First-Order Logic

Natalie Parde, Ph.D.

Department of Computer Science

University of Illinois at Chicago

CS 421: Natural Language Processing

Fall 2019

Many slides adapted from Jurafsky and Martin (https://web.stanford.edu/~jurafsky/slp3/).

What is first-order logic?

- A **meaning representation language** (a way to represent knowledge in a way that is computationally verifiable and supports semantic inference)

Why do we need meaning representations?

- Somehow, we need to bridge the gap between **linguistic input** and non-linguistic **world knowledge** to perform semantic processing tasks
- Everyday examples of (human) semantic processing:
  - Answering essay questions on exams
  - Deciding what to order at a restaurant
  - Detecting sarcasm
  - Following recipes
  - Learning how to convert sentences to first-order logic

Meaning Representations

- **Goal: Represent commonsense world knowledge in logical form**
- These representations are created and assigned to linguistic inputs via **semantic analysis**

First-order logic isn't the only way to represent meaning.

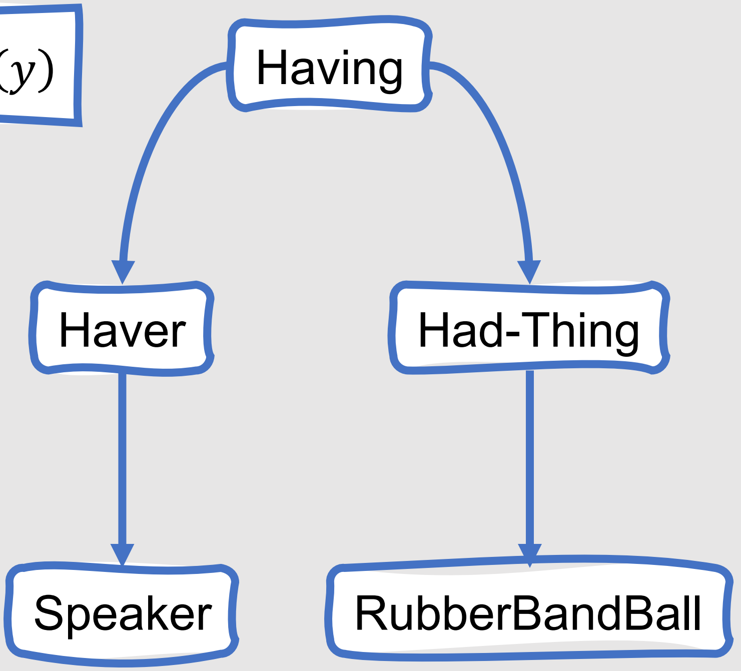
- Other methods:
  - **Semantic networks**
  - **Conceptual dependencies**
  - **Frame-based representations**
- However, all of these approaches assume that meaning representations consist of **structures** composed from a set of **symbols**
  - **Symbols:** Representational vocabulary

Sample Meaning Representations

I have a rubber band ball.

$\exists x, y \ \mathrm{Having}(x) \wedge \ \mathrm{Haver}(x, Speaker) \wedge \mathrm{HadThing}(x, y) \wedge \mathrm{RubberBandBall}(y)$

Having	
Haver:	Speaker
HadThing:	RubberBandBall



Symbols

- Correspond to **objects**, **properties** of objects, and **relations** among objects
- These are the links between linguistic input (words) and meaning (world knowledge)

Having	
Haver:	Speaker
HadThing:	RubberBandBall

Basic Characteristics of Meaning Representations

Verifiability

Unambiguous Representations

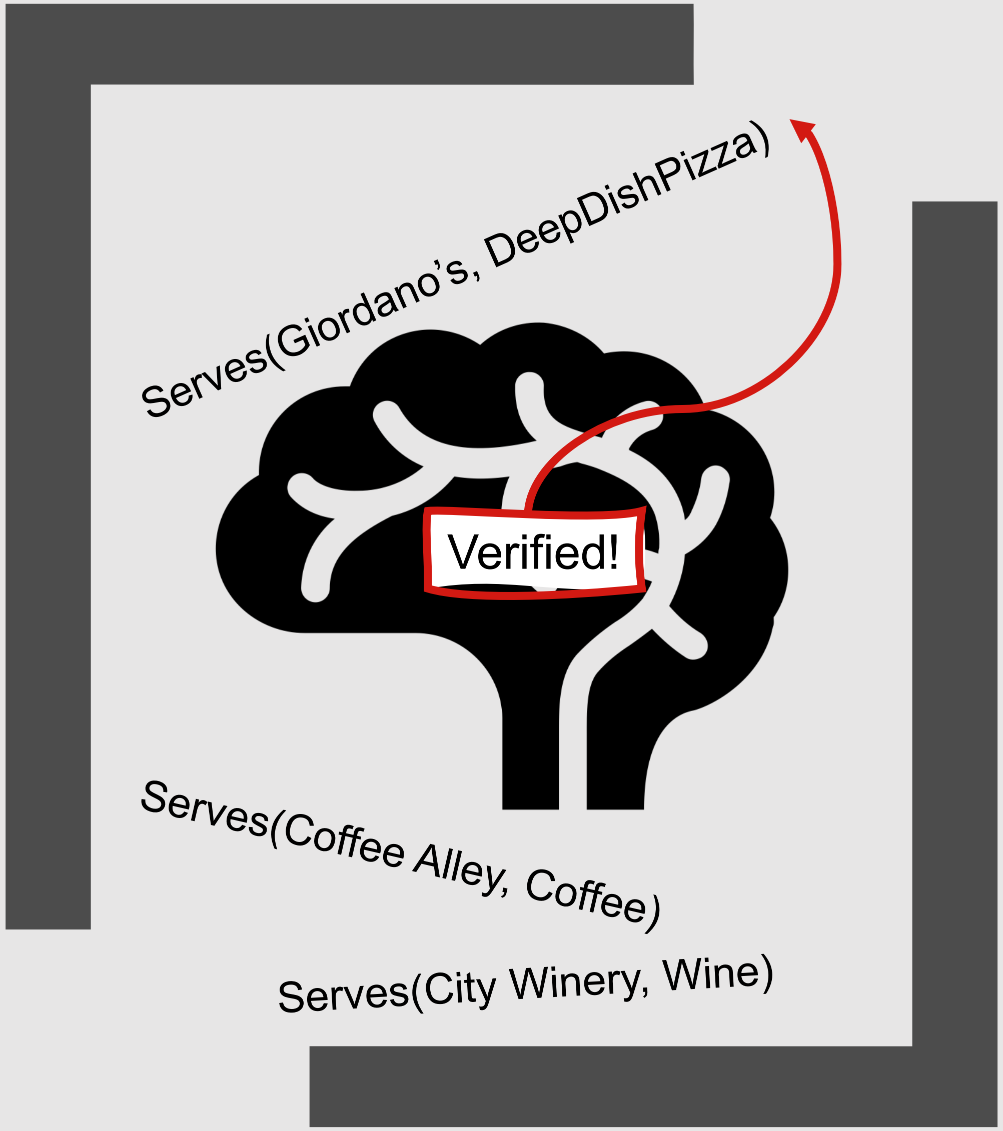
Canonical Form

Inference and Variables

Expressiveness

Verifiability

- Meaning representations determine the relationship between (a) **the meaning of a sentence** and (b) **the world as we know it**
- Computational systems can verify the truth of a meaning representation for a sentence by matching it with **knowledge base** representations
  - **Knowledge Base:** A source of information about the world

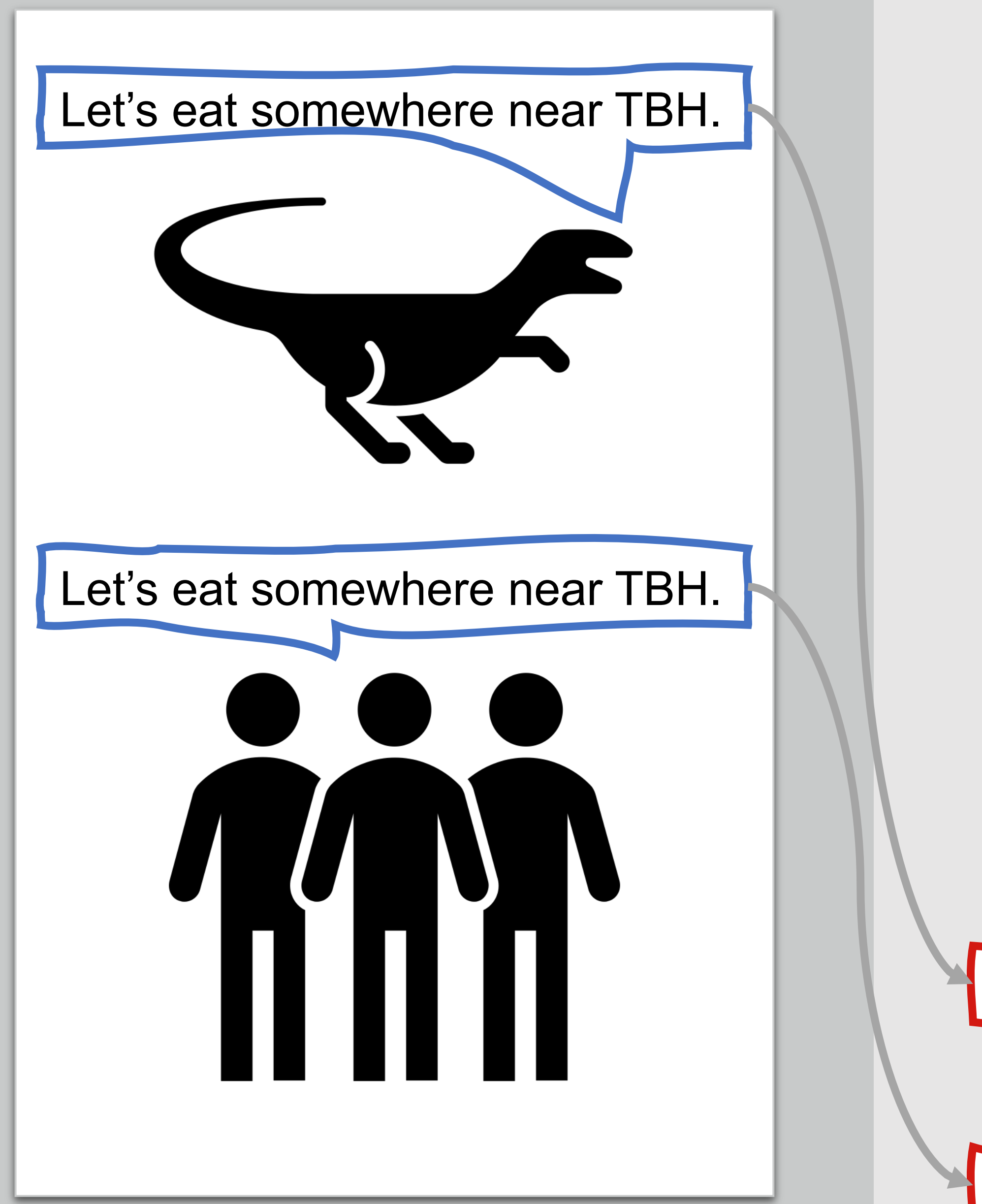

Serves(Giordano's, DeepDishPizza)
Verified!
Serves(Coffee Alley, Coffee)
Serves(City Winery, Wine)

Verifiability

- Example proposition: **Giordano's serves deep dish pizza**.
- We can represent this as: **Serves(Giordano's, DeepDishPizza)**
- To verify the truth of this proposition, we would search a knowledge base containing facts about restaurants
- If we found a fact matching this, we have verified the proposition
- If not, we must assume that the fact is incorrect or, at best, our knowledge base is incomplete

Unambiguous Representations

- Ambiguity does not stop at syntax!
- Semantic ambiguities are everywhere:
  - Sarcasm
  - Idiom
  - Metaphor
  - Hyperbole
- Specifically, individual expressions can have different meaning representations depending upon the circumstances in which they occur



Unambiguous Representations

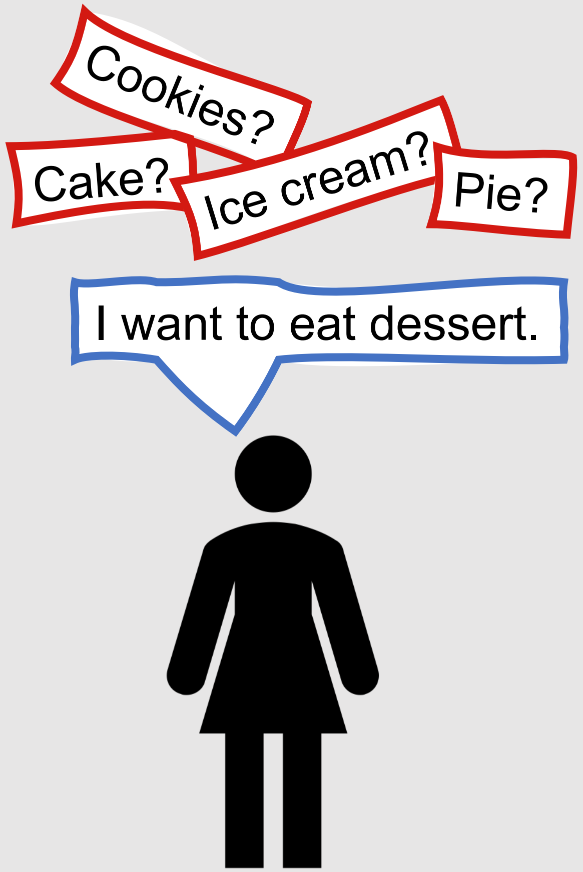
- We'll ignore ambiguities arising from figurative language in this course, and focus on the semantic ambiguities that can still arise from literal expressions
- To resolve semantic ambiguities, computational methods must be employed to select which from a set of possible interpretations is most correct, given the circumstances surrounding the linguistic input

Let's devour some building near TBH!

Let's eat at a restaurant near TBH!

Vagueness

- Closely related to ambiguity
- However, vagueness does not give rise to multiple representations
- In fact, it is advantageous for meaning representations to maintain a certain level of vagueness
  - Otherwise, you may be "overfitting" to your set of example sentences



Canonical Form

- Ambiguity means that a given sentence could be assigned multiple meaning representations
- However, **multiple sentences could also be assigned the same meaning representation**
  - Giordano's serves deep dish pizza.
  - They have deep dish pizza at Giordano's.
  - Deep dish pizza is served at Giordano's.
  - You can eat deep dish pizza at Giordano's.

In many cases, we want similar sentences that are phrased differently to be grouped together.

- Case in point: the previous examples
- How do we do this though?
    - Store all possible options 🤨
    - Reduce sentences to **canonical form** 🙂

How do we reduce sentences to canonical form?

- Exploit systematic meaning relationships among word senses and grammatical constructions
  - <subject> serves <object> ≈ <object> is served (at) <subject>
- Word sense: One (of potentially many) definition for a word
- Some word senses are synonymous with one another
  - Serves (Sense #2) ≈ Have (Sense #4)
- The process of choosing the correct sense for a word, given its context, is called word sense disambiguation

Inference and Variables

- It's impossible for a knowledge base to comprehensively cover all facts about the world, so computational systems also need to be able to draw commonsense inferences based on meaning representations
  - Will people who like deep dish pizza want to eat at Giordano's?
    - We don't have a fact explicitly specifying that they do, but we can infer that if they like deep dish pizza, they will probably like a restaurant that serves it

Inference

- **Inference:** A system's ability to draw valid conclusions based on the meaning representations of inputs and its store of background knowledge
- Systems must be able to draw conclusions about the truth of propositions that are not explicitly represented in the knowledge base but that are logically derivable from the propositions that are present

Variables

- Variables allow you to build propositions without requiring a specific instance of something
  - Serves(x, DeepDishPizza)
- These propositions can only be successfully matched by known instances from a knowledge base that would resolve in a truthful entire proposition
  - Serves(x, DeepDishPizza)
    - Serves(Giordano’s, DeepDishPizza) 🙂
    - Serves(Coffee Alley, DeepDishPizza) 🤨

Expressiveness

- **Expressive power:** The breadth of ideas that can be represented in a language
- Meaning representations must be **expressive** enough to handle a wide range of subject matter
- Is it possible to create a single meaning representation language that comprehensively represents all sensible natural language sentences?
  - Probably not quite, but first-order logic is expressive enough to handle a lot

What do most meaning representation schemes share in common?

- An ability to represent objects, properties of objects, and relations among objects (symbols)

A **model** is a formal construct that stands for a particular state of affairs in the world that we’re trying to represent

Expressions (words or phrases) in the meaning representation language can be mapped to elements of the model

Model-Theoretic Semantics

Relevant Terminology

- Vocabulary
  - **Non-Logical Vocabulary:** Open-ended sets of names for objects, properties, and relations in the world we're representing
  - **Logical Vocabulary:** Closed set of symbols, operators, quantifiers, links, etc. that provide the formal means for composing expressions in the language
- **Domain:** The set of objects that are part of the state of affairs being represented in the model
- **Each object in the non-logical vocabulary corresponds to a unique element in the domain**; however, each element in the domain does not need to be mentioned in a meaning representation
  - **Some elements in the domain might be represented multiple ways**
    - Natalie, ProfessorOf(CS), ResidentOf(Chicago), ViewerOf(Netflix)

Additional Terminology

- For a given domain, **objects** are elements
  - grapes, violets, plums, CS421, Mina, Mohammad
- **Properties** are sets of elements corresponding to a specific characteristic
  - purple = {grapes, violets, plums}
- **Relations** are sets of tuples, each of which contain domain elements that take part in a specific relation
  - StudentIn = {(CS421, Mina), (CS421, Mohammad)}

How do we create mappings from non-logical vocabulary to formal denotations (properties and relations)?

We create functions (interpretations)!

Example Application

- Assume that we have:
  - A collection of restaurant patrons and restaurants
  - Various facts regarding the likes and dislikes of patrons
  - Various facts about the restaurants
- In our current state of affairs (our **model**) we're concerned with four patrons designated by the non-logical symbols (**elements**) *Natalie*, *Usman*, *Shahla*, and *Yatri*
- We'll use the constants *a*, *b*, *c*, and *d* to refer to those respective elements

Example Application

patron = {Natalie, Usman, Shahla, Yatri} = {a, b, c, d}

- We're also concerned with three restaurants designated by the non-logical symbols *Giordano's*, *IDOF*, and *Artopolis*
- We'll use the constants *e*, *f*, and *g* to refer to those respective elements

Example Application

patron = {Natalie, Usman, Shahla, Yatri} = {a, b, c, d}

restaurants = {Giordano’s, IDOF, Artopolis} = {e, f, g}

- Finally, we’ll assume that our model deals with three cuisines in general, designated by the non-logical symbols *Italian*, *Mediterranean*, and *Greek*
- We’ll use the constants *i*, *j*, and *k* to refer to those elements

Example Application

patron = {Natalie, Usman, Shahla, Yatri} = {a, b, c, d}

restaurants = {Giordano's, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

- Now, let's assume we need to represent a few properties of restaurants:
  - *Fast* denotes the subset of restaurants that are known to make food quickly
  - *TableService* denotes the subset of restaurants for which a waiter will come to your table to take your order
- We also need to represent a few relations:
  - *Like* denotes the tuples indicating which restaurants individual patrons like
  - *Serve* denotes the tuples indicating which restaurants serve specific cuisines

Example Application

patron = {Natalie, Usman, Shahla, Yatri} = {a, b, c, d}

restaurants = {Giordano’s, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

- This means that we have created the domain D = {a, b, c, d, e, f, g, i, j, k}
- We can evaluate representations like *Natalie likes IDOF* or *Giordano’s serves Greek* by mapping the objects in the meaning representations to their corresponding domain elements, and any links to the appropriate relations in the model
  - Natalie likes IDOF → a likes f → Like(a, f) 🙂
  - Giordano’s serves Greek → e serves k, Serve(e, k) 🤨

Example Application

patron = {Natalie, Usman, Shahla, Yatri} = {a, b, c, d}

restaurants = {Giordano’s, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

- Thus, we’re just using sets and operations on sets to ground the expressions in our meaning representations
- What about more complex sentences?
  - Shahla likes Giordano’s and Usman likes Artopolis.
  - Yatri likes fast restaurants.
  - Not everybody likes IDOF.

Example Application

patron = {Natalie, Usman, Shahla, Yatri} = {a, b, c, d}

restaurants = {Giordano’s, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

- Plausible meaning representations for the previous examples will not map directly to individual entities, properties, or relations!
- They involve:
  - Conjunctions
  - Equality
  - Variables
  - Negations
- What we need are **truth-conditional semantics**
- This is where **first-order logic** comes in handy

Basic Elements of First-Order Logic

Term: First-order logic device for representing objects

- Constants
- Functions
- Variables

Common across all types of terms:

- Each one can be thought of as a way of pointing to an object in the world being considered

Basic Elements of First-Order Logic

- Terms:
  - **Constants:** Specific objects in the world being described
    - Conventionally depicted as single capitalized letters (A, B) or words (Natalie, Usman)
    - Refer to exactly one object, although objects can have more than one constant that refers to them
  - **Functions:** Concepts that are syntactically equivalent to single-argument predicates
    - Can refer to specific objects without having to associate a named constant with them, e.g., LocationOf(Giordano's)
  - **Variables:** Provide the ability to make assertions and draw inferences without having to refer to a specific named object
    - Conventionally depicted as single lowercase letters

Basic Elements of First-Order Logic

- **Predicates:** Symbols that refer to the relations between a fixed number of objects in the domain
  - Can have one or more arguments
    - Serve(Giordano's, Italian)
      - Relates two objects
    - Restaurant(Giordano's)
      - Asserts a property of a single object
- Predicates can be put together using **logical connectives**
  - and ∧
  - or ∨
  - implies →
- They can also be **negated**
  - not ¬

Variables and Quantifiers

- Two basic operators in first-order logic are:
  - ∃: The existential quantifier
    - Pronounced “there exists”
  - ∀: The universal quantifier
    - Pronounced “for all”
- These two operators make it possible to represent many more sentences!
  - a restaurant → $\exists x\ Restaurant(x)$
  - all restaurants → $\forall x\ Restaurant(x)$

We can combine these operators with other basic elements of first-order logic to build logical representations of complex sentences.

- Shahla likes Giordano’s and Usman likes Artopolis.
  - Like(Shahla, Giordano’s) $\land$ Like(Usman, Artopolis)
- Yatri likes fast restaurants.
  - $\forall$x Fast(x) $\rightarrow$ Like(Yatri, x)
- Not everybody likes IDOF.
  - $\exists$x Person(x) $\land$ $\neg$Like(x, IDOF)

In-Class Exercise

- Convert the following sentences to first-order logic:
  - **Natalie likes Giordano's.**
  - **Usman does not like Giordano's.**
  - **Natalie likes all restaurants with table service.**
  - **Usman doesn't like every restaurant.**

https://www.google.com/search?q=timer

Useful symbols:

∃: There exists
∀: For all
∧: and
∨: or
¬: not
→: implies

In-Class Exercise

- Natalie likes Giordano’s.
- Usman does not like Giordano’s.
- Natalie likes all restaurants with table service.
- Usman doesn’t like every restaurant.

Semantics of First-Order Logic

Symbols for objects, properties, and relations acquire meaning based on their correspondences to “real” objects, properties, and relations in the external world

The model-theoretic approach employs a simple set of notions to define meaning based on truth-conditional mappings between expressions in a meaning representation and the state of affairs being modeled

We can determine truth based on the presence of specified terms and predicates.

P	Q	¬P	P∧Q	P∨Q	P→Q
False	False	True	False	False	True
False	True	True	False	True	True
True	False	False	False	True	False
True	True	False	True	True	True

Example: Is the following sentence valid according to our model?

patron = {Natalie, Usman, Shahla, Yatri} = {a, b, c, d}

restaurants = {Giordano’s, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

Natalie likes Giordano’s and Usman likes Giordano’s.

Example: Is the following sentence valid according to our model?

patron = {Natalie, Usman, Shahla, Yatri} = {a, b, c, d}

restaurants = {Giordano’s, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

Natalie likes Giordano’s and Usman likes Giordano’s.

Likes(Natalie, Giordano’s) ∧ Likes(Usman, Giordano’s)

Example: Is the following sentence valid according to our model?

patron = {Natalie, Usman, Shahla, Yatri} = {a, b, c, d}

restaurants = {Giordano’s, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

Natalie likes Giordano’s and Usman likes Giordano’s.

Likes(Natalie, Giordano’s) ∧ Likes(Usman, Giordano’s)

Likes(a, e) ∧ Likes(b, e)

Example: Is the following sentence valid according to our model?

patron = {Natalie, Usman, Shahla, Yatri} = {a, b, c, d}

restaurants = {Giordano’s, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

Natalie likes Giordano’s and Usman likes Giordano’s.

Likes(Natalie, Giordano’s) ∧ Likes(Usman, Giordano’s)

Likes(a, e) ∧ Likes(b, e)

☺

Example: Is the following sentence valid according to our model?

patron = {Natalie, Usman, Shahla, Yatri} = {a, b, c, d}

restaurants = {Giordano's, IDOF, Artopolis} = {e, f, g}

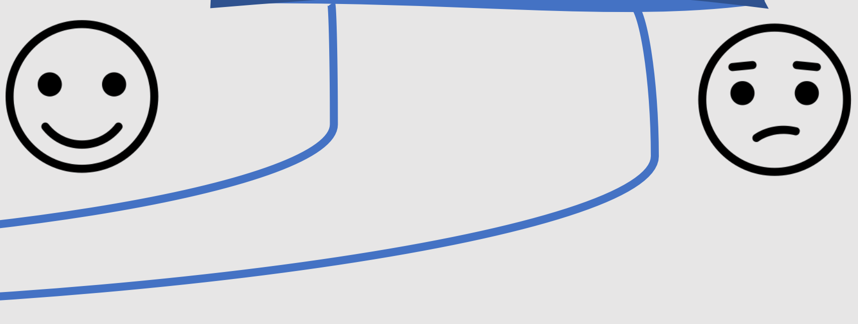
cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

Natalie likes Giordano's and Usman likes Giordano's.

Likes(Natalie, Giordano's) ∧ Likes(Usman, Giordano's)

Likes(a, e) ∧ Likes(b, e)



Example: Is the following sentence valid according to our model?

patron = {Natalie, Usman, Shahla, Yatri} = {a, b, c, d}

restaurants = {Giordano’s, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

Natalie likes Giordano’s and Usman likes Giordano’s.

Likes(Natalie, Giordano’s) ∧ Likes(Usman, Giordano’s)

Likes(a, e) ∧ Likes(b, e)

False …not valid!

A few additional notes….

- Formulas involving ∃ are true if there is *any* substitution of terms for variables that results in a formula that is true according to the model
- Formulas involving ∀ are true only if *all* substitutions of terms for variables result in formulas that are true according to the model

How do we infer facts not explicitly included in the knowledge base?

- **Modus ponens:** If a conditional statement is accepted (if p then q), and the **antecedent** (p) holds, then the **consequent** (q) may be inferred
- More formally:

$$\begin{array}{l} \alpha \\ a \Rightarrow \beta \\ \hline \quad \beta \end{array}$$

Example: Inference

$$\frac{\begin{array}{l}\text{GreekRestaurant}(Artopolis)\\ \forall x\ \text{GreekRestaurant}(x) \Rightarrow \text{Serves}(x, GreekFood)\end{array}}{\text{Serves}(Artopolis, GreekFood)}$$

conditional statement accepted ✓

Example: Inference

antecedent holds (our model says that Artopolis is a Greek restaurant) ✓

$$\frac{\begin{array}{l}\text{GreekRestaurant}(Artopolis)\\ \forall x\ \text{GreekRestaurant}(x) \Rightarrow \text{Serves}(x, GreekFood)\end{array}}{\text{Serves}(Artopolis, GreekFood)}$$

conditional statement accepted ✓

Example: Inference

antecedent holds (our model says that Artopolis is a Greek restaurant) ✓

$$\frac{\begin{array}{l}\mathrm{GreekRestaurant}(Artopolis)\\ \forall x\ \mathrm{GreekRestaurant}(x) \Rightarrow \mathrm{Serves}(x, GreekFood)\end{array}}{\mathrm{Serves}(Artopolis, GreekFood)}$$

conditional statement accepted ✓

consequent may be inferred 🙂

Representing States and Events

- **States:** Conditions, or properties, that remain unchanged over some period of time
- **Events:** Indicate changes in some state of affairs

Events can be particularly challenging to represent in formal logic!

You may need to:

- Determine the correct number of roles for the event
- Represent facts about different roles associated with the event
- Ensure that all correct inferences can be derived directly from the event representation
- Ensure that no incorrect inferences can be derived from the event representation

Some events may take a variable number of arguments

- Natalie drinks.
- Natalie drinks tea.

However, predicates in first-order logic have fixed **arity** (they accept a fixed number of arguments)

How do we deal with this?

- Make as many different predicates as are needed to handle all of the different ways an event can behave
  - $Drink_1(Natalie)$
  - $Drink_2(Natalie, tea)$
- Unfortunately, this can be costly (lots of different predicates would need to be stored for many words!)
- Another (also not-so-scalable) solution is to use **meaning postulates**
  - $\forall x,y\ Drink_2(x, y) \rightarrow Drink_1(x)$
- Finally, you can allow missing arguments
  - $\exists x\ Drink(Natalie, x)$
  - $Drink(Natalie, tea)$
- However, the last option still isn't perfect …in the example case, it implies that one always has to be drinking a specific thing

Instead of regular variables, we can add event variables.

- **Event variable:** The first argument to the representation of an event; allows for additional assertions to be included if needed
  - ∃e Drink(Natalie, e)
- If we determine that the actor must drink something specific: ∃e Drink(Natalie, e) ∧ Beverage(e, tea)
- More generally, we could define the representation:
  - ∃e Drink(e) ∧ Drinker(e, Natalie) ∧ Beverage(e, tea)
- With this change there is no need to specify a fixed number of arguments for a given surface predicate, and logical connections are satisfied without using meaning postulates

Ideally, meaning representations will also include information about time and aspect.

- Temporal information:
  - **Event time**
  - **Reference time**
  - **Time of utterance**

When Shahla leaves, Natalie will eat at Artopolis.

- Aspectual information:
  - **Stative:** Event captures an aspect of the world at a single time point
    - Natalie knew what she wanted to eat.
  - **Activity:** Event occurs over some span of time
    - Natalie is eating.
  - **Accomplishment:** Event has a natural end point and results in a particular state
    - Natalie ate lunch at Artopolis.
  - **Achievement:** Event happens in an instant, but still results in a particular state
    - Natalie finished her meal.

Summary: First-Order Logic

- **First-order logic** is a way to represent meaning by mapping linguistic input to world knowledge using logical rules
- Core components of a first-order logic model are:
  - **Objects**
  - **Properties**
  - **Relations**
- We can apply **truth-conditional logic** (**and**, **or**, and **not** operators) to sentences to determine whether they fit a given model based on their included terms
- First-order logic makes use of both **existential** and **universal** quantifiers
- Inferences can be drawn from first-order logic statements using **modus ponens**