Chapter 5: Logic

based largely on

Russell, S.J., and Norvig, P.: **“Artificial Intelligence - a modern approach”. Prentice Hall.**

Prepared for the class “Formal Methods for Computer Science II” (by Rolf Pfeifer and Ruedi Fuechslin)

(subject to change)

R/N: Russell/Norvig

Why study logic?

logical puzzles:

I say: I am a lier. Am I a lier?

(the barber says) I shave everyone in this town who doesn't shave himself. Does he have a beard?

God is almighty. Can he create a stone that is so big that he can't lift it any longer?

Why study logic?

formal languages and grammars:

string s: s∈L(G)?

logic:
relation between sentences - deriving one sentence from other sentences

enables us to formalize the notion of a *proof*

Why study logic?

- foundation of mathematics and computer science
- statements about the real world —> representation
- Artificial Intelligence:
  - agents acting in real world
  - expert systems / logic programming

By defining how theorems - i.e. valid formulas (see below) - can be derived in a purely syntactic way, computers can then be used to derive statements about the “real” world. We introduce the notion of a representation: how does a sentence relate to the real world?

modern robotics: representations largely logic-based

declarative vs. procedural programming

logic programming: e.g. Prolog - computational linguistics, expert systems

People reasoning “logically”

**(Belvedere’s Witch Trial:
see Monty Python and the holy grail)**

http://www.youtube.com/watch?v=kBcKyWbYXdM

FIRST VILLAGER: We have found a witch. May we burn her?
ALL: A witch! Burn her!
BEDEVERE: Why do you think she is a witch?
SECOND VILLAGER: She turned *me* into a newt.
BEDEVERE: A newt?
SECOND VILLAGER *(after looking at himself for some time)*: I got better.
ALL: Burn her anyway.
BEDEVERE: Quiet! Quiet! There are ways of telling whether she is a which.
BEDEVERE: Tell me ... what do you do with witches?
ALL: Burn them.
BEDEVERE: And what do you burn, apart from witches?
FOURTH VILLAGER: ... Wood?
BEDEVERE: So why do witches burn?

SECOND VILLAGER: Because they're made of wood?
BEDEVERE: Good.
ALL: I see. Yes, of course.
BEDEVERE: So how can we tell if she is made of wood?
FIRST VILLAGER: Make a bridge out of her.
BEDEVERE: Ah ... but can you not also make bridges out of stone?
ALL: Yes, of course ... um ... er ...
BEDEVERE: Does wood sink in water?
ALL: No, no, it floats. Throw her in the pond.
BEDEVERE: Wait. Wait ... tell me, what also floats on water?
ALL: Bread? No, no, no. Apples ... gravy ... very small rocks ...
BEDEVERE: No, no, no.
KING ARTHUR: A duck!
(The all turn and look at ARTHUR. BEDEVERE looks up very impressed.)

BEDEVERE: Exactly. so ... logically ...
FIRST VILLAGER *(beginning to pick up the thread)*: If she ... weighs the same as a duck ... she's made of wood.
BEDEVERE: And therefore?
ALL: A witch!

Example of "logical" reasoning gone wrong. From *Monty Python and the Holy Grail,* 1977, Reed Consumer Books.

(from Russell/Norvig: *Artificial Intelligence, a modern approach*, Prentice Hall, 1995).

Logic

- propositional calculus (sentence logic)
- first order predicate calculus, FOPC; or first order logic, FOL
- other kinds of logics: deontic logic, temporal logic, higher-order logic
- fuzzy logic

propositional calculus: German “Aussagenlogik”

getting into the spirit

Game

Task1: cards - letter side A, D opposite side - 4, 7

Rule: letter side: A —> number side: must be 4

Which cards to flip?

As, 7s

getting into the spirit

Game

Task 2: checks over USD 100.00 must have signature of manager on other side

Which checks to flip?

USD over 100 (As), without signature (7s)
Simple problems: common sense sufficient
more complex stuff —> logical formalisms required
AND: we want to automate the procedures (fewer errors)

A simple knowledge-based agent

The agent must be able to:

- represent states, actions, etc. (KB)
- incorporate new percepts (additions to KB)
- update internal representations of the world
- deduce hidden properties of the world
- deduce appropriate actions

Knowledge bases:

- inference engine: domain-independent algorithms
—> logic!
- knowledge base KB: domain-specific content

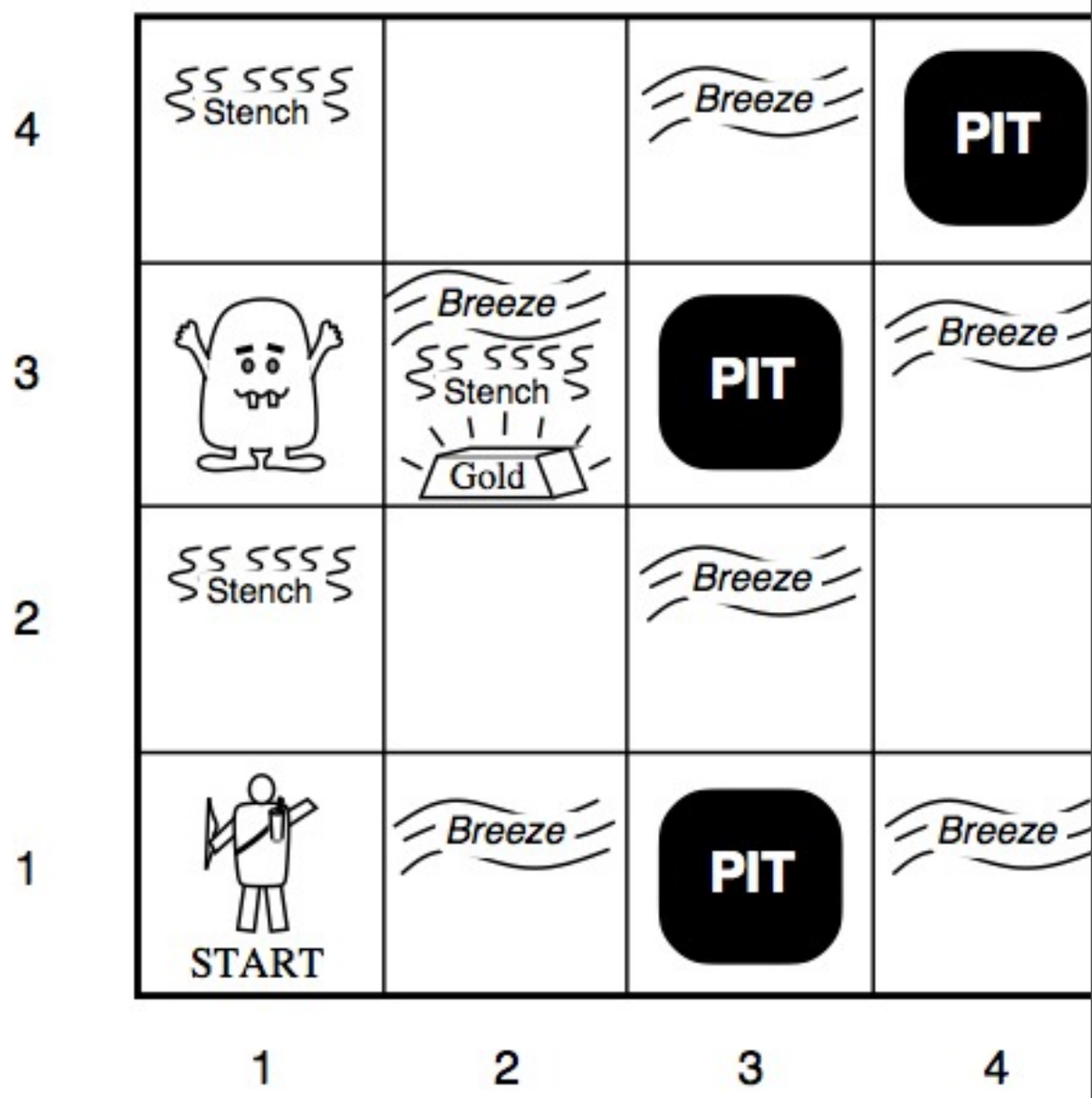
Logic: representing and reasoning about the world

Percepts: Breeze, Glitter, Smell

Actions: Left turn, Right turn, Forward, Grab, Release, Shoot

Goals Get gold back to start without entering pit or Wumpus square

Environment
- **Squares adjacent to Wumpus smell**
- **Squares adjacent to pit are breezy**
- **Glitter iff gold is in the same sqare**
- **Shooting kills Wumpus if facing it**
- **Shooting uses up the only arrow**
- **Grabbing picks up the gold if in same squre**
- **Releasing drops the gold in same square**



From: Russel/Norvig, p. 154

Representation

Sound, truth-preserving:

$$KB \vdash A \Rightarrow KB \models A$$

Completeness:

$$KB \models A \Rightarrow KB \vdash A$$

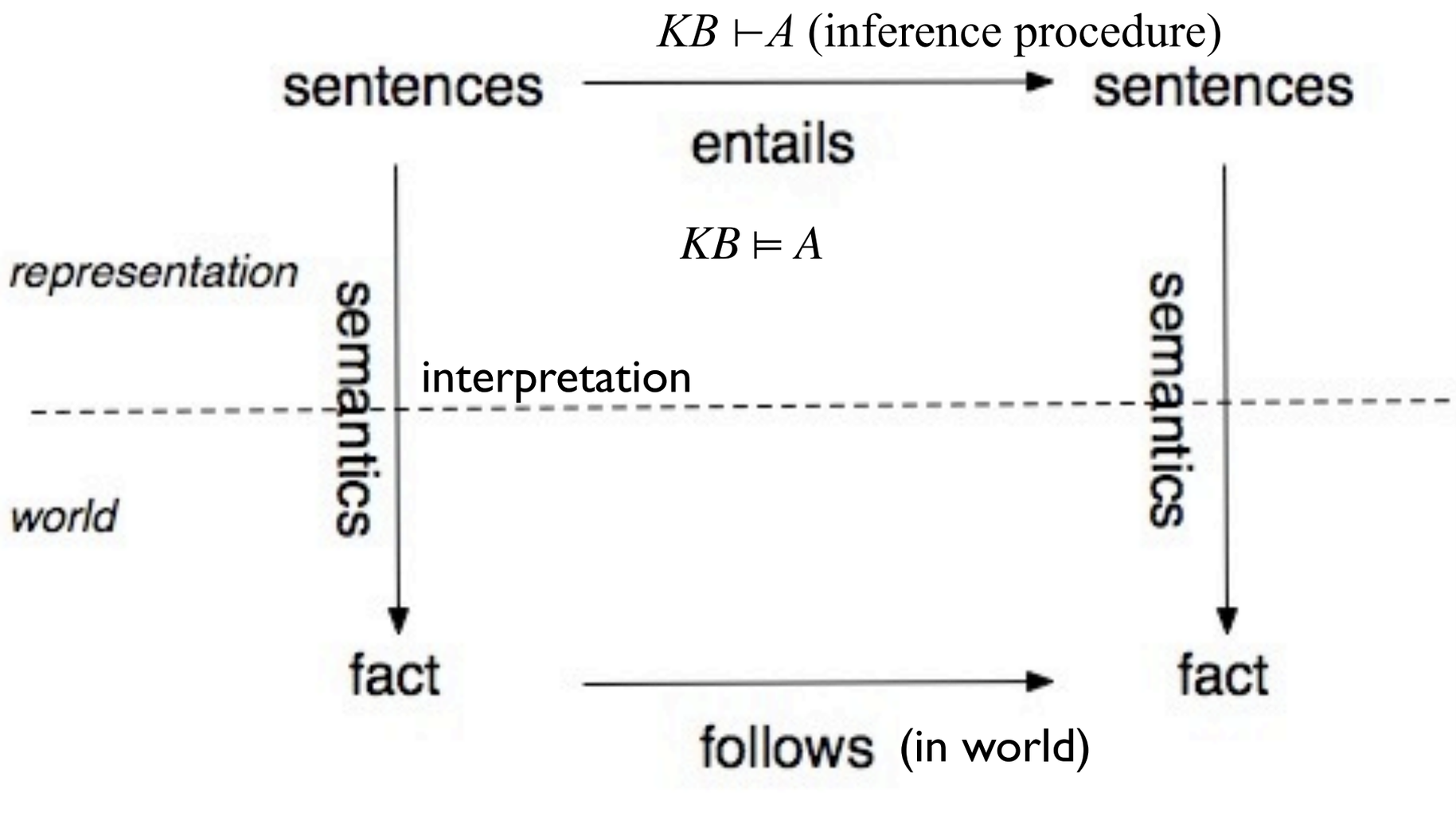
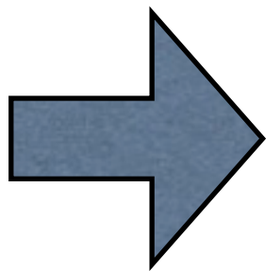


Figure: The connection between sentences and facts is provided by the semantics of the language. The property of one fact following from some other fact is mirrored b the property of one sentence being entailed by some other sentences. Logical inferen generates new sentences that are entailed by existing sentences (from Russell/Norvi

This is, in essence, the problem of representation: sentences corresponding to facts, entailment “modeling” the “follows” relation in the real world.

Logic

- **syntax (formal system)**



logic

- **semantics**

Propositional calculus

formal system

1. **formal language**
2. **set of axioms**
3. **set of inference rules (transformations)**

1. formal language

- a set P of atomic formulas, e.g. symbols such as p, q, r, …
- logical connectives: negation, and, or, implication, equivalence
- auxiliary symbols: (and)

Definition: Propositional formula:
- every atomic fomrula is a prop. formula
- if A and B are prop. formulas, then not A, AandB, AorB, AimpliesB, AequivB are prop. formulas
- every prop. formula arises from a finite number of applications of these two rules

More compact way?

Propositional calculus

formal system

1. formal language
2. set of axioms
3. set of inference rules (transformations)

1. formal language

Q: how can it be defined?

Propositional calculus

formal system

1. **formal language**
2. **set of axioms**
3. **set of inference rules (transformations)**

$$
\begin{aligned}
\langle \text{formula} \rangle &\rightarrow \langle \text{atomic formula} \rangle \mid \langle \text{propositional formula} \rangle \\
\langle \text{atomic formula} \rangle &\rightarrow T \mid F \mid p \mid q \mid r \mid \ldots \\
\langle \text{propositional formula} \rangle &\rightarrow (\langle \text{formula} \rangle) \\
&\mid \langle \text{formula} \rangle \langle \text{connector} \rangle \langle \text{formula} \rangle \\
&\mid \neg \langle \text{formula} \rangle \\
\langle \text{connector} \rangle &\rightarrow \wedge \mid \vee \mid \rightarrow \mid \leftrightarrow
\end{aligned}
$$

8

formal language: defined by grammar in EBNF notation

Atomic formulas/sentences

- **statements about world**
- **no decomposition:**
  Socrates is human —> human_socrates, hs
- **i.e. world with no objects, only propositions**

only atomic sentences possible

Ambiguity

- defining priorities of the operators
- parentheses

resolving ambiguities; there is an agreed-upon hierarchy of operators: not, and, or, implication/equivalence

Propositional calculus

2. Set of axioms (Lukasiewicz)

$$p \rightarrow (q \rightarrow p) \quad \text{(A1)}$$

$$(p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r)) \quad \text{(A2)}$$

$$(\neg p \rightarrow \neg q) \rightarrow (q \rightarrow p) \quad \text{(A3)}$$

using as primitives (elementary operators or connectives) only → and ¬

generates all wffs of propositional calculus (with inference rules)

wff: well-formed formula or sentence

many other sets of axioms have been suggested

“Axioms are used by mathematicians/logicians to capture the basic facts about a domain, define other concepts in terms of those basic facts, and then use the axioms and definitions to prove theorems.
In mathematics, an independent axiom is one that cannot be derived from all the other axioms. Mathematicians strive to produce a minimal set of axioms that are all independent. In more applied disciplines such as artificial intelligence it is common to include redundant axioms, not because they can have any effect on what can be proved, but because they can make the process of finding a proof more efficient” (Russell/Norvig, p. 198).

Propositional calculus

3. **Inference rules**
   only one necessary: MP (Modus Ponens)

   Q?

Propositional calculus

3. **Inference rules**
   only one necessary: MP (Modus Ponens)

$$\frac{p \rightarrow q,\ p}{q}$$

Proving a simple wff

proving the seemingly simple formula:

$A \rightarrow A$

$A \rightarrow ((A \rightarrow A) \rightarrow A)$	a case of (A1) axiom
$(A \rightarrow ((A \rightarrow A) \rightarrow A)) \rightarrow$ $[(A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A)]$	a case of (A2) axiom
$(A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A)$	modus ponens
$(A \rightarrow (A \rightarrow A))$	a case of (A1) axiom
$A \rightarrow A$	modus ponens

on white board

while this formula may be intuitively obvious, the point here is to use only axioms and modus ponens in a purely mechanical way

Propositional calculus

3. Inference rules

- **Modus Ponens:**

$$\frac{p \rightarrow q,\ p}{q}$$

- **And-Elimination (from a conjunction infer any of the conjuncts)**
- **And-Introduction (from a list of sentences, infer their conjunction)**
- **Or-Introduction (from a sentence, infer its disjunction with anything)**
- **Double-Negation Elimination (double negation is the same as the sentence itself)**
- **Resolution**

although only one inference rule (MP) is necessary, the introduction of additional ones makes inference more efficient and convenient.

and-elimination/introduction

$$\frac{a1 \wedge a2 \wedge \ldots \wedge an}{ai}$$

$$\frac{a1, a2, \ldots, an}{a1 \wedge a2 \wedge \ldots \wedge an}$$

If the conjunction of n propositions holds, I can infer every component in the conjunction individually. Conversely, if a set of individual proposition holds, I can infer its conjunction.

Propositional calculus

3. **Inference rules**

- **Modus Ponens:**

$$\frac{p \rightarrow q,\ p}{q}$$

- **And-Elimination (from a conjunction infer any of the conjuncts)**
- **And-Introduction (from a list of sentences, infer their conjunction)**
- **Or-Introduction (from a sentence, infer its disjunction with anything)**
- **Double-Negation Elimination (double negation is the same as the sentence itself)**
- **Resolution**

Propositional calculus

- **resolution (transitivity of implication)**

$$\frac{p \vee q,\ \neg q \vee r}{p \vee r}$$

or equivalently:

$$\frac{\neg p \rightarrow q,\ q \rightarrow r}{\neg p \rightarrow r}$$

Resolution is interesting because it is the inference rule used in the programming language Prolog. (a declarative language, see below)

Semantics of propositional calculus

concerned with truth or falsity of propositional formulas

Def.: Interpretation /
assignment of a truth value to every atomic formula

e.g. *I(p)=T; I(q)=F; I(r)=F.*

truth values of formulas: **unambiguously determinable from truth values of atomic formulas**

—> *Method:* ***truth tables***

Compositionality

interesting languages:

compositional: meaning of sentence is function of meaning of its parts

x2 + y**2 is related to meanings of**

x2 and y**2**

Natural language: Meaning of “John hits Mary” - is related to the meanings of John, hits, and Mary.

Truth tables for logical connectives

they define the semantics of the logical connectives:

P	Q	$\neg P$	$P \vee Q$	$P \wedge Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
T	T	F				
T	F	F				
F	T					
F	F					

p	q	r	p∨q	¬q∨r	p∨r
F	F	F	F	T	F
F	F	T	F	T	T
F	T	F	T	F	F
F	T	T	T	T	T
T	F	F			

truth table for resolution

Some theorems

- **De Morgan**

1. $(\neg (P \vee Q)) \leftrightarrow (\neg P) \wedge (\neg Q)$
2. $(\neg (P \wedge Q)) \leftrightarrow (\neg P) \vee (\neg Q)$

- **distributive laws**

1. $(P \vee (Q \wedge R)) \leftrightarrow ((P \vee Q) \wedge (P \vee Q))$
2. $(P \wedge (Q \vee R)) \leftrightarrow ((P \wedge Q) \vee (P \wedge R))$

script, p. 5-8, bottom
(write on white board)

Some theorems

- **De Morgan**

1. $\neg(P \wedge Q) \leftrightarrow (\neg P \vee \neg Q)$
2. $\neg(P \vee Q) \leftrightarrow (\neg P \wedge \neg Q)$

"good" theorems?

- **distributive laws**

1. $(P \vee (Q \wedge R)) \leftrightarrow ((P \vee Q) \wedge (P \vee Q))$
2. $(P \wedge (Q \vee R)) \leftrightarrow ((P \wedge Q) \vee (P \wedge R))$

script, p. 5-8, bottom
Using the truth tables, we can figure out whether they hold for any interpretation.

Some theorems

De Morgan

1. $(\neg (P \vee Q)) \leftrightarrow (\neg P) \wedge (\neg Q)$
2. $(\neg (P \wedge Q)) \leftrightarrow (\neg P) \vee (\neg Q)$

P	Q	$(\neg (P \vee Q))$	$(\neg P) \wedge (\neg Q)$
T	T	F	F
T	F	F	F
F	T		
F	F		

distributive laws

1. $(P \vee (Q \wedge R)) \leftrightarrow ((P \vee Q) \wedge (P \vee Q))$
2. $(P \wedge (Q \vee R)) \leftrightarrow ((P \wedge Q) \vee (P \wedge R))$

Illustration: Logical

1. If the weather is bad, I stay at home.

2. I die if and only if my house explodes because of a gas leak and I'm at home.

3. If my house explodes or I go out, and only then, my neighbors notice something strange.

4. Whenever my neighbors notice something strange (and as long as I'm alive), they call us at home the next day. They never call us otherwise.
My wife (who works at home) tells you that she received a call from the neighbors last Tuesday. Can you say something about the weather last Monday?

There is always a certain ambiguity in natural language texts; formalization is not unique; so such examples have to be taken with a grain of salt.

Formalization

a: bad_weather

b: stay_home

c: I_die

d: gas_leak_explosion

e: neighbors_notice_something_strange

f: neighbors_call_us_at_home

atomic sentences - no decomposition

Of course, if we had a computer program, the program would not be able to interpret these labels - they are only for us. For the computer, simply using a, b, c, etc. is exactly equivalent.

Knowledge base (axioms)

1. $a \rightarrow b$

2. $c \leftrightarrow d \wedge b$

3. $d \vee \neg b \leftrightarrow e$

4. $e \wedge \neg c \leftrightarrow f$

proof: see later

Proof, provability

a finite sequence of formulas

A1, A2, …, An

is a **proof** of formula *A* if *An* is the formula *A* and *Ai* is either an axiom or derived from previous formulas by a sound inference rule

If a proof of *A* exists, we say that *A* is **provable** and write $\vdash A$.

A is said to be **provable from *KB*** if *A* can be proven from axioms and formulas in KB using sound inference rules.
$KB \vdash A$

Inference procedure, soundness, entailment

An inference procedure is **sound** if it generates sentences that are necessarily true given that the old sentences are true. This relation is called ***entailment***. We write

$KB \models A$ and say *KB* **entails** A.

Inference procedures generating only entailed sentences are called ***sound*** or ***truth-preserving***.

Inference procedure:
- given a knowledge base *KB,* generate new sentences that (should be) entailed by *KB*

- given a *KB* and a sentence A, report whether or not *A* is entailed by *KB*.

Example

back to the “logical”

Knowledge base KB

1. $a \rightarrow b$

2. $c \leftrightarrow d \wedge b$

3. $d \vee \neg b \leftrightarrow e$

4. $e \wedge \neg c \leftrightarrow f$

inference rules to be used:

- de Morgan 2: $(\neg (P \wedge Q)) \leftrightarrow (\neg P) \vee (\neg Q)$
- distributive law 1: $(P \vee (Q \wedge R)) \leftrightarrow ((P \vee Q) \wedge (P \vee Q))$

Proof

$f \leftrightarrow e \wedge \neg c$ (formula 4 in *KB*)

$\leftrightarrow (d \vee \neg b)$ (f. 3 in *KB*) $\wedge \neg (d \wedge b)$ (f. 2 in *KB*)

$\leftrightarrow (d \vee \neg b) \wedge (\neg d \vee \neg b)$ (de Morgan 2.)

$\leftrightarrow (d \vee \neg d) \vee \neg b$ distributive law 1. with
$Q=d$, $P= \neg b$, $R= \neg d$

$\leftrightarrow \neg b$ (unit resolution)

implies $\neg a$

$a \rightarrow b \leftrightarrow \neg b \rightarrow \neg a$ (use formula 1 in KB)

Proof

sequence of formulas using

- elements of KB

- inference rules

sound inference procedure? truth preserving?

—> truth tables

Truth table

1. $(e \wedge \neg c)$
2. $(d \vee \neg b) \wedge \neg (d \wedge b)$
3. $(d \vee \neg b) \wedge (\neg d \vee \neg b)$
4. $(d \vee \neg d) \vee \neg b$
5. $\neg b$
6. $\neg a$

a	b	c	d	e	f	1	2	3	4	5
	F	F	F	T	T	T	T	T	T	T

Terminology: semantics (of propositional calculus)

- Interpretation:
- Model:
- Satisfiability:
- Tautologic consequence:
- Tautology:

Terminology: semantics (of propositional calculus)

- Interpretation I: **assignment of T and F to all atomic sentences in A (A a sentence or wff).**
- Model: **World in which A is true under a particular interpretation I (Russell/Norvig, p. 170)**

- Satisfiability**: A is satisfiable if there is an interpretation I under which A is true.**
- Tautologic consequence**: (**A is entailed by a set of formulas **M) if A is true for every interpretation I in which all formulas of M are true. We write: M I= A**
- Tautology**: If A is true under all interpretations. We write: I= A**

The logical notion of a model is somewhat counterintuitive (different from what we normally think of as models).

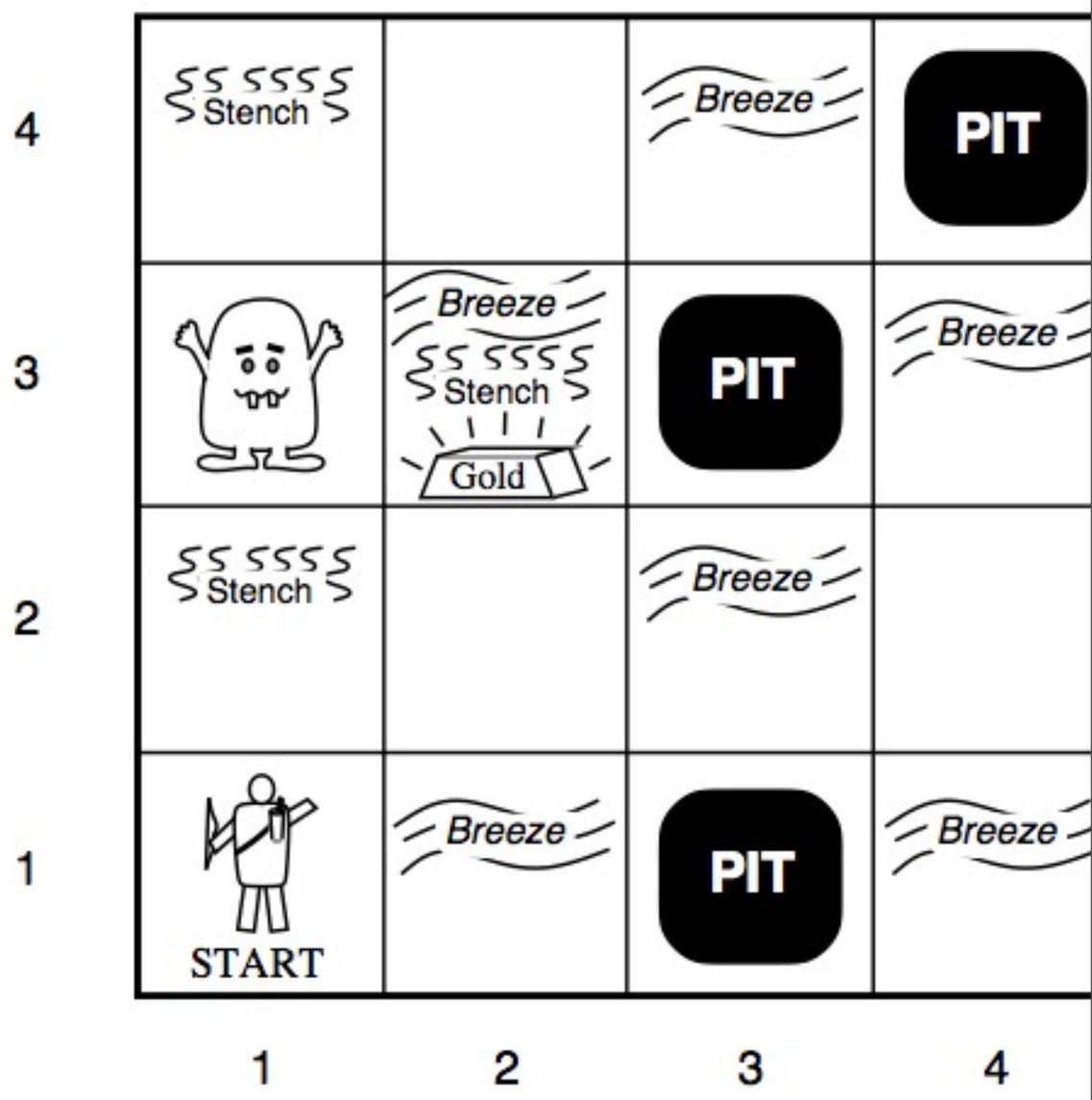
Logic: representing and reasoning about the world

Percepts: Breeze, Glitter, Smell

Actions: Left turn, Right turn, Forward, Grab, Release, Shoot

Goals Get gold back to start without entering pit or Wumpus square

Environment
- **Squares adjacent to Wumpus smell**
- **Squares adjacent to pit are breezy**
- **Glitter iff gold is in the same sqare**
- **Shooting kills Wumpus if facing it**
- **Shooting uses up the only arrow**
- **Grabbing picks up the gold if in same squre**
- **Releasing drops the gold in same square**



From: Russel/Norvig, p. 154

Terminology: semantics (of propositional calculus)

- Interpretation I: **assignment of T and F to all atomic sentences in A (A a sentence or wff).**
- Model: **World in which A is true under a particular interpretation I (Russell/Norvig, p. 170)**
  e.g. the Wumpus world (see previous slide) is a model of sentence "S(1,2)", under the interpretation in which it means that there is a stench in square (1,2).

Model: There are many other models of this sentence: e.g. worlds, in which there are no pits and gold. As long as the world has a stench in (1,2), it is a model of the sentence. The reason, there are so many models is because "S(1,2)" makes a very weak claim about the world. The more we claim (in the sentence A), i.e. the more conjunctions we add into the knowledge base, the fewer models there will be.

Terminology: semantics (of propositional calculus)

- Interpretation I: **assignment of T and F to all atomic sentences in A (A a sentence or wff).**
- Model: **World in which A is true under a particular interpretation I (Russell/Norvig, p. 170)**
- Satisfiability**: A is satisfiable if there is an interpretation I under which A is true.**
- Semantic consequence**: (**A is entailed by a set of formulas **M) if A is true for every interpretation I in which all formulas of M are true. We write: M I= A**
- Tautology**: If A is true under all interpretations. We write: I= A**

The logical notion of a model is somewhat counterintuitive (different from what we normally think of as models).

Semantic consequence (sometimes called tautologic consequence): Formula A is a semantic consequence of M iff there is no model in which all formulas of M are true and A is false. Stated differently: The set of interpretations that make all formulas of M true is a subset of the set of interpretations that make A true.

Syntax

- Proof:

- Provability:

- Sound, truth-preserving:

- Completeness:

Syntax

- Proof**: finite sequence of formulas A1, A2, … An is** proof **of A if An is A, and for any i, Ai is either an axiom or is derived from previous formulas.**
- Provability**: If A can be proved from a set of assumptions M, it is** provable from M**. We write**
  M |— A
- Sound, truth-preserving**: An inference rule is** *sound or truth-preserving*
  M |— A => M |= A
- Completeness: **M |= A => M |— A**
- Decidability**: a system is** decidable **if there is an effective procedure for demonstrating either (M |— A) or not(M |— A)**

A method is called effective for a class of problems iff

- it consists of a finite number of instructions
- when applied to a problem from its class, it always finishes (*terminates*) after a finite number of steps
- when applied to a problem from its class, it always produces a correct answer

Normal forms

- **CNF (Conjunctive Normal Form) conjunction of disjunctions**
- **DNF (Disjunctive Normal Form) disjunction of conjunctions**
- Theorem: **Every sentence of propositional calculus —> equivalent to normal form**

In other words, we only need proofs about normal forms.

Example: transformation to CNF

$A \rightarrow \neg(B \rightarrow C)$

1. elimination of $\rightarrow$ using $P \rightarrow A \equiv \neg P \vee Q$ (use $P{:}A$; $Q{:}\neg(B \rightarrow C)$)

$A \rightarrow \neg(B \rightarrow C) = (\neg A \vee \neg(\neg B \vee C))$ (using $(B \rightarrow C) = (\neg B \vee C)$)

2. distribution of $\neg$ onto atomic expression:

$(\neg A \vee \neg(\neg B \vee C))$
$(\neg A \vee (\neg \neg B \neg C))$ (using de Morgan 1 $\neg(P \vee Q) = \neg P \wedge \neg Q$)
$(\neg A \vee (B \wedge \neg C))$

3. transform into conjunction of disjunctions using distributive rule 1
$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$

$((\neg A \vee B) \wedge (\neg A \vee \neg C))$ is CNF

Propositional calculus and Boolean Logic

operators:
- multiple inputs, one output
- simulate gates

Propositional calculus and Boolean Logic

AND (all high = high, else low)

Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

OR (any high = high, else low)

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

NAND (all high = low, else high)

Input 1	Input 2	Output
0	0	1
0	1	1
1	0	1
1	1	0

NOR (any high = low, else high)

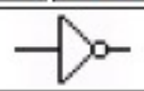
Input 1	Input 2	Output
0	0	1
0	1	0
1	0	0
1	1	0

XOR (different = high, same = low)

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

NOT (inverter)

Input = 1	Output = 0
Input = 0	Output = 1



Here is a link to an extremely easy-to-read intro to Boolean logic:
http://computer.howstuffworks.com/boolean.htm

Propositional calculus

main problem?

very limited expressivity (only atomic sentences)
(but nice mathematical properties)
This is a trade-off that is often found in formal systems (see also Chomsky-hierarchy).

First-order predicate calculus

main differences to propositional calculus?

digression: general comments on the use of logic

FOPC: objects and quantifiers, whereas propositional calculus only has atomic sentences.

General comments on the use of logic/FOPC: Introduction

"Although first-order logic commits to the existence of objects and relations, it does not make an ontological commitment to such things as categories, time and events, which also seem to show up in most facts about the world. Strangely enough, this reluctance to tackle categories, time and events has not hurt the popularity of first-order logic: in fact it has contributed to its success. Important as these things are, there are just too many different ways to deal with them, and a logic that committed to a single treatment would only have limited appeal. By remaining neutral (and very abstract - comment by Rolf Pfeifer), FOPC gives its users the freedom to describe these things in a way that is appropriate for the domain. This freedom of choice is a general characteristic of first-order logic. In the previous example we listed King as a property of people, but we could just as well have made King a relation between people and countries, or a function

Logic is also useful in general as an abstract descriptive language.

Also: There are programming languages (declarative, rather than imperative) that implement subsets of FOPC (and provide you the inference procedures, i.e. you only write what you know and the programming language will figure out the solutions; most notable example, Prolog).

General comment on logic and its use (1)

- **representation of real world (of interest)**
- typically: **knowledge bases, data bases**
- interested in: **automated inference procedures**
- **automatic theorem proving**
- looking for: **efficient automated proof procedure**

General comment on logic and its use (2)

- propositional calculus: **sound, complete, decidable**
- however: **extremely limited expressivity**
- FOPC: **much higher expressivity, but only semi-decidable (see later)**
- logic programming (e.g. Prolog): **compromise**
  - expressivity
  - efficiency of proof procedure
- **Horn Clause Logic with resolution as inference procedure (cf. Prolog)**

semi-decidable:
can show that sentences follow from premises if they do, but we cannot always show this if they do not (see later).

General comments on logic and its use (3) (skip)

- "refutation complete:" **If a set of sentences is unsatisfiable, then resolution will derive a contradiction (R/N p. 286)**
- sentence: **must be in conjunctive normal form (CNF) or implicative normal form**
- Horn Clauses: **subset of CNF/implicative normal form:**
  p ∨ ¬n1 ∨ ¬n2 ∨ … ∨ ¬nm; written as implication:
  (n1 ∧ ... ∧ nm) → p; or in Prolog Notation:
  p :- n1, … nm. (m>0)
  p. (m=0)
  ni, p atomic sentences

General comments on logic and its use (4)

- transformation to CNF (or implicative normal form)
- if Horn Clause: efficient procedure exists
- represents interesting compromise

Remember: representation

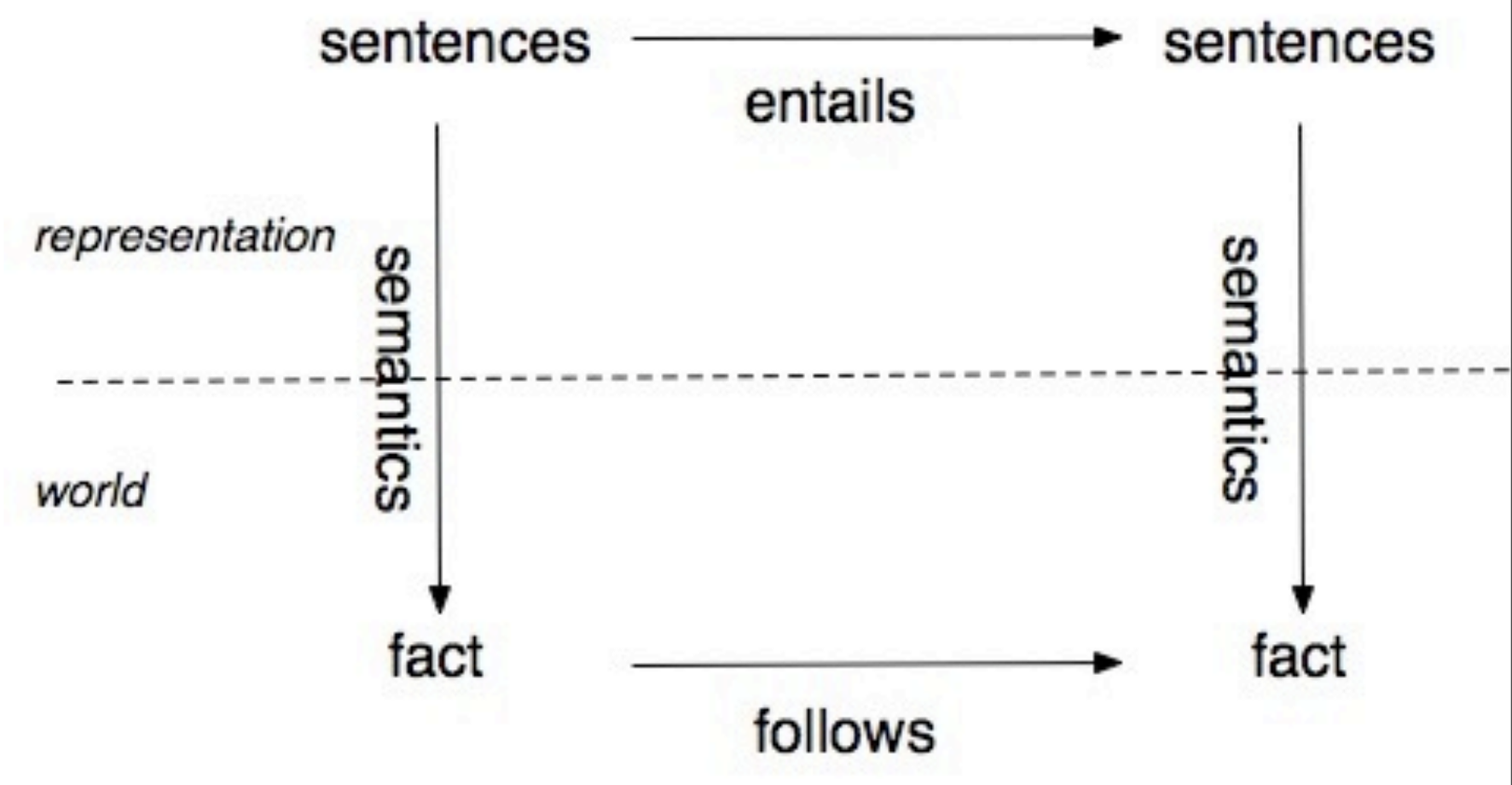


Figure: The connection between sentences and facts is provided by the semantics of the language. The property of one fact following from some other fact is mirrored the property of one sentence being entailed by some other sentences. Logical infer generates new sentences that are entailed by existing sentences (from Russell/Nor p. 158).

Introduction to:
First Order Logic

- most important ideas, intuitions
- not detailed technical knowledge
- part of computer science basic education

First Order Logic: intuition

(RN p. 185)

- **world consists of:** objects **(things with individual identities and properties that distinguish them from other objects)**
- **among these objects, various** relations **hold:**
- objects**: people, houses, numbers, theories, Barack Obama, colors, football games, wars, centuries, …**
- relations**: examples? …**
- properties**: examples? …**
- functions (mapping objects to one object)**: examples? …**
- **Almost any fact can be thought of as referring to objects and properties or relations**

First Order Logic: intuition

(RN p. 186)

- Almost any fact can be thought of as referring to objects and properties or relations, e.g.

- "Evil King John ruled England in 1200"
  objects: John, England, 1200
  relation: ruled
  properties: evil, king

Semantics
Interpretation: intuition

- constant symbol: **object in the real world (domain of discourse)**
- predicate symbols: **Round, Brother (predicate symbol -> relation; "Brother" might represent the relation "brotherhood") (relation symbols are also called predicates, thus predicate logic)**
- function symbols: **Cosine, FatherOf, LeftLegOf, …**
  function**: relation where**
  object -> (exactly one) other object
  e.g. angle -> exactly one number corresponding to Cosine(angle)
- Term: **logical expression that refers to an object**
  Ground term: **term with no variables**

Just as in the case of propositional calculus, we need an interpretation to define the semantics. But now that we have a much more rich and complicated formalism (with objects, relations, functions, etc.), the notion will be more complicated.

Atomic and complex sentences

(RN p. 189)

now that we have terms for objects and predicate symbols for referring to relations: combine them to make atomic sentences **that state facts, e.g.**

Brother(Richard,John) **or with arguments that are complex terms**

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

An atomic sentence is true **if the relation referred to by the predicate symbol holds between the objects referred to by the arguments.**

we can use logical connectives **to construct more** complex sentences**, as in propositional calculus**

Quantifiers

(RN p. 189)

Quantifiers:

universal quantification

(∀x) Cat(x) → Mammal(x)

(∀x) CitOfZRH(x) → Cool(x)

cannot say: *(∀CitOfZRH)*, **i.e. cannot quantify over predicates (first order)**

existential quanitification

(∃x) Student(x) → Intelligent (x)

all cats are mammals
all citizens of zurich are gool
cannot express: all citizens of zurich ... (this would be second order)

there exists and intelligent student

First-order predicate calculus

formal system:

- language

- axioms

- inference procedure

semantics

similar to propositional calculus, except that everything will be more complicated.

Grammar for the language of FOPC

Sentence → *AtomicSentence*
| *Sentence Connective Sentence*
| *Quantifier Variable, ... Sentence*
| ¬ *Sentence*
| (*Sentence*)

AtomicSentence → *Predicate*(*Term*, ...)
| *Term* = *Term*

Term → *Function*(*Term*, ...)
| *Constant*
| *Variable*

Connective → ∧ | v | ⇒| ⇔
Quantifier → ∀ | ∃
Constant → *A* | *X1* | *John* | ... | ...
Variable → *a* | *x* | *s* | ...
Predicate → *Before* | *HasColor* | *Raining* " ... |
Function → *Mother* | *LeftLegOf* | ...

in BNF notation (Backus-Naur-Form); see R/N p. 187

Some points concerning syntax: scope of Qs

- **scope of quantifiers (script p. 5-18)**

$(\forall x)\ (human(x) \rightarrow (\exists y)\ (human(y) \wedge loves(x,y)))$

(scope of y .....)

(scope of x .....)

scope can be clarified by using parentheses

Some points concerning syntax: bounded, free vars

- **bounded, free variables (script Def 5.16, p. 5.18)**
- Bounded**: A given occurrence of a variable x in a formula A is** bounded **if it is part of a subformula of A (i.e. a substring of A that is also a formula) of the form** *(∀x) B or (∃y) B.* **Otherwise it is** free**.**
- **A:** *(∀x) (c —> y)*
  x is bounded, y free

Some points concerning syntax: nested Qs

nested quantifiers (p. 192, R/N)

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

$\forall x,y$ *same as* $\forall x\ \forall y$

Everybody loves somebody:

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

There is someone who is loved by everyone:

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

ambiguity: $(\forall x)(Cat(x) \vee (\exists x\ Brother(Richard,x)))$

Rule: variable belongs to innermost Q

It is always a good idea to use parentheses for disambiguation reasons, in this case to designate the scope of the quantifiers.

Some points concerning syntax: relations between Qs

- **relations between quantifiers (p. 193 R/N**
- $(\forall x)\ \neg P \Leftrightarrow (\neg \exists x)\ P$
- etc.

Some points concerning syntax

- **prenex form (p. 14 Fuchs; notes p. 10)**
- **(Q1 x1) (Q2 x2) … (Qn xn) B; B contains no quantifiers, i.e. only free variables**
- **example:**

  $\neg\ (\forall x)(p(x,y)) \wedge q(u) \wedge \neg\ (\forall z)\ (r(z)) \vee s(w))$

  rule: $\neg(\forall x)(p(x)) \leftrightarrow(\exists x)(\neg p(x))$

  rule: $\neg(p \vee q) \leftrightarrow(\neg p \wedge \neg q)$

  $(\exists x)(\neg p(x,y)) \wedge q(u) \wedge \neg\ (\forall z)\ r(z)) \wedge \neg s(w)$

  rule: $\neg(\forall x)(p(x)) \leftrightarrow (\exists x)(\neg p(x))$

Some points concerning syntax

- **(Q1 x1) (Q2 x2) … (Qn xn) B; B contains no quantifiers**
- **example (ctd.):**
  $(\exists x)(\neg p(x,y)) \wedge q(u) \wedge \neg\ (\forall z)\ r(z)) \wedge \neg s(w)$
  rule: $\neg(\forall x)(p(x)) \leftrightarrow (\exists x)(\neg p(x))$

  $(\exists x)(\neg p(x,y)) \wedge q(u) \wedge (\exists z)\neg\ r(z)) \wedge \neg s(w)$
  rule: $p(x)\ (\exists z)\ q(z) \leftrightarrow (\exists z)p(x)\ q(z)$ *(z does not occur in p)*

  $(\exists x)\ (\exists z)(\neg p(x,y) \wedge q(u) \wedge \neg\ r(z)) \wedge \neg s(w))$
- **which is in prenex normal form**

Theorem: Prenex normal form

Theorem: **for every WFF there is (at least) one logically equivalent prenex form.**

p. 265 R/N

Inference in FOPC

p. 265 R/N

Motivation: Conversion to normal form

proofs only for normal form required

resolution theorem proving: Horn Clause Normal Form

- Horn Clauses: **subset of CNF/implicative normal form: p ∨ ¬n1 ∨ ¬n2 ∨ … ∨ ¬nm; written as implication: (n1 ∧ ... ∧ nm) → p; or in Prolog Notation:**
  p :- n1, … nm. (m>0)
  p. (m=0)

Complete inference procedure for FOPC for Horn Clause KB (Robinson)

—> PROLOG Programming Language

Universal generalization

if A can be proved and x is free in A, the following wff can also be proved:

$(\forall x)A$

Rationale: **This is because, from a semantic point of view, for free variables you would have to check all possible interpretations anyway.**

Universal elimination

sentence A; variable x; g: ground term (constant or function applied to ground terms - contains no variables)

(∀x)A

— — —-

subst({x/g},A)

e.g. (∀x) likes(x,IceCream)

— — —-

subst({x/Ben},A)

infer: likes(Ben,IceCream)

Intuition: what holds for everyone also holds for a particular individual

Existential elimination (and Skolem function) (1)

p. 281, R/N

(can be used for converting wffs to Horn Normal Form —> used in Logic Programming; complete inference procedure based on resolution)

Existential elimination

For any sentence A (e.g. *P(x)*) , variable x, and constant symbol k that does not appear elsewhere in KB:

(∃x) A (e.g. *P(x)*)

————-

subst({x/k}, *A*) (e.g. *P(k)*)

Complication with ∃: “Everyone has a heart”

Existential elimination (and Skolem function) (2)

p. 281, R/N

(can be used for converting wffs to Horn Normal Form —> used in Logic Programming; complete inference procedure based on resolution)

Generalization of "existential elimination"

translate $(\exists x)\ P(x)$ into $P(k)$ where k is a constant that does not appear elsewhere in KB.

Existential elimination (and Skolem function) (3)

p. 281, R/N

Complication with ∃: “Everyone has a heart”

$(\forall x)\ Person\ (x) \rightarrow (\exists y)\ Heart\ (y) \wedge Has(x,y)$

replacing simply y with a constant H:

$(\forall x)\ Person\ (x) \rightarrow Heart\ (H) \wedge Has(x,H)$ which says that everyone has the same heart H.

Existential elimination (and Skolem function) (4)

p. 281, R/N

We need to say that the heart they have is not necessarily shared: define function that maps from person to heart:

$(\forall x)\ Person\ (x) \rightarrow Heart\ (F(x)) \wedge Has(x,(F(x))$ where F is a function name that does not appear elsewhere in the KB.

F: **Skolem function**

Existential introduction

formula A, variable x (that does not occur in A), g ground term

A

A / (∃x) Subst({g/x}, A)

e.g. Likes (Jerry, IceCream) ⇒ (∃x) (Likes (x, IceCream))

Resolution

recall:

$a \vee b, \neg b \vee c$ $\quad\leftrightarrow\quad$ $\neg a \rightarrow b, b \rightarrow c$

$a \vee c$ $\qquad\qquad$ $\neg a \rightarrow c$

transitivity of implication

Summary: Conversion to implicative or conjunctive normal

- **eliminate implications**
- **move ¬ inwards**
- **standardize variables (change names of variables to make them unique)**
- **move quantifiers left**
- **Skolemize: remove existential quantifiers by elimination: careful - use Skolem function (remember: avoid “everyone has the same heart”**

R/N p. 281

Summary: Conversion to normal form (2)

- distribute ∧ over ∨ (a∧b)∨c becomes: (a∨c)∧(b∨c)
- flatten nested conjunctions and disjunctions (eliminate parentheses)
- convert disjunctions to implications

R/N p. 282

Horn Clause Logic

- **automated inference procedures: programming language Prolog**
- Horn Clauses: **subset of CNF/implicative normal form:**

p ∨ ¬n1 ∨ ¬n2 ∨ … ∨ ¬nm; written as implication:

(n1 ∧ ... ∧ nm) → p; or in Prolog Notation:

p :- n1, … nm. (m>0)

p. (m=0)

ni, p atomic sentences (Predicate(Term, …) or Term=Term)

The Prolog programming language

Prolog uses unification (finding proper substitutions)

Examples of Prolog programs; see later

Many books

Example: Colonel West

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

To prove: Colonel West is a criminal

Example: Colonel West

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

“the country Nono”

“Nono, and enemy of America”

(*ForAll X*) (*ForAll Y*) (*Forall Z*) (american(*X*) *AND* weapon(*Y*) *AND* nation(*Z*)
AND hostile(*Z*) *AND* sells(*X,Z, Y*) --> criminal(*X*))
nation(Nono)
enemy(Nono,America)

Semantics of predicate calculus (formal definition)

When is a sentence in FOPC true?
—> if it represents a true statement about a domain of discourse

Formally: **Sentences get their truth value through an interpretation I and a variable assignment V in a domain of discourse D**

Interpretation I and truth value of sentence

- constant **c: I(c) ∈ D (domain of discourse)**
- predicate **p: I(p) ∈ P (set of predicates)**
- function symbol f, arity n**: I(f): D^n —> D**
- relation r with arity n**: I(r) ⊆ D^n**
- variable assignment V**, variable x: V(x) ∈ D**
- truth value**: Given D, to each sentence A under Interpretation I and variable assignment V: a** truth value **T,F can be assigned.**

FOPC: more definitions for semantics

- **D domain of discourse; I interpretation**
- **Sentence A is called** true **under interpretation I (and I is called a model of A) if A is true for all variable assignments V, V(x) ∈ D.**
- **Set of expressions: KB**
- satisfiable**: if there is at least one interpretation I for which A is true for all variable assigments V**
- refutable: **if there is at least one interpretation I for which A is not true for all variable assignments V**
- tautological**: if every interpretation I is true for all variable assignments V**

Example of interpretations

- **given: set of expressions (KB)**
  {p(a), (∀x) (p(x) → q(x))}

- interpretation 1:
  I1(a) = Socrates
  I1(p) = human: {Socrates, Plato, John_Brown, ...}
  I1(q) = mortal: {Socrates, Plato, John_Brown, ...}

- interpretation 2:
  I2(a) = 8
  I2(p) = divisible_by_4: {0, 4, 8, ...}
  I2(q) = even: {0, 2, 4, ...}

First-order predicate calculus

- sound
- complete
- decidable?

First-order predicate calculus

- sound: **KB |— A => BK |= A**
- complete: **BK |= A => KB |— A (an inference procedure is complete if it can find a proof for any sentence that is entailed, i.e. is a semantic consequence of KB)**
- decidable**?**

  —> *semi-decidable:*
  can show that sentences follow from premises if they do, but we cannot always show this if they do not.
- **more complex system, e.g. arithmetic: incomplete (Gödel), i.e. there are true arithmetic sentences that cannot be proved**

Here is the text from R/N: “There is one problem with the completeness theorem that is a real nuisance. Note that we said that *if* a setence follows, then it can be proved. Normally, we do not know until the proof is done that the sentence *does* follow; what happens when the sentence doesn’t follow? Can we tell? Well, for first-order logic, it turns out that we cannot; our proof procedure can go on and on, but we will not know if it is stuck in a hopeless loop or if the proof is just about to pop out. (This is like the halting problem for Turing machines). Entailment in first-order logic is thus semidecidable, that is, we can show that sentences follow form premises if they do, but we cannot always show it if they do not.

Back to Horn Clause Logic

- **automated inference procedures: programming language Prolog**
- Horn Clauses: **subset of CNF/implicative normal form: p ∨ ¬n1 ∨ ¬n2 ∨ … ∨ ¬nm; written as implication: (n1 ∧ ... ∧ nm) → p; or in Prolog Notation:**
  p :- n1, … nm. (m>0)
  p. (m=0)
  ni, p atomic sentences
- **interesting compromise between expressivity and efficiency**

The Prolog programming language

(a few examples of Prolog programs; see notes)

Prolog uses unification/matching (finding proper substitutions)

Prolog implements a subset of FOPC, Horn-Clause Logic, and it uses only one inference rule, resolution (see earlier slides). As a convention, Prolog uses uppercase letters for variables and lowercase for constants. It also reverses the order of implications: Q:-P, instead of P—>Q. Commas indicate conjunctions: cat(X):-furry(X),meows(X),has(X,claws).
Prolog does not implement pure Horn-Clauses (which are a subset of FOPC), but to turn it into a practically useful programming language, it contains extensions, for example, for arithmetic.
Inference procedure: backward chaining with depth-first search
Order: left to right and clauses in the database are searched top to bottom (adapted from Russell/Norvig).
Efficiency has a price: Logic programming is incomplete, i.e. it may happen that Prolog fails to find existing solutions.

The Prolog programming language: family data base

```
male(paul).
male(sidney).
male(sam).
male(robert).

female(beverly).
female(fay).
female(mary).
female(lena).

father(sam,paul).
father(sidney,beverly).
father(sam,fay).
father(paul,mary).

mother(beverly,mary).
mother(lena,paul).
```

male: predicate name
paul: argument
constants:
individuals(objects) (e.g. paul)
predicates:
- male - unary
- father - binary

? male(robert)
—> yes
? female(anna)
—> no (not found in data base)
? mother(lena,paul)
—> yes
? father(X,mary)
—> X=paul
(who is the father of mary?)

add two rules:

```
parent(X,Y):-father(X,Y)
parent(X,Y):-mother(X,Y)
```

? parent(P,mary)
—> P=paul; P=beverly

define brother (X,Y):-

```
  male(X),
  parent(P,X),
  parent(P,Y),
  X\==Y (X,Y different).
```

? brother(X,Y)
—> X=paul; Y=fay

a parent is a father or a mother (different rules interpreted as “or”)

X is the brother of Y if X is male AND P is parent of X AND P is parent of Y, AND X and Y are different. Commas “,” indicate AND.

The Prolog programming language: family data base

```
male(paul).
male(sidney).
male(sam).
male(robert).

female(beverly).
female(fay).
female(mary).
female(lena).

father(sam,paul).
father(sidney,beverly).
father(sam,fay).
father(paul,mary).

mother(beverly,mary).
mother(lena,paul).
```

ancestor(X,Y):- ?

Prolog does not only give you answers but replies with all the formulas that it can prove —> extremely powerful

Inference rule in Prolog: resolution

The Prolog programming language: family data base

```
male(paul).
male(sidney).
male(sam).
male(robert).

female(beverly).
female(fay).
female(mary).
female(lena).

father(sam,paul).
father(sidney,beverly).
father(sam,fay).
father(paul,mary).

mother(beverly,mary).
mother(lena,paul).
```

recursive rules:

```
ancestor(X,Y):-
   parent(X,Y).
ancestor(X,Y):-
   parent(X,Z),
   ancestor(Z,Y).
```

What are the ancestors of mary?

```
? ancestor(X,mary)
—> ancestor=paul
     ancestor=beverly
     etc.
```

Prolog does not only give you answers but replies with all the formulas that it can prove —> extremely powerful

Inference rule in Prolog: resolution

The Prolog programming language: Unification

“John hates everyone he knows”

knows(john,X) —> hates(john,X)

KB:

knows(john,jane).
knows(Y,leonid).
knows(Y,mother(Y)).
knows(X,elizabeth).

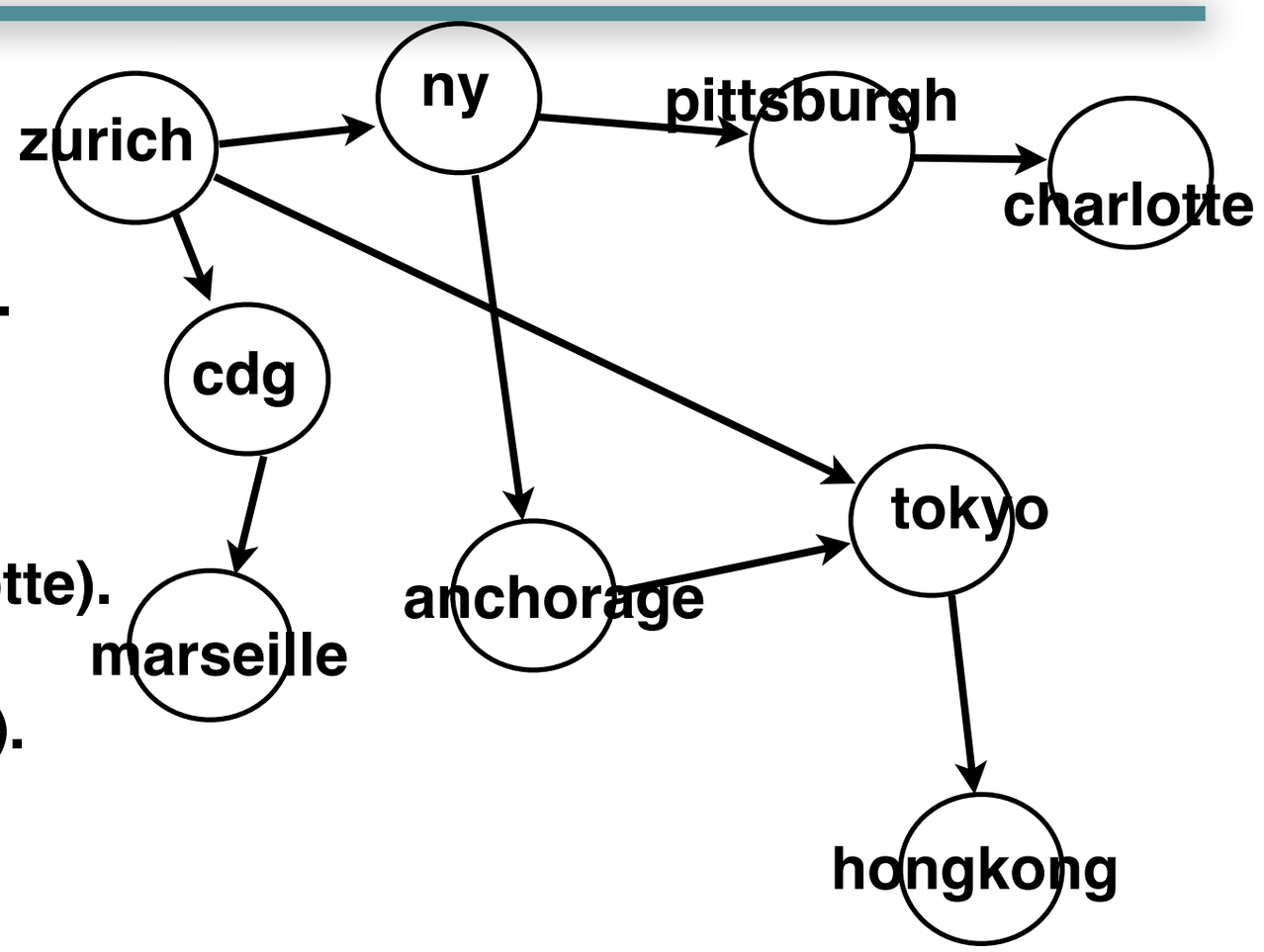
(X,Y: implicitely universally quantified)

UNIFY(knows(john,X),knows(john,jane)) = {X/jane}
UNIFY(knows(john,X),knows(Y,Leonid)) = {X/leonid,Y/jane}

etc.

The Prolog programming language (5): The flight network

```
direct_flight(zurich,ny).
direct_flight(zurich,cdg).
direct_flight(zurich,tokyo).
direct_flight(tokyo,hongkong).
direct_flight(zurich,fukuoka).
direct_flight(ny,anchorage).
direct_flight(ny,pittsburgh).
direct_flight(pittsburgh,charlotte).
direct_flight(cdg,marseille).
direct_flight(anchorage,tokyo).
```



flight from X to Y?

ny: New York
cdg: Paris Charles de Gaulle

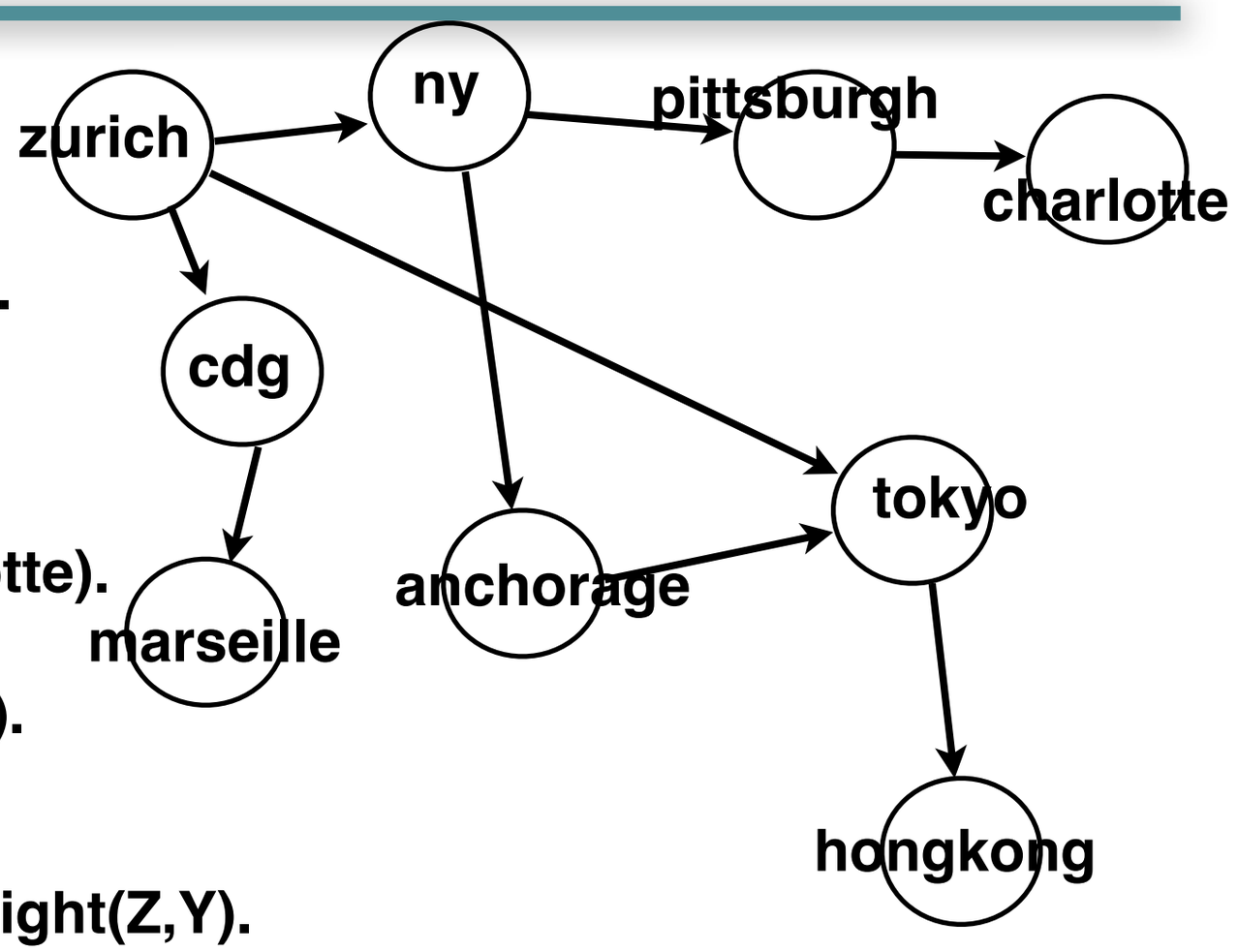
flight(X,Y):-direct_flight(X,Y).
flight(X,Y):-flight(X,Z),direct_flight(Z,Y).

? flight(zurich,charlotte). Given the axioms and the formulas, prove the formula flight(zurich,charlotte).

or: What are the destinations to which you can fly from Zurich?
? flight(zurich,X).
X=ny, X=cdg, X=tokyo, etc.

The Prolog programming language (5): The flight network

```
direct_flight(zurich,ny).
direct_flight(zurich,cdg).
direct_flight(zurich,tokyo).
direct_flight(tokyo,hongkong).
direct_flight(zurich,fukuoka).
direct_flight(ny,anchorage).
direct_flight(ny,pittsburgh).
direct_flight(pittsburgh,charlotte).
direct_flight(cdg,marseille).
direct_flight(anchorage,tokyo).
```

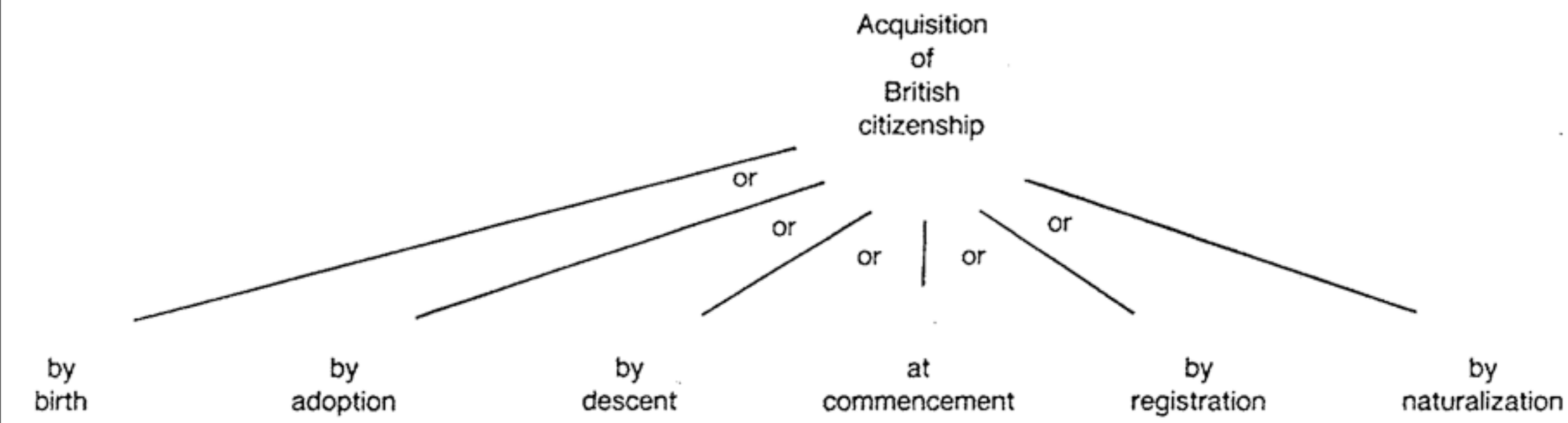



```
flight(X,Y):-direct_flight(X,Y).
flight(X,Y):-flight(X,Z),direct_flight(Z,Y).
```

? flight(zurich,charlotte). Given the axioms and the formulas, prove the formula flight(zurich,charlotte).

or: What are the destinations to which you can fly from Zurich?
? flight(zurich,X).
X=ny, X=cdg, X=tokyo, etc.

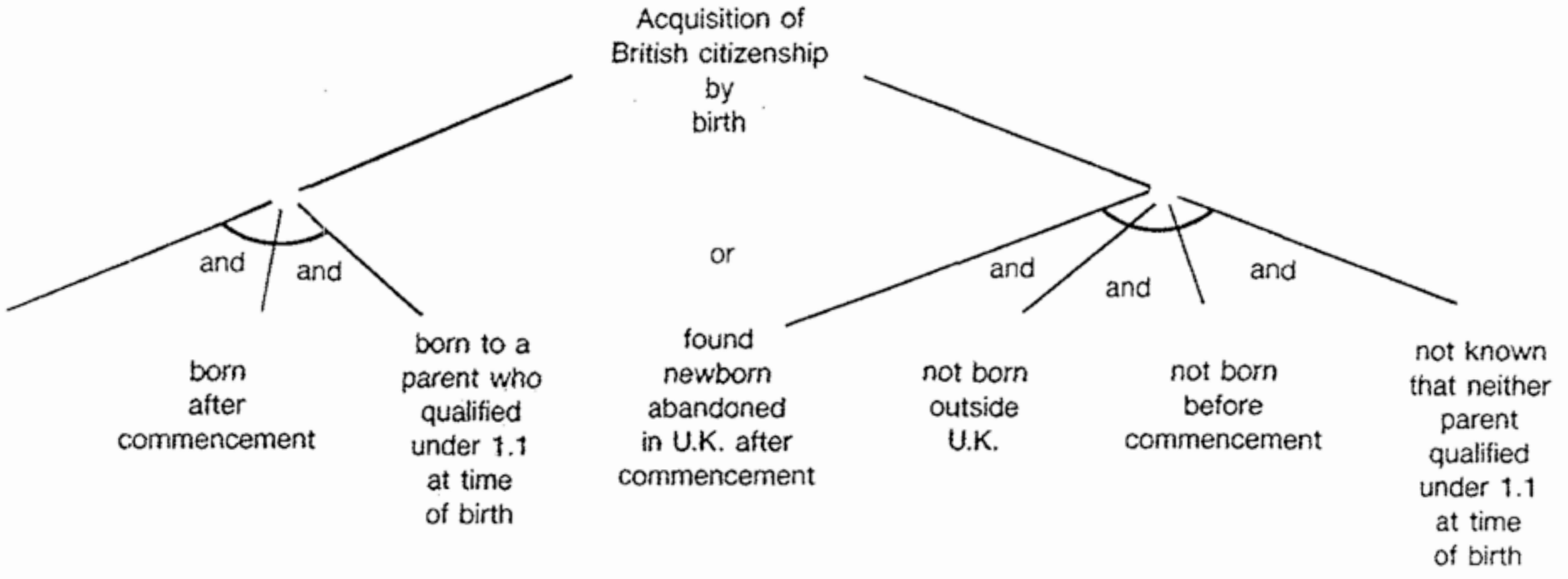
Famous Prolog program: “The British Nationality Act”



hown is an and/or graph representation of the top level of cquisition of British citizenship in Part 1 of the act. Within this part of the act, it is possible to acquire British citizenship by six different routes.

FIGURE 2. Six Routes to Acquisition of British Citizenship

“The British Nationality Act” backward reasoning



An and/or graph representation of the top level of acquisition of British citizenship by birth. This section of the act is divided into two cases, depending on whether or not the individual was found abandoned as a newborn infant.

FIGURE 3. Acquisition of British Citizenship by Birth

Illustrates the idea of backward reasoning (or backward chaining). If you want to verify British citizenship by birth, you can either use the lefthand side of the tree, or the righthand side. If you use the left-hand side, you need to check if the person was born in the UK AND was born after commencement, AND was born to a parent who qualified under 1.1. at the time of birth. etc.

“The British Nationality Act” example rules

x is a British citizen
if x was born in the UK
and x was born on date y
and y is after or on commencement
and z is a parent of x
and z is a British citizen on date y

facts: Peter was born in the UK; William is father of Peter

x has a parent who qualifies under 1.1 on date y
if z is a parent of x
and z is a British citizen on date y

x is a British citizen
if x was born in the UK
and x was born on date y
and y is after commencement
and x has a parent who qualifies under 1.1 on date y

Works well by way of its construction, as a logic-base system.

Limitations of FOPC?

Limitations of FOPC?

other kinds of logics:

- higher-order
- temporal
- modal
- FUZZY

- quantification over predicates
- time?
- world does not consist of true or false statements, e.g. I believe, this should be the case, I want an apple, etc. --> modal logics
- typically things only hold to a certain extent --> Fuzzy Logic

Stay tuned for FUZZY Logic