

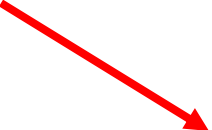
RN, Chapter 8



Predicate Calculus



Logical Agents

- Reasoning [Ch 6]
 - Propositional Logic [Ch 7]
 - Predicate Calculus
 - Representation [Ch 8]
 - Syntax, Semantics, Expressiveness
 - Example: Circuits
 - Inference [Ch 9]
 - Resolution
 - Implemented Systems [Ch 10]
 - Planning [Ch 11]
- 

Propositional Logic vs Predicate Calculus

- “Don't go forward if Wumpus is in front of you”

- $At_{11} \& East \& W_{12} \Rightarrow \neg Forward$

- $At_{12} \& East \& W_{13} \Rightarrow \neg Forward$

- ...

- $At_{34} \& East \& W_{44} \Rightarrow \neg Forward$

- $At_{11} \& North \& W_{12} \Rightarrow \neg Forward$

- ...

Requires ≈ 64 rules (4 x 4 positions, 4 orientations)

- Time!

Currently $A_{1,1}$ means Agent @ [1,1]

Now suppose agent moves: $A_{1,1}$ was true, not anymore!

\Rightarrow Need to have $A_{i,j}^k$ for agent @ [i, j], at time k

- If game has 100 moves

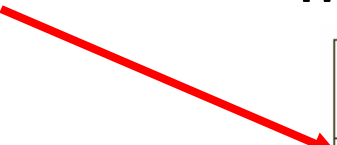
\Rightarrow need ≈ 6400 different rules, just for

“Don't go forward if Wumpus is in front of you” !

- In predicate calculus, only 1 !

Types of Logic

- Logics are characterized by what they commit to as “primitives”
- Ontological commitment:
What exists: facts? objects? time? beliefs?
- Epistemological commitment:
What states of knowledge?



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

Prop. Logic vs Predicate Calculus

- **Propositional Logic:** World consists of propositions . . . either true in world, or not.

$W_{1,2}$

$W_{1,1} \vee W_{1,2} \vee \dots \vee W_{4,4}$

$\neg(W_{1,1} \& W_{1,2}) \& \neg(W_{1,1} \& W_{1,3}) \& \dots \&$

Note: " $W_{1,2}$ " is (syntactically) $(W_{1,1} \& W_{1,3}) \& \dots \& (W_{4,4})$ to " $W_{1,3}$ "

- **Predicate Calculus**

World consists of

{ Objects, Predicates, Properties, relations, functions }

(which particular propositions, that could be true in world)

$At(Wumpus, r)$

$\exists r (At(Wumpus, r) \& \dots)$

$\forall r_1, r_2 (At(Wumpus, r_1) \& At(Wumpus, r_2)) \Rightarrow r_1 = r_2$

⇒ Predicate calculus is MORE POWERFUL than Propositional Logic



Parts of Predicate Calculus

- **Objects:** things w/ individual identity
(*people, houses, numbers, theories, colors, wars, Ronald McDonald, ...*)
- **Predicates:** distinguish objects from one another
 - **properties** (of single object)
(*red, round, bogus, prime, . . .*)
 - **relations** among sets of objects
(*brother-of, part-of, has-color, owns, . . .*)
 - **functions:** rel'n w/ single "value" for each input
(*father-of, one-more-than, . . .*)Each predicate \approx SET of object-tuples:
 - Red = { *rose#1, blood#7, ruby#109, . . .* }
 - Brother = { *< Fred, Sam >, < Tom, Mark >, . . .* }
 - Successor = { *< 1, 2 >, < 12, 13 >, . . .* }
- NEUTRAL: can describe categories, time, ... but nothing is built in...



Predicate Calculus: Syntax

- Atomic Propositions

“basic statements about world”

$At(Wumpus, [3, 4])$, $Adjacent([3; 4], r_2)$, $Smelly([3, 5])$

... built from ...

- Predicate Symbols: At , $Adjacent$, $=$, ...
- Constant Symbols: $Agent$, $Wumpus$, Pit , S_1 , $[3; 5]$, ...
- Variable Symbols: s , a , t , ...
- Function Symbols: $Turn$, $Result$, $Next$, ...

- Well-formed atomic proposition is

$P(t_1, \dots, t_n)$

where P is Predicate, of arity n , &

Each t_i is a term (representing object) is ...

- constant
- variable
- functional term: $F(t_1, \dots, t_m)$
where F is a Function, of arity m ; and each t_i is a term.



Examples of Atomic Propositions

- $\text{Stench}(t_2)$
 - Stench has arity 1
 - t_2 is term as it is a constant

- $\text{At}(\text{Wumpus}, r_2)$
 - At has arity 2
 - Wumpus is term as it is a constant
 - r_2 is term as it is variable

- $\text{AgentAt}(r_2, \text{Next}(S_0))$
 - AgentAt has arity 2;
 - r_2 is term as it is a variable
 - $\text{Next}(S_0)$ is term as Next is a function of 1 arg
and S_0 is term as it is a constant

Logical Connectives

- Build sentences from atomic prop's using:

- connectives:

\wedge	\vee	\neg	\Rightarrow	\Leftrightarrow
and	or	not	implies if... then	equivalence (biconditional)

- quantifiers:

\forall	\exists
for all	exists

- Examples

$\text{Adjacent}([1, 2], [1, 3])$

$\text{At}(\text{Wumpus}, [1, 3]) \vee \text{At}(\text{Wumpus}, [2, 2])$

$\forall r_1, r_2 \text{ At}(\text{Wumpus}, r_1) \& \text{Adjacent}(r_1, r_2) \Rightarrow \text{Smelly}(r_2)$

$\forall r_1, r_2 \text{ Pit}(r_1) \& \text{Adjacent}(r_1, r_2) \Rightarrow \text{Breezy}(r_2)$



BNF Form – Atomic Propositions

- $\langle \text{AtomicProp} \rangle ::= \langle \text{PredSym} \rangle (\langle \text{Term} \rangle^*)$
- $\langle \text{Term} \rangle ::= \langle \text{Constant} \rangle \mid \langle \text{Var} \rangle$
 $\mid \langle \text{FunctionalExpr} \rangle$
- $\langle \text{FunctionalExpr} \rangle ::= \langle \text{FnSym} \rangle (\langle \text{Term} \rangle^*)$
- $\langle \text{Constant} \rangle ::=$ string of letters/numbers starting with UPPER case
- $\langle \text{Var} \rangle ::=$ string of letters/numbers starting with LOWER case
- ... actually need “,”s : $(\langle \text{Term} \rangle, \langle \text{Term} \rangle)$

BNF Form – Propositions

- $\langle \text{Prop} \rangle ::= \langle \text{AtomicProp} \rangle \mid \neg(\langle \text{Prop} \rangle)$
 - | $(\langle \text{Prop} \rangle \wedge \langle \text{Prop} \rangle)$
 - | $(\langle \text{Prop} \rangle \vee \langle \text{Prop} \rangle)$
 - | $(\langle \text{Prop} \rangle \Rightarrow \langle \text{Prop} \rangle)$
 - | $(\langle \text{Prop} \rangle \Leftrightarrow \langle \text{Prop} \rangle)$
 - | $\forall \langle \text{Var} \rangle (\langle \text{Prop} \rangle)$
 - | $\exists \langle \text{Var} \rangle (\langle \text{Prop} \rangle)$

- In practice, can often omit “()”s

Predicate Calculus: Semantics

- Proposition Logic: Model m_f based on

$$f: \text{Var} \mapsto \{T, F\}$$

$$f = \left\{ \begin{array}{cccc} A & B & C & D \\ + & 0 & 0 & + \end{array} \right\}$$

$$m_f \models A, \quad m_f \models B \vee \neg C,$$

- Predicate Calculus Models** ... based on Extensions of Objects, Relations, Functions

$$f: \text{Symbol} \mapsto \text{Extension}$$

$$\begin{aligned}
 f(\text{"RG"}) &= \text{[Portrait of RG]} & f(\text{"JS"}) &= \text{[Portrait of JS]} & \dots \\
 f(\text{"Prof"}) &= \left\{ \text{[Portrait of RG]}, \text{[Portrait of JS]}, \text{[Stick Figure DL]}, \text{[Stick Figure TM]}, \dots \right\} \\
 f(\text{"Person"}) &= \left\{ \text{[Portrait of RG]}, \text{[Stick Figure JvR]}, \dots \right\}
 \end{aligned}$$

Semantics...


$$f(\text{"Red"}) = \{ \text{[cork]}, \text{[fire truck]}, \dots \}$$

$$f(\text{"Taller"}) = \left\{ \begin{array}{l} \langle \text{[fire truck]}, \text{[man with glasses]} \rangle, \\ \langle \text{[man with glasses]}, \text{[man with glasses]} \rangle, \\ \langle \text{[fire truck]}, \text{[man with glasses]} \rangle, \\ \langle \text{[man with glasses]}, \text{[cork]} \rangle, \\ \dots \end{array} \right\}$$

Using Extension

Q: Given extension $f(\cdot)$
is $m_f \models^? Prof(RG)$?









A: True as
 $f("RG") \in f("Prof")$





 $\in \left\{ \begin{array}{l} \text{Portrait of a man} \\ \text{Portrait of a man with glasses} \\ \text{Stick figure with DL} \\ \text{Stick figure with TM} \\ \dots \end{array} \right\}$

YES!

How Many Extensions?

- In Proposition logic...
if 7 symbols, 2^7 models
- In Predicate Calculus...
Suppose 7 objects in universe
 - 4 constants:
 $\{r, j, t, f\}$

	r	j	t	f
f_1				
f_{273}				

	r	j	t	f
f_2				



...

...

How Many Extensions?

- In Proposition logic...
if 7 symbols, 2^7 models
 - In Predicate Calculus...
Suppose 7 objects in universe
 - 4 constants: 7^4
{ r, j, t, f }
 - 3 unary predicates: $(2^7)^3$
{ prof(.), red(.), ugly(.) }
 - 1 binary predicate: $(2^7 \times 7)^1$
{ taller(., .) }
- Total: $7^4 \times (2^7)^3 \times (2^7 \times 7)^1$ models





How to Use Extensions

- Initially, gazillions $(7^4 \times (2^7)^3 \times (2^{7 \times 7})^1)$ of models
- Every statement ELIMINATES some models:
 - Eg, *prof*(*r*) means
only consider models m_f where $f(r) \in f(\textit{prof})$
 - ... initially eliminates $\frac{1}{2}$ of the models...
- ... just like Propositional Logic

Meaning of Quantifiers

- $\forall x Q(x)$
 $\approx Q(v_1) \wedge Q(v_2) \wedge \dots Q(v_n) \wedge \dots$
over all objects v_i (oo number of them)

- Eg:

“All cats are mammals”

$$\forall x \text{Cat}(x) \Rightarrow \text{Mammal}(x)$$

\approx

$$\begin{aligned} \text{Cat}(\text{Spot}) &\Rightarrow \text{Mammal}(\text{Spot}) \wedge \\ \text{Cat}(\text{Reb}) &\Rightarrow \text{Mammal}(\text{Reb}) \wedge \\ \text{Cat}(\text{Felix}) &\Rightarrow \text{Mammal}(\text{Felix}) \wedge \\ \text{Cat}(\text{Rich}) &\Rightarrow \text{Mammal}(\text{Rich}) \wedge \\ \text{Cat}(\text{John}) &\Rightarrow \text{Mammal}(\text{John}) \wedge \end{aligned}$$

...

- $\exists x Q(x)$
 $\approx Q(v_1) \vee Q(v_2) \vee \dots Q(v_n) \vee \dots$
over all objects v_i

- Eg:

- “Spot has a sister, who is a cat”

$$\exists x \text{Sister}(x, \text{Spot}) \wedge \text{Cat}(x)$$

\approx

$$\begin{aligned} \text{Sister}(\text{Spot}, \text{Spot}) &\wedge \text{Cat}(\text{Spot}) \vee \\ \text{Sister}(\text{Reb}, \text{Spot}) &\wedge \text{Cat}(\text{Reb}) \vee \\ \text{Sister}(\text{Felix}, \text{Spot}) &\wedge \text{Cat}(\text{Felix}) \vee \\ \text{Sister}(\text{Rich}, \text{Spot}) &\wedge \text{Cat}(\text{Rich}) \vee \\ \text{Sister}(\text{John}, \text{Spot}) &\wedge \text{Cat}(\text{John}) \vee \end{aligned}$$

...



Notes on Quantifiers

- **Common Mistakes:**

- $\forall x \text{ Cat}(x) \wedge \text{Mammal}(x)$

means

“everything is BOTH a cat and a mammal” !

- $\exists x \text{ Sister}(x, \text{Spot}) \Rightarrow \text{Cat}(x)$

means

“either something is not a sister of Spot, or something is cat” !

Notes on Quantifiers

Nesting: $\forall x \forall y \text{ Parent}(x, y) \Rightarrow \text{Child}(y, x)$
 $\equiv \forall y \forall x \text{ Parent}(x, y) \Rightarrow \text{Child}(y, x)$

Similarly

$$\exists x, y \dots \equiv \exists y, x \dots$$

But...

$$\forall x \exists y \text{ Loves}(x, y) \not\equiv \exists y \forall x \text{ Loves}(x, y)$$

(Everyone loves his mother... vs... GoodAngel loved by all)

Duality: “Everyone dislikes Parsnips” \equiv
“there is no one who likes Parsnips”

$$\forall x \neg \text{Likes}(x, \text{Parsnips}) \equiv \neg \exists x \text{ Likes}(x, \text{Parsnips})$$

De Morgan Rules:

$$\begin{array}{ll} \forall x \neg P \equiv \neg \exists x P & \neg P \wedge \neg Q \equiv \neg(P \vee Q) \\ \neg \forall x P \equiv \exists x \neg P & \neg(P \wedge Q) \equiv \neg P \vee \neg Q \\ \forall x P \equiv \neg \exists x \neg P & P \wedge Q \equiv \neg(\neg P \vee \neg Q) \\ \neg \forall x \neg P \equiv \exists x P & \neg(\neg P \wedge \neg Q) \equiv P \vee Q \end{array}$$

Sufficient Representation

- Kinship Domain

Definition: “A mother is a female parent”

$$\forall m, c \text{ Mother}(m, c) \Leftrightarrow \text{Female}(m) \wedge \text{Parent}(m, c)$$

Disjointness: “Males and females are disjoint”

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

Inverse: “Parent and child are inverse relations”

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

Intervening Entity: “A grandparent is a parent of one’s (intervening) parent”

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

Exclusive: “A sibling is *another* child of one’s parents”

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

- ...

+ Sets, lists, arithmetic, ...

R/N p 256ff

- ... Situations ...

Implementation of Agent

- “Tell” stores facts

Tell(KB, $\forall r_1, r_2$ At(Wumpus, r_1) & Adjacent(r_1, r_2)) \Rightarrow Smelly(r_2))

Tell(KB, Adjacent([3, 4], [2, 4]))

Tell(KB, Smelly([3, 1]))

...

- “Ask” poses queries (using inference rules)

Ask(KB, Adj([3, 4], [2, 4])) \rightarrow Yes

Ask(KB, Adj([3, 4], r)) \rightarrow Yes[r=[2, 4]]

Ask(KB, Action(x, 5)) \rightarrow Yes[x=Grab]



Substitution

- Given sentence S and substitution σ ,
 $S\sigma$ denotes result of plugging σ into S

Eg, $S = \text{Smarter}(x, y)$

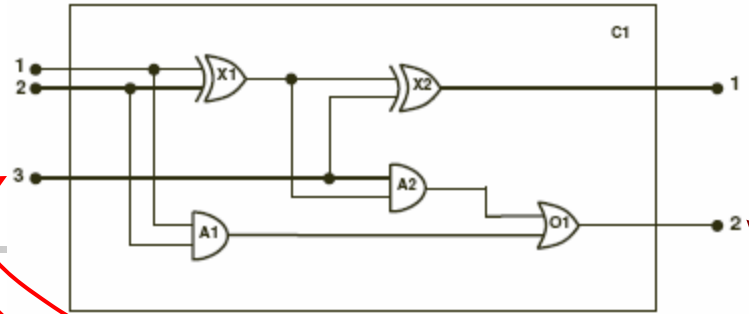
$\sigma = \{ x/\text{Hillary}, y/\text{Bill} \}$

$S\sigma = \text{Smarter}(\text{Hillary}, \text{Bill})$

- $\text{Ask}(\text{KB}, S)$ returns all σ 's such that
 $\text{KB} \models S\sigma$

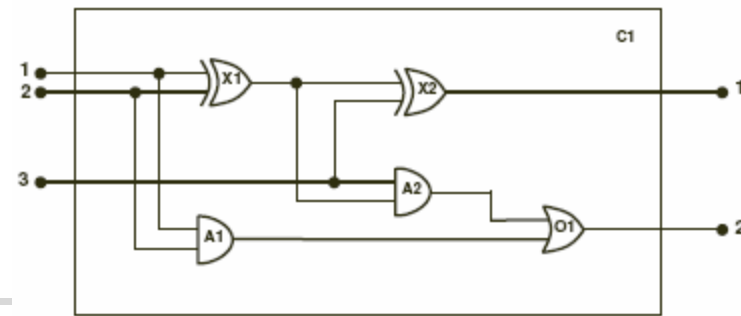
- More about substitutions. . . LATER!

Formalization



- "one-bit full adder":
 - Inputs: two inputs and a carry
 - Output: one output and carry
 - Four gates types: AND, OR, XOR, NOT
- Goal#1*: see if design matches specification
- Consider: circuits, terminals, signals
 - Keep task in mind
 - Eg, for fault diagnosis: if wires can be broken, may want to specify "wires" (eg, `Wire(x,y)`)

Vocabulary



- **Constants:** $X1, X2, \dots, XOR, AND, \dots$

On, Off (signal values)

- **Functions:** $Type(X1) = XOR$

Q. Advantage of function, vs $Type(X1, XOR)$?

$Out(1, X1), In(1, X1), In(2, X1), In(3, X1)$

terms, representing "terminals"

$Signal(x)$

signal value fn (eg, $Signal(In(1, X1))$)

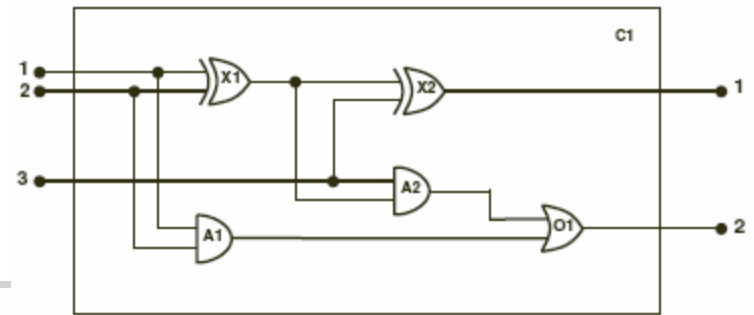
- **Relations:** $Connected(Out(1, X1), In(1, X2))$
for connectivity.

Note: don't have to name terminals explicitly!

"Semantics" of function will assign some unique "object" to it

General Rules

Tell agent...



- How signals behave:

R1 : $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$

R2a : $\forall t \text{ Signal}(t) = \text{On} \vee \text{Signal}(t) = \text{Off}$

R2b : **On \neq Off** **Important, but easy to forget!!**

R3 : $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Leftrightarrow \text{Connected}(t_2, t_1)$

- How gates behave:

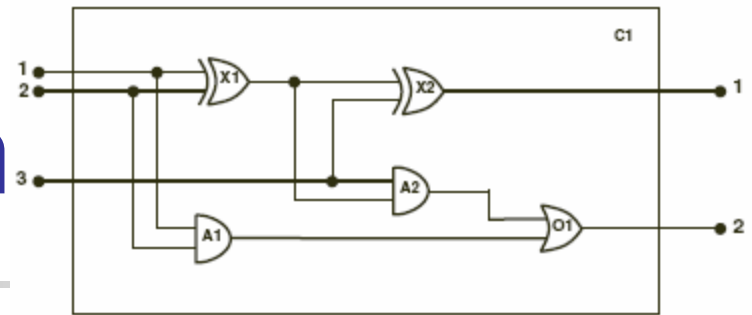
R4 : $\forall g \text{ Type}(g) = \text{OR} \Rightarrow$
 $\text{Signal}(\text{Out}(1, g)) = \text{On} \Leftrightarrow \exists n \text{ Signal}(\text{In}(n, g)) = \text{On}$

R5 : $\forall g \text{ Type}(g) = \text{AND} \Rightarrow$
 $\text{Signal}(\text{Out}(1, g)) = \text{On} \Leftrightarrow \forall n \text{ Signal}(\text{In}(n, g)) = \text{On}$

R6 : $\forall g \text{ Type}(g) = \text{XOR} \Rightarrow$
 $\text{Signal}(\text{Out}(1, g)) = \text{On} \Leftrightarrow (\text{Signal}(\text{In}(1, g)) \neq$
 $\text{Signal}(\text{In}(2, g)))$

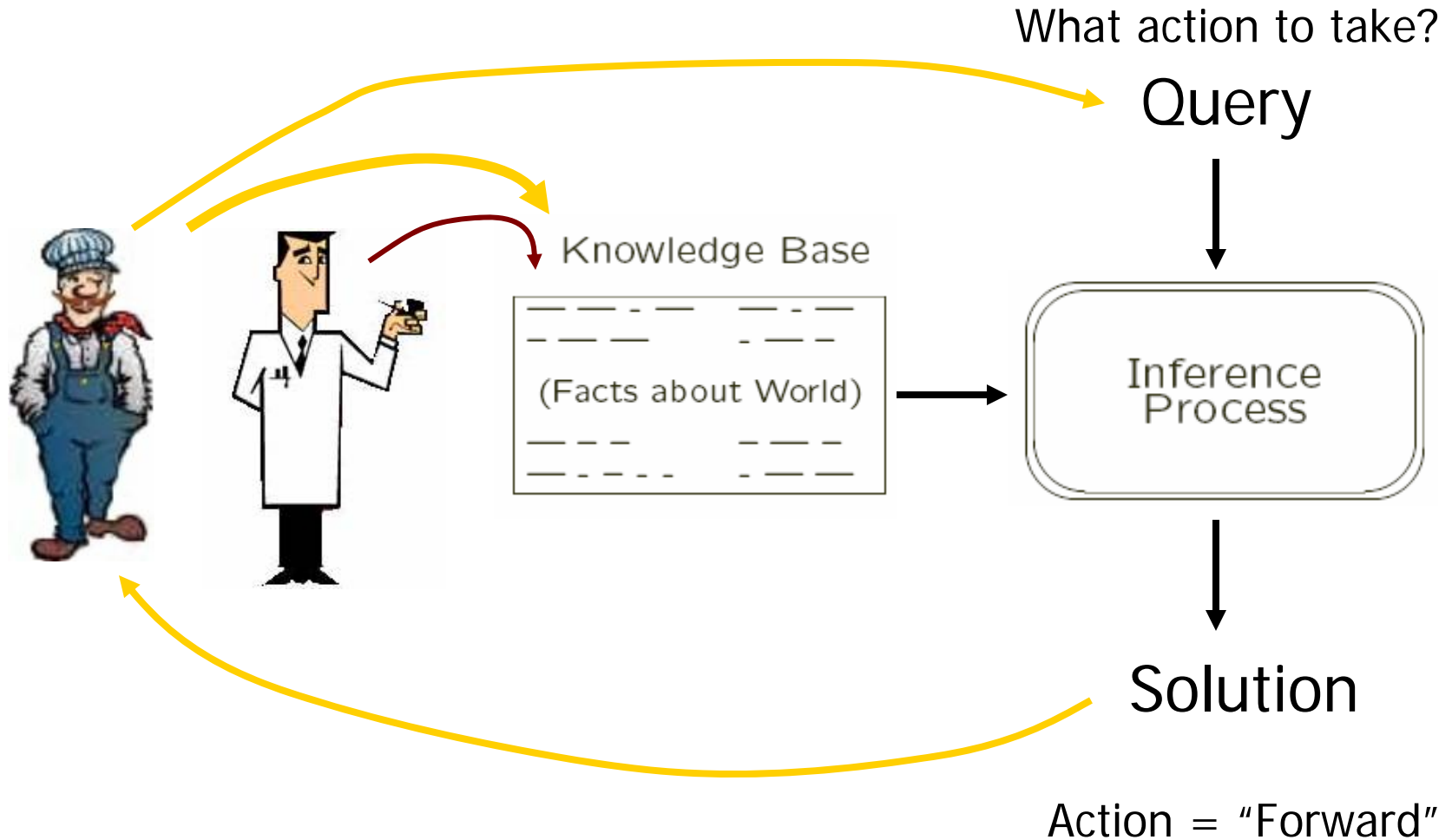
R7 : $\forall g \text{ Type}(g) = \text{NOT} \Rightarrow$
 $\text{Signal}(\text{Out}(1, g)) = \text{On} \Leftrightarrow (\text{Signal}(\text{In}(1, g)) = \text{Off})$

Current Situation



- General Rules are
 - Few (7) \Rightarrow good ontology
 - Clear \Rightarrow good vocabulary
- Now what?
 - Describe SPECIFIC circuit
 - Ask questions about this specific circuit

"Knowledge-Based" System



Describing Specific Circuit

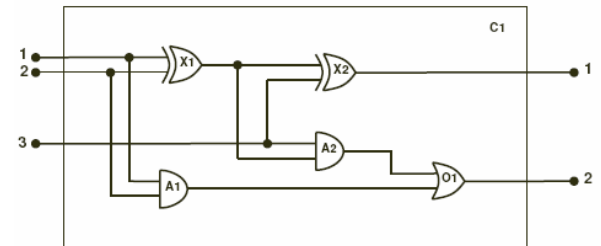
Tell agent:

- Types of gates:

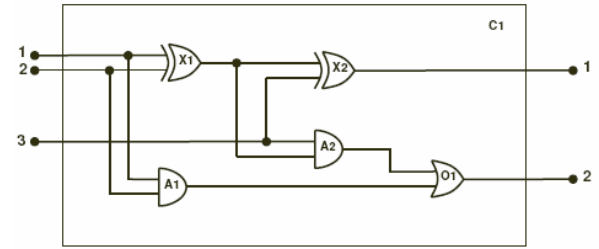
- Type(X1) = XOR
- Type(X2) = XOR
- Type(A1) = AND
- ...

- Connectivity

- Connected(Out(1,X1), In(1,X2))
- Connected(In(1,C1), In(1,X1))
- Connected(Out(1,X1), In(1,A2))
- Connected(In(1,C1), In(1,A1))
- ...



Standard Queries

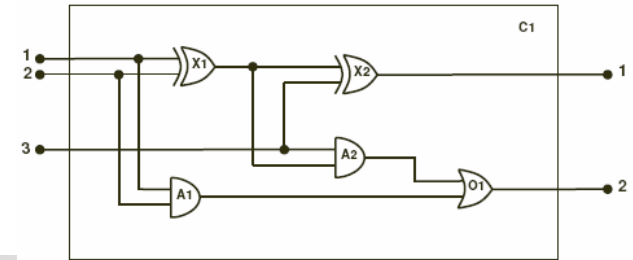


- Given input, what is output?

Q: $\exists o1, o2$ $\text{Signal}(\text{In}(1, C1)) = \text{On}$ &
 $\text{Signal}(\text{In}(2, C1)) = \text{On}$ & $\text{Signal}(\text{In}(3, C1)) = \text{Off}$ &
 $\text{Signal}(\text{Out}(1, C1)) = o1$ & $\text{Signal}(\text{Out}(2, C1)) = o2$
?

A: $o1 = \text{Off}$ & $o2 = \text{On}$

Queries



- Our KB captures full behavior.
- Can "Ask" different queries about behavior

Q: $\exists i1, i2, i3$ $\text{Signal}(\text{In}(1, C1)) = i1$ &
 $\text{Signal}(\text{In}(2, C1)) = i2$ & $\text{Signal}(\text{In}(3, C1)) = i3$ &
 $\text{Signal}(\text{Out}(1, C1)) = \text{Off}$ & $\text{Signal}(\text{Out}(2, C1)) = \text{On}$?

A: $(i1 = \text{On} \ \& \ i2 = \text{On} \ \& \ i3 = \text{Off}) \vee$
 $(i1 = \text{On} \ \& \ i2 = \text{Off} \ \& \ i3 = \text{On}) \vee$
 $(i1 = \text{Off} \ \& \ i2 = \text{On} \ \& \ i3 = \text{On})$

Q: What is the advantage over direct simulation?

A: Allows agent to reason about overall behavior.

Eg, What *inputs* give a particular *output*?



Pro/Con wrt Formalizing

- Used in analysis of circuits/systems
 - Contrast with Truth-Table method
- Formalization is facilitated by “closeness” between logical formalisms and digital circuitry
 - ... allows for very powerful design methods (but did not prevent Pentium bug. . .)
- Defining natural kinds:
game or **chair**
- ... difficulty w/ “necessary and sufficient” conditions.
- Problem with “strict definition” [Quine 1953]
 - “the Pope is a bachelor”



Summary

- First-order logic:
 - objects, relations are semantic primitives
 - syntax: constants, functions, predicates, equality, quantifiers
- Increased expressive power:
 - ... sufficient to define Wumpus World, succinctly!