CSCI 3230

Fundamentals of Artificial Intelligence

Chapter 8

FIRST ORDER LOGIC

Fall 2013

Outline

- Syntax and Semantics
- Extensions and Notational Variations
- Using First-Order Logic (FOL)
- Logical Agents for the Wumpus World
- A Simple Reflex Agent
- Representing Change in the World (Model-Based)
- Deducing Hidden Properties of the World
- Toward a Goal-Based Agent
- Knowledge Engineering Process

First-order logic

First-order Predicate Logic

basic categories and their relations

- First-order logic makes a stronger set of ontological (entities) commitments. The world consists of objects, i.e., things with individual identities and properties that distinguish them from other objects.
- Among these objects, various relations hold. Some relations are functions – relations with only one "value" for a given "input".
    - Objects: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries…terms
    - Relations (Predicate): brother of, bigger than, inside, part of, has color, occurred after, owns…
    - Properties (unary relations): red, round, bogus, prime, multistoried…
    - Functions: father of, best friend, third inning of, one more than…
    - Facts: "One plus two equals three", "Squares neighboring the wumpus are smelly"…(atomic sentences)

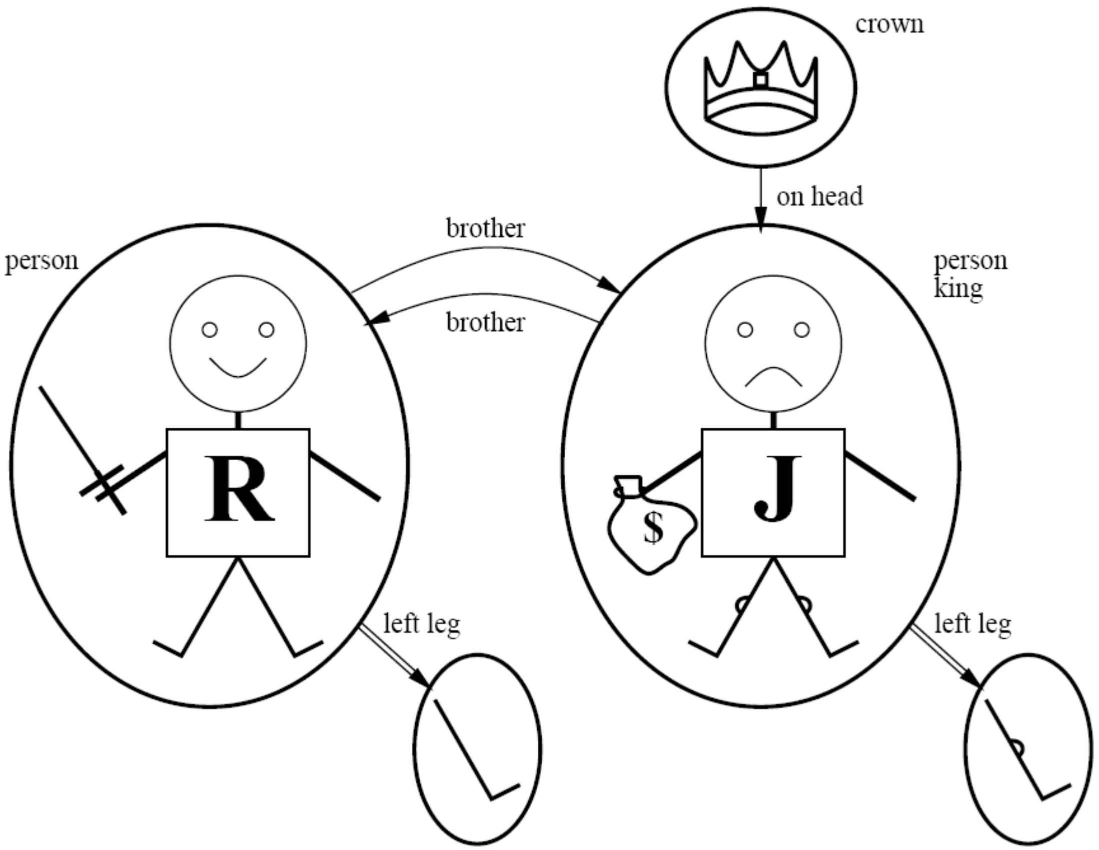
First-order logic

Objects?

Binary relations?

Unary relations
/properties?

Unary function?



predicates

Fig.8.0 A model containing 5 objects, 2 binary relations (brother, on head), 3 unary relations/properties (person, king, crown), and 1 unary function (left-leg).

First-order logic (FOL)

- FOL commits to the existence of objects and relations, it does not make an ontological commitment to such things as categories, time, and events. cf. OO classes
  E.g. John fell in love with Mary 2 hours ago.

- Existence defined by quantifiers: ∀ | ∃

- But FOL is universal in the sense that it can express anything (any algorithm) that can be programmed. cf. Turing Machine

Syntax and Semantics

- In propositional logic every expression is a sentence, which represents a fact.
- FOL has sentences, but also terms, which represent objects. Constant symbols, variables, and function symbols are used to build terms, and quantifiers and predicate symbols are used to build sentences.
  - Constant symbols: A, B, C, John…
    - An interpretation must specify which object in the world is referred to by each constant symbol.

Syntax and Semantics

述語

- Predicate symbols: Brother, Bigger…
  - An interpretation specifies that a predicate symbol refers to a particular relation in the model. In a model, the relation is defined by the set of tuples of objects that satisfy it. A tuple is a collection of objects arranged in a fixed order. (Extensional definition)
    E.g. { <John, Peter> … <examples or brothers>}

- Function symbols: Cosine, FatherOf, LeftLegOf…
  - Some relations are functional – that is, any given object (John) is related to exactly one other object (LeftLeg) by the relation. E.g.LeftLegOf(John).

Syntax and Semantics

Sentence → Atomic-Sentence

  | Sentence Connective Sentence

  | Quantifier Variable, … Sentence

  | ¬ Sentence | (Sentence)

Atomic-Sentence → Predicate(*Term*, …) | *Term* = *Term* *// facts*

Term → Function(*Term*, …) | Constant | Variable *// objects*

Connective → ⇒ | ∧ | ∨ | ⇔

Quantifier → ∀ | ∃

Constant → *A* | X_1 | *John* | …

Variable → *a* | *x* | *s*

Predicate → Before | HasColor | Raining | … *// relations:- unary(property) & binary*

Function → Mother | LeftLegof | … *// objects*

Fig 8.1 The syntax of first-order logic (with equality) in BNF (Backus-Naur Form)

Syntax and Semantics

- Terms
  - A term is a logical expression that refers to an object. Constant symbols are therefore terms.
  - The formal semantics of terms is straightforward. An interpretation specifies a functional relation referred to by the *function symbol*, and objects(John) referred to by the terms that are its arguments. E.g. John's left leg: LeftLegOf(John)
- Atomic sentences
  - Atomic sentences that state facts. An atomic sentence is formed from a predicate symbol followed by a parenthesized list of terms. e.g.,

    Brother (Richard, John)

    states, under the interpretation given before, that Richard the Lionheart is the brother of King John.

Syntax and Semantics

Atomic sentences can have arguments that are complex terms:

Married(FatherOf(Richard), MotherOf(John))

states that Richard the Lionheart's father is married to King John's mother

(again, under a suitable interpretation)

- An atomic sentences is true if the relation(Teacher) referred to by the predicate symbol holds between the objects (Leung, CSCI 3230) referred to by the arguments.
  E.g. Teacher (Leung, CSCI 3230)

Syntax and Semantics

- Complex sentences
  - We can use logical connectives to construct more complex sentences, like propositional calculus.
    - *Brother(Richard, John)* ∧ *Brother(John, Richard)* is true when John is the brother of Richard and Richard is the brother of John.
    - *Older(John, 30)* ∨ *Younger(John, 30)* is true when

      John is older than 30 or John is younger than 30.
    - *Older(John, 30)* ⇒ ¬ *Younger(John, 30)* states that

      if John is older than 30, then he is not younger than 30.
    - ¬ *Brother(Robin, John)* is true when

      Robin is not the brother of John.

Syntax and Semantics

- Quantifiers
  - Universal quantification (∀) (read: for all)
    - "All cats are mammals"

$$\forall x\ Cat(x) \Rightarrow Mammal(x)$$

    - The preceding sentence is therefore equivalent to

$$\begin{aligned} Cat(Spot) &\Rightarrow Mammal(Spot) \land \\ Cat(Rebecca) &\Rightarrow Mammal(Rebecca) \land \\ Cat(Felix) &\Rightarrow Mammal(Felix) \land \\ Cat(Richard) &\Rightarrow Mammal(Richard) \land \\ Cat(John) &\Rightarrow Mammal(John) \land \\ &\ldots \end{aligned}$$

    - Thus, it is true iff all these sentences are true, i.e., if P is true for all object x in the <u>universe</u>. Hence ∀ is called a universal quantifier.

Syntax and Semantics

- Existential quantification (∃) (read: there exist(s))
  - We can make a statement about some object in the universe without naming it using an existential quantifier. To say, e.g., that Spot has a sister who is a cat:

    ∃x Sister(x, Spot) ∧ Cat(x)
  - In general ∃x is true for some object in the universe.

    (Sister(Spot, Spot) ∧ Cat(Spot)) ∨
    (Sister(Rebecca, Spot) ∧ Cat(Rebecca)) ∨
    (Sister(Felix, Spot) ∧ Cat(Felix)) ∨
    (Sister(Richard, Spot) ∧ Cat(Richard)) ∨
    (Sister(John, Spot) ∧ Cat(John)) ∨
    …

    (?? c.f. propositional logic??) The existentially quantified sentence is *true* just in case at least *one* of these disjuncts is true.

Syntax and Semantics

- Nested quantifiers
  - Express more complex sentences using multiple quantifiers. E.g., “For all x and all y, if x is the parent of y then y is the child of x” becomes

$$\forall x, y \ Parent(x, y) \Rightarrow Child(y, x)$$

    - ∀x, y is equivalent to ∀x ∀y

All persons

  - We can have mixtures. “Everybody loves somebody” means that for every person, there is someone that person loves:

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

  - To say “There is someone who is loved by everyone” we write

?

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

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

Syntax and Semantics

- The order of quantification is ∴ important.
- ∀x (∃y P(x, y)), where P(x, y) says that *every* object x in the relation P to *some* y.
- ∃x (∀y P(x, y)) says that there is *some* object in the world that has a particular property, namely the property of being related by P to every object in the world.
- A minor difficulty – 2 quantifiers with the same variable name.
  - ∀x [Cat(x) ∨ (∃x Brother(Richard, x))] Standardize apart; overloaded variable name;
  - One interpretation: ∃x Brother(Richard, x) is a sentence about Richard (that he has a brother), not about x: So putting a ∀x outside it has no effect.
- The term well-formed formula or wff is sometimes used for sentences that have all their variables properly introduced. E.g. ∀x P(x, y), y is not properly introduced.

Syntax and Semantics

- Connections between ∀ and ∃
  - The 2 quantifiers are intimately connected with each other, through negation.
  - When one says that everyone dislikes parsnips, one is also saying that there does not exist someone who likes them; and vise versa:

    $\forall x\ \neg Likes(x, Parsnips) \equiv \neg \exists x\ Likes(x, Parsnips)$
  - "Everyone like ice-cream" means that there is no one who does not like ice-cream:

    $\forall x\ Likes(x, Ice\text{-}cream) \equiv \neg \exists x\ \neg Likes(x, Ice\text{-}cream)$
  - De Morgan rules: thus, we do not really need both ∀ and ∃, just as we do not need both ∧ and ∨. $\neg(\alpha \wedge \beta) \equiv (\neg \alpha \vee \neg \beta)$;
  - $\neg \forall x\ R(x, y)) \equiv \exists x\ \neg R(x, y)$??How to convert??
  - Double negation + De Morgan: $\neg(\neg \forall x\ Likes(x, Ice\text{-}cream)) \equiv \neg \exists x\ \neg Likes(x, Ice\text{-}cream)$

Syntax and Semantics

Equality

- FOL includes one more way to make atomic sentences, other than using a predicate and terms. We can use the equality symbol to make statements that 2 terms refer to the same object. E.g.,

  Prefix function

  Father (John) = Henry *// infix*

- Equality can be viewed as a predicate symbol with a predefined meaning, i.e., identity relation.

Syntax and Semantics

- The equality symbol can be used to describe the properties of a given function, as we did above for the Father symbol.

  Father (John) = Henry

- It can also be used with negation to insist that two terms are not the same object. To say that Spot has at least 2 sisters:

  $\exists x, y \; Sister(Spot, x) \wedge Sister(Spot, y) \wedge \neg(x = y)$

- Simply writing ∃x, y Sister(Spot, x) ∧ Sister(Spot, y) would not assert the existence of 2 distinct sisters, because nothing says that x and y have to be different.

Extensions and Notational Variations
-Higher-order logic

- FOL gets its name from the fact that one can quantify over objects (the first-order entities that actually exist in the world) but not over relations or functions on those objects.

- Higher-order logic allows us to quantify over relations and functions as well as over objects. E.g., in higher-order logic 2 objects are equal iff all properties, p, applied to them are equivalent:

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

- 2 functions, f & g, are equal iff they have the same value for all arguments:

$$\forall f, g\ (f = g) \Leftrightarrow \forall x\ f(x) = g(x))$$

- Higher-order logics have strictly more expressive power. However, logicians have little understanding of how to reason effectively with sentences in higher-order logic, and the general problem is undecidable.

Extensions and Notational Variations

–Functional and predicate expressions using the λ operator

- Useful to be able to construct complex predicates and functions from simpler components (e.g. P ∧ Q), or complex terms from simpler ones (e.g. $x^2 + y^3$).
- The operator λ (the Greek letter lambda) is used for this purpose. The function that takes the difference of the squares of its first and second arguments:

$$\lambda\, x, y\;\; x^2 - y^2$$

- This λ-expression can then be applied to arguments to yield a logical term in the same way that an ordinary, named function can:

$$(\lambda x, y\ x^2 - y^2)(25{,}24) = 25^2 - 24^2 = 49$$

$f(x,y)$

Extensions and Notational Variations
-Functional and predicate expressions using the λ operator

- E.g., the two-place predicate "are of differing gender and of the same address" can be written

$$\lambda x, y \ \ Gender(x) \neq Gender(y) \ \wedge \ Address(x) = Address(y)$$

the application of a predicate λ-expression to an appropriate number of arguments yields a logical sentence.

- The use of λ in this way does not increase the formal expressive power of FOL, because any sentence that includes a λ-expression can be rewritten by "plugging in" its arguments to yield a standard term or sentence.

Extensions and Notational Variations
–The uniqueness quantifier ∃!

- Some authors use the notation

  $$\exists! x \; King(x)$$

  to mean "there exists a unique object x satisfying King(x)" or more informally, "there's exactly one King."

  –not a new quantifier, ∃!, but a convenient abbreviation for the longer sentence

  $$\exists x \; King(x) \land \forall y \; King(y) \Rightarrow x = y$$

- A more complex example is “Every country has exactly one ruler”:

  $$\forall c \; Country\ (c) \Rightarrow \exists! r \; Ruler\ (r, c)$$

Extensions and Notational Variations
–The uniqueness operator ι

- More convenient to have a term representing the unique object directly. The notation ιx P(x) is commonly used.

  (The symbol ι is the Greek letter iota).

  To say that "the unique ruler of Freedonia is dead" or equivalently "the r that is the ruler of Freedonia is dead", we would write

  Dead(ι r Ruler(r, Freedonia))

- This is just an abbreviation for the following sentence:

  $\exists! r\ Ruler(r, Freedonia) \wedge \forall s\ Ruler(s, Freedonia) \Rightarrow Dead(s)$

Extensions and Notational Variations
-Notational variations

Some variations: (see table)

Prolog has 3 main differences:

- It uses uppercase letters for variables and lowercase for constants.
- Prolog also reverses the order of implications, writing Q:-P instead of P ⇒ Q.
- A comma is used both to separate arguments and for conjunction, and a period marks the end of a sentence:

```
Cat (X) :- furry(X), meows(X), has(X, claws).
```

Because Lisp does not distinguish between uppercase and lowercase symbols, variables are usually distinguished by an initial ? or $ character, e.g.:

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

Using First-Order Logic
–The kinship domain

In knowledge representation, a domain is a section of the world about which we wish to express some knowledge.

The kinship domain

includes facts such as "Elizabeth is the mother of Charles" and "Charles is the father of William." and

rules such as "If x is the mother of y and y is a parent of z, then x is a grandmother of z."

The objects in our domain are people whose properties include gender, related by relations such as parenthood, brotherhood, marriage, and so on. We ∴ have 2 unary predicates, Male and Female.

property

Using First-Order Logic
-The kinship domain

Most of the kinship relations will be binary predicates: *Parent, Sibling, Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, Grandparent, Grandchild, Cousin, Aunt, Uncle, Nephew(male), Niece(f).* *How to define Mother from above terms??*

We will use functions for Mother and Father, because every person has exactly one of each of these (at least according to nature's design).

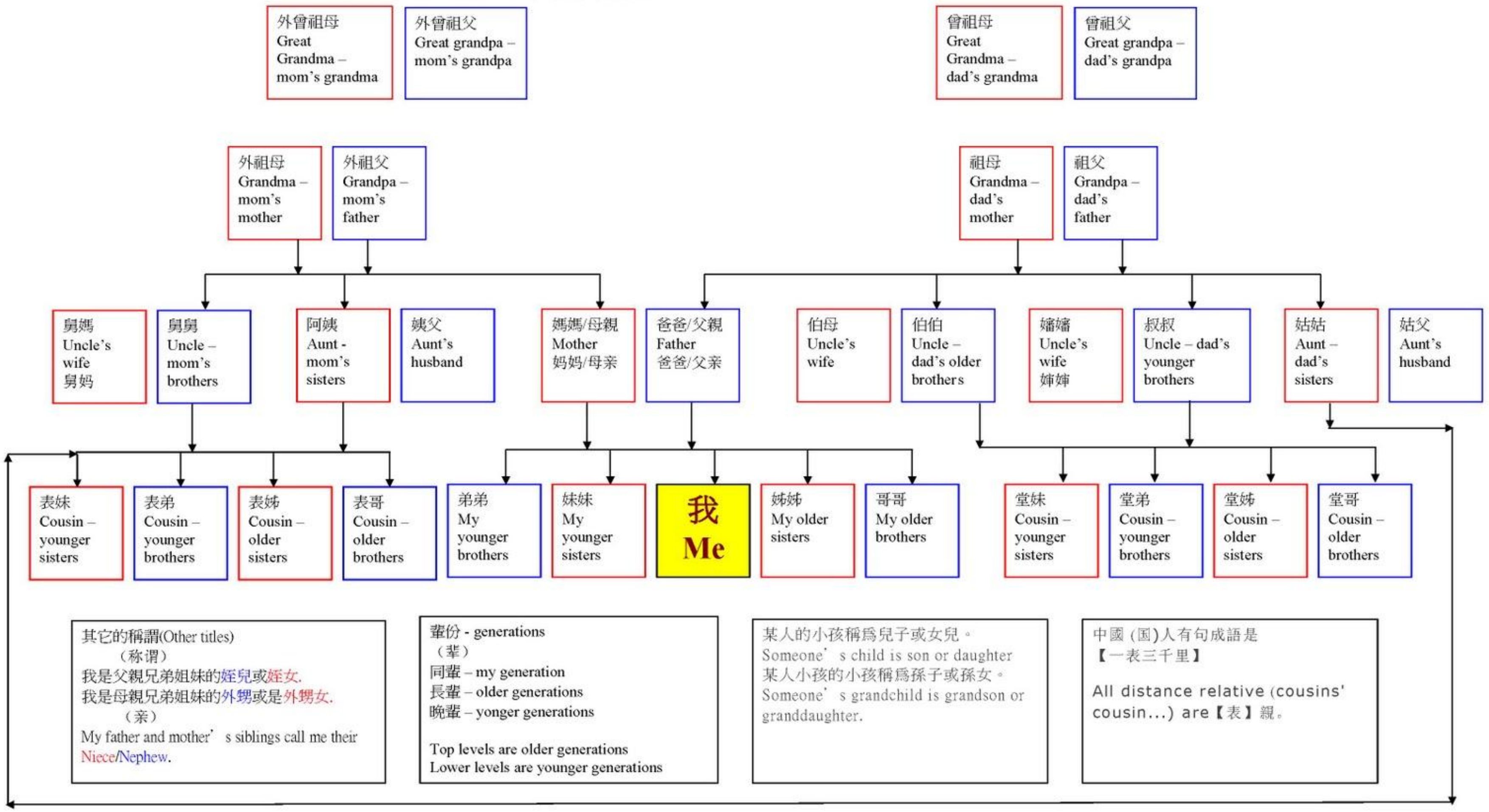
E.g.. One's Mother is one's female parent:

$$\forall m, c\ \overset{\text{function}}{\text{Mother}}(c) = m \Leftrightarrow \overset{\text{property}}{\text{Female}}(m) \wedge \overset{\text{relation}}{\text{Parent}}(m, c)$$

One's husband is one's male spouse:

$$\forall w, h\ \text{Husband}(h, w) \Leftrightarrow \text{Male}(h) \wedge \text{Spouse}(h, w)$$

中英文親戚稱謂表/亲戚称谓表/Relatives


外曾祖母 Great Grandma – mom's grandma
外曾祖父 Great grandpa – mom's grandpa
曾祖母 Great Grandma – dad's grandma
曾祖父 Great grandpa – dad's grandpa
外祖母 Grandma – mom's mother
外祖父 Grandpa – mom's father
祖母 Grandma – dad's mother
祖父 Grandpa – dad's father
舅媽 Uncle's wife 舅妈
舅舅 Uncle – mom's brothers
阿姨 Aunt - mom's sisters
姨父 Aunt's husband
媽媽/母親 Mother 妈妈/母亲
爸爸/父親 Father 爸爸/父亲
伯母 Uncle's wife
伯伯 Uncle – dad's older brothers
嬸嬸 Uncle's wife 婶婶
叔叔 Uncle – dad's younger brothers
姑姑 Aunt – dad's sisters
姑父 Aunt's husband
表妹 Cousin – younger sisters
表弟 Cousin – younger brothers
表姊 Cousin – older sisters
表哥 Cousin – older brothers
弟弟 My younger brothers
妹妹 My younger sisters
我 Me
姊姊 My older sisters
哥哥 My older brothers
堂妹 Cousin – younger sisters
堂弟 Cousin – younger brothers
堂姊 Cousin – older sisters
堂哥 Cousin – older brothers

其它的稱謂(Other titles)
（称谓）
我是父親兄弟姐妹的姪兒或姪女.
我是母親兄弟姐妹的外甥或是外甥女.
（亲）
My father and mother' s siblings call me their Niece/Nephew.

輩份 - generations
（辈）
同輩 – my generation
長輩 – older generations
晚輩 – yonger generations

Top levels are older generations
Lower levels are younger generations

某人的小孩稱爲兒子或女兒。
Someone' s child is son or daughter
某人小孩的小孩稱爲孫子或孫女。
Someone' s grandchild is grandson or granddaughter.

中國 (国)人有句成語是
【一表三千里】

All distance relative (cousins' cousin...) are【表】親。

Created by Tulan Hu 2009

Using First-Order Logic
-The kinship domain

Male and female are disjoint categories:

$\forall x\ \text{Male}(x) \Leftrightarrow \neg \text{Female}(x)$

Parent and child are inverse relations:

$\forall p, c\ \text{Parent}(p, c) \Leftrightarrow \text{Child}(c, p)$

A grandparent is a parent of one's parent:

$\forall g, c\ \text{Grandparent}(g, c) \Leftrightarrow \exists p\ \text{Parent}(g, p) \wedge \text{Parent}(p, c)$?$\exists p$?

Sibling (parent)??

A sibling is another child of one's parents:

$\forall x, y\ \text{Sibling}(x, y) \Leftrightarrow x \neq y \wedge \exists p\ \text{Parent}(p, x) \wedge \text{Parent}(p, y)$

Using First-Order Logic
-Axioms, definitions and theorems

- Mathematicians write axioms to capture the basic facts about a domain, define other concepts in terms of these basic facts, and then use the axioms and definitions to prove theorems. But the sentences that are in the knowledge base initially are sometimes called "axioms", or "definitions".
- Important question: how do we know when we have written down enough axioms to fully specify a domain? In many domains, there is no clearly identifiable basic set.
- The converse problem also arises: do we have too many sentences? E.g., do we need the following sentence, specifying that siblinghood is a symmetric relation?
    - $\forall x, y\ Sibling(x, y) \Leftrightarrow Sibling(y, x)$

Using First-Order Logic
-Axioms, definitions and theorems

- Answer is NO. From Sibling(John, Richard), we can infer that
  - ∃p Parent(p , John) ∧ Parent(p, Richard).

  And from that we can infer? Sibling(Richard, John). (last axiom, p27)

  $\forall x, y\ Sibling(x, y) \Leftrightarrow x \neq y \land \exists p\ Parent(p, x) \land Parent(p, y)$
- In mathematics, an independent axiom is one that cannot be derived from all the other axioms. Mathematicians strive to produce a minimal set of axioms that are all independent.
- In AI, common to include redundant axioms, not because of what can be proved, but because they make proof more efficient.
- An axiom of the form ∀x, y P(x, y) ≡ … is often called a definition of P, because it defines exactly for what objects P does and does not hold. Possible to have several definitions for the same predicate; e.g., a triangle could be defined both as a polygon with 3 sides and as a polygon with 3 angles.

Using First-Order Logic
-The domain of sets

Set is a predicate that is true only of sets. The following eight axioms provide this:

1. The only sets are the empty set and those made by adjoining something to a set.

$$\forall s\ Set(s) \Leftrightarrow (s = EmptySet)$$
$$\lor (\ \exists x, s_2\ \underline{Set(s_2)} \land \underline{s = Adjoin(x\ ,\ s_2)})$$

There does not exist

2. The empty set has no elements adjoined into it. (In other words, there is no way to decompose EmptySet in to a smaller set and an element.)

$$\neg\exists x, s\ Adjoin(x, s) = EmptySet$$

3. Adjoining an element already in the set has no effect:

$$\forall x, s\ Member(x, s) \Leftrightarrow \underline{s = Adjoin(x, s)}$$

Using First-Order Logic
-The domain of sets

4. The only members of a set are the elements that were adjoined into it. We express this recursively, saying that x is a member of s iff s is equal to some set s_2 adjoined with some element y, where either y is the same as x or x is a member of s_2.

$$\forall x, s\ \mathrm{Member}(x, s) \Leftrightarrow$$
$$\exists y, s_2\ (s = \mathrm{Adjoin}(y, s_2) \land (\overset{\text{newly added}}{\underline{x = y}} \lor \overset{\text{x already existed}}{\mathrm{Member}(x, s_2)})))$$

5. A set is subset of another iff all of the first set's members are members of the second set.

$$\forall s_1, s_2\ \mathrm{Subset}(s_1, s_2) \Leftrightarrow$$
$$(\forall x\ \mathrm{Member}(x, s_1) \Rightarrow \mathrm{Member}(x, s_2))$$

6. Two sets are equal iff each is a subset of the other.

$$\forall s_1, s_2\ (s_1 = s_2) \Leftrightarrow (\mathrm{Subset}(s_1, s_2) \land \mathrm{Subset}(s_2, s_1))$$

Using First-Order Logic
–The domain of sets

7. An object is a member of the intersection of two sets if and only if it is a member of each of the sets.

$$\forall x, s_1, s_2\ Member(x, Intersection(s_1, s_2)) \Leftrightarrow Member(x, s_1) \land Member(x, s_2)$$

8. An object is a member of the union of two sets if and only if it is a member of either set.

$$\forall x, s_1, s_2\ Member(x, Union\ (s_1, s_2)) \Leftrightarrow Member(x, s_1) \lor Member(x, s_2)$$

The domain of lists is very similar to the domain of sets. The difference is that lists are ordered, and the same element can appear more than once in a list.

Using First-Order Logic
-Asking questions and getting answers

- To add the kinship sentences to a knowledge base KB, e.g.

  TELL(KB, ($\forall$m, c Mother (c) = m
  $\Leftrightarrow$ Female (m) $\wedge$ Parent (m, c)))
- Now if we tell it

  TELL(KB, Female(Maxi) $\wedge$ Parent(Maxi, Spot)
  $\wedge$ Parent(Spot, Boots)))

  then we can
  - ASK(KB, Grandparent(Maxi, Boots))

  and receive an affirmative answer.

- The sentences added using TELL - called assertions
- The questions asked using ASK - called queries or goals (different to an agent's desired states).

- Thus, a query with existential variables is asking "Is there an x such that ...," and we solve it by providing such an x. The standard form for an answer of this sort is a substitution or binding list - a set of variable/term pairs.

Logical Agents for the Wumpus World

We will consider 3 agent architectures:

1. reflex agents that merely classify their percepts and act accordingly;
2. model-based agents that construct an internal representation of the world and use it to act; and
3. goal-based agents that form goals and try to achieve them. (Goal-based agents are usually also model-based agents.)

```
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 ← t + 1
    return action
```

Fig 8.2 A generic knowledge-based agent

A Simple Reflex Agent

- The simplest possible kind of agent has rules directly connecting percepts to actions. These rules resemble reflexes or instincts. E.g., if the agent sees a glitter, it should do a grab in order to pick up the gold.

  $\forall s, b, u, c, t\ \text{Percept}([s, b, \text{Glitter}, u, c], t) \Rightarrow \text{Action}(\text{Grab}, t)$

- The connection between percept and action can be mediated by rules for perception, which abstract the immediate perceptual input into more useful forms:

  concepts

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

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

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

  …

Then a connection can be made from these predicates to action choices:

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

A Simple Reflex Agent

Limitations of simple reflex agents

- They will have a hard time in the wumpus world. A pure reflex agent cannot know for sure when to climb, because neither having the gold nor being in the start square is part of the percept; they are things the agent knows by forming a representation (model) of the world. (state)
- They also cannot avoid infinite loops. Randomization provides some relief, but risking many fruitless actions.

Representing Change in the World (+Situation) location only (Model Based Agents)

The easiest way to deal with change is simply to change the knowledge base; to erase the sentence that says the agent is at [1,1], and replace it with says at [1,2]. But all knowledge about the past is lost, and it prohibits speculation about different possible futures.

Situation Calculus

It conceives of the world as consisting of a sequence of situations, each of which is a "snapshot" of the state of the world.

Situations are generated from previous situations by actions, as shown in the following figure.

Representing Change in the World (+Situation)

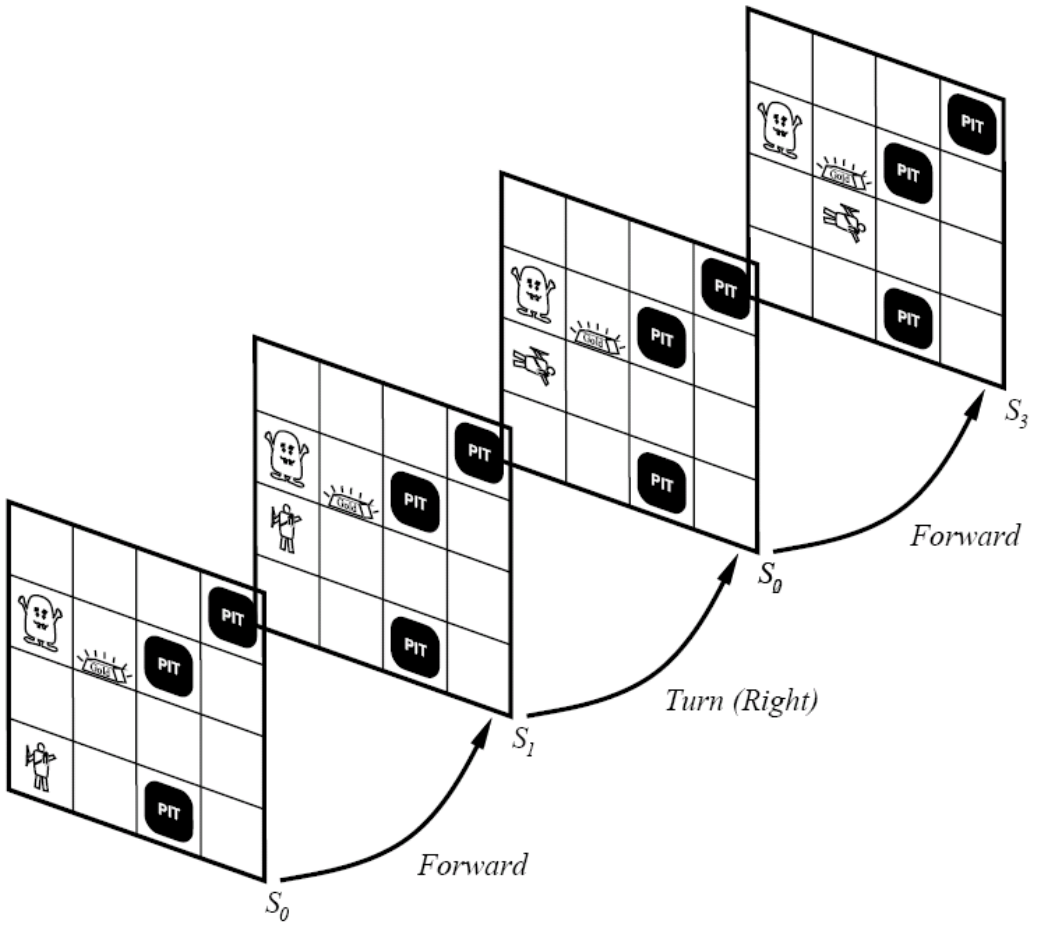


Fig 8.3, In situation calculus, the world is a sequence of situations linked by actions.

Deducing Hidden Properties of the World (Situation: location + properties)

Once the agent knows where it is, it can associate properties, situations (s), with the places (l).

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

$\forall l, s\ At(Agent, l, s) \land Stench(s) \Rightarrow Smelly(l)$

It is useful to know if a place is breezy or smelly because we know that wumpus and the pits cannot move about.

Deducing Hidden Properties of the World (Situation: location + properties)

2 main kinds of synchronic rules: (same world state(time))

- *Causal rules:* Causal rules reflect the assumed direction of causality in the world: some hidden property of the world causes certain percepts to be generated. E.g., we might have rules stating that squares adjacent to wumpuses are smelly and squares adjacent to pits are breezy:

$$\forall l_1, l_2, s\ At(Wumpus, l_1, s) \land Adjacent(l_1, l_2) \Rightarrow Smelly(l_2)$$

$$\forall l_1, l_2, s\ At(Pit, l_1, s) \land Adjacent(l_1, l_2) \Rightarrow Breezy(l_2)$$

?percepts

  - Systems that reason with causal rules are called model-based reasoning systems, which help to understand the reasoning chain explicitly

Deducing Hidden Properties of the World (Situation: location + properties)

Diagnostic rules: Diagnostic rules infer the presence of hidden properties directly from the percept-derived information.

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

For deducing the presence of wumpuses, a diagnostic rule can only draw weak conclusion.

$$\forall l_1, s\ Breezy(l_1) \Rightarrow \exists l_2\ At(Pit, l_2, s) \land Adjacent(l_1, l_2)$$

Symptoms (percepts) $\Rightarrow$ diagnosis

- Bi-conditional sentence can be diagnostic and casual rules:

$$\forall s\ Breezy(s) \Leftrightarrow \exists r\ Pit(r) \land Adjacent(r, s)$$

Toward a Goal-Based Agent

- Once the gold is found, the aim now is to return to the start square as quickly as possible. We would like to infer that the agent now has the goal of being at location [1,1]:

$$\forall s \; Holding(Gold, s) \Rightarrow GoalLocation([1,1], s)$$

- 3 ways to find the action sequence:
- Inference: Write axioms that allow us to ASK the KB for a sequence of actions to achieve the goal safely.
- Search: Use best-first search (Chap.4) to find a path to the goal.
- Planning: Use special-purpose reasoning systems designed to reason about action.

Knowledge Engineering Process

1. Identify the task (problem specification/ definition)
2. Assemble the relevant knowledge – knowledge acquisition (system analysis)
3. Decide on a vocabulary of predicates, function and constants (design)

iterate

4. Encode general knowledge about the domain (coding)
5. Encode a description of the specific problem instance – precepts/ inputs to handle (coding sub-programs, I/O)
6. Pose queries to the inference procedure to get answer (testing & evaluation)
7. Debug the knowledge base (testing & evaluation)