CS 561/571: Artificial Intelligence

First Order Logic

Introduction

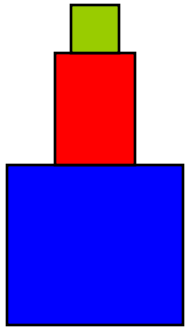
- Knowledge Representation means:
  - Capturing human knowledge
  - In a form computer can reason about
- Why?
  - Model human cognition
  - Add power to search-based methods
- Actually a component of all software development

Knowledge Representation (KR)

Given the world

- Express the general facts or beliefs using a language
- Determine what else we should (not) believe

Example



- Given:
  - "The red block is above the blue block"
  - "The green block is above the red block"
- Infer:
  - "The green block is above the blue block"
  - "The blocks form a tower"

Characteristics of a good KR:

It should

- Be able to represent the knowledge important to the problem
- Reflect the structure of knowledge in the domain
  - Otherwise our development is a constant process of distorting things to make them fit.
- Capture knowledge at the appropriate level of granularity
- Support incremental, iterative development

It should *not*

- Be too difficult to reason about
- Require that more knowledge be represented than is needed to solve the problem

A KR language needs to be

- Expressive (*should be able to describe common sense knowledge*)
- Unambiguous (*should be able to express the meaning of any instance uniquely*)
- Flexible (*easy to represent different facts and beliefs*)

The inference procedures need to be

- Correct (sound) (*all probable statements are true*)
- Complete (*any entailed sentence can be proved*)
- Efficient (*requires less time to derive*)

Kinds of Knowledge

Things we need to talk about and reason about; what do we know?

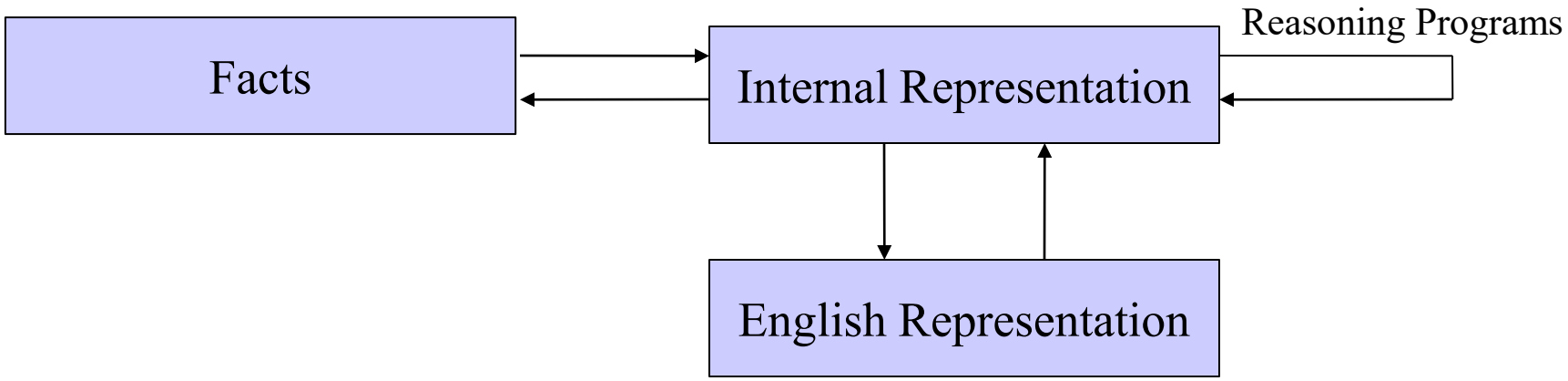
- Objects
  - Descriptions
  - Classifications
- Events
  - Time sequence
  - Cause and effect
- Relationships
  - Among objects
  - Between objects and events
- Meta-knowledge

Distinguish between knowledge and its representation

Mappings are not one-to-one

Never get it complete or exactly right

Representation Mappings



- Knowledge Level
- Symbol Level
- Mappings are not one-to-one
- Never get it complete or exactly right

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!

Some Typical Kinds of KR

- Logic and predicate calculus
- Rules: production systems
- Description logics, semantic nets, frames
- Scripts
- Ontologies

Pros and cons of propositional logic

☺Propositional logic is declarative and not procedural (domain-specific) like programming languages

☺Procedural

- ☺ lacks a general mechanism for deriving facts
- ☺each update to a data structure is done by a domain-specific procedures whose details are derived by the programmer from his or her own knowledge of the domain

☺Declarative

- ☺Knowledge and inference are separate
- ☺Inference is entirely domain-independent

☺ Propositional logic allows partial/disjunctive/negated information

- (unlike most data structures and databases)

Pros and cons of propositional logic

☺ Propositional logic is compositional:

- meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$

☺ Meaning in propositional logic is context-independent

- (unlike *natural language, where meaning depends on context*)

Pros and cons of propositional logic

- Propositional logic has very limited expressive power
  - (unlike natural language)
  - E.g., cannot say "person-x" is mortal "
    - except by writing one sentence for each person

- propositional logic assumes the world contains facts
  - Either holds true or false

Example

- Consider the problem of representing the following information:
  - Every person is mortal.
  - Confucius is a person.
  - Confucius is mortal.
- How can these sentences be represented so that we can infer the third sentence from the first two?

Example cont.

- In PL, create propositional symbols to stand for all or part of each sentence.
  - P = “person”; Q = “mortal”; R = “Confucius”
- so the above 3 sentences are represented as:
  - P => Q; R => P; R => Q
- Although the third sentence is entailed by the first two, we needed an explicit symbol, R, to represent an individual, Confucius, who is a member of the classes “person” and “mortal.”
- To represent other individuals we must introduce separate symbols for each one, with means for representing the fact that all individuals who are “people” are also "mortal.”

Proofs in propositional calculus

If it is sunny today, then the sun shines on the screen. If the sun shines on the screen, the blinds are brought down. The blinds are not down.

Is it sunny today?

P: It is sunny today.

Q: The sun shines on the screen.

R: The blinds are down.

Premises: $P \rightarrow Q$, $Q \rightarrow R$, $\neg R$

Question: P

Prove using a truth table

Variables			Premises			Trial Conclusions	
P	Q	R	$P \rightarrow Q$	$Q \rightarrow R$	$\neg R$	P	$\neg P$
T	T	T	T	T	F	T	F
T	T	F	T	F	T	T	F
T	F	T	F	T	F	T	F
T	F	F	F	T	T	T	F
F	T	T	T	T	F	F	T
F	T	F	T	F	T	F	T
F	F	T	T	T	F	F	T
F	F	F	T	T	T	F	T

Propositional calculus is cumbersome

If it is sunny today, then the sun shines on the screen. If the sun shines on the screen, the blinds are brought down. The blinds are not down.

Is it sunny today?

- - -

If it is sunny on a particular day, then the sun shines on the screen. If the sun shines on the screen on a particular day, the blinds are brought down. The blinds are not down today.

Is it sunny today?

Represent in predicate calculus

If it is sunny on a particular day, then the sun shines on the screen [on that day]. If the sun shines on the screen on a particular day, the blinds are down [on that day].
The blinds are not down today.

Is it sunny today?

Premises:

$\forall$D sunny (D)$\rightarrow$ screen-shines (D)

$\forall$D screen-shines(D) $\rightarrow$ blinds-down(D)

$\neg$ blinds-down (today)

Question: sunny(today)

First-order logic

- *First-order logic* (like natural language) assumes the world contains
  - Objects: people, houses, numbers, colors, baseball games, wars, …
  - Relations: red, round, prime, brother of, bigger than, part of, comes between, …
  - Functions: father of, best friend, one more than, plus, …

FOPL

Declarative, Context-independent and Unambiguous compositional semantics of PL + representational ideas of NLP

Symbols and terms

1. *Truth symbols:* true and false (these are reserved symbols)
2. *Constant symbols:* expressions having the first character lowercase

   E.g., today, fisher
3. *Variable symbols*: symbol expressions beginning with an uppercase character

   E.g., X, Y, Z, Building
4. *Function symbols:* symbol expressions having the first character lowercase

Arity: number of elements in the domain

E.g., mother-of (bill); maximum-of (7,8)

Symbols and terms (cont'd)

function expression: consists of a function constant of arity n, followed by n terms, t_1 ,t_2 ,…, t_n, enclosed in parentheses and separated by commas.

E.g., mother-of(mother-of(joe))

maximum(maximum(7, 18), add_one(18))

term: either a constant, variable, or function expression.

E.g. color_of(house_of(neighbor(joe)))

house_of(X)

Predicates and atomic sentences

Predicate symbols are symbols beginning with a lowercase letter. Predicates are special functions with true/false as the range

Arity: number of arguments

An atomic sentence is a predicate constant of arity n, followed by n terms, t_1 ,t_2 ,. . . , t_n, enclosed in parentheses and separated by commas

The truth values, true and false, are also atomic sentences.

Examples

greater_than(2, 3)

Predicate symbol

term (constant)

mother_of(joe, susan)

mother_of(sister_of(joe), susan)

Predicate Calculus Sentences

Every atomic sentence is a sentence

1. If s is a sentence, then so is its negation, $\neg s$.

If s_1 and s_2 are sentences, then so is their

2. Conjunction, $s_1 \wedge s_2$
3. Disjunction, $s_1 \vee s_2$
4. Implication, $s_1 \rightarrow s_2$
5. Equivalence, $s_1 \equiv s_2$

Predicate calculus sentences (cont'd)

If X is a variable and s is a sentence, then so are

6. $\forall X\ s$
7. $\exists X\ s$

Remember that logic sentences evaluate to true or false, therefore only such objects are atomic sentences. Functions are **not** atomic sentences

Well-formed formula

A well-formed formula (wff) is a sentence containing no "free" variables. i.e. all variables are "bound" by universal or existential quantifiers

- $\forall x$ R(x,y) has x bound as a universally quantified variable, but y is free

Interpretation

Let the domain D be a nonempty set.

An *interpretation* over D is an assignment of the entities of D to each of the constant, variable, predicate, and function symbols of a predicate calculus expression:

1. Each constant is assigned an element of D.
2. Each variable is assigned to a nonempty subset of D (*allowable substitutions*).
3. Each function f of arity m is defined (D^m to D)
4. Each predicate of arity n is defined (D^n to {T,F}).

Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains 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*$(term_1,\ldots,term_n)$ is true iff the objects referred to by $term_1,\ldots,term_n$ are in the relation referred to by the *predicate*

Universal quantification

- ∀<*variables*> <*sentence*>

Everyone at IITP is smart:

∀x At(x, IITP) ⇒ Smart(x)

- Ground term: term with no variables

- ∀x *P* is true in a model *m* iff *P* is true with *x* being each possible object in the model

- Roughly speaking, equivalent to the conjunction of instantiations of *P*

At(KingJohn, IITP) ⇒ Smart(KingJohn)
∧ At(Richard, IITP) ⇒ Smart(Richard)
∧ At(IITP, IITP) ⇒ Smart(IITP)
∧ …

A common mistake to avoid

Typically, $\Rightarrow$ is the main connective with $\forall$

- *Take care*: using $\wedge$ as the main connective with $\forall$:

  $\forall x$ At (x, IITP) $\wedge$ Smart(x)

  means "*Everyone is at IITP and everyone is smart*"

Existential quantification

- ∃<*variables*> <*sentence*>
- Someone at IITP is smart:

  $\exists x$ At(x, IITP) ∧ Smart(x)

- $\exists x$ *P* is true in a model *m* iff *P* is true with *x* being some possible object in the model
- Roughly speaking, equivalent to the disjunction of instantiations of *P*

  At(KingJohn, IITP) ∧ Smart(KingJohn)
  ∨ At(Richard, IITP) ∧ Smart(Richard)
  ∨ At(IITP,IITP) ∧ Smart(IITP)
  ∨ …

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists x \, \text{At}(x, \text{IITP}) \Rightarrow \text{Smart}(x)$$

is true if there is anyone who is not at IITP!

Properties of quantifiers

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$
- $\exists x\ \forall y$ is not the same as $\forall y\ \exists x$
- $\exists x\ \forall y$ Loves(x, y)
  - "There is a person who loves everyone in the world"
- $\forall y\ \exists x$ Loves(x, y)
  - "Everyone in the world is loved by at least one person"
- Quantifier duality: each can be expressed using the other

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

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

Equality

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

- E.g., definition of *Sibling* in terms of *Parent*:

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

First-order predicate calculus

First-order predicate calculus allows quantified variables to refer to *objects in the domain of discourse and not to predicates or functions*.

John likes to eat everything

$\forall X \text{ food}(X) \rightarrow \text{likes (john, X)}$

John likes at least one dish Jane likes

$\exists F \text{ food}(F) \wedge \text{ likes (jane, F)} \wedge \text{likes (john, F)}$

John "does" everything Jane does

$\forall P \; P(\text{Jane}) \rightarrow P(\text{john})$ This is not first-order

Using FOL

The kinship domain:

- Brothers are siblings

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

- One's mother is one's female parent

  $\forall m,c\ Mother(c) = m \Leftrightarrow (Female(m) \wedge Parent(m,c))$

- "Sibling" is symmetric

  $\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x)$

The set domain

- Only sets are the empty set and those made by adjoining something to a set
  - $\forall s\ \mathrm{Set}(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_2\ \mathrm{Set}(s_2) \wedge s = \{x \mid s_2\})$
- Empty set has no elements adjoining to it
  - $\neg\exists x, s\ \{x \mid s\} = \{\}$
- Adjoining an element already in the set has no effect
  - $\forall x, s\ x \in s \Leftrightarrow s = \{x \mid s\}$
- The only members of set are the elements that were adjoined into it
  - $\forall x, s\ x \in s \Leftrightarrow [\exists y, s_2\ (s = \{y \mid s_2\} \wedge (x = y \vee x \in s_2))]$
- A set is a subset of another set
  - $\forall s_1, s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$

The set domain

- Equal sets
  - $\forall s_1, s_2 \; (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
- Object that is in intersection of two sets
  - $\forall x, s_1, s_2 \; x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
- Object that is in the union of two sets
  - $\forall x, s_1, s_2 \; x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

Interacting with KB

- First-order sentence in KB must include
  - Percepts
  - Time

- Agent may be confused about when it saw what!

Interacting with FOL KBs

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

  Tell(KB, Percept ([Smell, Breeze, None, None, None, None],5))
  Ask (KB, ∃a BestAction(a,5))

- Does the KB entail some best action at *t*=5?
  Answer: *Yes*, {*a/Shoot*} ← substitution (binding list)

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

- Ask(KB,S) returns some/all σ such that KB ⊨ σ

Knowledge base for the Wumpus world

- **Perception**
  - $\forall t, s, b$ Percept([s, b, Glitter, None, None],t) $\Rightarrow$ Glitter(t)
- **Reflex**
  - $\forall t$ Glitter(t) $\Rightarrow$ BestAction (Grab, t)

Synchronic vs. Diachronic

- **Synchronic sentences**
  - Relate properties of the world state to the other properties of the same world state
  - Allow reasoning within same world state

- **Diachronic sentences**
  - Relate properties of the different world states
  - Allow reasoning across time

    e.g. *Agent needs to know how to combine information about its previous location with action just taken in order to determine its current location*

Environment and deducing hidden properties

- Adjacency of two squares
  - $\forall x,y,a,b$ *Adjacent*$([x,y],[a,b]) \Leftrightarrow$

$[a, b] \in \{[x+1,y], [x-1,y],[x,y+1],[x,y-1]\}$

- Properties of squares

$\forall s, t$ *At*$(\text{Agent}, s, t) \wedge \text{Breeze}(t) \Rightarrow \text{Breezy}(s)$

At (Agent, s, t): agent is at square s at time t

Infer properties of the square from the properties of its current percept

Deducing hidden properties

Squares are breezy near a pit:

- Diagnostic rule---infer cause from effect

  $\forall s\ \text{Breezy}(s) \Rightarrow [\exists\ r\ \text{Adjacent}(r,s) \wedge \text{Pit}(r)]$

- Causal rule---infer effect from cause

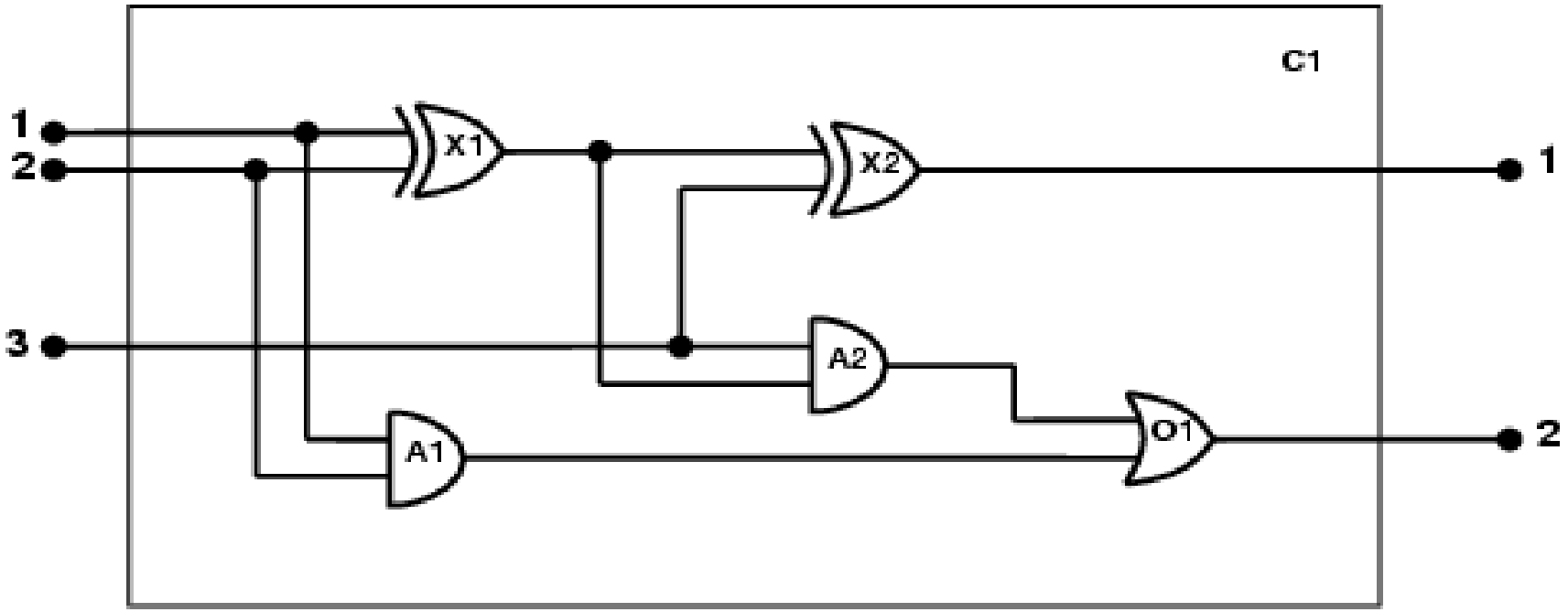
  $\forall r\ \text{Pit}(r) \Rightarrow [\forall s\ \text{Adjacent}(r,s) \Rightarrow \text{Breezy}(s)\ ]$

Knowledge Engineering in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

The electronic circuits domain

One-bit full adder



The electronic circuits domain

1. Identify the task
   - Does the circuit actually add properly? (*circuit verification*)
2. Assemble the relevant knowledge

-Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)

-Irrelevant: size, shape, color, cost of gates

3. Decide on a vocabulary

- Alternatives:

  Type(X_1) = XOR

  Type(X_1, XOR)

  XOR(X_1)

The Electronic Circuits Domain

4. Encode general knowledge of the domain
   - $\forall t_1,t_2\ \text{Connected}(t_1, t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$
   - $\forall t\ \text{Signal}(t) = 1 \vee \text{Signal}(t) = 0$

     $1 \neq 0$
   - $\forall t_1,t_2\ \text{Connected}(t_1, t_2) \Rightarrow \text{Connected}(t_2, t_1)$
   - $\forall g\ \text{Type}(g) = \text{OR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \exists n\ \text{Signal}(\text{In}(n,g)) = 1$
   - $\forall g\ \text{Type}(g) = \text{AND} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 0 \Leftrightarrow \exists n\ \text{Signal}(\text{In}(n,g)) = 0$
   - $\forall g\ \text{Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \text{Signal}(\text{In}(1,g)) \neq \text{Signal}(\text{In}(2,g))$
   - $\forall g\ \text{Type}(g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1,g)) \neq \text{Signal}(\text{In}(1,g))$

The electronic circuits domain

5. Encode the specific problem instance

$\text{Type}(X_1) = \text{XOR}$ $\text{Type}(X_2) = \text{XOR}$

$\text{Type}(A_1) = \text{AND}$ $\text{Type}(A_2) = \text{AND}$

$\text{Type}(O_1) = \text{OR}$

$\text{Connected}(\text{Out}(1,X_1),\text{In}(1,X_2))$

$\text{Connected}(\text{Out}(1,X_1),\text{In}(2,A_2))$

$\text{Connected}(\text{Out}(1,A_2),\text{In}(1,O_1))$

$\text{Connected}(\text{Out}(1,A_1),\text{In}(2,O_1))$

$\text{Connected}(\text{Out}(1,X_2),\text{Out}(1,C_1))$

$\text{Connected}(\text{Out}(1,O_1),\text{Out}(2,C_1))$

$\text{Connected}(\text{In}(1,C_1),\text{In}(1,X_1))$

$\text{Connected}(\text{In}(1,C_1),\text{In}(1,A_1))$

$\text{Connected}(\text{In}(2,C_1),\text{In}(2,X_1))$

$\text{Connected}(\text{In}(2,C_1),\text{In}(2,A_1))$

$\text{Connected}(\text{In}(3,C_1),\text{In}(2,X_2))$

$\text{Connected}(\text{In}(3,C_1),\text{In}(1,A_2))$

The electronic circuits domain

6. Pose queries to the inference procedure

   What are the possible sets of values of all the terminals for the adder circuit?

$\exists i_1,i_2,i_3,o_1,o_2$ Signal(In(1, C_1))= i_1 $\wedge$ Signal(In(2,C_1)) = i_2 $\wedge$ Signal(In(3,C_1)) = i_3 $\wedge$ Signal(Out(1,C_1)) = o_1 $\wedge$ Signal(Out(2,C_1)) = o_2

7. Debug the knowledge base

   May have omitted assertions like $1 \neq 0$

First-order inference

- **Inference rules**
  - Universal instantiation
  - Existential instantiation
  - Can be applied to sentences with quantifiers to obtain sentences without quantifiers

- **First-order inference**
  - Convert the knowledge base (KB) to propositional logic
  - Use propositional inference

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., $\forall x\ \mathit{King}(x) \wedge \mathit{Greedy}(x) \Rightarrow \mathit{Evil}(x)$ yields:

$\text{King(John)} \wedge \text{Greedy(John)} \Rightarrow \text{Evil(John)}$

$\text{King(Richard)} \wedge \text{Greedy(Richard)} \Rightarrow \text{Evil(Richard)}$

$\text{King(Father(John))} \wedge \text{Greedy(Father(John))} \Rightarrow \text{Evil(Father(John))}$

Existential instantiation (EI)

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

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

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

Reduction to propositional inference

Suppose the KB contains just the following:

$\forall x\ \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$
King(John)
Greedy(John)
Brother(Richard, John)

- Instantiating the universal sentence in all possible ways, we have:
  $\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$
  $\text{King}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$
  King(John)
  Greedy(John)
  Brother(Richard, John)

- The new KB is propositionalized: proposition symbols are

  King(John), Greedy(John), Evil(John), King(Richard), etc.

Reduction Contd...

- Every FOL KB can be propositionalized so as to preserve entailment
- A ground sentence is entailed by new KB iff entailed by original KB
- Idea: propositionalize KB and query, apply resolution, return result
- **Problem**: with function symbols, there are infinitely many ground terms,
  - e.g., Father(Father(Father(John)))

Reduction contd.

Theorem: Herbrand (1930): If a sentence α is entailed by a FOL KB, it is entailed by a finite subset of the propositionalized KB

Idea: For $n = 0$ to ∞ do
create a propositional KB by instantiating with depth n terms
see if α is entailed by this KB

e.g. constant symbols: *Richard* and *John*
terms with depth 1: (Father(Richard)) and (Father (John))

Problem: works if α is entailed, loops if α is not entailed

Theorem: Turing (1936), Church (1936) Entailment for FOL is semi-decidable (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.)

Problems with Propositionalization

- Propositionalization is inefficient
  - seems to generate lots of irrelevant sentences

E.g., for query: Evil (x)

KB:

$\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)

King(John)

$\forall y$ Greedy(y)

Brother(Richard, John)

Perverse inference: King (Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)

Irrelevant facts: King (Richard), Greedy(Richard), Evil(Richard)

But, Evil (John)-perfect for human being

- With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations.

First-order inference rule

- If there is some substitution θ that makes the premise of the implication identical to sentences already in the KB then we can assert the conclusion of implication after applying
  - {x/John} achieves the aim

- Let us assume that "*everyone is greedy* "rather than knowing *Greedy (John)*
  - ∀y Greedy(y)

We can still conclude that Evil(John):

-John is king

-John is greedy as everyone is greedy

- Apply substitution {x/John, y/John} to

  premises: King (x) and Greedy (y) === conclusion of implication can be

  KB: King (John) and Greedy (John) inferred

Generalized Modus Ponens (GMP)

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$

where $p_i'\theta = p_i\,\theta$ for all i

p_1' is *King*(*John*) p_1 is *King*(x)

p_2' is *Greedy*(y) p_2 is *Greedy*(x)

θ is {x/John, y/John} q is *Evil*(x)

$q\,\theta$ is *Evil*(*John*)

- GMP used with KB of definite clauses (exactly one positive literal)
- All variables assumed universally quantified

Soundness of GMP

- Need to show that

$$p_1', \ldots, p_n', (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\theta$$

provided that $p_i'\theta = p_i\theta$ for all i

- Lemma: For any sentence p, we have $p \models p\theta$ by UI

1. $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \ldots \wedge p_n\theta \Rightarrow q\theta)$
2. $p_1' \wedge \ldots \wedge p_n' \models p_1'\theta \wedge \ldots \wedge p_n'\theta$
3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

GMP

- A lifted version of Modus Ponens
  - Raises MP from propositional to FOPL

- Advantages over prositionalization
  - Make only those substitutions which are required to allow particular inferences to proceed

- MP allows any single sentence to the left of implication
- GMP allows any number of sentences

Unification

- Unification-Process of finding substitutions that make different logical expressions identical
- 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 αθ = βθ

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)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- Unification-Process of finding substitutions that make different logical expressions identical
- 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 αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- Unification-Process of finding substitutions that make different logical expressions identical
- 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 αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- Unification-Process of finding substitutions that make different logical expressions identical
- 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 αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,OJ)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- Unification-Process of finding substitutions that make different logical expressions identical
- 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 αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,OJ)	{fail}

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- Complication
  - Returns substitutions that make two arguments same
  - But, could be more than one such unifier

- To unify *Knows(John, x)* and *Knows(y, z)*,
  $\theta = \{y/John, x/z\}$ or $\theta = \{y/John, x/John, z/John\}$

- The first unifier is more general than the second
  - Second one can be obtained from the first by additional substitution $\{z/John\}$
  - Places fewer restrictions on the values of the variables

- There is a single most general unifier (MGU) that is unique up to renaming of variables
  $MGU = \{ y/John, x/z \}$

First order definite clauses

- Resembles with propositional horn clauses
  - Disjunctions of literals of which at *most one is positive*
- Definite clause
  - Atomic sentence or
  - Implication whose
    - antecedent is a conjunction of positive literals
    - Consequent is a single positive literal
- E. g. of FO definite clause

  $\text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$
  $\text{King}(\text{John})$
  $\text{Greedy}(y)$
- Unlike propositional definite clause, FO can include variables
  - Variables: universally quantified
  - General norm is to remove universal quantifiers for writing definite clause

First order definite clauses

- Definite clauses: suitable for use in GMP

Every knowledge base can't be converted into a set of definite clauses because of single positive literal restriction

Example knowledge base

- The law says that it is a crime for an American to sell weapons to hostile nations. The country C, 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.

…

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

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

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

$Owns(C,M_1)$ and $Missile(M_1)$: **definite clauses**

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

$Missile(x) \land Owns(C, x) \Rightarrow Sells(West, x, C)$

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

West, who is American …

$American(West)$

The country C, an enemy of America …

$Enemy(C, America)$

Data-log KB: as does not contain any function symbol

Forward chaining algorithm (simple)

- **Overall procedure**
  - Start from the known facts
  - Trigger all rules whose premises are satisfied
  - Add the conclusions of the rules to the known facts
  - Process continues until the query is answered or no new facts are added
  - Fact is not new if it is just a renaming of known fact
- **Renaming**
  - One sentence is a renaming of another if they are identical except for the names of the variables
  - E.g.

  Likes (x, IceCream) and Likes (y, IceCream) are identical

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**
    $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ 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*

Example knowledge base contd.

…

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

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

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

$Owns(C,M_1)$ and $Missile(M_1)$: **definite clauses**

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

$Missile(x) \land Owns(C, x) \Rightarrow Sells(West, x, C)$

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

West, who is American …

$American(West)$

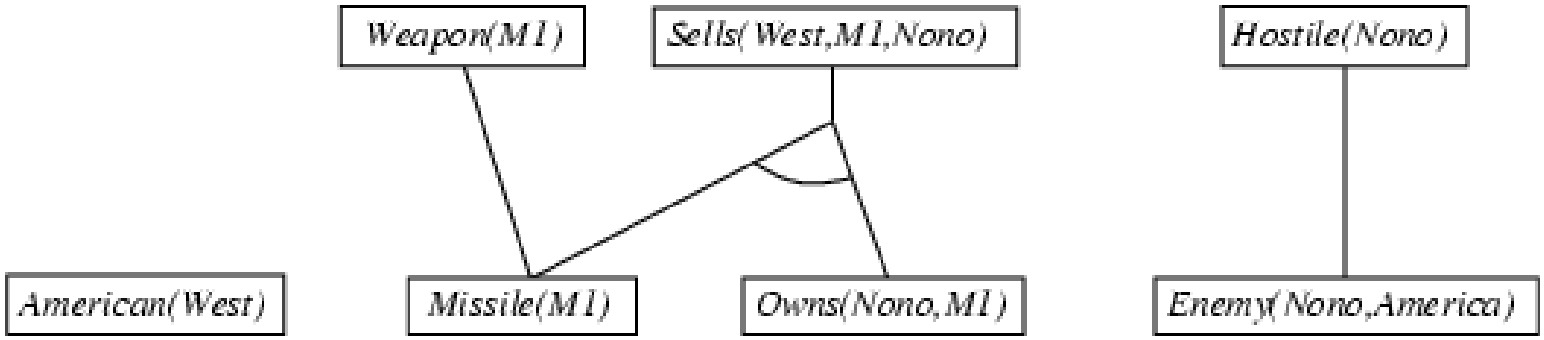
The country C, an enemy of America …

$Enemy(C, America)$

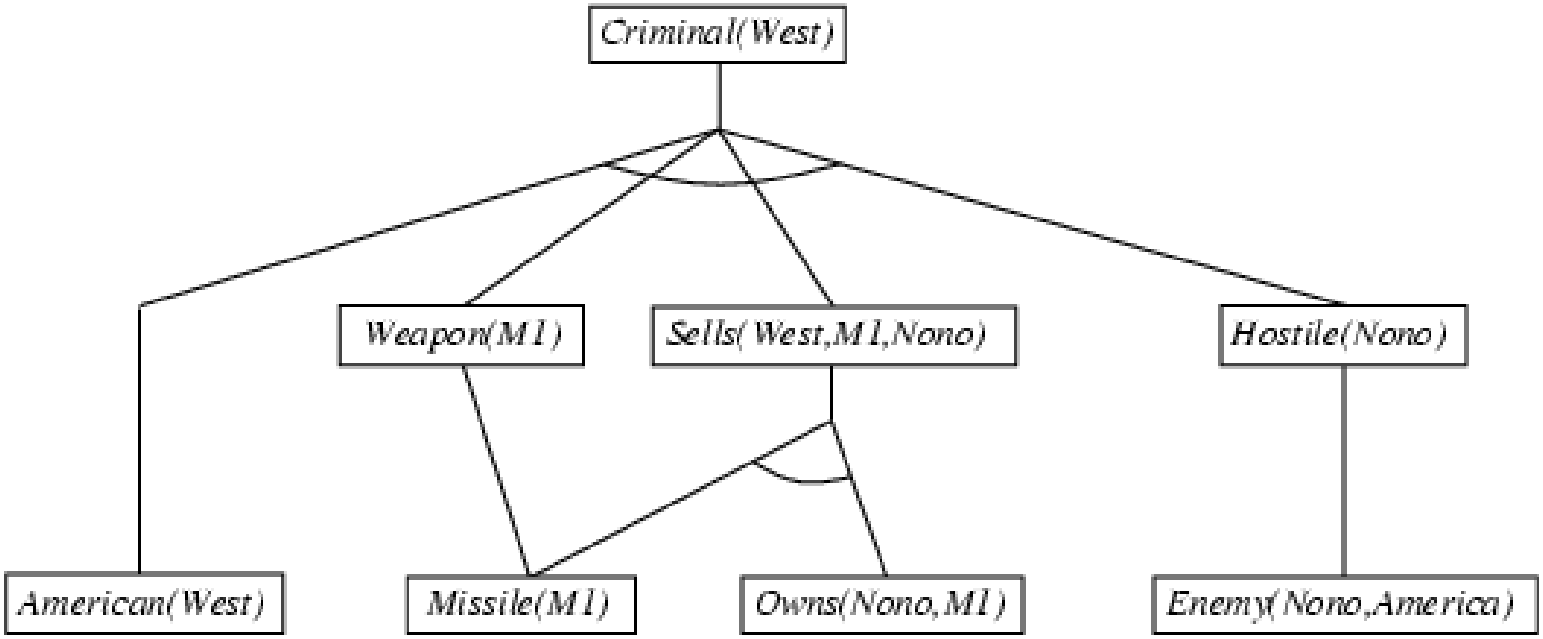
Data-log KB: as does not contain any function symbol

Forward chaining proof (*for a particular value of C, i.e. Nono*)

Forward chaining proof



Forward chaining proof



First order forward chaining

- Fixed-point KB
  - No new inferences are possible

- Fixed point of FO forward chaining vs. Fixed point propositional forward chaining
  - Almost equivalent
  - Difference: FO fixed point can include universally quantified atomic sentences

Properties of forward chaining

- Sound
  - Every inference is just an application of GMP, which is sound
- Complete for definite KB
  - Answers every query whose answers are entailed by any KB of definite clauses
- Datalog KB = first-order definite clauses + no functions
- FC terminates for Datalog in finite number of iterations=number of possible facts that can be added
  - k: maximum arity (# arguments) of any predicate
  - p: # predicates
  - n: # constant symbols
  - Distinct ground facts <= pn^k (algo reaches fixed point after this # iterations)

Efficiency of Forward chaining

- Irrelevant facts
  - Forward chaining makes allowable inferences based on known facts even if they are irrelevant to the goal at hand
- Avoiding irrelevant facts
  - Backward chaining
  - Restrict forward chaining to a selected subset of rules
  - Rewrite the rule set using the information about the goals
    - Allow only relevant variable bindings (*magic set*) during forward inference
  - E.g. goal: Criminal (West)

    new rule: $Magic(x) \wedge American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$

    Magic (West) added to the KB

    KB may contain millions of Americans, only Colonel West will be considered during forward inference process: *Hybrid approach*

Backward chaining

- Work backward from the goal, chain through rules to find known facts that support the proof
- Overall procedure
  - Given: set of goals, original query
  - Returns the set of all substitutions satisfying the query
  - List of goals put as a stack (current branch of the proof succeeds if all the goals are satisfied)
  - Take the first goal from the list
  - Find every clause in KB whose *head* unifies with the goal
  - *Body* (premise) added to the goal stack
  - When goal unifies with known fact (clause with a head but no body)
    - No new goals are added
    - Goal is solved

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

SUBST(COMPOSE(θ_1, θ_2), p) = SUBST(θ_2, SUBST(θ_1, p))

Example knowledge base Contd

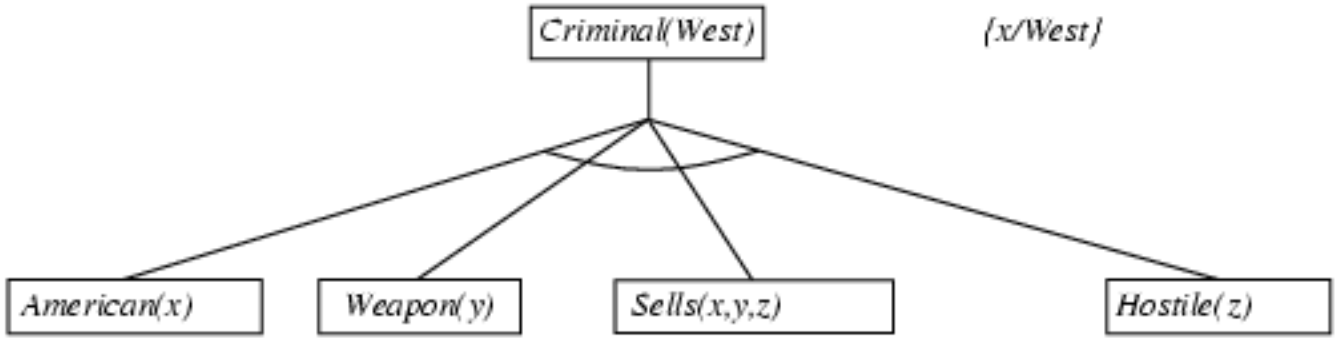
…
it is a crime for an American to sell weapons to hostile nations:
American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Nono … has some missiles, i.e., ∃x Owns(Nono, x) ∧ Missile(x):
Owns(Nono,M_1) and Missile(M_1): ***definite clauses***

… all of its missiles were sold to it by Colonel West
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(West, x, Nono)
Missiles are weapons:
Missile(x) ⇒Weapon(x)
An enemy of America counts as "hostile“:
Enemy(x, America) ⇒ Hostile(x)
West, who is American …
American(West)
The country Nono, an enemy of America …
Enemy(Nono, America)

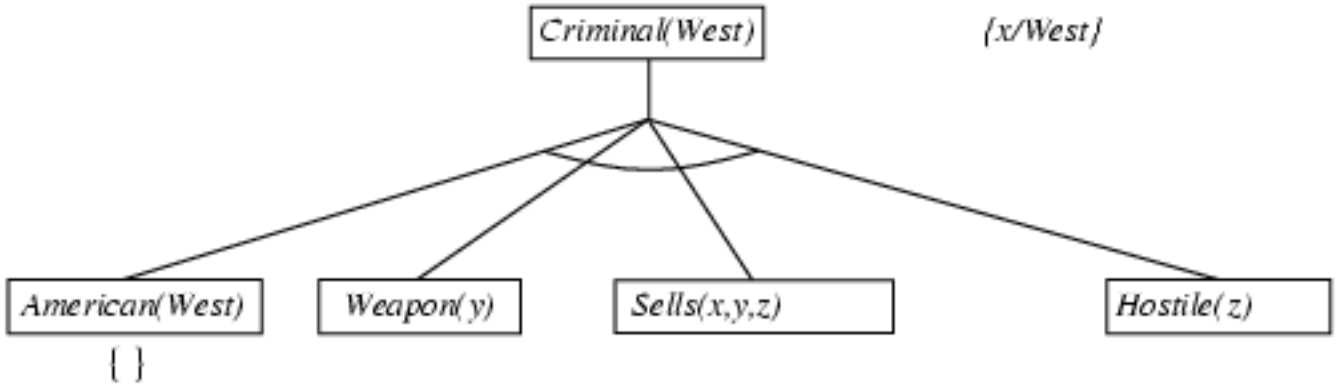
Data-log KB: as does not contain any function symbol

Backward chaining example

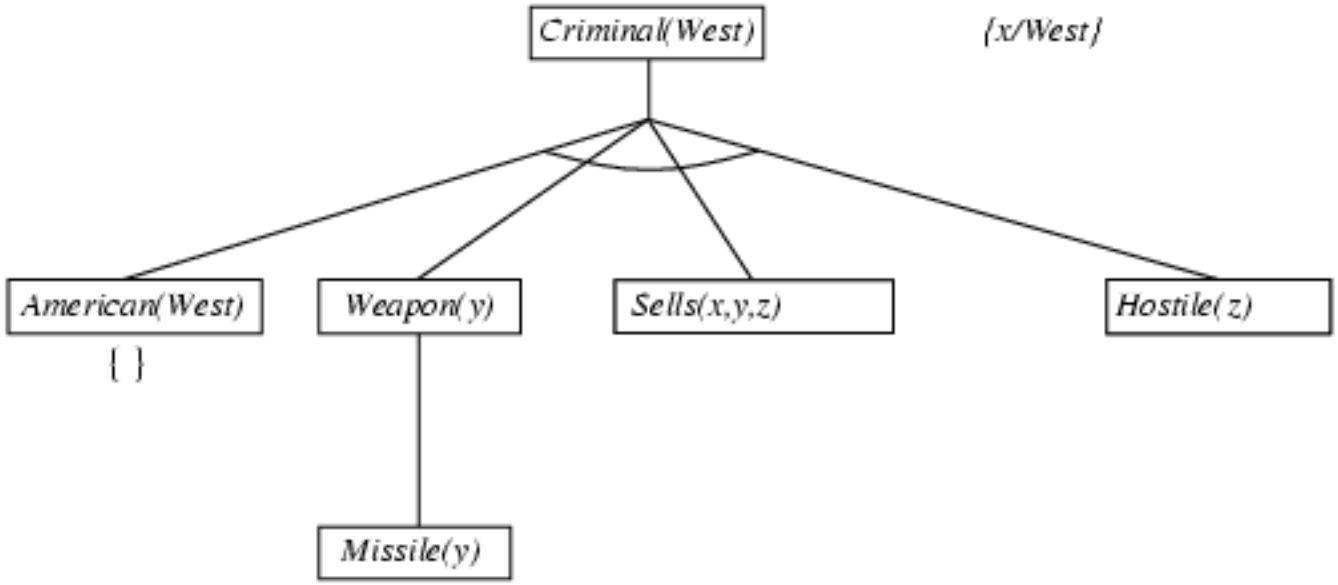
Backward chaining example



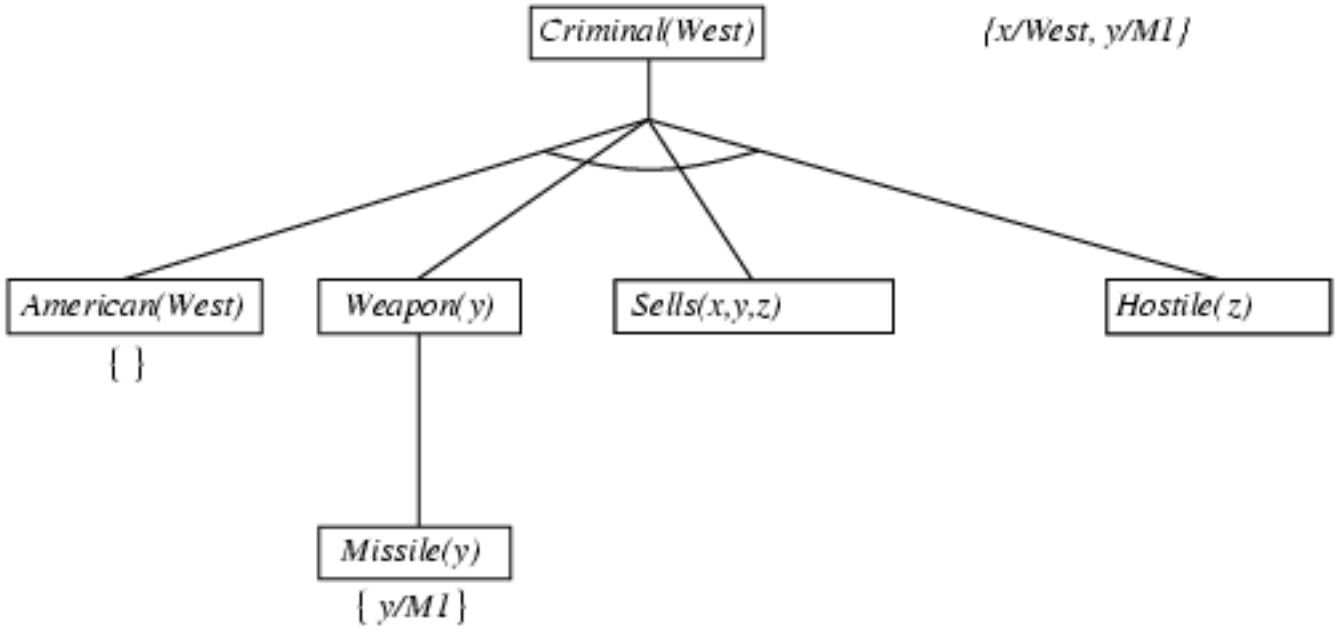
Backward chaining example



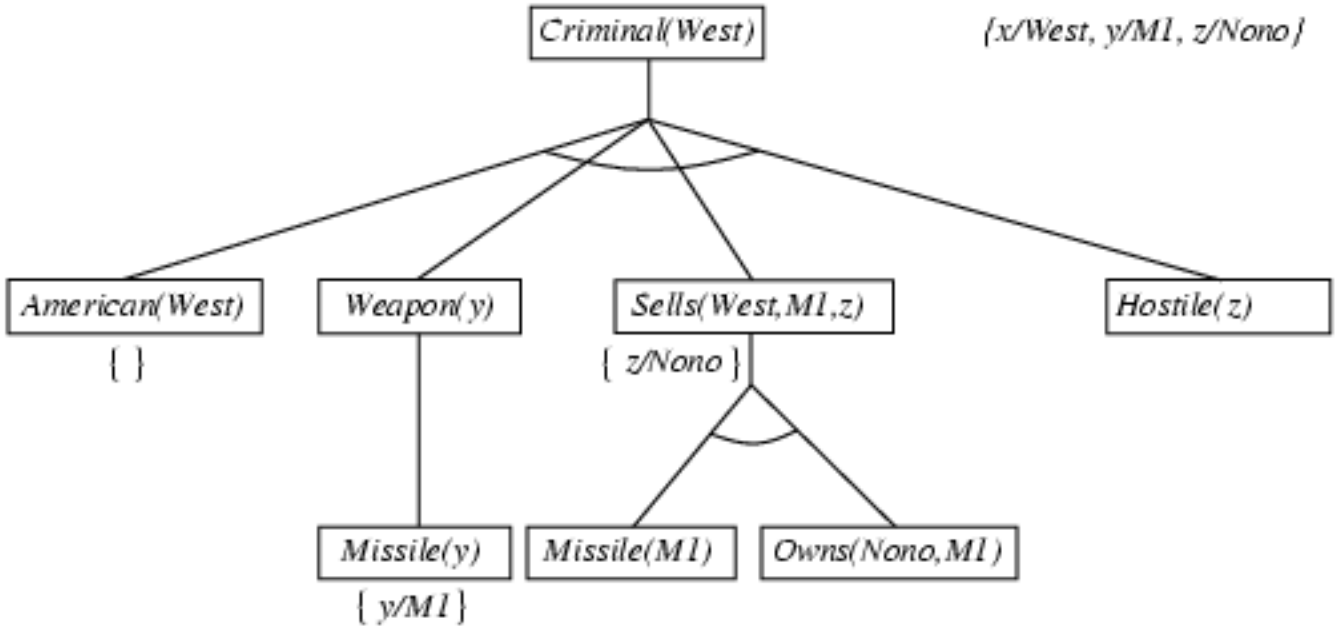
Backward chaining example



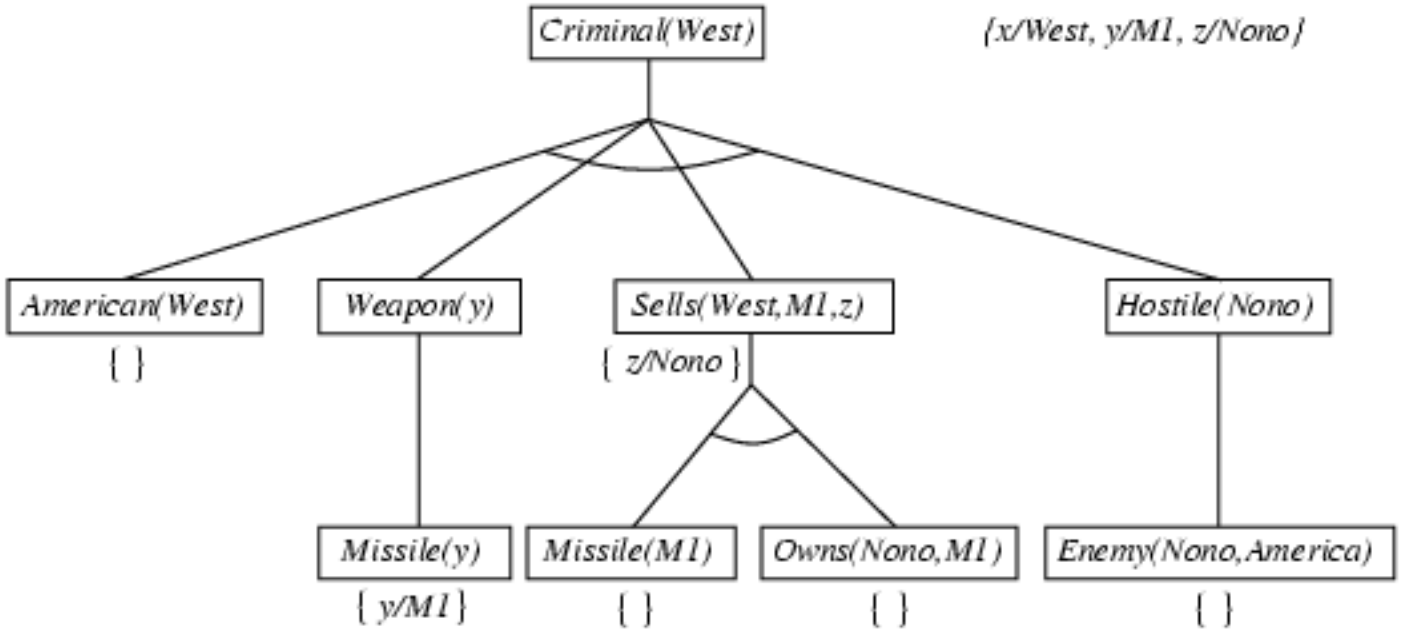
Backward chaining example



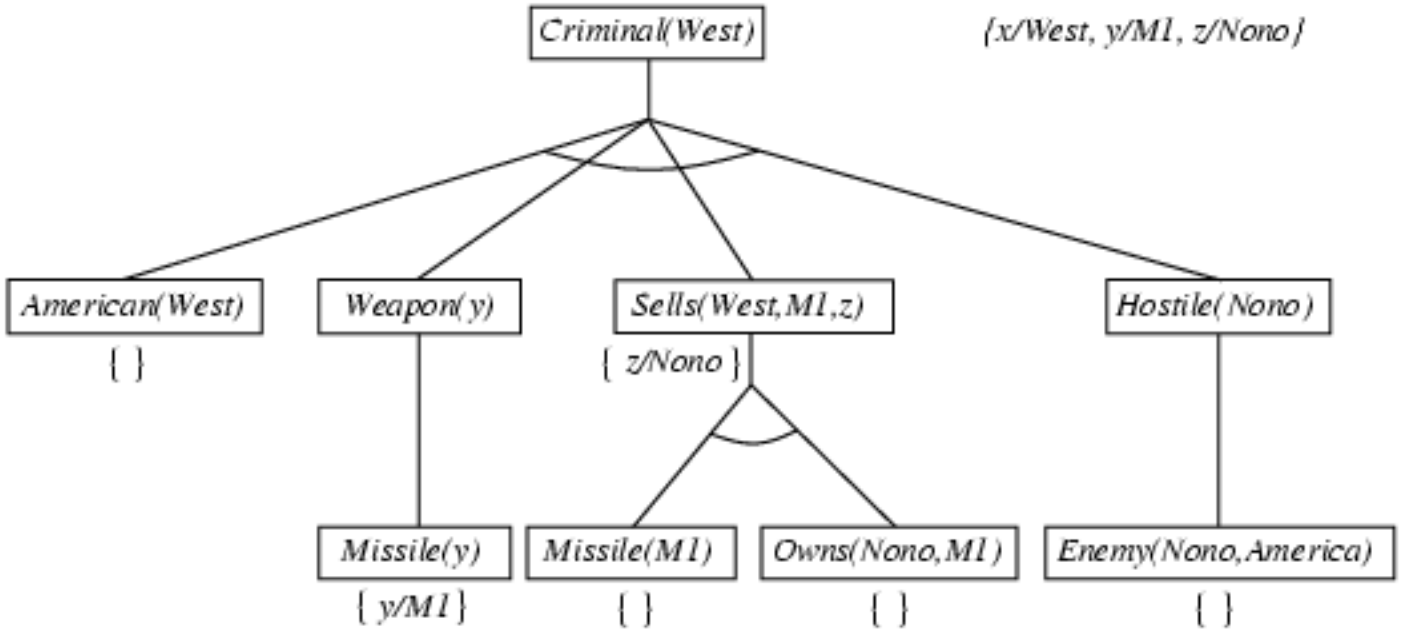
Backward chaining example



Backward chaining example



Backward chaining example



Properties of backward chaining

- Equivalent to DFS
- Depth-first recursive proof search: space is linear in size of proof (neglecting the space required for accumulating the solutions)
- Incomplete due to infinite loops

  $\Rightarrow$ fix by checking current goal against every goal on stack
- Inefficient due to repeated sub-goals (both success and failure)

  $\Rightarrow$ fix using caching of previous results (extra space)

- Widely used for logic programming

Resolution: brief summary

- Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\texttt{Unify}(\ell_i, \neg m_j) = \theta$.

- Two clauses are assumed to be standardized apart so that they share no variables

For example,

$$\frac{\neg Rich(x) \vee Unhappy(x) \qquad Rich(Ken)}{Unhappy(Ken)}$$

with $\theta = \{x/Ken\}$

- Apply resolution steps to CNF(KB $\wedge \neg\alpha$); complete for FOL

Conversion to CNF

- Everyone who loves all animals is loved by someone:
  $\forall x\ [\forall y\ \mathit{Animal}(y) \Rightarrow \mathit{Loves}(x,y)] \Rightarrow [\exists y\ \mathit{Loves}(y,x)]$

- 1. Eliminate biconditionals and implications
  $\forall x\ [\neg \forall y\ \neg \mathit{Animal}(y) \vee \mathit{Loves}(x,y)] \vee [\exists y\ \mathit{Loves}(y,x)]$

- 2. Move $\neg$ inwards: $\neg \forall x\ p \equiv \exists x\ \neg p,\ \neg \exists x\ p \equiv \forall x\ \neg p$
  $\forall x\ [\exists y\ \neg(\neg \mathit{Animal}(y) \vee \mathit{Loves}(x,y))] \vee [\exists y\ \mathit{Loves}(y,x)]$
  $\forall x\ [\exists y\ \neg\neg \mathit{Animal}(y) \wedge \neg \mathit{Loves}(x,y)] \vee [\exists y\ \mathit{Loves}(y,x)]$
  $\forall x\ [\exists y\ \mathit{Animal}(y) \wedge \neg \mathit{Loves}(x,y)] \vee [\exists y\ \mathit{Loves}(y,x)]$

Conversion to CNF contd.

3. Standardize variables: each quantifier should use a different one

$\forall x\ [\exists y\ Animal(y) \land \neg Loves(x,y)] \lor [\exists z\ Loves(z,x)]$

4. **Skolemization**: process of removing existential quantifiers by elimination

$\exists x\ P(x) = P(A)$, where A is a new constant

$\forall x\ [Animal(A) \land \neg Loves(x, A)] \lor Loves(B,x)$

- Completely wrong meaning " *It says that everyone either fails to love a particular animal A or is loved by some particular entity B*"

-original meaning: *allows each person to fail to love a different animal or to be loved by a different person*

Solution: Skolem entities depend on x.

Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

$\forall x\ [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$

Conversion to CNF contd.

5. Drop universal quantifiers:

$$[Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x)$$

6. Distribute $\lor$ over $\land$:

$$[Animal(F(x)) \lor Loves(G(x), x)] \land [\neg Loves(x, F(x)) \lor Loves(G(x), x)]$$

Example knowledge base contd.

…

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

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

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

$Owns(Nono, M_1)$ and $Missile(M_1)$: ***definite clauses***

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

West, who is American …

$American(West)$

The country Nono, an enemy of America …

$Enemy(Nono, America)$

Data-log KB: as does not contain any function symbol

CNF of Crime Example

$\neg$American(x) $\vee\neg$Weapon(y) $\vee\neg$Sells(x,y,z) $\vee\neg$Hostile(z) $\vee$Criminal(x)

$\neg$ Missile(x) $\vee\neg$Owns(Nono,x) $\vee$Sells(West, x, Nono)

- $\neg$ Enemy(x, America) $\vee$Hostile(x)
- $\neg$ Missile(x) $\vee$ Weapon(x)

Owns(Nono, M_1)

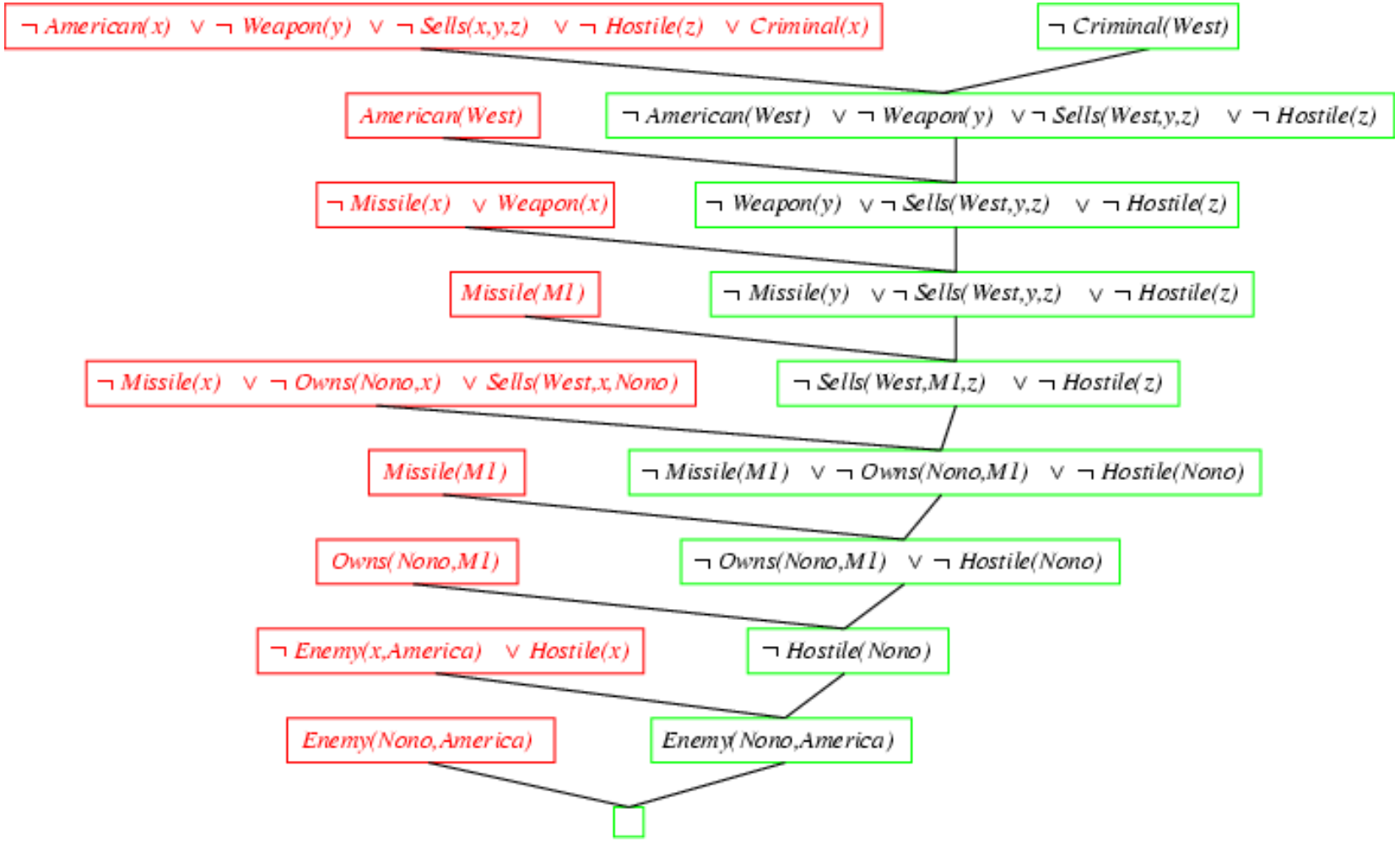
Missile (M_1)

American (West)

Enemy (Nono, America)

Include goal: $\neg$Criminal (West)

Resolution proof: definite clauses



Structured Knowledge Representations

- Modeling-based representations reflect the structure of the domain, and then reason based on the model
  - Semantic Nets
  - Frames
  - Scripts
- Sometimes called associative networks

Basics of Associative Networks

- All include
  - Concepts
  - Various kinds of links between concepts
    - "has-part" or aggregation
    - "is-a" or specialization
    - More specialized depending on domain
- Typically also include
  - Inheritance
  - Some kind of procedural attachment

Semantic Nets

graphical representation for propositional information

originally developed by M. R. Quillian as a model for human memory

labeled, directed graph

nodes represent objects, concepts, or situations

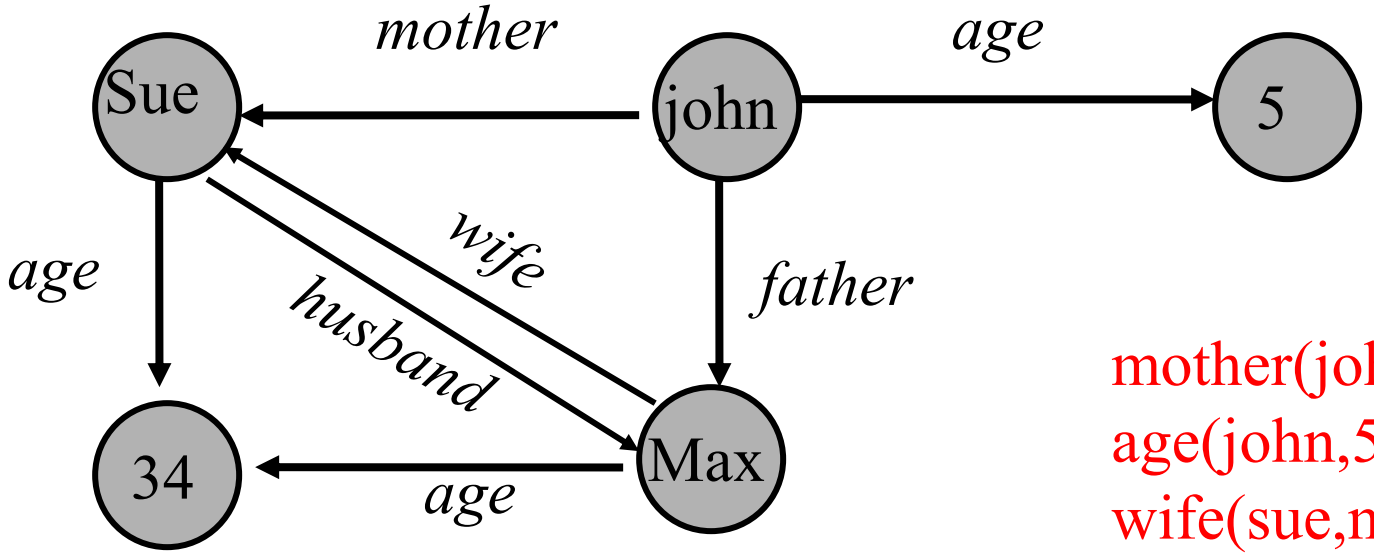
- labels indicate the name
- nodes can be instances (individual objects) or classes (generic nodes)

links represent relationships

- the relationships contain the structural information of the knowledge to be represented
- the label indicates the type of the relationship

Nodes and Arcs

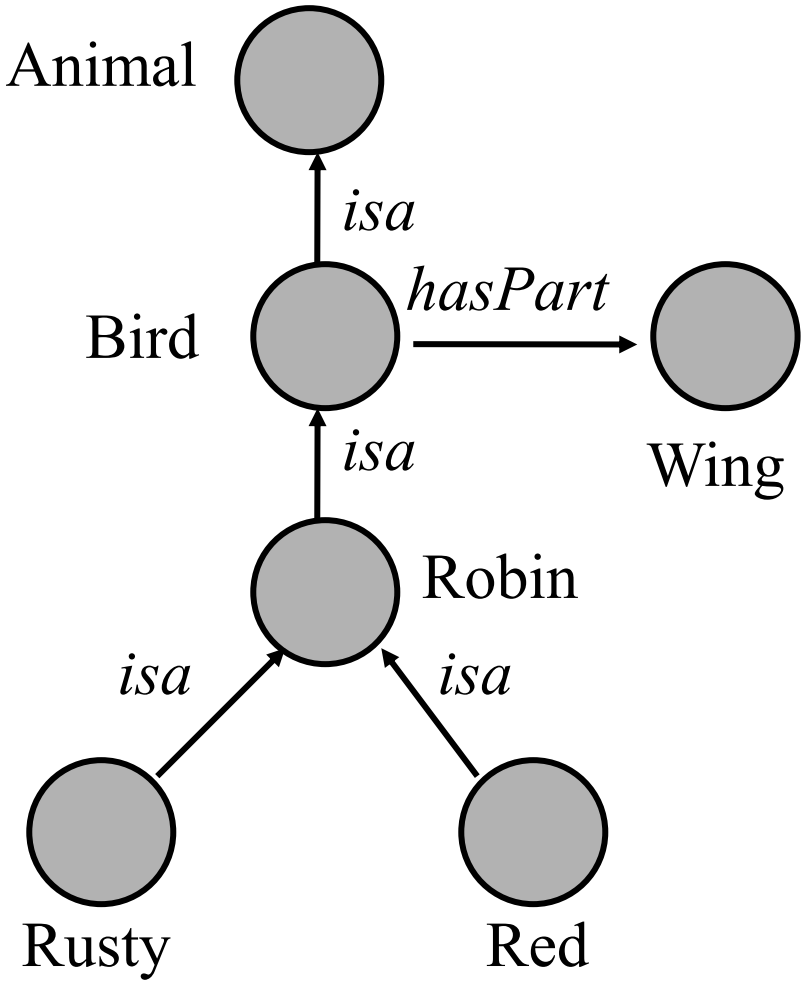
- Arcs define binary relationships that hold between objects denoted by the nodes



mother(john,sue)
age(john,5)
wife(sue,max)
age(max,34)
...

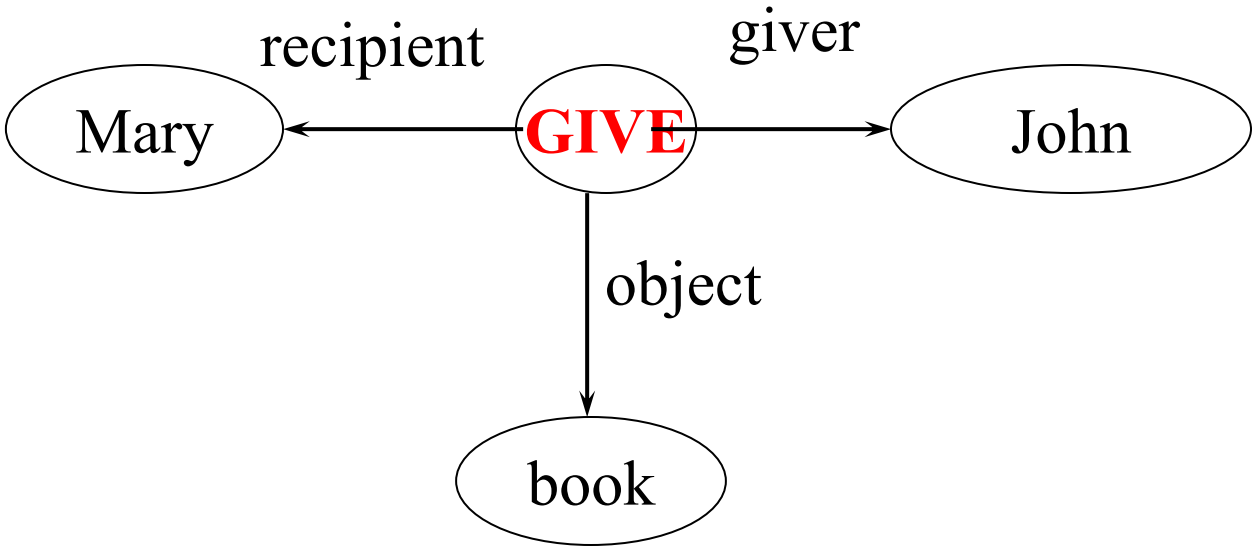
Semantic Networks

- The ISA (is-a) or AKO (a-kind-of) relation is often used to link instances to classes, classes to superclasses
- Some links (e.g. hasPart) are inherited along ISA paths
- The semantics of a semantic net can be relatively informal or very formal
  - often defined at the implementation level



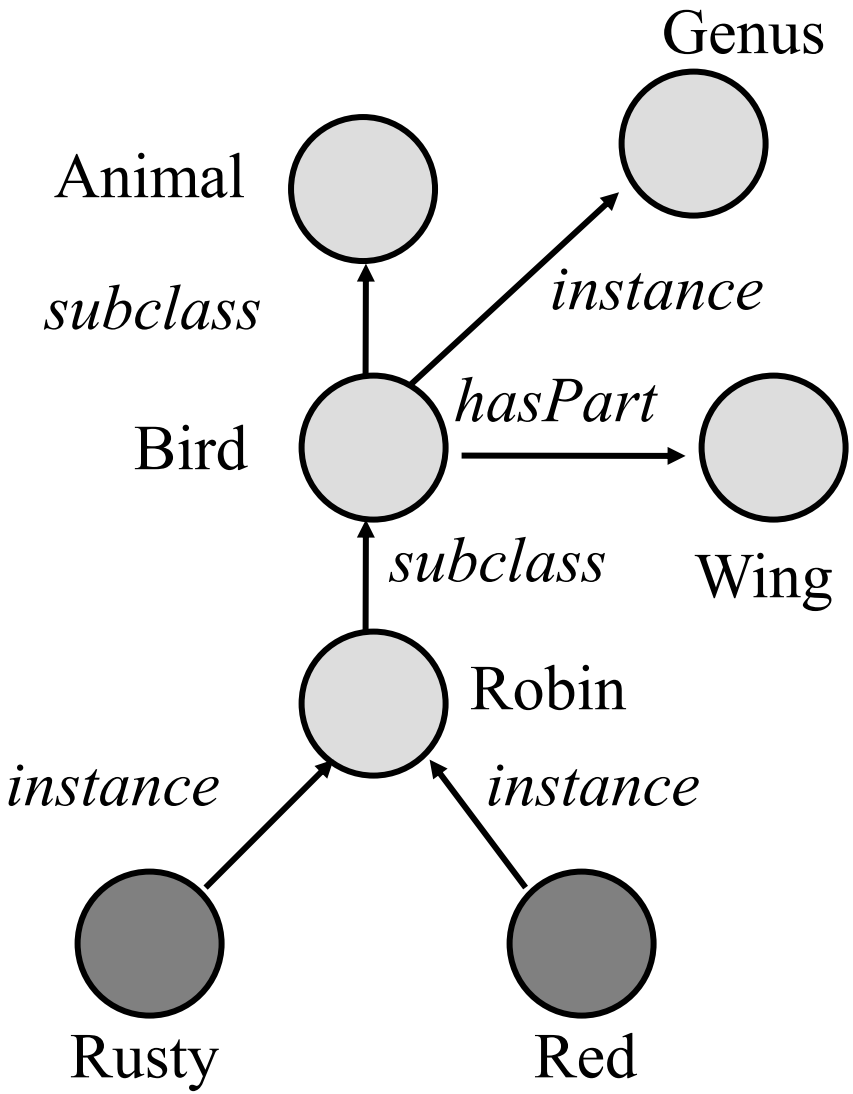
Non-binary relations: Reification

- We can represent the generic *give* event as a relation involving three things: A giver, A recipient, An object
- This is an example of what logicians call "reification"
  - reify v : consider an abstract concept to be real



Individuals and Classes

- **Many semantic** networks distinguish
  - –nodes representing individuals and those representing classes
  - –the "subclass" relation form the "instance-of" relation



Link types

Link Type	Semantics	Example
$A \xrightarrow{Subset} B$	$A \subset B$	$Cats \subset Mammals$
$A \xrightarrow{Member} B$	$A \in B$	$Bill \in Cats$
$A \xrightarrow{R} B$	$R(A,B)$	$Bill \xrightarrow{Age} 12$
$A \Rightarrow B$ (R)	$\forall x\ x \in A \Rightarrow R(x,B)$	$Birds \Rightarrow 2$ (Legs)
$A \Rightarrow B$ (R)	$\forall x\ \exists y\ x \in A \Rightarrow y \in B \wedge R(x,y)$	$Birds \Rightarrow Birds$ (Parent)

Inference by Inheritance

- One of the main kinds of reasoning done in a semantic net is the inheritance of values along the subclass and instance links

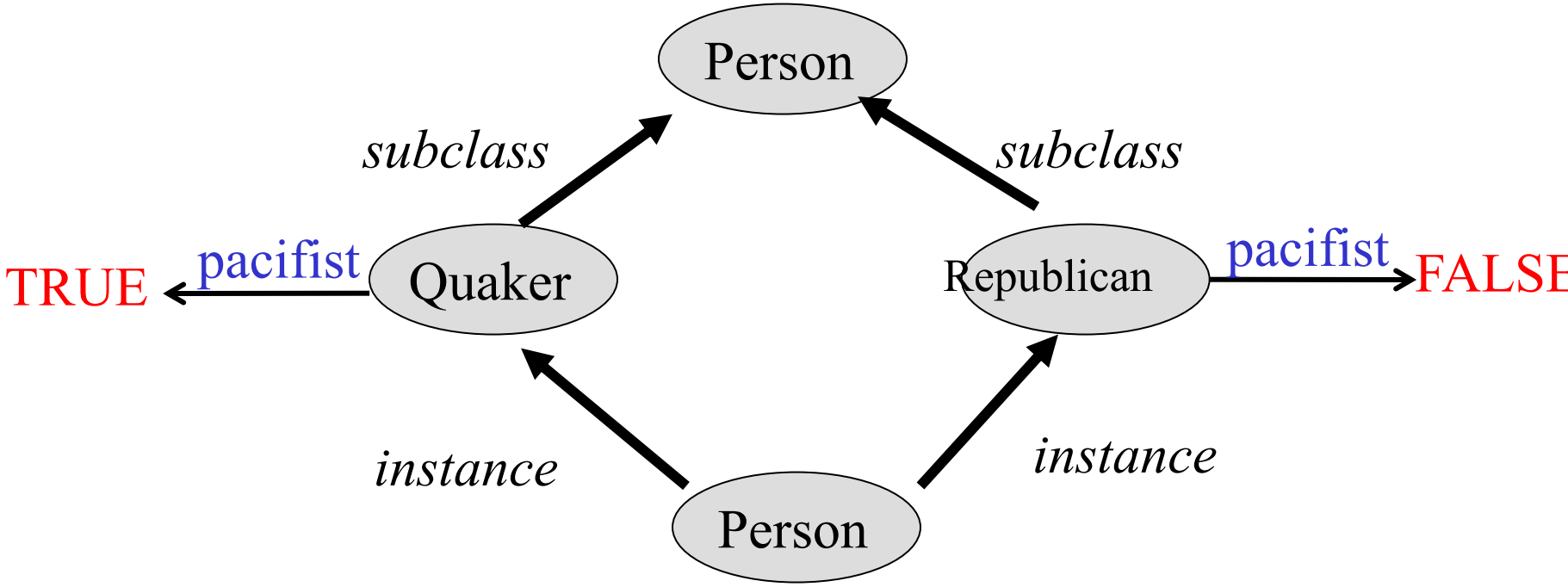
- Semantic networks differ in how they handle the case of inheriting multiple different values
  - All possible values are inherited, or
  - Only the "lowest" value or values are inherited

Multiple inheritance

- A node can have any number of superclasses that contain it, enabling a node to inherit properties from multiple "parent" nodes and their ancestors in the network
- Conflict or inconsistent properties can be inherited from different ancestors
- Rules are used to determine inheritance in such "tangled" networks where multiple inheritance is allowed:
  - If $X \subseteq A \subseteq B$ and both A and B have property P (possibly with different variable instantiations), then X inherits A's property P instance (closer ancestors override far away ones)
  - If $X \subseteq A$ and $X \subseteq B$ but neither $A \subseteq B$ nor $B \subseteq A$ and both A and B have property P with different and inconsistent values, then X will not inherit property P at all; or X will present both instances of P (from A and B) to the user

Nixon Diamond

- Scenario in which default assumptions lead to mutually inconsistent conclusions
  - usually, Quakers are pacifist
  - usually, Republicans are not pacifist
  - Richard Nixon is both a Quaker and a Republican



Nixon Diamond (Contd..)

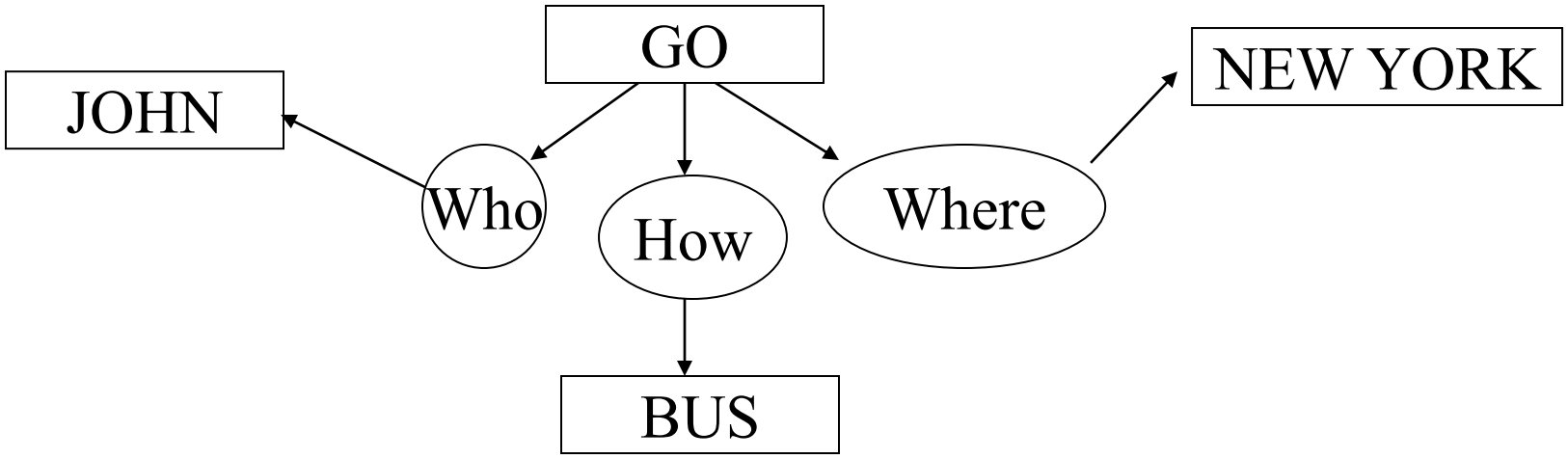
- Nixon is a Quaker→ one could assume that he is a pacifist
- Nixon is a Republican→ one could assume he is not a pacifist
- Problem: how a formal logic of nonmonotonic reasoning should deal with such cases?
- Two approaches can be adopted:
  - Skeptical (not easily convinced): No conclusion
    - Since Nixon can neither be proved to be a pacifist nor the contrary
  - Credulous (willing to believe):
    - Nixon can be proved to be a pacifist in at least one case-believed to be a pacifist;
    - Nixon can also be proved not to be a pacifist- believed not to be a pacifist

Nixon Diamond (Contd..)

- Two solutions
  - Skeptical is more preferred as
    - Credulous can allow proving both something and its contrary
  - Attach priorities to default assumptions
    - fact that "usually, Republicans are not pacifist" can be assumed more likely than "usually, Quakers are pacifist", leading to the conclusion that Nixon is not pacifist
- The name ***diamond*** comes from the fact that such a scenario, when expressed in inheritance networks, is a diamond shape

Conceptual Graphs

- ***Conceptual graphs*** are semantic nets representing the meaning of (simple) sentences in natural language
- Two types of nodes:
  - *Concept nodes:* there are two types of concepts, individual concepts and generic concepts
  - *Relation nodes* (binary relations between concepts)

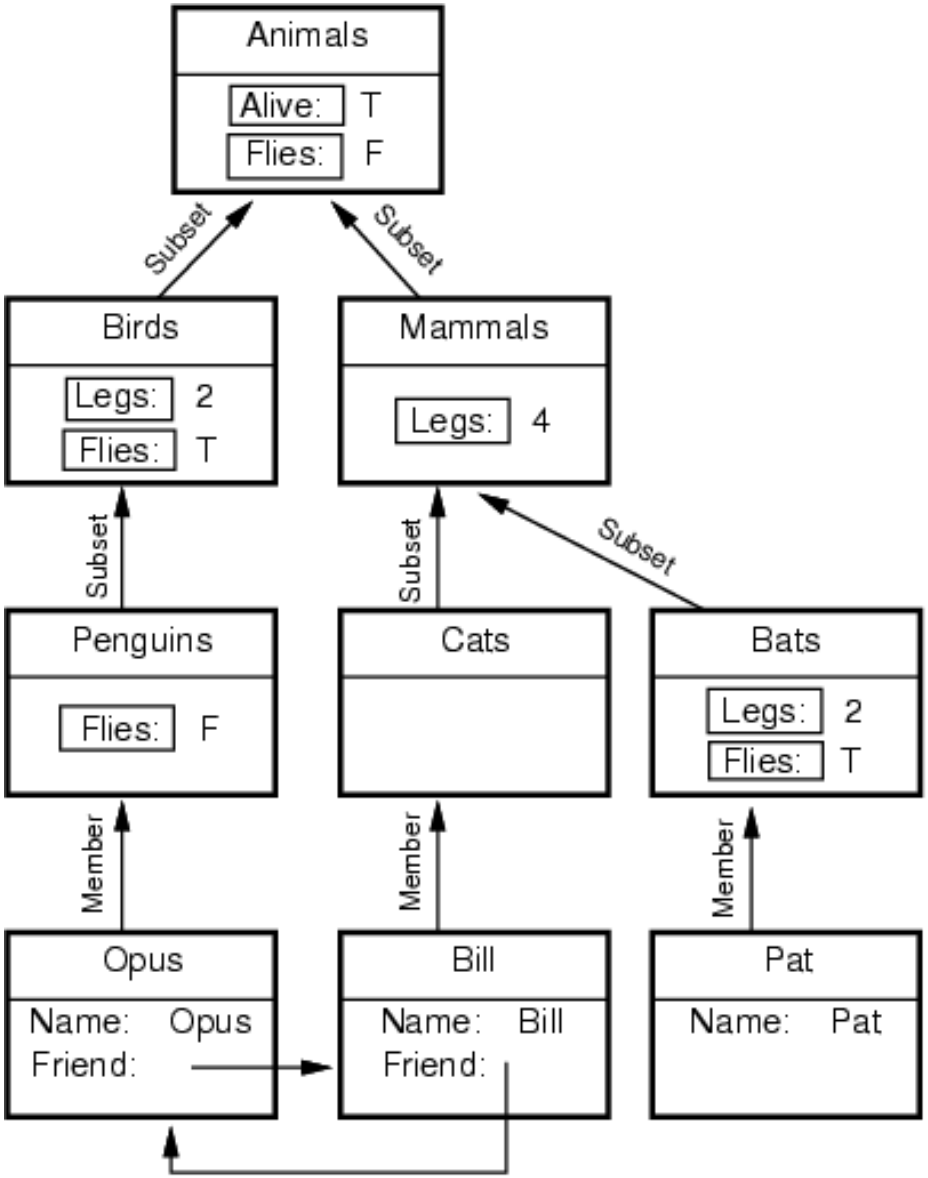


From Semantic Nets to Frames

- Semantic networks morphed into Frame Representation Languages in the ‘70s and ‘80s
- A frame is a lot like the notion of an object in OOP, but has more meta-data
  - Represents related knowledge about a subject
  - A **frame** has a set of **slots**
  - A **slot** represents a relation to another frame (or value)
  - A slot has one or more **facets**
  - A **facet** represents some aspect of the relation

Facets

- A slot in a frame holds more than a value
- Other facets might include:
  - current fillers (e.g., values)
  - default fillers
  - minimum and maximum number of fillers
  - type restriction on fillers (usually expressed as another frame object)
  - attached procedures (if-needed, if-added, if-removed)
  - salience measure
  - attached constraints or axioms
- In some systems, the slots themselves are instances of frames



(a) A frame–based knowledge base

Rel(Alive,Animals,T)
Rel(Flies,Animals,F)

Birds $\subset$ Animals
Mammals $\subset$ Animals

Rel(Flies,Birds,T)
Rel(Legs,Birds,2)
Rel(Legs,Mammals,4)

Penguins $\subset$ Birds
Cats $\subset$ Mammals
Bats $\subset$ Mammals

Rel(Flies,Penguins,F)
Rel(Legs,Bats,2)
Rel(Flies,Bats,T)

Opus $\in$ Penguins
Bill $\in$ Cats
Pat $\in$ Bats

Name(Opus,"Opus")
Name(Bill,"Bill")
Friend(Opus,Bill)
Friend(Bill,Opus)
Name(Pat,"Pat")

(b) Translation into first–order logic

Usage of Frames

- filling slots in frames
  - can inherit the value directly
  - can get a default value
  - these two are relatively inexpensive
  - can derive information through the attached procedures (or methods) that also take advantage of current context (slot-specific heuristics)
  - filling in slots also confirms that frame or script is appropriate for this particular situation

Example Frame: Low-level frame

Object	This Special AI Class
ISA	Special AI Class
Date	Spring, 2015
Time	11:00-13:00 Tuesday
Instructor	AE
Enrollment	6:15-9PM Thurs
Has_A	ROSTER

Frame Example: default frame

Object	Special AI Class
ISA	Class
Date	
Time	*14:00-15:00*
Instructor	IF-NEEDED: Ask department IF-ADDED: Update payroll
Enrollment	Count ROSTER: Student-IDs
Has_A	ROSTER

Another Example-high level frame:

- Object ROSTER
- ISA Class Record
- Enrollment Limit 60
- Enrollment Status *Open*
- Student-IDs IF-ADDED

  Increment Filler(Enrollment)

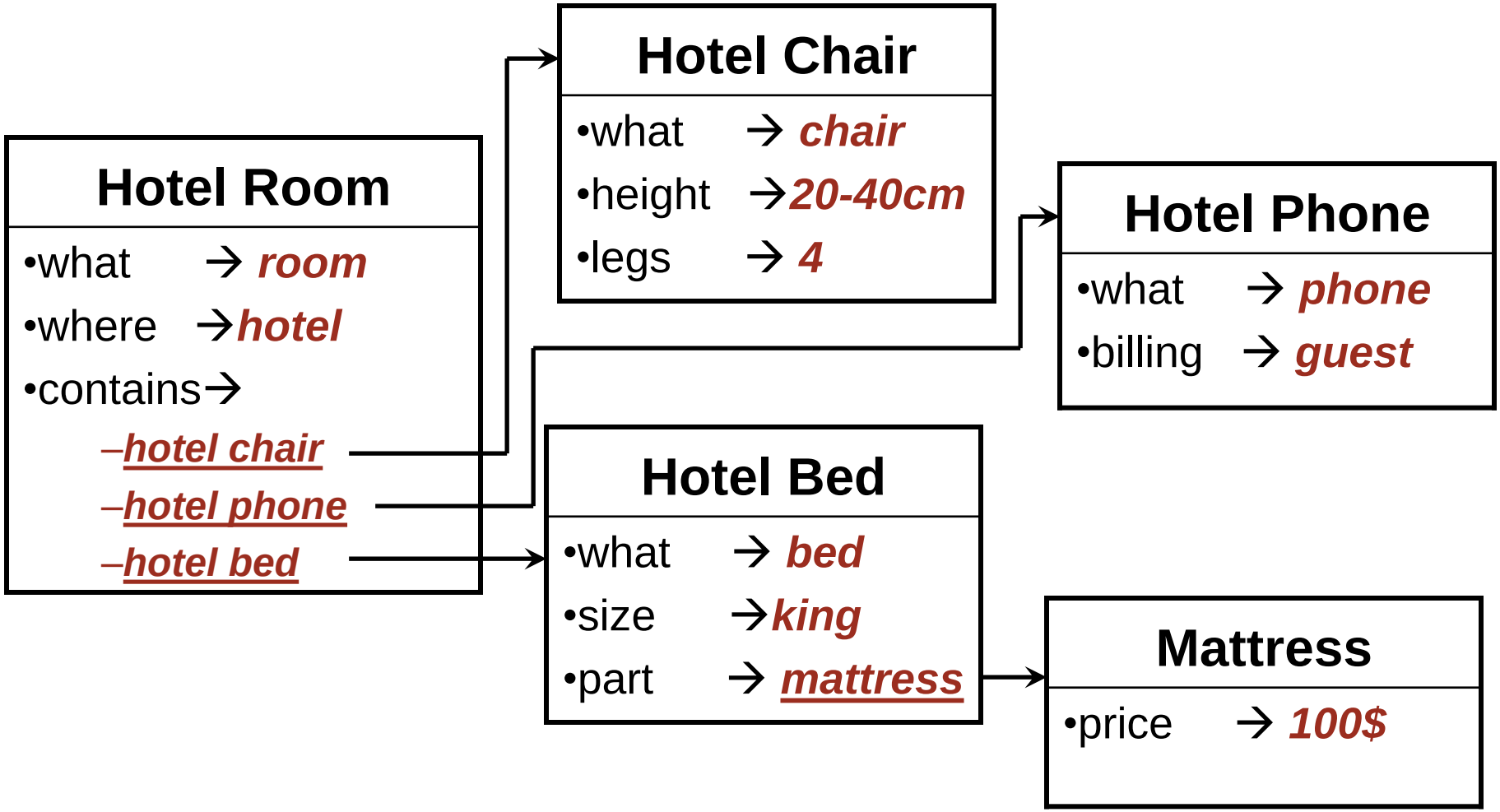
  If Filler(Enrollment) = Filler(Enrollment Limit)

  Then Filler(Status) = Closed

  IF-DROPPE

Inheritance

- Similar to Object-Oriented programming paradigm



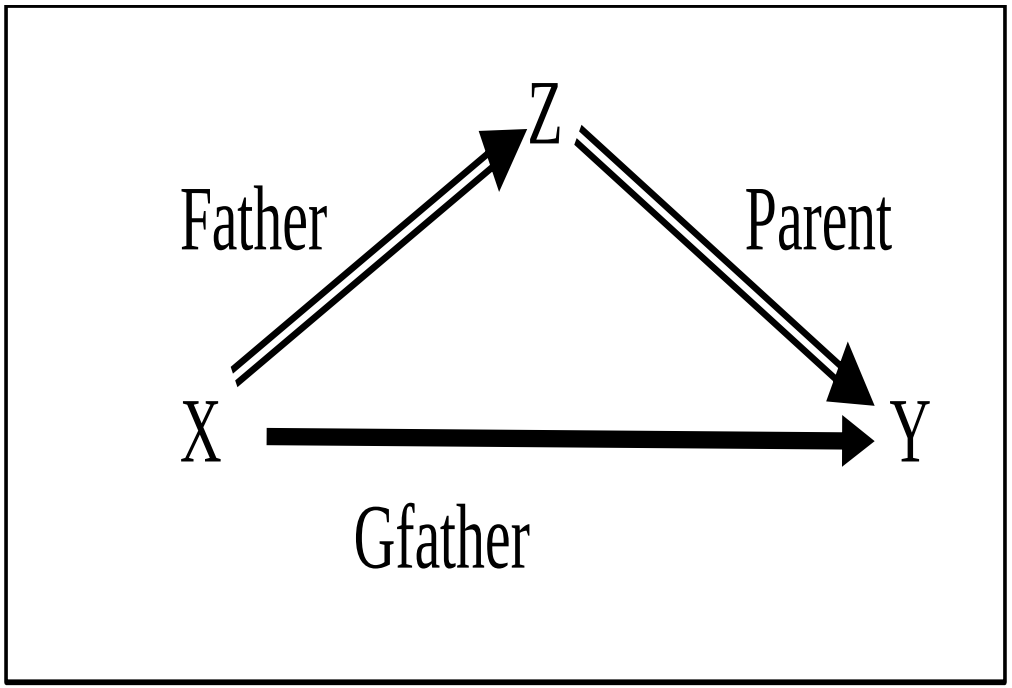
Extended Semantic Network

- In conventional Sem Net, clausal form of logic can not be expressed
- Extended Semantic Network (ESNet) combines the advantages of both logic and semantic network
- In the ESNet, terms are represented by nodes similar to Sem Net
- Binary predicate symbols in clausal logic are represented by labels on arcs of ESNet
  - An *atom* of the form "Love(john, mary)" is an arc labeled as 'Love' with its two end nodes representing 'john' and 'mary'
- *Conclusions* and *conditions* in clausal form are represented by different kinds of arcs
  - Conditions are drawn with two lines ⟸ and conclusions are drawn with one heavy line ⟵.

Examples

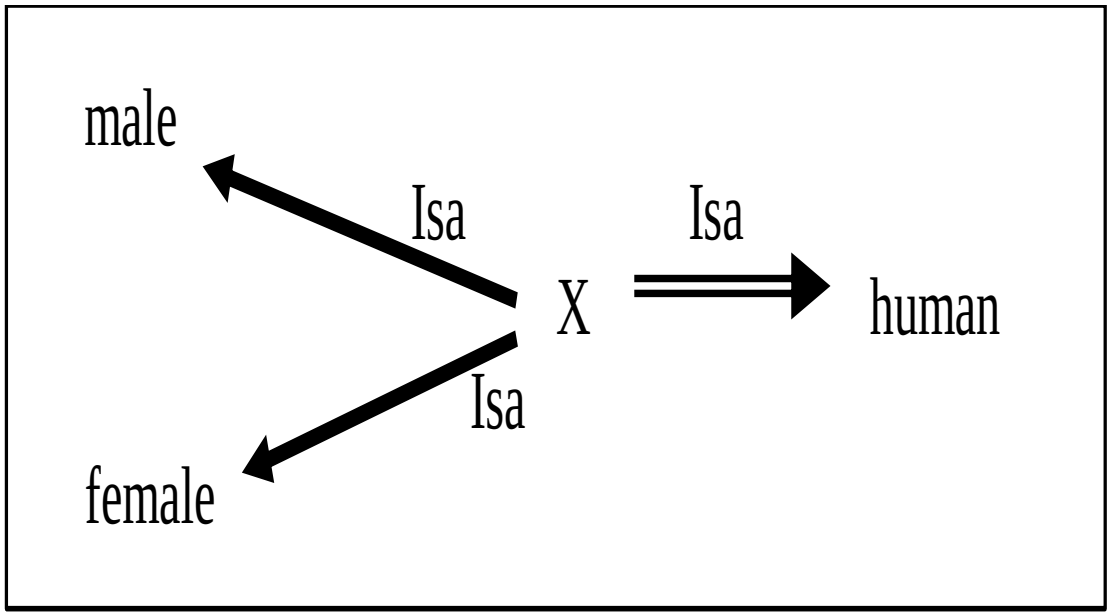
- Represent 'grandfather' definition

  Gfather(X, Y) ← Father(X, Z), Parent(Z, Y) in ESNet.



Cont...Example

- Represent clausal rule **"Male(X), Female(X) ← Human(X)"** using binary representation as "Isa(X, male), Isa(X, female) ← Isa(X, human)" and subsequently in ESNet as follows:

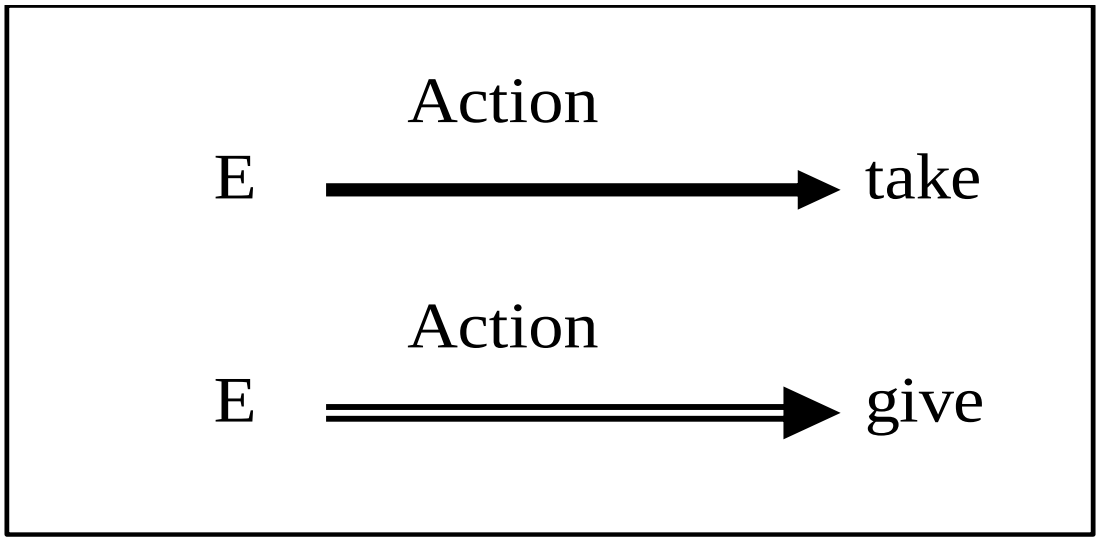


Inference Rules in ESNet

- Inference rules are embedded in the representation itself
- The inference that "for every action of giving, there is an action of taking" in clausal logic written as

  "Action(E, take) ← Action(E, give)"

ESNet

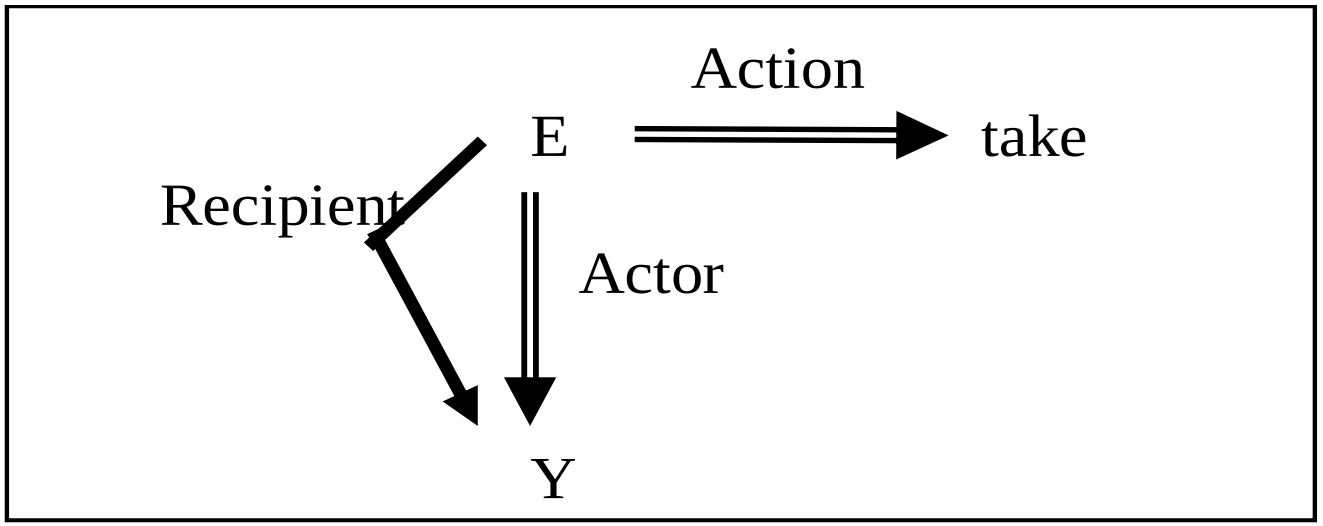


Cont...

- The inference rule such as "an actor of taking action is also the recipient of the action" can be easily represented in clausal logic as:
  - Here E is a variable representing an event where an action of taking is happening)

Recipient(E, Y) ← Action(E, take), Actor (E, Y)

ESNet



Example

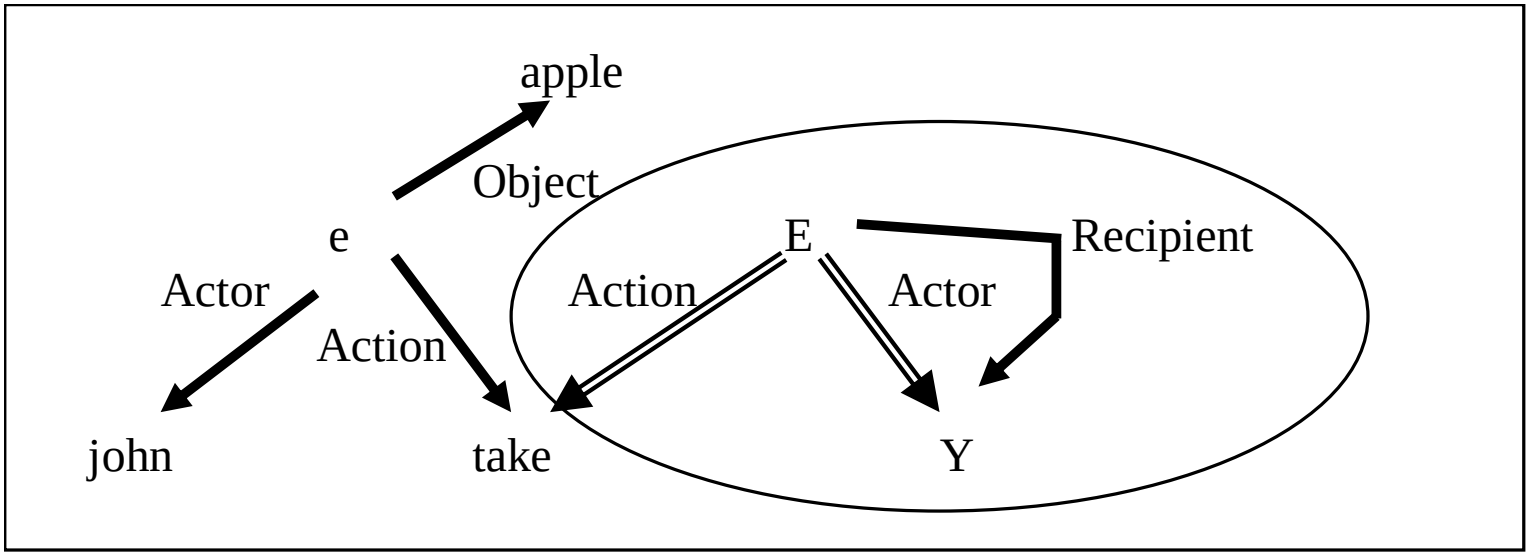
- Represents the following clauses of Logic in ESNet

  Recipient(E, Y) ← Action (E, take), Actor (E, Y)

  Object (e, apple)

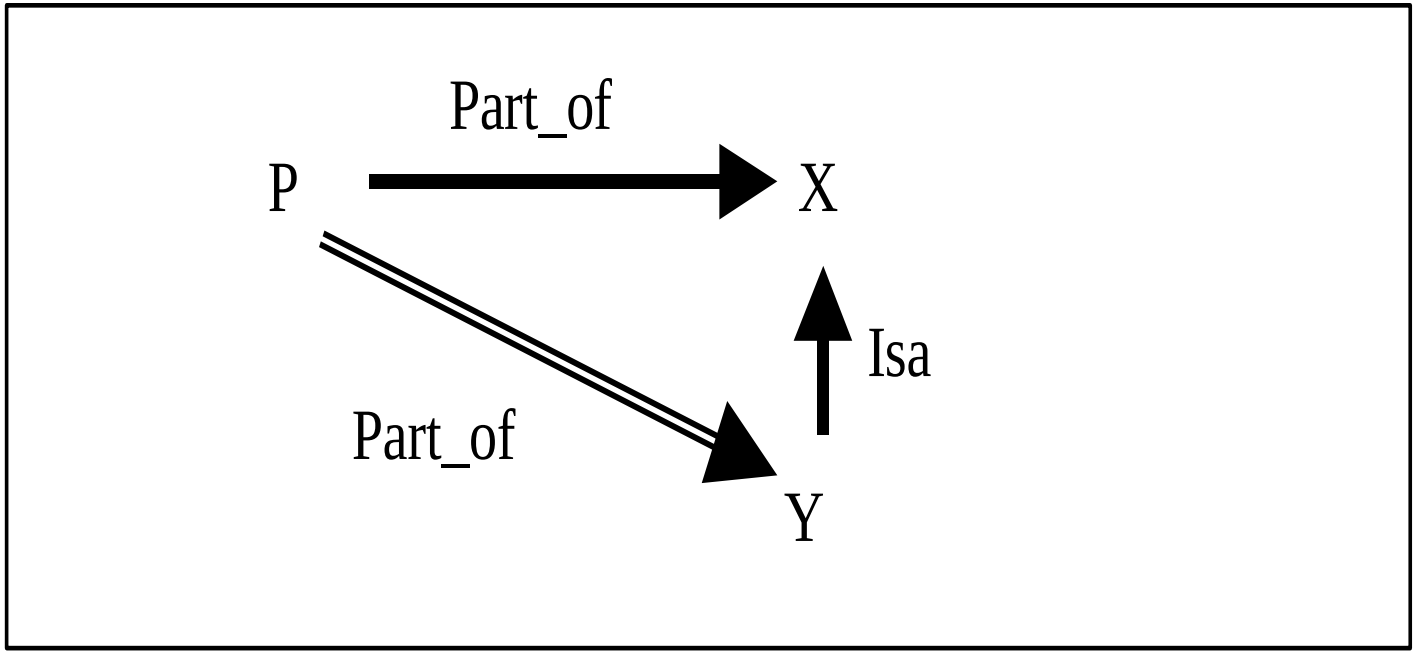
  Action(e, take)

  Actor (e, john)



Contradiction

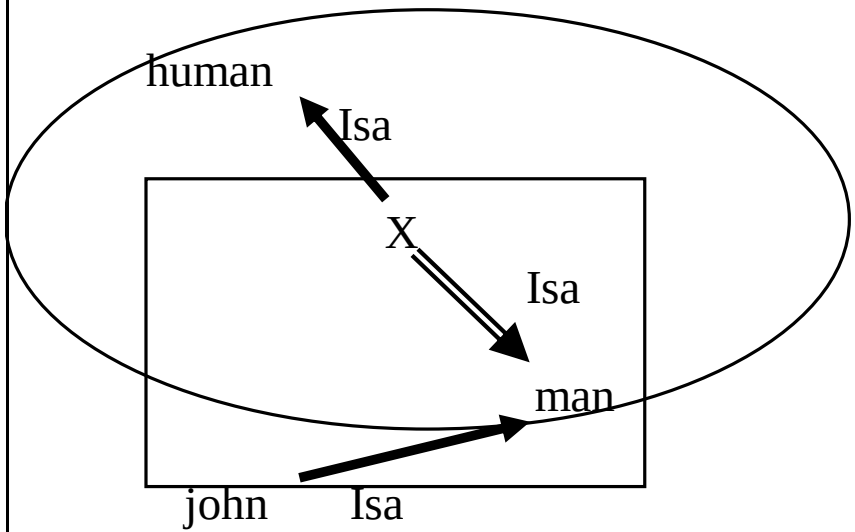
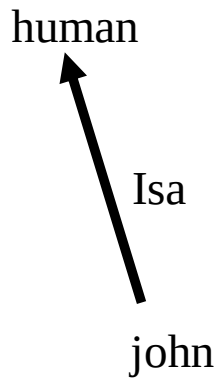
- The contradiction in the ESNet arises if we have the following situation



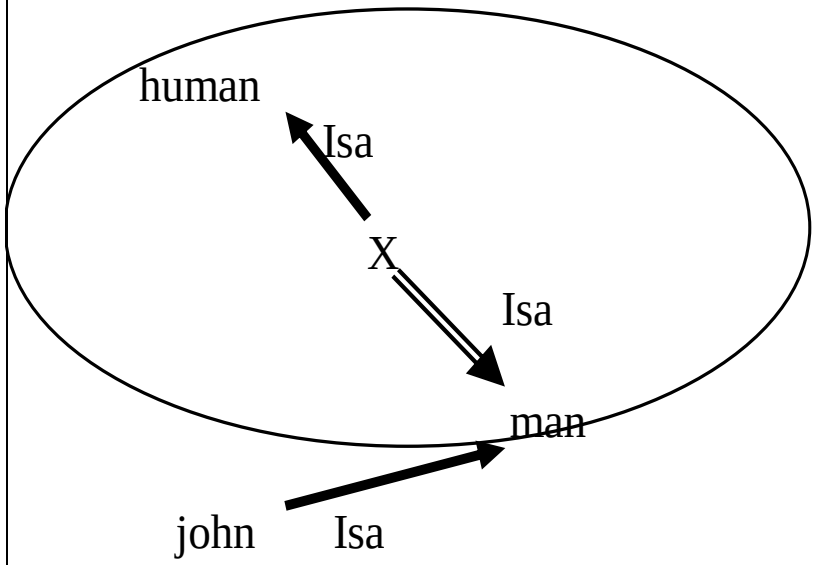
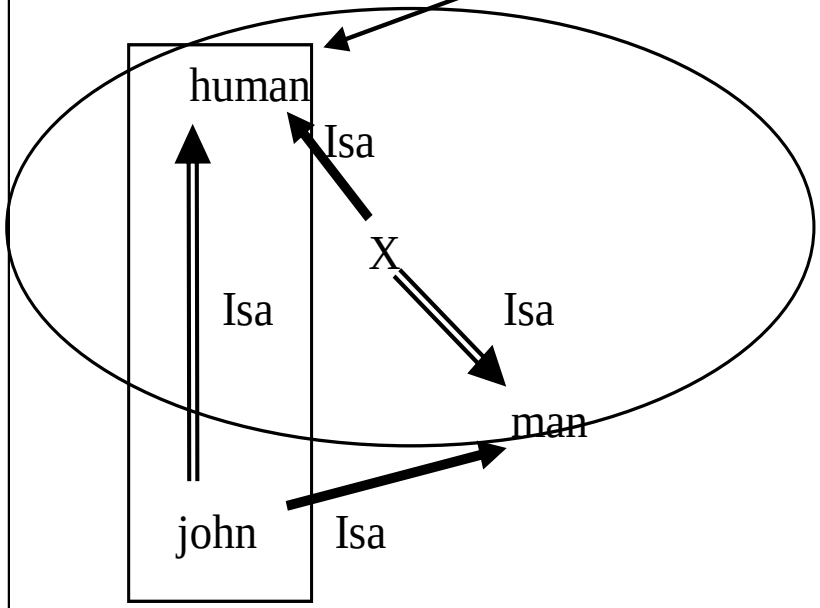
Deduction in ESNet

- Both of the following inference mechanisms are available in ESNet
  - Forward reasoning inference (uses bottom up approach)
    - ***Bottom Up Inferencing:*** Given an ESNet, apply the following reduction (resolution) using Modus Ponens rule of logic ({A ← B, B} then A)
  - Backward reasoning inference (uses top down approach)
    - ***Top Down Inferencing:*** Prove a conclusion from a given ESNet by adding the denial of the conclusion to the network and show that the resulting set of clauses in the network is inconsistent

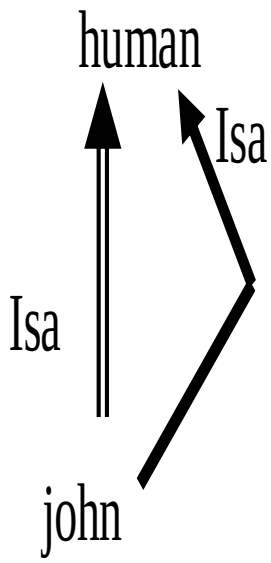
Example: Bottom Up Inferencing

Given set of clauses	**Inferencing**
Isa(X, human) ← Isa(X, man) Isa(john, man).	Isa(john, human)
 Here X is bound to john	

Example: Top Down Inferencing

Given set of clauses	**Prove conclusion**
Isa(X, human) ← Isa(X, man) Isa(john, man).	Query: Isa(john, human) denial of query
	

Cont...



X = john

Contradiction or Empty network is generated. Hence “Isa(john, human)” is proved.

Description Logics

- Description logics provide a family of frame-like KR systems with a formal semantics
  - E.g., KL-ONE, LOOM, Classic, …
- An additional kind of inference done by these systems is automatic classification
  - finding the right place in a hierarchy of objects for a new description
- Current systems take care to keep the languages simple, so that all inference can be done in polynomial time (in the number of objects)
  - ensuring tractability of inference

Description Logics

- Notations to make it easier to describe definitions and properties of categories
- Taxonomic structure is organizing principle
  - Subsumption: Determine if one category is a subset of another
  - Classification: Determine the category in which an object belongs
  - Consistency: Determine if membership criteria are logically satisfiable

Some DL Representations

- DAML+OIL
- OWL
- RDF
- CYCL

Ontologies

- Structuring knowledge in a useful fashion
- An ontology formally represents concepts in a domain and relationships between those concepts
- The concept originated in philosophy; a model of a theory of nature or existence
- An ontology describes the things we want to talk about, including both objects and relationships

Abduction

- **Abduction** is a reasoning process that tries to form plausible explanations for abnormal observations
  - Abduction is distinctly different from deduction and induction
  - Abduction is inherently uncertain
- Uncertainty is an important issue in abductive reasoning
- Some major formalisms for representing and reasoning about uncertainty
  - Mycin's certainty factors (an early representative)
  - **Probability theory (esp. Bayesian belief networks)**
  - Dempster-Shafer theory
  - Fuzzy logic
  - Truth maintenance systems
  - Nonmonotonic reasoning

Abduction

- **Definition** (Encyclopedia Britannica): reasoning that derives an explanatory hypothesis from a given set of facts
  - The inference result is a **hypothesis** that, if true, could **explain** the occurrence of the given facts
- **Examples**
  - Dendral, an expert system to construct 3D structure of chemical compounds
    - Fact: mass spectrometer data of the compound and its chemical formula
    - KB: chemistry, esp. strength of different types of bounds
    - Reasoning: form a hypothetical 3D structure that satisfies the chemical formula, and that would most likely produce the given mass spectrum

Abduction examples (cont.)

- Medical diagnosis
  - Facts: symptoms, lab test results, and other observed findings (called manifestations)
  - KB: causal associations between diseases and manifestations
  - Reasoning: one or more diseases whose presence would causally explain the occurrence of the given manifestations

- Many other reasoning processes (e.g., word sense disambiguation in natural language processing, image understanding, criminal investigation) can also been seen as abductive reasoning

Comparing abduction, deduction, and induction

Deduction: major premise: All balls in the box are black
minor premise: These balls are from the box
conclusion: These balls are black

A => B
A

B

Abduction: rule: All balls in the box are black
observation: These balls are black
explanation: These balls are from the box

A => B
B

Possibly

Induction: case: These balls are from the box
observation: These balls are black
hypothesized rule: All ball in the box are black

Whenever A then B

Possibly

Deduction
Abduction
Induction

Characteristics of abductive reasoning

- "Conclusions" are **hypotheses**, not theorems (may be false *even if* rules and facts are true)
  - E.g., misdiagnosis in medicine

- There may be multiple plausible hypotheses
  - Given rules A => B and C => B, and fact B, both A and C are plausible hypotheses
  - Abduction is inherently uncertain
  - Hypotheses can be ranked by their plausibility (if it can be determined)

Characteristics of abductive reasoning (cont.)

- Reasoning is often a hypothesize-and-test cycle
  - **Hypothesize**: Postulate possible hypotheses, any of which would explain the given facts (or at least most of the important facts)
  - **Test**: Test the plausibility of all or some of these hypotheses
  - One way to test a hypothesis H is to ask whether something that is currently unknown–but can be predicted from H–is actually true
    - If we also know A => D and C => E, then ask if D and E are true
    - If D is true and E is false, then hypothesis A becomes more plausible (**support** for A is increased; **support** for C is decreased)

Characteristics of abductive reasoning (cont.)

- Reasoning is **non-monotonic**
  - That is, the plausibility of hypotheses can increase/decrease as new facts are collected
  - In contrast, deductive inference is **monotonic:** it never change a sentence's truth value, once known
  - In abductive (and inductive) reasoning, some hypotheses may be discarded, and new ones formed, when new observations are made

Sources of uncertainty

- Uncertain **inputs**
  - Missing data
  - Noisy data
- Uncertain **knowledge**
  - Multiple causes lead to multiple effects
  - Incomplete enumeration of conditions or effects
  - Incomplete knowledge of causality in the domain
  - Probabilistic/stochastic effects
- Uncertain **outputs**
  - Abduction and induction are inherently uncertain
  - Default reasoning, even in deductive fashion, is uncertain
  - Incomplete deductive inference may be uncertain

▸ Probabilistic reasoning only gives probabilistic results (summarizes uncertainty from various sources)

Decision making with uncertainty

- **Rational** behavior:
  - For each possible action, identify the possible outcomes
  - Compute the **probability** of each outcome
  - Compute the **utility** of each outcome
  - Compute the probability-weighted **(expected) utility** over possible outcomes for each action
  - Select the action with the highest expected utility (principle of **Maximum Expected Utility**)

Bayesian reasoning

- Probability theory
- Bayesian inference
  - Use probability theory and information about independence
  - Reason diagnostically (from evidence (effects) to conclusions (causes)) or causally (from causes to effects)
- Bayesian networks
  - Compact representation of probability distribution over a set of propositional random variables
  - Take advantage of independence relationships

Other uncertainty representations

- Default reasoning
  - Nonmonotonic logic: Allow the retraction of default beliefs if they prove to be false
- Rule-based methods
  - Certainty factors (Mycin): propagate simple models of belief through causal or diagnostic rules
- Evidential reasoning
  - Dempster-Shafer theory: Bel(P) is a measure of the evidence for P; Bel(¬P) is a measure of the evidence against P; together they define a belief interval (lower and upper bounds on confidence)
- Fuzzy reasoning
  - Fuzzy sets: How well does an object satisfy a vague property?
  - Fuzzy logic: "How true" is a logical statement?

Uncertainty tradeoffs

- **Bayesian networks:** Nice theoretical properties combined with efficient reasoning make BNs very popular; limited expressiveness, knowledge engineering challenges may limit uses
- **Nonmonotonic logic:** Represent commonsense reasoning, but can be computationally very expensive
- **Certainty factors:** Not semantically well founded
- **Dempster-Shafer theory:** Has nice formal properties, but can be computationally expensive, and intervals tend to grow towards [0,1] (not a very useful conclusion)
- **Fuzzy reasoning:** Semantics are unclear (fuzzy!), but has proved very useful for commercial applications

Knowledge Engineering

- Process of representing domain knowledge formally
- Includes several components or phases:
  - Becoming familiar with the domain
  - Choosing a knowledge representation
  - Adding high-level knowledge
  - Adding more detailed knowledge
  - Testing Knowledge
  - Updating and maintaining the knowledge base
- The knowledge engineering bottleneck is a significant problem.

KE: Familiarization -- process of becoming acquainted with *domain* for which the KB is being developed.

- General Domain
  - Typical goals or purpose
  - Training, skills needed
  - Typical sources of knowledge used
  - Typical information gathered
  - May include observations, interviews, training
- Specific Setting
  - Goal or purpose for this organization
  - Typical setup and roles
  - Materials and processes
  - Specific knowledge involved

Typical Familiarization Activities

- Interview sponsors of system: establish goals, scope, identify experts
- Interview experts: get general feel for their activities, clarify result expert system should produce
- Observe setting and experts
- Your goal at this stage is not to start capturing knowledge, it is to learn enough about the domain and the requirements to make some decisions about knowledge representation

Knowledge Representation: *choosing a representation appropriate for the domain*

- Requires familiarity with the general domain
- Desirable characteristics
  - Easy to represent relevant knowledge
  - Doesn't require irrelevant knowledge
  - Easy to integrate new knowledge
  - Can reason about it appropriately and efficiently
  - Reflects the semantics of the domain
- Sometimes hard to assess initially; good to do a small proof of concept of what you choose

Domain Characteristics: Relevant to determining Knowledge Representation

- Kind of reasoning of human experts
- Level of complexity of domain
  - Inputs
  - Outcomes
  - Relationships between inputs and outcomes
- Kinds of knowledge
  - Structure
  - Heuristics
  - Inheritance
- Stability of knowledge

KE: High-Level Knowledge

- High level knowledge is
  - Knowledge about structure of domain
  - Organizing knowledge
  - Initial Information
- Tasks include
  - Preparation
  - Knowledge Acquisition
  - Knowledge Analysis
  - Coding
  - Testing
- Best carried out with sources who have an organized, coherent view of the entire domain, if available

KR: Detailed Knowledge

- More specific knowledge
- Lower level details
- Actual cases
- Expected outcomes for cases
- Steps
  - Preparation
  - Knowledge Acquisition
  - Knowledge Analysis
  - Coding
  - Testing

E: Detailed Knowledge -- Preparation

- Preparing for actual knowledge acquisition steps

- The goal is to be ready to make optimal use of source's time
  - Review Materials
  - Become familiar with terminology
  - Collect sources
  - Training
  - Review any existing adjacent knowledge bases

KE: Detailed Knowledge -- Sources

- Sources can include
  - Textbooks and manuals
  - Reports
  - Databases, empirical data, case studies
  - Human experts
- Issues include
  - Scattered knowledge
  - Multiple sources of knowledge
  - Contradictory knowledge
  - Irrelevant knowledge
- A significant amount of the knowledge available is "noise" for the purposes of a specific system. The knowledge engineer needs skill at screening out the noise and organizing the remainder.

KE: Detailed Knowledge – Books and Manuals

- Usually easy to obtain
- Optimal use of documents typically includes
  - Aid to becoming familiar with terminology and general subject matter
  - Reference between sessions with an expert
  - Good source for diagrams, detailed names, etc
  - Source for online aids built in to expert system
- Documents used extensively by experts may be especially helpful
- Screen carefully – it's easy to get buried.

Knowledge Engineering

- Actually capturing the information from the human subject matter expert (SME) in any of these formats is difficult and time-consuming
  - An iterative process of add knowledge/test.
  - Often a knowledge engineer or ontological engineer works with the SME
  - "What is the system for?" is critical
- Automated learning of knowledge is a very active research field right now.