COMP 424 - Artificial Intelligence
Lecture 10: First Order Logic

Instructor: Jackie CK Cheung (jcheung@cs.mcgill.ca)

Readings: R&N Ch 8, 9

Propositional logic

- **Propositions** are assertions about the state of the world/game/problem. They can be true or false.
  - e.g. “Today is Tuesday.”, “Today is Wednesday.”
  - Can be combined using logical connectives

- **Syntax** – specifies the valid sentences in the logic
  - **Atomic sentences** S_1, S_2, etc. are sentences.
  - If S is a sentence, $\neg S$ is a sentence.
  - If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence.
  - If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence.
  - If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence.
  - If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence.

Forward vs backward chaining

- Choice depends on the problem at hand.
- **Backward chaining**:
  - parsimonious in the amount of computation done.
  - does not grow the KB as much.
  - focuses on the proof that needs to be generated, so generally more efficient.
  - does nothing until questions are asked.
  - used in proofs by contradiction.
- **Forward chaining**:
  - extends the KB.
  - improves our understanding of the world.
  - used in tasks where focus is on finding model of the world.

Other useful rules

- And-elimination:

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

- Implication elimination:

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

- De Morgan's law:

$$\neg(\alpha \vee \beta) \Leftrightarrow (\neg\alpha) \wedge (\neg\beta)$$
$$\neg(\alpha \wedge \beta) \Leftrightarrow (\neg\alpha) \vee (\neg\beta)$$

Example

Knowledge base:

- HaveAILecture $\Rightarrow$ (TodayIsMonday $\vee$ TodayIsWednesday)
- $\neg$TodayIsWednesday
- HaveAILecture $\vee$ HaveNoClass
- HaveNoClass $\Rightarrow$ Sad
- $\neg$Sad

Can you infer what day it is?

Example

$$\begin{tikzcd}[column sep=small]
\neg \text{Sad} \arrow[dr, dash] & & \text{HaveNoClass} \Rightarrow \text{Sad} \arrow[dl, dash] & & \\
& \neg \text{HaveNoClass} \arrow[dr, dash] & & \text{HaveAILecture} \vee \text{HaveNoClass} \arrow[dl, dash] & \\
& & \text{HaveAILecture} \arrow[dr, dash] & & \text{HaveAILecture} \Rightarrow (\text{TodayIsMonday} \vee \text{TodayIsWednesday}) \arrow[dl, dash] \\
& \neg \text{TodayIsWednesday} \arrow[dr, dash] & & \text{TodayIsMonday} \vee \text{TodayIsWednesday} \arrow[dl, dash] & \\
& & \text{TodayIsMonday!} & &
\end{tikzcd}$$

Complexity of inference

- What is the complexity of verifying the validity of a sentence of *n* literals?

  2^n

- What if our knowledge is expressed only in terms of Horn clauses?

  Inference time is polynomial!

  - Every Horn clause establishes exactly one fact.
  - We can state all new facts implied by the KB in *n* passes.
  - This is why Horn clauses are often used in **expert systems**.

Example: The Wumpus World

- KB:

$\neg S_{1,1} \quad \neg S_{2,1} \quad S_{1,2}$

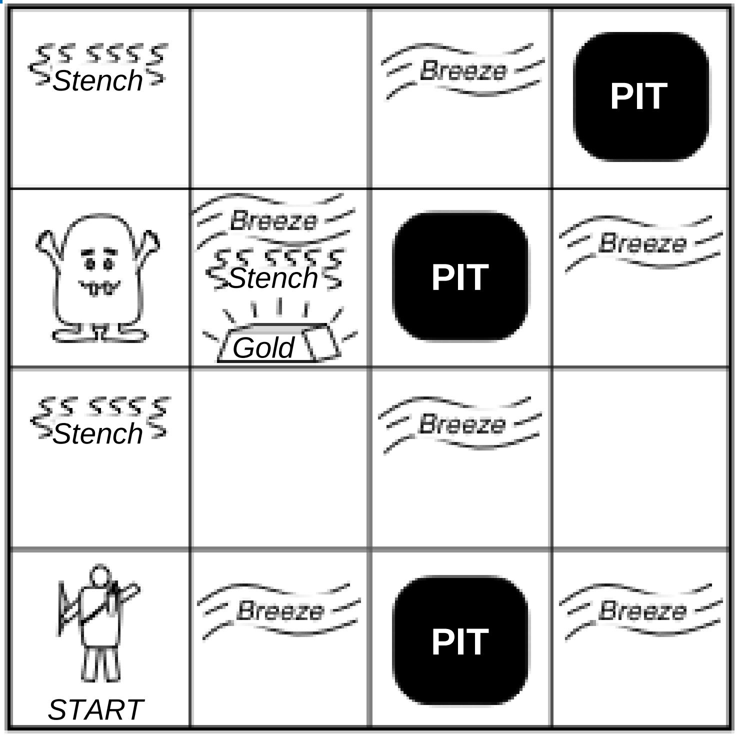
$\neg B_{1,1} \quad B_{2,1} \quad \neg B_{2,1}$

- Knowledge about the environment:

$\neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$

$\neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$

$S_{1,2} \Rightarrow W_{1,1} \vee W_{1,2} \vee W_{2,2} \vee W_{1,3}$



Now we can use inference rules to find out where the Wumpus is!

Example: The Wumpus World

- KB:

$\neg S_{1,1}$ $\neg S_{2,1}$ $S_{1,2}$

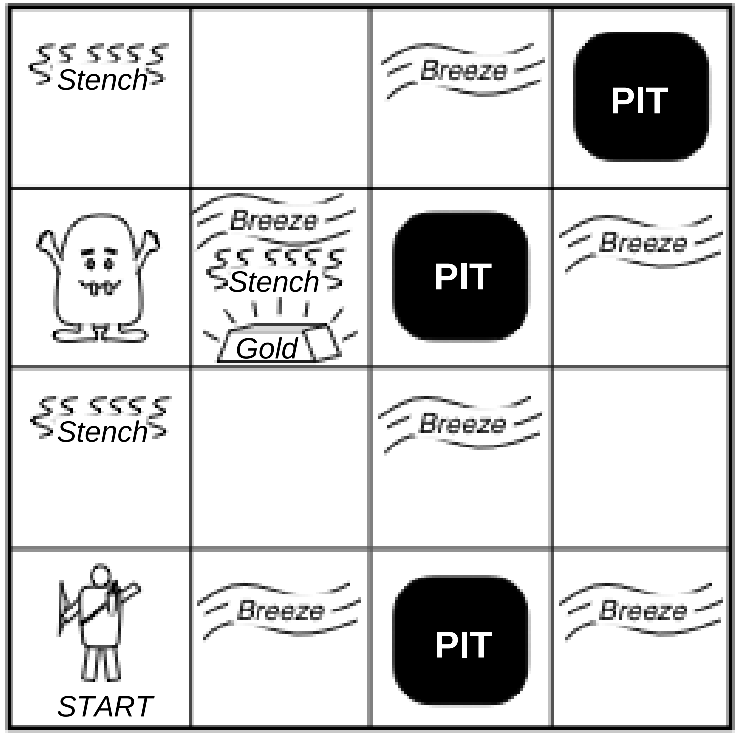
$\neg B_{1,1}$ $B_{2,1}$ $\neg B_{2,1}$

- Knowledge about the environment:

$\neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$

$\neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$

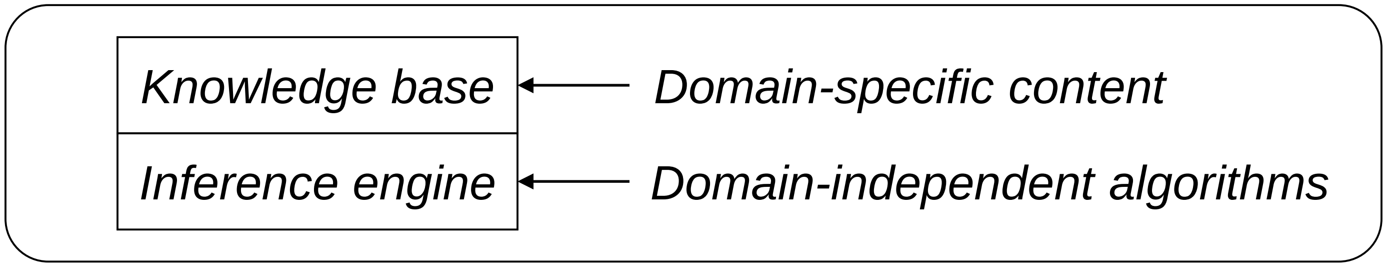
$S_{1,2} \Rightarrow W_{1,1} \vee W_{1,2} \vee W_{2,2} \vee W_{1,3}$



Must specify knowledge for each location on the grid!

But we know that the rules for stench/breeze/arrows work the same across the board!

Summary of propositional logic



- The good: propositional logic is very simple!
  - Inference is simple, few rules.
- The bad: propositional logic is very simple!
  - We cannot express things in a compact way.
  - e.g. for the Wumpus world, we need propositions for ALL positions. We cannot say things like “for all squares, try X”.

Describing the world

- Need to describe the world in a more compact and efficient way
- We'd like to be able to describe:
  - Objects in the world (e.g., cells, pits, arrows)
  - Relationships between objects (cell 1,1 is adjacent to cell 1,2)
  - Properties of those objects (cell 2,2 is safe)
- We'd also like to be able to quantify over objects
  - e.g., *All cells next to a pit are breezy. There is a wumpus somewhere in the cave.*

First-order logic (FOL)

- Add a few **new elements**:
  1. **Predicates** are used to describe objects, properties, relationships.
  2. **Quantifiers** (∀ = *"for all"*, ∃ = *"there exists"*) are used for statements that apply to a class of objects.
  3. **Functions** are used to give you an object that is related to another object in a specific way. (e.g., the cell to the RightOf cell 1,1 is cell 2,1)
     - These objects (**domain elements**) are drawn from the **domain**.

Example: FOL sentence

$\forall x$ On(x,`Table`) $\rightarrow$ Fruit(x)

$\forall$ is a **quantifier**

x is a **variable**

`Table` is a **constant**

On is a **predicate**

Note: Quantifiers allows FOL to handle **infinite domains**, while propositional local can only handle finite domains.

Syntax of FOL: Basic elements

- **Connectives** $\wedge$, $\vee$, $\neg$, $\Rightarrow$
- **Variables** *x*, *y*, …
    - Ranges over the domain
- **Quantifiers** $\forall$, $\exists$
- **Predicates** At(`Wumpus`,*x*,*y*), IsPit(*x*,*y*), …
    - Map domain element(s) to True/False
    - **Equality** = predicate that checks whether two objects refer to the same domain element
- **Functions** `SonOf`(*x*), `PlusOne`(*x*)
    - Map domain element(s) to domain element
    - **Constants** map 0 domain elements to a domain element
      `Wumpus`, `2`, `CS424`, …

Types of sentences

1. **Term**: *constant, variable, function(term$_1$, …, term$_n$)*

2. **Atomic sentences** *predicate(term, term)*

   term$_1$ = term$_2$

   e.g. *At(*`Wumpus,2,1`*)*

3. **Complex sentences**
   - Combine atomic sentences using connectives

   e.g. *At(*`Wumpus,2,1`*)* $\rightarrow \neg$*At(*`Wumpus,1,2`*)*

Universal quantification

- Form: $\forall$ <variables> <sentence>

  "Everyone taking AI is smart"

  $\forall x.\ \mathrm{Taking}(x, \texttt{AI}) \rightarrow \mathrm{Smart}(x)$

- This is equivalent to the conjunction of all variable instantiations

  $(\mathrm{Taking}(\texttt{Alice}, \texttt{AI}) \rightarrow \mathrm{Smart}(\texttt{Alice})) \wedge (\mathrm{Taking}(\texttt{Bob}, \texttt{AI}) \rightarrow \mathrm{Smart}(\texttt{Bob})) \wedge \ldots$

- Typically, $\rightarrow$ is the main connector with $\forall$

Example

- What does this statement mean?

$$\forall x.\ \text{Taking}(x, \texttt{AI}) \land \text{Smart}(x)$$

Example

- What does this statement mean?

  $\forall x.\ \text{Taking}(x, \texttt{AI}) \land \text{Smart}(x)$

Problem!

Translation is: *"Everyone is taking AI and everyone is smart."*

- **Common mistake**: Using $\land$ as the main connective with $\forall$.

Existential quantification

- Form: ∃ *<variables> <sentence>*

  "Someone taking AI is smart"

  $\exists x.\ \text{Taking}(x, \texttt{AI}) \wedge \text{Smart}(x)$

- This is equivalent to the disjunction of all variable instantiations

  $(\text{Taking}(\texttt{Alice}, \texttt{AI}) \wedge \text{Smart}(\texttt{Alice})) \vee (\text{Taking}(\texttt{Bob}, \texttt{AI}) \wedge \text{Smart}(\texttt{Bob})) \vee \ldots$

- Typically, ∧ is the main connector with ∃

Example

- What does this statement mean?

  $\exists x.\ \text{Taking}(x, \mathtt{AI}) \rightarrow \text{Smart}(x)$

Example

- What does this statement mean?

  $\exists x.\ \text{Taking}(x, \texttt{AI}) \rightarrow \text{Smart}(x)$

Problem! This sentence is true if there is anyone who is not taking AI.

Remember:

$\mathit{Taking}(x,AI) \rightarrow \mathit{Smart}(x)$ is equivalent to $\neg \mathit{Taking}(x,AI) \vee \mathit{Smart}(x)$

- **Common mistake**: Using $\rightarrow$ as the main connective with $\exists$.

Properties of quantifiers

Basic rules:

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$
- $\exists x\ \forall y$ is not the same as $\forall y\ \exists x$

Properties of quantifiers

Basic rules:

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$
- $\exists x\ \forall y$ is not the same as $\forall y\ \exists x$

E.g. $\exists x\ \forall y\ Loves(x,y)$ *Translate to English?*

$\forall y\ \exists x\ Loves(x,y)$

Properties of quantifiers

<u>Basic rules:</u>

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$
- $\exists x\ \forall y$ is <u>not</u> the same as $\forall y\ \exists x$

E.g. $\exists x\ \forall y\ Loves(x,y)$

"There is a person who loves everyone in the world."

$\forall y\ \exists x\ Loves(x,y)$

"Everyone in the world is loved by at least one person."

Quantifier duality

- Each quantifier can be expressed by using the other quantifier and negation:

$\forall x\ Loves(x, \texttt{IceCream})$ equivalent to

$\neg\exists x\ \neg Loves(x, \texttt{IceCream})$

$\exists x\ Loves(x, \texttt{Broccoli})$ equivalent to

$\neg\forall x\ \neg Loves(x, \texttt{Broccoli})$

Equality

- Expression $term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object in the domain. This is a special predicate!

E.g.	$x = y$	is satisfiable.
	$2 = 2$	is valid.

Exercise: Fun with Sentences

Given predicates *Brother(x,y), Sibling(x,y), Mother(x,y), Female(x), Parent(x,y), FirstCousin(x,y),* translate these from English to FOL:

1. Brothers are siblings.
2. Siblinghood is symmetric.
3. One’s mother is one’s female parent.
4. A first cousin is a child of a parent’s sibling.

Exercise: Fun with Sentences

Given predicates *Brother(x,y), Sibling(x,y), Mother(x,y), Female(x), Parent(x,y), FirstCousin(x,y),* translate these from English to FOL:

1. Brothers are siblings.

$$\forall x \forall y Brother(x,y) \rightarrow Sibling(x,y)$$

2. Siblinghood is symmetric.

$$\forall x \forall y Sibling(x,y) \leftrightarrow Sibling(y,x)$$

3. One's mother is one's female parent.

$$\forall x \forall y Mother(x,y) \leftrightarrow (Female(x) \wedge Parent(x,y))$$

4. A first cousin is a child of a parent's sibling.

$$\forall x \forall y FirstCousin(x,y) \leftrightarrow \exists p \exists ps Parent(p,x) \wedge$$
$$Sibling(ps,p) \wedge Parent(ps,y)$$

Truth in first-order logic

Sentences are true with respect to a **model**

A model is M = (D, I), where:

- D = Domain of objects
- I = **Interpretation**:
    - Constant symbols $\rightarrow$ *objects*
    - Predicate symbols $\rightarrow$ *relations over objects*
        - Basically, definition of the predicates as applied to the domain
    - Function symbols $\rightarrow$ *functional relations over objects*
        - Basically, definition of the functions as applied to the domain

Example of truth in FOL

- Suppose we have the sentence:

  Brother(`John`, `Richard`)

- And the model:

  Domain: D = {R, J}

  Interpretation:

  - **Predicates:**

    Brother: {(J, R), (R, J)}

  - **Functions**:

    `John`: J

    `Richard`: R

- In this model, *Brother*(`John`, `Richard`) is True.

Example of non-truth in FOL

- Suppose we have the sentence:

  Brother(`John`, `Richard`)

- And the model:

  Domain: D = {R, J}

  Interpretation:

  - **Predicates:**

    Brother: {}

  - **Functions**:

    `John`: J

    `Richard`: R

- In this model, *Brother*(`John`, `Richard`) is False.

Questions

- Suppose the domain of x and y must be a set of real numbers, and must be the same.
- Suppose > means what you would expect.
- For each of the following sentences, give a domain for which the sentence evaluates to true (if possible), and a domain for which it evaluates to false (if possible)

1. $\forall x \, \exists y \;\; x > y$
2. $\exists x \, \forall y \;\; x > y$

Inference in FOL

- Unlike with propositional logic, we have to deal with quantifiers and variables!

- Simple strategy: get rid of the quantifiers!
  - ∀: try for all possible substitutions of the variable
    e.g., if we know that $\forall x.\, Nice(x)$, replace this with
    $Nice(Sam), Nice(Aisha), Nice\big(FatherOf(Sam)\big)$ …

  - ∃: add a new constant symbol (**Skolemization**)
    e.g., if we know that $\exists x.\, Nice(x)$, replace this with
    $Nice(SomePerson)$

Inference algorithms for FOL

1. **Propositionalize** the FOL into propositional logic
   - Too expensive for all but the most trivial cases!
   - See R&N 9.1 for more details
2. **Search**
   - Forward/backward chaining using generalized modus ponens
3. **Resolution**

Inference algorithms for FOL

1. **Propositionalize** the FOL into propositional logic
   - Too expensive for all but the most trivial cases!
   - See R&N 9.1 for more details
2. **Search**
   - Forward/backward chaining using generalized modus ponens
3. **Resolution**

Proofs in FOL as search

- The proof process can be viewed as a search in which the operators are inference rules:
  - Modus Ponens (MP)

$$\frac{\alpha, \quad \alpha \rightarrow \beta}{\beta} \qquad \frac{\mathit{Takes(Joe,AI)} \quad \mathit{Takes(Joe,AI)} \rightarrow \mathit{Cool(Joe)}}{\mathit{Cool(Joe)}}$$

  - And-Introduction (AI)

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

  - Universal Elimination (UE) [aka Universal Instantiation]:

$$\frac{\forall x \alpha}{\alpha\{x/\tau\}} \qquad \frac{\forall x \ \mathit{Takes(x,AI)} \rightarrow \mathit{Cool(x)}}{\mathit{Takes(Pat,AI)} \rightarrow \mathit{Cool(Pat)}}$$

Example Proof

KB:

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

Question: Who is faster, Bob or Pat?

Proof:

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 Inference Rules

- **Operators** are inference rules.
  - MP, IA, UE
- **States** are sets of sentences.
- **Goal** test checks state to see if it contains query sentence.

Solution: Apply standard search techniques!

Problem: Branching factor is huge! Especially for UE.

- **Idea**: Find a substitution that makes the rule premise match some known facts

Unification

Pattern matching to find promising candidates for UE

We say a substitution σ unifies atomic sentences p and q if
$p\sigma = q\sigma$.

e.g.

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

Assume free variables in table are universally quantified

"If we plug in John for y and Mother(John) for x, the two sides are equal."

Unification

Idea: Unify complex sentences with known facts to draw conclusions.

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

e.g., If we know everything in *q* and also:

Knows(John,x) $\rightarrow$ *Likes(John,x)*

Then we can conclude:

Likes(John, Jane)

Likes(John, Mary)

Likes(John, Mother(John))

Generalized Modus Ponens (GMP)

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

E.g.		
	$p_1{}'$	= *Faster(Bob,Pat)*
	$p_2{}'$	= *Faster(Pat,Steve)*
	$p_1 \wedge p_2 \Rightarrow q$	= *Faster(x,y)* $\wedge$ *Faster(y,z) => Faster(x,z)*
	σ	= *x/Bob, y/Pat, z/Steve*
	$q\sigma$	= *Faster(Bob,Steve)*

Generalized Modus Ponens (GMP)

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

E.g.	p_1'	= *Faster(Bob,Pat)*
	p_2'	= *Faster(Pat,Steve)*
	$p_1 \wedge p_2 \Rightarrow q$	= *Faster(x,y)* $\wedge$ *Faster(y,z)* => *Faster(x,z)*
	σ	= *x/Bob, y/Pat, z/Steve*
	$q\sigma$	= *Faster(Bob,Steve)*

GMP used with KB of **Horn clauses** (=exactly 1 positive literal):

- a single atomic sentence;
- or a clause of the form: (conjunction of atomic sentences) => (atomic sentence).

- All variables assumed to be universally quantified.

Completeness in FOL

- Procedure *i* is complete if and only if:

$$KB \vdash_i \alpha \quad \text{whenever} \quad KB \models \alpha$$

- GMP is **complete** for KBs of universally quantified Horn clauses, but incomplete for general first-order logic.
- Entailment in FOL is only **semi-decidable**: can find a proof when KB entails *α*, but not always when KB does not entail *α*.
    - Reduces to **Halting Problem**: proof may be about to terminate with success or failure, or may go on forever.

Inference algorithms for FOL

1. **Propositionalize** the FOL into propositional logic
   - Too expensive for all but the most trivial cases!
   - See R&N 9.1 for more details
2. **Search**
   - Forward/backward chaining using generalized modus ponens
3. **Resolution**

Resolution

Two clauses can be resolved if they contain complementary literals, where one literal unifies with the negation of the other

- Intuition: Like resolution in propositional logic, but taking into account possible unifications

Example:

$Animal(F(x)) \vee Loves(G(x), x) \qquad \neg Loves(u, v) \vee \neg Kills(u, v)$

resolves to

$Animal(F(x)) \vee \neg Kills(G(x), x)$

because

$Loves(G(x), x)$ unifies with $\neg Loves(u, v)$ using substitution $\theta = \{u/G(X), v/x\}$

Resolution

- Sound and complete inference method for FOL.
- **Proof by negation**: To prove that KB entails α, instead we prove that ($KB \wedge \neg\alpha$) is **unsatisfiable**.
- **Method**:
  - The KB and $\neg\alpha$ are expressed in universally quantified, **conjunctive normal form**.
  - Repeat: The resolution inference rule combines two clauses to make a new one.
  - Continue until an empty clause is derived (contradiction).

Conjunctive Normal Form in FOL

- Literal = (possibly negated) atomic sentence, e.g., *¬Rich(Me)*
- Clause = disjunction of literals, e.g. *¬Rich(Me) v Unhappy(Me)*
- The KB is a big conjunction of clauses.

E.g.

$$\frac{\neg Rich(x) \vee Unhappy(x) \quad Rich(Me)}{Unhappy(Me)}$$

with $\sigma=\{x / Me\}$

Converting a KB to CNF

1. Replace $P \Rightarrow Q$ by $\neg P \vee Q$
2. Move $\neg$ inwards, e.g. $\neg \forall x\ P$ becomes $\exists x\ \neg P$
3. Standardize variables apart, e.g. $\forall x\ P \vee \exists x\ Q$ becomes $\forall x\ P \vee \exists y\ Q$
4. Move quantifiers left in order, e.g. $\forall x\ P \vee \exists y\ Q$ becomes $\forall x \exists y\ P \vee Q$
5. Eliminate existential quantifiers by **Skolemization**

Skolemization

- We want to get ride of existentially quantified variables:

  $\exists x\ Rich(x)$ becomes $Rich(G1)$

  where $G1$ is a new **Skolem constant**.

- It gets more tricky when $\exists$ is inside $\forall$

e.g. "Everyone has a heart"

$$\forall x\ Person(x) \Rightarrow \exists y\ Heart(y) \wedge Has(x,y)$$

How should we replace y here?

- Incorrect: $\forall x\ Person(x) \Rightarrow Heart(H1) \wedge Has(x,H1)$
- Correct: $\forall x\ Person(x) \Rightarrow Heart(H(x)) \wedge Has(x,H(x))$

  where H is a new symbol called a **Skolem function**.

Converting a KB to CNF

1. Replace $P \Rightarrow Q$ by $\neg P \vee Q$
2. Move $\neg$ inwards, e.g. $\neg \forall x\ P$ becomes $\exists x\ \neg P$
3. Standardize variables apart, e.g. $\forall x\ P \vee \exists x\ Q$ becomes $\forall x\ P \vee \exists y\ Q$
4. Move quantifiers left in order, e.g. $\forall x\ P \vee \exists x\ Q$ becomes $\forall x \exists y\ P \vee Q$
5. Eliminate existential quantifiers by Skolemization
6. Drop universal quantifiers
7. Distribute over $\vee$, e.g. $(P \wedge Q) \vee R$ becomes $(P \vee R) \wedge (Q \vee R)$

R&N 9.5.1

Example

Jack owns a dog.

Every dog owner is an animal lover.

No animal lover kills an animal.

Either Jack or Curiosity killed the cat, who is named Tuna.

Did Curiosity kill the cat?

Sentences + background knowledge

1. $\exists x : Dog(x) \wedge Owns(Jack, x)$

2. $\forall x;\ \ (\exists y\ \ Dog(y) \wedge Owns(x, y)) \rightarrow AnimalLover(x)$

3. $\forall x;\ \ AnimalLover(x) \rightarrow (\forall y\ \ Animal(y) \rightarrow \neg Kills(x, y))$

4. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

5. $Cat(Tuna)$

6. $\forall x :\ \ Cat(x) \rightarrow Animal(x)$

Example: Conjunctive Normal Form

$Dog(D)$

(D is a placeholder for the dogs unknown name (i.e. Skolem symbol/function). Think of D like "JohnDoe")

$Owns(Jack, D)$

$\neg Dog(y) \vee \neg Owns(x, y) \vee AnimalLover(x)$

$\neg AnimalLover(w) \vee \neg Animal(y) \vee \neg Kills(w, y)$

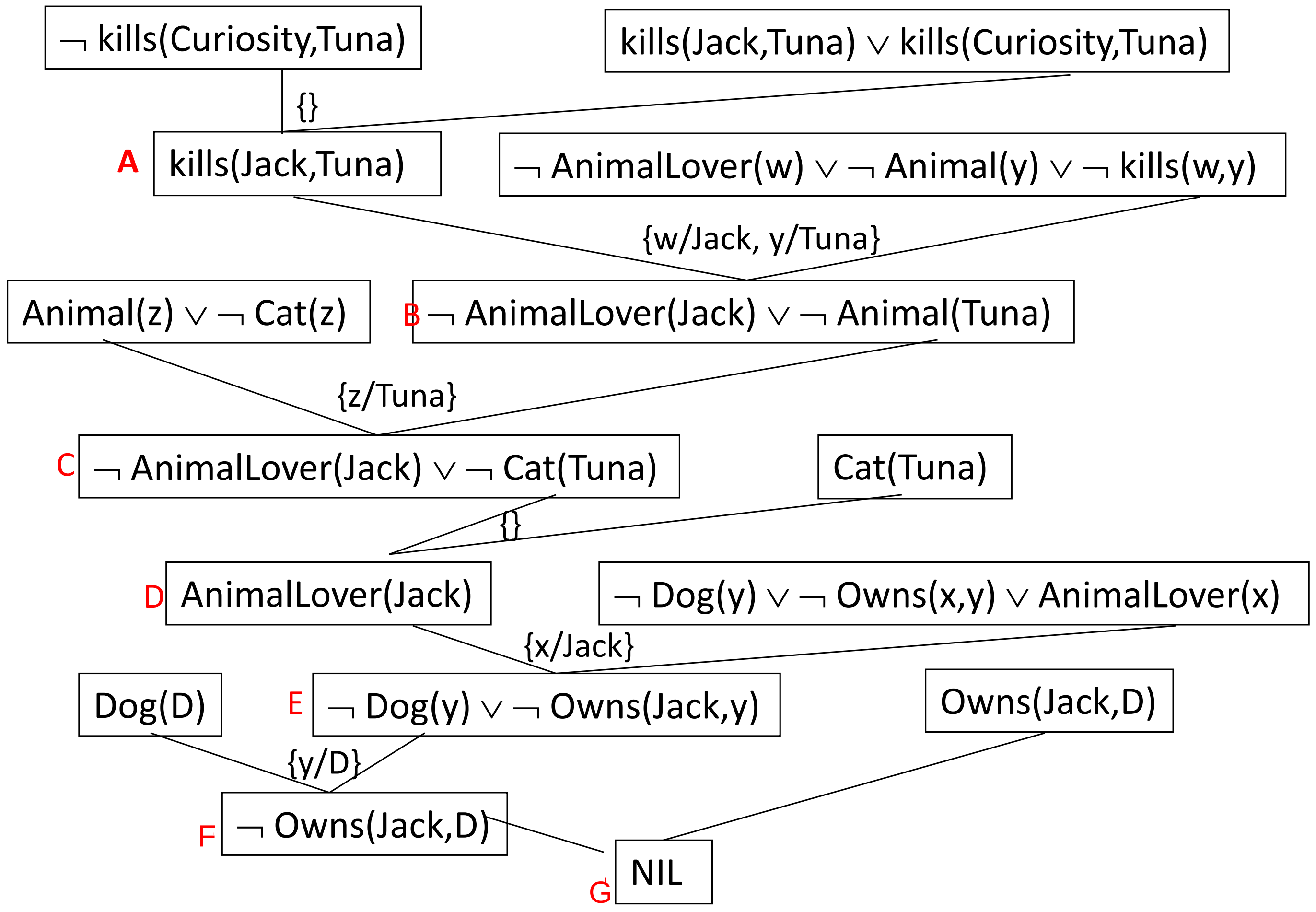
$Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

$Cat(Tuna)$

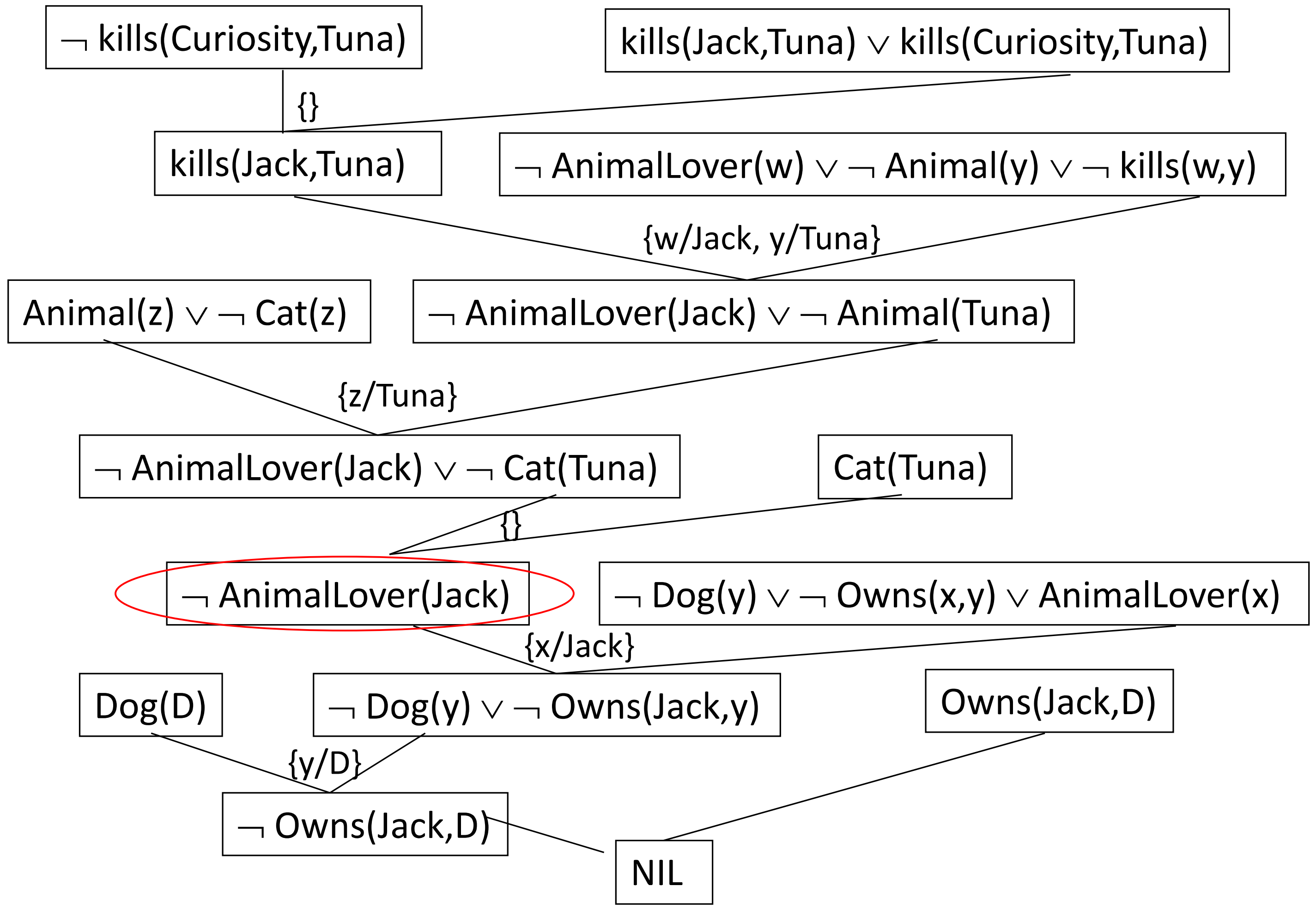
$\neg Cat(z) \vee Animal(z)$

$\neg Kills(Curiosity, Tuna)$

Example: Find the mistake!


¬ kills(Curiosity,Tuna)
kills(Jack,Tuna) ∨ kills(Curiosity,Tuna)
{}
A
kills(Jack,Tuna)
¬ AnimalLover(w) ∨ ¬ Animal(y) ∨ ¬ kills(w,y)
{w/Jack, y/Tuna}
Animal(z) ∨ ¬ Cat(z)
B
¬ AnimalLover(Jack) ∨ ¬ Animal(Tuna)
{z/Tuna}
C
¬ AnimalLover(Jack) ∨ ¬ Cat(Tuna)
Cat(Tuna)
{}
D
AnimalLover(Jack)
¬ Dog(y) ∨ ¬ Owns(x,y) ∨ AnimalLover(x)
{x/Jack}
Dog(D)
E
¬ Dog(y) ∨ ¬ Owns(Jack,y)
Owns(Jack,D)
{y/D}
F
¬ Owns(Jack,D)
G
NIL

Example: Correct proof


¬ kills(Curiosity,Tuna)
kills(Jack,Tuna) ∨ kills(Curiosity,Tuna)
{}
kills(Jack,Tuna)
¬ AnimalLover(w) ∨ ¬ Animal(y) ∨ ¬ kills(w,y)
{w/Jack, y/Tuna}
Animal(z) ∨ ¬ Cat(z)
¬ AnimalLover(Jack) ∨ ¬ Animal(Tuna)
{z/Tuna}
¬ AnimalLover(Jack) ∨ ¬ Cat(Tuna)
Cat(Tuna)
{}
¬ AnimalLover(Jack)
¬ Dog(y) ∨ ¬ Owns(x,y) ∨ AnimalLover(x)
{x/Jack}
Dog(D)
¬ Dog(y) ∨ ¬ Owns(Jack,y)
Owns(Jack,D)
{y/D}
¬ Owns(Jack,D)
NIL

Resolution Strategies

Heuristics that impose a sensible order on the resolutions we attempt

- **Unit resolution**: prefer to perform resolution if one clause is just a literal – yields shorter sentences.
- **Set of support**: identify a subset of the KB (hopefully small); every resolution will take a clause from the set and resolve it with another sentence, then add the result to the set of support.
  - Can make inference incomplete!
- **Input resolution**: always combine a sentence from the query or KB with another sentences. Not complete in general.

Simple problem

- Schubert Steamroller:
  - Wolves, foxes, birds, caterpillars, and snails are animals and there are some of each of them. Also there are some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to eat some plants.
- Prove: there is an animal that likes to eat a grain-eating animal
- Some of the necessary logical forms:
  - $\forall x\ (\text{Wolf}(x) \rightarrow \text{animal}(x))$
  - $\forall x\ \forall y\ \ ((\text{Caterpillar}(x) \vee \text{Bird}(y)) \rightarrow \text{Smaller}(x,y)$
  - $\exists x\ \text{bird}(x)$
- Requires almost 150 resolution steps (minimal)

Proofs can be lengthy!

Properties of knowledge-based systems

Advantages

1. Expressibility*: Human readable
2. Simplicity of inference procedures*: Rules/knowledge in same form
3. Modifiability*: Easy to change knowledge
4. Explainability: Answer “how” and “h y” questions.
5. Machine readability
6. Parallelism*

Disadvantages

1. Difficulties in expressibility
2. Undesirable interactions among rules
3. Non-transparent behavior
4. Difficult debugging
5. Slow
6. Where does the knowledge base come from???

Applications of FOL

- Expert systems
- Prolog: a logic programming language
- Production systems
- Semantic nets
- Automated theory proving
- **Planning (next class)**