Midterm Overview

Midterm Format

- Open-book and Open-Notes!!
- NO LAP TOPS or PDAs – please print materials
- BYOS – Bring Your Own Stuff!! No sharing of notes or books will be allowed during the test
- 75 minutes typically
- Will include all the material done up to now (Please see class Schedule and AIMA book chapters corresponding to the materials)
- **Only those portions that we have emphasized in class are critical**
- Focus on the material in the lecture notes

Topics Included

- Uninformed search: breadth-first, depth-first, uniform cost search, depth limited, iterative deepening depth first search
- Informed search: heuristic based search, greedy search, A*, and properties of admissible heuristic functions
- Game Trees: How to create a game tree, minmax algorithm, alpha beta pruning.
- Propositional Logic: including manipulation of logic formulae, de Morgan’s laws, conversion to CNF, and proof by enumeration, forward chaining, backward chaining, and resolution
- First Order Logic (as of Lecture 1 on October 4th – translating English to FOL) – NO INFERENCE

Nature of Questions

- Category 1: Translating informal English language problem descriptions into formal structures more amenable to algorithmic manipulation:
  - 2-3 questions of this category
- Category 2: Testing knowledge and skills with different algorithms and procedures, given an already formalized problem:
  - 2-3 questions of this category
- First category could be open-ended if you get confused – so doing Category 2 first seems wise

How to Prepare

- Make sure you take the sample test under battle conditions:
  - Solutions are also posted on the website for reality check!
- Always easier to understand a solution than to reason it out for yourself when you are asked to write it down
- Lecture notes include several exercises on all the different major uninformed and informed search algorithms, conversion of propositional logic formulae to CNF, proof by resolution of propositional logic queries, and first order logic.
- All were described in detail in class and most were also done by you as brief class exercises.
- Go over the exercises in the lecture notes

Inference in First-Order Logic

- Proofs

- Unification
- Generalized modus ponens
- Forward and backward chaining

- Completeness

- Resolution

- Logic programming

Inference in First-Order Logic

- Proofs – extend propositional logic inference to deal with quantifiers

- Unification – coming up with valid substitutions to apply rules
- Generalized modus ponens
- Forward and backward chaining – inference rules and reasoning program
- Completeness – Gödel’s theorem: for FOL, any sentence entailed by another set of sentences can be proved from that set
- Resolution – inference procedure that is complete for any set of sentences
- Logic programming

Logic as a representation of the World

$$\begin{tikzcd}
\text{Representation:} & \text{Sentences} \arrow[r, "\text{entails}"] \arrow[d, "\text{Refers to (Semantics)}"'] & \text{Sentence} \arrow[d] \\
\text{World} & \text{Facts} \arrow[r, "\text{follows}"] & \text{Fact}
\end{tikzcd}$$

Desirable Properties of Inference Procedures

$$\begin{tikzcd}
\text{Sentences} \arrow[r, bend left=50, "\text{derivation}", "\vdash"'] \arrow[r, "\text{entail}", "\models"'] \arrow[d] & \text{Sentence} \arrow[d] \\
\text{Facts} \arrow[r, "\text{Follows} - \text{from}^{-1}"] & \text{Fact}
\end{tikzcd}$$

Reduction to propositional inference

Suppose the KB contains just the following:

$\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
King(John)
Greedy(John)
Brother(Richard,John)

- Instantiating the universal sentence in all possible ways, we have:
  King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)
  King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)
  King(John)
  Greedy(John)
  Brother(Richard,John)

- The new KB is **propositionalized**: proposition symbols are

  King(John), Greedy(John), Evil(John), King(Richard), etc.

Reduction contd.

- Every FOL KB can be "**propositionalized**" so as to preserve entailment
- (A ground sentence is entailed by new KB iff entailed by original KB)
- Idea: propositionalize KB and query, apply propositional resolution, return result
- Problem: with **function symbols**, there are infinitely many ground terms,
  - e.g., *Father*(*Father*(*Father*(*John*)))
  -

Reduction contd.

Theorem: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a **finite** subset of the propositionalized KB

```
Idea: For $n$ = 0 to ∞ do
        create a propositional KB by instantiating with depth-$n$ terms
        see if α is entailed by this KB
```

Problem: works if α is entailed, loops if α is not entailed

Theorem: Turing (1936), Church (1936) Entailment for FOL is **semidecidable** (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)

Problems with propositionalization

- Propositionalization seems to generate lots of irrelevant sentences.

- E.g., from:
  $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  $King(John)$
  $\forall y\ Greedy(y)$
  $Brother(Richard,John)$

- it seems obvious that *Evil*(*John*), but propositionalization produces lots of facts such as *Greedy*(*Richard*) that are irrelevant

- With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations.

Remember: propositional logic inference rules

- ◇ **Modus Ponens** or **Implication-Elimination**: (From an implication and the premise of the implication, you can infer the conclusion.)

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

- ◇ **And-Elimination**: (From a conjunction, you can infer any of the conjuncts.)

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

- ◇ **And-Introduction**: (From a list of sentences, you can infer their conjunction.)

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

- ◇ **Or-Introduction**: (From a sentence, you can infer its disjunction with anything else at all.)

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

- ◇ **Double-Negation Elimination**: (From a doubly negated sentence, you can infer a positive sentence.)

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

- ◇ **Unit Resolution**: (From a disjunction, if one of the disjuncts is false, then you can infer the other one is true.)

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

- ◇ **Resolution**: (This is the most difficult. Because β cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)

$$\frac{\alpha \vee \beta, \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma} \qquad \text{or equivalently} \qquad \frac{\neg\alpha \Rightarrow \beta, \qquad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

Reminder

- Ground term: A term that does not contain a variable.
  - A constant symbol
  - A function applies to some ground term

- {x/a}: substitution/binding list

Added Inference Rules for FOL

The three new inference rules for FOL (compared to propositional logic) are:

- **Universal Elimination (UE):**
  for any sentence α, variable x and ground term τ,

$$\frac{\forall x \quad \alpha}{\alpha\{x/\tau\}}$$

- **Existential Elimination (EE):**
  for any sentence α, variable x and constant symbol k not in KB,

$$\frac{\exists x \quad \alpha}{\alpha\{x/k\}}$$

- **Existential Introduction (EI):**
  for any sentence α, variable x not in α and ground term g in α,

$$\frac{\alpha}{\exists x \quad \alpha\{g/x\}}$$

Added Inference Rules for FOL

The three new inference rules for FOL (compared to propositional logic) are:

- **Universal Elimination (UE):**
  for any sentence α, variable x and ground term τ,

$$\frac{\forall x \quad \alpha}{\alpha\{x/\tau\}}$$

e.g., from $\forall$x Likes(x, Candy) and {x/Joe} we can infer Likes(Joe, Candy)

- **Existential Elimination (EE):**
  for any sentence α, variable x and constant symbol k not in KB,

$$\frac{\exists x \quad \alpha}{\alpha\{x/k\}}$$

e.g., from $\exists$x Kill(x, Victim) we can infer Kill(Murderer, Victim), if Murderer new symbol

- **Existential Introduction (EI):**
  for any sentence α, variable x not in α and ground term g in α,

$$\frac{\alpha}{\exists x \quad \alpha\{g/x\}}$$

e.g., from Likes(Joe, Candy) we can infer $\exists$x Likes(x, Candy)

Applying inference rules in FOL

Sound inference: find α such that $KB \models \alpha$.
Proof process is a search, operators are inference rules.

E.g., Modus Ponens (MP)

$$\frac{\alpha, \quad \alpha \Rightarrow \beta}{\beta} \qquad \frac{At(Joe, UCB) \quad At(Joe, UCB) \Rightarrow OK(Joe)}{OK(Joe)}$$

E.g., And-Introduction (AI)

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta} \qquad \frac{OK(Joe) \quad CSMajor(Joe)}{OK(Joe) \wedge CSMajor(Joe)}$$

E.g., Universal Elimination (UE)

$$\frac{\forall x \;\; \alpha}{\alpha\{x/\tau\}} \qquad \frac{\forall x \;\; At(x, UCB) \Rightarrow OK(x)}{At(Pat, UCB) \Rightarrow OK(Pat)}$$

τ must be a ground term (i.e., no variables)

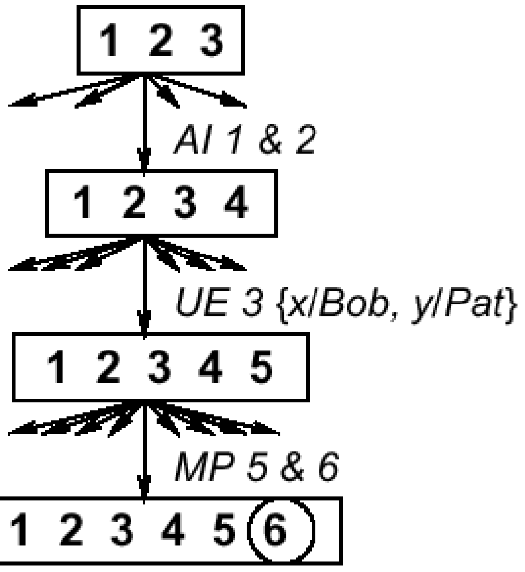
Example Proof using Current Inference Rules

Bob is a buffalo	1. $Buffalo(Bob)$
Pat is a pig	2. $Pig(Pat)$
Buffaloes outrun pigs	3. $\forall x, y \;\; Buffalo(x) \wedge Pig(y) \;\Rightarrow\; Faster(x, y)$
Bob outruns Pat	

AI 1 & 2	4. $Buffalo(Bob) \wedge Pig(Pat)$
UE 3, $\{x/Bob, y/Pat\}$	5. $Buffalo(Bob) \wedge Pig(Pat) \;\Rightarrow\; Faster(Bob, Pat)$
MP 4 & 5	6. $Faster(Bob, Pat)$

Search with primitive example rules

Operators are inference rules
States are sets of sentences
Goal test checks state to see if it contains query sentence



AI, UE, MP is a common inference pattern

Problem: branching factor huge, esp. for UE

Idea: find a substitution that makes the rule premise match some known facts

$\Rightarrow$ a single, more powerful inference rule

Unification

A substitution σ unifies atomic sentences p and q if $\underline{\underline{p\sigma = q\sigma}}$

p	q	σ
$Knows(John, x)$	$Knows(John, Jane)$	
$Knows(John, x)$	$Knows(y, OJ)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	

Goal of unification: finding σ

Unification

p	q	σ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/John, y/OJ\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$

<u>Idea</u>: Unify rule premises with known facts, apply unifier to conclusion

E.g., if we know q and $Knows(John, x) \Rightarrow Likes(John, x)$

then we conclude $Likes(John, Jane)$

$Likes(John, OJ)$

$Likes(John, Mother(John))$

Extra example for unification

P	Q	σ
Student(x)	Student(Bob)	{x/Bob}
Sells(Bob, x)	Sells(x, Pepsi)	{x/Pepsi, x/Bob} Is it correct?

Extra example for unification

P	Q	σ
Student(x)	Student(Bob)	{x/Bob}
Sells(Bob, x)	Sells(y, Pepsi)	{x/Pepsi, y/Bob} Rename variable names as needed!

More Unification Examples

1 – unify(P(a,X), P(a,b))	$\sigma = \{\overset{\text{VARIABLE}}{X}/\overset{\text{term}}{b}\}$
2 – unify(P(a,X), P(Y,b))	$\sigma = \{Y/a, X/b\}$
3 – unify(P(a,X), P(Y,f(a))	$\sigma = \{Y/a, X/f(a)\}$
4 – unify(P(a,X), P(X,b))	σ = failure

Note: If P(a,X) and P(X,b) are **independent**, then we can replace X with Y and get the unification to work.

Unification

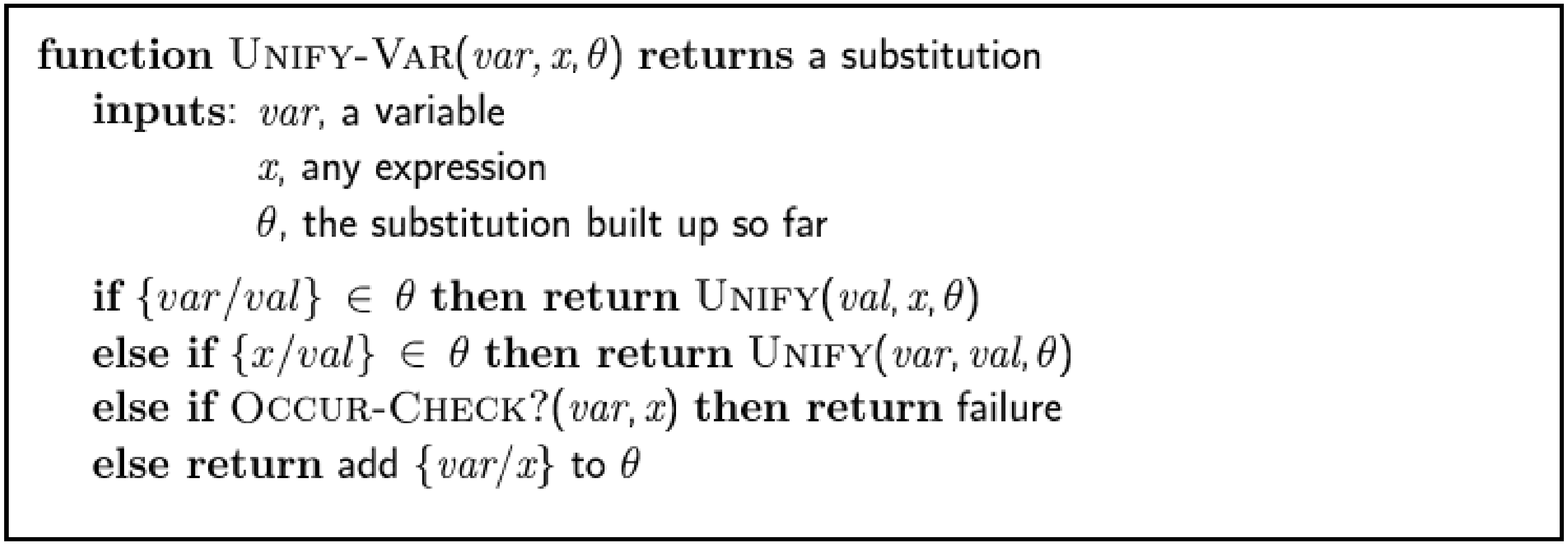
- To unify *Knows(John,x)* and *Knows(y,z)*,
  θ = {y/John, x/z } or θ = {y/John, x/John, z/John}
- The first unifier is **more general** than the second.
- There is a single **most general unifier** (MGU) that is unique up to renaming of variables.
  MGU = { y/John, x/z }

The unification algorithm

```
function UNIFY($x, y, \theta$) returns a substitution to make $x$ and $y$ identical
    inputs: $x$, a variable, constant, list, or compound
            $y$, a variable, constant, list, or compound
            $\theta$, the substitution built up so far

    if $\theta$ = failure then return failure
    else if $x = y$ then return $\theta$
    else if VARIABLE?($x$) then return UNIFY-VAR($x, y, \theta$)
    else if VARIABLE?($y$) then return UNIFY-VAR($y, x, \theta$)
    else if COMPOUND?($x$) and COMPOUND?($y$) then
        return UNIFY(ARGS[$x$], ARGS[$y$], UNIFY(OP[$x$], OP[$y$], $\theta$))
    else if LIST?($x$) and LIST?($y$) then
        return UNIFY(REST[$x$], REST[$y$], UNIFY(FIRST[$x$], FIRST[$y$], $\theta$))
    else return failure
```

The unification algorithm



```
function UNIFY-VAR($var, x, \theta$) returns a substitution
    inputs: $var$, a variable
            $x$, any expression
            $\theta$, the substitution built up so far

    if $\{var/val\} \in \theta$ then return UNIFY($val, x, \theta$)
    else if $\{x/val\} \in \theta$ then return UNIFY($var, val, \theta$)
    else if OCCUR-CHECK?($var, x$) then return failure
    else return add $\{var/x\}$ to $\theta$
```

OCCUR-CHECK?(var, x) returns true if “var” occurs inside the sentence.
If it does, there is no substitution possible and “failure” is returned

Generalized Modus Ponens (GMP)

$$\frac{p_1{}',\ p_2{}',\ \ldots,\ p_n{}',\ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\sigma} \qquad \text{where } p_i{}'\sigma = p_i\sigma \text{ for all } i$$

$$\begin{aligned}
\text{E.g. } p_1{}' &= \text{Faster(Bob,Pat)} \\
p_2{}' &= \text{Faster(Pat,Steve)} \\
p_1 \wedge p_2 \ \Rightarrow\ q &= Faster(x,y) \wedge Faster(y,z) \ \Rightarrow\ Faster(x,z) \\
\sigma &= \{x/Bob, y/Pat, z/Steve\} \\
q\sigma &= Faster(Bob, Steve)
\end{aligned}$$

GMP used with KB of definite clauses (*exactly* one positive literal):
either a single atomic sentence or
(conjunction of atomic sentences) $\Rightarrow$ (atomic sentence)
All variables assumed universally quantified

Soundness of GMP

Need to show that

$$p_1', \ldots, p_n', \;\; (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\sigma$$

provided that $p_i'\sigma = p_i\sigma$ for all i

Lemma: For any definite clause p, we have $p \models p\sigma$ by UE

1. $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \Rightarrow q)\sigma = (p_1\sigma \wedge \ldots \wedge p_n\sigma \Rightarrow q\sigma)$

2. $p_1', \ldots, p_n' \models p_1' \wedge \ldots \wedge p_n' \models p_1'\sigma \wedge \ldots \wedge p_n'\sigma$

3. From 1 and 2, $q\sigma$ follows by simple MP

Properties of GMP

- Why is GMP an efficient inference rule?

  - It takes bigger steps, combining several small inferences into one

  - It takes sensible steps: uses eliminations that are guaranteed to help (rather than random UEs)

  - It uses a precompilation step which converts the KB to canonical form (Horn sentences)

  Remember: sentence in Horn from is a conjunction of Horn clauses (clauses with at most one positive literal), e.g.,
  $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$, that is $(B \Rightarrow A) \land ((C \land D) \Rightarrow B)$