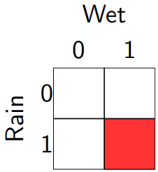
CS-541: Artificial Intelligence
Lecture 11

Abdul Rafae Khan
Department of Computer Science
Stevens Institute of Technology
akhan4@stevens.edu

April 25, 2022

Ingredients of a logic

Syntax: defines a set of valid formulas (Formulas)
Example: Rain $\wedge$ Wet
Semantics: for each formula, specify a set of models (assignments/configurations of the world)
Example:



Inference rules: given f, what new formulas g can be added that are guaranteed to follow ($\frac{f}{g}$)?
Example: from Rain $\wedge$ Wet, derive Rain

Recap: Generating Inference Rules

Start with KB
Repeatedly apply inference rules
Finally get f

Recap: Generating Inference Rules

Modus Ponens

$$\frac{p, p \rightarrow q}{q}$$

$$e.g., \quad \frac{Rain, Rain \rightarrow Wet}{Wet}$$

Modus Ponens with horn clauses

$$\frac{p1, \cdots, p_k, (p1 \wedge \cdots \wedge p_k) \rightarrow q}{q}$$

$$e.g., \quad \frac{Wet, Weekday, Wet \wedge Weekday \rightarrow Traffic}{Traffic}$$

Resolution

$$\frac{f_1 \vee \cdots \vee f_n \vee p, \neg p \vee g_1 \vee \cdots \vee g_m}{f_1 \vee \cdots \vee f_n \vee g_1 \vee \cdots \vee g_m}$$

$$e.g., \quad \frac{Rain \vee Snow, \neg Snow \vee Traffic}{Rain \vee Traffic}$$

Recap: Soundness & Completeness

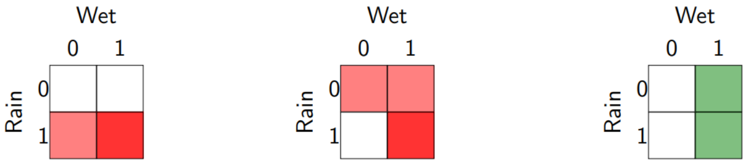
Soundness: *nothing but the truth*
Entailment ($KB \models f$)

Completeness: *whole truth*
Derivation ($KB \vdash f$)

Recap: Soundness

Is $\frac{Rain, Rain \rightarrow Wet}{Wet}$ (Modus ponens) sound?

$$\mathcal{M}(Rain) \;\cap\; \mathcal{M}(Rain \rightarrow Wet) \;\subseteq\; \mathcal{M}(Wet)$$



Sound!

Recap: Completeness

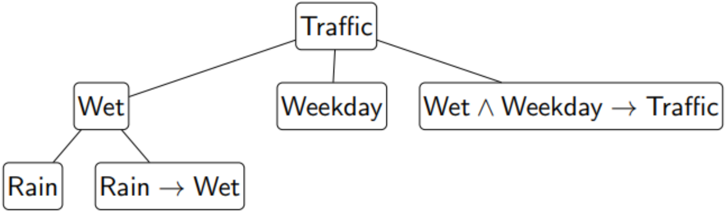
KB

Rain
Weekday
Rain $\rightarrow$ Wet
Wet $\wedge$ Weekday $\rightarrow$ Traffic
Traffic $\wedge$ Careless $\rightarrow$ Accident

Modus ponens with horn clauses

$$\frac{p_1, \cdots, p_k, (p_1 \wedge \cdots \wedge p_k) \rightarrow q}{q}$$

$$KB \models Traffic \leftrightarrow KB \vdash Traffic$$



Limitations of propositional logic

Alice and Bob both know arithmetic.

Limitations of propositional logic

Alice and Bob both know arithmetic.

$\mathit{AliceKnowsArithmetic} \wedge \mathit{BobKnowsArithmetic}$

Limitations of propositional logic

Alice and Bob both know arithmetic.
$AliceKnowsArithmetic \land BobKnowsArithmetic$

All students know arithmetic.

Limitations of propositional logic

Alice and Bob both know arithmetic.
$AliceKnowsArithmetic \wedge BobKnowsArithmetic$

All students know arithmetic.
$AliceIsStudent \rightarrow AliceKnowsArithmetic$
$BobIsStudent \rightarrow BobKnowsArithmetic$
$\ldots$

Limitations of propositional logic

Alice and Bob both know arithmetic.
$\mathit{AliceKnowsArithmetic} \wedge \mathit{BobKnowsArithmetic}$

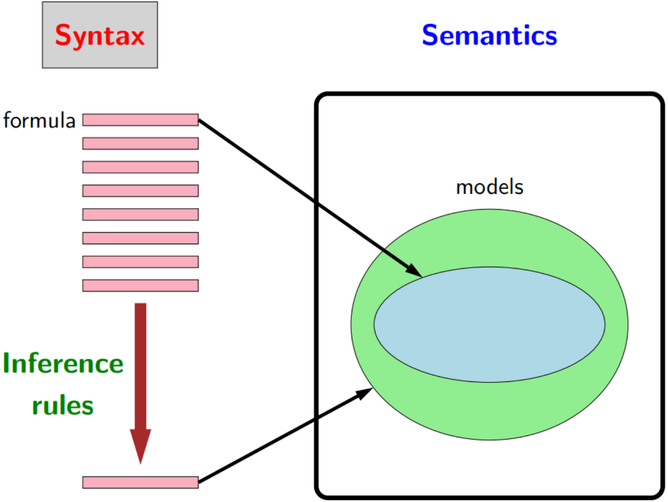
All students know arithmetic.
$\mathit{AliceIsStudent} \rightarrow \mathit{AliceKnowsArithmetic}$
$\mathit{BobIsStudent} \rightarrow \mathit{BobKnowsArithmetic}$
$\ldots$

Every even integer greater than 2 is the sum of two primes.
?

Limitations of propositional logic

All students know arithmetic.
$AliceIsStudent \rightarrow AliceKnowsArithmetic$
$BobIsStudent \rightarrow BobKnowsArithmetic$
. . .
Propositional logic is very clunky. What's missing?
• Objects and predicates: propositions (e.g., *AliceKnowsArithmetic*) have more internal structure (alice, Knows, arithmetic)
• Quantifiers and variables: all is a quantifier which applies to each person, don't want to enumerate them all...

First-order logic



First-order logic: examples

Alice and Bob both know arithmetic.

$$Knows(alice, arithmetic) \wedge Knows(bob, arithmetic)$$

First-order logic: examples

Alice and Bob both know arithmetic.

$$\mathit{Knows}(\mathit{alice}, \mathit{arithmetic}) \wedge \mathit{Knows}(\mathit{bob}, \mathit{arithmetic})$$

All students know arithmetic.

$$\forall x \; \mathit{Student}(x) \rightarrow \mathit{Knows}(x, \mathit{arithmetic})$$

Syntax of first-order logic

Terms (refer to objects):

- Constant symbol (e.g., $\mathit{arithmetic}$)
- Variable (e.g., x)
- Function of terms (e.g., $\mathit{Sum}(3, x)$)

Syntax of first-order logic

Terms (refer to objects):

- Constant symbol (e.g., $\mathit{arithmetic}$)
- Variable (e.g., x)
- Function of terms (e.g., $\mathit{Sum}(3, x)$)

Formulas (refer to truth values):

- Atomic formulas (atoms): predicate applied to terms (e.g., $\mathit{Knows}(x, \mathit{arithmetic})$)
- Connectives applied to formulas (e.g., $\mathit{Student}(x) \rightarrow \mathit{Knows}(x, \mathit{arithmetic})$)
- Quantifiers applied to formulas (e.g., $\forall x\ \mathit{Student}(x) \rightarrow \mathit{Knows}(x, \mathit{arithmetic})$)

Quantifiers

Universal quantification ($\forall$):
Think conjunction: $\forall x \, P(x)$ is like $P(A) \wedge P(B) \wedge \cdots$
Existential quantification ($\exists$):
Think disjunction: $\exists x \, P(x)$ is like $P(A) \vee P(B) \vee \cdots$

Some properties:
- $\neg \forall x \, P(x)$ equivalent to $\exists x \, \neg P(x)$
- $\forall x \, \exists y \, \mathit{Knows}(x, y)$ different from $\exists y \, \forall x \, \mathit{Knows}(x, y)$

Natural language quantifiers

Universal quantification ($\forall$):

Every student knows arithmetic.

$\forall x \; Student(x) \wedge Knows(x, arithmetic)$

Natural language quantifiers

Universal quantification ($\forall$):
Every student knows arithmetic.
$\forall x\ Student(x) \rightarrow Knows(x, arithmetic)$

Natural language quantifiers

Universal quantification ($\forall$):
Every student knows arithmetic.
$\forall x\ Student(x) \rightarrow Knows(x, arithmetic)$

Existential quantification ($\exists$):
Some student knows arithmetic.
$\exists x\ Student(x) \wedge Knows(x, arithmetic)$

Natural language quantifiers

Universal quantification ($\forall$):
Every student knows arithmetic.
$\forall x \; Student(x) \rightarrow Knows(x, arithmetic)$

Existential quantification ($\exists$):
Some student knows arithmetic.
$\exists x \; Student(x) \land Knows(x, arithmetic)$

Note the different connectives!

Some examples of first-order logic

There is some course that every student has taken.

Some examples of first-order logic

There is some course that every student has taken.
$\exists y\ Course(y) \wedge [\forall x\ Student(x) \rightarrow Takes(x, y)]$

Some examples of first-order logic

There is some course that every student has taken.
$\exists y \; Course(y) \land [\forall x \; Student(x) \to Takes(x, y)]$

Every even integer greater than 2 is the sum of two primes.

Some examples of first-order logic

There is some course that every student has taken.
$\exists y\, \mathit{Course}(y) \wedge [\forall x\, \mathit{Student}(x) \rightarrow \mathit{Takes}(x, y)]$

Every even integer greater than 2 is the sum of two primes.
$\forall x\, \mathit{EvenInt}(x) \wedge \mathit{Greater}(x, 2) \rightarrow \exists y\, \exists z\, \mathit{Equals}(x, \mathit{Sum}(y, z)) \wedge \mathit{Prime}(y) \wedge \mathit{Prime}(z)$

Some examples of first-order logic

There is some course that every student has taken.
$\exists y\ Course(y) \wedge [\forall x\ Student(x) \rightarrow Takes(x, y)]$

Every even integer greater than 2 is the sum of two primes.
$\forall x\ EvenInt(x) \wedge Greater(x, 2) \rightarrow \exists y\ \exists z\ Equals(x, Sum(y, z)) \wedge Prime(y) \wedge Prime(z)$

If a student takes a course and the course covers a concept, then the student knows that concept.

Some examples of first-order logic

There is some course that every student has taken.
$\exists y \; Course(y) \wedge [\forall x \; Student(x) \rightarrow Takes(x, y)]$

Every even integer greater than 2 is the sum of two primes.
$\forall x \; EvenInt(x) \wedge Greater(x, 2) \rightarrow \exists y \; \exists z \; Equals(x, Sum(y, z)) \wedge Prime(y) \wedge Prime(z)$

If a student takes a course and the course covers a concept, then the student knows that concept.
$\forall x \; \forall y \; \forall z \; (Student(x) \wedge Takes(x, y) \wedge Course(y) \wedge Covers(y, z)) \rightarrow Knows(x, z)$

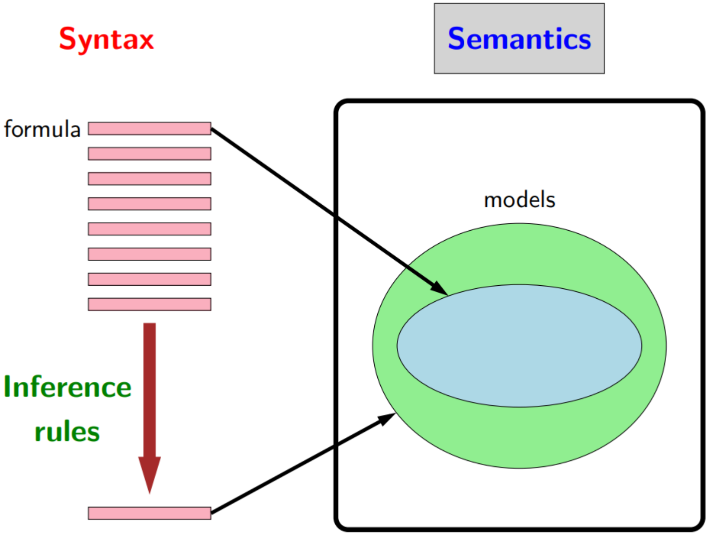
Some examples of first-order logic

There is some course that every student has taken.
$\exists y\ Course(y) \wedge [\forall x\ Student(x) \rightarrow Takes(x, y)]$

Every even integer greater than 2 is the sum of two primes.
$\forall x\ EvenInt(x) \wedge Greater(x, 2) \rightarrow \exists y\ \exists z\ Equals(x, Sum(y, z)) \wedge Prime(y) \wedge Prime(z)$

If a student takes a course and the course covers a concept, then the student knows that concept.
$\forall x\ \forall y\ \forall z\ (Student(x) \wedge Takes(x, y) \wedge Course(y) \wedge Covers(y, z)) \rightarrow Knows(x, z)$
$\forall x \forall y \forall z (Student(x) \wedge Takes(x, y) \wedge Course(y) \wedge Covers(y, z) \wedge Concept(z)) \rightarrow Knows(x, z)$

First-order logic



Models in first-order logic

Recall a model represents a possible situation in the world.

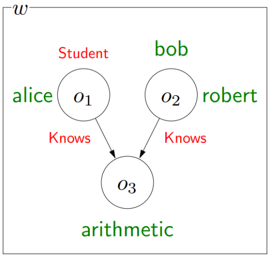
Propositional logic: Model w maps propositional symbols to truth values.

$$w = \{AliceKnowsArithmetic : 1, BobKnowsArithmetic : 0\}$$

First-order logic: ?

Graph representation of a model

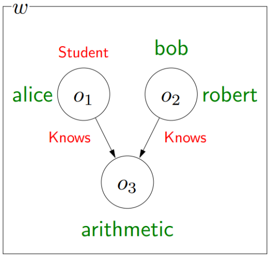
If only have unary and binary predicates, a model w can be represented as a directed graph:



- Nodes are objects, labeled with constant symbols (bob, arithemtic etc.)

Graph representation of a model

If only have unary and binary predicates, a model w can be represented as a directed graph:



- Nodes are objects, labeled with constant symbols (bob, arithemtic etc.)
- Directed edges are binary predicates, labeled with predicate symbols; unary predicates are additional node labels (Knows, Student)

Models in first-order logic

A model w in first-order logic maps:

- constant symbols to objects

$w(alice) = o_1, w(bob) = o_2, w(arithmetic) = o_3$

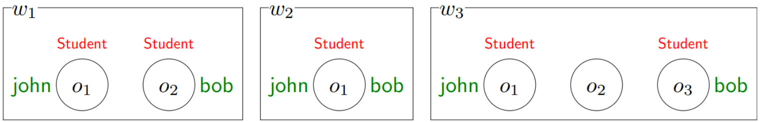
- predicate symbols to tuples of objects

$w(Knows) = \{(o_1, o_3), (o_2, o_3), \cdots\}$

A restriction on models

John and Bob are students.

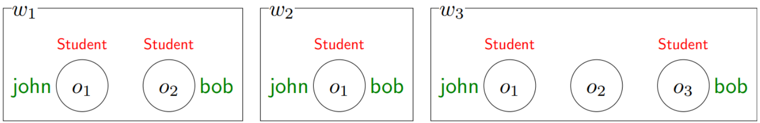
$$Student(john) \wedge Student(bob)$$



A restriction on models

John and Bob are students.

$$Student(john) \wedge Student(bob)$$



- Unique names assumption: Each object has at most one constant symbol.

This rules out w_2.

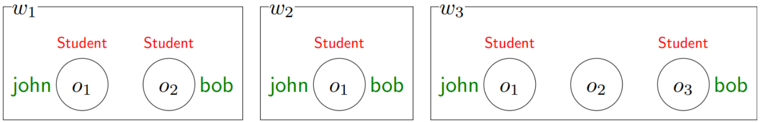
- Domain closure: Each object has at least one constant symbol.

This rules out w_3.

A restriction on models

John and Bob are students.

$$Student(john) \wedge Student(bob)$$



- Unique names assumption: Each object has at most one constant symbol.

This rules out w_2.

- Domain closure: Each object has at least one constant symbol.

This rules out w_3.

Point:

constant symbol $\longleftrightarrow$ object

Propositionalization

If one-to-one mapping between constant symbols and objects (**unique names** and **domain closure**),
first-order logic is syntactic sugar for propositional logic:
Knowledge base in first-order logic

$$\boxed{\begin{array}{c} \mathit{Student}(\mathit{alice}) \wedge \mathit{Student}(\mathit{bob}) \\ \forall x\ \mathit{Student}(x) \rightarrow \mathit{Person}(x) \\ \exists x\ \mathit{Student}(x) \wedge \mathit{Creative}(x) \end{array}}$$

Propositionalization

If one-to-one mapping between constant symbols and objects (**unique names** and **domain closure**),
first-order logic is syntactic sugar for propositional logic:

Knowledge base in first-order logic

$$\boxed{\begin{array}{c} Student(alice) \land Student(bob) \\ \forall x\ Student(x) \rightarrow Person(x) \\ \exists x\ Student(x) \land Creative(x) \end{array}}$$

Knowledge base in propositional logic

$$\boxed{\begin{array}{c} Studentalice \land Studentbob \\ (Studentalice \rightarrow Personalice) \land (Studentbob \rightarrow Personbob) \\ (Studentalice \land Creativealice) \lor (Studentbob \land Creativebob) \end{array}}$$

Propositionalization

If one-to-one mapping between constant symbols and objects (**unique names** and **domain closure**),
first-order logic is syntactic sugar for propositional logic:

Knowledge base in first-order logic

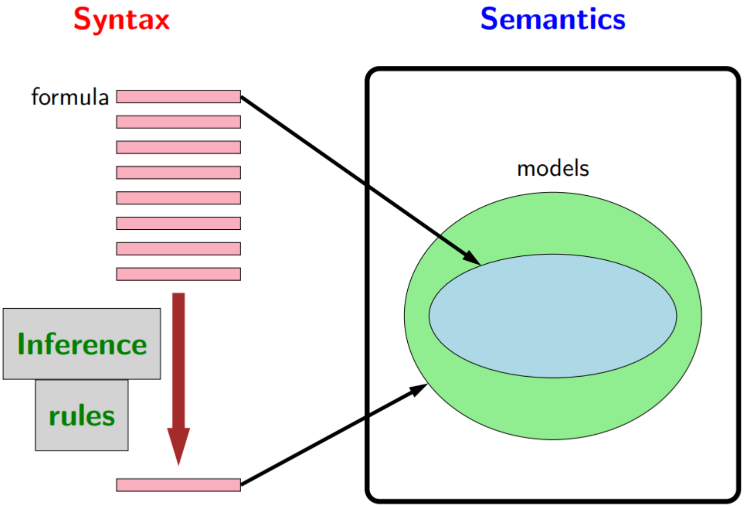
$$\boxed{\begin{array}{c} Student(alice) \wedge Student(bob) \\ \forall x\, Student(x) \rightarrow Person(x) \\ \exists x\, Student(x) \wedge Creative(x) \end{array}}$$

Knowledge base in propositional logic

$$\boxed{\begin{array}{c} Studentalice \wedge Studentbob \\ (Studentalice \rightarrow Personalice) \wedge (Studentbob \rightarrow Personbob) \\ (Studentalice \wedge Creativealice) \vee (Studentbob \wedge Creativebob) \end{array}}$$

Point: use any inference algorithm for propositional logic!

First-order logic



Definite clauses

$$\forall x \, \forall y \, \forall z \, (\mathit{Takes}(x, y) \land \mathit{Covers}(y, z)) \rightarrow \mathit{Knows}(x, z)$$

Definite clauses

$$\forall x\, \forall y\, \forall z\, (\mathit{Takes}(x, y) \wedge \mathit{Covers}(y, z)) \rightarrow \mathit{Knows}(x, z)$$

Note: if propositionalize, get one formula for each value to (x, y, z), e.g., (alice, cs541, mdp)

Definite clauses

$$\forall x\, \forall y\, \forall z\, (Takes(x, y) \wedge Covers(y, z)) \rightarrow Knows(x, z)$$

Note: if propositionalize, get one formula for each value to (x, y, z), e.g., (alice, cs541, mdp)

Definite clause (first-order logic)
A definite clause has the following form:

$$\forall x_1 \cdots \forall x_n (a_1 \wedge \cdots \wedge a_k) \rightarrow b$$

for variables $x_1, \cdots, x_n$ and atomic formulas $a_1, \cdots, a_k, b$ (which contain those variables)

Modus ponens (first attempt)

$$\frac{a_1, \cdots, a_k, \quad \forall x_1 \cdots \forall x_n \, (a_1 \wedge \cdots \wedge a_k) \rightarrow b}{b}$$

Modus ponens (first attempt)

$$\frac{a_1, \cdots, a_k, \quad \forall x_1 \cdots \forall x_n \, (a_1 \wedge \cdots \wedge a_k) \rightarrow b}{b}$$

Setup:
Given $P(alice)$ and $\forall x \, P(x) \rightarrow Q(x)$.
Problem:
Can't infer $Q(alice)$ because $P(x)$ and $P(alice)$ don't match!

Modus ponens (first attempt)

$$\frac{a_1, \cdots, a_k, \quad \forall x_1 \cdots \forall x_n \, (a_1 \wedge \cdots \wedge a_k) \rightarrow b}{b}$$

Setup:
Given $P(alice)$ and $\forall x \, P(x) \rightarrow Q(x)$.
Problem:
Can't infer $Q(alice)$ because $P(x)$ and $P(alice)$ don't match!

Solution: substitution and unification

Substitution

$$Subst[\{x/alice\}, P(x)] = P(alice)$$

$$Subst[\{x/alice, y/z\}, P(x) \wedge K(x, y)] = P(alice) \wedge K(alice, z)$$

Substitution:
A substitution θ is a mapping from variables to terms.
Subst$[\theta,\ f]$ returns the result of performing substitution θ on f.

Unification

$$Unify[Knows(alice, arithmetic), Knows(x, arithmetic)] = \{x/alice\}$$

Unification

$$Unify[Knows(alice, arithmetic), Knows(x, arithmetic)] = \{x/alice\}$$
$$Unify[Knows(alice, y), Knows(x, z)] = \{x/alice, y/z\}$$

Unification

$$Unify[Knows(alice, arithmetic), Knows(x, arithmetic)] = \{x/alice\}$$
$$Unify[Knows(alice, y), Knows(x, z)] = \{x/alice, y/z\}$$
$$Unify[Knows(alice, y), Knows(bob, z)] = fail$$

Substitution can only replace variables

Unification

$$Unify[Knows(alice, arithmetic), Knows(x, arithmetic)] = \{x/alice\}$$
$$Unify[Knows(alice, y), Knows(x, z)] = \{x/alice, y/z\}$$
$$Unify[Knows(alice, y), Knows(bob, z)] = fail$$
$$Unify[Knows(alice, y), Knows(x, F(x))] = \{x/alice, y/F(alice)\}$$

Unification

$Unify[Knows(alice, arithmetic), Knows(x, arithmetic)] = \{x/alice\}$
$Unify[Knows(alice, y), Knows(x, z)] = \{x/alice, y/z\}$
$Unify[Knows(alice, y), Knows(bob, z)] = fail$
$Unify[Knows(alice, y), Knows(x, F(x))] = \{x/alice, y/F(alice)\}$

Unification:
Unification takes two formulas f and g and returns a substitution θ which is the most general unifier:
$Unify[f, g] = \theta$ such that $Subst[\theta, f] = Subst[\theta, g]$ or "fail" if no such θ exists.

Modus ponens

$$\frac{a_1', \cdots, a_k' \quad \forall x_1 \forall x_n \, (a_1 \wedge \cdots \wedge a_k) \rightarrow b}{b}$$

Get most general unifier θ on premises:

- $\theta = Unify[a_1' \wedge \cdots \wedge a_k', a_1 \wedge \cdots \wedge a_k]$

Apply θ to conclusion:

- $Subst[\theta, b] = b'$

Modus ponens example

Modus ponens in first-order logic

Premises:

$$\begin{aligned} &\mathit{Takes}(\mathit{alice}, \mathit{cs}541) \\ &\mathit{Covers}(\mathit{cs}541, \mathit{mdp}) \\ &\forall x\, \forall y\, \forall z\; \mathit{Takes}(x, y) \wedge \mathit{Covers}(y, z) \rightarrow \mathit{Knows}(x, z) \end{aligned}$$

Conclusion:

$$\theta = \{x/\mathit{alice}, y/\mathit{cs}541, z/\mathit{mdp}\}$$

Derive $\mathit{Knows}(\mathit{alice}, \mathit{mdp})$

Complexity

$$\forall x\, \forall y\, \forall z\; P(x, y, z)$$

Each application of Modus ponens produces an atomic formula.

- If no function symbols, number of atomic formulas is at most

$$(\text{num-constant-symbols})^{(\text{maximum-predicate-arity})}$$

Complexity

$$\forall x \, \forall y \, \forall z \; P(x, y, z)$$

Each application of Modus ponens produces an atomic formula.

- If no function symbols, number of atomic formulas is at most

$$(\text{num-constant-symbols})^{(\text{maximum-predicate-arity})}$$

- If there are function symbols (e.g., F), then infinite...

$$Q(a) \quad Q(F(a)) \quad Q(F(F(a))) \quad Q(F(F(F(a)))) \quad \cdots$$

Complexity

Completeness:
Modus ponens is complete for first-order logic with only Horn clauses.

Complexity

Completeness:
Modus ponens is complete for first-order logic with only Horn clauses.

Semi-decidability:
First-order logic (even restricted to only Horn clauses) is semi-decidable.

- If $KB \models f$, forward inference on complete inference rules will prove f in finite time.
- If $KB \not\models f$, no algorithm can show this in finite time.

Resolution

Recall: First-order logic includes non-Horn clauses

$$\forall x\ Student(x) \rightarrow \exists y\ Knows(x, y)$$

Resolution

Recall: First-order logic includes non-Horn clauses

$$\forall x\, \mathit{Student}(x) \rightarrow \exists y\, \mathit{Knows}(x, y)$$

High-level strategy (same as in propositional logic):

- Convert all formulas to CNF
- Repeatedly apply resolution rule

Conversion to CNF

Input:

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

Conversion to CNF

Input:

$$\forall x\,(\forall y\; \mathit{Animal}(y) \rightarrow \mathit{Loves}(x, y)) \rightarrow \exists y\; \mathit{Loves}(y, x)$$

Output:

$$(\mathit{Animal}(Y(x)) \vee \mathit{Loves}(Z(x), x)) \wedge (\neg \mathit{Loves}(x, Y(x)) \vee \mathit{Loves}(Z(x), x))$$

Conversion to CNF

Input:

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

Output:

$$(Animal(Y(x)) \vee Loves(Z(x), x)) \wedge (\neg Loves(x, Y(x)) \vee Loves(Z(x), x))$$

New to first-order logic:

- All variables (e.g., x) have universal quantifiers by default
- Introduce **Skolem functions** (e.g., $Y(x)$) to represent existential quantified variables

Conversion to CNF (part 1)

Anyone who likes all animals is liked by someone.

Input:
$\forall x\, (\forall y\; Animal(y) \rightarrow Loves(x, y)) \rightarrow \exists y\; Loves(y, x)$

Eliminate implications (old):
$\forall x \neg(\forall y\; Animal(y) \rightarrow Loves(x, y)) \vee \exists y\; Loves(y, x)$

Conversion to CNF (part 1)

Anyone who likes all animals is liked by someone.

Input:
$\forall x\, (\forall y\ Animal(y) \rightarrow Loves(x, y)) \rightarrow \exists y\ Loves(y, x)$

Eliminate implications (old):
$\forall x \neg(\forall y\ \neg Animal(y) \vee Loves(x, y)) \vee \exists y\ Loves(y, x)$

Conversion to CNF (part 1)

Anyone who likes all animals is liked by someone.

Input:

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

Eliminate implications (old):

$\forall x \neg(\forall y\ \neg Animal(y) \vee Loves(x,y)) \vee \exists y\ Loves(y,x)$

Push $\neg$ inwards, eliminate double negation (old):

$\forall x\,(\exists y\ Animal(y) \wedge \neg Loves(x,y)) \vee \exists y\ Loves(y,x)$

Standardize variables (new):

$\forall x\,(\exists y\ Animal(y) \wedge \neg Loves(x,y)) \vee \exists z\ Loves(z,x)$

Conversion to CNF (part 2)

$$\forall x\, (\exists y\, Animal(y) \land \neg Loves(x, y)) \lor \exists z\, Loves(z, x)$$

Replace existentially quantified variables with Skolem functions (new):
$\forall x\, [Animal(Y(x)) \land \neg Loves(x, Y(x))] \lor Loves(Z(x), x)$

Distribute $\lor$ over $\land$ (old):
$\forall x\, [Animal(Y(x)) \lor Loves(Z(x), x)] \land [\neg Loves(x, Y(x)) \lor Loves(Z(x), x)]$

Remove universal quantifiers (new):
$[Animal(Y(x)) \lor Loves(Z(x), x)] \land [\neg Loves(x, Y(x)) \lor Loves(Z(x), x)]$

Resolution

Resolution rule (first-order logic)

$$\frac{f_1 \vee \cdots \vee f_n \vee p, \neg q \vee g_1 \vee \cdots \vee g_m}{Subst[\theta, f_1 \vee \cdots \vee f_n \vee g_1 \vee \cdots \vee g_m]}$$

where $\theta = Unify[p, q]$

Resolution

Resolution rule (first-order logic)

$$\frac{f_1 \vee \cdots \vee f_n \vee p, \neg q \vee g_1 \vee \cdots \vee g_m}{\mathit{Subst}[\theta, f_1 \vee \cdots \vee f_n \vee g_1 \vee \cdots \vee g_m]}$$

where $\theta = \mathit{Unify}[p, q]$

Example:

$$\frac{\mathit{Animal}(Y(x)) \vee \mathit{Loves}(Z(x), x), \neg \mathit{Loves}(u, v) \vee \mathit{Feeds}(u, v)}{\mathit{Animal}(Y(x)) \vee \mathit{Feeds}(Z(x), x)}$$

Substitution: $\theta = \{u/Z(x), v/x\}$

Summary

Propositional logic	**First-order logic**
	$\Leftarrow$ propositionalization
modus ponens (Horn clauses)	modus ponens++ (Horn clauses)
resolution (general)	resolution++ (general)

++: unification and substitution

Summary

Propositional logic	First-order logic
	$\Leftarrow$ propositionalization
modus ponens (Horn clauses)	modus ponens++ (Horn clauses)
resolution (general)	resolution++ (general)

++: unification and substitution

Key idea: variables in first-order logic
Variables yield compact knowledge representations

Recap

First-order logic
Syntax & Semantics
Inference Rules
Substitution & Unification
Modus ponens
Resolution

References

Stuart Russell and Xiaodong Song (2021)
CS 188 — Introduction to Artificial Intelligence
University of California, Berkeley

Chelsea Finn and Nima Anari (2021)
CS221 — Artificial Intelligence: Principles and Techniques
Stanford University

The End