TFIP-AI - Machine Learning

Unit 3 Logical and Reasoning Systems

Part 3 First-Order Logic

Outline

1. Need for first-order logic
2. Syntax and semantics
3. Planning with FOL
4. Inference with FOL

Pros and Cons of Propositional Logic

- Propositional logic is declarative: pieces of syntax correspond to facts
- Propositional logic allows partial/disjunctive/negated information
  (unlike most data structures and databases)
- 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)
- Propositional logic has very limited expressive power
  (unlike natural language)
  E.g., cannot say "pits cause breezes in adjacent squares"
  except by writing one sentence for each square

Pros and Cons of Propositional Logic

Rules of chess:

- 100,000 pages in propositional logic
- 1 page in first-order logic

Rules of pacman:

- $\forall x,y,t\ At(x,y,t) \Leftrightarrow [At(x,y,t-1) \wedge \neg\exists\ u,v\ Reachable(x,y,u,v,Action(t-1))]\ \vee$
  $[\exists\ u,v\ At(u,v,t-1) \wedge Reachable(x,y,u,v,Action(t-1))]$

First-Order Logic (First-Order Predicate Calculus)

Whereas propositional logic assumes world contains facts,
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, …
- Functions: father of, best friend, third inning of, one more than, end of, …

Logics in General

Language	What exists in the world	What an agent believes about facts
Propositional logic	Facts	true / false / unknown
First-order logic	facts, objects, relations	true / false / unknown
Probability theory	facts	degree of belief
Fuzzy logic	facts + degree of truth	known interval value

Syntax of FOL

Basic Elements

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

Syntax of FOL

Atomic sentence = $predicate(term_1, \ldots, term_n)$
or $term_1 = term_2$

Term = $function(term_1, \ldots, term_n)$
or $constant$
or $variable$

Examples

$Brother(KingJohn, RichardTheLionheart)$

$> (Length(LeftLegOf(Richard)),\ Length(LeftLegOf(KingJohn)))$

Syntax of FOL

Complex sentences are made from atomic sentences using connectives

$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2$

Examples

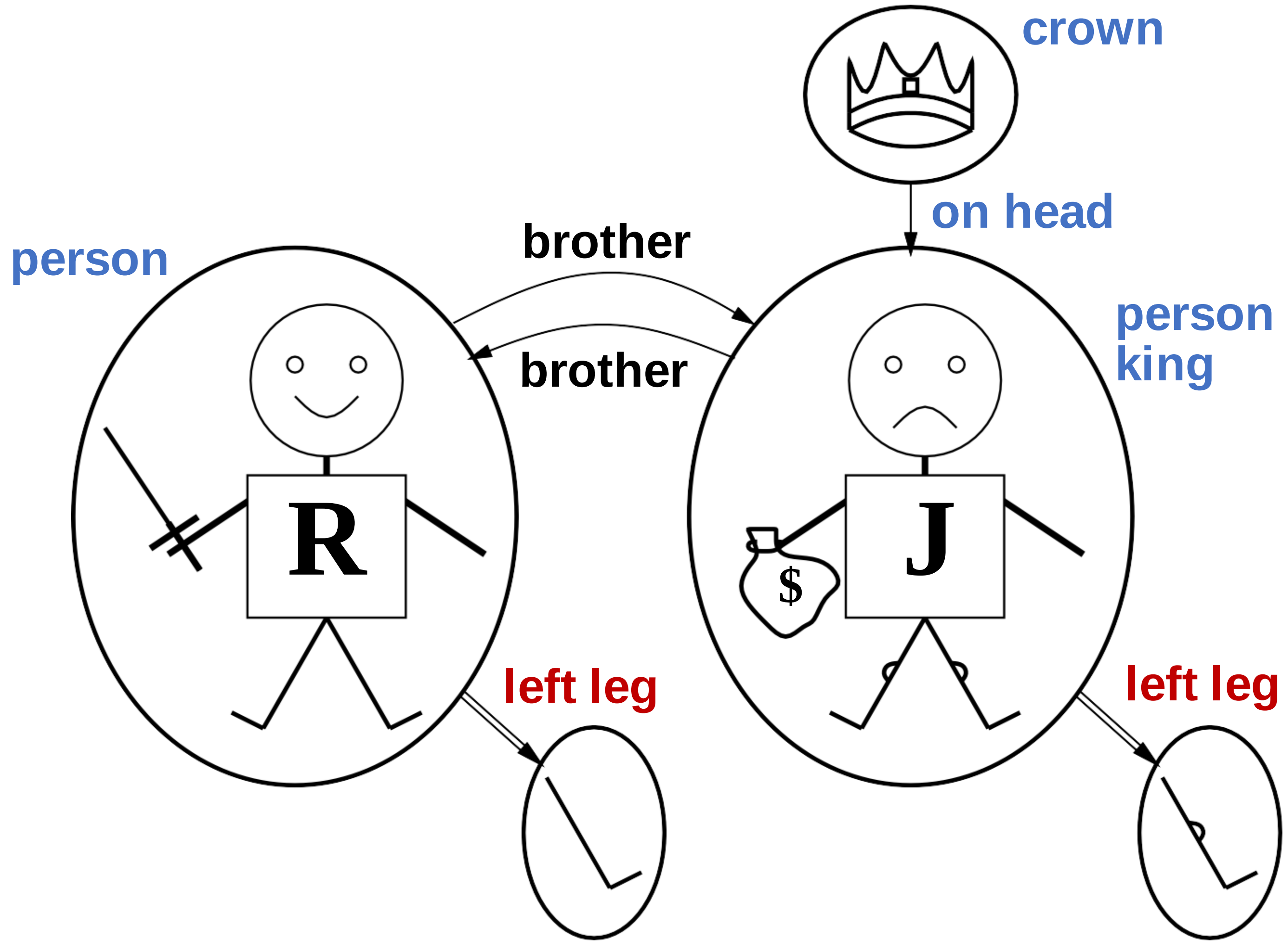
$Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$

$>(1, 2) \vee \leq(1, 2)$

$>(1, 2) \wedge \neg >(1, 2)$

Models for FOL

Example

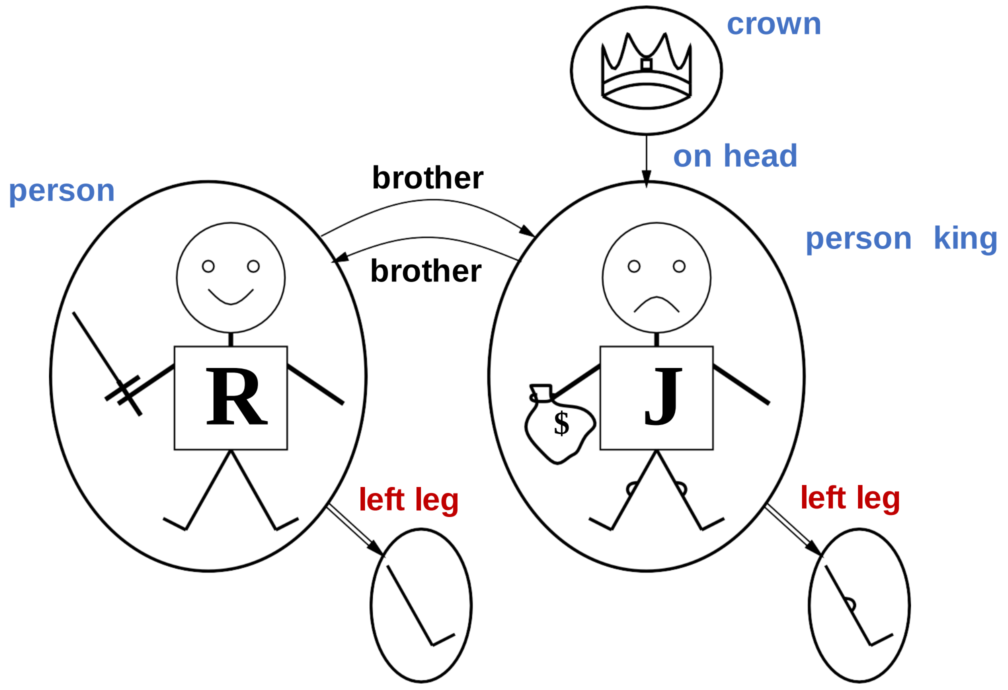


Models for FOL

Brother(Richard, John)

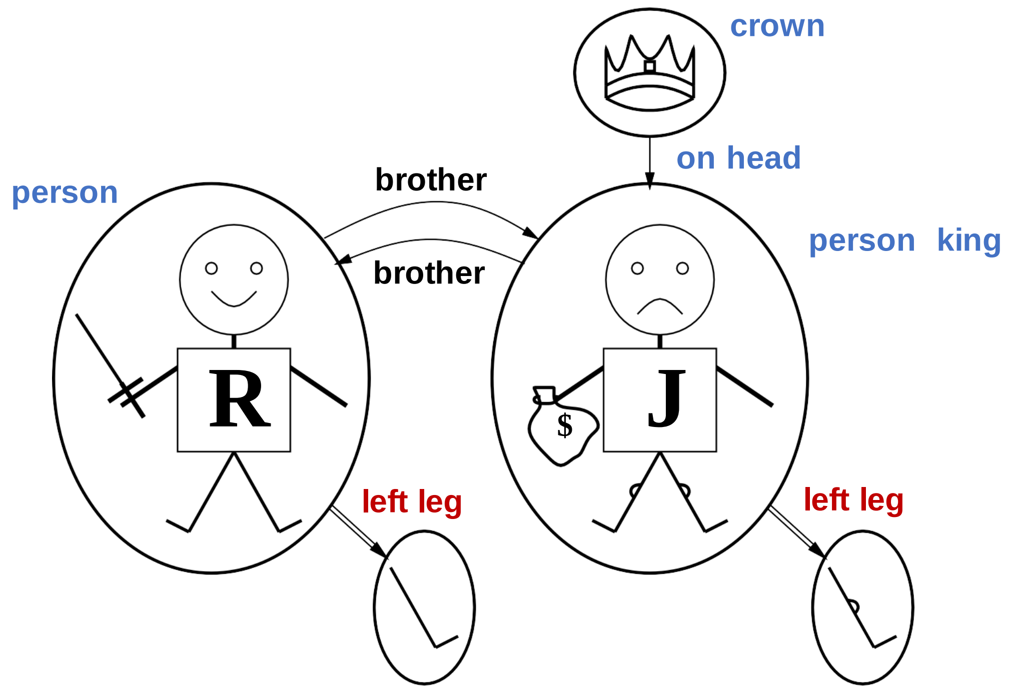
Consider the interpretation in which:

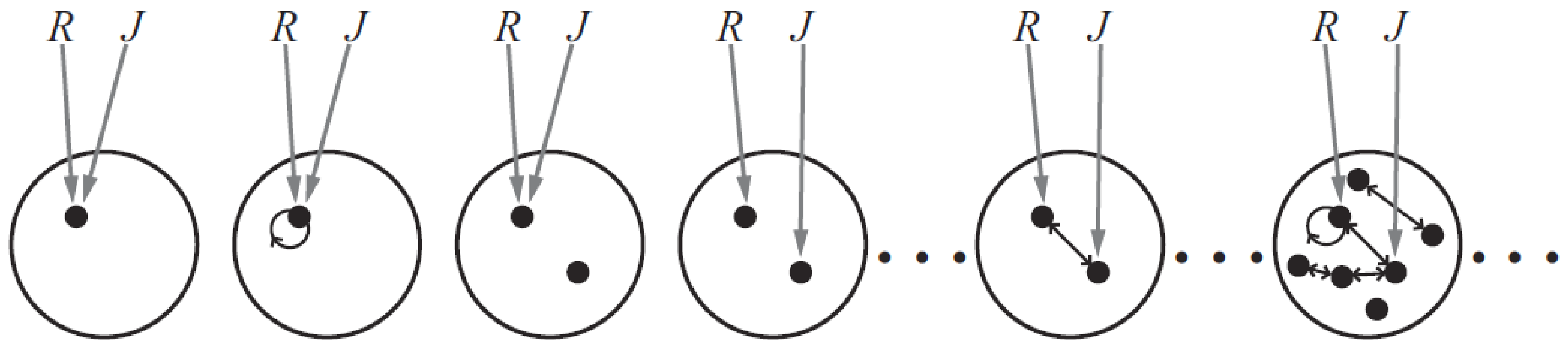
Richard → Richard the Lionheart
John → the evil King John
Brother → the brotherhood relation



Model for FOL

Lots of models!





Model for FOL

Lots of models!

Entailment in propositional logic can be computed by enumerating models

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

```
For each number of domain elements $n$ from 1 to $\infty$
  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 FOL models is not easy!

Truth in First-Order Logic

Sentences are true with respect to a model and an interpretation

Model contains $\geq$ 1 objects (domain elements) and relations among them

Interpretation specifies referents for

constant symbols $\rightarrow$ objects
predicate symbols $\rightarrow$ relations
function symbols $\rightarrow$ functional relations

An atomic sentence $predicate(term_1, \ldots, term_n)$ is true:

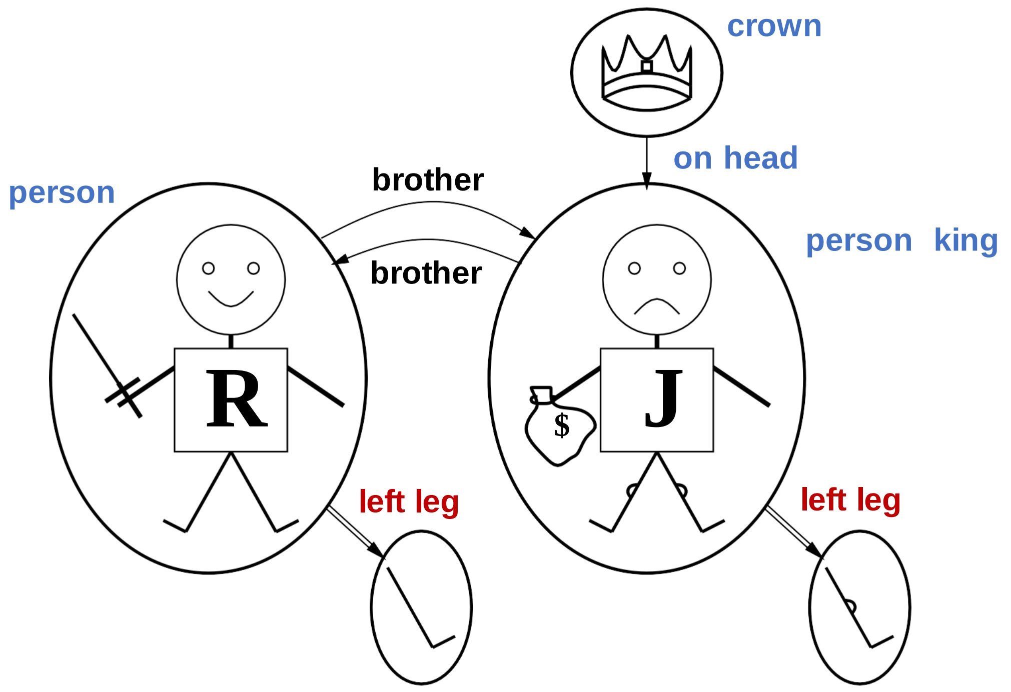
iff the objects referred to by $term_1, \ldots, term_n$

are in the relation referred to by $predicate$

Models for FOL

Consider the interpretation in which:

Richard → Richard the Lionheart
John → the evil King John
Brother → the brotherhood relation



Under this interpretation, *Brother*(*Richard*, *John*) is true just in the case Richard the Lionheart and the evil King John are in the brotherhood relation in the model

Universal Quantification

$\forall \langle variables \rangle \quad \langle sentence \rangle$

Everyone at the banquet is hungry:

$\forall x \quad At(x, Banquet) \Rightarrow Hungry(x)$

$\forall x \quad P$ 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

$$\begin{aligned}
&(At(KingJohn, Banquet) \Rightarrow Hungry(KingJohn)) \\
\wedge\ &(At(Richard, Banquet) \Rightarrow Hungry(Richard)) \\
\wedge\ &(At(Banquet, Banquet) \Rightarrow Hungry(Banquet)) \\
\wedge\ &\ldots
\end{aligned}$$

Universal Quantification

Common mistake

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

Common mistake: using $\wedge$ as the main connective with $\forall$:

$$\forall x\, At(x,\ Banquet) \wedge Hungry(x)$$

means "Everyone is at the banquet and everyone is hungry"

Existential Quantification

$\exists\,(variables) \qquad (sentence)$

Someone at the tournament is hungry:
$\exists\, x At(x, Tournament) \land Hungry(x)$

$\exists\, x P$ 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

$$
\begin{aligned}
&(At(KingJohn, Tournament) \land Hungry(KingJohn))\\
\lor\ &(At(Richard, Tournament) \land Hungry(Richard))\\
\lor\ &(At(Tournament, Tournament) \land Hungry(Tournament))\\
\lor\ &\ldots
\end{aligned}
$$

Existential Quantification

Common mistake

Typically, $\wedge$ is the main connective with $\exists$

Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists x At(x, Tournament) \Rightarrow Hungry(x)$$

is true if there is anyone who is not at the tournament!

Properties of Quantifiers

$\forall x \quad \forall y$ is the same as $\forall y \quad \forall x$

$\exists x \quad \exists y$ is the same as $\exists y \quad \exists x$

$\exists x \quad \forall y$ 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

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

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

Fun with Sentences

Brothers are siblings

$$\forall x, y\, Brother(x, y) \Rightarrow Sibling(x, y).$$

"Sibling" is symmetric

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

A first cousin is a child of a parent's sibling

$$\forall x, y\, FirstCousin(x, y) \Leftrightarrow \exists\, p, ps\;\; Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$$

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., $1 = 2$ and $\forall x \ \times(Sqrt(x), Sqrt(x)) = x$ are satisfiable
$2 = 2$ is valid

E.g., definition of (full) $Sibling$ in terms of $Parent$:

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

What is the answer for this?

Given the following two FOL sentences:

$$\gamma: \quad \forall\, x \quad Hungry(x)$$

$$\delta: \quad \exists\, x \quad Hungry(x)$$

Which of these is true?

A) $\gamma \vDash \delta$

B) $\delta \vDash \gamma$

C) Both

D) Neither

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 $t = 5$:

$Tell(KB, Percept([Smell, Breeze, None], 5))$
$Ask(KB, \exists a\ Action(a, 5))$

i.e., does KB entail any particular actions at $t = 5$?

Answer: $Yes, \{a/Shoot\}$ $\leftarrow$ substitution (binding list)

Notation Alert!

Given a sentence S and a substitution σ,
$S\sigma$ denotes the result of plugging σ into S; e.g.,
$S = Smarter(x, y)$
$\sigma = \{x/EVE, y/WALL\text{-}E\}$
$S\sigma = Smarter(EVE, WALL\text{-}E)$

Notation Alert!

$Ask(KB, S)$ returns some/all σ such that $KB \models S\sigma$

Knowledge Base for Wumpus World

"Perception"

$\forall\, b, g, t \quad Percept([Smell, b, g], t) \Rightarrow Smelt(t)$

$\forall\, s, b, t \quad Percept([s, b, Glitter], t) \Rightarrow AtGold(t)$

Reflex: $\forall\, t \quad AtGold(t) \Rightarrow Action(Grab, t)$

Reflex with internal state: do we have the gold already?

$\forall\, t \quad AtGold(t) \wedge \neg Holding(Gold, t) \Rightarrow Action(Grab, t)$

$Holding(Gold, t)$ cannot be observed

$\Rightarrow$ keeping track of change is essential

Deducing Hidden Properties

Properties of locations:

$\forall x, t \quad At(Agent, x, t) \wedge Smelt(t) \Rightarrow Smelly(x)$

$\forall x, t \quad At(Agent, x, t) \wedge Breeze(t) \Rightarrow Breezy(x)$

Squares are breezy near a pit:

Diagnostic rule—infer cause from effect

$$\forall y \quad Breezy(y) \Rightarrow \exists x \quad Pit(x) \wedge Adjacent(x, y)$$

Causal rule—infer effect from cause

$$\forall x, y \quad Pit(x) \wedge Adjacent(x, y) \Rightarrow Breezy(y)$$

Neither of these is complete — e.g., the causal rule doesn't say whether squares far away from pits can be breezy

Definition for the *Breezy* predicate:

$$\forall y \quad Breezy(y) \Leftrightarrow [\exists x \quad Pit(x) \wedge Adjacent(x, y)]$$

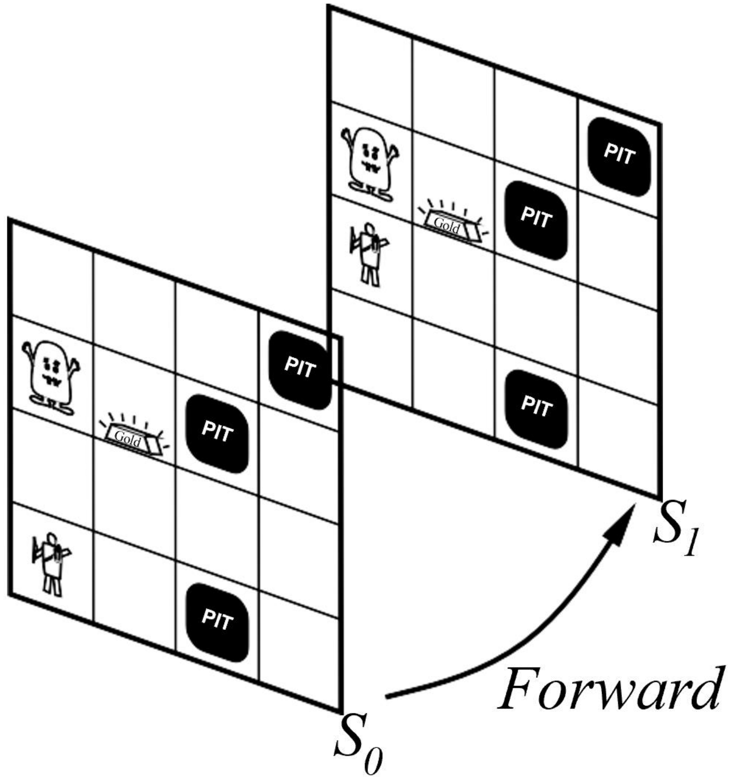
Keeping Track of Change

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

"Effect" axiom—describe changes due to action

$\forall s \quad AtGold(s) \Rightarrow \quad Holding(Gold, Result(Grab, s))$

"Frame" axiom—describe non-changes due to action

$\forall s \quad HaveArrow(s) \Rightarrow \quad HaveArrow(Result(Grab, s))$

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:

$$\begin{aligned} \forall a, s \quad & Holding(Gold, Result(a, s)) \Leftrightarrow \\ & [(a = Grab \wedge AtGold(s)) \\ & \vee (Holding(Gold, s) \wedge \neg(a = Release))] \end{aligned}$$

Describing Actions

Initial condition in KB:

$At(Agent, [1,1], S_0)$
$At(Gold, [1,2], S_0)$

Query: $Ask(KB, \exists\, s\ Holding(Gold, s))$

i.e., in what situation will I be holding the gold?

Answer: $\{s/Result(Grab, Result(Forward, S_0))\}$

i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at S_0 and that S_0 is the only situation described in the KB

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 $Result$:

$\forall s \quad PlanResult([\,], s) = s$

$\forall a, p, s \quad PlanResult([a, p], s) = PlanResult(p, Result(a, s))$

Outline

1. Need for first-order logic
2. Syntax and semantics
3. Planning with FOL
4. Inference with FOL

Inference in First-Order Logic

A) Reducing first-order inference to propositional inference

- Removing ∀
- Removing ∃
- Unification

B) *Lifting* propositional inference to first-order inference

- Generalized Modus Ponens
- FOL forward chaining

Universal Instantiation

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

$$\forall v\; a$$

$$\mathrm{Subst}(\{v/g\}, a)$$

for any variable v and ground term g

E.g., $\forall x \quad King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ 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

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

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

E.g., $\exists x \quad Crown(x) \land OnHead(x, John)$ yields

$Crown(C_1) \land OnHead(C_1, John)$

provided C_1 is a new constant symbol, called a Skolem constant

First Order Logic

The De Morgan rules for quantified and unquantified sentences are as follows:

$$\begin{array}{lcl} \forall x \;\; \neg P & \equiv & \neg \exists x \;\; P \\ \neg \forall x \;\; P & \equiv & \exists x \;\; \neg P \\ \forall x \;\; P & \equiv & \neg \exists x \;\; \neg P \\ \exists x \;\; P & \equiv & \neg \forall x \;\; \neg P \end{array}$$

$$\begin{array}{lcl} \neg(P \vee Q) & \equiv & \neg P \wedge \neg Q \\ \neg(P \wedge Q) & \equiv & \neg P \vee \neg Q \\ P \wedge Q & \equiv & \neg(\neg P \vee \neg Q) \\ P \vee Q & \equiv & \neg(\neg P \wedge \neg Q) \end{array}$$

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.

Reduction to Propositional Inference

Claim: a ground sentence* is entailed by new KB iff entailed by original KB

Claim: every FOL KB can be propositionalized so as to preserve entailment

Idea: propositionalize KB and query, apply resolution, return result Problem: with function symbols,

there are infinitely many ground terms,

e.g., $Father(Father(Father(John)))$

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

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

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

Theorem: Turing (1936), Church (1936), entailment in FOL is semidecidable

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

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 $Greedy(y)$

$\theta = \{x/John,\ y/John\}$ works

$\mathrm{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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., $Knows(z_{17}, OJ)$

Generalized Modus Ponens (GMP)

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

where $p_i{}^t\theta = p_i\theta$ for all i

$p_1{}^t$ is $King(John)$ p_1 is $King(x)$
$p_2{}^t$ 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

FOL Forward Chaining

```
function FOL-FC-Ask($KB$, $\alpha$) 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 $\theta$ 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($\theta, q$)
                if $q'$ is not a renaming of a sentence already in $KB$ or $new$ then do
                    add $q'$ to $new$
                    $\varphi \leftarrow$ Unify($q', \alpha$)
                    if $\varphi$ is not $fail$ then return $\varphi$
        add $new$ to $KB$
    return $false$
```

Knowledge Engineering Introduction

Knowledge engineering includes the following steps:

1. Identify the task
2. Assemble the relevant knowledge (Knowledge acquisition)
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

Knowledge Engineering Example

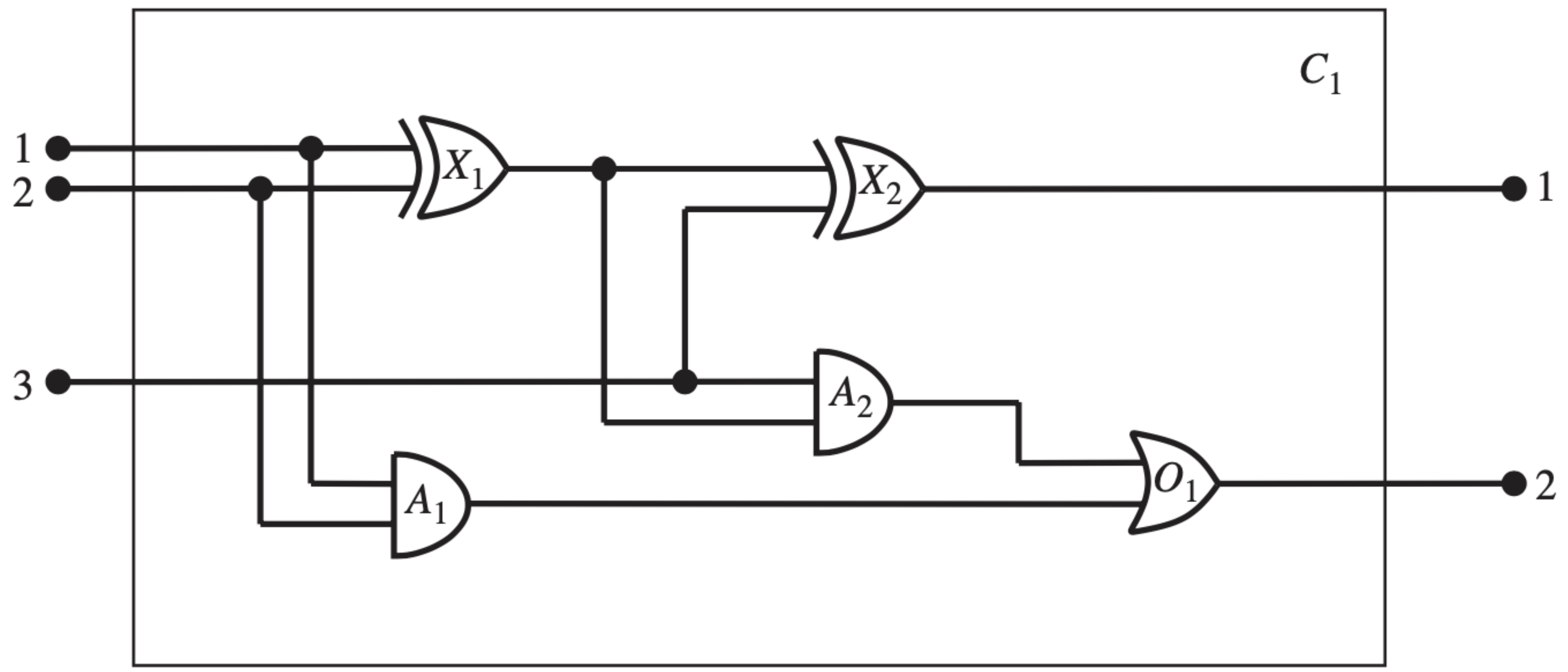


Figure: A digital circuit C1, purporting to be a one-bit full adder. The first two inputs are the two bits to be added, and the third input is a carry bit. The first output is the sum, and the second output is a carry bit for the next adder. The circuit contains two XOR gates, two AND gates, and one OR gate.

Knowledge Engineering Example cont...

The axioms we need are as follows:

1. If two terminals are connected, then they have the same signal:
$$\forall t_1, t_2 \;\; Terminal(t_1) \land Terminal(t_2) \land Connected(t_1, t_2) \;\Rightarrow\; Signal(t_1) = Signal(t_2)\,.$$
2. The signal at every terminal is either 1 or 0:
$$\forall t \;\; Terminal(t) \;\Rightarrow\; Signal(t) = 1 \lor Signal(t) = 0\;.$$
3. Connected is commutative:
$$\forall t_1, t_2 \;\; Connected(t_1, t_2) \;\Leftrightarrow\; Connected(t_2, t_1)\,.$$
4. There are four types of gates:
$$\forall g \;\; Gate(g) \land k = Type(g) \;\Rightarrow\; k = AND \lor k = OR \lor k = XOR \lor k = NOT\,.$$
5. An AND gate's output is 0 if and only if any of its inputs is 0:
$$\forall g \;\; Gate(g) \land Type(g) = AND \;\Rightarrow\; Signal(Out(1, g)) = 0 \;\Leftrightarrow\; \exists n \;\; Signal(In(n, g)) = 0\,.$$

Knowledge Engineering Example cont...

The axioms we need are as follows cont...:

6. An OR gate's output is 1 if and only if any of its inputs is 1:

$$\forall g \;\; Gate(g) \land Type(g) = OR \;\Rightarrow$$
$$Signal(Out(1,g)) = 1 \;\Leftrightarrow\; \exists n \;\; Signal(In(n,g)) = 1 \,.$$

7. An XOR gate's output is 1 if and only if its inputs are different:

$$\forall g \;\; Gate(g) \land Type(g) = XOR \;\Rightarrow$$
$$Signal(Out(1,g)) = 1 \;\Leftrightarrow\; Signal(In(1,g)) \neq Signal(In(2,g)) \,.$$

8. A NOT gate's output is different from its input:

$$\forall g \;\; Gate(g) \land (Type(g) = NOT) \;\Rightarrow$$
$$Signal(Out(1,g)) \neq Signal(In(1,g)) \,.$$

9. The gates (except for NOT) have two inputs and one output.

$$\forall g \;\; Gate(g) \land Type(g) = NOT \;\Rightarrow\; Arity(g,1,1) \;.$$
$$\forall g \;\; Gate(g) \land k = Type(g) \land (k = AND \lor k = OR \lor k = XOR) \;\Rightarrow$$
$$Arity(g,2,1)$$

Knowledge Engineering Example cont...

The axioms we need are as follows cont...:

10. A circuit has terminals, up to its input and output arity, and nothing beyond its arity:

$$\begin{aligned}&\forall\, c,i,j\;\; Circuit(c) \land Arity(c,i,j) \;\Rightarrow\\ &\quad \forall\, n\;\; (n \le i \;\Rightarrow\; Terminal(In(c,n))) \land (n > i \;\Rightarrow\; In(c,n) = Nothing) \land\\ &\quad \forall\, n\;\; (n \le j \;\Rightarrow\; Terminal(Out(c,n))) \land (n > j \;\Rightarrow\; Out(c,n) = Nothing)\end{aligned}$$

11. Gates, terminals, signals, gate types, and *Nothing* are all distinct.

$$\begin{aligned}&\forall\, g,t\;\; Gate(g) \land Terminal(t) \;\Rightarrow\\ &\quad g \ne t \ne 1 \ne 0 \ne OR \ne AND \ne XOR \ne NOT \ne Nothing\,.\end{aligned}$$

12. Gates are circuits.

$$\forall\, g\;\; Gate(g) \;\Rightarrow\; Circuit(g)$$

Knowledge Engineering Example cont...

We categorize the circuit and its component gates as follows:

$$Circuit(C_1) \land Arity(C_1, 3, 2)$$
$$Gate(X_1) \land Type(X_1) = XOR$$
$$Gate(X_2) \land Type(X_2) = XOR$$
$$Gate(A_1) \land Type(A_1) = AND$$
$$Gate(A_2) \land Type(A_2) = AND$$
$$Gate(O_1) \land Type(O_1) = OR\,.$$

Knowledge Engineering Example cont...

The connections between them are as follows:

$$\begin{array}{ll}
Connected(Out(1,X_1),In(1,X_2)) & Connected(In(1,C_1),In(1,X_1)) \\
Connected(Out(1,X_1),In(2,A_2)) & Connected(In(1,C_1),In(1,A_1)) \\
Connected(Out(1,A_2),In(1,O_1)) & Connected(In(2,C_1),In(2,X_1)) \\
Connected(Out(1,A_1),In(2,O_1)) & Connected(In(2,C_1),In(2,A_1)) \\
Connected(Out(1,X_2),Out(1,C_1)) & Connected(In(3,C_1),In(2,X_2)) \\
Connected(Out(1,O_1),Out(2,C_1)) & Connected(In(3,C_1),In(1,A_2))
\end{array}$$

Knowledge Engineering Example cont...

Pose queries to the inference procedure (circuit verification).

What combinations of inputs would cause the first output of C1 (the sum bit) to be 0 and the second output of C1 (the carry bit) to be 1?

$$\exists\, i_1, i_2, i_3 \;\; Signal(In(1, C_1)) = i_1 \wedge Signal(In(2, C_1)) = i_2 \wedge Signal(In(3, C_1)) = i_3$$
$$\wedge\, Signal(Out(1, C_1)) = 0 \wedge Signal(Out(2, C_1)) = 1 \; .$$

ASKVARS will give us three such substitutions.

$$\{i_1/1,\ i_2/1,\ i_3/0\} \quad \{i_1/1,\ i_2/0,\ i_3/1\} \quad \{i_1/0,\ i_2/1,\ i_3/1\}$$

Knowledge Engineering Example cont...

Pose queries to the inference procedure (circuit verification) cont...

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

$$\begin{aligned} \exists\, i_1, i_2, i_3, o_1, o_2 \;\; & Signal(In(1, C_1)) = i_1 \land Signal(In(2, C_1)) = i_2 \\ & \land Signal(In(3, C_1)) = i_3 \land Signal(Out(1, C_1)) = o_1 \land Signal(Out(2, C_1)) = o_2 \end{aligned}$$