Artificial Intelligence

Computer Science, CS541 - A

Jonggi Hong

Announcements

- HW #2: Logic
  - Has been released! Due 11:59 pm, Tuesday, March 7.

- Let me know if you cannot see slides at the beginning of each class.

Recap

Uninformed search

STEVENS
INSTITUTE OF TECHNOLOGY
1870

Propositional Logic Syntax

- Syntax: what you are allowed to write
  - for (thing t=fizz; t==fuzz; t++) { … }
  - “Colorless green ideas sleep furiously.”
- Sentences (wffs: well formed formulas)
  - <u>true</u> and <u>false</u> are sentences.
  - Propositional variables are sentences: P, Q, R, Z
  - If φ and ψ are sentences, then so are
    - (φ), $\neg \varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$, $\varphi \leftrightarrow \psi$
  - Nothing else is a sentence.

Semantic Rules

- holds(<u>true</u>, i) for all i
- fails(<u>false</u>, i) for all i
- holds(¬φ, i) if and only if fails fails(φ, i) — negation
- holds(φ ∧ ψ, i) iff holds(φ, i) and holds(ψ, i) — conjunction
- holds(φ ∨ ψ, i) iff holds(φ, i) or holds(ψ, i) — disjunction
- holds(P, i) iff i(P) = **t**
- fails(P, i) iff i(P) = **f**

Some Important Shorthand

- $\varphi \to \psi \equiv \neg \varphi \vee \psi$ (conditional, implication)
  - Implication: antecedent → consequent
- $\varphi \leftrightarrow \psi \equiv (\varphi \to \psi) \wedge (\psi \to \varphi)$ (biconditional, equivalence)

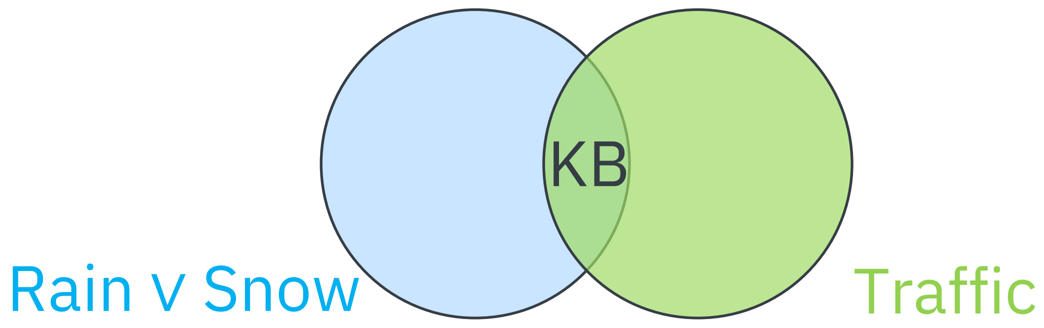
P	Q	¬ P	P ∧ Q	P ∨ Q	P → Q	Q → P	P ↔ Q
f	**f**	**t**	**f**	**f**	**t**	**t**	**t**
f	**t**	**t**	**f**	**t**	**t**	**f**	**f**
t	**f**	**f**	**f**	**t**	**f**	**t**	**f**
t	**t**	**f**	**t**	**t**	**t**	**t**	**t**

Note that implication is not "causality", if P is **f** then P→ Q is **t**.

Knowledge Base

- A **knowledge base** (KB) is a set of sentences representing their conjunction/intersection.
- Intuition
  - KB specifies constraints on the world. KB represents the set of interpretations that satisfies those constraints.

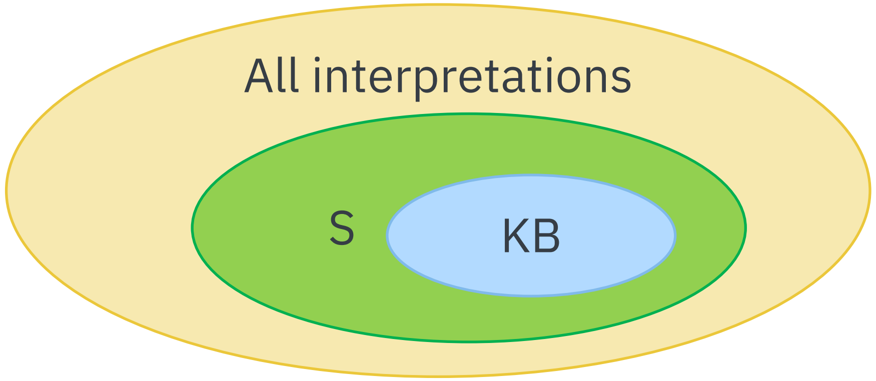
Let KB = {Rain ∨ Snow, Traffic}



Entailment

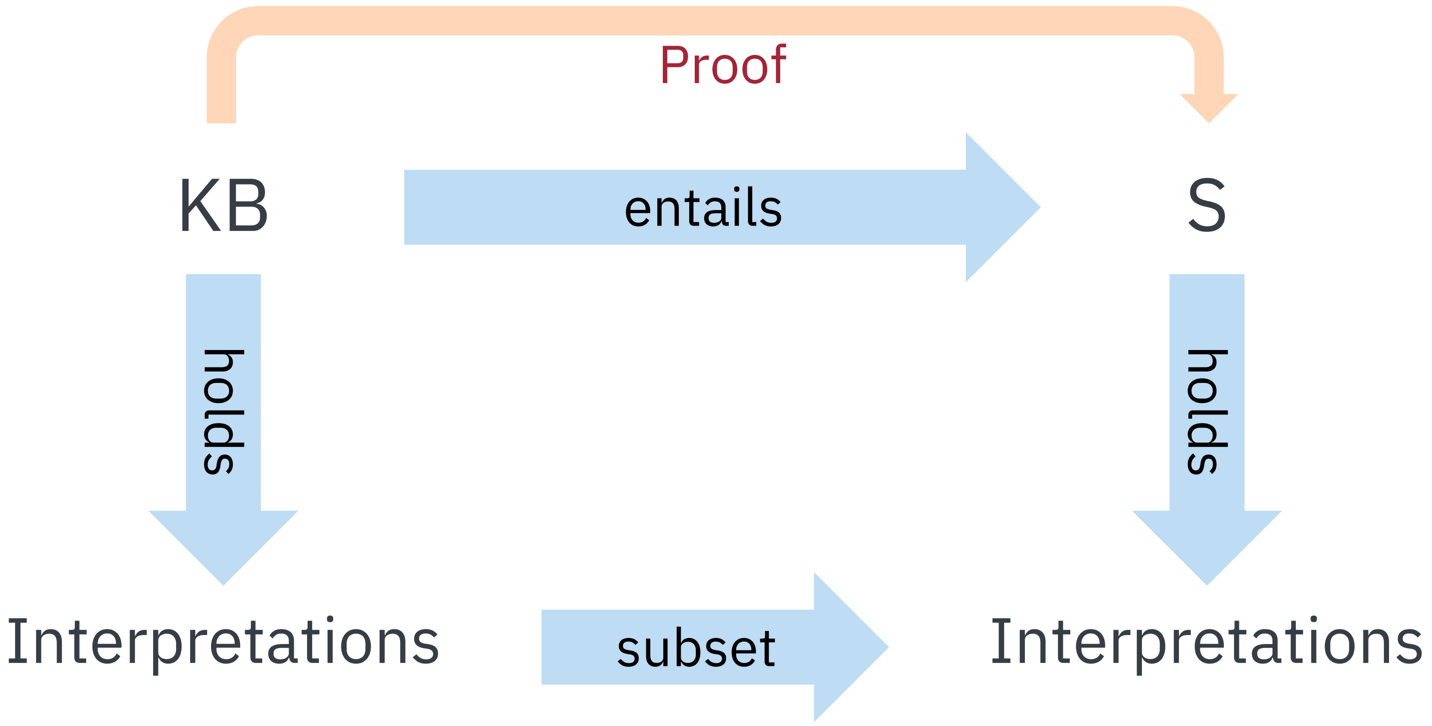
A knowledge base (KB) **entails** a sentence S iff every interpretation that makes KB true also makes S true.

- Intuition
  - The sentence adds no information/constraints (it was already known)
- Entailment
  - KB entails S (written KB $\models$ S)
- Example
  - Rain $\wedge$ Snow $\models$ Snow



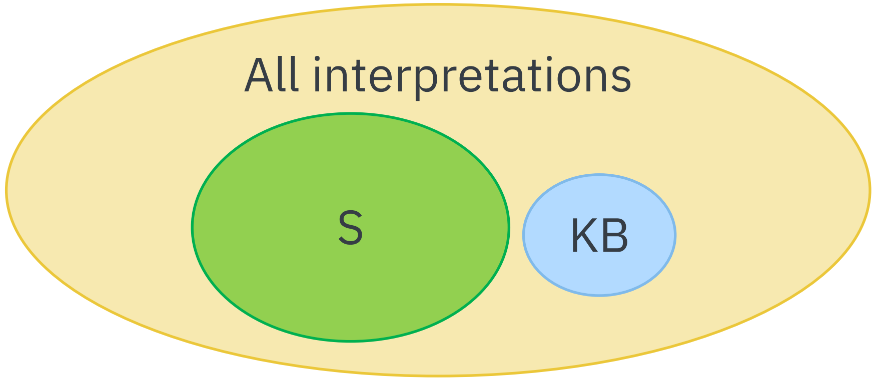
Entailment and Proof

A **proof** is a way to test whether a KB entails a sentence, without enumerating all possible interpretations.

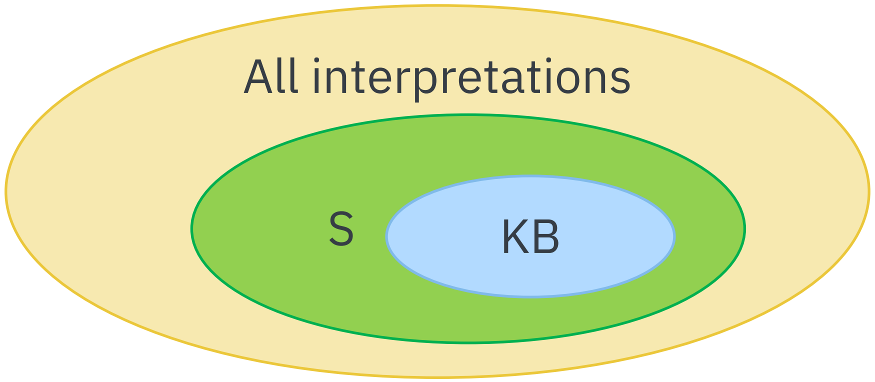


Contradiction

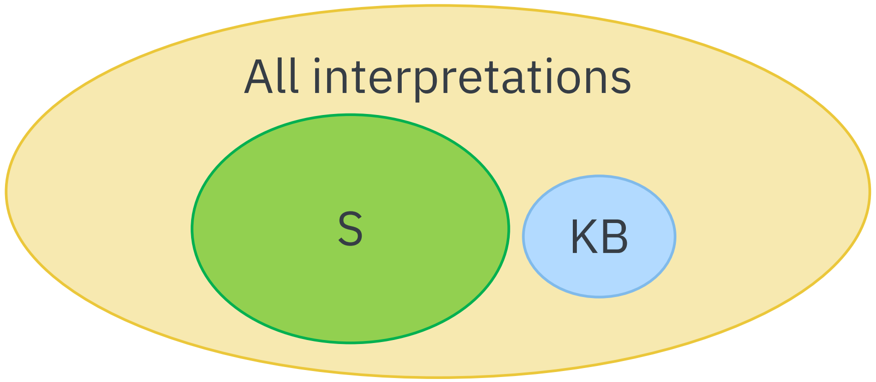
- Intuition
  - The sentence contradicts what we know (captured in KB)
- Contradiction
  - KB contradicts f iff $M(\text{KB}) \cap M(\text{S}) = \emptyset$
- Example
  - Rain ∧ Snow contradicts ¬ Snow



Entailment and Contradiction



Entailment



Contradiction

KB contradicts S iff KB entails ¬S.

Proof

- If inference rules are **sound**, then any S you can prove from KB is entailed by KB.
  - $KB \vdash S \rightarrow KB \models S$

- If inference rules are **complete**, then any S that is entailed by KB can be proved from KB.
  - $KB \models S \rightarrow KB \vdash S$

Natural Deduction

- Example of making an inference
  - It is raining. (Rain)
  - If it is raining, then it is wet. (Rain → Wet)
  - Therefore, it is wet. (Wet)

$$\frac{\begin{array}{c}\text{Rain}\\ \text{Rain} \rightarrow \text{Wet}\end{array}}{\text{Wet}}$$

(Modus ponens rule)

Natural Deduction

- Some inference rules

$$\frac{\alpha \rightarrow \beta \qquad \alpha}{\beta}$$

Modus ponens

$$\frac{\alpha \rightarrow \beta \qquad \neg \beta}{\neg \alpha}$$

Modus tolens

$$\frac{\alpha \qquad \beta}{\alpha \wedge \beta}$$

And-introduction

$$\frac{\alpha \wedge \beta}{\alpha}$$

And-elimination

Natural Deduction Example

Step	Formula	Derivation
1	$P \land Q$	Given
2	$P \rightarrow R$	Given
3	$(Q \land R) \rightarrow S$	Given
4	P	1 And-Elim
5	R	4, 2 Modus Ponens
6	Q	1 And-Elim
7	$Q \land R$	5, 6 And-Intro
8	**S**	7, 3 Modus Ponens

Proof Systems

- There are many natural deduction (ND) systems.
  - They are typically “proof checkers”.
  - The ND systems can be complete and sound with propositional logic.
- Natural deduction uses lots of inference rules which introduces a large branching factor in the search for a proof.
- In general, you need to do “proof by cases” which introduces even more branching.

Prove R

1	$P \vee Q$
2	$Q \rightarrow R$
3	$P \rightarrow R$

Propositional Resolution

- Resolution rule

$$\frac{\begin{array}{c}\alpha \vee \beta \\ \neg \beta \vee \gamma\end{array}}{\alpha \vee \gamma}$$

- A single inference rule is a sound and complete proof system.
- Requires all sentences to be converted to **conjunctive normal form**.

Converting to CNF

1. Eliminate arrows (implications) using definitions.
2. Drive in negations using De Morgan's Laws.

$$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$$
$$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$$

3. Distribute **or** over **and.**

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$

Every sentence can be converted to CNF, but it may grow exponentially in size.

Propositional Resolution

- Resolution rule

$$\frac{\begin{array}{c}\alpha \lor \beta \\ \neg \beta \lor \gamma\end{array}}{\alpha \lor \gamma}$$

- Resolution refutation
  - Convert all sentences to CNF.
  - Negate the desired conclusion (converted to CNF).
  - Apply resolution rule until either
    - Derive false (a contradiction)
    - Can’t apply any more

Quiz

- What do we get with the resolution rule?

$$\frac{\begin{array}{c} A \lor B \lor C \\ A \lor \neg B \lor \neg C \end{array}}{\text{True}}$$

The two sentences do not add any information.

Propositional Resolution Example

Prove R

1	P ∨ Q
2	$Q \rightarrow R$
3	$P \rightarrow R$

Step	Formula	Derivation
1	P ∨ Q	Given
2	¬ P ∨ R	Given
3	¬ Q ∨ R	Given
4	¬ R	Negated conclusion
5	Q ∨ R	1, 2
6	¬ P	2, 4
7	¬ Q	3, 4
8	R	5, 7
9	•	4, 8

false ∨ R

¬ R ∨ false

---

false ∨ false

First-Order Logic

First-Order Logic (FOL)

- **Variables** refer to things in the world.
- You can **quantify** over the variables.
  - You can make sentences talking about all of them or some of them without having to name them explicitly.

Semantics of Quantifiers

- Extend an interpretation I to bind variable x to element a ∈ U: $I_{x/a}$
    - holds(∀x.Φ), I) iff holds(Φ, $I_{x/a}$) for all a ∈ U
    - holds(∃x.Φ), I) iff holds(Φ, $I_{x/a}$) for some a ∈ U
- Quantifier applies to formula to right until an enclosing right parenthesis.

$$\left(\forall x.\, P(x)\ \vee Q(x)\right) \vee\ \exists x.\, R(x) \rightarrow Q(x)$$

Writing FOL

1. Cats are mammals. [Cat^1, Mammal^1]
   - $\forall x.\ \mathrm{Cat}(x) \rightarrow \mathrm{Mammal}(x)$

2. Jane is a tall surveyor. [Tall^1, Surveyor^1, Jane]
   - $\mathrm{Tall}(\mathrm{Jane}) \wedge \mathrm{Surveyor}(\mathrm{Jane})$

3. A nephew is a sibling's son. [Nephew^2, Sibling^2, Son^2]
   - $\forall xy.\ [\mathrm{Nephew}(x,y) \leftrightarrow \exists z.\ [\mathrm{Sibling}(y,z) \wedge \mathrm{Son}(x,z)]]$

4. A maternal grandmother is a mother's mother. [functions: mgm, mother-of]
   - $\forall xy.\ x=\mathrm{mgm}(y) \rightarrow \exists z.\ x=\text{mother-of}(z) \wedge z=\text{mother-of}(y)$

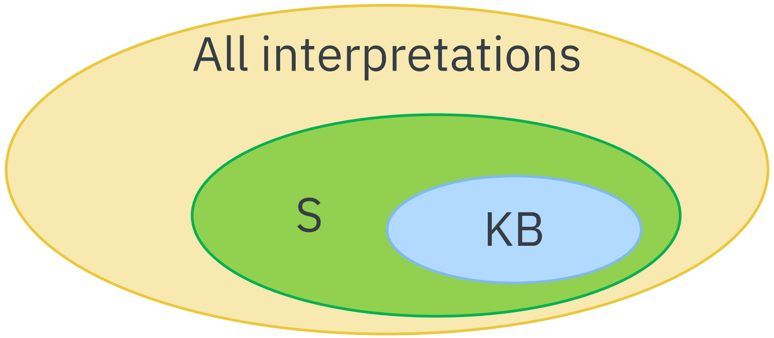
Logic

Chapter 7

Entailment in First-Order Logic

KB entails S

- **For every interpretation I, if KB holds in I, then S holds in I.**



- Computing entailment is impossible in general because there are infinitely many possible interpretations.
- Even computing holds is impossible for interpretations with infinite universes.

Intended Interpretations

KB: (∀x.Circle(x) → Oval(x)) ∧ (∀x.Square(x) → ¬ Oval(x))

S: ∀x.Square(x) → ¬ Oval(x)

- We know holds(KB, I).
- We wonder whether holds(S, I).
- We could ask “Does KB entail S?”
- Or we could just try to check whether holds(S, I)

- I(Fred) = ▲
- I(Above) = {<■, ▲>, <● , ⬬>}
- I(Circle) = {<● >}
- I(Oval) = {<●>, <⬬>}
- I(hat) = {<▲, ■ >, <⬬ ,● >, <■, ■>, <●,●>}
- I(Square) = {<▲>}

An Infinite Interpretation

KB: $(\forall x.\text{Circle}(x) \rightarrow \text{Oval}(x)) \land (\forall x.\text{Square}(x) \rightarrow \neg \text{Oval}(x))$

S: $\forall x.\text{Square}(x) \rightarrow \neg \text{Oval}(x)$

- Does KB hold in I_1?
  - Yes, but can't verify by enumerating U.
- S also holds in I_1.
  - Yes, but can't verify by enumerating U.

- $U_1 = \{1, 2, 3, \ldots\}$
- $I_1(\text{Circle}) = \{<4, 8, 12, 16, \ldots>\}$
- $I_1(\text{Oval}) = \{2, 4, 6, 8, \ldots\}$
- $I_1(\text{Square}) = \{1, 3, 5, 7, \ldots\}$

An Argument for Entailment

KB: $(\forall x.Circle(x) \rightarrow Oval(x)) \land (\forall x.Square(x) \rightarrow \neg Oval(x))$

S_1: $\forall xy.\ (Circle(x) \land Oval(y) \land \neg Circle(y)) \rightarrow Above(x,y)$

- I(Fred) = ▲
- I(Above) = {<■, ▲>, <●, ⬬>}
- I(Circle) = {<●>}
- I(Oval) = {<●>, <⬬>}
- I(hat) = {<▲, ■>, <⬬, ●>,
  <■, ■>, <●, ●>}
- I(Square) = {<▲>}
- holds(KB, I)
- holds(S_1, I)

- U_1 = {1, 2, 3, …}
- I_1 (Circle) = {<4, 8, 12, 16, …>}
- I_1 (Oval) = {2, 4, 6, 8, …}
- I_1 (Square) = {1, 3, 5, 7, …}
- I_1 (Above) = >
- holds(KB, I_1)
- fails(S_1, I_1)

KB doesn't entail S_1!

Proof and Entailment

- Entailment captures general notion of “follows from.”
- We can’t evaluate it directly by enumerating interpretations.
- So, we will do proofs.
- In FOL, if S is entailed by KB, then there is a finite proof of S from KB.

Axiomatization

- What if we have a particular interpretation, I, in mind, and want to test whether holds(S, I)?
  - Write down a set of sentences, called **axioms**, that will serve as our KB.
  - We would like KB to hold in I, and as few other interpretations as possible.
  - No matter what,
    - If holds(KB, I) and KB entails S, then holds (S, I).
  - If your axioms are week, it might be that
    - holds(KB, I) and holds(S, I), but
    - KB doesn’t entail S.

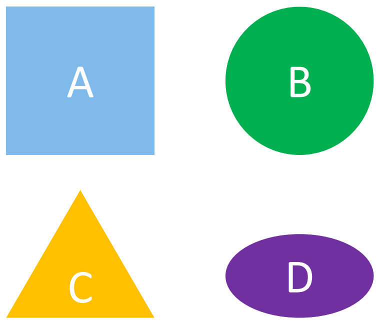
Axiomatization Example

KB_2

- Above(A, C)
- Above(B, D)
- $\forall xy.$ Above(x,y) → hat(y) = x
- $\forall x.(\neg\exists y.$ Above(y,x)) → hat(x) = x

S

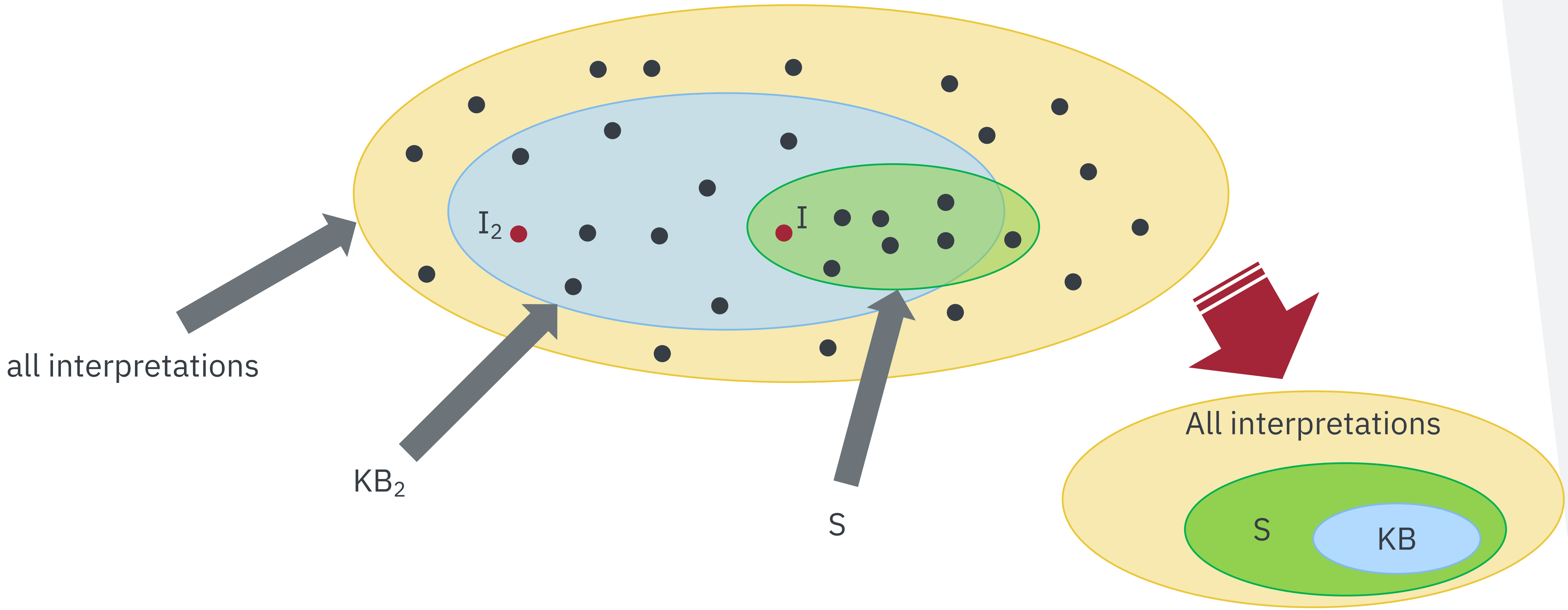
- hat(A) = A



- I_2(A) = ■
- I_2(B) = ●
- I_2(C) = ▲
- I_2(D) = ⬬
- I_2(Above) = {<■,▲>, <●, ⬬>, <▲, ■>, <⬬, ●>}
- I_2(hat) = {<▲,■>, <⬬,●>, <●, ⬬>, <■,▲>}

- holds(KB_2, I_2)
- fails(S, I_2)
- KB_2 doesn't entail S.

KB_2 is a Weakling!



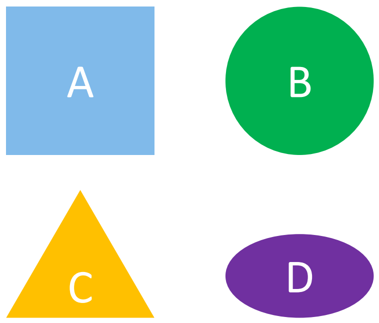
Axiomatization Example: Another Try

KB_3

- Above(A, C)
- Above(B, D)
- $\forall xy.\ Above(x,y) \rightarrow hat(y) = x$
- $\forall x.(\neg\exists y.\ Above(y,x)) \rightarrow hat(x) = x$
- **$\forall xy.\ Above(x,y) \rightarrow \neg Above(y,x)$**

S

- hat(A) = A



- $I_3(A) =$
- $I_3(B) =$
- $I_3(C) =$
- $I_3(D) =$
- $I_3(Above) = \{<\ ,\ >, <\ ,\ >, <\ ,\ >\}$
- $I_3(hat) = \{<\ ,\ >, <\ ,\ >, <\ ,\ >, <\ ,\ >\}$

- fails (KB_3, I_2)
- holds(KB_3, I_3)
- fails(S, I_3)
- KB_3 doesn't entail S.

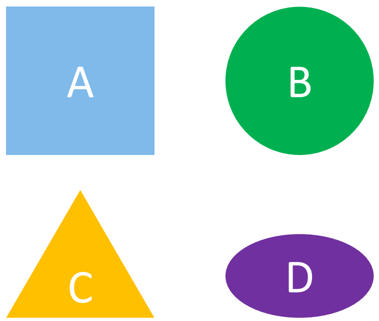
Axiomatization Example: Another Try

KB$_4$

- Above(A, C)
- Above(B, D)
- **¬∃x. Above(x, A)**
- **¬∃x. Above(x, B)**
- ∀xy. Above(x,y) → hat(y) = x
- ∀x.(¬∃y. Above(y,x)) → hat(x) = x

S

- hat(A) = A



- fails (KB$_4$, I$_3$)
- KB$_4$ entails S.

We will prove KB$_4$ entails S.

First-Order Resolution

First-Order Resolution

- Constants
  - Uppercase letters
- Variables
  - Lowercase letters

$$\frac{\forall x. P(x) \rightarrow Q(x) \qquad P(A)}{Q(A)}$$

Syllogism

All men are mortal.

Socrates is a man.

Socrates is mortal.

First-Order Resolution

1. Convert to clausal form

$$\frac{\forall x. \neg P(x) \vee Q(x) \qquad P(A)}{Q(A)}$$

2. Unification

Substitute A for x, still true

$$\frac{\neg P(A) \vee Q(A) \qquad P(A)}{Q(A)}$$

$$\frac{\forall x. P(x) \rightarrow Q(x) \qquad P(A)}{Q(A)}$$

3. Propositional resolution

Clausal Form

- Like CNF in outer structure
- No quantifiers

$$\forall x.\ \exists y.\ P(x) \rightarrow R(x,y)$$

$$\Downarrow$$

$$\neg\, P(x) \lor R(x, f(x))$$

Converting to Clausal Form

1. Eliminate arrows

$$\alpha \leftrightarrow \beta \quad \Rightarrow \quad (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$$

$$\alpha \rightarrow \beta \quad \Rightarrow \quad \neg \alpha \vee \beta$$

2. Drive in negation

$$\neg (\alpha \vee \beta) \quad \Rightarrow \quad \neg \alpha \wedge \neg \beta$$

$$\neg (\alpha \wedge \beta) \quad \Rightarrow \quad \neg \alpha \vee \neg \beta$$

$$\neg \neg \alpha \quad \Rightarrow \quad \alpha$$

$$\neg \forall x.\ \alpha \quad \Rightarrow \quad \exists x.\ \neg\alpha$$

$$\neg \exists x.\ \alpha \quad \Rightarrow \quad \forall x.\ \neg\alpha$$

3. Rename variables apart

$$\forall x.\ \exists y.\ (\neg P(x) \vee \exists x.\ Q(x,y)) \quad \Rightarrow \quad \forall x_1.\ \exists y_2.\ (\neg P(x_1) \vee \exists x_3.\ Q(x_3,y_2))$$

Converting to Clausal Form

4. Skolemize

Substitute a new name for each existential variable

$$\exists x.\ P(x) \rightarrow P(Fred)$$

$$\exists xy.\ R(x,y) \rightarrow R(Thing1, Thing2)$$

$$\exists x.\ P(x) \wedge Q(x) \rightarrow P(Fleep) \wedge Q(Fleep)$$

$$(\exists x.\ P(x)) \wedge (\exists x. Q(x)) \rightarrow P(Frog) \wedge Q(Grog)$$

$$\exists y.\ \forall x.\ Loves(x,y) \rightarrow \forall x.\ Loves\ (x, Englebert)$$

Substitute a new function of all universal vars in outer scopes

$$\forall x.\ \exists y.\ Loves(x,y) \rightarrow \forall x.\ Loves\ (x, Beloved(x))$$

$$\forall x.\ \exists y.\ \forall z.\ \exists w.\ P(x, y, z) \wedge R(y, z, w) \rightarrow P(x, f(x), z) \wedge R(f(x), z, g(x, z))$$

Converting to Clausal Form

5. Drop universal quantifiers

$$\forall x.\ Loves(x, Beloved(x)) \rightarrow Loves(x, Beloved(x))$$

6. Distribute or over and; return clauses

$$P(z) \lor (Q(z, w) \land R(w, z)) \rightarrow \{\{P(z), Q(z, w)\}, \{P(z), R(w, z)\}\}$$

7. Rename the variables in each clause

$$\{\{P(z), Q(z, w)\}, \{P(z), R(w, z)\}\} \rightarrow \{\{P(z_1), Q(z_1, w_1)\}, \{P(z_2), R(w_2, z_2)\}\}$$

Example: Converting to Clausal Form

- John owns a dog.

  $\exists x.\ D(x) \land O(J, x)$

  ➔ $D(Fido) \land O(J, Fido)$

- Anyone who owns a dog is a lover-of-animals.

  $\forall x.\ (\exists y.\ D(y) \land O(x, y)) \rightarrow L(x)$

  ➔ $\forall x.\ (\neg\exists y.\ D(x) \land O(x, y)) \lor L(x)$

  ➔ $\forall x.\ \forall y.\ \neg(D(y) \land O(x, y)) \lor L(x)$

  ➔ $\forall x.\ \forall y.\ \neg D(y) \lor \neg O(x, y) \lor L(x)$

  ➔ $\neg D(y) \lor \neg O(x, y) \lor L(x)$

Example: Converting to Clausal Form

- Lovers-of-animals do not kill animals.

  $\forall x.\ L(x) \rightarrow (\forall y.\ A(y) \rightarrow \neg K(x, y))$

  ➔ $\forall x.\ \neg L(x) \lor (\forall y.\ A(y) \rightarrow \neg K(x, y))$

  ➔ $\forall x.\ \neg L(x) \lor (\forall y.\ \neg A(y) \lor \neg K(x, y))$

  ➔ $\neg L(x) \lor \neg A(y) \lor \neg K(x, y)$

- Either Jack killed Tuna or curiosity killed Tuna.

  $K(J, T) \lor K(C, T)$

- Tuna is a cat.

  $C(T)$

Example: Converting to Clausal Form

- All cats are animals.

  $\forall x.\ C(x) \rightarrow A(x)$

  ➔ $\forall x.\ \neg C(x) \vee A(x)$

  ➔ $\neg C(x) \vee A(x)$

First-Order Resolution

1. Convert to Clausal Form

$$\forall x. \neg P(x) \vee Q(x)$$

$$\frac{P(A)}{Q(A)}$$

$$\forall x.P(x) \to Q(x)$$

$$\frac{P(A)}{Q(A)}$$

2. Unification

Substitute A for x, still true

$$\neg P(A) \vee Q(A)$$

$$\frac{P(A)}{Q(A)}$$

3. Propositional resolution

The key is ***finding the correct substitutions*** *for the variables.*

Substitutions

- An atomic sentence (ex: P(x, f(y), B)

Substitution instances	Substitution $\{v_1/t_1, \dots, v_n/t_n\}$	Comment

Substitutions

- An atomic sentence (ex: P(x, f(y), B)

Substitution instances	Substitution $\{v_1/t_1, \ldots, v_n/t_n\}$	Comment
P(z, f(w), B)	{x/z, y/w}	Alphabetic variant

Substitutions

- An atomic sentence (ex: P(x, f(y), B)

Substitution instances	Substitution $\{v_1/t_1, \ldots, v_n/t_n\}$	Comment
P(z, f(w), B)	{x/z, y/w}	Alphabetic variant
P(x, f(A), B)	{y/A}	

Substitutions

- An atomic sentence (ex: P(x, f(y), B)

Substitution instances	Substitution $\{v_1/t_1, \ldots, v_n/t_n\}$	Comment
P(z, f(w), B)	{x/z, y/w}	Alphabetic variant
P(x, f(A), B)	{y/A}	
P(g(z), f(A), B)	{x/g(z), y/A}	

Substitutions

- An atomic sentence (ex: P(x, f(y), B)

Substitution instances	Substitution $\{v_1/t_1, \ldots, v_n/t_n\}$	Comment
P(z, f(w), B)	{x/z, y/w}	Alphabetic variant
P(x, f(A), B)	{y/A}	
P(g(z), f(A), B)	{x/g(z), y/A}	
P(C, f(A), B)	{x/C, y/A}	Ground instance

Applying a substitution:

- P(x, f(y), B) {y/A} = P(x, f(A), B)
- P(x, f(y), B) {y/A, x/y} = P(A, f(A), B)

Unification

- Expressions ω_1 and ω_2 are unifiable iff there exists a substitution s such that $\omega_1 s = \omega_2 s$.
- Let $\omega_1 = x$ and $\omega_2 = y$, the following are unifiers.

s	ω_1 s	ω_2 s
{y/w}	x	x
{x/y}	y	y
{x/f(f(A)), y/f(f(A))}	f(f(A))	f(f(A))
{x/A, y/A}	A	A

Most General Unifier

g is a **most general unifier (MGU)** of ω_1 and ω_2 iff for all unifiers s, there exists s′ such that $\omega_1\, s = (\omega_1\, g)\, s'$ and $\omega_2\, s = (\omega_2\, g)\, s'$.

ω_1	ω_2	MGU
P(x)	P(A)	{x/A}

Most General Unifier

g is a **most general unifier (MGU)** of ω_1 and ω_2 iff for all unifiers s, there exists s′ such that $\omega_1\, s = (\omega_1\, g)\, s'$ and $\omega_2\, s = (\omega_2\, g)\, s'$.

ω_1	ω_2	MGU
P(x)	P(A)	{x/A}
P(f(x), y, g(x))	P(f(x), x, g(x))	{y/x} or {x/y}
P(f(x), y, g(y))	P(f(x), z, g(x))	{y/x, z/x}
P(g(f(v)), g(u))	P(x, x)	{x/g(f(v)), u/f(v)}

Unification Algorithm

```
Unify (Expr x, Expr y, Subst s) {
    if s = fail, return fail
    else if x = y, return s
    else if x is a variable, return unify-var(x, y, s)
    else if y is a variable, return unify-var(y, x, s)
    else if x is a predicate or function application,
          if y has the same operator,
                return unify(args(x), args(y), s)
          else return fail
    else  // x and y have to be lists
          return unify(rest(x), rest(y), unify(first(x), first(y), s))
}
```

Unify-var Subroutine

```
Unify-var(Variable var, Expr x, Subst s) {
     if var is bound to val in s,
            return unify (val, x, s)
     else if x is bound to val in s,
            return unify-var(var, val, s)
     else if var occurs anywhere in (x s),
            return fail
     else
            return add({var/x}, s)
}
```

What is MGU of P(x, B, B) and P(A, y, z)?

- {x/A, y/B, z/B}
- {x/A, y/A, z/A}
- {x/B, y/A, z/A}
- {P/x, x/y, z/y}

What is MGU of P(x, f(x)) and P(x, x)

- {x/P(f(x))}
- {x/f(x)}
- {P/f}
- None of the above

Some Examples

g is a **most general unifier (MGU)** of ω_1 and ω_2 iff for all unifiers s, there exists s' such that $\omega_1\, s = (\omega_1\, g)\, s'$ and $\omega_2\, s = (\omega_2\, g)\, s'$.

ω_1	ω_2	MGU
P(x, B, B)	P(A, y, z)	{x/A, y/B, z/B}
P(x, f(x))	P(x, x)	No MGU!

15 min. break

Resolution with Variables

$\mathrm{MGU}(\varphi, \psi) = \theta$

$\theta = \{x/A\}$

$$\frac{\begin{array}{c}\alpha \vee \varphi \\ \neg \psi \vee \beta\end{array}}{(\alpha \vee \beta)\,\theta}$$

$$\frac{\begin{array}{ll}\forall xy. & P(x) \vee Q(x, y) \\ \forall z. & \neg P(A) \vee R(B, z)\end{array}}{(Q(x, y) \vee R(B, z))\,\theta}$$

$$(Q(A, y) \vee R(B, z))$$

$$\frac{\begin{array}{ll}\forall xy. & P(x) \vee Q(x, y) \\ \forall x. & \neg P(A) \vee R(B, x)\end{array}}{(Q(A, y) \vee R(B, A))\,\theta}$$

$$(Q(A, y) \vee R(B, A))$$

Resolution with Variables

$$\mathrm{MGU}(\varphi, \psi) = \theta$$

$$\frac{\begin{array}{c}\alpha \vee \varphi \\ \neg \psi \vee \beta\end{array}}{(\alpha \vee \beta)\,\theta}$$

$$\theta = \{x/A\}$$

$$\frac{\begin{array}{l}\forall xy.\ P(x) \vee Q(x, y) \\ \forall z.\ \neg P(A) \vee R(B, z)\end{array}}{(Q(x, y) \vee R(B, z))\,\theta}$$

$$(Q(A, y) \vee R(B, z))$$

$$\theta = \{x_1/A\}$$

$$\frac{\begin{array}{l}\forall xy.\ P(x_1) \vee Q(x_1, y_1) \\ \forall x.\ \neg P(A) \vee R(B, x_2)\end{array}}{(Q(x_1, y_1) \vee R(B, z_1))\,\theta}$$

$$(Q(A, y_1) \vee R(B, z_1))$$

Curiosity Killed the Cat

1	D(Fido)	a			
2	O(J, Fido)	a			
3	¬ D(y) v ¬ O(x, y) v L(x)	b			
4	¬ L(x) v ¬ A(y) v ¬ K(x, y)	c			
5	K(J, T) v K(C, T)	d			
6	C(T)	e			
7	¬ C(x) v A(x)	f			
8	¬ K(C, T)	Neg	12	¬ L(J)	10, 11
9	K(J, T)	5, 8	13	¬ D(y) v ¬ O(J,y)	3, 12 {x/J}
10	A(T)	6, 7 {x/T}	14	¬ D(Fido)	13, 2 {y/Fido}
11	¬ L(J) v ¬ A(T)	4, 9 {x/J, y/T}	15	•	14, 1

Proving validity

- How do we use resolution refutation to prove something is valid?
- Normally, we prove a sentence is entailed by the set of axioms.
- Valid sentences are entailed by the empty set of sentences.
- To prove validity by refutation, negate the sentence and try to derive contradiction.

Proving Validity: Example

- Syllogism

  $(\forall x.\ P(x) \rightarrow Q(x)) \land P(A) \rightarrow Q(A)$

- Negate and convert to clausal form

  $\neg\ ((\forall x.\ P(x) \rightarrow Q(x)) \land P(A) \rightarrow Q(A))$

  $\neg(\neg(\forall x.\ \neg P(x) \lor Q(x)) \lor \neg P(A) \lor Q(A))$

  $(\forall x.\ \neg P(x) \lor Q(x)) \land P(A) \land \neg Q(A)$

  $(\neg P(x) \lor Q(x)) \land P(A) \land \neg Q(A)$

Proving Validity: Example

- Do proof

  $(\neg P(x) \vee Q(x)) \wedge P(A) \wedge \neg Q(A)$

1	$\neg P(x) \vee Q(x)$	Given
2	$P(A)$	Given
3	$\neg Q(A)$	Given
4	$Q(A)$	1, 2
5	•	3, 4

Miscellaneous Logic Topics

- Factoring
- Green's trick
- Equality
- Completeness and decidability

Binary Resolution

- Binary resolution matches one literal from each clause.
- Binary resolution is **not** complete.
- Can we get a contradiction from these clauses?

$$P(x) \vee P(y)$$

$$\neg P(y) \vee \neg P(w)$$

We should!

However, all we can get is $P(x) \vee \neg P(w)$. From here, all we can do is getting back to one of the original clauses.

Factoring

- Generalized resolution lets you resolve away multiple literals at once.
- It's simpler to introduce a new inference rule, called **factoring**.

$$\frac{\alpha \vee \beta \vee \gamma}{(\alpha \vee \gamma)\,\theta} \qquad \mathrm{MGU}(\alpha, \beta) = \theta$$

- Example

$$\frac{Q(y) \vee P(x, y) \vee P(v, A)}{}$$

Binary resolution with factoring is complete.

Green's Trick

- Use resolution to get answers to existential queries.

$$\exists x. \text{Mortal}(x)$$

1	¬ Man(x) ∨ Mortal(x)	
2	Man(Socrates)	
3	¬ Mortal(x) ∨ **Answer(x)**	
4	Mortal(Socrates)	1, 2
5	Answer(Socrates)	3, 5

Equality

- Special predicate in syntax and semantics; need to add something to our proof system.
- Add another special inference rule called **paramodulation**.
- Instead, we will axiomatize equality as an equivalence relation.

  $\forall x.\ Eq(x, x)$

  $\forall xy.\ Eq(x, y) \rightarrow Eq(y, x)$

  $\forall xyz.\ Eq(x, y) \land Eq(y, z) \rightarrow Eq(x, z)$
- For all predicate, allow substitutions

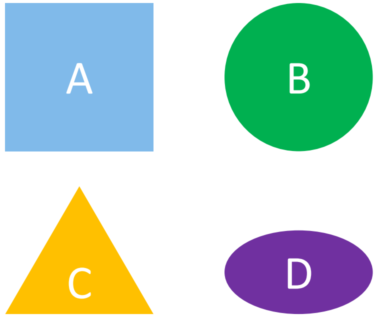
  $\forall xy.\ Eq(x,y) \rightarrow (P(x) \rightarrow P(y))$

Proof Example

- What is the hat of A?
- Axioms in FOL + equality axioms
- Desired conclusion

  $\exists x.\ hat(A) = x$

- Use Green's trick to get the binding of x.



KB_4

- Above(A, C)
- Above(B, D)
- **¬∃x. Above(x, A)**
- **¬∃x. Above(x, B)**
- $\forall xy.\ Above(x,y) \rightarrow hat(y) = x$
- $\forall x.(\neg\exists y.\ Above(y,x)) \rightarrow hat(x) = x$

What is the hat of A?

Step	Formula	Derivation
1	Above(A, C)	Given
2	Above(B, D)	Given
3	¬ Above(x, A)	Given
4	¬ Above(x, B)	Given
5	¬ Above(x, y) ∨ Eq(hat(y), x)	Given
6	Above(sk(x), x) ∨ Eq(hat(x), x)	Given
7	Eq(x, x)	Given
8	¬ Eq(x, y) ∨ ¬ Eq(y, z) ∨ Eq(x, z)	Given
9	¬ Eq(x, y) ∨ Eq(y, x)	Given
10	**¬ Eq(hat(A), x)** ∨ **Answer(x)**	Negated Conclusion

What is the hat of A?

Step	Formula	Derivation
1	Above(A, C)	Given
2	Above(B, D)	Given
3	¬ Above(x, A)	Given
4	¬ Above(x, B)	Given
5	¬ Above(x, y) ∨ Eq(hat(y), x)	Given
6	Above(sk(x), x) ∨ Eq(hat(x), x)	Given
7	Eq(x, x)	Given
8	¬ Eq(x, y) ∨ ¬ Eq(y, z) ∨ Eq(x, z)	Given
9	¬ Eq(x, y) ∨ Eq(y, x)	Given
10	**¬ Eq(hat(A), x) ∨ Answer(x)**	Negated Conclusion
11	Above(sk(A), A) ∨ Answer (A)	6, 10 {x/A}
12	Answer(A)	11, 3 {x/sk(A)}

What is the hat of A?

Step	Formula	Derivation
1	Above(A, C)	Given
2	Above(B, D)	Given
3	¬ Above(x, A)	Given
4	¬ Above(x, B)	Given
5	¬ Above(x, y) ∨ Eq(hat(y), x)	Given
6	Above(sk(x), x) ∨ Eq(hat(x), x)	Given
7	Eq(x, x)	Given
8	¬ Eq(x, y) ∨ ¬ Eq(y, z) ∨ Eq(x, z)	Given
9	¬ Eq(x, y) ∨ Eq(y, x)	Given
10	**¬ Eq(hat(D), x)** ∨ **Answer(x)**	Negated Conclusion

What is the hat of D?

Step	Formula	Derivation
1	Above(A, C)	Given
2	Above(B, D)	Given
3	¬ Above(x, A)	Given
4	¬ Above(x, B)	Given
5	¬ Above(x, y) ∨ Eq(hat(y), x)	Given
6	Above(sk(x), x) ∨ Eq(hat(x), x)	Given
7	Eq(x, x)	Given
8	¬ Eq(x, y) ∨ ¬ Eq(y, z) ∨ Eq(x, z)	Given
9	¬ Eq(x, y) ∨ Eq(y, x)	Given
10	**¬ Eq(hat(D), x) ∨ Answer(x)**	Negated Conclusion
11	¬ Above(x, D) ∨ Answer(x)	5, 10 {x1/x}
12	Answer(B)	11, 2 {x/B}

Who is Jane's Lover?

- Jane's lover drives a red car.
- Fred is the only person who drives a red car.
- Who is Jane's lover?

1	Drives(lover(Jane))	
2	¬ Drives(x) ∨ Eq(x, Fred)	
3	¬ Eq(lover(Jane), x) ∨ Answer(x)	
4	Eq(lover(Jane), Fred)	1, 2 {x/lover(Jane)}
5	Answer(Fred)	3, 4 {x/Fred}

Completeness and Decidability

- Definition of a complete proof system
  - If KB entails S, we can prove S from KB.
- Gödel’s completeness theorem (1929)
  - There exists a complete proof system for FOL.
- Robin’s completeness (1965)
  - Resolution refutation is a complete proof system for FOL.
- FOL is semi-decidable.
  - If the desired conclusion follows from the premises, resolution refutation will find a contradiction.
    - If there is a proof, we will halt with it.
    - If not, maybe we will halt, maybe not.

Adding Arithmetic

- Gödel's incompleteness theorem (1931)
  - There is no consistent, complete proof system for FOL+arithmetic.
  - There are sentences that are true, but not provable.
  - There are sentences that are provable, but not true.
- Arithmetic gives you the ability to construct code-names for sentences within the logic.
  - ex) P = "P is not provable"
    - If P is true: it is not provable (incomplete).
    - If P is false: it is provable (inconsistent).

Expressing Uniqueness

- Sometimes we want to say that there is a single, unique object that satisfies a certain condition.
- "There exists a unique x such that king(x) is true."
  - $\exists x.\ king(x) \land \forall y.\ (king(y) \rightarrow x=y)$
  - $\exists x.\ king(x) \land \neg\exists y.\ (king(y) \land \rightarrow x!=y)$
  - $\exists! x.\ king(x)$

Logic in the World

- Information encoded formally in web pages
- Business rules
- Airfare pricing

Airfare Pricing

- Ignore finding the best itinerary for now.
- Given an itinerary, what is the least amount we can pay for it?
- You cannot just add up prices for the flight legs due to different prices for different flights in various combinations and circumstances.

Fare Restrictions

- The passenger is under 2 or over 65.
- The passenger is accompanying another passenger who is paying full fare.
- There are no flights during rush hour (defined in local time).
- The itinerary stays over a Saturday night.
- Layovers are legal: not too short; not too long.
- …

Ontology

"Ontology is the science of something and of nothing, of being and not-being, of the thing and the mode of the thing, of substance and accident.

- Gottfried Wilhelm Leibniz (a philosopher from the 17^{th} century)

Ontology

- What kinds of things are there in the world?
- What are their properties and relations?

Airfare Domain Ontology

- passenger
- flight
- city
- airport
- terminal
- flight segment (list of flights, to be flown all in one “day”)
- itinerary (a passenger and list of flight segments)
- list
- number

Representing Properties

- Object P is red.
  - Red(P)
  - Color(P, Red)
  - color(P) = Red
  - Property(P, Color, Red)
- All the blocks in stack S are the same color.

  $\exists c.\ \forall b.\ In(b, S) \rightarrow Color(b, c)$
- All the blocks in stack S have the same properties.
  - $\forall p.\ \exists v.\ \forall b.\ In(b, S)\ \forall\ Property(b, p, v)$

Basic Relations

- Age(passenger, number)
- Nationality(passenger, country)
- Wheelchair(passenger)
- Origin(flight, airport)
- Passenger(itinerary, passenger)
- …

Example: Fred

- Age(Fred, 47)
- Nationality(Fred, US)
- ¬ Wheelchair(Fred)

Defined Relations

- Define complex relations in terms of basic ones
  - $\forall i.\ P(i) \land Q(i) \rightarrow \text{Qualifies37}(i)$
- Implication rather than equivalence
  - Easier to specify definitions in pieces
    - $\forall i.\ P(i) \land S(i) \rightarrow \text{Qualifies37}(i)$
  - Can’t use the other direction
    - $\text{Qualifies37}(i) \rightarrow ?$
  - If you need it, write the equivalence.
    - $\forall i.\ (P(i) \land Q(i)) \lor (P(i) \land S(i)) \leftrightarrow \text{Qualifies37}(i)$

Infant Fare

$$\forall i, a, p.\ Passenger(i, p) \wedge Age(p, a) \wedge a < 2 \rightarrow InfantFare(i)$$

$$\forall i\ (\exists p, a.\ Passenger(i, p) \wedge Age(p, a) \wedge a < 2) \rightarrow InfantFare(i)$$

- $a < 2$?
  - axiomatize arithmetic
  - build it in to theorem prover
    - $P(a) \vee (3>2) \vee (a > 1+2) \Rightarrow P(a) \vee (a > 3)$

Well-Formed Segment

- The departure and arrival airports match up correctly.
- The layovers (gaps between arriving in an airport and departing from it) are not too short.
- The layovers are not too long.

Lists in Logic

- Nil
  - constant
- cons
  - function
- cons(A, cons(B, Nil))
  - list with two elements
- $\forall$. LengthOne(cons(x, Nil))
  - $\forall l, x.\ l = cons(x, Nil) \rightarrow LengthOne(l)$
  - $\forall l.\ (\exists x.\ l=cons(x, Nil)) \rightarrow LengthOne(l)$

Well-Formed Segment: Base Case

- Define recursively, going down the list of flights.
- Any segment with 1 flight is well-formed.

$$\forall f.\ \mathrm{WellFormed}(\mathrm{cons}(f, \mathrm{Nil}))$$

Well-Formed Segment: Recursion

- A flight segment with at least two flights is well-formed if
  - first two flights are contiguous
  - layover time between first two flights is legal
  - rest of the flight segment is well-formed

$$\forall f_1, f_2, r.\ \mathrm{Contiguous}(f_1, f_2) \land \mathrm{LegalLayover}(f_1, f_2) \land \mathrm{WellFormed}(\mathrm{cons}(f_2, r)) \rightarrow \mathrm{WellFormed}(\mathrm{cons}(f_1, \mathrm{cons}(f_2, r)))$$

Helper Relations

- Flights are contiguous if the arrival airport of the first is the same as the departure airport of the second.

  $\forall f_1, f_2. (\exists\ c.\ Destination(f_1, c) \land Origin(f_2, c)) \rightarrow Contiguous(f_1, f_2)$

- Layovers are legal if they are not too short and not too long.

  $\forall\ f_1, f_2.\ LayoverNotTooShort(f_1, f_2) \land LayoverNotTooLong(f_1, f_2)$
  $\rightarrow LayoverLegal(f_1, f_2)$

Not Too Short

- A layover is not too short if it is more than 30 minutes long.

  $\forall f_1, f_2. (\exists\, t_1, t_2. \text{ArrivalTime}(f_1, t_1) \wedge \text{DepartureTime}(f_2, t_2) \wedge (t_2 - t_1 > 30)) \rightarrow \text{LayoverNotTooShort}(f_1, f_2)$

Not Too Long

- A layover is not too long if it's less than three hours.

  $\forall f_1, f_2. (\exists\, t_1, t_2. \mathrm{ArrivalTime}(f_1, t_1) \land \mathrm{DepartureTime}(f_2, t_2) \land \quad (t_2 - t_1 < 180)) \rightarrow \mathrm{LayoverNotTooLong}(f_1, f_2)$

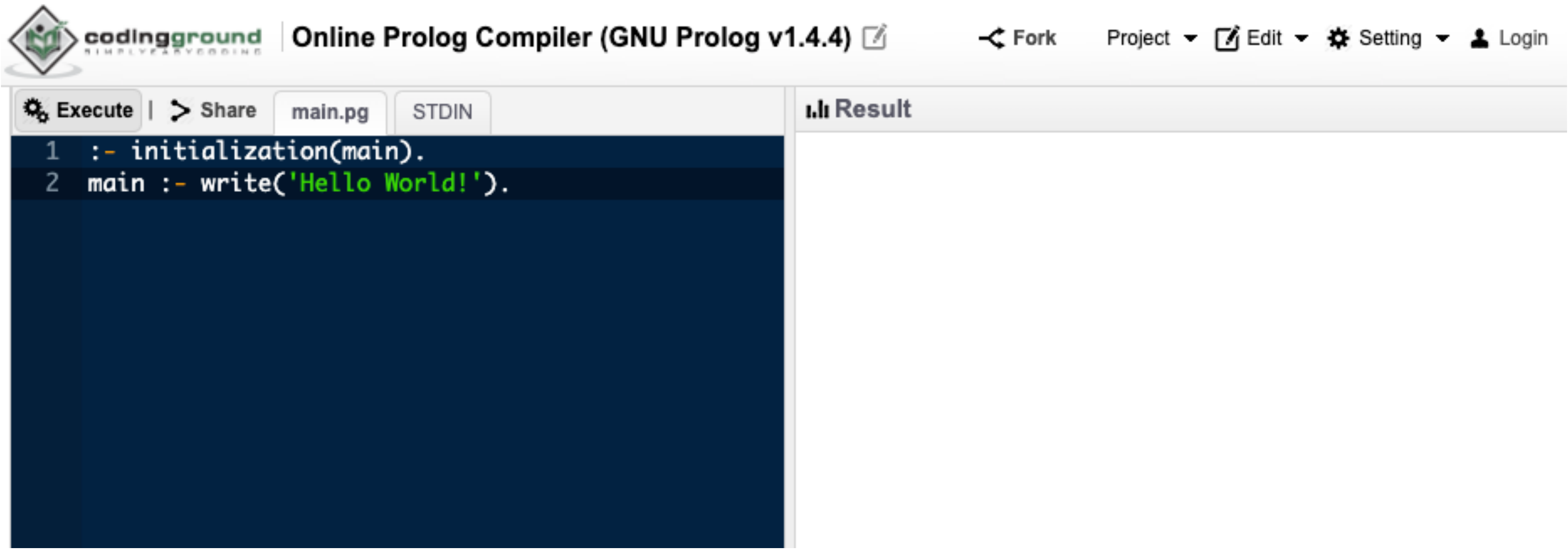
- A layover is also not too long if there was no other way to make the next leg of your journey sooner.

  $\forall f_1, f_2. (\exists o, d, t_2, \mathrm{Origin}(f_2, o) \land \mathrm{Destination}(f_2, d) \land \mathrm{DepartureTime}(f_2, t_2) \land$

  $\neg\exists f_3, t_3. (\mathrm{Origin}(f_3, o) \land \mathrm{Destination}(f_3, d) \land \mathrm{DepartureTime}(f_3, t_3) \land (t_3 < t_2) \land \mathrm{LayoverNotTooShort}(f_1, f_3))) \rightarrow \mathrm{LayoverNotTooLong}(f_1, f_2)$

Prolog

- A logic programming language associated with artificial intelligence and computational linguistics.
  - https://www.swi-prolog.org/
  - https://www.tutorialspoint.com/execute_prolog_online.php



Higher-Older Logic

- FOL only allows to quantify over variables, and variables can only range over objects.
- HOL allows us to quantify over relations.
- Example (quantify over functions)
  - "Two functions are equal iff they produce the same value for all arguments."
  - $\forall f, g.\ (f=g) \leftrightarrow (\forall x.\ f(x) = g(x))$
- Example (quantify over predicates)
  - $\forall r.\ \text{transitive}(r) \rightarrow (\forall x, y, z.\ r(x,y) \land r(y,z) \rightarrow r(x,z))$
- More expressive, but undecidable.

Probability

Chapter 13

Probability Distributions

- Unobserved random variables have distributions

$P(T)$

T	P
hot	0.5
cold	0.5

$P(W)$

W	P
sun	0.6
rain	0.1
fog	0.3
meteor	0.0

- A distribution is a TABLE of probabilities of values
- A probability (lower case value) is a single number

$$P(W = rain) = 0.1$$

- Must have: $\forall x \;\; P(X = x) \geq 0$ and $\sum_x P(X = x) = 1$

Shorthand notation:

$$P(hot) = P(T = hot),$$
$$P(cold) = P(T = cold),$$
$$P(rain) = P(W = rain),$$
$$\dots$$

OK *if* all domain entries are unique

Joint Distributions

- A *joint distribution* over a set of random variables: $X_1, X_2, \ldots X_n$ specifies a real number for each assignment (or *outcome*):

$$P(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n)$$

$$P(x_1, x_2, \ldots x_n)$$

  - Must obey:

$$P(x_1, x_2, \ldots x_n) \geq 0$$

$$\sum_{(x_1, x_2, \ldots x_n)} P(x_1, x_2, \ldots x_n) = 1$$

$$P(T, W)$$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

- Size of distribution if n variables with domain sizes d?
  - For all but the smallest distributions, impractical to write out!

Marginal Distributions

- Marginal distributions are sub-tables which eliminate variables.
- Marginalization (summing out): Combine collapsed rows by adding.

$P(T,W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

$$P(t) = \sum_s P(t,s)$$

$$P(s) = \sum_t P(t,s)$$

$P(T)$

T	P
hot	0.5
cold	0.5

$P(W)$

W	P
sun	0.6
rain	0.4

$$P(X_1 = x_1) = \sum_{x_2} P(X_1 = x_1, X_2 = x_2)$$

Conditional Distributions

- Conditional distributions are probability distributions over some variables given fixed values of others.

$$P(x_1|x_2) = \frac{P(x_1, x_2)}{P(x_2)}$$

$$= \frac{P(x_1, x_2)}{\sum_{x_1} P(x_1, x_2)}$$

Conditional Distributions

$P(W|T)$

$$P(W|T = hot)$$

W	P
sun	0.8
rain	0.2

$$P(W|T = cold)$$

W	P
sun	0.4
rain	0.6

Joint Distribution

$$P(T, W)$$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

Normalization Trick

$$P(T, W)$$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

$$P(W = s|T = c) = \frac{P(W = s, T = c)}{P(T = c)}$$
$$= \frac{P(W = s, T = c)}{P(W = s, T = c) + P(W = r, T = c)}$$
$$= \frac{0.2}{0.2 + 0.3} = 0.4$$

$$P(W = r|T = c) = \frac{P(W = r, T = c)}{P(T = c)}$$
$$= \frac{P(W = r, T = c)}{P(W = s, T = c) + P(W = r, T = c)}$$
$$= \frac{0.3}{0.2 + 0.3} = 0.6$$

$$P(W|T = c)$$

W	P
sun	0.4
rain	0.6

The Product Rule

- Sometimes have conditional distributions but want the joint

$$P(y)P(x|y) = P(x,y)$$

$$\Updownarrow$$

$$P(x|y) = \frac{P(x,y)}{P(y)}$$

The Chain Rule

- More generally, can always write any joint distribution as an incremental product of conditional distributions

$$P(x_1, x_2, x_3) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)$$

$$P(x_1, x_2, \ldots x_n) = \prod_i P(x_i|x_1 \ldots x_{i-1})$$

Bayes' Rule

- Two ways to factor a joint distribution over two variables:

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

- Dividing, we get:

$$P(x|y) = \frac{P(y|x)}{P(y)}P(x)$$

- Why is this at all helpful?
  - Let us build one conditional from its reverse
  - Often one conditional is tricky but the other one is simple
  - Foundation of many systems we'll see later (*e.g.*, ASR, MT)

Inference with Bayes' Rule

- Example: Diagnostic probability from causal probability:

$$P(\text{cause}|\text{effect}) = \frac{P(\text{effect}|\text{cause})P(\text{cause})}{P(\text{effect})}$$

- Example:
  - M: meningitis, S: stiff neck

$$\left.\begin{aligned} P(+m) &= 0.0001 \\ P(+s|+m) &= 0.8 \\ P(+s|-m) &= 0.01 \end{aligned}\right\} \text{Example givens}$$

$$P(+m|+s) = \frac{P(+s|+m)P(+m)}{P(+s)} = \frac{P(+s|+m)P(+m)}{P(+s|+m)P(+m) + P(+s|-m)P(-m)} = \frac{0.8 \times 0.0001}{0.8 \times 0.0001 + 0.01 \times 0.999}$$

  - Note: posterior probability of meningitis still very small.
  - Note: you should still get stiff necks checked out! Why?

Questions?

Attendance

- We use QR codes to save time.
  - Submit a google form through the QR code.

- How to scan a QR code?
  - Android
    - Built-in camera (Google lens)
    - https://play.google.com/store/apps/details?id=com.camvision.qrcode.barcode.reader&hl=en_US&gl=US&pli=1
  - iOS
    - Built-in camera
    - https://apps.apple.com/us/app/qr-code-reader/id1200318119