Last time: Inference example

Show that the hypotheses:

- It is not sunny this afternoon and it is colder than yesterday. $\neg s \wedge c$
- We will go swimming only if it is sunny. $w \rightarrow s$
- If we do not go swimming, then we will take a canoe trip. $\neg w \rightarrow t$
- If we take a canoe trip, then we will be home by sunset. $t \rightarrow h$

lead to the conclusion:

- We will be home by the sunset. h

Step	Reason
1. $\neg s \wedge c$	hypothesis
2. $\neg s$	simplification
3. $w \rightarrow s$	hypothesis
4. $\neg w$	modus tollens of 2 and 3
5. $\neg w \rightarrow t$	hypothesis
6. t	modus ponens of 4 and 5
7. $t \rightarrow h$	hypothesis
8. h	modus ponens of 6 and 7

Where:
s: "it is sunny this afternoon"
c: "it is colder than yesterday"
w: "we will go swimming"
t: "we will take a canoe trip.
h: "we will be home by the sunset."

Last time: Inference example

Show that the hypotheses:

- It is not sunny this afternoon and it is colder than yesterday. $\neg s \wedge c$
- We will go swimming only if it is sunny. $w \rightarrow s$
- If we do not go swimming, then we will take a canoe trip. $\neg w \rightarrow t$
- If we take a canoe trip, then we will … $t \rightarrow h$

lead to the conclu…

- We will be h…

What about ChatGPT?

Step	R…
1. $\neg s \wedge c$	h…
2. $\neg s$	si…
3. $w \rightarrow s$	hy…
4. $\neg w$	modus tollens of 2 and 3
5. $\neg w \rightarrow t$	hypothesis
6. t	modus ponens of 4 and 5
7. $t \rightarrow h$	hypothesis
8. h	modus ponens of 6 and 7

Where:
s: "it is sunny this afternoon"
c: "it is colder than yesterday"
w: "we will go swimming"
t: "we will take a canoe trip.
h: "we will be home by the sunset."

SP

It is not sunny this afternoon and it is colder than yesterday.

Yes, weather can change quickly and unpredictably. It's important to always be prepared for different conditions, especially when it comes to dressing appropriately for the weather. In case of cold and cloudy weather, it's a good idea to wear warm layers, such as a coat, hat, gloves, and a scarf, to keep yourself protected from the elements.

SP

We will go swimming only if it is sunny.

That sounds like a good plan. Swimming in sunny weather is more enjoyable as the warm sun can help to make the water feel even warmer. It's also easier to stay outside for longer periods when the sun is shining, compared to swimming on a cloudy or overcast day. Just remember to protect your skin from the sun by applying sunscreen and wearing a hat if you'll be spending a lot of time in the sun.

SP

If we do not go swimming, then we will take a canoe trip.

A canoe trip can be a great alternative if you can't go swimming due to weather conditions. Canoeing can be a peaceful and scenic way to enjoy nature and get some exercise. It's a good idea to check the local weather and water conditions before heading out on a canoe trip, to make sure it's safe and comfortable. Remember to bring appropriate clothing and supplies, such as a life jacket, insect repellent, and a hat, and to always follow safety guidelines and rules for canoeing in your area.

SP

If we take a canoe trip, then we will be home by sunset.

That sounds like a good plan. Being back home by sunset is a good idea for a number of reasons. For one, it helps to ensure that you have enough time to get back before it gets dark. In addition, canoeing can be tiring, and you may want to relax and rest after your trip. Being back home before sunset also gives you a chance to clean up, have a meal, and get ready for any evening activities or plans you may have. Just remember to take all necessary safety precautions while canoeing and to follow the rules of the waterway to ensure a safe and enjoyable trip.

SP

Will we be home by sunset?

SP

Will we be home by sunset?

I am an AI language model and don't have the ability to predict the future or know your current location and plans, but if your plan is to be home by sunset, then it's important to keep track of the time and plan your canoe trip accordingly. Make sure to leave enough time for your trip and any unexpected events that may occur, such as bad weather, equipment issues, or taking longer than expected to get back. It's also a good idea to bring a reliable form of light and communication with you, in case you need to return after sunset.

Add some prompt engineering…

SP

Please remember that it is not sunny this afternoon and it is colder than yesterday.
Please remember that we will go swimming only if it is sunny.
Please remember that if we do not go swimming, then we will take a canoe trip.
Please remember that if we take a canoe trip, then we will be home by sunset.
Will we be home by sunset?

It is not stated whether it is sunny or not, so it is impossible to determine with certainty if you will be home by sunset. If it is not sunny and you do not go swimming, then you will take a canoe trip and should be home by sunset. However, if it is sunny and you go swimming, it is not specified what your plans are for the rest of the day and whether you will be home by sunset.

SP

but I said that it is not sunny this afternoon

Ah, I apologize for the confusion. If it is not sunny this afternoon, then you will not go swimming and will instead take a canoe trip. If you take a canoe trip, you will be home by sunset. So, in this case, you will be home by sunset.

Split first sentence and simplify it…

SP

Please remember that it is not sunny.
Please remember that it is colder than yesterday.
Please remember that we will go swimming only if it is sunny.
Please remember that if we do not go swimming, then we will take a canoe trip.
Please remember that if we take a canoe trip, then we will be home by sunset.
Will we be home by sunset?

It is not specified whether it is sunny or not, so it is unclear whether you will go swimming or take a canoe trip. If you take a canoe trip, then you will be home by sunset.

SP

it is not sunny

If it is not sunny, then you will not go swimming and will instead take a canoe trip. In that case, you will be home by sunset.

Some recency effects it seems…

SP

Please remember that it is colder than yesterday.
Please remember that we will go swimming only if it is sunny.
Please remember that if we do not go swimming, then we will take a canoe trip.
Please remember that if we take a canoe trip, then we will be home by sunset.
Please remember that it is not sunny.
Will we be home by sunset?

Yes, if it is not sunny, and if the previous conditions are followed, then you will be home by sunset after taking a canoe trip, instead of going swimming.

One last try: explicitly ask to solve a problem.

SP

please solve this problem step by step:

Sure! Please provide the problem statement that you would like me to solve step by step.

SP

Please remember that it is not sunny this afternoon and it is colder than yesterday.
Please remember that we will go swimming only if it is sunny.
Please remember that if we do not go swimming, then we will take a canoe trip.
Please remember that if we take a canoe trip, then we will be home by sunset.
Will we be home by sunset?

Yes, if the conditions are as stated in the problem, then we will be home by sunset.

Here's a step-by-step breakdown of the reasoning:

1. It is not sunny this afternoon, so we will not go swimming.
2. If we do not go swimming, then we will take a canoe trip.
3. If we take a canoe trip, then we will be home by sunset.

Therefore, we will be home by sunset.

Overall

- ChatGPT can understand (mostly) the problem setup.

- It can derive conclusions from the knowledge given to it.

But…

Conclusions depend on order in which knowledge was given?
Sometimes knowledge is clearly given but then ignored?
When that happens, answer sounds authoritative but actually is completely wrong.

Provably sound inference engine?

Provably complete inference engine?

Overall

- ChatGPT can understand (mostly) the problem setup.

- It can derive conclusions from the knowledge given to it.

But…

Conclusions depend on order in which knowledge was given?
Sometimes knowledge is clearly given but then ignored?
When that happens, answer sounds authoritative but actually is completely wrong.

Provably sound inference engine? ✗

Provably complete inference engine? ✗

Resolution algorithm is proven sound and complete for FOL

Maybe it can help with translation

Since it seems to understand English quite well…

SP translate to first-order-logic that it is not sunny this afternoon

One possible way to translate "it is not sunny this afternoon" into first-order logic is:

¬Sunny(ThisAfternoon)

Where "¬" represents negation, "Sunny" is a predicate symbol representing the property of being sunny, and "ThisAfternoon" is a constant symbol denoting the time period in question. In this representation, the sentence says that there exists a time period called "ThisAfternoon" such that it is not the case that "ThisAfternoon" has the property of being sunny.

Maybe it can help with translation

SP

translate to first-order logic that we will go swimming only if it is sunny

Here's a translation of the sentence "We will go swimming only if it is sunny" into first-order logic:

Let "S" be a predicate that represents the statement "it is sunny".
Let "G" be a predicate that represents the statement "we will go swimming".

Then, the sentence can be translated into the following first-order logic formula:

(S -> G) ∧ ¬(G -> S)

This formula states that if it is sunny, then we will go swimming (S -> G), and it is not the case that if we are going swimming, then it is sunny (¬(G -> S)).

Maybe it can help with translation

SP

translate to first-order logic that we will go swimming only if it is sunny

Here's a translation of the sentence "We will go swimming only if it is sunny" into first-order logic:

Let "S" be a predicate that represents the statement "it is sunny".
Let "G" be a predicate that represents the statement "we will go swimming".

Then, the sentence can be translated into the following first-order logic formula:

(S -> G) ∧ ¬(G -> S)

We had G => S

This formula states that if it is sunny, then we will go swimming (S -> G), and it is not the case that if we are going swimming, then it is sunny (¬(G -> S)).

Maybe it can help with translation

R: "we will go swimming (G) only if it is sunny (S)"

Remember: R is true iff it is true to say "we will go swimming (G) only if it is sunny (S)"

We translated to G => S

G	S	G=>S	
T	T	T	went swim, was sunny. R is True.
T	F	F	went swim, was not sunny. R is False.
F	T	T	no swim. R True because it only tells us what we will do when G is true. When G is False, R is true because no matter what we do, that does not contradict R.
F	F	T	

Maybe it can help with translation

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

(S => G) ^ ~(G => S)	(from ChatGPT)
= (~S v G) ^ ~(~G v S)	(definition of =>)
= (~S v G) ^ (G ^ ~S)	(work negation inward)
= (~S ^ G ^ ~S) v (G ^ G ^ ~S)	(distributivity of ^ over v)
= (~S ^ G) v (G ^ ~S)	(remove redundancy, A^A = A)
= ~S ^ G	(A v A = A)
= "it is not sunny, and we will go swimming"	

We had G => S (= ~G v S)

ChatGPT completely wrong again! Yet sounds authoritative again!

Last time: Logic and Reasoning

- Knowledge Base (KB): contains a set of <u>sentences</u> expressed using a **knowledge representation language**
  - TELL: operator to add a sentence to the KB
  - ASK: to query the KB
- Logics are KRLs where conclusions can be drawn
  - Syntax
  - Semantics
- Entailment: KB |= a iff a is true in all worlds where KB is true
- Inference: KB $|-_i$ a = sentence a can be derived from KB using procedure *i*
  - Sound: whenever KB $|-_i$ a then KB |= a is true
  - Complete: whenever KB |= a then KB $|-_i$ a

Last Time: Syntax of propositional logic

Propositional logic is the simplest logic-

The proposition symbols P_1, P_2 etc are sentences

If S is a sentence, $\neg S$ is a sentence

If S_1 and S_2 is a sentence, $S_1 \wedge S_2$ is a sentence

If S_1 and S_2 is a sentence, $S_1 \vee S_2$ is a sentence

If S_1 and S_2 is a sentence, $S_1 \Rightarrow S_2$ is a sentence

If S_1 and S_2 is a sentence, $S_1 \Leftrightarrow S_2$ is a sentence

Last Time: Semantics of Propositional logic

Each model specifies true/false for each proposition symbol

E.g. $\begin{array}{ccc} A & B & C \\ True & True & False \end{array}$

Rules for evaluating truth with respect to a model m:

$$\begin{array}{rlllll}
\neg S & \text{is true iff} & S & \text{is false} & & \\
S_1 \wedge S_2 & \text{is true iff} & S_1 & \text{is true } \underline{\text{and}} & S_2 & \text{is true} \\
S_1 \vee S_2 & \text{is true iff} & S_1 & \text{is true } \underline{\text{or}} & S_2 & \text{is true} \\
S_1 \Rightarrow S_2 & \text{is true iff} & S_1 & \text{is false } \underline{\text{or}} & S_2 & \text{is true} \\
\text{i.e.,} & \text{is false iff} & S_1 & \text{is true } \underline{\text{and}} & S_2 & \text{is false} \\
S_1 \Leftrightarrow S_2 & \text{is true iff} & S_1 \Rightarrow S_2 & \text{is true } \underline{\text{and}} & S_2 \Rightarrow S_1 & \text{is true}
\end{array}$$

Last Time: Inference rules for propositional logic

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This time

- **First-order logic**
  - Syntax
  - Semantics
  - Wumpus world example

- **Ontology** (ont = ‘to be’; logica = ‘word’): kinds of things one can talk about in the language

Why first-order logic?

- We saw that propositional logic is limited because it only makes the ontological commitment that the world consists of **facts**.

- Difficult to represent even simple worlds like the Wumpus world;

  e.g.,

  "don't go forward if the Wumpus is in front of you" takes 64 rules

First-order logic (FOL)

- Ontological commitments:
  - **Objects**: wheel, door, body, engine, seat, car, passenger, driver
  - **Relations**: Inside(car, passenger), Beside(driver, passenger)
  - **Functions**: ColorOf(car)
  - **Properties**: Color(car), IsOpen(door), IsOn(engine)

- Functions are relations with single value for each object

Semantics

there is a correspondence between

- functions, which return values
- predicates, which are true or false

Function: father_of(Mary) = Bill

Predicate: father_of(Mary, Bill) [true or false]

Examples:

- "One plus two equals three"

  Objects:

  Relations:

  Properties:

  Functions:

- "Squares neighboring the Wumpus are smelly"

  Objects:

  Relations:

  Properties:

  Functions:

Examples:

- "One plus two equals three"

  Objects: one, two, three, one plus two

  Relations: equals

  Properties: --

  Functions: plus ("one plus two" is the name of the object obtained by applying function plus to one and two; three is another name for this object)

- "Squares neighboring the Wumpus are smelly"

  Objects: Wumpus, square

  Relations: neighboring

  Properties: smelly

  Functions: --

FOL: Syntax of basic elements

- **Constant symbols:** 1, 5, A, B, USC, JPL, Alex, Manos, …
- **Predicate symbols:** >, Friend, Student, Colleague, …
- **Function symbols:** +, sqrt, SchoolOf, TeacherOf, ClassOf, …
- **Variables:** *x, y, z, next, first, last, …*
- **Connectives:** $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$
- **Quantifiers:** $\forall$, $\exists$
- **Equality:** =

FOL: Atomic sentences

$$\text{AtomicSentence} \rightarrow \text{Predicate(Term, …)} \mid \text{Term} = \text{Term}$$

$$\text{Term} \rightarrow \text{Function(Term, …)} \mid \text{Constant} \mid \text{Variable}$$

- Examples:
  - SchoolOf(Manos)
  - Colleague(TeacherOf(Alex), TeacherOf(Manos))
  - >((+ x y), x)

FOL: Complex sentences

Sentence $\rightarrow$ AtomicSentence
| Sentence Connective Sentence
| Quantifier Variable, … Sentence
| $\neg$ Sentence
| (Sentence)

- Examples:
  - $S1 \wedge S2$, $S1 \vee S2$, $(S1 \wedge S2) \vee S3$, $S1 \Rightarrow S2$, $S1 \Leftrightarrow S3$
  - Colleague(Paolo, Maja) $\Rightarrow$ Colleague(Maja, Paolo)
    Student(Alex, Paolo) $\Rightarrow$ Teacher(Paolo, Alex)

Semantics of atomic sentences

- Sentences in FOL are interpreted with respect to a model
- Model contains objects and relations among them
- Terms: refer to objects (e.g., Door, Alex, StudentOf(Paolo))
  - Constant symbols: refer to objects
  - Predicate symbols: refer to relations
  - Function symbols: refer to functional Relations

- An atomic sentence *predicate*($term_1$, …, $term_n$) is **true** iff the relation referred to by *predicate* holds between the objects referred to by $term_1$, …, $term_n$

Example model

- Objects: John, James, Marry, Alex, Dan, Joe, Anne, Rich

- Relation: sets of tuples of objects
  {<John, James>, <Marry, Alex>, <Marry, James>, …}
  {<Dan, Joe>, <Anne, Marry>, <Marry, Joe>, …}

- E.g.:
  Parent relation -- {<John, James>, <Marry, Alex>, <Marry, James>}

  then Parent(John, James) is true
  Parent(John, Marry) is false

Quantifiers

- Expressing sentences about **collections** of objects without enumeration (naming individuals)

- E.g., All Trojans are clever

  Someone in the class is sleeping

- Universal quantification (for all): $\forall$

- Existential quantification (there exists): $\exists$

Universal quantification (for all): $\forall$

$\forall$ *<variables> <sentence>*

- *"Every one in the cs561 class is smart"*:
  $\forall\ x \quad In(cs561, x) \Rightarrow Smart(x)$

- **$\forall$ P corresponds to the conjunction of instantiations of P**
  $(In(cs561, Manos) \Rightarrow Smart(Manos)) \wedge$
  $(In(cs561, Dan) \Rightarrow Smart(Dan)) \wedge$
  …
  $(In(cs561, Bush) \Rightarrow Smart(Bush))$

Universal quantification (for all): $\forall$

- $\Rightarrow$ is a natural connective to use with $\forall$

- Common mistake: to use $\wedge$ in conjunction with $\forall$
  e.g.: $\forall\, x \quad In(cs561, x) \wedge Smart(x)$

Universal quantification (for all): $\forall$

- $\Rightarrow$ is a natural connective to use with $\forall$

- Common mistake: to use $\wedge$ in conjunction with $\forall$
  e.g.: $\forall\, x \quad In(cs561, x) \wedge Smart(x)$
  means *"everyone is in cs561 and everyone is smart"*

Existential quantification (there exists): ∃

∃ *<variables> <sentence>*

- *"Someone in the cs561 class is smart"*:
  $\exists\, x \;\; \mathrm{In}(cs561, x) \wedge \mathrm{Smart}(x)$

- **∃ P corresponds to the disjunction of instantiations of P**
  $\mathrm{In}(cs561, Manos) \wedge \mathrm{Smart}(Manos) \vee$
  $\mathrm{In}(cs561, Dan) \wedge \mathrm{Smart}(Dan) \vee$
  …
  $\mathrm{In}(cs561, Bush) \wedge \mathrm{Smart}(Bush)$

Existential quantification (there exists): $\exists$

- $\wedge$ is a natural connective to use with $\exists$

- Common mistake: to use $\Rightarrow$ in conjunction with $\exists$
  e.g.: $\exists\, x \quad \text{In}(\text{cs561}, x) \Rightarrow \text{Smart}(x)$

Existential quantification (there exists): $\exists$

- $\wedge$ is a natural connective to use with $\exists$

- Common mistake: to use $\Rightarrow$ in conjunction with $\exists$
  e.g.: $\exists\, x \quad In(cs561, x) \Rightarrow Smart(x)$
  is true if there is anyone that is not in cs561!
  (remember, false $\Rightarrow$ true is valid).

Properties of quantifiers

$\forall x \ \forall y$ is the same as $\forall y \ \forall x$ (why??)

$\exists x \ \exists y$ is the same as $\exists y \ \exists x$ (why??)

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

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

Not all by one person but each one at least by one

Quantifier duality: each can be expressed using the other

$\forall x \ \ Likes(x, IceCream) \qquad \neg\exists x \ \ \neg Likes(x, IceCream)$

$\exists x \ \ Likes(x, Broccoli) \qquad \neg\forall x \ \ \neg Likes(x, Broccoli)$

Proof?

Proof

- In general we want to prove:

$\forall x \; P(x) \iff \neg \exists x \neg P(x)$

- $\forall x \, P(x) = \neg(\neg(\forall x \, P(x))) = \neg(\neg(P(x1) \wedge P(x2) \wedge \ldots \wedge P(xn))) = \neg(\neg P(x1) \vee \neg P(x2) \vee \ldots \vee \neg P(xn)))$

- $\exists x \, \neg P(x) = \neg P(x1) \vee \neg P(x2) \vee \ldots \vee \neg P(xn)$

- $\neg\exists x \, \neg P(x) = \neg(\neg P(x1) \vee \neg P(x2) \vee \ldots \vee \neg P(xn))$

Example sentences

- Brothers are siblings

  .

- Sibling is transitive

  .

- One’s mother is one’s sibling’s mother

  .

- A first cousin is a child of a parent’s sibling

  .

Example sentences

- Brothers are siblings

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

- Sibling is transitive

  $\forall$ x, y, z Sibling(x, y) $\wedge$ Sibling(y, z) $\Rightarrow$ Sibling(x, z)

- One's mother is one's sibling's mother

  $\forall$ m, c, d Mother(m, c) $\wedge$ Sibling(c, d) $\Rightarrow$ Mother(m, d)

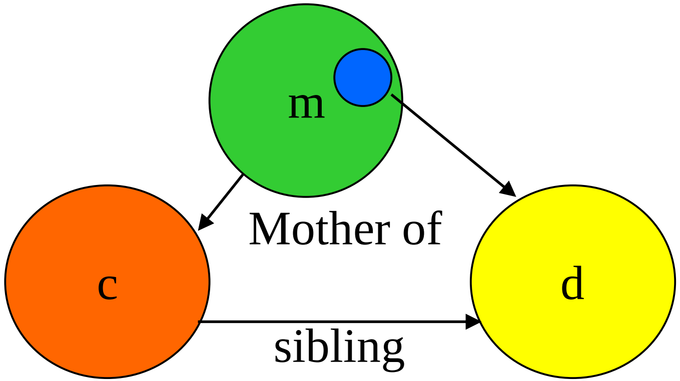
- A first cousin is a child of a parent's sibling

  $\forall$ c, d FirstCousin(c, d) $\Leftrightarrow$
  $\exists$ p, ps Parent(p, d) $\wedge$ Sibling(p, ps) $\wedge$ Child(c, ps)

Example sentences

- One's mother is one's sibling's mother
  $\forall$ m, c, d Mother(m, c) $\wedge$ Sibling(c, d) $\Rightarrow$ Mother(m, d)

- $\forall$ c, d $\exists$**m** Mother(m, c) $\wedge$ Sibling(c, d) $\Rightarrow$ Mother(m, d)



Translating English to FOL

- Every gardener likes the sun.
  `∀ x gardener(x) => likes(x,Sun)`

- You can fool some of the people all of the time.
  `∃ x ∀ t (person(x) ^ time(t)) => can-fool(x,t)`

Translating English to FOL

- You can fool all of the people some of the time.

  `∀ x person(x) => ∃ t time(t) ^ can-fool(x,t)`

- All purple mushrooms are poisonous.

  `∀ x (mushroom(x) ^ purple(x)) => poisonous(x)`

Caution with nested quantifiers

- $\forall$ x $\exists$ y P(x,y) is the same as $\forall$ x ($\exists$ y P(x,y)) which means "for every x, it is true that there exists y such that P(x,y)"

- $\exists$ y $\forall$ x P(x,y) is the same as $\exists$ y ($\forall$ x P(x,y)) which means "there exists a y, such that it is true that for every x P(x,y)"

Translating English to FOL…

- No purple mushroom is poisonous.

```
¬(∃ x) purple(x) ^ mushroom(x) ^ poisonous(x)
```

or, equivalently,

```
(∀ x) (mushroom(x) ^ purple(x)) =>
  ¬poisonous(x)
```

Translating English to FOL…

- There are exactly two purple mushrooms.

```
(∃ x)(∃ y) mushroom(x) ^ purple(x) ^
  mushroom(y) ^ purple(y) ^ ¬(x=y) ^ (∀ z)
  (mushroom(z) ^ purple(z)) => ((x=z) v (y=z))
```

- Deb is not tall.

```
¬tall(Deb)
```

Translating English to FOL…

- X is above Y iff X is directly on top of Y or else there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.

(∀ x)(∀ y) above(x,y) <=> (on(x,y) v (∃ z) (on(x,z) ^ above(z,y)))

Equality

$term_1 = term_2$ is true under a given interpretation
if and only if $term_1$ and $term_2$ refer to the same object

E.g., $1 = 2$ and $\forall x \;\; \times(Sqrt(x), Sqrt(x)) = x$ are satisfiable
$2 = 2$ is valid

E.g., definition of (full) $Sibling$ in terms of $Parent$:

$$\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; [\neg(x = y) \land \exists m, f \;\; \neg(m = f) \land$$
$$Parent(m, x) \land Parent(f, x) \land Parent(m, y) \land Parent(f, y)]$$

Higher-order logic?

- First-order logic allows us to quantify over objects (= the first-order entities that exist in the world).

- Higher-order logic also allows quantification over relations and functions.
  e.g., "two objects are equal iff all properties applied to them are equivalent":

  $\forall x,y \quad (x=y) \Leftrightarrow (\forall p,\ p(x) \Leftrightarrow p(y))$

- Higher-order logics are more expressive than first-order; however, so far we have little understanding on how to effectively reason with sentences in higher-order logic.

Logical agents for the Wumpus world

Remember: generic knowledge-based agent:

function KB-AGENT(*percept*) **returns** an *action*
  static: *KB*, a knowledge base
    t, a counter, initially 0, indicating time

  TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
  action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*))
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
  $t \leftarrow t + 1$
  return *action*

1. TELL KB what was perceived
   Uses a KRL to insert new sentences, representations of facts, into KB

2. ASK KB what to do.
   Uses logical reasoning to examine actions and select best.

Using the FOL Knowledge Base

Suppose a wumpus-world agent is using an FOL KB
and perceives a smell and a breeze (but no glitter) at $t = 5$:

$\textsc{Tell}(KB, Percept([Smell, Breeze, None], 5))$
$\textsc{Ask}(KB, \exists a \;\; Action(a, 5))$

I.e., does the KB entail any particular actions at $t = 5$?

Answer: $Yes,\ \{a/Shoot\}$ $\leftarrow$ substitution (binding list)

Set of solutions

Given a sentence S and a substitution σ,
$S\sigma$ denotes the result of plugging σ into S; e.g.,
$S = Smarter(x, y)$
$\sigma = \{x/Hillary, y/Bill\}$
$S\sigma = Smarter(Hillary, Bill)$

$\textsc{Ask}(\mathrm{KB}, \mathrm{S})$ returns some/all σ such that $KB \models S\sigma$

Wumpus world, FOL Knowledge Base

"Perception"

$\forall b, g, t \;\; Percept([Smell, b, g], t) \Rightarrow Smelt(t)$

$\forall s, b, t \;\; Percept([s, b, Glitter], t) \Rightarrow AtGold(t)$

Reflex: $\forall t \;\; AtGold(t) \Rightarrow Action(Grab, t)$

Reflex with internal state: do we have the gold already?

$\forall t \;\; AtGold(t) \wedge \neg Holding(Gold, t) \Rightarrow Action(Grab, t)$

$Holding(Gold, t)$ cannot be observed

$\Rightarrow$ keeping track of change is essential

Deducing hidden properties

Properties of locations:

$\forall l,t \;\; At(Agent,l,t) \land Smelt(t) \;\Rightarrow\; Smelly(l)$

$\forall l,t \;\; At(Agent,l,t) \land Breeze(t) \;\Rightarrow\; Breezy(l)$

Squares are breezy near a pit:

Diagnostic rule—infer cause from effect

$$\forall y \;\; Breezy(y) \;\Rightarrow\; \exists x \;\; Pit(x) \land Adjacent(x,y)$$

Causal rule—infer effect from cause

$$\forall x,y \;\; Pit(x) \land Adjacent(x,y) \;\Rightarrow\; Breezy(y)$$

Neither of these is complete—e.g., the causal rule doesn't say whether squares far away from pits can be breezy

Definition for the $Breezy$ predicate:

$$\forall y \;\; Breezy(y) \;\Leftrightarrow\; [\exists x \;\; Pit(x) \land Adjacent(x,y)]$$

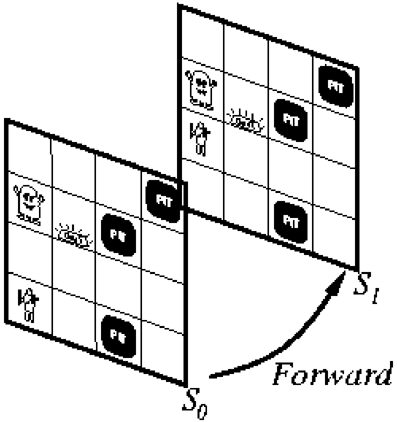
Situation calculus

Facts hold in <u>situations</u>, rather than eternally
E.g., $Holding(Gold, Now)$ rather than just $Holding(Gold)$

<u>Situation calculus</u> is one way to represent change in FOL:

> Adds a situation argument to each non-eternal predicate
> E.g., Now in $Holding(Gold, Now)$ denotes a situation

Situations are connected by the $Result$ function
$Result(a, s)$ is the situation that results from doing a is s



Describing actions

"Effect" axiom—describe changes due to action

$\forall s \;\; AtGold(s) \;\Rightarrow\; Holding(Gold, Result(Grab, s))$

"Frame" axiom—describe non-changes due to action

$\forall s \;\; HaveArrow(s) \;\Rightarrow\; HaveArrow(Result(Grab, s))$

May result in too many frame axioms

Frame problem: find an elegant way to handle non-change

(a) representation—avoid frame axioms

(b) inference—avoid repeated "copy-overs" to keep track of state

Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or . . .

Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, . . .

Describing actions (cont'd)

Successor-state axioms solve the representational frame problem

Each axiom is "about" a predicate (not an action per se):

P true afterwards $\Leftrightarrow$ [an action made P true

$\vee$ P true already and no action made P false]

For holding the gold:

$$\forall a,s \;\; Holding(Gold, Result(a,s)) \Leftrightarrow$$
$$[(a = Grab \wedge AtGold(s))$$
$$\vee \; (Holding(Gold,s) \wedge a \neq Release)]$$

Planning

Initial condition in KB:

$At(Agent, [1,1], S_0)$
$At(Gold, [1,2], S_0)$

Query: $\text{Ask}(KB, \exists s \;\; Holding(Gold, s))$
i.e., in what situation will I be holding the gold?

Answer: $\{s/Result(Grab, Result(Forward, S_0))\}$
i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at S_0 and that S_0 is the only situation described in the KB

Generating action sequences

Represent plans as action sequences $[a_1, a_2, \ldots, a_n]$

$PlanResult(p, s)$ is the result of executing p in s

Then the query $\text{Ask}(KB, \exists p \;\; Holding(Gold, PlanResult(p, S_0)))$
has the solution $\{p/[Forward, Grab]\}$

Definition of $PlanResult$ in terms of $Result$:

$\forall s \;\; PlanResult([], s) = s$ [ ] = empty plan

$\forall a, p, s \;\; PlanResult([a|p], s) = PlanResult(p, Result(a, s))$

Recursively continue until it gets to empty plan [ ]

Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

Summary

First-order logic:

- objects and relations are semantic primitives
- syntax: constants, functions, predicates, equality, quantifiers

Increased expressive power: sufficient to define wumpus world

Situation calculus:

- conventions for describing actions and change in FOL
- can formulate planning as inference on a situation calculus KB