Unit 4

Semantics

Requirements for Representation.

Expressiveness: Expressiveness refers to the ability of the representation system to capture a wide range of meanings, nuances, and contexts.

- Rich Vocabulary: The system should have a rich set of symbols to describe various entities, actions, properties, and relationships.
- Complex Structures: It should support the representation of complex structures such as nested and hierarchical relationships.
- Context Sensitivity: The representation should capture context to accurately reflect meaning (e.g., the difference between "bank" in "river bank" and "financial bank").
- Ambiguity Resolution: It should handle and resolve ambiguities in natural language.

Formalism: Formalism ensures that the representation has a well-defined syntax and semantics, allowing for unambiguous interpretation and processing.

- Clear Syntax: The rules for forming valid expressions should be precise and unambiguous.
- Defined Semantics: Each expression should have a clear meaning, defined in a way that machines can interpret consistently.
- Standardization: Use of standard representation languages (e.g., First-Order Logic, Description Logic) can help in maintaining consistency and interoperability.

Inference: Inference capability allows the system to derive new information from existing knowledge, which is essential for reasoning and decision-making.

- **Logical Deduction**: The system should support logical inference mechanisms such as modus ponens, modus tollens, and syllogism.
- **Probabilistic Inference**: For handling uncertainty, the system might need to support probabilistic reasoning (e.g., Bayesian networks).
- **Temporal Reasoning**: For applications involving time, the system should support reasoning about temporal relationships and events.

Propositional Logic

1. "If it is raining, then the ground is wet." - $P \to Q$
2. "Either the system is down, or the network is slow." - $P \lor Q$
3. "If the user is logged in and the session is active, then the user can access the dashboard." - $(P \land Q) \to R$
4. "The application will fail unless it is updated." - $\neg Q \to P$ or $P \lor Q$

5. "If the file is not found, an error message is displayed." - $\neg P \to Q$6. "The server is running if and only if the power is on." - $P \leftrightarrow Q$
7. "If the temperature is above 30°C, then the air conditioner turns on." - $P \to Q$
8. "If the database is backed up, the system will restart, and the logs will be cleared." - $P \to (Q \land R)$
9. "If the user presses the submit button, then the form is validated and the data is sent." - $P \to (Q \land R)$
10. "The document is either saved or discarded." - $P \lor Q$

First-Order Logic (FOL),

- **First-Order Logic (FOL)**, also known as Predicate Logic or First-Order Predicate Calculus, is a formal system used to express statements about objects, their properties, and their relationships. FOL extends propositional logic by incorporating quantifiers and predicates, making it much more expressive.

Key Components of First-Order Logic

Terms:

- Constants: Represent specific objects in the domain (e.g., a, b, Raju).
- Variables: Represent any object in the domain (e.g., x, y, z).
- Functions: Map a set of objects to another object
  - e.g., ‘fatherOf(Raju)’

Predicates: Represent properties or relationships between objects. For example,

- Student(Raju) means “Raju is a student," and Likes(Raju, Pizza) means "John likes pizza."

Logical Connectives:

- Conjunction (∧): P ∧ Q means "P and Q."
- Disjunction (∨): P ∨ Q means "P or Q."
- Negation (¬): ¬P means "not P."
- Implication (→): P → Q means "if P then Q."
- Biconditional (↔): P ↔ Q means "P if and only if Q."

Quantifiers:

- Universal Quantifier (∀): ∀x P(x) means "for all x, P(x) is true."
- Existential Quantifier (∃): ∃x P(x) means "there exists an x such that P(x) is true."

- "All humans are mortal."
- "Some students are brilliant." –
- "No dogs can fly.“
- "If a person is a parent, then they have a child." -
- "There is a cat that is black."
- "Every student loves some book." –
- "Someone likes everyone." -
- "If an animal is a bird, then it can fly." -
- "There exists a unique number that is even."
- "For every number, there is a greater number." -

1. "All humans are mortal." - $\forall x(\mathrm{Human}(x) \rightarrow \mathrm{Mortal}(x))$ x: human
2. "Some students are brilliant." - $\exists x(\mathrm{Student}(x) \land \mathrm{Brilliant}(x))$ x: student
3. "No dogs can fly." - $\forall x(\mathrm{Dog}(x) \rightarrow \neg \mathrm{CanFly}(x))$ x: dog
4. "If a person is a parent, then they have a child." - $\forall x(\mathrm{Parent}(x) \rightarrow \exists y(\mathrm{Child}(y, x)))$ x: parent, y: child
5. "There is a cat that is black." - $\exists x(\mathrm{Cat}(x) \land \mathrm{Black}(x))$ x: cat
6. "Every student loves some book." - $\forall x(\mathrm{Student}(x) \rightarrow \exists y(\mathrm{Book}(y) \land \mathrm{Loves}(x, y)))$ x: student, y: book
7. "Someone likes everyone." - $\exists x(\forall y(\mathrm{Person}(y) \rightarrow \mathrm{Likes}(x, y)))$ x: someone, y: person
8. "If an animal is a bird, then it can fly." - $\forall x(\mathrm{Bird}(x) \rightarrow \mathrm{CanFly}(x))$ x: bird

9. "There exists a unique number that is even." - $\exists x(\mathrm{Even}(x) \wedge \forall y(\mathrm{Even}(y) \rightarrow x = y))$ x: number, y: number
10. "For every number, there is a greater number." - $\forall x(\mathrm{Number}(x) \rightarrow \exists y(\mathrm{Number}(y) \wedge \mathrm{Greater}(y, x)))$ x: number, y: number

CMSC 471

First-Order Logic

Chapter 8.1-8.3

Adapted from slides by
Tim Finin and
Marie desJardins.

Some material adopted from notes
by Andreas Geyer-Schulz

Outline

- First-order logic
  - Properties, relations, functions, quantifiers, …
  - Terms, sentences, axioms, theories, proofs, …
- Extensions to first-order logic
- Logical agents
  - Reflex agents
  - Representing change: situation calculus, frame problem
  - Preferences on actions
  - Goal-based agents

First-order logic

- First-order logic (FOL) models the world in terms of
  - **Objects,** which are things with individual identities
  - **Properties** of objects that distinguish them from other objects
  - **Relations** that hold among sets of objects
  - **Functions,** which are a subset of relations where there is only one “value” for any given “input”
- Examples:
  - Objects: Students, lectures, companies, cars ...
  - Relations: Brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
  - Properties: blue, oval, even, large, ...
  - Functions: father-of, best-friend, second-half, one-more-than ...

User provides

- **Constant symbols,** which represent individuals in the world
  - Mary
  - 3
  - Green
- **Function symbols,** which map individuals to individuals
  - father-of(Mary) = John
  - color-of(Sky) = Blue
- **Predicate symbols,** which map individuals to truth values
  - greater(5,3)
  - green(Grass)
  - color(Grass, Green)

FOL Provides

- **Variable symbols**
  - E.g., x, y, foo
- **Connectives**
  - Same as in PL: not ($\neg$), and ($\wedge$), or ($\vee$), implies ($\rightarrow$), if and only if (biconditional $\leftrightarrow$)
- **Quantifiers**
  - Universal $\forall$**x** or **(Ax)**
  - Existential $\exists$**x** or **(Ex)**

Sentences are built from terms and atoms

- A **term** (denoting a real-world individual) is a constant symbol, a variable symbol, or an n-place function of n terms.
  - x and $f(x_1, \ldots, x_n)$ are terms, where each x_i is a term.
  - A term with no variables is a **ground term**
- An **atomic sentence** (which has value true or false) is an n-place predicate of n terms
- A **complex sentence** is formed from atomic sentences connected by the logical connectives:
  - $\neg P$, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ where P and Q are sentences
- A **quantified sentence** adds quantifiers $\forall$ and $\exists$
- A **well-formed formula (wff)** is a sentence containing no "free" variables. That is, all variables are "bound" by universal or existential quantifiers.
  - $(\forall x)P(x,y)$ has x bound as a universally quantified variable, but y is free.

Quantifiers

- Universal **quantification**
  - $(\forall x)P(x)$ means that P holds for **all** values of x in the domain associated with that variable
  - E.g., $(\forall x)$ dolphin(x) $\rightarrow$ mammal(x)
- Existential **quantification**
  - $(\exists x)P(x)$ means that P holds for **some** value of x in the domain associated with that variable
  - E.g., $(\exists x)$ mammal(x) $\wedge$ lays-eggs(x)
  - Permits one to make a statement about some object without naming it

Quantifiers

- Universal quantifiers are often used with "implies" to form "rules":

  $(\forall x)$ student(x) $\rightarrow$ smart(x) means "All students are smart"

- Universal quantification is *rarely* used to make blanket statements about every individual in the world:

  $(\forall x)$student(x)$\wedge$smart(x) means "Everyone in the world is a student and is smart"

- Existential quantifiers are usually used with "and" to specify a list of properties about an individual:

  $(\exists x)$ student(x) $\wedge$ smart(x) means "There is a student who is smart"

- A common mistake is to represent this English sentence as the FOL sentence:

  $(\exists x)$ student(x) $\rightarrow$ smart(x)

  - But what happens when there is a person who is *not* a student?

Quantifier Scope

- Switching the order of universal quantifiers *does not* change the meaning:
  - $(\forall x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\forall x) P(x,y)$
- Similarly, you can switch the order of existential quantifiers:
  - $(\exists x)(\exists y)P(x,y) \leftrightarrow (\exists y)(\exists x) P(x,y)$
- Switching the order of universals and existentials *does* change meaning:
  - Everyone likes someone: $(\forall x)(\exists y)$ likes(x,y)
  - Someone is liked by everyone: $(\exists y)(\forall x)$ likes(x,y)

Connections between All and Exists

We can relate sentences involving $\forall$ and $\exists$ using De Morgan's laws:

$$(\forall x)\, \neg P(x) \leftrightarrow \neg(\exists x)\, P(x)$$

$$\neg(\forall x)\, P \leftrightarrow (\exists x)\, \neg P(x)$$

$$(\forall x)\, P(x) \leftrightarrow \neg\, (\exists x)\, \neg P(x)$$

$$(\exists x)\, P(x) \leftrightarrow \neg(\forall x)\, \neg P(x)$$

Quantified inference rules

1. Universal instantiation: If a property is true for all elements in a domain, it is true for any specific element in that domain.
    1. $\forall x\ P(x)\ \therefore\ P(A)$
2. Universal generalization: If a property is true for an arbitrary element in a domain, then it is true for all elements in the domain.
    1. $P(A) \wedge P(B) \ldots\ \therefore\ \forall x\ P(x)$
3. Existential instantiation: If there exists an element in a domain for which a property is true, we can infer that the property is true for some specific element.
    1. $\exists x\ P(x)\ \therefore P(F)$
4. Existential generalization: If a property is true for some specific element in a domain, then there exists an element in the domain for which the property is true.
    1. $P(A)\ \therefore\ \exists x\ P(x)$

1. Statement: "All dogs bark."

Inference: "Rover is a dog, so Rover barks.“

2. **Observation**: "If any student studies hard, they pass the exam."

Generalization: "Therefore, all students who study hard pass the exam.“

3. **Statement**: "There is someone in the office who can help you."

Inference: "Let's call this person Alex. Alex can help you.“

4. **Observation**: "Maria can solve the puzzle."

Generalization: "Therefore, there exists someone who can solve the puzzle."

Universal instantiation (a.k.a. universal elimination)

- If $(\forall x)\ P(x)$ is true, then $P(C)$ is true, where C is *any* constant in the domain of x
- Example:

  $(\forall x)$ eats(Ziggy, x) $\Rightarrow$ eats(Ziggy, IceCream)
- The variable symbol can be replaced by any ground term, i.e., any constant symbol or function symbol applied to ground terms only

Existential instantiation (a.k.a. existential elimination)

- From $(\exists x)\ P(x)$ infer $P(c)$
- Example:
  - $(\exists x)$ eats(Ziggy, x) $\rightarrow$ eats(Ziggy, Stuff)
- Note that the variable is replaced by a **brand-new constant** not occurring in this or any other sentence in the KB
- Also known as skolemization; constant is a **skolem constant**
- In other words, we don't want to accidentally draw other inferences about it by introducing the constant
- Convenient to use this to reason about the unknown object, rather than constantly manipulating the existential quantifier

Existential generalization (a.k.a. existential introduction)

- If P(c) is true, then $(\exists x)$ P(x) is inferred.
- Example

  eats(Ziggy, IceCream) $\Rightarrow$ $(\exists x)$ eats(Ziggy, x)
- All instances of the given constant symbol are replaced by the new variable symbol
- Note that the variable symbol cannot already exist anywhere in the expression

Translating English to FOL

Every gardener likes the sun.

$\forall x\ gardener(x) \rightarrow likes(x,Sun)$

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

$\exists x\ \forall t\ \ person(x) \wedge time(t) \rightarrow can\text{-}fool(x,t)$

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

$\forall x\ \exists t\ (person(x) \rightarrow time(t) \wedge can\text{-}fool(x,t))$

$\forall x\ (person(x) \rightarrow \exists t\ (time(t) \wedge can\text{-}fool(x,t))$

Equivalent

All purple mushrooms are poisonous.

$\forall x\ (mushroom(x) \wedge purple(x)) \rightarrow poisonous(x)$

No purple mushroom is poisonous.

$\neg \exists x\ purple(x) \wedge mushroom(x) \wedge poisonous(x)$

$\forall x\ \ (mushroom(x) \wedge purple(x)) \rightarrow \neg poisonous(x)$

Equivalent

There are exactly two purple mushrooms.

$\exists x\ \exists y\ mushroom(x) \wedge purple(x) \wedge mushroom(y) \wedge purple(y) \ \hat{}\ \neg(x=y) \wedge \forall z\ (mushroom(z) \wedge purple(z)) \rightarrow ((x=z) \vee (y=z))$

Clinton is not tall.

$\neg tall(Clinton)$

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

$\forall x\ \forall y\ above(x,y) \leftrightarrow (on(x,y) \vee \ \exists z\ (on(x,z) \wedge above(z,y)))$

Example: A simple genealogy KB by FOL

- **Build a small genealogy knowledge base using FOL that**
  - contains facts of immediate family relations (spouses, parents, etc.)
  - contains definitions of more complex relations (ancestors, relatives)
  - is able to answer queries about relationships between people
- **Predicates:**
  - parent(x, y), child(x, y), father(x, y), daughter(x, y), etc.
  - spouse(x, y), husband(x, y), wife(x,y)
  - ancestor(x, y), descendant(x, y)
  - male(x), female(y)
  - relative(x, y)
- **Facts:**
  - husband(Joe, Mary), son(Fred, Joe)
  - spouse(John, Nancy), male(John), son(Mark, Nancy)
  - father(Jack, Nancy), daughter(Linda, Jack)
  - daughter(Liz, Linda)
  - etc.

- **Rules for genealogical relations**
  - $(\forall x,y)$ parent(x, y) $\leftrightarrow$ child (y, x)
    $(\forall x,y)$ father(x, y) $\leftrightarrow$ parent(x, y) $\wedge$ male(x) (similarly for mother(x, y))
    $(\forall x,y)$ daughter(x, y) $\leftrightarrow$ child(x, y) $\wedge$ female(x) (similarly for son(x, y))
  - $(\forall x,y)$ husband(x, y) $\leftrightarrow$ spouse(x, y) $\wedge$ male(x) (similarly for wife(x, y))
    $(\forall x,y)$ spouse(x, y) $\leftrightarrow$ spouse(y, x) (**spouse relation is symmetric**)
  - $(\forall x,y)$ parent(x, y) $\rightarrow$ ancestor(x, y)
    $(\forall x,y)(\exists z)$ parent(x, z) $\wedge$ ancestor(z, y) $\rightarrow$ ancestor(x, y)
  - $(\forall x,y)$ descendant(x, y) $\leftrightarrow$ ancestor(y, x)
  - $(\forall x,y)(\exists z)$ ancestor(z, x) $\wedge$ ancestor(z, y) $\rightarrow$ relative(x, y)
    (related by common ancestry)
    $(\forall x,y)$ spouse(x, y) $\rightarrow$ relative(x, y) (related by marriage)
    $(\forall x,y)(\exists z)$ relative(z, x) $\wedge$ relative(z, y) $\rightarrow$ relative(x, y) (**transitive**)
    $(\forall x,y)$ relative(x, y) $\leftrightarrow$ relative(y, x) **(symmetric**)
- **Queries**
  - ancestor(Jack, Fred) /* the answer is yes */
  - relative(Liz, Joe) /* the answer is yes */
  - relative(Nancy, Matthew)
    /* no answer in general, no if under closed world assumption */
  - $(\exists z)$ ancestor(z, Fred) $\wedge$ ancestor(z, Liz)

Semantics of FOL

- **Domain M**: the set of all objects in the world (of interest)
- **Interpretation I**: includes
  - Assign each constant to an object in M
  - Define each function of n arguments as a mapping M^n => M
  - Define each predicate of n arguments as a mapping M^n => {T, F}
  - Therefore, every ground predicate with any instantiation will have a truth value
  - In general there is an infinite number of interpretations because |M| is infinite
- **Define logical connectives**: **~, ^, ∨, =>, <=>** as in PL
- **Define semantics of (∀x) and (∃x)**
  - (∀x) P(x) is true iff P(x) is true under all interpretations
  - (∃x) P(x) is true iff P(x) is true under some interpretation

- **Model**: an interpretation of a set of sentences such that every sentence is *True*
- **A sentence is**
  - **satisfiable** if it is true under some interpretation
  - **valid** if it is true under all possible interpretations
  - **inconsistent** if there does not exist any interpretation under which the sentence is true
- **Logical consequence**: S |= X if all models of S are also models of X

Axioms, definitions and theorems

- **Axioms** are facts and rules that attempt to capture all of the (important) facts and concepts about a domain; axioms can be used to prove **theorems**
  - Mathematicians don't want any unnecessary (dependent) axioms –ones that can be derived from other axioms
  - Dependent axioms can make reasoning faster, however
  - Choosing a good set of axioms for a domain is a kind of design problem
- A **definition** of a predicate is of the form "p(X) $\leftrightarrow$ …" and can be decomposed into two parts
  - **Necessary** description: "p(x) $\rightarrow$ …"
  - **Sufficient** description "p(x) $\leftarrow$ …"
  - Some concepts don't have complete definitions (e.g., person(x))

More on definitions

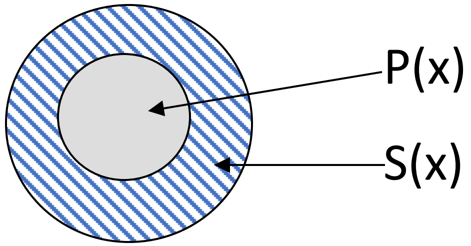
- A **necessary** condition must be satisfied for a statement to be true.
- A **sufficient** condition, if satisfied, assures the statement's truth.
- Duality: "P is sufficient for Q" is the same as "Q is necessary for P."
- Examples: define father(x, y) by parent(x, y) and male(x)
  - parent(x, y) is a necessary (**but not sufficient**) description of
    father(x, y)
    - father(x, y) $\rightarrow$ parent(x, y)
  - parent(x, y) ^ male(x) ^ age(x, 35) is a **sufficient** (**but not necessary**) description of father(x, y):

    father(x, y) $\leftarrow$ parent(x, y) ^ male(x) ^ age(x, 35)
  - parent(x, y) ^ male(x) is a **necessary and sufficient** description of father(x, y)

    parent(x, y) ^ male(x) $\leftrightarrow$ father(x, y)

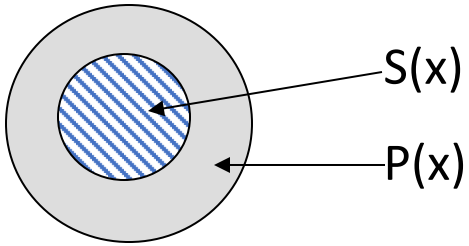
More on definitions

S(x) is a necessary condition of P(x)



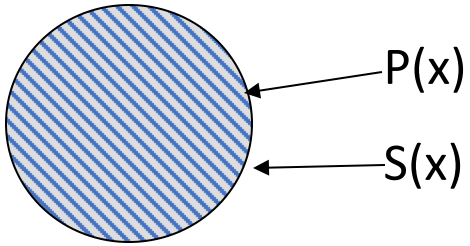
$(\forall x)\ P(x) \Rightarrow S(x)$

S(x) is a sufficient condition of P(x)



$(\forall x)\ P(x) \Leftarrow S(x)$

S(x) is a necessary and sufficient condition of P(x)



$(\forall x)\ P(x) \Leftrightarrow S(x)$

Higher-order 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"

  $\forall f\ \forall g\ (f = g) \leftrightarrow (\forall x\ f(x) = g(x))$
- Example: (quantify over predicates)

  $\forall r\ \text{transitive}(\ r\ ) \rightarrow (\forall xyz)\ r(x,y) \wedge r(y,z) \rightarrow r(x,z))$
- More expressive, but **undecidable**. (there isn't an effective algorithm to decide whether all sentences are valid)
  - First-order logic is decidable only when it uses predicates with only one argument.

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) $\wedge$ $\forall$y (king(y) $\rightarrow$ x=y)
  - $\exists$x king(x) $\wedge$ $\neg\exists$y (king(y) $\wedge$ x$\neq$y)
  - $\exists$! x king(x)
- "Every country has exactly one ruler"
  - $\forall$c country(c) $\rightarrow$ $\exists$! r ruler(c,r)
- Iota operator: "ι x P(x)" means "the unique x such that p(x) is true"
  - "The unique ruler of Freedonia is dead"
  - dead(ι x ruler(freedonia,x))

Notational differences

- **Different symbols** for *and, or, not, implies, ...*
  - $\forall$ $\exists$ $\Rightarrow$ $\Leftrightarrow$ $\wedge$ $\vee$ $\neg$ $\bullet$ $\supset$
  - p v (q ^ r)
  - p + (q * r)
  - etc
- **Prolog**

  cat(X) :- furry(X), meows (X), has(X, claws)
- **Lispy notations**

```
(forall ?x (implies (and (furry ?x)
                         (meows ?x)
                         (has ?x claws))
                    (cat ?x)))
```

Logical agents for the Wumpus World

Three (non-exclusive) agent architectures:

- Reflex agents
  - Have rules that classify situations, specifying how to react to each possible situation
- Model-based agents
  - Construct an internal model of their world
- Goal-based agents
  - Form goals and try to achieve them

A simple reflex agent

- Rules to map percepts into observations:

  $\forall b,g,u,c,t\ \text{Percept}([\text{Stench}, b, g, u, c], t) \rightarrow \text{Stench}(t)$

  $\forall s,g,u,c,t\ \text{Percept}([s, \text{Breeze}, g, u, c], t) \rightarrow \text{Breeze}(t)$

  $\forall s,b,u,c,t\ \text{Percept}([s, b, \text{Glitter}, u, c], t) \rightarrow \text{AtGold}(t)$

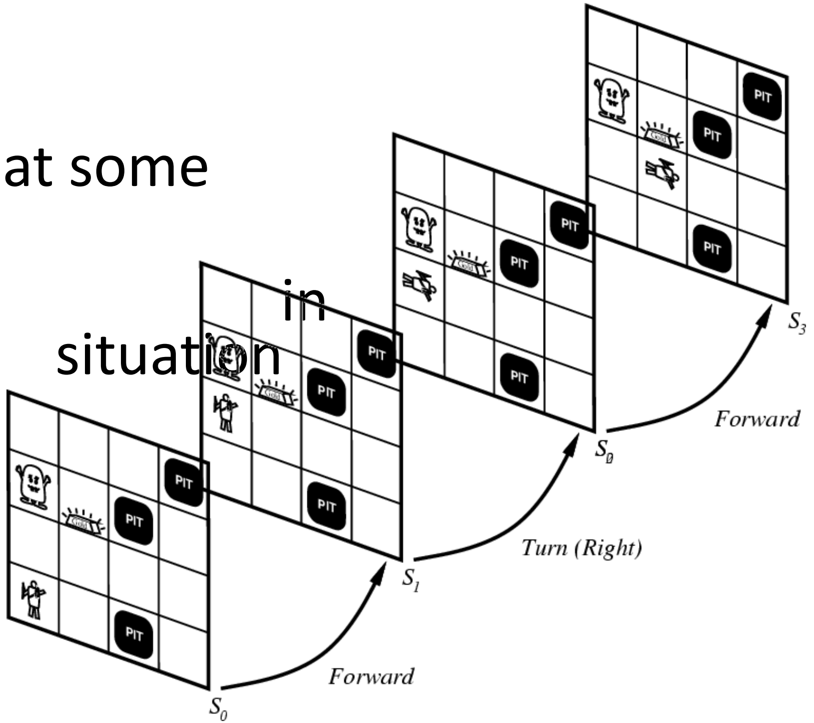
- Rules to select an action given observations:

  $\forall t\ \text{AtGold}(t) \rightarrow \text{Action}(\text{Grab}, t);$

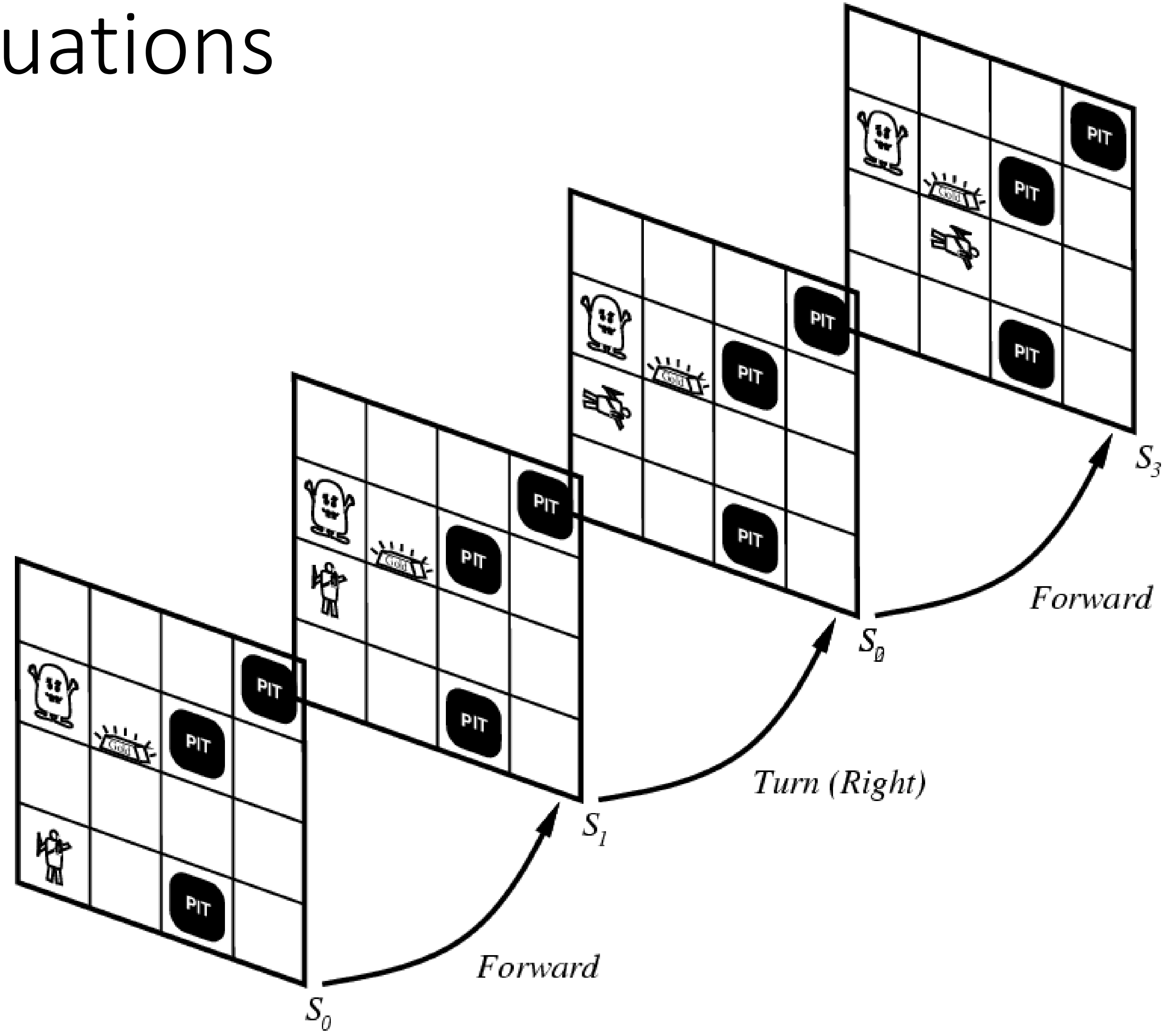
- Some difficulties:
  - Consider Climb. There is no percept that indicates the agent should climb out – position and holding gold are not part of the percept sequence
  - Loops – the percept will be repeated when you return to a square, which should cause the same response (unless we maintain some internal model of the world)

Representing change

- Representing change in the world in logic can be tricky.
- One way is just to change the KB
  - Add and delete sentences from the KB to reflect changes
  - How do we remember the past, or reason about changes?
- **Situation calculus** is another way
- A **situation** is a snapshot of the world at some instant in time
- When the agent performs an action A in situation S1, the result is a new situation S2.



Situations


PIT
PIT
PIT
Forward
S_0
PIT
PIT
PIT
Turn (Right)
S_1
PIT
PIT
PIT
Forward
S_2
PIT
PIT
PIT
S_3

Situation calculus

- A **situation** is a snapshot of the world at an interval of time during which nothing changes
- Every true or false statement is made with respect to a particular situation.
  - Add **situation variables** to every predicate.
  - at(Agent,1,1) becomes at(Agent,1,1,s0): at(Agent,1,1) is true in situation (i.e., state) s0.
  - Alternatively, add a special 2nd-order predicate, **holds(f,s),** that means "f is true in situation s." E.g., holds(at(Agent,1,1),s0)
- Add a new function, **result(a,s),** that maps a situation s into a new situation as a result of performing action a. For example, result(forward, s) is a function that returns the successor state (situation) to s
- Example: The action agent-walks-to-location-y could be represented by
  - $(\forall x)(\forall y)(\forall s)$ (at(Agent,x,s) $\wedge \neg$onbox(s)) $\rightarrow$ at(Agent,y,result(walk(y),s))

Deducing hidden properties

- From the perceptual information we obtain in situations, we can **infer properties of locations**

  $\forall l,s\ at(Agent,l,s) \wedge Breeze(s) \rightarrow Breezy(l)$

  $\forall l,s\ at(Agent,l,s) \wedge Stench(s) \rightarrow Smelly(l)$

- Neither Breezy nor Smelly need situation arguments because pits and Wumpuses do not move around

Deducing hidden properties II

- We need to write some rules that relate various aspects of a single world state (as opposed to across states)
- There are two main kinds of such rules:
  - **Causal rules** reflect the assumed direction of causality in the world:

    $(\forall l1,l2,s)$ At(Wumpus,l1,s) $\wedge$ Adjacent(l1,l2) $\rightarrow$ Smelly(l2)

    $(\forall$ l1,l2,s) At(Pit,l1,s) $\wedge$ Adjacent(l1,l2) $\rightarrow$ Breezy(l2)

    Systems that reason with causal rules are called **model-based reasoning systems**
  - **Diagnostic rules** infer the presence of **hidden properties** directly from the percept-derived information. We have already seen two diagnostic rules:

    $(\forall$ l,s) At(Agent,l,s) $\wedge$ Breeze(s) $\rightarrow$ Breezy(l)

    $(\forall$ l,s) At(Agent,l,s) $\wedge$ Stench(s) $\rightarrow$ Smelly(l)

Representing change: The frame problem

- **Frame axioms**: If property x doesn't change as a result of applying action a in state s, then it stays the same.
  - $On\ (x, z, s) \wedge Clear\ (x, s) \rightarrow$
    $On\ (x, table, Result(Move(x, table), s)) \wedge$
    $\neg On(x, z, Result\ (Move\ (x, table), s))$
  - $On\ (y, z, s) \wedge y \neq x \rightarrow On\ (y, z, Result\ (Move\ (x, table), s))$
  - The proliferation of frame axioms becomes very cumbersome in complex domains

The frame problem II

- **Successor-state axiom**: General statement that characterizes every way in which a particular predicate can become true:
  - Either it can be **made true**, or it can **already be true and not be changed**:
  - On (x, table, Result(a,s)) $\leftrightarrow$
    [On (x, z, s) $\wedge$ Clear (x, s) $\wedge$ a = Move(x, table)] $\wedge$
    [On (x, table, s) $\wedge$ a $\neq$ Move (x, z)]
- In complex worlds, where you want to reason about longer chains of action, even these types of axioms are too cumbersome
  - Planning systems use special-purpose inference methods to reason about the expected state of the world at any point in time during a multi-step plan

Qualification problem

- Qualification problem:
  - How can you possibly characterize every single effect of an action, or every single exception that might occur?
  - When I put my bread into the toaster, and push the button, it will become toasted after two minutes, unless…
    - The toaster is broken, or…
    - The power is out, or…
    - I blow a fuse, or…
    - A neutron bomb explodes nearby and fries all electrical components, or…
    - A meteor strikes the earth, and the world we know it ceases to exist, or…

Ramification problem

- Similarly, it’s just about impossible to characterize every side effect of every action, at every possible level of detail:
    - When I put my bread into the toaster, and push the button, the bread will become toasted after two minutes, and…
        - The crumbs that fall off the bread onto the bottom of the toaster over tray will also become toasted, and…
        - Some of the aforementioned crumbs will become burnt, and…
        - The outside molecules of the bread will become “toasted,” and…
        - The inside molecules of the bread will remain more “breadlike,” and…
        - The toasting process will release a small amount of humidity into the air because of evaporation, and…
        - The heating elements will become a tiny fraction more likely to burn out the next time I use the toaster, and…
        - The electricity meter in the house will move up slightly, and…

Knowledge engineering!

- Modeling the “right” conditions and the “right” effects at the “right” level of abstraction is very difficult
- Knowledge engineering (creating and maintaining knowledge bases for intelligent reasoning) is an entire field of investigation
- Many researchers hope that automated knowledge acquisition and machine learning tools can fill the gap:
  - Our intelligent systems should be able to **learn** about the conditions and effects, just like we do!
  - Our intelligent systems should be able to learn when to pay attention to, or reason about, certain aspects of processes, depending on the context!

Preferences among actions

- A problem with the Wumpus world knowledge base that we have built so far is that it is difficult to decide which action is best among a number of possibilities.
- For example, to decide between a forward and a grab, axioms describing when it is OK to move to a square would have to mention glitter.
- This is not modular!
- We can solve this problem by **separating facts about actions from facts about goals**. This way our **agent can be reprogrammed just by asking it to achieve different goals**.

Preferences among actions

- The first step is to describe the desirability of actions independent of each other.
- In doing this we will use a simple scale: actions can be Great, Good, Medium, Risky, or Deadly.
- Obviously, the agent should always do the best action it can find:

$$(\forall a,s)\ Great(a,s) \rightarrow Action(a,s)$$

$$(\forall a,s)\ Good(a,s) \wedge \neg(\exists b)\ Great(b,s) \rightarrow Action(a,s)$$

$$(\forall a,s)\ Medium(a,s) \wedge (\neg(\exists b)\ Great(b,s) \vee Good(b,s)) \rightarrow Action(a,s)$$

...

Preferences among actions

- We use this action quality scale in the following way.
- Until it finds the gold, the basic strategy for our agent is:
  - Great actions include picking up the gold when found and climbing out of the cave with the gold.
  - Good actions include moving to a square that’s OK and hasn't been visited yet.
  - Medium actions include moving to a square that is OK and has already been visited.
  - Risky actions include moving to a square that is not known to be deadly or OK.
  - Deadly actions are moving into a square that is known to have a pit or a Wumpus.

Goal-based agents

- Once the gold is found, it is necessary to change strategies. So now we need a new set of action values.
- We could encode this as a rule:
  - $(\forall s)$ Holding(Gold,s) $\rightarrow$ GoalLocation([1,1]),s)
- We must now decide how the agent will work out a sequence of actions to accomplish the goal.
- Three possible approaches are:
  - **Inference**: good versus wasteful solutions
  - **Search**: make a problem with operators and set of states
  - **Planning**: to be discussed later