀ and ∃?

Everyone has exactly one mother. $\exists!$: $\forall x, \exists! y\ Mother(y, x)$

If only use ∀ and ∃: $\forall x\ \exists y\ Mother(y,x) \wedge [\forall z\ Mother(z,x) \Rightarrow (z=y)]$

∃: E ∀: A ∨: | ∧: &

Uniqueness

- ~~$\forall x\ \exists y\ Mother(y, x)$~~
  - Everyone has at least one mother
- $\forall x\ \exists y\ Mother(y, x) \wedge [\forall z\ Mother(z, x) \Rightarrow (y=z)]$

Translating English into FOL

How to write the following English sentences by FOL?

1. Sibling” is symmetric

2. “Every gardener likes sunshine

3. Some people can be fooled all the time.

- $\exists x \, \forall t$

4. Everyone can be fooled some of the time.

- $\forall x \, \exists t$

Translating English into FOL

How to write the following English sentences by FOL?

1. "Sibling" is symmetric

$$\forall x, y \quad Sibling(x, y) \Leftrightarrow Sibling(y, x)$$

2. "Every gardener likes sunshine

$$\forall x \quad gardener(x) \Rightarrow likes(x, Sunshine)$$

3. Some people can be fooled all the time.

- $\exists x \, \forall t \; (person(x) \land time(t)) \Rightarrow can\text{-}fool(x, t)$

4. Everyone can be fooled some of the time.

- $\forall x \, \exists t \; (person(x) \land time(t)) \Rightarrow can\text{-}fool(x, t)$

Translating English into FOL

- Every gardener likes the sun.
- You can fool some of the people all of the time.
- You can fool all of the people some of the time.
- All purple mushrooms are poisonous.
- No purple mushroom is poisonous.
- There are exactly two purple mushrooms.

Translating English into FOL

- Every gardener likes the sun.

  ```
  (Ax) gardener(x) => likes(x,Sun)
  ```

  ∀

- You can fool some of the people all of the time.

  ```
  (Ex)(At) (person(x) ^ time(t)) => can-fool(x,t)
  ```

  ∃ ∧

- You can fool all of the people some of the time.

  ```
  (Ax)(Et) (person(x) ^ time(t) => can-fool(x,t)
  ```

- All purple mushrooms are poisonous.

  ```
  (Ax) (mushroom(x) ^ purple(x)) => poisonous(x)
  ```

- No purple mushroom is poisonous.

  ```
  ~(Ex) purple(x) ^ mushroom(x) ^ poisonous(x)
  ```

  or, equivalently,

  ```
  (Ax) (mushroom(x) ^ purple(x)) => ~poisonous(x)
  ```

- There are exactly two purple mushrooms.

  ```
  (Ex)(Ey) mushroom(x) ^ purple(x) ^ mushroom(y) ^ purple(y) ^ ~(x=y) ^ (Az)
  (mushroom(z) ^ purple(z)) => ((x=z) v (y=z))
  ```

FOL - Models

In logical system, a sentence is evaluated as True or False with respect to **a model (possible world).**

In Propositional Logic, a model is an assignment for some sentences

- E.g., $f = (\neg A \land B) \leftrightarrow C$
  $w = \{A : 1, B : 1, C : 0\}$

w: $\frac{A \quad B \quad C}{T \quad T \quad F}$ → n var.

2^n # of worlds/models.

- If a sentence α is true in model m, we say that model m satisfies α
- $M(\alpha) :=$ the set of all the models that satisfy α

Q: What about in First-Order Logic?

A: Much more complex!

FOL - Models

A model in FOL consists of:

- A set of objects (domain elements)
- A set of predicates + what values will be returned (relations of objects)
- A set of functions + what values will be returned (functional relations of objects)

An atomic sentence predicate(term1, …, termN) is true iff the objects referred to by term1, …, termN are in the relation referred to by predicate

FOL - Model Example

Consider:

Objects

$Orange$

$Apple$

Predicates → T/F

$IsRed(\cdot)$

$HasVitaminC(\cdot)$

Functions → an object

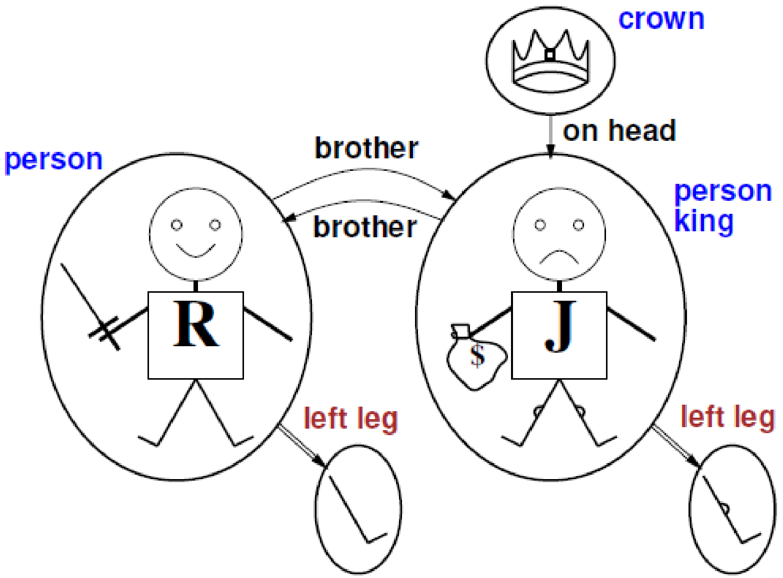
$OppositeOf(\cdot)$

Example model:

Predicate	**Argument**	**Value**
$IsRed$	$Orange$	$False$
$IsRed$	$Apple$	$True$
$HasVitaminC$	$Orange$	$True$
$HasVitaminC$	$Apple$	$True$

Function	**Argument**	**Return**
$OppositeOf$	$Orange$	$Apple$
$Opposite$	$Apple$	$Orange$

FOL - Model Example



- Five objects: Richard, John, Richard's left leg, John's left leg, crown
- Two binary relations: brother(,), on head (,)
  predicates
- Three unary relations: person(), king(), crown()
- Unary function: left leg()

KB in FOL

A knowledge base (KB) is now:

- A set of objects.
- A set of predicates.
- A set of functions.
- A set of sentences using the predicates, functions, and objects, and **asserted to be true**.

Objects	Predicates	Functions
$Orange$	$IsRed(\cdot)$	$OppositeOf(\cdot)$
$Apple$	$HasVitaminC(\cdot)$	

$$IsRed(Apple) \rightarrow T$$
$$HasVitaminC(Orange) \rightarrow T$$

KB!

KB in FOL - Example

Another example of KB:

==

$\forall x\ King(x) \Rightarrow Person(x)$

$\forall x, y\ Person(x) \land Brother(x, y) \Rightarrow Person(y)$

$\forall x, y\ Brother(x, y) \Rightarrow Brother(y, x)$

General Knowledge

$King(Richard)$

$Brother(John, Richard)$

specific problem

True

==

Questions?

My slides take the following materials as references:

- Prof. Darwiche's lecture video
- Shirley Chen's slides
- Yewen Wang's (Winter 2020's TA) slides
- Prof. Quanquan Gu's (Winter 2020) slides
- https://www2.cs.duke.edu/courses/spring15/compsci270/slides/270_6.pdf

Thank you!