Today's topics

- Applications of predicate logic
- Nested quantifiers

Logic programming enables automated reasoning

Prolog

- Programming in logic
- Developed in the 1970s for AI purposes

Datalog

- Logical formalization of databases
- Developed in the 1980s

For our purposes, we can consider Prolog and Datalog to be the same, though in reality they have very important differences.

Two main constructs:

Lower case = constant

- Facts
  - instructor(litman, cs441)
  - student(smith, cs441)
- Rules
  - teaches(P,S) :- instructor(P,C), student(S,C)

Upper case = variable

Rules and facts define predicates

Facts define predicates by explicitly listing elements that satisfy those predicates

- "Prof. Litman is the instructor for CS441" ≡ instructor(litman, cs441)

Rules define predicates by combining previously specified predicates

- "Professors teach the students enrolled in the courses for which they are the instructor" ≡
  teaches(P,S) :- instructor(P,C), student(S,C)

Prolog is an environment that lets us issue queries to determine which predicates are true!

A Security Example

grant(U, projector) :- located(U, 104), role(U, presenter)
located(U, R) :- owns(U, D), dev_loc(D, R)

role(bob, presenter)	owns(alice, laptop12)	dev_loc(laptop12, 104)
role(carol, presenter)	owns(bob, pda23)	dev_loc(pda23, 104)
	owns(carol, cell42)	dev_loc(cell42, 104)

Knowledge base

Can Bob run the projector?

- *Query:* ?grant(bob, projector)
- *Solution:* true

Who is in room 104?

- *Query:* ?location(X, 104)
- *Solution:* alice, bob, carol

Write and evaluate the following queries

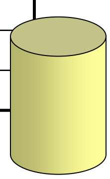
```
grant(U, projector) :- located(U, 104), role(U, presenter)
located(U, R) :- owns(U, D), dev_loc(D, R)

role(bob, presenter)   owns(alice, laptop12)  dev_loc(laptop12, 104)
role(carol, presenter) owns(bob, pda23)       dev_loc(pda23, 104)
                       owns(carol, cell42)    dev_loc(cell42, 104)
```

- Can Alice use the projector?
  - ?grant(alice, projector)
  - false
- Can Carol use the projector
  - ?grant(carol, projector)
  - true
- Which devices does Alice own?
  - ?owns(alice, X)
  - laptop12

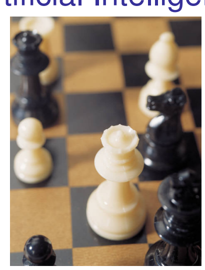
Logic programming is a useful tool!

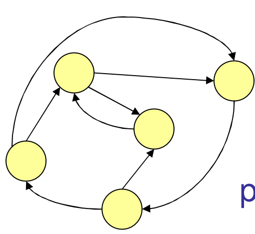
Name	Age	Phone
Alice	19	555-1234
Danielle	33	555-5353
Zach	27	555-3217
Charlie	21	555-2335



Databases

Artificial Intelligence





Route planning

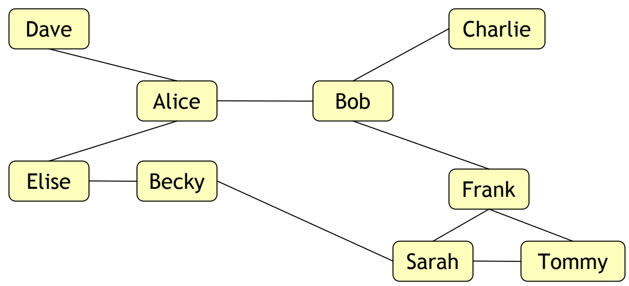
Security

Just for grins…

If you are interested in playing around with logic programming, download SWI-Prolog

- URL: http://www.swi-prolog.org/

This (free) package is a runtime environment in which you can write logic programs and evaluate queries.



Nested quantifiers!?!?

Many times, we need the ability to nest one quantifier within the scope of another quantifier

Example: All integers have an additive inverse. That is, for any integer x, we can choose an integer y such that the sum of x and y is zero.

There is no way to express this statement using only a single quantifier!

Deciphering nested quantifiers isn't as scary as it looks…

… you just read from left to right!

$$\underbrace{\forall x}_{\text{For all x...}}\ \underbrace{\exists y}_{\text{... there exists a y such that...}}\ \underbrace{\forall z}_{\text{... for all z...}}\ \underbrace{[(x + y) \times z = 0]}_{\text{... } (x + y) \times z = 0}$$

A few more examples…

$\forall x\ \forall y\ (x + y = y + x)$ This is the commutative law for addition!

- For all integers x and for all integers y, $x + y = y + x$

$\forall x\ \forall y\ \forall z\ [(x+y)+z = x+(y+z)]$ This is the associative law for addition!

- For all integers x, for all integers y, and for all integers z, $(x+y)+z = x+(y+z)$

$\exists x\ \forall y\ (x \times y = 0)$

- There exists an x such that for all y, $x \times y = 0$

Since we always read from left to right, the order of quantifiers matters!

Consider: $\forall x\, \exists y\, (x + y = 0)$

Clearly true! Just set y = -x

Transpose: $\exists y\, \forall x\, (x + y = 0)$

Not true…

Remember: As long as you read from left to right, you won't have any problems!

Many mathematical statements can be translated into logical statements with nested quantifiers

Translating mathematical expressions is often easier than translating English statements!

Steps:

1. Rewrite statement to make quantification and logical operators more explicit
2. Determine the order of in which quantifiers should appear
3. Generate logical expression

Let's try a translation…

Universal quantifier

Statement: Every real number except zero has a multiplicative inverse

$x \times y = 1$

Singular---suggestive of an existential quantifier

$\forall x$

Rewrite: For every real number x, if $x \neq 0$, then there exists a real number y such that $x \times y = 1$.

$\ldots \exists y\ (x \times y = 1)$

$(x \neq 0) \rightarrow \ldots$

More examples…

Statement: The product of any two negative integers is always positive

Statement: For any real number a, it is possible to choose real numbers b and c such that $a^2 + b^2 = c^2$

Translating quantified statements to English is as easy as reading a sentence!

Let:

- $C(x) \equiv$ x is enrolled in CS441
- $M(x) \equiv$ x has an MP3 player
- $F(x, y) \equiv$ x and y are friends
- Domain of x and y is "all students"

Statement: $\forall x\ [C(x) \rightarrow M(x) \vee (\exists y\ (F(x,y) \wedge M(y))]$

For every student x…

… if x is enrolled in CS441, then…

… x has an MP3 player…

… or there exists another student y such that…

… x and y are friends…

… and y has an MP3 player.

Translate the following expressions into English

Let:

- $O(x,y) \equiv$ x is older than y
- $F(x,y) \equiv$ x and y are friends
- The domain for variables x and y is "all students"

Statement: $\exists x\ \forall y\ O(x,y)$

Statement: $\exists x\ \exists y\ [F(x,y) \wedge \forall z\ [(y \neq z) \rightarrow \neg F(x,z)]]$

Group work!

Problem 1: Translate the following mathematical statement into predicate logic: Every even number is a multiple of 2. Assume that the predicate E(x) means "x is even."

- **Hint:** What does "x is a multiple of 2" mean algebraically?

Translating from English to a logical expression with nested quantifiers is a little bit more work…

Steps:

1. If necessary, rewrite the sentence to make quantifiers and logical operations more explicit
2. Create propositional functions to express the concepts in the sentence
3. State the domains of the variables in each propositional function
4. Determine the order of quantifiers
5. Generate logical expression

Let's try an example…

Universal quantifier

Statement: Every student has asked at least one professor a question.

Existential quantifier

Rewrite: For every person x, if x is a student, then there exists a professor whom x has asked a question.

Let:

- S(x) ≡ x is a student
- P(x) ≡ x is a professor
- Q(x,y) ≡ x has asked y a question

Domains for x and y are "all people"

Translation: $\forall x\ (S(x) \rightarrow \exists y\ [P(y) \wedge Q(x,y)])$

Translate the following from English

Statement: There is a man who has tasted every type of beer.

Domain: all people

Let:

Domain: all drinks

Domains: x = all people,
y = all drinks

Translation:

Negating expression with nested quantifiers is actually pretty easy…

… you just repeatedly apply DeMorgan's laws!

$\neg[\exists x\ (M(x) \land \forall y\ [B(y) \rightarrow T(x,y)])]$

$a \rightarrow b \equiv \neg a \lor b$

In English: For all people x, if x is a man, then there exists some type beer that x has not tasted.

Negate $\forall x\ (S(x) \rightarrow \exists y\ [P(y) \land Q(x,y)])$

$\neg \forall x\ (S(x) \rightarrow \exists y\ [P(y) \land Q(x,y)])$

In English: There exists a student x such that for all people y, if y is a professor then x has not asked y a question.

Alternatively: There exists a student that has never asked any professor a question.

Group Work!

Problem 1: Translate the following English sentences into predicate logic.

a) There is a woman has tried every flavor of Ben and Jerry's ice cream.

b) Every student has at least one friend that is dating Penguins fan.

c) If a person is a parent and a man, then they are the father of some child.

Problem 2: Negate the results from Problem 1 and translate the negated expressions back into English.

Final Thoughts

- Logic programming is an interesting application of predicate logic that is used throughout computer science
- Quantifiers can be nested
  - Nested quantifiers are read left to right
  - Order is important!
  - Translation and negation work the same as they did before!
- Next lecture:
  - Rules of inference and proofs
  - Please read section 1.5 and 1.6