Practical AI
(a.k.a. Business Intelligence)

Abraham Bernstein

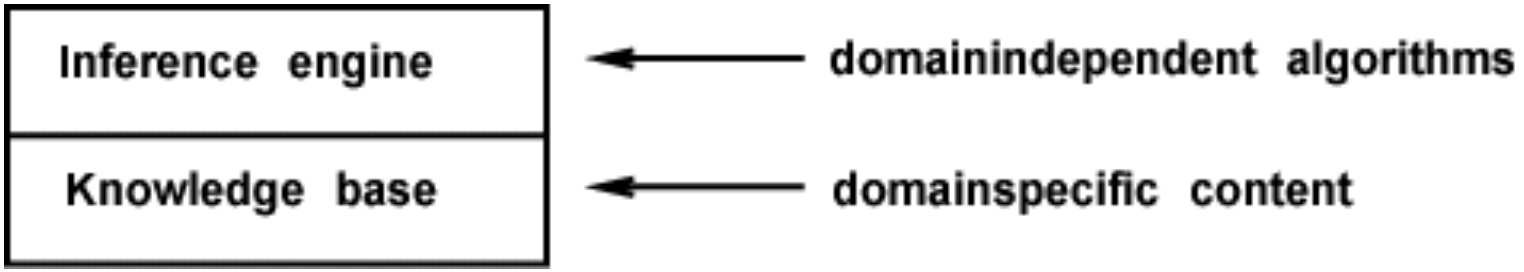
Universität Zürich[UZH]

Dynamic and Distributed Information Systems

Overview

- Logic in general
- First Order Logic (FOL)
  - Why FOL?
  - Syntax and semantics of FOL
  - Fun with sentences
- Reasoning in FOL
- Knowledge Representation (KR)
  - Semantic Networks
  - Knowledge Graphs

Knowledge Bases



- Knowledge base = set of *sentences* in a *formal* language
- **Declarative** approach to building an agent (or other system):
  `Tell` it what it needs to know
- Then it can `Ask` itself what to do – answers should follow from the KB
- Agents can be viewed at the *knowledge level*
  i.e., what they know, regardless of how implemented
- Or at the *implementation level*
  i.e., data structures in KB and algorithms that manipulate them

A simple 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 ← *t* + 1
  return *action*

- The agent must be able to:
  - Represent states, actions, etc.
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions

Logic in general

Logics are formal languages for representing information such that conclusions can be drawn

Syntax defines the sentences in the language

Semantics define the „meaning“' of sentences;
i.e., define *truth* of a sentence in a world

E.g., the language of arithmetic

x+2 ≥ *y* is a sentence; *x*2+*y*> is not a sentence

x+2 ≥ *y* is true iff the number x+2 is no less than the number *y*

x+2 ≥ *y* is true in a world where *x*= *7, y*= *1*

x+2 ≥ *y* is false in a world where *x*= *0, y*= *6*

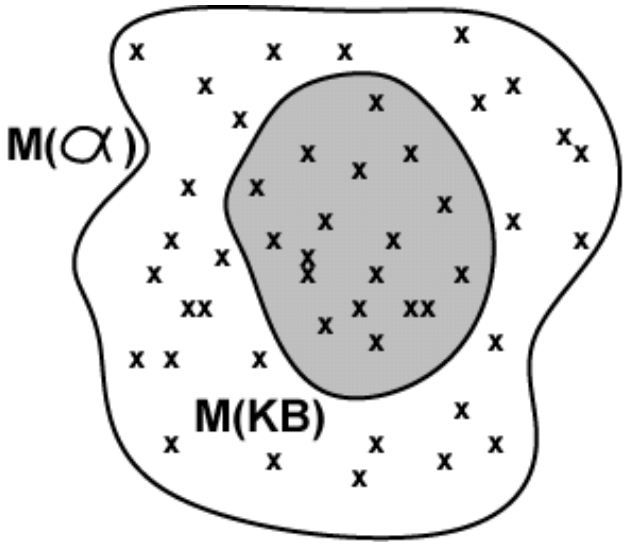
Entailment ⊢

- **Entailment** means that one thing *follows from* another:
- $KB \vdash \alpha$ (≡ α can be derived from *KB)*
- Knowledge base *KB* entails sentence α
  if and only if
  α is true in all worlds where *KB* is true
  - E.g., the *KB* containing "FCZ won"' and "GCZ won" entails "Either FCZ won or GCZ won"
  - E.g., *x+y= 4* entails *4= x+y*

- Entailment is a relationship between sentences (i.e., *syntax*) that is based on *semantics*
- Note: brains process *syntax* (of some sort)

Models ⊨

- Logicians typically think in terms of *models*, which are formally structured worlds with respect to which truth can be evaluated
- We say *m is a model of* a sentence α if α is true in *m*
- *M(α)* is the set of all models of α
- Then *KB* ⊨ α if and only if *M(KB)* $\subseteq$ *M(α)*
- E.g. *KB* = FCZ won and GCZ won
  α = FCZ won



Inference

- $KB \vdash_i \alpha$ = sentence α can be derived from *KB* by procedure *i*
  - Consequences of *KB* are a haystack; α is a needle.

- Entailment = needle in haystack; inference = finding it

- ***Soundness***: *i* is sound if
  whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
- ***Completeness***: *i* is complete if
  whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
    - Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.
      That is, the procedure will answer any question whose answer follows from what is known by the *KB*.

Summary

- We can apply *inference* to a *knowledge base* to derive new information and make decisions
- Basic concepts of logic:
  - *syntax*: formal structure of *sentences*
  - *semantics*: *truth* of sentences wrt *models*
  - *entailment*: necessary truth of one sentence given another
  - *inference*: deriving sentences from other sentences
  - *soundess*: derivations produce only entailed sentences
  - *completeness*: derivations can produce all entailed sentences

Overview

- Logic in general
- First Order Logic (FOL)
  - Why FOL?
  - Syntax and semantics of FOL
  - Fun with sentences
- Reasoning in FOL
- Knowledge Representation (KR)
  - Semantic Networks
  - Knowledge Graphs

First-order logic

First-order logic (like natural language) assumes the world contains

- ***Objects***: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries …
- ***Relations***: red, round, bogus, prime, multistoried, …, brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, …
- ***Functions***: father of, best friend, third inning of, one more than, beginning of …

Logics in general

Language	Ontological Commitment	EpistemologicalCommitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief$\in [0, 1]$
Fuzzy logic	facts + degree of truth $\in [0, 1]$	known interval value

Syntax of FOL: Basic elements

- Constants *KingJohn, 2, UZH, …*
- Predicates *Brother, >, …*
- Functions *Sqrt, LeftLegOf, …*
- Variables *x, y, a, b, …*
- Connectives $\wedge \vee \neg \Rightarrow \Leftrightarrow$
- Equality $=$
- Quantifiers $\forall \exists$

FOL sentences

- Atomic
  - Atomic sentence = *predicate*($term_1$, …,$term_n$)
    or $term_1 = term_2$
  - Term = *function*($term_1$, …,$term_n$)
    or *constant* or *variable*
  - E.g., *Brother(KingJohn, RichardTheLionheart)*
  - *>(Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))*

- Complex
  - Complex sentences are made from atomic sentences using connectives
    $\neg S$ $S_1 \wedge S_2$ $S_1 \vee S_2$ $S_1 \Rightarrow S_2$ $S_1 \Leftrightarrow S_2$
  - E.g. *Sibling(KingJohn, Richard)* $\Rightarrow$ *Sibling(Richard, KingJohn)*
    $>(1,2) \vee \geq(1,2)$
    $>(1,2) \wedge \neg >(1,2)$

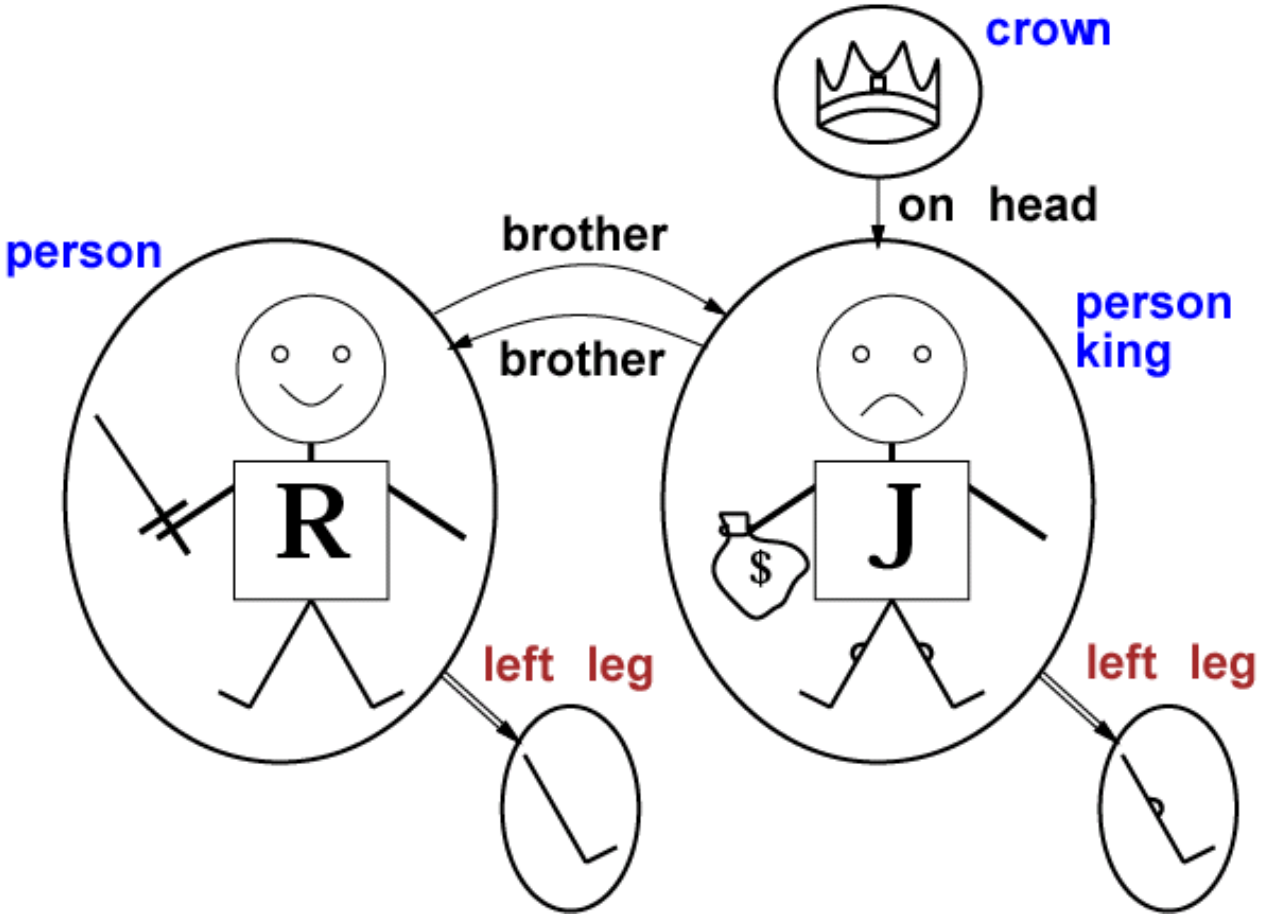
Truth in first-order logic

- Sentences are true with respect to a *model* and an *interpretation*
- Model contains ≥ 1 objects (*domain elements*) and relations among them
- Interpretation specifies referents for
  - *constant symbols* → objects
  - *predicate symbols* → relations
  - *function symbols* → functional relations
- An atomic sentence *predicate*($term_1$, …,$term_n$) is true iff the *objects* referred to by $term_1$, …,$term_n$ are in the *relation* referred to by *predicate*

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

Models for FOL: Example



Models for FOL: Lots!

We ***can*** enumerate the models for a given KB vocabulary:

For each number of domain elements n from 1 to ∞

For each k-ary predicate P_k in the vocabulary

For each possible k-ary relation on n objects

For each constant symbol C in the vocabulary

For each choice of referent for C from n objects …

Computing entailment by enumerating models is not going to be easy!

Fun with sentences

- Brothers are siblings
  - $\forall x,y\ Brother(x,y) \Rightarrow Sibling(x,y)$
- “Sibling”' is symmetric
  - $\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x)$
- One's mother is one's female parent
  - $\forall x,y\ Mother(x,y) \Leftrightarrow (Female(x) \wedge Parent(x,y))$
- A first cousin is a child of a parent's sibling
  - $\forall x,y\ FirstCousin(x,y) \Leftrightarrow$
    $\exists p,ps\ Parent(p,x) \wedge Sibling(ps,p) \wedge Parent(ps,y)$

Interacting with FOL KBs

- Suppose a "wumpus-world" agent is using a FOL KB and perceives a smell and a breeze (but no glitter) at *t=5*:
  - ***Tell****(KB,Percept([Smell,Breeze,None],5))*
  - ***Ask****(KB, ∃a Action(a,5))*
    I.e., does the KB entail any particular actions at *t=5*?
  - Answer: *Yes, {a/Shoot}* ← *substitution* (binding list)

- Given a sentence *S* and a substitution σ,
  - *S*σ denotes the result of plugging σ into *S*; e.g.,
    - *S = Smarter(x,y)*
    - σ = *{x/Hillary, y/Bill}*
    - *S*σ = Smarter(Hillary, Bill)
  - ***Ask****(KB,S)* returns some/all σ such that *KB* |= σ

Keeping track of change

- Facts hold in *situations*, rather than eternally
  E.g., *Holding(Gold,Now)* rather than just *Holding(Gold)*
- *Situation calculus* 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* in *s*

Describing actions I

- "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
  $\forall s\ HaveArrow(s) \Rightarrow HaveArrow(Result(Grab,s))$
- *Frame problem*: find an elegant way to handle non-change
  - Representation – avoid frame axioms
  - 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 II

- *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:

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

Making plans

- 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 ***Ask***$(KB,\exists p,$ $Holding(Gold,PlanResult(p,S_0)))$ has the solution {p/[*Forward,Grab*]}
- Definition of *PlanResult* in terms of *Result*:
  - $\forall s\ PlanResult([],s) = s$
  - $\forall a,p,s\ PlanResult([a|p],s) = PlanResult(p,Result(a,s))$
- *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

Overview

- Logic in general
- First Order Logic (FOL)
  - Why FOL?
  - Syntax and semantics of FOL
  - Fun with sentences
- Reasoning in FOL
- Knowledge Representation (KR)
  - Semantic Networks
  - Knowledge Graphs

A brief history of reasoning

450 b.c.	Stoics	propositional logic, inference (maybe)
322 b.c.	Aristotle	“syllogisms” (inference rules), quantifiers
1565	Cardano	probability theory (propositional logic + uncertainty)
1847	Boole	propositional logic (again)
1879	Frege	first-order logic
1922	Wittgenstein	proof by truth tables
1930	Gödel	∃ complete algorithm for FOL
1930	Herbrand	complete algorithm for FOL (reduce to propositional)
1931	Gödel	¬∃ complete algorithm for arithmetic
1960	Davis/Putnam	“practical” algorithm for propositional logic
1965	Robinson	“practical” algorithm for FOL –resolution

Universal instantiation (UI)

- Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall\, v \quad \alpha}{\text{Subst}(v/g, \alpha)}$$

- for any variable *v* and ground term *g*

- E.g., $\forall\, x$: $King(x) \wedge Greedy(x) \Rightarrow Evil(x)$, yields

  - $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  - $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
  - $King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John)$
  - …

Existential instantiation (EI)

- For any sentence α, variable *v*, and constant symbol *k* *that does not appear elsewhere in the knowledge base*:

$$\frac{\exists\, v \quad \alpha}{\text{Subst}(v/k, \alpha)}$$

- E.g., $\exists\, x : Crown(x) \wedge OnHead(x,John)$ yields
  - $Crown(C_1) \wedge OnHead(C_1,John)$
- provided C_1 is a new constant symbol, called a *Skolem constant*

- Another example: from $\exists\, x : d(x^y)/dy = x^y$ we obtain
  - $d(x^y)/dy = e^y$
- provided *e* is a new constant symbol

Existential instantiation contd.

- UI can be applied several times to *add* new sentences;
  the new KB is logically equivalent to the old
- EI can be applied once to *replace* the existential sentence;
  the new KB is *not* equivalent to the old,
  but is satisfiable iff the old KB was satisfiable

Reduction to propositional inference

- Suppose the KB contains just the following:
  - $\forall\ x : King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  - King(John)
  - Greedy(John)
  - Brother(Richard,John)
- Instantiating the universal sentence in *all possible* ways, we have
  - $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  - $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
  - King(John)
  - Greedy(John)
  - Brother(Richard,John)
- The new KB is *propositionalized*: proposition symbols are: *King(John), Greedy(John), Evil(John), King(Richard), etc.*

Problems with propositionalization

- Propositionalization seems to generate lots of irrelevant sentences
- E.g., from
  - $\forall\ x : King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  - King(John)
  - $\forall\ y : Greedy(y)$
  - Brother(Richard,John)
- it seems obvious that *Evil(John)*, but propositionalization produces lots of facts such as *Greedy(Richard)* that are irrelevant
- With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations!

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

- θ = *{x/John,y/John}* works

- Unify(α, β) = θ if $\alpha\,\theta = \beta\,\theta$

p	*q*	θ
Knows(John,x)	*Knows(John,Jane)*	
Knows(John,x)	*Knows(y,OJ)*	
Knows(John,x)	*Knows(y,Mother(y))*	
Knows(John,x)	*Knows(x,OJ)*	

Example knowledge base

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
- Prove that Col. West is a criminal

Example knowledge base contd.

… is a crime for an American to sell weapons to hostile nations:

Nono … has some missiles, i.e., $\exists\, x : \mathit{Owns}(\mathit{Nono},x) \wedge \mathit{Missile}(x)$

… all of its missiles were sold to it by Colonel West

Missiles are weapons:

An enemy of America counts as “hostile”:

West, who is American …

The country Nono, an enemy of America…

Example knowledge base contd.

… is a crime for an American to sell weapons to hostile nations:

$\forall x \exists y \exists z : American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \Rightarrow Criminal(x)$

Nono … has some missiles, i.e., $\exists x : Owns(Nono,x) \land Missile(x)$

$Owns(Nono,M_1) \land Missile(M_1)$

… all of its missiles were sold to it by Colonel West

$\forall x: Missile(x) \land Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$

Missiles are weapons:

$\forall x: Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as “hostile”:

$\forall x: Enemy(x,America) \Rightarrow Hostile(x)$

West, who is American …

$American(West)$

The country Nono, an enemy of America…

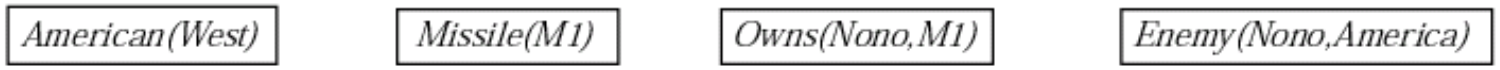
$Enemy(Nono,America)$

Forward chaining algorithm

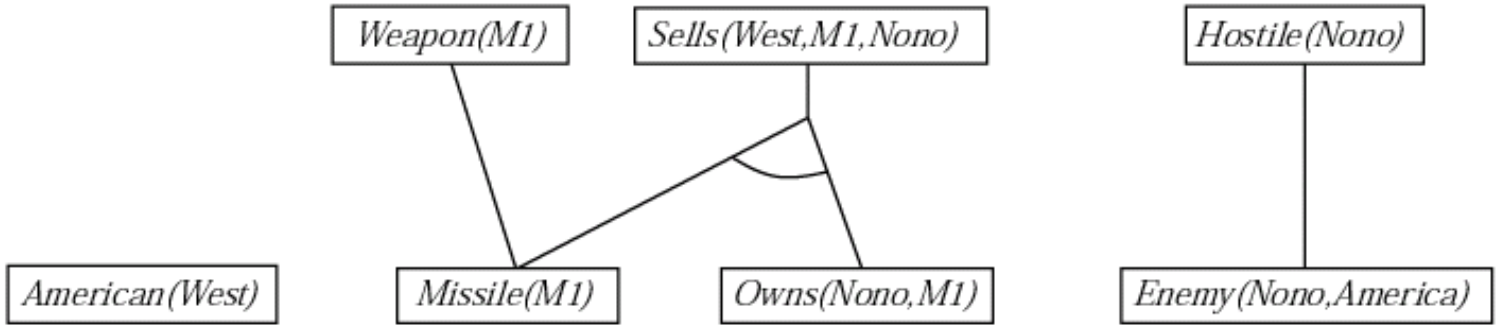
function FOL-FC-ASK(KB, α) **returns** a substitution or *false*

repeat until *new* is empty
  new $\leftarrow \{\}$
  for each sentence r **in** KB **do**
    $(p_1 \wedge \ldots \wedge p_n \implies q) \leftarrow$ STANDARDIZE-APART(r)
    for each θ such that $(p_1 \wedge \ldots \wedge p_n)\theta = (p'_1 \wedge \ldots \wedge p'_n)\theta$
        for some $p'_1, \ldots, p'_n$ in KB
      $q' \leftarrow$ SUBST(θ, q)
      if q' is not a renaming of a sentence already in KB or *new* **then do**
        add q' to *new*
        $\phi \leftarrow$ UNIFY(q', α)
        if ϕ is not *fail* **then return** ϕ
  add *new* to KB
return *false*

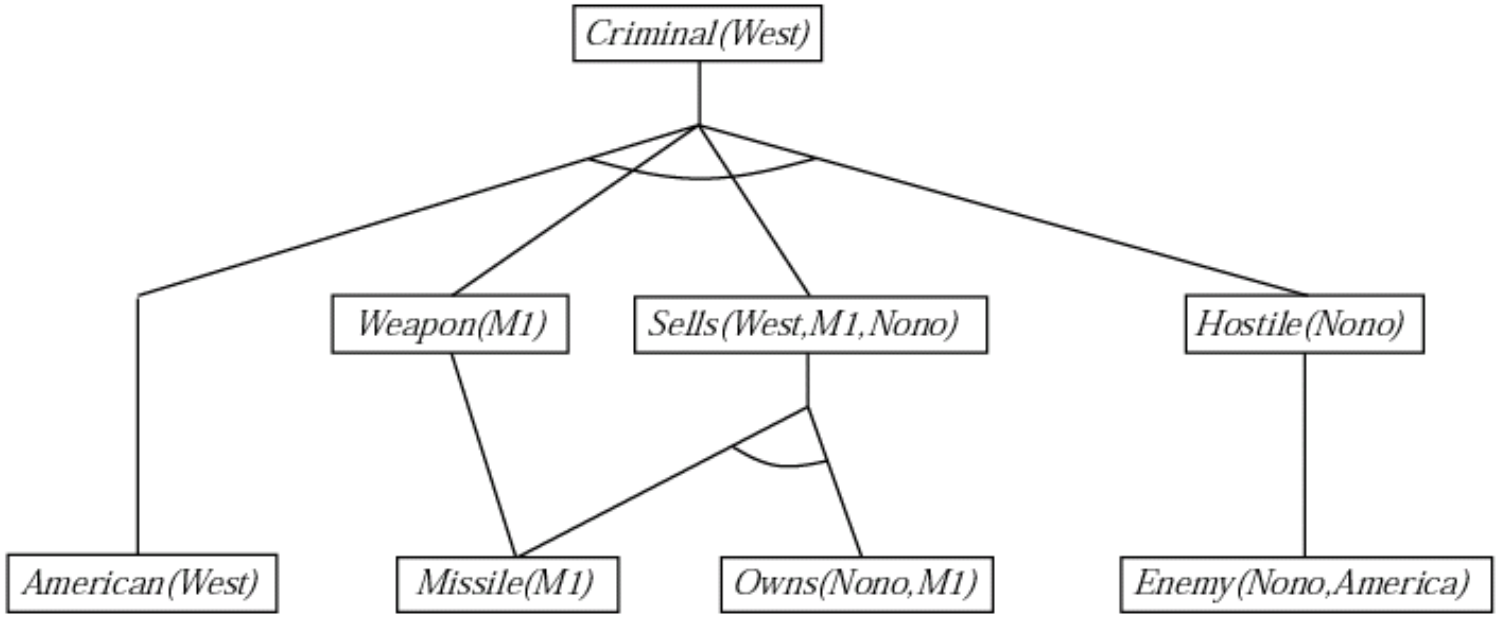
Forward chaining proof



Forward chaining proof



Forward chaining proof



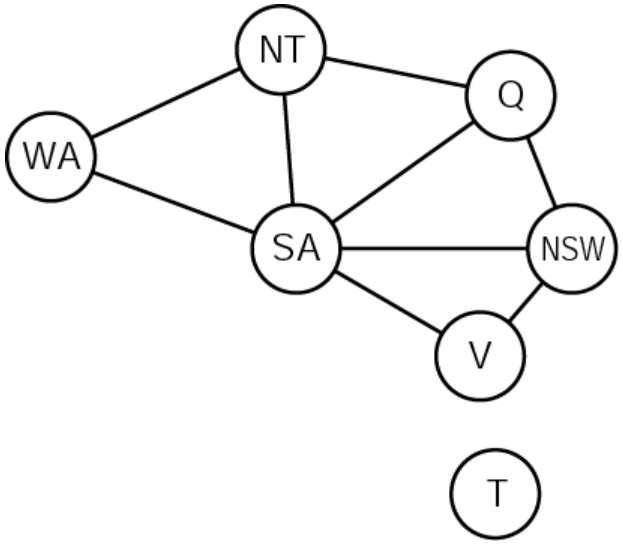
Properties of forward chaining

- Sound and complete for first-order definite clauses
- *Datalog* = first-order definite clauses + *no functions*
  (e.g., crime KB)
- FC terminates for Datalog in poly iterations: at most $p \cdot n^k$ literals
- May not terminate in general if α is not entailed
- This is unavoidable: entailment with definite clauses is semidecidable

Efficiency of forward chaining

- Simple observation: no need to match a rule on iteration *k*
- if a premise wasn't added on iteration *k-1*
  $\Rightarrow$ match each rule whose premise contains a newly added literal
- Matching itself can be expensive
- *Database indexing* allows *O(1)* retrieval of known facts
  - e.g., query Missile(x) retrieves Missile(M_1)
- Matching conjunctive premises against known facts is NP-hard
- Forward chaining is widely used in *deductive databases*

Hard matching example



- *Diff(wa,nt)* $\wedge$ *Diff(wa,sa)* $\wedge$ *Diff(nt,q)* $\wedge$ *Diff(nt,sa)* $\wedge$ *Diff(q,nsw)* $\wedge$ *Diff(q,sa)* $\wedge$ *Diff(nsw,v)* $\wedge$ *Diff(nsw,sa)* $\wedge$ *Diff(v,sa)* $\Rightarrow$ *Colorable()*

- *Diff(Red,Blue)* *Diff(Red,Green)*
- *Diff(Green,Red)* *Diff(Green,Blue)*
- *Diff(Blue,Red)* *Diff(Blue,Green)*

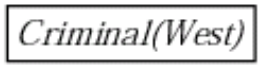
- *Colorable()* is inferred iff the CSP has a solution
- CSPs include 3SAT as a special case, hence matching is NP-hard

Backward chaining algorithm

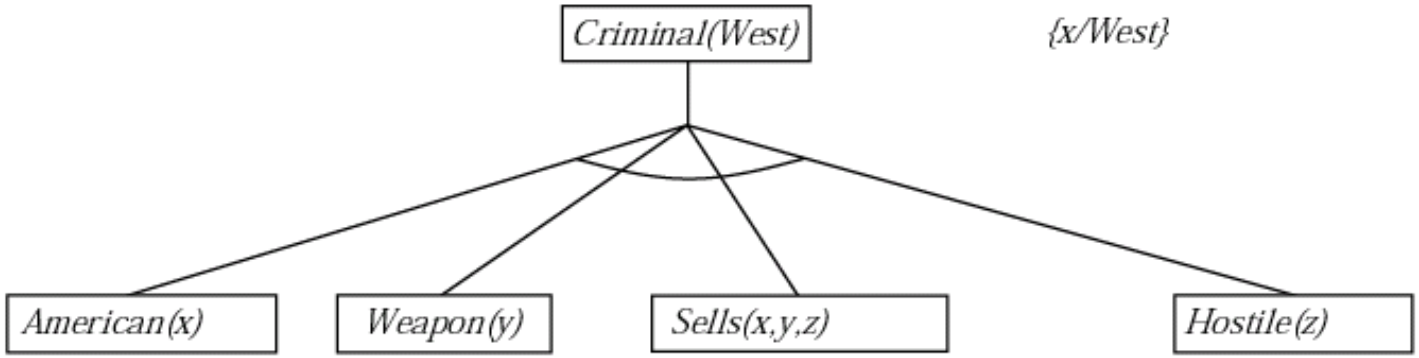
function FOL-BC-Ask(KB, goals, θ) **returns** a set of substitutions
  inputs: KB, a knowledge base
    goals, a list of conjuncts forming a query
    θ, the current substitution, initially the empty substitution $\{\}$
  local variables: *ans*, a set of substitutions, initially empty

  if *goals* is empty **then return** $\{\theta\}$
  $q' \leftarrow$ Subst(θ, First(*goals*))
  for each r **in** KB where Standardize-Apart(r) $= (p_1 \wedge \ldots \wedge p_n \Rightarrow q)$
    and $\theta' \leftarrow$ Unify(q, q') succeeds
    ans $\leftarrow$ FOL-BC-Ask(KB, $[p_1, \ldots, p_n | \text{Rest}(goals)]$, Compose($\theta', \theta$)) $\cup$ *ans*
  return *ans*

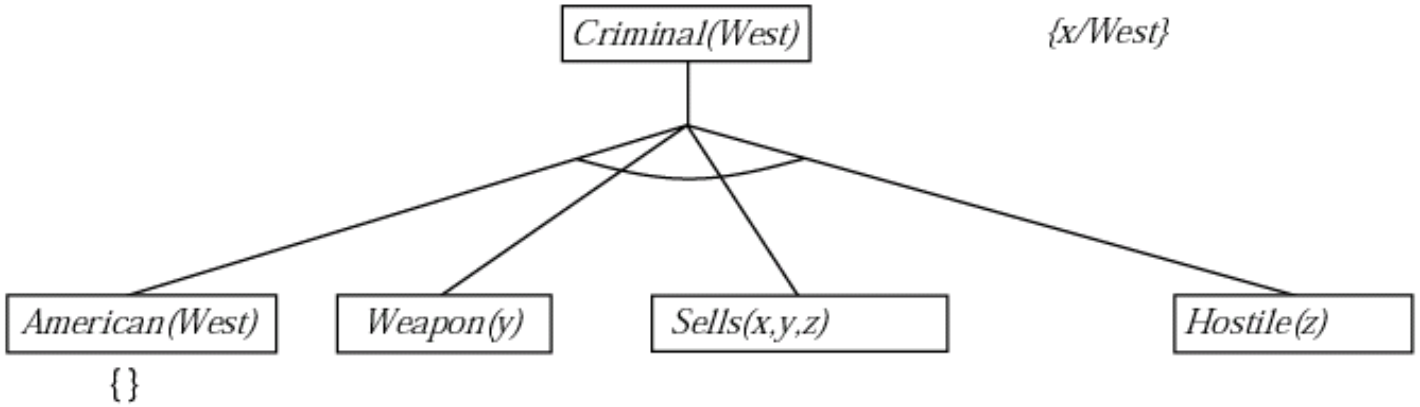
Backward chaining example



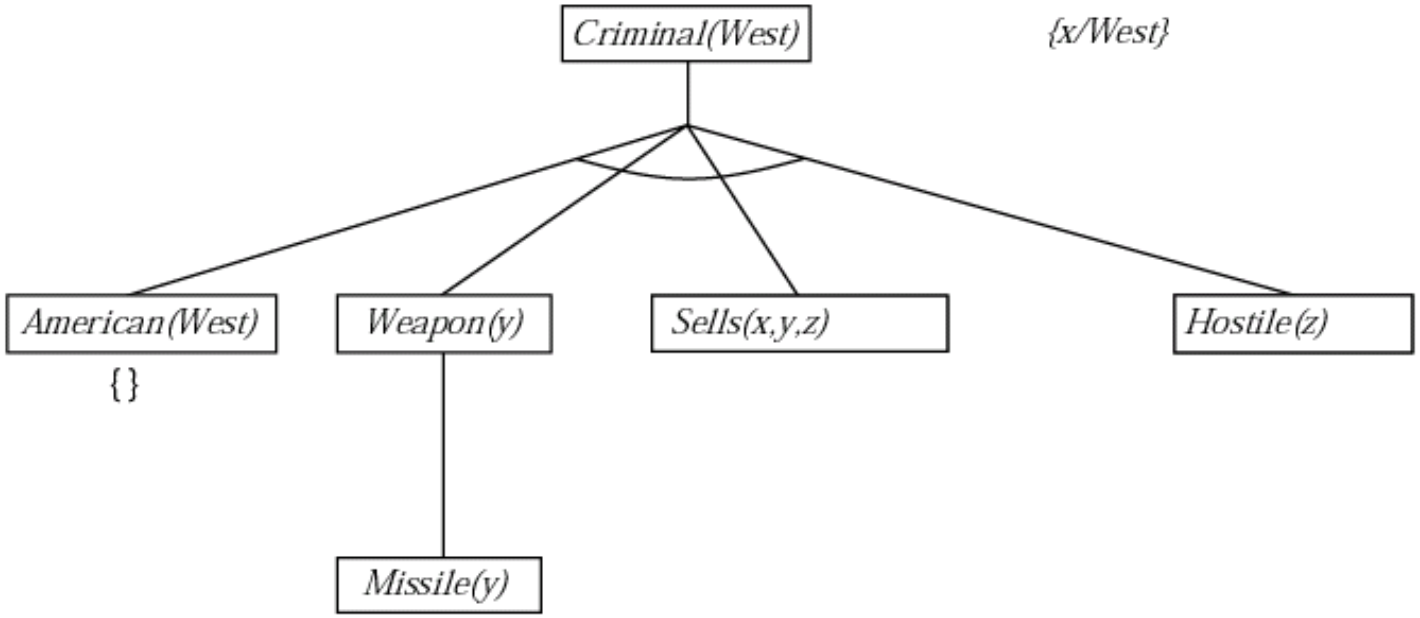
Backward chaining example



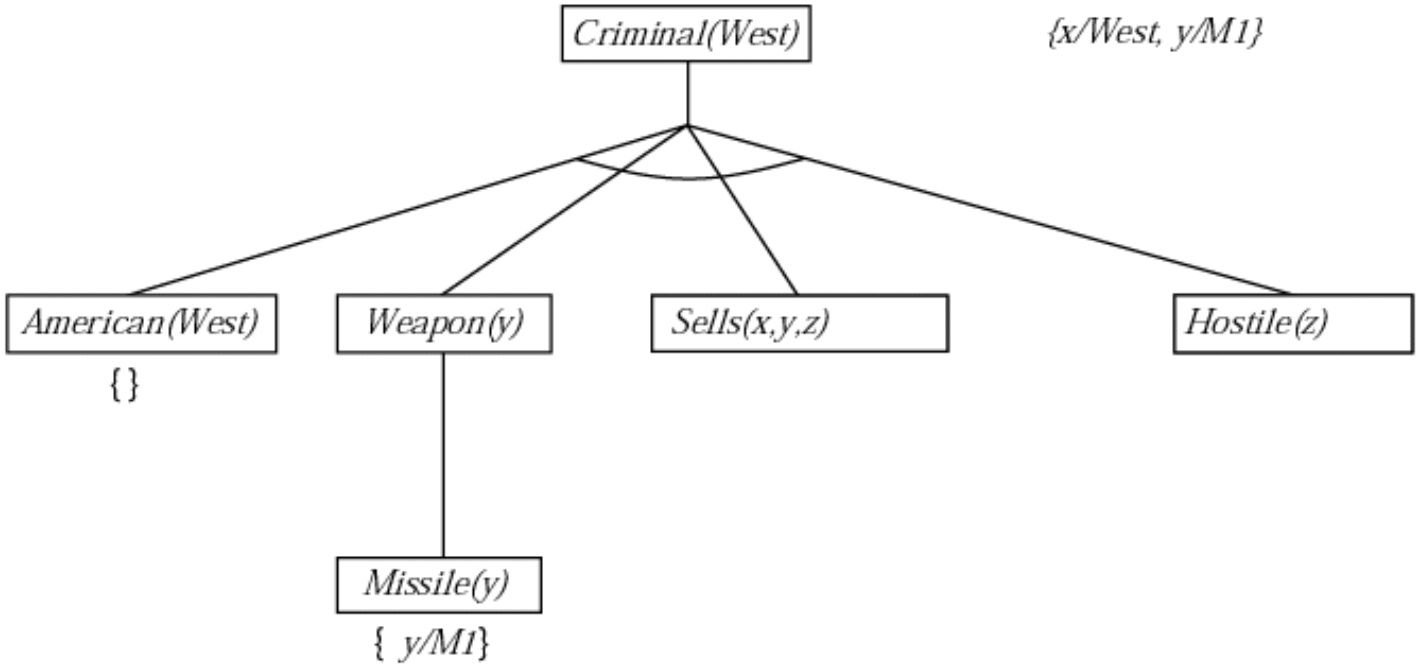
Backward chaining example



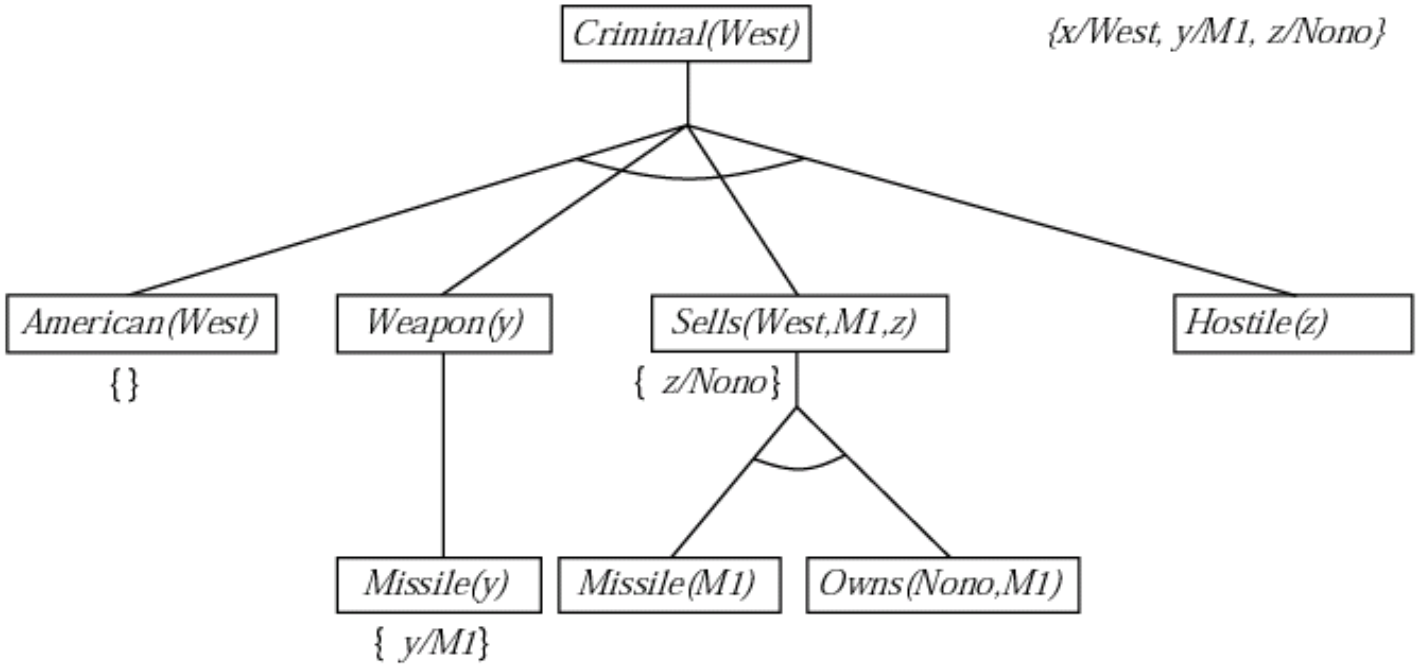
Backward chaining example



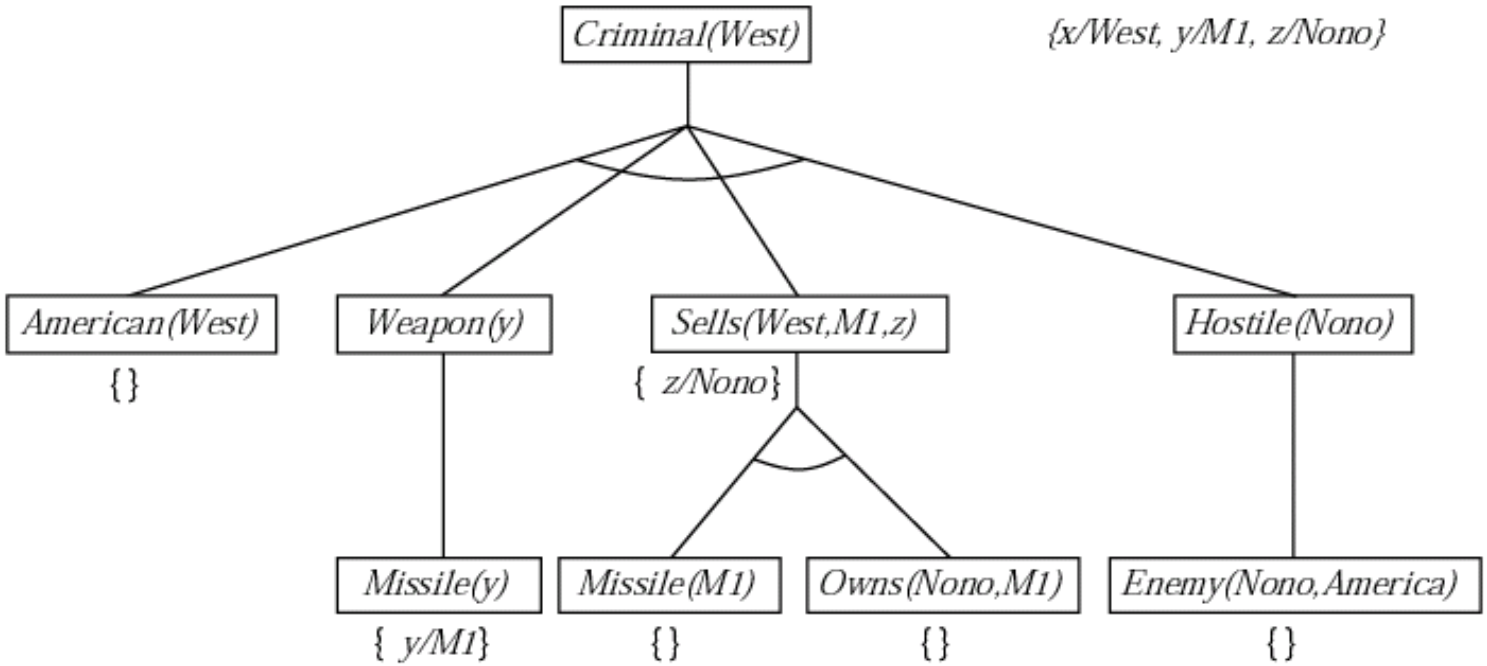
Backward chaining example



Backward chaining example



Backward chaining example



Properties of backward chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops
  $\Rightarrow$ fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)
  $\Rightarrow$ fix using caching of previous results (extra space!)
- Widely used (without improvements!) for *logic programming*

Forward vs. backward chaining

- FC is *data-driven*, cf. automatic, unconscious processing,
  - e.g., object recognition, routine decisions

  May do lots of work that is irrelevant to the goal

- BC is *goal-driven*, appropriate for problem-solving,
  - e.g., Where are my keys? How do I get into a PhD program?

  Complexity of BC can be ***much less*** than linear in size of KB

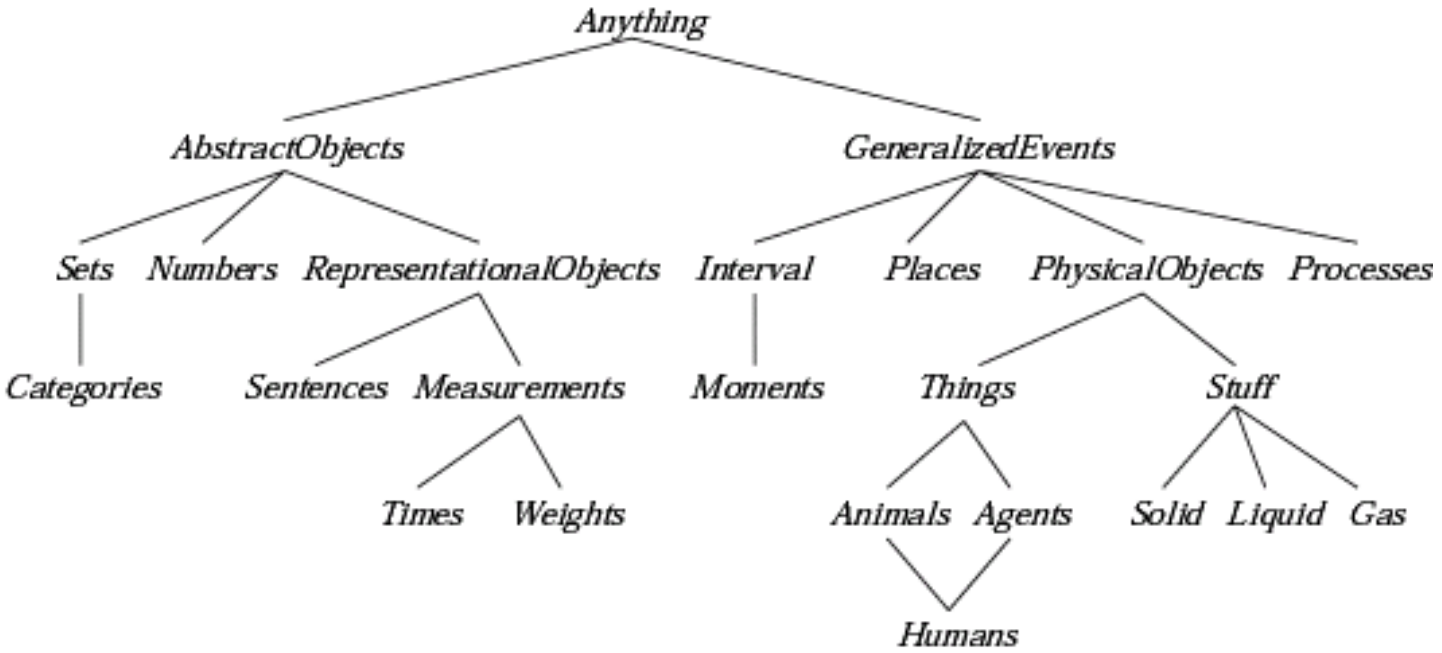
Overview

- Logic in general
- First Order Logic (FOL)
  - Why FOL?
  - Syntax and semantics of FOL
  - Fun with sentences
- Reasoning in FOL
- Knowledge Representation (KR)
  - Semantic Networks
  - Knowledge Graphs

Knowledge Engineering

- Identify the task
- Assemble the relevant knowledge
- Decide on a vocabulary of
  - Predicates
  - Functions
  - Constants
- Encode general knowledge about the domain
- Encode a description of the specific problem instance
- Pose queries to inference procedure and get answers

Ontological Engineering



- Like knowledge engineering, but on a larger scale
- Uses an upper ontology describing the major objects in the considered universe
  - Should be applicable in many (>1) situations
  - Multiple ontologies may have to be integarted

Categories and Objects

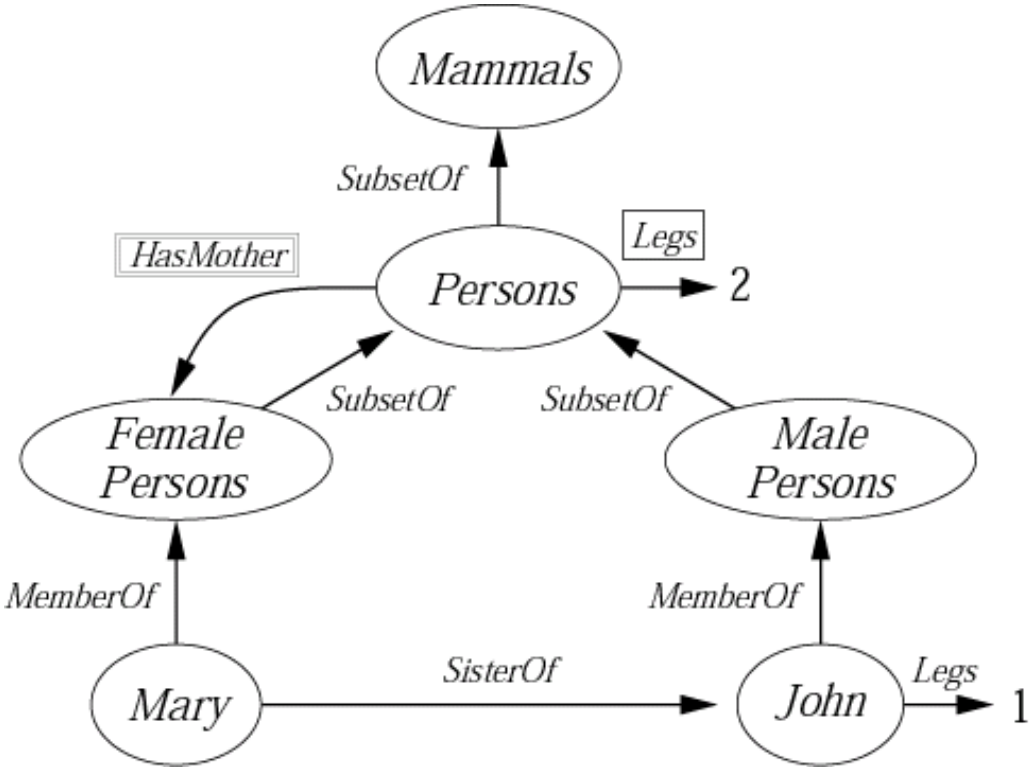
- One can divide objects into categories
  - Basketball(x), x is a Basketball, x $\in$ Basketballs
  - Reified: Member(x, Basketballs)
- Subclass relationships ➔ isa ➔ inheritance
  - Basketball isa Ball, Basketballs $\subset$ Balls
  - In FOL:
- Categories may have properties
  - x $\in$ Basketballs $\Rightarrow$ Round(x)
- Members can be recognized through properties:
  - Orange(x) $\wedge$ Round(x) $\wedge$ Diameter(x) = 9.5” $\wedge$ x $\in$ Balls $\Rightarrow$ x $\in$ Basketballs

Categories and Objects II

- Categorical descriptors:
  - Disjoint({animals,vegetables})
  - Exhaustive Decomposition({US citizens, Canadians, Mexicans}, {NorthAmericans})
  - Partition({males, females}, Animals)
- Composition
  - PartOf(Zurich, Switzerland)
  - PartOf(Switzerland, Europe)
  - Reflexive: PartOf(x,x)
  - Transitive:
    $\text{PartOf}(x,y) \wedge \text{PartOf}(y,z) \Rightarrow \text{PartOf}(x,z)$
- I am skipping: Actions, Situations, Events, Belief

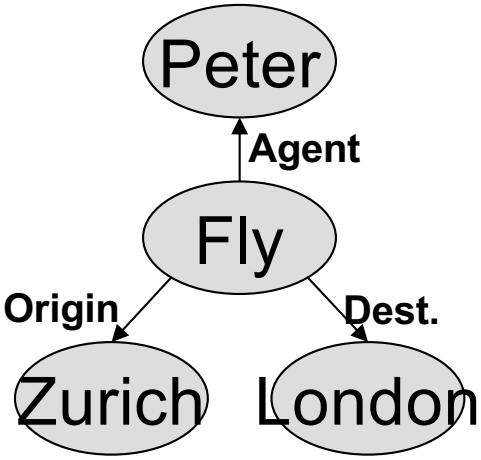
Semantic Networks (Pierce 1909)

- Categories in bubbles
- Relationships as labeled arrows
- ***But***: *HasMother* is a relationship between ***a*** Person and his Mother (and not between the concept Person and the concept FemalePerson)
  ➔ *Double Boxed Arrow*
- Legs is a property of a concept (inherited)



Semantic Networks cont.

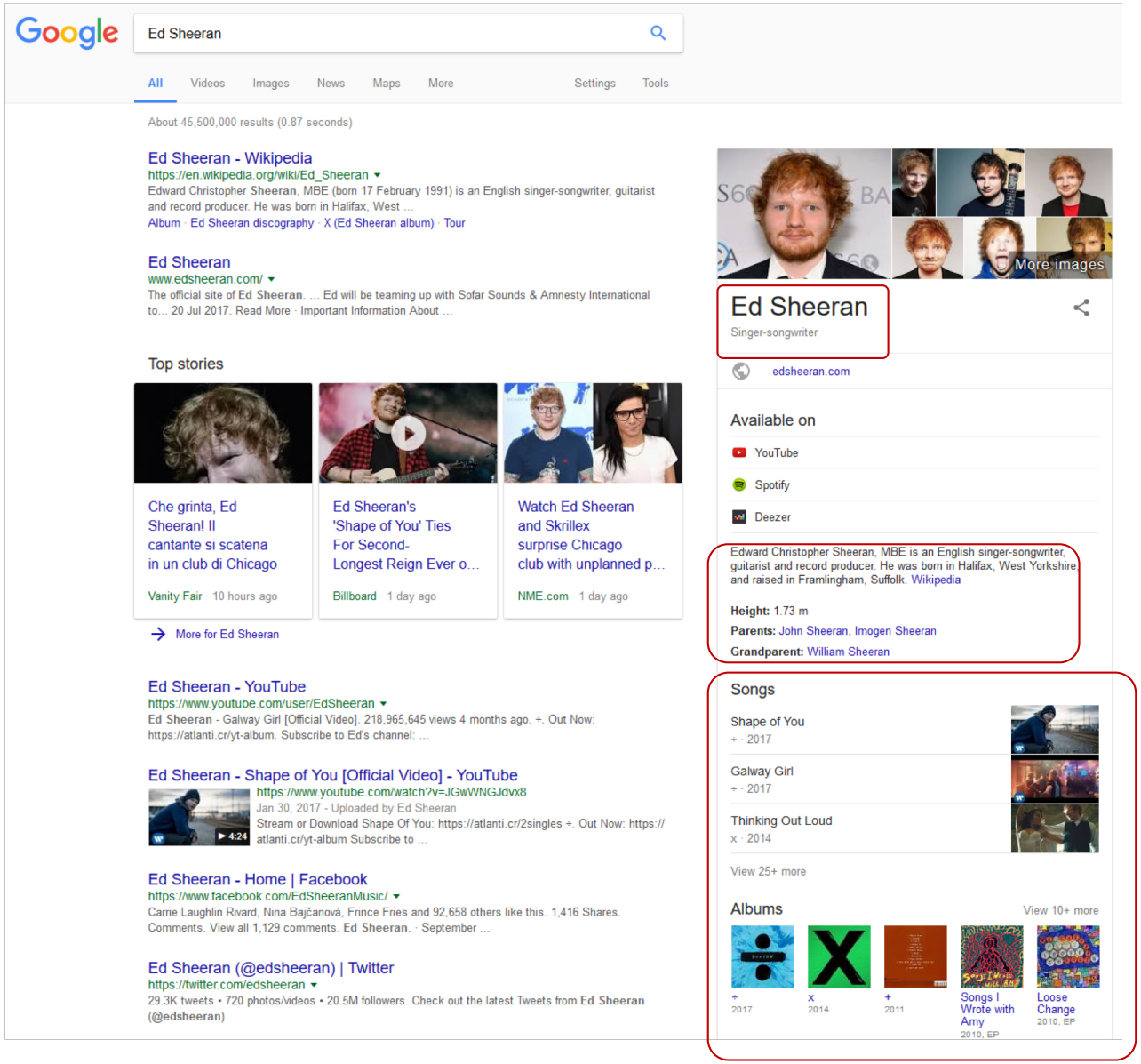
- Limited to binary relationships
  - Fly(Peter, Zurich, London)
  - Solution: ***Reification***
  - ➔ SN Looses simplicity
- Default Values
  - John has only one leg
  - $\forall x: x \in Persons \wedge x \neq John \Rightarrow Legs(x, 2)$
  - This overrides inheritance
- Multiple inheritance
  - What do we do with inheritance conflicts?



Knowledge graphs

- Big web companies are building their own knowledge graphs (KGs), e.g.
  - Google KG
    https://developers.google.com/knowledge-graph/
  - Facebook Open Graph
    https://developers.facebook.com/docs/sharing/opengraph
  - AirBnb KG
    https://medium.com/airbnb-engineering/scaling-knowledge-access-and-retrieval-at-airbnb-665b6ba21e95
  - Uber food KG
    https://ubereng.wpengine.com/uber-eats-query-understanding/

The Google Knowledge Graph



The Google Knowledge Graph